Conferences in Research and Practice in
Information Technology

Volume 63

Database Technologies 2007

Australian Computer Science Communications, Volume 29, Number 2.

Database Technologies 2007

Proceedings of the
Eighteenth Australasian Database Conference (ADC 2007),
Ballarat, Victoria, Australia,
January 30 to February 2, 2007

James Bailey and Alan Fekete, Eds.

Volume 63 in the Conferences in Research and Practice in Information Technology Series.
Published by the Australian Computer Society Inc.

Published in association with the ACM Digital Library.

Proceedings of the Eighteenth Australasian Database Conference (ADC 2007), Ballarat, Victoria, January 30 to February 2, 2007

Conferences in Research and Practice in Information Technology, Volume 63.

Editors:
James Bailey
Department of Computer Science and Software Engineering
The University of Melbourne
Victoria 3010,
Australia
Email: `jbailey@csse.unimelb.edu.au`

Alan Fekete
School of Information Technologies
University of Sydney
NSW 2006,
Australia
Email: `fekete@it.usyd.edu.au`

Series Editors:
Vladimir Estivill-Castro, Griffith University, Queensland
John F. Roddick, Flinders University, South Australia
Simeon Simoff, University of Technology, Sydney, NSW
`crpit@infoeng.flinders.edu.au`

Publisher: Australian Computer Society Inc.
PO Box Q534, QVB Post Office
Sydney 1230
New South Wales
Australia.

Conferences in Research and Practice in Information Technology, Volume 63.
ISSN 1445-1336.
ISBN 1-920-68244-9.

Printed, November 2006 by Flinders Press, PO Box 2100, Bedford Park, SA 5042, South Australia.
Cover Design by Modern Planet Design, (08) 8340 1361.

The *Conferences in Research and Practice in Information Technology* series aims to disseminate the results of peer-reviewed research in all areas of Information Technology. Further details can be found at http://crpit.com/.

Table of Contents

Preface

The Australasian Database Conference (ADC) series is an annual forum, exploring research, development and novel applications of databases systems. This volume contains papers presented at the Eighteenth ADC in Ballarat, Victoria, Australia. ADC 2007 is a specialist conference in the Australasian Computer Science Week, which ran from January 29 to February 2, 2007.

Database systems is a rich and broad research area and the call for papers reflected this diversity. Submissions were requested in the areas of advanced database applications, advanced foundations of databases, databases for bioinformatics, data mining/knowledge discovery, data warehousing, database system integration issues, database schema integration, embedded and mobile databases, federated, distributed, parallel and grid databases, extended data type management, high dimensional and temporal data, image/video retrieval and databases, information retrieval, filtering and dissemination, logic in databases, performance issues of databases, privacy and secure databases, databases query languages, query processing and optimization, semi-structured data, spatial data processing/management, stream data management, transaction processing, Web access to databases, Web information systems, XML and databases.

In response to the call for papers, we received 35 full papers. The range of submissions was truly international, with papers submitted from Australia (14), New Zealand (2), Canada (1), Japan (3), France (1), United Kingdom (3), China(2), Germany(1), Finland(1), South Korea(2), Kuwait(1), Taiwan(1), Italy(2), Vietnam(1).

The international Program Committee contained experts from across all aspects of database research and had wide representation across Australia and New Zealand, as well as integrating some highly regarded researchers from Asia and the United States.

All papers were sent to four programme committee members for review and nearly all papers received four reviews. Every paper received at least three reviews. Of the 35 papers submitted, 16 were selected for presentation at the conference. Professor Krithi Ramamritham was invited to give a keynote on *Taming the Dynamics of Distributed Data*. Professor Ramamritham is the Vijay and Sita Vashee Chair at the Indian Institute of Technology, Bombay. The committee also invited Associate Professor Xuemin Lin to give a talk on *Continuously Maintaining Order Statistics over Data Streams*.

We thank all authors who submitted papers and all conference participants for helping to make the conference a success. We also thank the members of the programme committee and the external referees for their expertise in carefully reviewing the papers. Lastly, we express our gratitude to our hosts in Ballarat.

James Bailey
University of Melbourne
Alan Fekete
University of Sydney

ADC 2007 Programme Chairs
January 2007

Programme Committee

Chairs

James Bailey, University of Melbourne
Alan Fekete, University of Sydney

Members

Dave Abel, CSIRO
Stijn Dekeyser, University of Southern Queensland
Gill Dobbie, University of Auckland
David Edmond, QUT
Raj P Gopalan, Curtin University of Technology
Guido Governatori, University of Queensland
Hongfei Guo, Microsoft Research
Sven Hartmann, Massey University
David Hawking, CSIRO Canberra
Annika Hinze, University of Waikato
Patrick Hung, University Of Ontario Institute of Technology
Xue Li, University of Queensland
Xuemin Lin, University of New South Wales
Sebastian Link, Massey University
Beng Chin Ooi, National University of Singapore
John Roddick, Flinders University
Uwe Roehm, University of Sydney
John Shepherd, University of New South Wales
Markus Stumptner, University of South Australia
S. Sudarshan IIT Bombay
Saied Tahaghoghi, RMIT University
Kian-Lee Tan, National University of Singapore
Egemen Tanin, University of Melbourne
Rodney Topor, Griffith University
Andrew Turpin, RMIT University
Wei Wang, University of New South Wales
Gerald Weber, University of Auckland
Raymond Wong, University of New South Wales
Jeffrey Yu, Chinese University of Hong Kong
Yanchun Zhang, Victoria University
Xiaofang Zhou, University of Queensland

Additional Reviewers

Eunus Ali
Rukshan Athauda
Ding-Yi Chen
Muhammad Cheema
Yueguo Chen
Roozbeh Derakhshan
Mohamed Gaber
Yanan Hao
Dayang Iskandar
Qing Liu
Yi Luo
Jiangang Ma
Sarana Nutanong
Justin O'Sullivan
Simon Puglisi
Xingzhi Sun
Falk Scholer
Yanfeng Shu
Ranjan Sinha
Muhammed Umer
Florian Verhein
Timo Volkmer
Moe Wynn
Bei Yu
Wenjie Zhang
Rui Zhang
Emily Zhou

Organising Committee

Welcome

I would like to welcome you to the University of Ballarat and ACSW 2007.

Ballarat is one of Australia's largest inland cities with a population of 83,000, and is nestled peacefully in the heart of Victoria just over an hour from Melbourne. Ballarat is regarded as the birthplace of democracy in Australia and has one of the finest tourist attractions, namely Sovereign Hill.

The University of Ballarat is the third oldest tertiary institute in Australia. It is a medium-size University with about 22,000 students. The School of Information Technology and Mathematical Sciences of the University of Ballarat has 80 academic and general staff and includes the research Centre for Informatics and Applied Optimization (CIAO) and the Collaborative Centre for eHealth (CCeH).

ACSW 2007 includes the following conferences:

– Australasian Computer Science Conference (ACSC),
– Australasian Database Conference (ADC),
– Australasian Computer Education Conference (ACE),
– Computing: The Australian Theory Symposium (CATS),
– Asia-Pacific Conference of Conceptual Modelling (APCCM),
– Australasian User Interface Conference (AUIC),
– Australasian Symposium on Grid Computing and Research (AUSGRID),
– Australasian Workshop on Health Knowledge Management and Discovery (HKMD),
– Australasian Information Security Workshop:Privacy Enhancing Systems (AISW), and the
– Australasian Computing Doctoral Consortium (ACDC).

I thank all those who have worked to ensure the success of ACSW2007 including the Organizing Committee, the Conference Chairs and Programme Committees, the invited speakers and the delegates.

Professor Sid Morris
Head, School of Information Technology and Mathematical Sciences
University of Ballarat
January, 2007

General Chair

Professor Sid Morris, School of Information Technology and Mathematical Sciences, University of Ballarat

Organising Committee Members

Ms Nadine Gass
Mr Sasha Ivkovic
Ms Kathleen Keogh
Dr Liping Ma
Dr Prabhu Manyem
Mr Greg Simmons
Ms Rosemary Torney
Dr Chris Turville
Ms Belinda Wallesz
Dr David Yost

CORE - Computing Research and Education

CORE welcomes all delegates to ACSW2007 in Ballarat.

ACSW, the Australasian Computer Science Week continues to grow with new conferences becoming entrenched in the week. As the premier annual Computer Science event in Australia and New Zealand, it provides an unparalleled opportunity for the wide community of Computer Science academics and researchers to meet, network, promote IT research and be exposed to the latest research in other areas of IT. The research presented at each conference is of the highest standard and essential for the growth and future of our region, in an ever more competitive world.

2006 has been a difficult year for IT and especially CORE's members. Falling student numbers have meant cutbacks in many of our universities. However, despite offshoring, industry is now calling for more graduates. We'll continue to work with ACS and industry bodies to try to convey this message to school leavers and their parents. We also have to continue to impress on industry the need to provide entry level positions so that young people can work towards the more senior positions which are so understaffed.

RQF is still hovering over Australian universities. In preparation, CORE has been part of a major exercise this year to rank ICT conferences and we'll contribute to the work of our sister organisation, ACPHIS in a similar journal ranking exercise.

Thank you all for your contributions in 2006 and we look forward to an exciting 2007.

COmputing
Research
& Education

Jenny Edwards
President, Computing Research and Education
January, 2007

ACSW Conferences and the Australian Computer Science Communications

The Australasian Computer Science Week of conferences has been running in some form continuously since 1978. This makes it one of the longest running conferences in computer science. The proceedings of the week have been published as the *Australian Computer Science Communications* since 1979 (with the 1978 proceedings often referred to as *Volume 0*). Thus the sequence number of the Australasian Computer Science Conference is always one greater than the volume of the Communications. Below is a list of the conferences, their locations and hosts.

2009 (Proposed). Communications Volume Number 31. Host and Venue - Victoria University, Wellington, New Zealand.
2008. Volume 30. Host and Venue - University of Wollongong, NSW.

2007. **Volume 29. Host and Venue - University of Ballarat, VIC.**

2006. Volume 28. Host and Venue - University of Tasmania, TAS.
2005. Volume 27. Host - University of Newcastle, NSW. APBC held separately from 2005.
2004. Volume 26. Host and Venue - University of Otago, Dunedin, New Zealand. First running of APCCM.
2003. Volume 25. Hosts - Flinders University, University of Adelaide and University of South Australia. Venue - Adelaide Convention Centre, Adelaide, SA. First running of APBC. Incorporation of ACE. ACSAC held separately from 2003.
2002. Volume 24. Host and Venue - Monash University, Melbourne, VIC.
2001. Volume 23. Hosts - Bond University and Griffith University (Gold Coast). Venue - Gold Coast, QLD.
2000. Volume 22. Hosts - Australian National University and University of Canberra. Venue - ANU, Canberra, ACT. First running of AUIC.
1999. Volume 21. Host and Venue - University of Auckland, New Zealand.
1998. Volume 20. Hosts - University of Western Australia, Murdoch University, Edith Cowan University and Curtin University. Venue - Perth, WA.
1997. Volume 19. Hosts - Macquarie University and University of Technology, Sydney. Venue - Sydney, NSW. ADC held with DASFAA (rather than ACSW) in 1997.
1996. Volume 18. Host - University of Melbourne and RMIT University. Venue - Melbourne, Australia. CATS joins ACSW.
1995. Volume 17. Hosts - Flinders University, University of Adelaide and University of South Australia. Venue - Glenelg, SA.
1994. Volume 16. Host and Venue - University of Canterbury, Christchurch, New Zealand. CATS run for the first time separately in Sydney.
1993. Volume 15. Hosts - Griffith University and Queensland University of Technology. Venue - Nathan, QLD.
1992. Volume 14. Host and Venue - University of Tasmania, TAS. (ADC held separately at La Trobe University).
1991. Volume 13. Host and Venue - University of New South Wales, NSW.
1990. Volume 12. Host and Venue - Monash University, Melbourne, VIC. Joined by Database and Information Systems Conference which in 1992 became ADC (which stayed with ACSW) and ACIS (which now operates independently).
1989. Volume 11. Host and Venue - University of Wollongong, NSW.
1988. Volume 10. Host and Venue - University of Queensland, QLD.
1987. Volume 9. Host and Venue - Deakin University, VIC.
1986. Volume 8. Host and Venue - Australian National University, Canberra, ACT.
1985. Volume 7. Hosts - University of Melbourne and Monash University. Venue - Melbourne, VIC.
1984. Volume 6. Host and Venue - University of Adelaide, SA.
1983. Volume 5. Host and Venue - University of Sydney, NSW.
1982. Volume 4. Host and Venue - University of Western Australia, WA.
1981. Volume 3. Host and Venue - University of Queensland, QLD.
1980. Volume 2. Host and Venue - Australian National University, Canberra, ACT.
1979. Volume 1. Host and Venue - University of Tasmania, TAS.
1978. Volume 0. Host and Venue - University of New South Wales, NSW.

Conference Acronyms

ACE. Australian/Australasian Conference on Computing Education.
ACSAC. Asia-Pacific Computer Systems Architecture Conference (previously Australian Computer Architecture Conference (ACAC).
ACSC. Australian/Australasian Computer Science Conference.
ACSW. Australian/Australasian Computer Science Week.
ADC. Australian/Australasian Database Conference.
AISW. Australasian Information Security Workshop.
APBC. Asia-Pacific Bioinformatics Conference.
APCCM. Asia-Pacific Conference on Conceptual Modelling.
AUIC. Australian/Australasian User Interface Conference.
AusGrid. Australasian Workshop on Grid Computing and e-Research.
CATS. Computing - The Australian/Australasian Theory Symposium.

Note that various name changes have occurred, most notably the change of the names of conferences to reflect a wider geographical area.

ACSW and ADC 2007 Sponsors

We wish to thank the following sponsors for their contribution towards this conference. For an up-to-date overview of sponsors of ACSW 2007 and ADC 2007, please see `http://www.ballarat.edu.au/acsw/`.

University of Ballarat, Australia

Australian Computer Society

CORE - Computing Research and Education

Department of Computer Science and Software Engineering

School of Information Technologies

Keynote

Taming the Dynamics of Distributed Data

Krithi Ramamritham
Department of Computer Science and Engineering
Indian Institute of Technology, Bombay
krithi@iitb.ac.in

The Internet and the web are increasingly used to disseminate fast changing data such as sensor data, weather information, stock prices, sports scores, and even health monitoring information. These data items are highly dynamic, i.e., the data changes continuously and rapidly, streamed in real-time, i.e., new data can be viewed as being appended to the old or historical data, and aperiodic, i.e., the time between the updates and the value of the updates are not known a priori. Increasingly, more and more users are interested in monitoring such data for on-line decision making. Traditional dissemination methods involve a pull or a push of data between a source of the data and a client. However, resource limitations at the source limits the number of users that can be served directly by it.

A natural solution to this is to have a set of repositories which replicate the source data and serve it to geographically closer users. Services like Akamai and IBM's edge server technology are exemplars of such networks of repositories, which aim to provide better services by shifting most of the work to the edge of the network (closer to the end users). Although such systems scale quite well, when the data changes rapidly, the quality of service at a repository farther from the data source will deteriorate. In general, replication can reduce the load on the sources, but replication of time-varying data introduces new challenges. Unless updates to the data are carefully disseminated from sources to repositories (to keep them coherent with the sources), the communication and computation overheads involved can result in delays as well as scalability problems, further contributing to loss of data coherence.

In situations where the data is to be used for on-line monitoring or online decision making, users specify the bound on the tolerable imprecision associated with each requested data item, this can be viewed as *coherence* requirement associated with the data. The coherence requirements associated with a time-varying data item depend on the nature of the item and user tolerances. For example, a user involved in exploiting exchange disparities in different markets or an on-line stock trader may impose stringent coherence requirements (e.g., the stock price reported should never be out-of sync by more than one cent from the actual value) whereas a casual observer of currency exchange rate fluctuations or stock prices may be content with a less stringent coherence requirement.

What is needed is a dynamic data distribution system that is *coherence-preserving*, i.e., the delivered data must preserve associated coherence requirements, and *resilient*, i.e., the system should be resilient to failures. Needless to say, it should work effectively with the minimal provisioning of resources.

Given such a data dissemination network, users can execute queries over distributed data by obtaining the data required for the query from one or more data repositories in the network. How this mapping – between query needs and data repositories – is done, depends on (a) the coherency of the data available at each repository and the precision requirements associated with the query, (b) the dynamics of the data, etc. Solving this optimization problem turns out to be challenging, especially since the mapping must be revisited as data characteristics change.

In this talk we will present work done at IIT Bombay towards solving these problems and also relate to ongoing work on sensor networks and stream processing systems.

Invited Paper

Continuously Maintaining Order Statistics over Data Streams (Extended Abstract)

Xuemin Lin

School of Computer Science and Engineering
University of New South Wales,
NSW 2052, Australia
Email: lxue@cse.unsw.edu.au

1 Introduction

A rank query is essentially to find a data element with a given rank against a monotonic order specified on data elements. It has several equivalent variations [8, 17, 30]. Rank queries over data streams have been investigated in the form of *quantile computation*. A *ϕ-quantile* ($\phi \in (0, 1]$) of a collection of N data elements is the element with rank $\lceil \phi N \rceil$ against a monotonic order specified on data elements. Rank and quantile queries have many applications [1, 3, 6, 7, 10, 14–16, 26, 27], including monitoring high speed networks, trends and fleeting opportunities detection in the stock market, sensor data analysis, Web ranking aggregation and log mining, etc. In these applications, they not only play very important roles in the decision making but also have been used in summarizing data distributions of data streams. The following example shows a popular tool to compare the distributions of two data sets (data streams).

Example 1 *An information provider may provide various real-time statistics of the stock market to its clients, through the Internet or telecommunication, for trends' analysis. One of the most popular charts is the* quantile-quantile *(Q-Q) plot [28] for comparing two data distributions. In a stock market, price and volume distributions are two key indexes to monitor. Figure 1 illustrates such a Q-Q plot by using two real datasets AOL and Technique_section. In AOL,* 1.3*M (millions) "tick-tick" transactions during the period Dec/2000 - July/2001 sorted increasingly against the volume of each transaction (deal) are collected from NYSE (New York Stock Exchange) for the stock AOL. In Technique_section,* 27*M tick-tick transactions are collected in the same period for the stocks csco, ibm, dell, sun, ca and also sorted on volumes. The figure demonstrates that clients can view the global chart (with very coarse information due to physical limits of display) for a general comparison, and can also click on such a chart graph to zoom in a particular range of quantiles interactively for more accurate information. Such Q-Q plots combining with other statistic display tools greatly facilitate clients detection of trade trends and thus make good trade decisions.*

The work is partially supported by an ARC Discovery Grant (DP0666428).

In such an application, clients may require on-line updates (processing) of Q-Q plots to monitor trading trends and detect fleeting opportunities in real time.

It has been shown in [20] that an exact computation of rank queries requires memory size linearly proportional to the size of a dataset by any one-scan technique; this may be impractical in on-line data stream computation where streams are massive in size and fast in arrival speed. Approximately computing rank queries over data streams, thus, has received a great deal of attention recently. The main paradigm is to continuously maintain a small space data structure, called summary or sketch, to summarize *order statistics*. In this talk, we will introduce the space- and time-efficient one-scan techniques of continuously maintaining order statistics for supporting approximate rank (quantile) queries. These include deterministic and randomized approximate techniques. We also discuss open issues in the area.

2 ϵ-approximation

In the problem setting, an element x may be augmented to (x, v) where $v = f(x)$ (called "value") is to rank elements according to a monotonic order of v, and f is a pre-defined function. Without loss of generality, we assume $v > 0$ and a monotonic order is always an increasing order. We study the following rank queries over a data stream S.

Rank Query (RQ) : Given a rank r, find the rank r element in S.

Suppose that r is the given rank in a RQ query, and r' is the rank of an approximate solution. We could use the constant-based absolute error metric; that is, enforce $|r' - r| \le \epsilon$ for a given ϵ. It is immediate that such an absolute error precision guarantee leads to the space requirement $\Omega(N)$ even for an off-line computation where $N = |S|$. Therefore, two error metrics have been used.

Uniform Error. $\frac{|r'-r|}{N} \le \epsilon$.

Relative Error. $\frac{|r'-r|}{r} \le \epsilon$.

An answer to a RQ regarding r is *uniform ϵ-approximate* if its rank r' has the precision $|r' - r| \le \epsilon N$; it is *relative ϵ-approximate* if its rank r' has the precision $|r' - r| \le \epsilon r$. In this talk, we present the techniques of continuously maintaining a sketch (consisting of several sub-sketches) over a data stream S such that at any time, the sketch can be used to return a (relative or uniform) ϵ-approximate answer to a RQ. The focus is to minimize the *maximum memory space required in such a continuous computation* of sketch/summary.

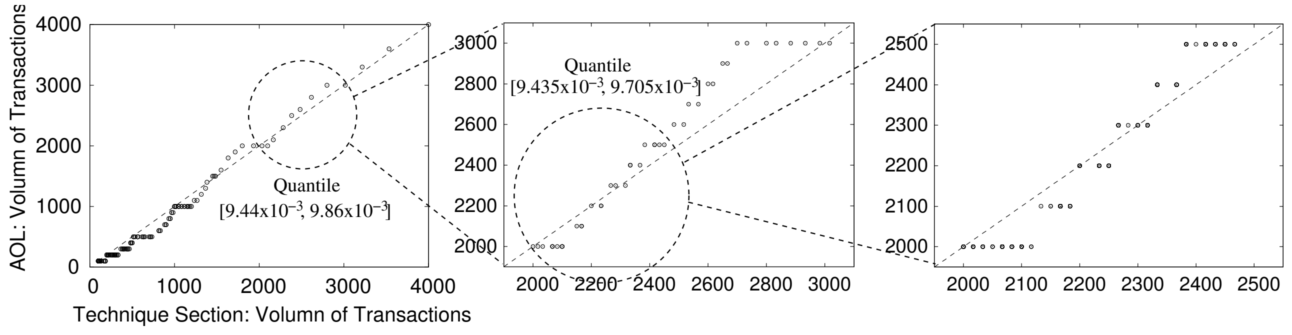


Figure 1: Q-Q Plot

3 Uniform Error Techniques

It has been shown in [15,25] that a space-efficient, uniform ϵ-approximate quantile summary can be maintained so that, for a quantile ϕ, it is always possible to find an element at rank r' with the precision guarantee $|\lceil \phi N \rceil - r'| \leq \epsilon N$. Greenwald and Khanna [15] developed a one-scan technique with $O(\frac{1}{\epsilon}\log(\epsilon N))$ space bound and the deterministic error guarantee $|r - r'| \leq \epsilon N$.

Manku *et al* [24] provided a space efficient randomized algorithm, based on an adaptive sampling technique, to achieve the uniform precision guarantee ϵN with confidence (probability) at least $1 - \delta$ and space $O(\frac{1}{\epsilon}\log^2 \frac{1}{\epsilon\delta})$. It has been shown that if the GK-algorithm [15] is applied, the space bound can be reduced to $O(\frac{1}{\epsilon}\log \frac{1}{\epsilon\delta})$.

Gilbert *et al* [13] proposed space-efficient randomized techniques for processing rank queries (quantile queries) when elements already seen may be updated; the uniform precision guarantee ϵN is also used. Cormode and Muthukrishnan [5] showed that an application of their *count-min sketch* technique can reduce the space bound in [13] from $O(\frac{1}{\epsilon^2}\log^2 |U| \log \frac{\log |U|}{\delta})$ to $O(\frac{1}{\epsilon}\log^2 |U| \log \frac{\log |U|}{\delta})$ where U is the value domain.

In [27], Shrivastava *et al* investigated the problem of minimizing uniform errors, when a space bound is pre-given, with the applications in sensor networks.

While quantile (order statistic) summaries over whole data streams have their applications, such summaries do not have the concept of *aging*, that is, quantiles are computed for *all* N data elements seen so far, including those seen long time ago. There are a broad spectrum of applications where data elements seen early could be outdated and quantile summaries for the most recently seen data elements are more important. For example, the top ranked Web pages among most recently accessed N pages should produce more accurate web page access prediction than the top ranked pages among all pages accessed so far as users' interests are changing. In financial market, investors are often interested in the price quantile of the most recent N bids. Motivated by this, in [21] we developed space- and time- efficient *sliding window* techniques to continuously maintaining order statistics over fixed-length sliding windows and variable sliding window techniques, respectively. Our techniques are based on a combination of GK-algorithm [15] and the *exponential histogram* techniques in [11]; they provide uniform ϵ-approximation. Arasu and Manku [2] replace the exponential histogram based data structure by an interval-tree like data structure to improved the space bound in our paper [21].

4 Relative Error Techniques

In many applications, it is desirable to investigate relative errors (or *biased* errors); that is, enforce error precision ϵr (relative ϵ-approximation) instead of ϵN. As pointed out in [7], finer error guarantees at higher ranks are often desired in network management.[1] This is because IP traffic data often exhibits skew towards the tail and it is exactly in the most skewed region where one wants finer rank error guarantees, to get more precise information about changes in values.

The problem of finding approximate quantiles with relative error guarantees was first studied by Gupta and Zane [17], who developed a one-scan randomized technique with $O(\frac{1}{\epsilon^3}\log^2 N)$ space requirement for approximately counting *inversions*, by maintaining an order sketch with the relative rank error guarantee ϵ. However, the technique requires advance knowledge of (an upper bound on) N to do one-scan sampling. This potentially limits its applications. Cormode *et al.* [7] studied the related problem - computing *biased quantiles*, that is, the set of quantiles $\Phi = \{\phi_i = \phi_0^i : 1 \leq i \leq k\}$, for a fixed k and some ϕ_0, which are estimated with precision $\epsilon\phi_i N$. [7] gives an algorithm to approximate such biased quantiles with deterministic error guarantees which performs very well against many real data sets. As shown in [30], to enforce ϵ-approximation of rank queries it requires a linear space $\Omega(N)$ in the worst case.

In [30], we developed a novel, space- and time- efficient multi-layer sampling technique. It guarantees relative ϵ-approximation with high confidence $1 - \delta$ ($\delta > 0$) and requires space $O(\frac{1}{\epsilon^2}\log\frac{1}{\delta}\log\epsilon^2 N)$ in the worst case and $O(\frac{1}{\epsilon}\log(\frac{1}{\epsilon}\log\frac{1}{\delta})\frac{\log^{2+\alpha}\epsilon N}{1-1/2^\alpha})$ (for $\alpha > 0$) on average.

Restricted to a fixed value domain, Cormode *et al* [8] recently developed a novel deterministic algorithm, by significantly extending the technique in [27], to ensure relative ϵ-approximation with space bound $O(\frac{\log |U|}{\epsilon}\log \epsilon N)$.

The problem of sliding windows is not well solved though there are some discussions in [7].

5 Duplicate-insensitive

In many real applications, duplicates may occur when data elements are observed and recorded multiple times at different data sites. For instance, as pointed out in [7,9] the same packet may be seen at many tap points within an IP network depending on how the

[1]Note that the form of our relative error metric is biased towards the *head* (i.e., finer error guarantees towards lower ranks). Clearly, finer error guarantees towards the *tail* may be obtained if the data elements are ordered in reverse.

packet is routed; thus it is important to discount those duplicates while summarizing data distributions by rank queries (quantiles). Moreover, to deal with possible communication loss TCP retransmits lost packets and leads to the same packet being seen even at a given monitor more than once. Furthermore, duplicates may often occur due to the projection on a subspace if elements have multiple attributes.

In such applications, there may be many *duplicated elements* in a data stream S. To discount the duplicates in S, rank queries have to be issued against D_S instead of S where D_S denote the set of distinct data elements in S. Note that in D_S, there are no duplicates but many different elements may happen to have the same values. Consequently, all the challenges in approximately computing rank queries remain the same. The unique challenge is to discount duplicates without keeping all elements in a summary/sketch.

With the recent data-intensive applications in sensor/P2P networks, duplicate-insensitive techniques are also highly desirable to achieve high communication fault-tolerance. In [23], Manjhi, Nath, and Gibbons propose an effective adaption paradigm for in-network aggregates computation over stream data with the aim to minimize communication costs and to achieve high fault-tolerance. As indicated, a duplicate-insensitive technique for approximately computing quantiles may be immediately obtained by a combination of their tree-based approximation technique and the existing *distinct counting* technique in [4]. It can be immediately applied to a single site, where a data stream has duplicated elements, with the uniform precision guarantee $|r' - r| \leq \epsilon n$ by confidence $1 - \delta$ and space $O(1/\epsilon^3 \log 1/\delta \log m)$ where $n = |D_S|$.

In [9], Cormode and Muthukkrishnan present a *DISTINCT RANGE SUMS* technique by applying the FM [12] technique on the top of the *count-min* [5]. The technique can be immediately used to approximately processing RQ with the uniform precision guarantee $|r' - r| \leq \epsilon n$, confidence $1 - \delta$, and space $O(\frac{1}{\epsilon^3} \log \frac{1}{\delta} \log^2 m)$. Independently, Hadjieleftheriou, Byers, and Kollios [18] also developed two novel duplicate-insensitive techniques to approximately compute quantiles in a distributed environment. Applying their techniques to a single site immediately leads the uniform precision guarantee $|r' - r| \leq \epsilon n$ by confidence $1 - \delta$ and space $O(\frac{1}{\epsilon^3} \log \frac{1}{\delta} \log m)$.

Very recently, we developed the first space- and time- efficient, duplicate-insensitive algorithms [31] to continuously maintain a sketch of order statistics over data stream to enforce relative ϵ-approximation. They have been developed based on the probabilistic counting techniques in [4, 12]. They not only improve the existing precision guarantee (from uniform ϵ-approximation to relative ϵ-approximation) but also reduce the space from $O(\frac{1}{\epsilon^3} \log \frac{1}{\delta} \log m)$ to $O(\frac{1}{\epsilon^2} \log \frac{1}{\delta} \log m)$ where m is the element domain size.

The sliding window problem remains open.

6 Miscellaneous

Continuous queries are issued once and run continuously to update query results along with updates of the underlying datasets. In [6], Cormode *et al* provide a novel algorithm to continuously processing a rank query in a sensor/network environment with the aim to minimize the communication costs. We [22] recently developed novel techniques to efficiently processing a massive set of rank queries. The objective is to share the computation as much as possible.

Quantiles computation against multi-dimensional datasets has been recently investigated in [19,29]. Yiu *et al* [29] developed an efficient R-tree based algorithm to provide exact solutions regarding an off-line computing environment, while Hershberger *et al* [19] presented an effective one-scan approximation technique to maintain a small space sketch with the uniform precision guarantee ϵN. The problems of relative error guarantee and duplicate-insensitiveness over data streams remain open.

7 Future Studies

While a number of theoretical problems still remain open, the following two new applications in continuously maintaining order statistics may be worth some exploration.

Uncertainty. In many applications, we may need to deal with data sets with uncertainty; that is, the value of a data element is not fixed. In such applications, ranks of data elements have to be specified probabilistically. The challenge is to effectively build a small space summary by one-scan techniques while the distribution of each data element is continuously sampled.

Graphs. Graphs are very common to model many real applications; for instance, IP network and communication network. To manage and explore such networks, summarizing distributions of various node degrees information is an important issue. The challenge is to maintain a small space summary by one-scan while the underlying graph structure is continuously "disclosed".

References

[1] M. Ajtai, I. S. Jayram, R. Kumar, and D. Sivakumar. Approximate counting of inversions in a data stream. In *STOC 2002*.

[2] A. Arasu and G. S. Manku. Approximate counts and quantiles over sliding windows. In *PODS04*.

[3] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and issues in data stream systems. In *PODS'02*.

[4] Z. Bar-Yossef, T. S. Jayram, R. Kumar, D. Sivakumar, and L. Trevisan. Counting distinct elements in a data stream. In *RANDOM'02*.

[5] C. Cormode and S. Muthukrishnan. An improved data stream: The count-min sketch and its applications. In *Latin American Informatics*, 2004.

[6] G. Cormode, M. Garofalakis, S. Muthukrishnan, and R. Rastogi. Holistic aggregates in a networked world: Distributed tracking of approximate quantiles. In *SIGMOD'05*.

[7] G. Cormode, F. Korn, S. Muthukrishnan, and D. Srivastava. Effective computation of biased quantiles over data streams. In *ICDE'05*.

[8] G. Cormode, F. Korn, S. Muthukrishnan, and D. Srivastava. Space- and time-efficient deterministic algorithms for biased quantiles over data streams. In *PODS'06*, 2006.

[9] G. Cormode and S. Muthukrishnan. Space efficient mining of multigraph streams. In *PODS'05*.

[10] G. Cormode, S. Muthukrishnan, and W. Zhuang. What's different: Distributed, continuous monitoring of duplicate-resilient aggregates on data streams. In *ICDE'06*.

[11] M. Datar, A. Gionis, P. Indyk, and R. Motwani. Maintaining stream statistics over sliding windows. In *SODA03*.

[12] P. Flajolet and G. N. Martin. Probabilistic counting algorithms for data base applications. *Journal of Computer and System Sciences*, 31(2):182–209, 1985.

[13] A. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. Strauss. How to summarize the universe: Dynamic maintenance of quantiles. In *VLDB2002*.

[14] M. Greenwald and S. Khanna. Power-conserving computation of order-statistics over sensor networks. In *PODS'04*.

[15] M. Greenwald and S. Khanna. Space-efficient online computation of quantile summaries. In *SIGMOD'01*.

[16] S. Guha, N. Koudas, and K. Shim. Data-streams and histograms. In *STOC 2001*.

[17] A. Gupta and F. Zane. Counting inversions in lists. In *SODA'03*.

[18] M. Hadjieleftheriou, J. W. Byers, and G. Kollios. Robust sketching and aggregation of distributed data streams. Technical report, Boston University, 2005.

[19] J. Hershberger, N. Shrivastava, S. Suri, and C. Toth. Adaptive spatial partitioning for multidimensional data streams. In *ISAAC'04*.

[20] J.I.Munro and M.S.Paterson. Selection and sorting with limited storage. In *TCS12*, 1980.

[21] X. Lin, H. Lu, J. Xu, and J. X. Yu. Continuously maintaining quantile summaries of the most recent n elements over a data stream. In *ICDE'04*.

[22] X. Lin, J. Xu, Q. Zhang, H. Lu, J. Yu, X. Zhou, and Y. Yuan. Approximate processing of massive continuous quantile queries over high speed data streams. *TKDE*, 18(2):683–698, 2006.

[23] A. Manjhi, S. Nath, and P. B. Gibbons. Tributaries and deltas: Efficient and robust aggregation in sensor network streams. In *SIGMOD'05*.

[24] G. S. Manku, S. Rajagopalan, and B. G. Lindsay. Random sampling techniques for space efficient online computation of order statistics of large datasets. In *SIGMOD'99*.

[25] G. S. Manku, S. Rajagopalan, and B. G. Lindsay. Approximate medians and other quantiles in one pass and with limited memory. In *SIGMOD*, 1998.

[26] S. Nath, P. B. Gibbons, S. Seshan, and Z. R. Anderson. Synopsis diffusion for robust aggregation in sensor networks. In *SenSys'04*.

[27] N. Shrivastava, C. Buragohain, D. Agrawal, and S. Suri. Medians and beyond: new aggregation techniques for sensor networks. In *SenSys'04*, pages 239–249, 2004.

[28] Version2.0. Financial analysis package mt for gauss^{TM} - version 2.0. In *Aptech System, INC*, 2003.

[29] M. Yiu, N. Marmoulis, and Y. Tao. Efficient quantile retrieval on multi-dimensional data. In *EDBT'06*.

[30] Y. Zhang, X. Lin, J. Xu, F. Korn, and W. Wang. Space-efficient relative relative error order sketch over data streams. In *ICDE'06*.

[31] Y. Zhang, X. Lin, Y. Yuan, M. Kitsuregawa, X. Zhou, and J. Yu. Summarizing order statistics over data streams with duplicates (poster). In *ICDE2007*, 2007.

Contributed Papers

Managing Ontologies: A Comparative Study of Ontology Servers

Mohammad Nazir Ahmad and Robert M. Colomb
School of Information Technology and Electrical Engineering
The University of Queensland
AUSTRALIA
{nazir,colomb}@itee.uq.edu.au

Abstract

An ontology is increasingly becoming an essential tool for solving problems in many research areas. The ontology is a complex information object. It can contain millions of concepts in complex relationships. When we want to manage complex information objects, we generally turn to information systems technology. An information system intended to manage ontology is called an ontology server. The ontology server technology is at the time of writing quite immature. Therefore, this paper reviews and compares the main ontology servers that have been reported in the literatures. As a result, we point out several research questions related to server technology.

Keywords: Ontology, Ontology Server, Ontology Tool.

1 Introduction

An ontology is an explicit specification of a conceptualization (Gruber, 1993). It is a designed artefact that formally represents agreed semantics of a domain interest in computer resources (Gruber, 1993; Guarino, 1998). This enables the sharing and reuse of information and allows for the interoperation of information systems (Pretorius, 2005). Although not a new field, ontology research has recently received renewed interest in many fields such as semantic web (e.g., Berners-Lee, 1999; Mika, 2004), databases (e.g., Guarino, 2002), electronic commerce (e.g., Barley et al., 1997; Corcho et al., 2001), knowledge management (e.g., Rothenburger et al., 2006; Chi et al., 2006), Khosla, 2003), information retrieval (e.g., Guarino et al., 1999; Abdelali et al., 2003), bioinformatics (e.g., Karp et al., 2002; Bada et al., 2004), software engineering (e.g., Welty et al., 1999; Derridder et al., 2000), intelligent systems (e.g., Akkermans et al., 2004) and so forth. Ontology applications have been classified in Jasper et al. (1999), Mizoguchi (2003) and most comprehensively survey proposed by Hart et al. (2004).

2 Managing Ontologies and Ontology Tools

A major problem with ontologies is how to manage them. Thus, there are related research problems in the field of ontologies which include: (a) creating large-scale ontologies (e.g., Lenat et al., 1995), (b) defining expressive languages for representing ontological knowledge (e.g., Lopez et al., 1999; Karp et al., 1999; Fensel et al., 2001; Bechhofer et al., 2004; Klyne et al., 2004; Brickley et al., 2004) and (c) implementing systems or tools that support ontology-based applications (e.g., Farquhar et al., 1997; Chung et al., 2000; Cui et al., 2000; Li et al., 2003; Dameron et al., 2004; Valo et al., 2005; Starlab, 2006).

When a new ontology is going to be developed, several basic questions arise related to the methodologies, tools and languages to be used in ontology development as reported in (Corcho et al., 2003). However, the management tasks in ontologies are not ontology development, but it should focus on other aspects such as how to use ontologies for example in supporting information systems interoperation. We would say these aspects are from two fundamental ontology engineering challenges: reusability and usability. The former aspect is main goal in building ontologies (Uschold et al., 1996; Motta et al., 1999; Gomez-Perez et al., 1999; Gruber, 1995). The latter aspect is how to make use of ontologies. This is an important goal in interoperating systems, where an exchange can involve thousands of players. We believe that most research focuses on the development of single or integrated tools for supporting ontology development. In other words, they aim to support ontology reusability. The main ontology tools in the mainstream literature for ontology development are well reviewed in (Corcho et al., 2003; Duineveld et al., 2000; Mizoguchi, 2003; Fensel, 2001).

We believe that the notion of ontology server has originally stemmed from the research of ontology development tools. Most works describe implicitly or explicitly the server as a kind of isolated or integrated tool for building ontology (e.g., Li et al., 2003; Farquhar et al., 1997; Eklund et al., 2002; Noy et al., 2002; Sure et al., 2002; Noy et al., 2000). The isolated tools are not fully integrated with other activities of the ontology lifecycle (Corcho et al., 2003). Such tools are Ontolingua Server (Farquhar et al., 1997), Ontosaurus (Swartout et al., 1997) and OntoRama (Eklund et al., 2002). The latter is much more ambitious, built as robust integrated environment or suite that provides technological support to most of the ontology lifecycles activities. They have extensible, component-based architectures, where modules can easily be added to provide more functionality to the environment. (Corcho et al., 2003). Among these environments, we can cite Protégé-2000 (Noy et al., 2000), WebODE (Arpirez et al., 2001) and OntoEdit (Sure et al., 2002).

However, the discussion of ontology server in those ontology development tools is quite confusing and still unclear. Some studies discuss the server as ontology repository. In this context, the server is mainly about database technology (e.g., Pan et al., 2003; Harrison et al., 2005). Some other studies discuss both ontology repository and server functionality. In this context, the server is described as an information system (e.g., Farquhar et al., 1997; Li et al., 2003; Starlab, 2006; Mauger, 2005; Dameron et al., 2004; Chung et al., 2000; Swartout et al., 1997; Arpirez et al., 2001; Sure et al., 2002). Therefore, we would like to highlight three important points about the notion of ontology server discussed in many studies as follows: (a) many studies discuss ontology tools mainly for building ontology. Some of these tools consist of many other modules including the ontology server. In this context, the notion of ontology server is mostly a database issue. For example, Corcho et al. (2003) describe Ontosaurus (Swartout et al., 1997) as consisting of two modules: ontology server and ontology browser. In other words, the notion of ontology server is only referred to ontology repository such as in (Pan et al., 2003; Harrison et al., 2005). (b) Some studies discuss clearly ontology servers and use the word "ontology server" in their work. These studies include both server ontology repository and functionalities (e.g., Li et al., 2003; Starlab, 2006; Mauger, 2005; Dameron et al., 2004; Oberle et al., 2002), (c) In some other studies are similar to (b), except the notion of ontology server is limited to ontology repository as mentioned in (a). An example is the Ontolingua server.

We draw three conclusions for research directions form understanding in three important things: First the points above, we argue with the notion of ontology server as mentioned in (a) and (c). The central issues in ontology server development should include ontology repository and server functionalities. Further, an ontology server is a kind of information system and we can discuss the server from information systems perspective. Second, most of the ontology servers in the literature are used for building ontologies. In other words, the current studies give emphasis more the ontology server supporting some activities in the ontology building lifecycle. We argue that, third, to have a clearer understanding in this area, we would need a so-called "framework" to give a brief idea of ontology server research.

To extend these positions, we point out several basic questions related to the functionalities, repository, and methodological support and so forth to be used in the ontology server development. These questions would include:

What are the main components of ontology server? What kind of framework can we use to understand the server? What methodologies can we use for building an ontology server? Does any methodology provide support for building the server? What general activities should we consider when building the server? Which is the life-cycle of an ontology that the server can support? How are the ontologies stored in the server? What are the main server functions? How can applications interoperate with ontology servers and/or use the ontologies? What sort of ontology languages is the server supposed to support? Is the language chosen appropriate for exchanging message between different applications? What is the main purpose of ontology server to be developed? What sort of the ontology server architecture can we propose? And so forth.

In this paper, we will present the main characteristics of ontology servers, which can assist practitioners and researchers in this field to gain answers to the questions above. Since at the time of writing this field is fairly immature, we will provide guidelines that outline several opportunities related to ontology server research.

3 Ontologies and Semantic Web

Ontologies are an important part of the semantic web (Berners-Lee, 1999). The semantic web is an extension of the current web in which information is given well defined meaning, better enabling computers and people to work in cooperation (Berners-Lee, 1999). In the semantic web, ontologies can be used to encode meaning into a web page, so that intelligent agents (applications) can understand what the web page is about, and therefore provide humans with more useful cooperative services (Berners-Lee, 1999). Driven by application needs and the semantic web vision, the ontology server is believed to be a key component for supporting semantic web applications (Agrawal, 2002). Therefore, in this study, we will only focus on the ontology servers that support ontology-based applications in open environments such as the web and semantic web. Furthermore, we will look into several ontology servers: (a) are mostly cited in literatures, (b) they explicitly use word "ontology server" in their reports and address server development issues, (c) we assume that several of them are still progressing and (d) focusing on two main aspects of the ontology server: server functionalities and ontology repository.

4 An Ontology Server

Ahmad et al (2006) show that the ontology server is used at design, commit and runtime. At design time, the ontologist uses the server for developing ontologies. At commit-time, players (i.e. applications) will commit to a limited part of the ontology to enable the exchange of messages. At runtime, players exchange messages mediated by shared ontologies. Zachman (1987) has proposed an information system architecture framework, which has been well accepted (Evernden, 1996; Inmon et al., 1997; Ells, 1998; O'Rourke et al., 2003; Jones, 2005). There are three key concepts (knowledge, process and communication) and related enabling technologies described by Zachman (1985). Since the ontology server is a kind of information system, it can be described in terms of this framework (see details in (Ahmad, 2006)). In the context of server development, it is firstly important to concentrate on the process aspect (server functionality) and knowledge aspects (ontology repository) of ontology server design.

4.1 Ontology Server Functionality

There is a number of different ontology server implementation available. Their functionality focuses on editing, browsing and storing ontologies. In some cases the ontology server also provides an inference engine that allows statements about the relationships between entities in different ontologies to be tested or retrieved (Li et al., 2003). Pan et al (2003) state a number of facilities the server may provide such as support for creating and editing ontologies, support for publishing and retrieving ontologies (by humans via a graphical user interface and/or via a network protocol), support for recording metadata about ontologies and the relationships between them, support for interactively browsing the structure of ontologies and inference mechanisms to verify the consistency of ontologies.

We would say that there are two main approaches to implementing ontology server functionalities: (a) tool development and (b) Application Programming Interfaces (APIs). The former approach is about developing any kind of application on top of the ontology repository such as ontology browser, ontology editor, ontology translator and so on. In this approach, those tools typically perform a single aspect of ontology server functionality. Those tools can make use of an ontology stored in the ontology repository. For example, Starlab (2006) has developed an ontology browser, ontology manager and ontology modeller to make use ontologies stored in the server repository. The API approach is to write a program that provides some services to the ontology repository. Typically, any applications or tools can use those APIs to interface the ontology repository. For example, Starlab (2006) has provided some basic database APIs for accessing ontologies stored in the ontology repository. With respect to server functionality, to avoid confusion, we should be aware of many terms appearing in studies of server functions such as server functionality and functions (Starlab, 2006), ontology service (FIPA, 2001), server process (Chung et al., 2000), server services (Farquhar et al., 1997), server operation and ontology operation (Dameron et al., 2004), and server facilities (Pan et al., 2003).

4.2 Ontology Server Repository

In this paper, we use term “ontology repository” and “server repository” interchangeably. We would say that the discussion of ontology repository gives emphasis more to database issues such as storing and organizing ontologies in database. There are such studies discussing specifically ontology repository aspects such as Pan et al. (2003) and Harrison et al. (2005). For example, Harrison et al. (2005) have proposed a generic representation for ontology repository. Similar to Pan et al. (2003), they have introduced a so-called lightweight ontology repository for enabling shareable and maintainable ontologies. A lightweight ontology is referred to a kind of ontology that simple, generic form and does not include axiom that allow deductions to be made. Repository is a prototype implementation of a design to allow ontology designers and agents to use open web standards to publish and retrieve ontologies and metadata about them. Key features of the system include the use of the HTTP protocol following the REST (Representational State Transfer) architectural style, the representation of recorded information about ontologies and the repository information schema using RDF and its schema language RDFS, and the use of URNs to identify ontologies. The use of web standards for communication between agents and web-based resources such as ontology repositories enables a more lightweight and open architecture for agent interaction with these resources (Pan et al., 2003). Harrison et al. (2005) discuss two methods of storing the ontologies. The first method involves storing individual ontology in a separate flat file. The file provides a more straightforward view where its content can be inspected. However, the main problem with this method is that a search engine would need to be developed to search the contents of the ontology files (Harrison et al., 2005). The second method uses a database to store ontologies. Databases have indexing and other capabilities that enable faster searching. Therefore, to ensure scalability and maintainable large ontologies, the database method rather than a flat file we think would preferable for storing ontologies. However, how the ontologies are stored depends on how the ontology representation should appear in the server and what types of database are considered (i.e. relational, object-oriented or object-relational).

4.3 A Comparison of Ontology Servers

After an extensive search on the Internet and several journals and conferences, and with several focus as stated in section 3, we selected the following eight ontology servers: Ontolingua server (Farquhar et al., 1997), ACOS (Li et al., 2003), Starlab (2006), KAON (Oberle et al., 2002), OntoRama (Eklund et al., 2002), OWS (Dameron et al., 2004), FIPA server (2001), and Adapted Ontology Server (Chung et al., 2000).

The Ontolingua server has been running since 1995. It has been developed in the Knowledge Systems Laboratory (KSL) at Stanford University. It is a tool that supports distributed, collaborative editing, browsing and creation of Ontolingua ontologies with a form-based web application (Corcho et al., 2003). Remote editors can browse and edit ontologies, and remote or local applications can access any of the ontologies in the ontology repository with the OKBC protocol (Farquhar et al., 1997). The Ontolingua ontology uses the representation languages, Ontolingua Frame Ontology and KIF, which are wide spectrum language capable of representing fine features of concepts (i.e. are based on description logics). The Ontolingua server supports ontology inclusion and circular dependencies (Farquhar et al., 1997). Its consistency-check capability, however, is restricted to the functions similar to database schema checking (Li et al., 2003). For instance, “all slots, slot values, facets and facets values are checked to make sure that they conform to the constraints that they apply (i.e. domain, range, slot value type, and cardinality constraints)” (Li et al., 2003). This means that semantic consistency checking is done in domain experts’ heads. It is a large project focusing on ontology development (Cui et al., 2000). It has built sophisticated tools for developing and maintaining frame-based ontologies. It focuses on formal ontology specifications, and reuse and translation to different ontology implementation systems (Cui et al., 2000). However, it does not address problems related to legacy systems and tools to merge ontologies (Cui et al., 2000). The users interact with the single server through a web browser to create, edit and browse ontologies. Users have to tolerate the network delays and server response delays. The Ontolingua server uses Ontolingua-based repository for storing ontologies and developing a set of tools for demonstrating server functions.

The second server is ACOS. It has been developed in the Intelligent System Laboratory (ISLab) at British Telecom Research. Li et al. (2003) claim it to be a community-oriented ontology server. It provides the way for the so-called "community" to construct ontologies. The concept of "community" in ontology management enables everyone to have the opportunities of influencing ontology construction. This depends on the users' importance score, and such score is computed mechanically based on how active these users are in contributing to the knowledge base. They came up this kind of vision because in an open environment, ontology is the asset of all participants; every user can join, contribute and leave such community. The opposite of this approach is called "central". The central-controlled mechanism would not appear to be appropriate for this situation (Li et al., 2003). However, most of the ontology server implementations in many studies apply the "central" approach to ontology development (e.g., Farquhar et al., 1997; Chung et al., 2000; Frehiwot et al., 2001; Eklund et al., 2002; Starlab, 2006). The "central" means a centralized ontology server, only a fixed group of users have the rights to modify ontologies, which is similar to the situation in database management (Li et al., 2003). Similar to Ontolingua server, the ACOS has also provided consistency checking but not limited to humans. A wide range of software agent communities also can share it and the server facilitates on-line ontology construction, consistency-check and use (Li et al., 2003). Li et al. (2003) claim that ACOS is designed to be an online community in which a diverse group of software agents can contribute and use ontologies at runtime. A key enabler in this scenario is a high degree of "shareness" of the ontologies maintained by the server. This relates to designing an appropriate knowledge representation, which is the first step towards building an ontology server. In this context, Li et al. (2003) believe that to achieve "shareness", it is required that local features of ontology are removed. Examples of constructors that can bring in local features include *property* in DAML-OIL or *slot* in Ontolingua (Farquhar et al., 1997). Other examples of constructors that bring in local effect are *part-of* in Framework (Farquhar et al., 1997) and *disjoint-with* in Descriptions Logic (Borgida et al., 1989). In other words, Li et al. (2003) accept the shared ontologies should have minimal expressiveness, which consist of a minimal set of axioms written in a language of minimal expressivities. Thus, to achieve a high degree of shareness, Li et al. (2003) believe that constructors *Class*, *subClassOf*, *SameClassAs*, *SuperClassOf* and *InstanceOf* offer more stability in the process evolution and minimal expressivities. Compared to Ontolingua server, the ACOS uses DAML-OIL as its ontology representation language. The ACOS server uses file-based repository for storing DAML-OIL ontologies and developing a set of APIs for implementing server functions. To facilitate collaborative ontology development, this server implemented an import mechanism that is similar to the *inclusion model* in Ontolingua server and *extension relationship* in FIPA ontology server (FIPA, 2001).

Third is the Starlab. This server is under development and still progressing several deliverables. This research initially falls within the DOGMA research framework (Jarrar et al., 2002). At the time of writing, we would say that this is a new ontology server research project, is funded for 5 years by the Vrije Universitet Brussel. The mission of this server is to assist the gathering and incremental growth of ontologies (Starlab, 2006). In terms of ontology representation, this server is in line with the rationale of ACOS server previously discussed. The proposed ontology model consists of five basic elements: context, terms, concepts, roles and lexicon (Starlab, 2006). Starlab (2006) claims that constraints and derivation rules are intentionally left outside the ontology. At this moment, they are still experimenting with the ontology model and an early version of ontology server has been implemented. In the first prototype, consistency-check and user control are not included (Starlab, 2006). This server uses MSQL server to store the ontology and ontology objects are expressed in XML. Its server functions are implemented based on developed tools and basic database APIs (Starlab, 2006). In addition, the server's ontology representation also benefits from graphical notation based on Object Role Modeling (ORM) languages.

Fourth is the KAON server developed by AIFB in Karlsruhe University. Similarly to Ontolingua server and Starlab, the KAON server is a result of sophisticated tools delivered by a large research project conducted at the University of Karlsruhe. This server provides its functions through a set of APIs and store RDF-based ontologies using a relational database (Oberle et al., 2002; Maedche et al., 2003). It is an early prototype of an ontology-based application server. In essence, it uses JBoss and extends by the tools of the KAON tool suite for reasoning with software components (Oberle et al., 2002). The goal is to support the developer in his daily tasks with reasoning. In terms of ontology representation, it is similar to Ontolingua server: the complexity of ontology is not left outside the ontology, which contrasts with Starlab and ACOS.

Fifth is the OntoRama server was developed by Eklund et al. (2002). The main functions of OntoRama include search, compare and modify WebKB ontologies. It does not support consistency-check or cross-ontology queries; as a result, its capability of supporting online collaborative ontology construction is restricted (Li et al., 2003). In our opinion, this server demonstrates functions of storing and browsing RDF-based ontologies, while supporting ontology development is not its main purpose.

Sixth, the OWS server proposed by Dameron et al. (2004) at the Stanford Medical Informatics (SMI), Stanford University. This server does not generally deal with ontology development. It is used for all other tasks, assuming ontologies already exist. The implementation of this server is bit tricky. In fact, the server described by Dameron et al. (2004) does not itself serve as ontology repository (i.e. the server itself does not store the ontologies). The server provides services, taking ontologies as inputs. Hence, they claim that ontology evolution is not a problem here, because a change simply gives the server a new ontology as inputs. In our opinion, we would say that this server supports runtime interoperation. Server functions are implemented using web services assuming that ontologies naturally reside in any player's sites. In other words, the ontology storage is file-based, which are held on different players' sites.

Seventh is the FIPA ontology server. The FIPA (Foundation for Physical Agents) is a standards body, which has developed interaction standards for agents in open environment. In the FIPA's ontology server specification, its server's ontological representation is divided into a *fine-grained ontology,* called heavyweight ontology in (Pretorius, 2005) and *a coarse ontology* that consists of a minimal set of axioms written in a language of minimal expressivity, called lightweight ontology in (Pretorius, 2005). However, the FIPA's server supports these two kinds of ontology representation but with different scope of use and level of detail (FIPA, 2001).

Eighth is the server proposed by Chung et al. (2000) at SungKyunKwan University, Korea. This server is intended specifically for ontology developed for electronic commerce applications. In terms of ontology representation, Chung et al. (2000) define two main criteria for ontology: (a) ontology can be translated and (b) ontology should be practical. Its server functions include gathering information from the web, creating a relation, modifying and rebuilding the standard ontology and servicing the standard ontology. This server uses a MySQL database to store ontologies and Java APIs for implementing its server functions (Chung et al., 2000). We assume that this server allows medium ontological representation. This is because the standard ontology is built from the local terms used in sites, and then the server provides an editor tool for making relation between ontologies (Chung et al., 2000). In addition, they claim that a standard ontology necessarily has the objective and concrete property (Chung et al., 2000). In general, this server supports ontology development but is limited to electronic commerce and deals with the current web.

4.4 Comparison Framework

A survey of ontology servers was firstly done in (Li et al., 2003). However, Li et al (2003) did not include several key ontology servers such as KAON and OWS in their survey. In addition, the proposed comparison is too general and does not highlight other important aspects of development issues such as a standard development approach, methodological support and the use of graphical modeling language. In this study, we have compared ontology servers with respect to the similar criteria to (Li et al, 2003), extending some of them, also adopting several dimensions used in (Duineveld et al., 2000; Corcho et al., 2003). Consequently, we incorporate all these criteria into five dimensions as follows:

General: It refers to the generic aspects of the server development such as: *What is the main goal of developed ontology server? Who are those developers? Can we access to those servers? What is the main phase that server can support? What are the main deliverables of the server development? Which kinds of management provision does the server have? Does the server support collaborative environment? What is the status of the server?* **Ontology:** This dimension refers to the general questions about ontology representation that the server supports, such as: *What the main ontology engineering challenges that the server mostly focuses on? What types of ontology can the server support? What does knowledge representation in server repository look like? What ontology language is the server based on? How expressive is language? Does the server use all the language features (i.e. ontological constructors) to represent shared ontologies?* **Features:** It represents general features of ontology server in terms of: *What are the main server functions have been developed? What types of repository platform the server use to store ontologies? What is the architecture of the server? Does the server provide extensibility?* **Implementation:** This dimension addresses some aspects of the server technology including: *How is the server functionality implemented? What type of technology platform does the server use and support? Does the server use a standard technology? What type of technology is used to access to the ontology repository?* **Methodology:** This dimension concerning the methodology that ontology server gives support to, and also the methodology used to develop a server such as: *Is there any methodology support for building the server? Is there any methodology support for the ontology?* To have a clearer understanding of this study, as a result, we summarize this comparative study in Table 1 (see appendix).

5 Conclusions and Future Works

We point out some important points based on a Table 1. First, we see that most ontology servers available mostly focus on design time (see *main phase of ontology life cycl*e). It gives supports more to ontology development and tends assume its users to be ontologists. Its main goal is reusability (see *main ontology engineering challenges*) since it saves time and cost. From this, we argue that the server development should be determined by what ontology is made for. For example, the ontologies supporting runtime interoperation, which used in business application, are slightly different from ontologies for supporting engineering applications (Colomb et al., 2006; Hart et al., 2004). Therefore, we should firstly understand the intended use of the ontology which will then lead to how it is engineered and then how the supporting server is going to be developed. Second, surprisingly, none of the existing ontology servers are discussed in the context of commit-time issue (see *main phase of ontology life cycl*e). Although in general we have a design and runtime, but the issue how players can be helped to see the parts of ontology before joining the exchange should be firstly addressed. Third, a standard methodological approach for designing ontology and ontology servers is still missing (see *a methodology dimension*). Therefore, there is no server modelling profiles that can be referenced as useful example to guide a developer on how to develop ontology server. However, recently, there have been initiatives to bridge MDA (Model Driven Architecture) and ontologies. Many researchers suggest the use of Unified Modeling Languages (UML) in ontology development (e.g., Cranfield et al., 1999; Backlawski et al., 2002; Kogut et al., 2002; Colomb et al., 2006; Gasevic et al., 2005; Djuric et al., 2005). Thus, it would be advantageous to consider these works in the context of ontology and ontology server development. For example, it would be useful to benefit from the standard graphical ontological representations like UML. Unfortunately, most of the ontology and server developed do not consider the use of a standard graphical language.

Fourth, there is almost no explicit discussion of servers for supporting information systems interoperation in the mainstream literatures. So, the use of server as a runtime tool is not well developed. Thus, we would say, the server functions to support many aspects of the semantic web are still missing. Fifth, there are issues in ontological representation that relate to ontological repository. Most of the servers do not use database to store ontologies. However, we believe that most of the servers are moving towards using database and Java platforms (see *implementation*). Sixth, generally, there are two types of ontologies; lightweight and heavyweight (see *types of ontologies*). The former simply sees ontology as a description with the aim of organizing concepts. The latter defines ontology is a complete theory consisting of both a formal vocabulary and defined axioms that allow further deductions or inferences to be made. Most of the servers are influenced by lightweight ontologies. These servers only support using minimal language constructors, which have very minimal expressivity of ontology precision (i.e. *class*, *subClassof, IsA*). The resultant ontology from a knowledge representation point of view looks simple compare to heavyweight ontology. However, we argue that this situation is also determined by what the ontology is made for. For example, ontology supporting information systems interoperation needs heavyweight ontology that can be used by software agents.

From the perspective of a comparative study, we report the current technologies of ontology and the semantic web, particularly on ontology server development. Although we do not provide complete technical details, it is good enough to illustrate opportunities to enhance ontology server development. We know that ontology server is built around database issues but considering both server functionalities and server repository lead to treat the server is a kind of information systems that can benefit from other field like software engineering. Our future work is to investigate a useful methodological approach for designing ontology and ontology server in the context of commit-time issues. We argue that this should importantly be addressed since in the complex and large-scale ontologies, most of the players (i.e. software agents) are directly interested in the portion of the ontologies before joining some exchanges.

6 References

Ahmad, M.N., Colomb, R.M. and Cole, J. (2006): An Ontology Server: A Knowledge Tool For Systems Interoperability in Semantic Web Context. *Malaysian Journal of Information Technology,* 18(1): 1-23, June 2006. ISSN-0128-3790

Arpirez, J.C., Corcho, O., Fernandez-Lopez, M., Gomez-Perez, A. (2001): WebODE: a scalable ontological engineering workbench. *Proc. First International Conference on Knowledge Capture (KCAP_01),* Victoria, pp. 6–13. ACM Press.

Agrawal, R. Making Semantic Web Real: Some Building Blocks. (2002): *Proc. International Workshop on Semantic Web,* Hawaii, USA.

Abdelali, A., Cowie, J., Farwell, D., Ogden, B., Helmreich, S. (2003): Cross-Language Information Retrieval Using Ontology. *Proc. TALN Batz-Sur-Mer,* France.

Akkermans, J., Baida, Z., Gordijn, J., Pe˜na, N., Altuna, A., Laresgoiti, I. (2004): Value Webs: Using ontologies to Bundle Real-World Services. *IEEE Intelligent Systems,* 19(4): 57–66.

Baclawski, K., Kokar, M., Kogut, P., Hart, L., Smith, J., Holmes, W., Letkowski, J., Aronson, M., Emery, P. (2002): Extending the UML for Ontology Development. *International Journal of Software and Systems Modeling,* 1(2): 142-156.

Barley, M., Clark, P., Williamson, K., Woods, S. The Neutral Representation Project. (1997): *Proc. AAAI'97 Spring Symposium on Ontological Engineering,* AAAI Press.

Borgida, A., Brachman, R.J., McGuiness, D.L. and Resnick, L.A. CLASSIC: A Structural Data Model for Objects. (1989): *Proc. ACM SIGMOD International Conference on Management of Data,* Portland, 58-67.

Bada, M., Stevens, R., Goble, C., Gil, Y., Ashburner, M., Blake, J. A., Cherry, M., Harris, M., Lewis, S. (2004): A Short Study on the Success of Gene Ontology. *Web Semantics: Science, Services and Agents on the World Wide Web,* 1: 235–240.

Berners-Lee, T. *Weaving the Web.* (1999): London, Orion Business Books, 1999.

Bechhofer, S., Van Harmelen, F., Hendler, J., Horrocks, I., McGuiness, D.L. Patel-Schneider, P.F. and Stein, L.A. OWL Web Ontology Language Reference. W3C Recommendation, http://www.w3.org/TR/2004/REC-owl-ref-20040210/. Accessed 23 July 2006

Baclawski, K., Kokar, M., Kogut, P., Hart, L., Smith, J., Holmes, W., Letkowski, J., Aronson, M. (2001): Extending UML to Support Ontology Engineering for the Semantic Web. *Proc. UML 2001,* Toronto, CA.

Baclawski, K., Kokar, M., Kogut, P., Hart, L., Smith, J., Holmes, W., Letkowski, J., Aronson, M., Emery, P.

(2002): Extending the UML for Ontology Development. *International Journal of Sofware and Systems Modeling* ,1(2):142-156.

Brickley, D., Guha, R.V. and McBride, B. RDFS Resource Description Framework Schema. W3C Recommendation, http://www.w3.org/TR/2004/REC-rdf-schema-20040210/. Accessed 23 July 2006

Cranfield, S. and Purvis, M. UML as an Ontology Modeling Language. (1999): *Proc. Workshop on Intelligent Information Integration, 16th International Joint Conference on Artificial Intelligence, IJCAI-99.*

Chung, H., Choi, J., Yi, J., Han, J., Lee, E. (2000): A Construction of the Adapted Ontology Server in EC, *Lecture Notes in Computer Science*, 1983:355-360.

Colomb, R.M., Raymond, K., Hart, L., Emery, P., Welty, C., Xie, G.T. and Kendall, E. (2006): Version 3.3: The Object Management Group Ontology Definition Metamodel. In *Ontologies for Software Engineering and Software Technology,* 1-25. Calero, Coral, Francisco and Mario, P. (eds). Springer.

Corcho, O., Gemez-Perez, A. (2001): Solving Integration Problems of e-Commerce Standards and Initiatives through Ontological Mappings. *Proc. IJCAI'01.*

Corcho, O., Fernandez-Lopez, M. and Gomez-Perez, A. (2003): Methodologies, Tools and Languages for Building Ontologies: Where is their meeting point? *Data & Knowledge Engineering,* 46: 41–64.

Chi, Y.L., Hsu, T.Y., Yang, W.P. (2006): Ontological Techniques for Reuse and Sharing Knowledge in Digital Museums. *Journal of Electronic Library,* 24(20): 147-159.

Cui, Z. and O'Brien, P. Domain Ontology Management Environment, *In the Proceedings of the 33rd Hawaii International Conference on System Sciences*, IEEE, 2000

Derridder, D., Wouters, B. The use of an Ontology to Support a Coupling between Software Models and Implementation. (2000): *Proc. European Conference on Object-Oriented Programming (ECOOP'00), International Workshop on Model Engineering.*

Dameron, O., Noy, N.F., Knublauch, H. and Musen, M.A. Accessing and Manipulating Ontologies Using Web Services. (2004): *Proc. Workshop on Semantic Web Services: Preparing to Meet the World of Business Applications at the Third International Conference on the Semantic Web (ISWC-2004),* Hiroshima, Japan.

Djuric, D., Gasevic, D. and Devedzic, V. Ontology Modeling and MDA. (2005): *Journal of Object Technology,* 4(1):109-128.

Devedzic, V. Understanding Ontological Engineering. (2002): *Communications of the ACM,* 45(4): 136-144.

Duineveld, A.J., Stoter, R., Weiden, M.R., Kenepa, B. and Benjamins, V.R. (2000): WonderTools? A comparative study of ontological engineering tools. *International Journal of Human-Computer Studies,* 52:1111-1133.

Eklund, P., Roberts, N., Gree, S. OntoRama: Browsing RDF Ontologies using a Hyperbolic-style Browser. (2002): *Proc. First International Symposium on Cyber Worlds (CW.02),* IEEE Computer Society.

Ells, R. Hypermedia, Help and How-To. (1998): *Proc. 16th Annual ACM SIGUCCS Conference on User Services*, California, United States, 363-368.

Evernden, R. The Information Framework. (1996): *IBM Systems Journal*, 35(1): 37-68.

Fensel, D. (2001): *Ontologies: A Silver Bullet for Knowledge Management and Electronic Management.* Berlin, Springer.

Farquhar, A., Fikes, R. and Rice, J. (1997): The Ontolingua Server: A Tool for Collaborative Ontology Construction. *International Journal of Human-Computer Studies,* 46:707-728.

Foundation for Intelligent Physical Agents (FIPA), FIPA Ontology Service Specification 2001, http://www.fipa.org/specs/fipa00086/index.html. Accessed 23 July 2006.

Gruber T.R. (1993): A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition,* 5(2): 199-220.

Gasevic, D.V., Djuric, D.O. and Devedzic, V.B. (2005): Bridging MDA and OWL Ontologies. *Journal of Web Engineering,* 4(2): 18-143.

Gruber, T.R. (1995): Towards principles for the design of ontologies used for knowledge sharing. *International Journal of Human-Computer Studies,* 43(5/6).

Guarino, N. (1998): Formal Ontology in Informations Systems. *Proc. 1st International Conference on Formal Ontology in Information Systems (FOIS'98),* Trento, Italy.

Guarino, N., Masolo, C., Vetere, G. (1999): OntoSeek: Content-Based Access to the Web. *IEEE Intelligent Systems,* 70-80.

Guarino, N. (2002): Ontology-Driven Conceptual Modelling. *Tutorial at 21st International Conference on Conceptual Modeling (ER'02),* Tempere, Finland.

Gomez-Perez, A., Benjamin, R. (1999): Overview of Knowledge Sharing and Reuse Components: Ontologies and Problem-Solving Methods. *Proc. IJCAI-99 Workshop on Ontologies and Problem-Solving Methods (KRR5),* Morgan-Kaufmann.

Hart, L., Emery, P., Colomb, R. M., Raymond, K., Chang, D., Ye, Y., Kendall, E. and Dutra, M. (2004): Usage Scenarios and Goals for Ontology Definition Metamodel. *Lecture Notes in Computer Science,* 3306:596. Springer.

Harrison, R and Chan, C.W. (2005). Distributed Ontology Management System. *Proc. 18th Annual Canadian Conference on Electrical and Computer Engineering, (CCECE'05),* Saskatoon, IEEE.

Inmon, W.H., Zachman, J.A., Geiger, J.G. (1997): *Data Stores, Data Warehousing, and the Zachman*

Framework: Managing Enterprise Knowledge. USA, McGraw-Hill.

Jasper, R. and Uschold, M. (1999): A Framework For Understanding and Classifying Ontology Applications. *Proc. IJCAI-99 Workshop on Ontologies and Problem-Solving Method (KRR5),* Stockholm, Sweeden.

Jarrar, M and Meersman, R. (2002): Formal Ontology Engineering in the DOGMA Approach. *Proc. ODBASE'02,* 2519:1238-1254, LNCS.

Jones, S. (2005): Toward an Acceptable Definition of Service, *IEEE Software*,22(3): 87-93.

Karp, P.D., Chaundhri, V.K and Thomere, J. (1999): XOL: An XML-based Ontology Exchange Language. Version 0.3.

Karp, P.D., Riley, M., Saier, M., Paulsen, I.T., Paley, S., Pellegrini-Toole, A. (2002): The Ecocyc Database. *Nucleic Acids Research*, 30(1):56.

Klyne, G., Carroll, J.J and McBride, B. RDF Resource Description Framework Reference. W3C Recommendation, http://www.w3.org/RDF/. Accessed 23 July 2006

Kogut, P., Cranfield, S., Hart, L., Dutra, M., Baclawski, K., Kokar, M. and Smith, J. (2002): UML for Ontology Development. *The Knowledge Engineering Review*, 17(1): 61-64

Lenat, D.B. (1995): CYC: A Large-scale investment in knowledge infrastructure. *Communication of ACM*, 38(11):33-38.

Li, Y., Thompson, S., Tan, Z., Giles, N. and Gharib, H. (2003): Beyond Ontology Construction: Ontology Services as Online Knowledge Sharing Communities. *Lecture Notes in Computer Science*, 2870: 469 – 483, Springer-Verlag.

Lopez, M.F., Gomez-Perez, A., Sierra, J.P. and Sierra, A.P. (1999): Building a Chemical Ontology using Methontology and the Ontology Design Environment. *IEEE Intelligent Systems*, 14(1): 37-46.

Mika, P. Social Networks and the Semantic Web. (2004). *Proc. International Conference on Web Intelligence (WI'04),* Beijing, China, 285-291, IEEE Computer Society.

Mizoguchi, R. (2003): Tutorial on Ontological Engineering: Part 2: Ontology Development, Tools and Languages. *New Generation Comput.* 22(1).

Mauger, B. (2005): Tools to Help an Agent Commit to an Ontology. Undergraduate Thesis. Computer Science Department, University of Queensland.

Motta, E., Fensel, D., Gaspari, M. and Benjamins, R. (1996): Specifications of Knowledge Components for Reuse. *Proc. 11th International Conference on Software Engineering and Knowledge Engineering,* Kaiserslautern, Germany, 36-43, KSI Press.

Maedche, A., Motik, B. and Stojanovic, L. (2003): Managing Multiple and Distributed Ontologies on the Semantic Web. *The VLDB Journal*, 12(4):286-302.

Noy, N.F. and Fergerson, R.W. and Musen, M.A. (2000): The knowledge model of protege-2000: combining interoperability and flexibility. *Proc. 12th International Conference in Knowledge Engineering and Knowledge Management (EKAW_00),* Lecture Notes in Artificial Intelligence, 1937:17–32. Springer.

O'Rourke, C., Fishman, N., Selkow, W. (2003): *Enterprise Architecture Using the Zachman Framework.* USA, Course Technology.

Oberle, D., Volz, R., Motik, B. and Staab, S. (2002): *KAON Server Prototype.* Technical Report, IST Project 2001-33052 WonderWeb, Department of Computer Science, The Victoria University of Manchester.

Pan, J., Cranefield, S. and Carter, D. (2003): A Lightweight Ontology Repository, *Proc. AAMAS'03*, Melbourne, Australia. ACM Press.

Pretorius, A.J. (2005): Visual Analysis for Ontology Engineering, *Journal of Visual Languages and Computing*, 16:359-381.

Rothenburger, B. and Galarreta, D. (2006): Facing Knowledge Evolution in Space Project: a Multi-Viewpoint Approach. *Journal of Knowledge Management*, 10(2): 52-65

Sure, Y., Erdmann, M., Angele, J., Staab, S., Studer, R. and Wenke, D. (2002): OntoEdit: Collaborative Ontology Engineering for the Semantic Web. *Proc. 1st International Semantic Web Conference*, Sardinia, Italy, Lecture Notes in Computer Science (LNCS 2342), Springer.

Swartout, B., Patil, R., Knight, K. and Russ, T. (1996). Towards Distributed use of large-scale ontologies. *Proc. 10th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop*, Banff, Canada.

Star Lab. V.U.B. Ontology Server Research. http://www.starlab.vub.ac.be/research/dogma/OntologyServer.htm . Accessed 23 July 2006.

Uschold, M. and Gruninger, M. (1996): Ontologies: principles, methods and applications. *The Knowledge Engineering Review*, 11(2).

Valo, A. and Hyvönen, A and Komulainen, V. (2005): Tool for Collaborative Ontology Development for the Semantic Web. *Proc. International Conference on Dublin Core and Metadata Applications (DC 2005)..*

Welty, C. and Ferrucci, D. (1999): A Formal Ontology for Reuse of Software Architecture Documents. *Proc. International Conference on Automated Software Engineering*, 259-262, IEEE Computer Society Press.

Zachman, J.A. (1987). A Framework for Information Systems Architecture. *IBM Systems Journal*, 26(3):276-292.

(Appendix) Table 1. A Comparison of Ontology Servers

	Ontolingua	ACOS	Starlab	KAON	OntoRama	OWS	FIPA Ontology Service Specification	Adapted Ontology Server
General								
Main goal	Collaborative ontology development	Collaborative ontology development	Ontology development (i.e. gathering and incremental growth)	Ontology development	Storing, browsing ontology. Limited support to ontology development	Accessing and manipulating ontologies	Ontology development and ontology-based agent Communication	Ontology development specifically in electronic commerce
Developers	KSL	ISLab	STARLab	AIFB	ITEE and DSTC	SMI	FIPA	SECE
Release	Free web access	Not mentioned	Free web access	Open source	Free web access	No	Updated in 2001	No
Main phase of ontology life cycle	Design time	Design time	Design time	Design time	Design time	Runtime	Design time, Runtime	Design time
Main deliverables	Server functions mainly for supporting ontology development	Server functions mainly for supporting ontology development	Server functions mainly for supporting ontology development	Server functions mainly for supporting ontology development	Server functions mainly for browsing ontology	Server functions mainly for accessing and manipulating ontologies	Server functions mainly for ontology development and agent communication	Server functions mainly for supporting ontology development
Management provision	Central	Community	Central	Central	Central	Central – (Tasks)	N/A	Central
Collaborative (i.e. building, access and manipulating, use)	Yes	Yes	Not Mentioned	Yes	No	Yes	Not mentioned	Not mentioned
Overall remarks	Lack of extensibility, strictly oriented to research activities.	Lack of extensibility, strictly oriented to research activities.	Lack of extensibility, strictly oriented to research activities.	To support integrated environment and much more ambitious.	Lack of extensibility, strictly oriented to research activities.	Lack of extensibility, strictly oriented to research activities.	Provide standards ontology service that used as guidance in specifying ontology server functions.	Strictly oriented to research activities and do not support toward semantic web vision.
Status	Rolled out in 1995	Rolled out in 2002	Being Developed	Rolled out in 2001	Rolled out in 2002	Being Developed	Specification 2001	Rolled out in 2000
Ontology								
Main ontology engineering challenges	Reusability	Reusability	Reusability	Reusability	Viewing	Usability	Reusability	Extracting, Reusability
Type of ontologies	Heavyweight	Lightweight	Lightweight	Heavyweight	Lightweight	Heavyweight	Heavyweight, Lightweight	Lightweight
Knowledge representation	Complex	Simple	Medium	Complex	Medium	Open (unspecified)	Complex	Medium
Ontology model (i.e. expressiveness	Allow maximal expressiveness	Allow minimal expressiveness	Allow minimal expressiveness	Allow maximal expressiveness	Not mentioned	Not mentioned	Open (unspecified)	Not mentioned
Ontology languages	Ontolingua and KIF	DAML+OIL	XML	RDF	RDF	Open (unspecified)	Open (unspecified)	Not mentioned

Features								
Import mechanism	Yes	Yes	No	Yes	No	No	Yes	No
Manner of consistency checking	Human Approval, Simple Data Type Checking	Auto	Not mentioned	Yes	Not mentioned	Semi-auto	Open (unspecified)	Human
User access control	Yes	Yes	No	Yes	No	Yes	N/A	Not mentioned
Ontology repository platform	File-Based	File-based	DBMS	File-based + DBMS	File-Based	Open (unspecified)	Open (unspecified)	DBMS
Software architecture (i.e. client-server, n-tier application, etc)	Client-server	Client-server	Client-server	Client-server	Client-server	Client-server	Client-server	Client-server
Extensibility (i.e. plug-in features, etc)	No	No	No	Yes	No	No	Not Mentioned	No
Implementation								
Server functionality implementation	Tool development	APIs	Database APIs + Tool development	APIs + Tool development	Tool development	Semantic web services	Tool development	APIs
Server functionality platform	Not Mentioned	Java	Java	Java	Java	Java	Open (unspecified)	Java
User / Application remote support to ontologies (i.e. protocol used)	HTTP /OKBC	HTTP/ JDBC-ODBC	JDBC-ODBC	HTTP/JDBC-ODBC	HTTP/ JDBC-ODBC	HTTP/ JDBC-ODBC	HTTP/ OKBC	HTTP/JDBC-ODBC
Ontology repository technology	Ontolingua-based repository	Jena + DAML + OIL	MSQL Server	RDF + RDBMS	Jena + RDF	Open (unspecified)	Open (unspecified)	MySQL
Methodology								
Ontology-related methodology	No	No	Ontology double articulation and ontology modularization	No	No	No	METHONTOLOGY	No
Server development methodology	No	No	No	No	No	No	No	No
Benefit from graphical language	No	No	Yes – ORM for ontology modeling	No	Yes – Hyperbolic style for browsing ontologies	No	No	No
Server design profiles	No	No	No	No	No	No	No	No

Beyond Purpose-Based Privacy Access Control

Sabah S. Al-Fedaghi
Computer Engineering Department
Kuwait University
PO Box 5969 Safat 13060 Kuwait
sabah@eng.kuniv.edu.kw

Abstract

Research efforts have been directed toward the improvement of privacy protecting technology by incorporating privacy protection into database systems. Purpose acts as a central concept on which access decisions are made. A complexity of purpose and users role hierarchies is utilized to manage the mapping between users and purposes. In this paper, we propose a personal information flow model that specifies a limited number of acts on this type of information. Chains of these acts can be used instead of the "intended/business purposes" used in privacy access control.

Keywords: personal information, privacy, database system.

1 Introduction

Privacy is becoming an important feature in modern society. The rapid advances in information technology and the emergence of privacy-invasive technologies have made informational privacy a critical area to be protected. Personal information is used in making decisions about an individual's life. Regulations and laws have been established to allow people to control the way in which their personal information is used. Guidelines such as the 1980 OECD guidelines, legislation such as the Health Insurance Portability and Accountability Act (HIPAA) and systems such as P3P are not sufficient to safeguard privacy because "they do not address how personal data is actually handled after it is collected … Privacy protection can only be achieved by enforcing privacy policies within an organization's online and offline data processing systems" (He et al., 2003). Also, "privacy cannot be efficiently implemented solely by legislative means. Data protection commissioners are therefore demanding that legal privacy requirements should be technically enforced and should be a design criteria for information systems." (Fischer-Hübner and Ott, 1998)

Privacy-enhancing technology aims at making privacy protection guidelines and laws an integrated part of the technology. Thus, an information system is designed to embed components that allow monitoring compliance of the system to privacy rules.

The notion of *Purpose* is the basic concept on which decisions to access personal information are made. A complexity of purpose hierarchies and users' role hierarchies are utilized to manage the mapping between users and purposes. We propose a personal information flow model that specifies acts on this type of information. Chains of these acts can be used to control acting on personal information instead of purposes used in privacy access control. The method is distinguished by the limited number of acts that form chains of acts on personal information.

2 Related Works

The Platform for Privacy Preferences (P3P) provides means for policy privacy specification and exchange but "does not provide any mechanism to ensure that these promises are consistent with the internal data processing." (Byun et al., 2005) Hippocratic databases have been introduced as systems that integrate privacy protection within relational database systems (Agrawal et al., 2002). A Hippocratic database includes privacy policies and authorizations that associate with each attribute and each user the usage purpose(s).

Privacy protecting access control deals with privacy policy specification and private data management systems. In privacy protecting access control models, "the notion of purpose plays a central role as the purpose is the basic concept on which access decisions are made" (Byun et al., 2005). "Most privacy-aware technologies use purpose as a central concept around which privacy protection is built." (Massacci1 et al., 2005)

Example: (from Byun et al. (2005)) Sample purposes defined on a given hierarchy of purposes are:

Allowable purposes = {Admin, Profiling, Analysis} ∪ {Direct, D-Email, D-Phone, Special-Offers, Service-Updates}

Prohibited purposes = {D-Email, Special-Offers, Service-Updates} U {D-Email, Direct, Marketing, General-Purpose}

An access is granted if the access purpose (stated by user) is entailed by the allowed purposes and not entailed by the prohibited purposes (Byun et al., 2005).

Users role hierarchies similar to ones used in security policies (e.g., RBAC: Role Based Access Control mechanisms (Sandhu et al., 2000)) are used to simplify the management of the mapping between users and purposes. A request to access to data is accompanied by access purpose, and accessing permission is determined after comparing such a purpose with the intended purposes of that data in privacy policies. Each user has authorizations for a set of access purposes.

Nevertheless, the management of attributes and users purposes is a complicated issue. Attributes and users purposes form hierarchies of the generalization and specialization (email marketing and telephone marketing is generalized as marketing) that are organized according to users' roles. To simplify the mapping process users are assigned roles, and access purpose permissions are granted to roles associated with tasks or functionalities, not directly to individual users (Byun et al., 2005). To support dynamic changes in purposes, "role" may be assigned attributes with hierarchy inheritance characteristics "to assign an access purpose to a specific subset of users in the same role" (Byun et al., 2005).

Purpose management introduces a great deal of complexity at the access control level. In this paper we introduce an alternative privacy access control mechanism that is not based on purpose.

3 Purpose

The notion of *purpose* appears in all privacy codes and legislations. For example, the Data Quality Principle in the OECD guidelines specifies:

> Personal data should be *relevant* to the *purposes* for which they are to be used, and, to the extent necessary for those purposes, should be accurate, complete and kept up-to-date. (OECD, 1980)

Purposes are usually categorized into two types:

Consumer data purpose: This purpose expresses how the collected data can be used. P3P (2002) specified purposes include: Web Site and System Administration, Research and Development, Individual Decision, Contacting Visitors for Marketing of Services or Products, Historical Preservation, Telephone Marketing, and profiling.

Business purpose: This purpose is for business actions that involve certain consumer data operations.

Customer purposes are typically mapped to the more specific business purposes. For example, telemarketing is mapped to direct telemarketing and third party telemarketing and these in turn may be categorized into email telemarketing and telephone telemarketing such that the relationship among purposes modelled as a purpose hierarchy.

"A purpose specifies the intended use of the data element" and it "describes the reason(s) for data collection and data access" (Byun et al., 2005). Privacy policies utilize purposes in their accessing Personal Information (PI) mechanism.

Nevertheless, *purpose* remains a semantic concept that is "pasted" superficially to personal information. To utilize it in privacy protecting access control, we identify the relationship between purposes in order to build a hierarchy of purposes that tells us such things as which purpose-based access control is embedded into a more general purpose control. This method is basically a privacy-covered version of a security access control mechanism. For the system, *purposes* could be described as "level 1," "level 2," … instead of as Administration, Profiling, and Analysis, and it would not make any difference.

Instead, we propose to define the intended purpose of personal information as a chain of acts on this type of information. For example, the purpose could be:

Collecting (act 1) personal information, processing (act 2) it, in order to create (act 3) new information (e.g., *John is a risk*), to be used (act 4) in deciding a loan.

We claim that the number of types of acts on personal information is limited. In this case a chain of acts are permitted and other chains are prohibited. The chain represents a syntactical form of controlling access (acts in general) to personal information.

4 Personal Information

This section reviews the definition of personal information (PI) and its flow model given in Al-Fedaghi (2006a) and Al-Fedaghi (2006b).

4.1 Definition of personal information:

Personal information theory assumes two fundamental types of entities: *Individuals* and *Non-individuals* (Al-Fedaghi, 2005a and 2005b). The term *Individuals* represents the set of natural persons and *Non-individuals* represents the set of non-persons. *Personal information* (PI) is any linguistic expression that has referent(s) in *Individuals*. Assuming that p is a sentence such that X is the set of its *referents*, then there are two types of PI:

(1) p is the atomic personal information if $X \cap$ *Individuals* is the singleton set $\{X\}$. That is, atomic personal information is an assertion that has a single human referent (e.g., *John is 25 years old*). "Referent," here, implies an identifiable (natural) person.

(2) p is the compound personal information if $X \cap$ *Individuals* is a set of more than one person (e.g., *John loves Mary*). That is, compound personal information is an expression that has more than one human referent.

The relationship between individuals and their own atomic personal information is called *proprietorship*. If p is a piece of atomic personal information of $v \in$ *Individuals*, then p is proprietary personal information of v, and v is its *proprietor*.

A single piece of atomic personal information may have many possessors; where its proprietor may or may not be among them. A *possessor* refers to any entity that knows, stores, or owns the information. Any compound personal statement is privacy-reducible to a set of atomic personal statements.

Personal information privacy involves acts on personal information in the context of creating, collecting, processing, and disclosing this type of information.

4.2 Personal Information Flow Model:

The personal information flow model divides functionality into four modules or phases that include informational privacy entities and processes, as shown in Figure 1.

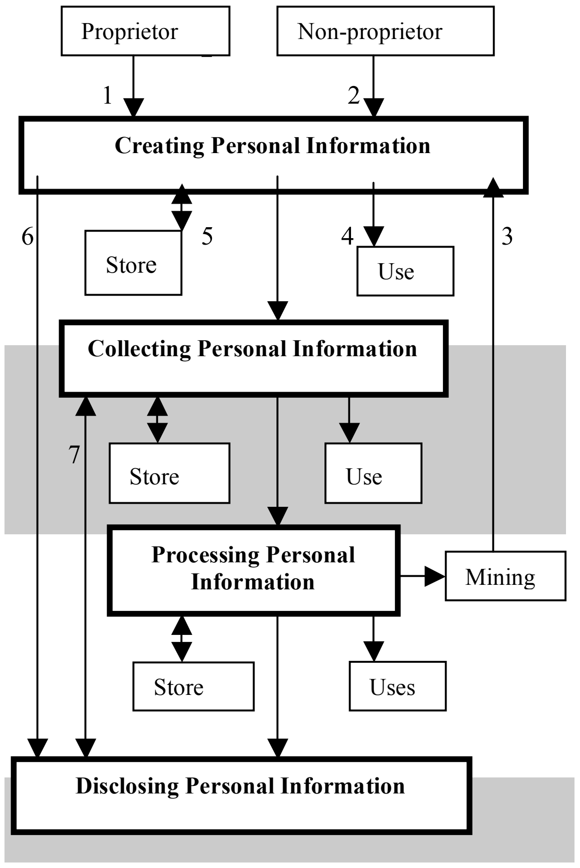

Proprietor
Non-proprietor
1
2
Creating Personal Information
6
5
4
3
Store
Use
Collecting Personal Information
7
Store
Use
Processing Personal Information
Mining
Store
Uses
Disclosing Personal Information

Figure 1: Personal information flow model.

New PI is created at Points 1, 2, and 3 by proprietors or non-proprietors (e.g., medical diagnostics by physicians), or is deduced by someone (e.g., data mining that generates new information from existing information). The created information is used either at Point 4 (e.g., decision making), Point 5 (stored), or Point 6, where it is immediately disclosed. Processing the personal information phase involves acting (e.g., anonymization, data mining, summarizing, translating) on PI. The disclosure phase involves releasing PI to insiders or outsiders. The "disposal" or disappearance of PI can happen anywhere in the model, such as in the transformation to an anonymous form in the processing phase. "Store" in Figure 1 denotes both storing and retrieving operations.

Example: Consider the situation where we have one proprietor and two PI agents (e.g., companies, departments, agencies, other individuals). Suppose that the roles of these three actors are defined as follows:

Proprietor: *Creates*, *stores*, and *discloses* PI to Agent 1.

Agent 1: *Collects*, *stores*, *uses*, and *discloses* PI to Agent 2.

Agent 2: *Collects* and *processes* PI through a *mining* technique that *creates* new PI that is *stored* and *used* in some applications (e.g., decision making). The PI flow model for this simple environment can be drawn based on Figure 1, as shown in Figure 2.

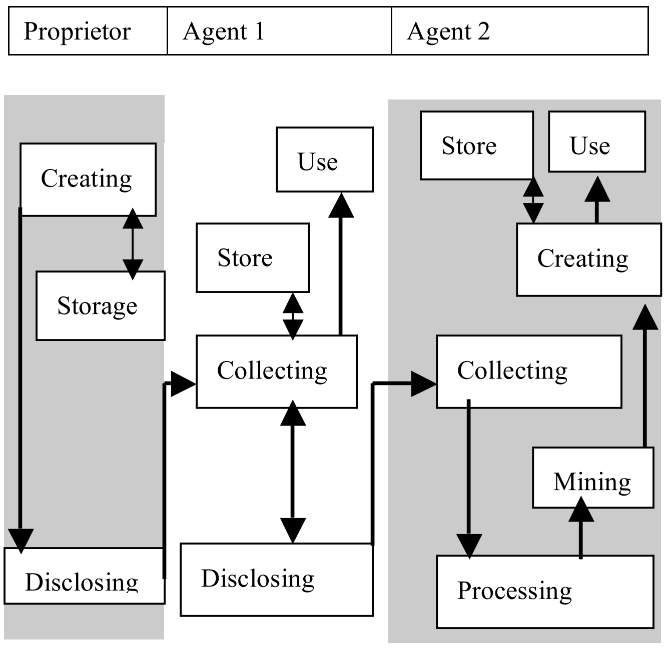


Figure 2: A proprietor and two agents model.

For each actor in this scenario we will make a copy of the PI flow model. However, because the proprietor does not collect or process PI, these phases are not shown in his/her region. The creation and processing phases are not shown for Agent 1 because it does not create or process PI. Similarly, the disclosure phase is not shown in the region of Agent 2. Let t be a piece of PI of the proprietor. It originates in the *Creating* box in the proprietor's region. It is stored in *Store*, and moves through the *Disclosing* box to the *Collecting* box of Agent 1's region. It is stored and used there, and moves through the *Disclosing* box to the *Collection* phase of Agent 2. There, it moves to the *Processing* to the *Mining* boxes where it generates new PI in the *Creation* phase for storage and later use. We can add details to any phase as the situation requires. For example, Agent 2 may add *Store* to keep a copy of the original PI or a *Disclosure* phase can be added if Agent 2 discloses the resultant new PI to a third agent.

5 The Proprietor-Others Architecture

This section and Section 6 review with some additional materials the architecture given in Al-Fedaghi (2006a).

Using the PI flow model, we can build a system that involves a proprietor on one side and others (other persons, agencies, companies, etc.) who perform different types of activities in the PI transformations among the four phases of flow of personal information. We will refer to any of these as PI agents. PI agents may include any one who participates in activities over PI. The proprietor is not accepted as agent with respect to his/her own PI.

Figure 3 illustrates this type of mode of operation on PI with sample PI agents. The solid arrows reflect that the proprietor is a source of PI for agents. The dotted lines reflect that this PI of the proprietor may also be directly shared and exchanged among agents.

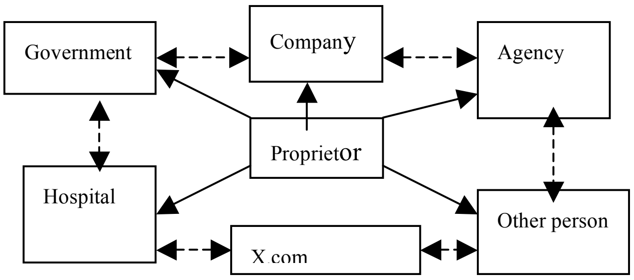


Figure 3. Proprietor/others PI exchange scheme.

The EU Privacy Directive manages this type of system: organizing (1) the relationship between the proprietor and agents that utilize his/her personal information and (2) the relationship among the agents.

According to the EU Directive (1995):

> (25) Whereas the principles of protection must be reflected, ***on the one hand***, in the obligations imposed on persons, public authorities, enterprises, agencies or other bodies responsible for processing, in particular regarding data quality, technical security, notification to the supervisory authority, and the circumstances under which processing can be carried out, and, ***on the other hand***, in the right conferred on individuals, the data on whom are the subject of processing, to be informed that processing is taking place, to consult the data, to request corrections and even to object to processing in certain circumstances" (Italics added).

Notice that this type of system is basically a "binary" system that involves the proprietor on one side and all agents, represented in the EU Privacy Directive by the "controller" on the other.

6 The Acts on the Proprietor's PI

As a result, we need two *types* of PI flow models: one for proprietors and one for agents. We construct this proprietor/agent PI flow architecture with two regions: the proprietor's region of activities on his/her PI and the others' region of activities on the proprietor's PI, as shown in Figure 4. Notice that we concentrate in the figure on the agent's region. This region is a copy of the personal information flow model.
We assume that there is no interest in how a proprietor collect and process his own PI. For example, the EU Directive specifies that "there should be excluded the processing of data carried out by a natural person in the exercise of activities which are exclusively personal or domestic, such as correspondence and the holding of records of addresses" (EU Directive, 12).

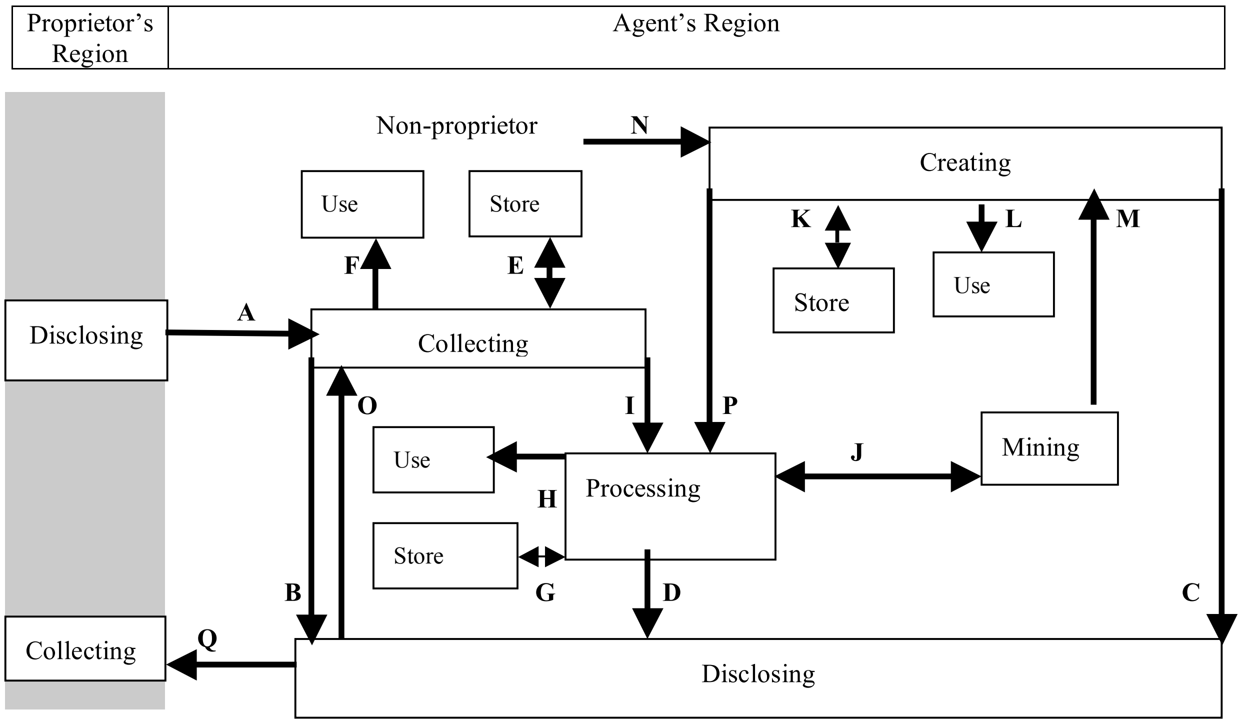


Figure 4. Architecture of Proprietor/Agent PI flow

We distinguish 17 types of acts on PI (labelled A through O) as shown in Figure 4 and described in Table 1. These acts form ordered sequences or chains as will be discussed later.

Acts	Descriptions	Comments
A	Disclosing PI by a proprietor	Act A also represents collecting PI (by a collecting agent).
B	Disclosing PI by a collecting agent	B implies O (collecting PI by another collecting agent)
C	Disclosing PI by a creating agent	In Figure 4, C implies B, that is, the disclosed PI flows from a collecting agent to another collecting agent.
D	Disclosing PI by a processing agent	In Figure 4, D implies B, that is, the disclosed PI flows from a processing agent to a collecting agent.
E	Storing PI by a collecting agent	E (double arrow in figure 4) includes retrieval. of PI.
F	Using PI by a collecting agent	"Using" indicate non-informational operations.
G	Storing PI by a processing agent	We can separate the storing and retrieval acts as two independent acts in the model.
H	Using PI by a processing agent	Using processed PI may be different from uses of other types of PI.
I	Processing PI by an agent	PI flows from the collecting phase to a processing phase, assuming the same agent.
J	Mining PI by a mining agent	Mining is a type of processing (notice the bi-directional arrow). This type of mining (back arrow) produces implied PI but not new PI.
K	Storing PI by a creating agent	
L	Using PI by a creating agent	
M	Creating PI by a mining agent	Automatic creation of PI. Notice that mining is a special type of processing.
N	Creating PI by a non-proprietor	Non-automatic creation of PI (e.g., gossip).
O	Collecting PI from non-proprietor	O occurs simultaneously with B: If an agent discloses PI then there is an agent that collects that PI.
P	Processing of created PI	Non-proprietor may creates PI and process it immediately without storing or acting on it.
Q	Disclosing to proprietor	E.g., Informing a person of results of medical tests.

Table 1. Acts on Personal Information

Examples:

Act A: A person discloses PI to a hospital.

Act B: A hospital discloses PI to an insurance company.

Act C: PI is produced by a mining program (e.g., *John is a high-risk customer*) of a bank is disclosed to crediting company.

Act D: Processed PI (changing the original data to another form through such operations as modification, translation, summarization, generalization) is released from a hospital to an insurance company.

Act E: A hospital stores PI in its automated or manual system, accessing stored data.

Act F: A hospital uses PI to contact its patients.

Act G: A hospital stores processed PI (e.g., mined PI) in its system .

Act H: An agent uses processed data for research.

Act J: An agent mines PI to extract implied PI (e.g., the information *Z is the grandfather of Y* is taken from *Z is the father of W* and *W is the father of Y*).

Act K: An agent stores PI produced by a mining program.

Act L: An agent makes decisions based on PI that is produced from a mining program.

Act M: An agent produces new PI using a mining program (e.g., an analysis of customers produces the information that *John is a high-risk person*).

Act N: A newspaper writer publishes PI.

Act O: An agent collects PI from another agent.

Act P: An agent creates PI (gossip) and processes it.

The advantage of this categorizing of "processing" of personal data is that we can specify different types of rules, requirements, restrictions for each type of act on PI (Al-Fedaghi, 2006a). The next section proposes to use the chains of acts on PI as a mechanism that is tied to the user purpose.

7 Chains of Acts on Personal Information

We have now a foundation for developing a purpose-less privacy protecting access control mechanism. We assume that each purpose can be translated to a chain of acts on PI. To simplify the discussion, the concept is introduced through known examples in the research literature.

Chains of acts on PI are chains of *information handling* that starts with one of the following acts:

Act A: A proprietor discloses his/her PI. This act can also be described as an agent collecting PI from a proprietor.

Act O: An agent collects PI from another agent. In this case O may be preceded by the act of disclosing agent to indicate where the PI coming from.

Act N: A non-proprietor creates PI.

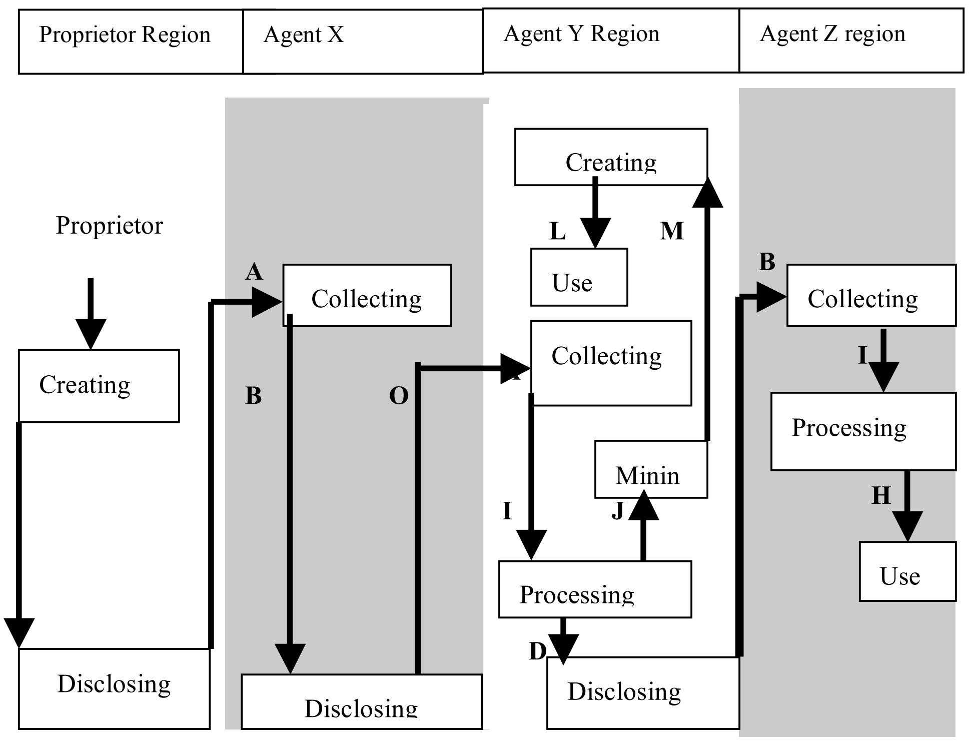


Figure 5. The architecture of the flow of PI for the given example.

These three acts, A, O, and N, are the only sources that supply any agent with PI. Suppose that a company has a piece of personal information. This piece of information is either collected from its proprietor, from another agent, or created internally by the agent. Starting with any of these sources, that piece of PI flows into the PI information handling system (manual or automatic) subjected to different acts such as processing, utilization, mining and so forth. This track of acts can be traced through *chains*.

One benefit of these chains is designing the PI handling system such that each piece of PI is constrained to flow in specific chains.

Example: Suppose that a company collects PI from proprietors and utilizes it in direct email telemarketing. The chain of acts that describes the flow of PI in this operation is AF (the sequence starts with act A then act F) which includes:

Act A: Collecting PI from a proprietor

Act F: Using PI in direct telemarketing operation

If the company also stores the PI (act E: Storing and retrieving) then we have two possibilities:

(1) PI is collected, stored, then used in telemarketing. The chain AEF represents this sequence of acts.

(2) The PI is collected and utilized directly in telemarketing, and in parallel stored and utilized later in telemarketing. This is represented by the chains AF and AEF. AF and AEF are two paths of flow of information, one is directly from collecting to telemarketing and the other goes from *Collecting* to *Store* to *Use*.

Suppose that the company also use the PI for third party marketing (i.e., it sells the PI to another telemarketing company). In this case, samples of possible chains:**(1)**

AB: Disclosing PI straight to the other agent without using it itself in telemarketing.

(2) Two chains, AB and AF: Disclosing PI to the other agent and using it in its own telemarketing. A sample situation that reflects AB is the following: a customer requests PI about a certain person from a private investigation agency under the condition that the agency does not keep records of PI after delivering it to the requester.

(3) Two chains, AEF and AEB: Storing PI and using it, and also disclosing PI to the other agent.

These types of acts on PI represent constraints on methods of handling PI. The "enterprise purpose" is represented by a set of these chains.

Example: Consider a proprietor, X, and two agents, Y and Z. Assume that the flow of information reflects the following *roles* for agents:

Proprietor: discloses his/her PI to collecting agent Y

Agent X: collects PI of X and discloses it to Agent Z.

Agent Y: collects PI from Agent X and processes it through a data mining program that creates new PI which is used in decision-making (e.g., denying a loan).

Agent Z: Collects PI from Y and processes it to use the results in research.

Figure 5 shows the architecture of the flow of PI in such a system.

	Possible chain of acts	Examples	Explanations
1	A	Collecting PI to satisfy curiosity without storing, using, or disclosing it.	The purpose of act A alone without being followed by storing, using, or disclosing the PI is a "mere collecting" act.
2	A then B	Selling PI to agent Y	Collected PI is immediately disclosed to another agent.
3	A then E then B	Selling PI to agent Y while keeping a copy of it	As when a telemarketer uses PI and also sells it to others.
4	A then E	Just storing collected information.	As in archival storage where data *may* never be needed.
5	A then F	Direct use of PI without storing it	
6	A then E then F	Storing PI then using it	
	B…	The flow of PI starts from an origin that creates it. These chains are illegal because they should start with A (PI is created by its proprietor), M (PI is manually created by a non-proprietor), or N (PI is automatically created by a non-proprietor).	
	E…		
	F…		

Table 2. Six possible chains that can be assigned to X.

Chains of X: Table 2 shows six possible chains that can be assigned to X. Notice that a sequence such as AFB is not mentioned because it means that the agent collects PI then used (act F), and disclosed (act B) it. Acts F and B are not dependent (in the flow of PI) on each other, hence, we can write them in terms of the two basic sequences: AF and AB.

In our example, by assumption, Agent X does not merely collect PI, neither *store* nor *use*. The only chain used by X is AB. We can observe here that such categorization of utilization of PI is based on purely syntactical consideration (the 17 acts and the relationships among them) and does not depend on any particular situation.

Chains of Y: Possible chains that are applied to Agent Y are: B, BI, BIJ, BIJM, BIJMN, and BID. However, according to our example Agent Y use only the chain BIJM.

Chains of Z: Possible chains that are applied to Agent Z are: B, BI, and BIH.

Suppose that all three agents are in the same enterprise. Then the enterprise system can organize the activities of these agents such that each is constrained to a certain chain. For example, X is not permitted to access PI produced by the mining program, Y is not permitted to disclose PI to X, Z is not permitted to disclose PI to anyone.

8 Privacy Access Control

We are proposing to use chains of acts on PI to control acting on PI instead of the so-called "business purposes" used in privacy access control.

Example: This example is a revised version of Byun et al. (2005). Suppose that we wish to allow three types of users:

E-Analysts are the users who analyze the customer information and prepare the contents of emails. They have the permissions to access the customer profiles.

Writers are the users who write and send out emails to customers. They have the permissions to access the email addresses of the customers.

Service-Update refers to workers who send out updated service information and then update the information.

In this scenario, we have three agents, the proprietor and the system as an agent. The *system* represents the total information system that the three agents are its users. The chains for each agent are as follows, where we specify only the relevant acts (qualified with "′" for the system):

System: B′, **O′G′**: These are the required acts from the system in the context of the example: B′ (disclosing PI) which refers to provide viewing (access) of PI to other agents; and O′G′ (collecting PI released by other agents and storing it).

E-Analysts: B′OIBO′G′: That is, collecting PI from the system (B′O), processing (altering) it (I), and disclosing it to the system (O′) to be stored back (G′).

Writers: B′OF: That is, collecting PI from the system (B′O) and use it to send out emails to customers. We assume that sending out emails to customers is one of the uses (Use box).

Service-Update: Includes two chains:

a) B′OF: That is, collecting PI from the system (B′O) and using it to communicate with the customers to update their PI.

(b) ABO′G′: That is, collecting the updated PI from customers (A) and disclosing (releasing) it to the system (BO′) to be stored back.

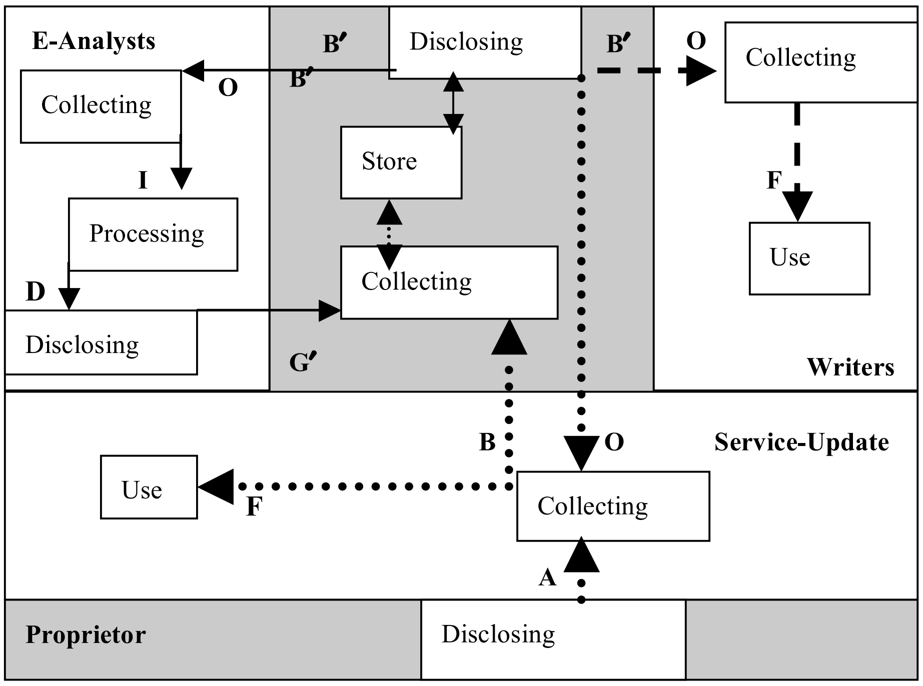


Figure 6. Chains of three agents, the proprietor and the system.

Figure 6 shows the PI flow models for the three agents, system, and proprietor. Each chain in this case is a constraint on its relevant PI: email of Writers, profile of E-Analysts, and the PI relevant to the Service-Update workers.

This methodology is an alternative method to *enterprise purpose* and roles hierarchies. Each PI is associated with a chain that reflects permissible acts to accomplish the *customer purpose*. If the purpose is *direct email telemarketing* then the chain OF is associated with the email information. The users (e.g., Writers in the example) in this case can only collect the email information and use it to send out emails to customers. We assume that there is a system application program that is designed to send out these emails to customers. The chain OF is implemented by programs that allow these users to access the email information. This mapping between chains and users is a replacement of the mapping between purposes hierarchies and users roles hierarchies.

9 Architecture for Personal Information Database System

The high level description of architecture for a personal information database system will be refers to as PIDB.

PIDB is formed from regions for proprietors and agents. Each region is a copy of the PI flow model as described previously.

Example: This example is a revised version of Massacci et al. (2005) who built it from the case study proposed by Agrawal et al. (2002).

Mississippi relies on Worldwide Express (WWEx) for shipping books. WWEx is a delivery company that offers a global network of specialized services – transportation, international trade support and supply chain services. WWEx also needs personal information to deliver books for Mississippi. This information includes customer name and shipping address. In turn, WWEx depends on local delivery companies for door-to-door delivery. To this end, WWEx delegates customer information to them. Furthermore, Mississippi relies on the Credit Card Company (CCC) for credit assessment. CCC needs to obtain some information for providing credit assessment. This information includes customer's name and credit card number, and the transaction between Mississippi and the customer. For making credit decisions, CCC wants a credit rating. For this, CCC depends on the Credit Rating Company (CRC). CRC uses statistics to summarize past experience so that predictive analysis can be used to generate a rating for the customer. Based on the rating, CCC can decide to accept or not the customer transaction.

This scenario includes one (type of) proprietor and six agents as illustrated in Figure 7. The figure shows the relationships between these acts. Mississippi has two internal users.

Figure 8 shows the architecture of PIDB for the actors Proprietor, Mississippi, Credit Card Company, Worldwide Express and Local delivery companies. The flow model of the internal departments can be specified in similar way to the agents E-Analysts, Writers, and Service-Update in the example in the previous section. For example, the chain from ordering books to delivering it to customer is described in the following chains:

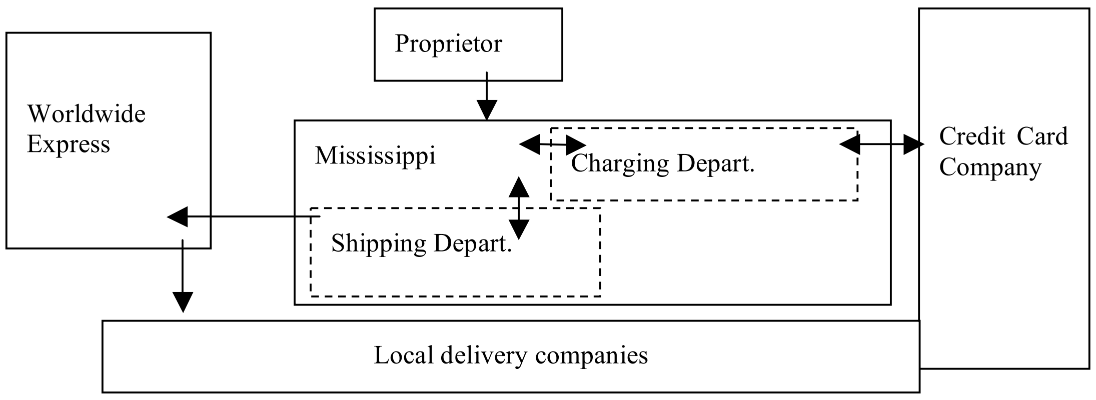


Figure 7. General view of the relationships among six acts in the Mississippi example.

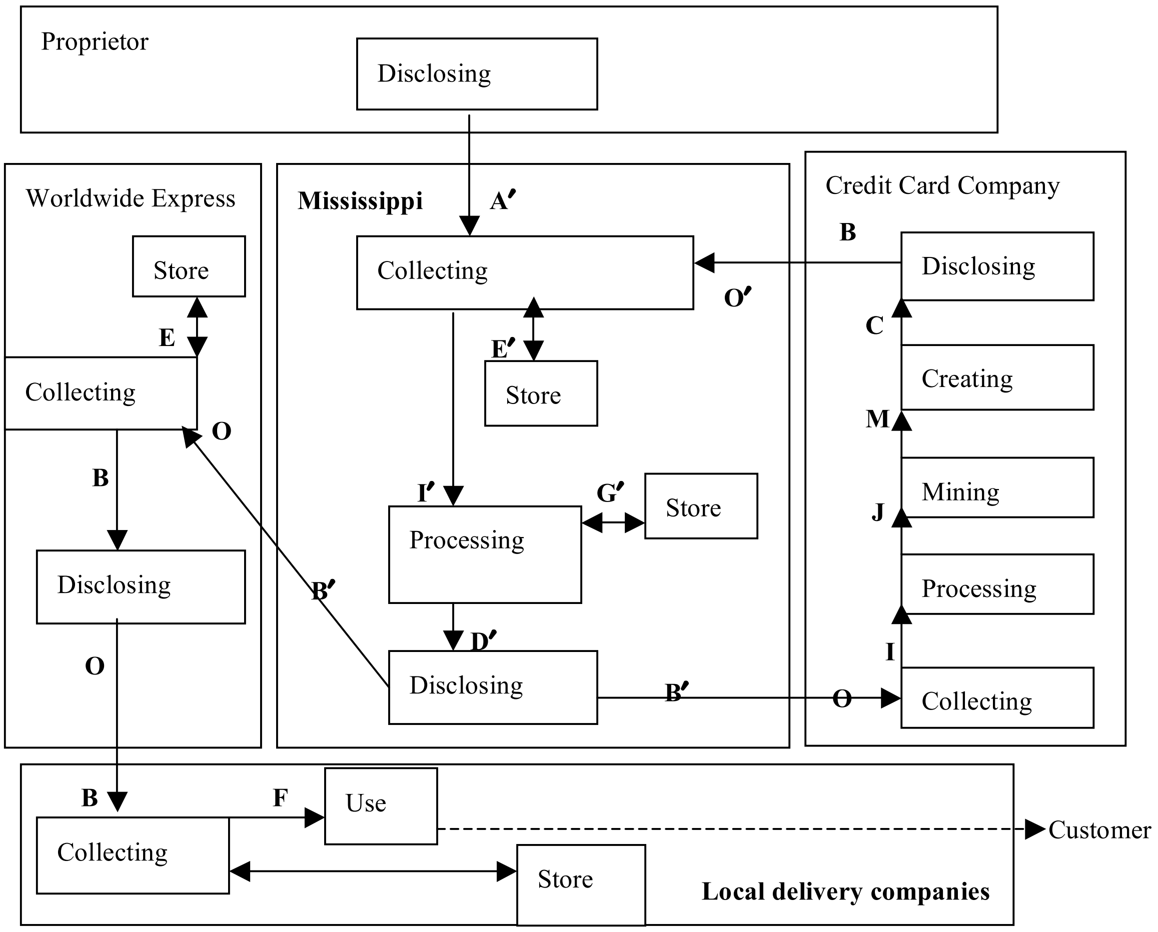


Figure 8. The architecture of PIDB for the Mississippi example.

Mississippi: A′ E′ I′ G′ D′ B′

Mississippi collects PI from customer, stores, process, stores (any results from processing), release Pi to be disclosed to the Credit Card Company.

Credit Card Company: OIJMCB

Credit Card Company collects PI (from Mississippi), process, mine, release (new PI, e.g., John's credit is OK), and disclose it to Mississippi

Mississippi: O′I′D′

Mississippi collects results, process, and send delivering order to Worldwide Express (Assuming, credit is OK).

Worldwide Express: OEBO

Worldwide Express collects PI, stores, and discloses it to Local delivery companies.

Local delivery companies: BEF

Local delivery companies collect PI, store and perform the actual act of delivering the purchase.

As we see that other chains can be added to this scenario. Privacy Constrains can be supper imposed on internal or cross-organizational handling of personal information. The method is distinguished by the limited number of acts that form well defined chains of acts on personal information. In comparison with the complexity of hierarchies of purposes and roles and the mapping between them this method promises an attractive alternative. Additionally, the method is a manifestation of the notion "privacy by design" that embeds privacy constrains inside the system.

Figure 9 shows all possible chains of acts. PI entry points are acts A, B, and N. It generates non-informational acts (e.g., decisions) at F, H, and L. The dotted line between B and O indicates simultaneous acts (disclosure and collection) of two different agents as described previously.

10 Conclusion

The concept of representing database system's privacy constraints as chains of acts on personal information has been shown to be a possible alternative method to database techniques based on purpose hierarchies A great deal of investigation is needed to formalize and experiment with such a mechanism. The general method has more applications than mere privacy protecting access control method. It can additionally be applied in writing privacy codes, guidelines, and statutes (Al-Fedaghi (2006c)).

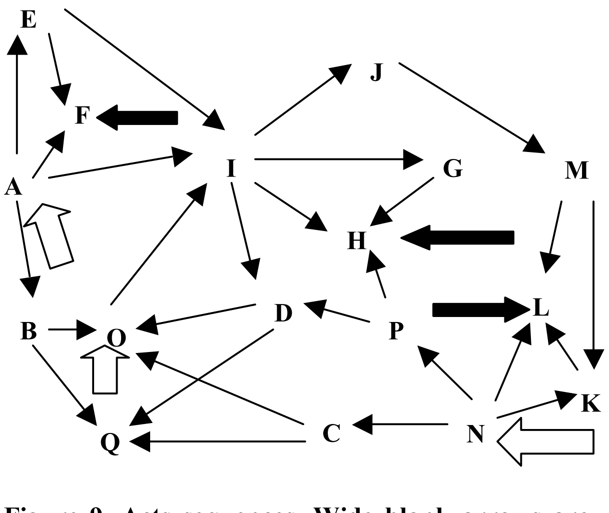


Figure 9. Acts sequences. Wide blank arrows are PI entry points and wide black arrows are uses where PI generates non-informational acts.

11 References

Agrawal, R. Kiernan, J. Srikant, R. and Xu. Y. (2002). Hippocratic databases. In The 28th International Conference on Very Large Databases (VLDB), Hong Kong, China, August.

Al-Fedaghi, S. (2006a) Anatomy of Personal Information Processing: Application to the EU Privacy Directive, International Conference on Business, Law and Technology (IBLT 2006), Copenhagen on December 5-7, 2006.

Al-Fedaghi, S. (2006b). Aspects of Personal Information Theory, 7th, The Seventh Annual IEEE Information Assurance Workshop (IEEE-IAW), West Point, NY: United States Military Academy, June 20-23.

Al-Fedaghi, S. (2006c). Personal Information Flow Model for P3P, *W3C Workshop on Languages for Privacy Policy Negotiation and Semantics-Driven Enforcement*, Ispra (Italy), 17-18 October 2006

Al-Fedaghi, S. (2005a). How to Calculate the Information Privacy, The Third Annual Conference on Privacy, Security and Trust, St. Andrews, New Brunswick, Canada.

Al-Fedaghi, S. Fiedler G. and B. Thalheim B. (2005). Privacy Enhanced Information Systems, Proceedings of The 15th European-Japanese Conference on Information Modeling And Knowledge Bases: Tallinn, Estonia, 2005.

Byun, J. Bertino, E. and Li, N. (2005). Purpose Based Access Control of Complex Data for Privacy Protection, SACMAT'05, June 1–3, 2005, Stockholm, Sweden.

EU Directive. (1995). DIRECTIVE 95/46/EC OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL, 24 October. http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:31995L0046:EN:HTML

Fischer-Hübner, S. and Ott, A. (1998). From a Formal Privacy Model to its Implementation, Proceedings of the 21st National Information Systems Security Conference, Arlington, VA, October 5-8, 1998.

He O. and Antón. A. I. (2003). A Framework for Modeling Privacy Requirements in Role Engineering, International Workshop on Requirements Engineering for Software Quality (REFSQ 2003), Klagenfurt / Velden, Austria, 16 - 17 June, 2003.

Massacci, F. and Mylopoulos, J. and Zannone, N. (2005) Minimal Disclosure in Hierarchical Hippocratic Databases with Delegation. 10TH EUROPEAN SYMPOSIUM ON RESEARCH IN COMPUTER SECURITY, Milan, Italy - September 12-14, 2005.

OECD (1980). Guidelines on the Protection of Privacy and Transborder Flows of Personal Data, http://www.oecd.org/document/18/0,2340,en_2649_34255_1815186_1_1_1_1,00.html

P3P (2002). The Platform for Privacy Preferences 1.0 (P3P1.0) Specification, The World Wide Web Consortium, April 16, 2002, http://www.w3.org/p3p/.

Sandhu, R. Ferraiolo, D. and Kuhn, R. (2000). The NIST model for role-based access control: Towards a unified standard. In the fifth ACM workshop on Role-based access control, July 26-27, 2000.

The privacy of k-NN retrieval for horizontal partitioned data — new methods and applications

Artak Amirbekyan **Vladimir Estivill-Castro**

School of ICT, Griffith University,
Meadowbrook QLD 4131, Australia
Email: A.Amirbekyan@gu.edu.au

Abstract

Recently, privacy issues have become important in clustering analysis, especially when data is horizontally partitioned over several parties. Associative queries are the core retrieval operation for many data mining algorithms, especially clustering and k-NN classification. The algorithms that efficiently support k-NN queries are of special interest. We show how to adapt well-known data structures to the privacy preserving context and what is the overhead of this adaptation. We present an algorithm for k-NN in secure multiparty computation. This is based on presenting private computation of several metrics. As a result, we can offer three approaches to associative queries over horizontally partitioned data with progressively less security. We show privacy preserving algorithms for data structures that induce a partition on the space; such as *KD*-Trees. Our next preference is our Privacy Preserving *SASH*. However, we demonstrate that the most effective approach to achieve privacy is separate data structures for parties, where associative queries work separately, followed by secure combination to produce the overall output. This idea not only enhances security but also reduces communication cost between data holders. Our results and protocols also enable us to improve on previous approaches for k-NN classification.

Keywords: Privacy Preserving Data Mining, Horizontally Partitioned Data, Data Structures for Associative Queries. k-NN queries.

1 Introduction

Never have globalization and international collaborations placed as much demand on partnerships between governments and/or corporations. Data mining has been identified as one of the most useful tools for the fight on terror and crime (Mena 2003). However, the information needed resides with many different data holders that must share their data with each other; thus, data privacy becomes extremely important. Parties may mutually not trust each other, but all parties are aware of the benefit brought by such collaboration. In the privacy preserving model, all parties of the partnership promise to provide their private data to the collaboration, but none of them wants the others or any third party to learn much about their private data.

For most data mining algorithms, the data is encoded as vectors in high dimensional space[1]. For these algorithms, a measure of similarity (or dissimilarity) is necessary and many times fundamental for their operation. More importantly, in large databases, attribute-vectors are the result of feature-extraction processes and constitute elements of high-dimensional spaces. These are subject to the curse of dimensionality and in such settings content-based retrieval under the vector model must typically be implemented as k-nearest-neighbour (k-NN) queries. k-NN queries produce the k items whose features most closely resemble those of the query vector according to the similarity measure. Therefore, the efficiency and scalability of k-NN methods strongly depends on that of k-NN search. And this directly impacts data mining algorithms. Thus, k-NN search is fundamental for for many data mining results.

Current database applications regularly use some from of similarity search. Well known examples include medical imaging (Korn, Sidropoulos, Faloutsos, Siegel & Proropapas 1996), and bioinformatics (Kahveci & Singh. 2001); moreover, also time-series databases (Faloutsos, Ranganathan & Manolopoulos May 1994), multimedia information systems (Subrahmanian 1999), geographical information systems (GIS) (Cheng, Dolin, Neary, Prabhakar, Ravikanth, Wu, Agrawal, El Abbadi, Freeston, Singh, Smith & Su 1997) and CAD/CAM (Berchtold 1997). These applications commonly have some distance function that acts as the similarity between two objects (among them Euclidean distance between the corresponding feature vectors). Two illustrations are the area of image databases (where the most similar images to a given image (Ankerst, Kriegel & Seidl 1998) are retrieved in this way) and molecular biology (Ankerst, Kastenmueller, Kriegel & Seidl 1999) (where features are derived from 3D shape histograms to find similar 3D proteins).

Data structures that efficiently find k-NN includes search structures include quadtrees (Samet 1984), *KD*-Trees (Bentley 1975), and *R*-Trees (Guttmann 1984), and newer structures such as SR-trees (Katayama & Satoh 1997), X-trees (Berchtold, Keim & Kriegel 1996), A-trees (Sakurai, Yoshikawa, S. & Kojima 2000), and iDistance (Yu, Ooi, Tan & Jagadish 2001). The central role of these spatial indices in data mining algorithms is illustrated by *R*-Trees for the efficiency of the popular clustering algorithm *DBSCAN* (Ester, Kriegel, Sander & Xu 1996).

We review secure multiparty computation in Section 2 because it provides the context for our protocols. While we review some existing protocols several new extensions are provided here. In Section 3

[1]Attribute-vectors are the common input for learning algorithms like decision trees, artificial neural network or for clustering algorithms like k-Means or *DBSCAN*.

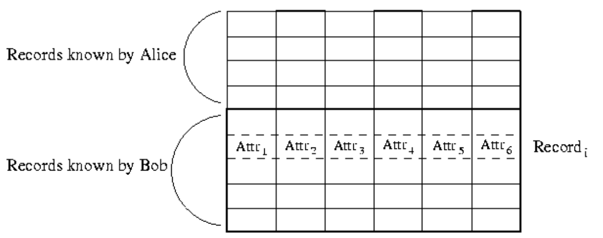


Figure 1: Horizontally partitioned data.

we introduce privacy preserving computation of some common metrics. Then, Section 4 shows privacy preserving k-NN queries. We include an analysis of what is the realistic expectation of k-NN for secure multiparty computation even in the ideal case. Section 6 demonstrates that our protocols are sufficient to produce a much better privacy preserving k-NN classifier for horizontally partitioned data. From here, we move into three models for associative queries. First, Section 7 proposes and demonstrates that the most effective approach to achieve privacy is separate data structures for parties, where associative queries work separately, followed by a secure combination protocol to produce the overall output. This idea not only enhances security but also reduces communication cost between data holders. However, most privacy preserving data mining approaches suggest a common data structure for mounting algorithms on top. The most modern and effective k-NN searcher for high dimensions with least assumptions on the metric is the *SASH* (Houle 2003*b*, Houle & Sakuma 2005). In Section 8, we show a privacy preserving *SASH*. We chose the *SASH* because it is neither a spatial index nor a metric index: it makes no assumptions on the nature of the database elements other than the existence of a pairwise distance measure, nor does it require the measure to satisfy the triangle inequality. Houle (Houle 2003*b*) has shown that for approximate k-NN queries on the largest sets, the *SASH* consistently returns a high proportion of the true k-NNs at speeds of roughly two orders of magnitude faster than sequential search. The *SASH* has been successfully applied to clustering and navigation of very large, very high dimensional text data sets (Houle 2003*a*), and spatial data mining of web documents (Morimoto, Aono, Houle & McCurley 2003). The fact that the *SASH* does not produce a partition of the search space adds additional security. In Section 9 we demonstrate that data structures that produce partitions (like R-Trees or KD-Trees) can be converted to the privacy preserving context but they are less secure. Section 10 offers a comparison and analysis while we conclude with final remarks in Section 11.

2 Privacy Preserving Computation

We study collaboration between several parties that wish to compute a function of their collective data. In fact, they are to conduct data mining tasks on the joint data set that is the union of all individual data sets. Each wants the others to find as little as possible of their own private data. To focus the discussion on privacy preserving collaboration, we will regularly use two parties we name Alice and Bob. Of course, the algorithms we describe here enable us to involve more than two parties, but for simplicity we will use only two parties. We focus on horizontally partitioned data (see Fig. 1). Some of the records are owned by Alice and the others by Bob. In the case of more than two parties, then every party will own some part (a number of records) from the database. A direct and naive use of data mining algorithms on the union of the data requires one party to receive data (every record) from all other parties, or all parties to send their data to a trusted central place. The recipient of the data would conduct the computation in the resulting union. In settings where each party must keep their data private, this is unacceptable. Note that, for horizontally partitioned data, the more parties are involved, the more records are involved and the larger is the global database.

For simplicity, we may assume each party owns one record only, so the number P of parties is also the number m of records. Typically there would be more records than parties (as in Fig. 1 where two parties have data for 9 records). However, we consider Alice as 4 virtual parties (one for each of the records) and Bob as 5 virtual parties each controlling one of Bob's records. This simplifies the notation in some of the algorithms (and communication between two virtual parties of the same party just does not need to occur).

The setting for our methods is privacy preserving computation originally developed under the name of "secure multiparty computation" (SMC) (Goldreich 1998). Here Alice holds one input vector $\vec{x}$ and Bob holds an input vector $\vec{y}$. They both want to compute a function $f(\vec{x},\vec{y})$ without each other learning anything about each other's input except what can be inferred from $f(\vec{x},\vec{y})$. Yao's Millionaires Problem (Yao 1982) provides the origin for SMC. In the millionaires, Alice holds a number a while Bob holds b. They want to identify who holds the larger value (they compute if $a > b$) without neither learning anything else about the others value. The function $f(\vec{x},\vec{y})$ is the predicate $f(a,b) = a > b$. There are now many solutions (Atallah & Du 2000) improving Yao's original solution (that required exponential complexity on the number of bits of $(a+b)$; however, a recent solution (Cachin 1999) is linear on the number of bits of $(a+b)$). We also adopt the *semi-honest* (Goldreich 1998) model for secure computation, which means both parties will follow the protocol since both are interested in the results. They will attempt to discover data (infer information) that they do not own, but will not supply false information in the protocol, neither will they fail to complete the protocol. To be totally precise one would have to produce proofs for the notion to "infer information $f(\vec{x})$ from $\vec{x}$". Such definition (Goldreich 1998) involves simulated computation in polynomial time when $\vec{x}$ is known. However, for space reasons we will not be so rigorous and we will use the following definitions.

DEFINITION 2.1 *We say that algorithm* A *is more secure (preferred) than algorithm* B *if from the output of algorithm* A *one can infer less information* [2] *than from algorithm* B.

Note that adversaries will always know at least a set of possible values of the confidential data. We express this as a bounding box or range.

EXAMPLE 2.1 *Alice has private data $\vec{p}$, and after applying algorithm* A*, Bob can discover a bounding box BB_A where $\vec{p}$ lies. However, after applying algorithm* B*, Bob can discover another bounding box BB_B. If $BB_B \subset BB_A$, then we say algorithm* A *is more secure (preferred) than algorithm* B.

EXAMPLE 2.2 *Alice has private data $\vec{p}_1$, $\vec{p}_2$, $\dots$, $\vec{p}_n$ and after applying algorithm* A*, Bob can discover bounding boxes BB_{A_i} with $\vec{p}_i \in BB_{A_i}$, for $i = 1,\dots,n$. After applying algorithm* B*, Bob can discover boxes BB_{B_j} with $\vec{p}_j \in BB_{B_j}$, for $j = 1,\dots,n$. If $BB_{B_j} \subset BB_{A_j}$ (for all j), then we say algorithm* A *is more secure (preferred) than algorithm* B.

[2] Less information means not only discovering approximate values for less number of private values, but it includes also a larger bounding box for data approximations.

2.1 Review of the Commodity Server

Although not strictly necessary, for performance reasons, we can use the help from an extra server, the commodity server, belonging to a third party. Alice and Bob could then send request to the commodity server and receive data (called commodities) from the server, but the commodities should be independent of Alice's or Bob's private data. The purpose of the commodities is to help Alice and Bob conduct the desired computation. The third party is semi-trusted in the following sense: (1) The third party should not be trusted; therefore it should not be possible (for this third party) to derive the private information of the data from Alice or Bob; it should not learn the computation result either. (2) The third party should not collude with both Alice and Bob. (3) The third party follows the protocol correctly. Because of these attributes, we say that the third party is a semi-trusted party. In the real world, finding such a semi-trusted third party is much easier than finding a trusted third party. As we will see from our solutions, the commodity server does not participate in the actual computation between Alice and Bob, it only supplies commodities that are independent of Alice and Bob's private data. Therefore, the server can even generate independent data off-line beforehand, and sell them as commodities to the prover and the verifier (hence the name "commodity server"). The commodity server model have been used previously in the literature (Beaver 1998) for solving private information retrieval problems.

2.2 Review of the Scalar Product Protocol

In this protocol, Alice has a vector $\vec{x}$ and Bob has another vector $\vec{y}$ (both with n elements). Alice and Bob use the protocol to compute the scalar product $\vec{x}^T \cdot \vec{y}$ between $\vec{x}$ and $\vec{y}$, such that Alice gets V_1 and Bob gets V_2, where $V_1 + V_2 = \vec{x}^T \cdot \vec{y}$ and V_2 is randomly generated by Bob. Namely, the scalar product of $\vec{x}$ and $\vec{y}$ is divided into two secret pieces, with one going to Alice and the other going to Bob. The computation is performed in the domain of the reals $\Re$ and by *Theorem 4.1* in (Du & Zhan 2002) both Alice and Bob cannot learn each other's private data.

Protocol (Scalar Product Protocol)

1. The Commodity Server generates two random vectors $\vec{R}_a$ and $\vec{R}_b$ of size n, and lets $r_a + r_b = \vec{R}_a^T \cdot \vec{R}_b$, where r_a (or r_b) is a randomly generated number. Then the server sends $(\vec{R}_a; r_a)$ to Alice, and $(\vec{R}_b; r_b)$ to Bob.

2. Alice sends $\vec{x}' = \vec{x} + \vec{R}_a$ to Bob, and Bob sends $\vec{y}' = \vec{y} + \vec{R}_b$ to Alice.

3. Bob generates a random number V_2, and computes $(\vec{x}'^T \cdot \vec{y}) + (r_b - V_2)$; then he sends the result to Alice.

4. Alice computes $(\vec{x}'^T \cdot \vec{y} + (r_b - V_2)) - (\vec{R}_a^T \cdot \vec{y}') + r_a = \vec{x}^T \cdot \vec{y} - V_2 + (r_b - \vec{R}_a^T \cdot \vec{R}_b + r_a) = \vec{x}^T \cdot \vec{y} - V_2 = V_1$.

The communication cost of this protocol is $4n$, which is 4 times more expensive than the optimal cost of a two-party scalar product (the optimal cost of a scalar product is defined as the cost of conducting the product of $\vec{x}$ and $\vec{y}$ without the privacy constraints, namely one party just sends its data in plain to the other party). The cost can be further improved to $2n$ because the vectors $\vec{R}_a$ and $\vec{R}_b$ are random generated by the commodity server; therefore the commodity server can send just the seeds (numbers of constant size) to Alice and Bob, and the seeds can be used to compute the random vector. Solutions to the scalar product protocol have been proposed before (Atallah & Du 2000, Vaidya & C. Clifton 2002) and both of these solutions achieves the security without using a third party. While the commodity server is not strictly necessary for our algorithms, in practice the communication and computation costs is more expensive than the solution without it that have a large constant under the $O(n)$ complexity, where n is the size of the vectors.

2.3 Extension to the Add Vectors Protocol

The technique was introduced for manipulation of vector operations as the "permutation protocol" (Du & Atallah 2001) and is also known as the "permutation algorithm" (Vaidya & C. Clifton 2003)). In this protocol, Alice has a vector $\vec{x}$ while Bob has vector $\vec{y}$ and a permutation π. The goal is for Alice to obtain $\pi(\vec{x}+\vec{y})$; that is Alice obtains the sum $\vec{s}$ of the vectors in some sense. The entries are randomly permuted, so Alice cannot perform $\vec{s} - \vec{x}$ to find $\vec{y}$. Also, Bob is not to learn $\vec{x}$. The solution is based on homomorphic encryption for which many implementations are possible. The protocol works as follows.

1. Alice produces a key pair for a homomorphic public key system and sends the public key to Bob. We denote by $E(\cdot)$ and $D(\cdot)$ the corresponding encryption and decryption system.

2. Alice encrypts $\vec{x} = (x_1, \cdots, x_n)^T$ and sends $E(\vec{x}) = (E(x_1), \cdots, E(x_n))^T$ to Bob.

3. Using the public key from Alice, Bob computes $E(\vec{y}) = (E(y_1), \cdots, E(y_n))^T$ and uses the homomorphic property to compute $E(\vec{x} + \vec{y}) = E(\vec{x}) \times E(\vec{y})$. Then, he permutes the entries by π and sends $\pi(E(\vec{x} + \vec{y}))$ to Alice.

4. Alice decrypts to obtain $D(\pi(E(\vec{x}+\vec{y}))) = \pi(\vec{x}+\vec{y})$.

We will extend this protocol to the case of $P \geq 3$ vectors, that is $P > 2$ parties are involved. In this case there is no need to permute the result, because $D(\cdot)$ is known only by Alice, and Alice will get the value $E(\vec{v}_2 + \cdots + \vec{v}_P)$, where $\vec{v}_i$ is the vector owned by i^{th} party. The algorithm is as follows:

1. The 1^{st} party (Alice), generates $E(\cdot)$ and $D(\cdot)$, then sends only $E(\cdot)$ to the other parties.

2. Then, the P^{th} party encrypts his data $E(\vec{v}_P)$ and sends it to $(P-1)^{th}$ party.

3. Next, the $(P-1)^{th}$ party encrypts his data $E(\vec{v}_{P-1})$ and using homomorphic encryption property computes
$$E(\vec{v}_{P-1}) \times E(\vec{v}_P) = E(\vec{v}_{P-1} + \vec{v}_P)$$
and sends this to the $(P-2)^{th}$ party.

4. The protocol continues until Alice (the first party) will get $E(\vec{v}_2 + \cdots + \vec{v}_{P-1} + \vec{v}_P)$ and she adds her data in the same way.

5. As Alice owns $D(\cdot)$, she decrypts the results and sends them to all other parties.

Note that in Step 1, the P^{th} party does not need to permute his result because the $(P-1)^{th}$ party does not know $D(\cdot)$ to decrypt. In this case $E(\cdot)$ could be as simple as adding random number (or X-OR with a random bit mask) and consequently $D(\cdot)$ will be subtracting the random number previously added.

2.4 New Protocol for the Maximum Value in the Sum of Vectors.

We will look at the two party case separately from three or more parties since these require slightly different solutions.

Two Party Case: Consider two attribute vectors $\vec{x}$ and $\vec{y}$, where only Alice knows $\vec{x}$ and only Bob knows $\vec{y}$. (where $\vec{x} = (x_1, \cdots, x_n)^T$ and $\vec{y} = (y_1, \cdots, y_n)^T$). The goal is to obtain $\max_i(x_i + y_i)$. There is a variant that obtains the index i_0 for which $x_i + y_i$ is maximum. However, the protocol should never reveal both, because then the parties find a value for the other. For example, Alice would find $\max_i(x_i+y_i) - x_{i_0} = y_{i_0}$. Even if the maximum value is the only outcome revealed, and the protocol is ideal, each party learns the value $x_{i_0} + y_{i_0}$ of the maximum, and thus, each party can subtract its data vector to find n values among which on value is from the other party. For example, Alice can compute $x_{i_0} + y_{i_0} - x_i$ (for $i = 1, \ldots, n$) and she finds n values one of which must be y_{i_0} owned by Bob.

Bob starts by generating a random value r and using $\vec{y} + r = (y_1 + r, \ldots, y_n + r)^T$ in the add vector protocol from Section 2.3. This provides Alice with $\pi(\vec{x} + \vec{y} + r)$; that is, Alice obtains the sum vector in permuted order, which will suffice for Alice to find the maximum and inform Bob. If the maximum value is the outcome sought, Alice provides only the value $x_{i_0} + y_{i_0} + r$. Bob will subtract r and pass $x_{i_0} + y_{i_0}$ to Alice. If the index of where the maximum is sought, then Alice provides only the index of where she found the maximum and Bob applies π^{-1} to broadcast the index. Note that Alice cannot learn which of the coordinates provided the maximum value, until Bob broadcast it. Note however, only when the maximum is sought, Alice learns the random number r and all the values in the entries of $\pi(\vec{x} + \vec{y})$, but this is not enough to learn any of the Bobs private data.

The Multiparty Case ($P > 2$): Here, direct use of our extension to the add vectors protocol will reveal slightly more information that we would like to; namely, Alice obtains the sum of the vectors without the effect of the permutation (see Subsection 2.3). She is able not only to find the maximum, but the index i_0 which hold the maximum of $x_i + y_i$ as well. So, Alice will know a bit more than the others. Therefore, we will improve this protocol. The proposed protocol works as follows. Assume there are $P > 2$ parties and every party has a vector $\vec{v}_i$.

1. The 1st party (Alice), generates a random vector $\vec{R} = (r_1, \ldots, r_n)$ and sends it to the P^{th} party.
2. Then, the P^{th} party adds his vector $\vec{v}_P$ to the random vector received from Alice and sends the sum to the $(P-1)^{th}$ party.
3. The protocol continues until Bob (the second party) receives the sum $\vec{S}_b = \vec{v}_3 + \cdots + \vec{v}_P + \vec{R}$
4. Alice and Bob use our finding the maximum value in sum (or the index) for $P = 2$ with the vectors $\vec{S}_a = \vec{v}_1 - \vec{R}$ owned by Alice and $\vec{S}_b = \vec{v}_3 + \cdots + \vec{v}_P + \vec{R} + r$ owned by Bob (Bob randomly generates r), which allows Alice to obtain $\pi(S_a + S_b) + r$; that is the sum (translated by r) of the vectors in permuted order.
5. Alice finds and passes the maximum value to Bob who subtracts r and broadcast the result to the other parties (or alternatively, if the index is sought, Alice will pass the index to Bob who will apply π^{-1} and broadcasts the index to all parties).

Lemma *Under the semi-honest model, for $P > 2$, the version* FINDING MAXIMUM VALUE IN THE SUM OF VECTORS *protocol does not allow any party to learn any party's value nor the index for where such maximum lies.*

Proof: The lemma holds in the multiparty ($P > 2$) case because of the random vector generated by Alice masks the data from the 3^{rd} party to the P^{th}. For Alice and Bob (the first and second parties), the protocol is secure as per the two party case, except that this time, even if Alice subtracts all the values in its vector from the maximum value, she only obtains partial sums and not value from the other parties.$\square$

A similar result applies to the version that broadcast the index but not the maximum value. In the literature, there are several algorithms to find privately not the maximum value but the index of the maximum value in a sum of vectors (Estivill-Castro 2004, Vaidya & C. Clifton 2003). Such algorithms are very costly; some require a theoretical construction hard to implement (Vaidya & C. Clifton 2003) while others (Estivill-Castro 2004) required P iterations of the add vectors protocol and a matrix of random values, plus m iterations of a division protocol. Those algorithms become slightly more efficient if they allow one party to find all the sums. Note that our index version of the algorithm does not allow this to happen (only a translation of the sums is found by Alice).

3 Privacy Preserving Metrics

Here Alice has again a vector $\vec{x}$ while Bob has vector $\vec{y}$. We introduce here secure computation of the Euclidean distance between these vectors. Alice replaces each component x_i with three components $x_i^2, -2x_i, 1$ while Bob replaces each y_i component with $1, y_i, y_i^2$. The dot product for these three components will then be $x_i^2 - 2x_iy_i + y_i^2 = (x_i - y_i)^2$. In general

$$\begin{aligned}\sum_i (x_i - y_i)^2 &= (\sum_i x_i^2, -2x_1, ..., -2x_n, 1) \cdot \\ &\quad (1, y_1, ..., y_n, -\sum_i y_i^2),\end{aligned}$$

and thus the Euclidean distance between two feature vectors can also be expressed as a scalar product of two vectors. Hence one could use the secure scalar product protocol (Du & Zhan 2002) (see Section 2.2), to compute a secure Euclidean distance. The result is two pieces of information V_1 and V_2, with V_1 going to Alice, V_2 going to Bob and $\sum_i (x_i - y_i)^2 = V_1 + V_2$.

It is known that the Euclidean distance is not robust in higher dimensions due to the many terms involved in the sum and the potential that a few values are magnified by squaring them. In other words, distinguishing what is close and what is far becomes very difficult. This is called the *curse of dimensionality.* To ameliorate this unpleasant fact, other metrics may be needed. We explore here the chessboard distance which is defined as

$$L^\infty(\vec{x}, \vec{y}) = \max(|x_1 - y_1|, ..., |x_n - y_n|).$$

Note that

$$(|x_1 - y_1|, \ldots, |x_n - y_n|)^T = |\vec{x} - \vec{y}| = |\vec{x} + (-\vec{y})|.$$

In order to compute the chessboard distance securely we can use our protocol for finding the maximum value in a sum of vectors for two parties. Note that Alice learns $\pi(|x_i - y_i|)$ which is not sufficient to learn values from Bob.

Other metrics that weight attributes differently and used in k-NN classification can also be computed securely using a strategy similar to the one above (for example, variants like $(\vec{x}-\vec{y})^T W(\vec{x}-\vec{y}) = \sum_i w_i(x_i - y_i)^2$ where W is a diagonal matrix of weights known to all parties). It should now be clear that it is also possible to compute securely the cosine metric $\vec{x}^T \cdot \vec{y}/(\|\vec{x}\|\|\vec{y}\|)$ as $(\vec{x}/\|\vec{x}\|)^T \cdot (\vec{y}/\|\vec{y}\|)$.

4 Privacy Preserving k-NN Queries

This section describes our privacy preserving k-NN algorithm. Later, we will show that with this operation we can build a privacy preserving *SASH*.

For the PP-k-NN protocol we are given a set of vectors

$$\vec{v}_1^T = (v_{11}, \cdots, v_{1n}), \cdots, \vec{v}_m^T = (v_{m1}, \cdots, v_{mn}) \quad (1)$$

and a vector $\vec{q}^T = (q_1, q_2, \cdots, q_n)$, where m is the number of records/vectors involved in the computation. The goal is to find $NN(\vec{q}, k)$ where $NN(\vec{q}, k)$ is the set of indices of the k nearest neighbours to the vector q. Assume the query vector $\vec{q}$ and the first $l < m$ vectors are owned by Alice while the other $m - l$ vectors are owned by Bob. In the case of the Euclidean distance, the distance values between $\vec{q}$ and $\vec{v}_i$, be partially distributed between Alice and Bob. Alice will have

$$V^1_{q,v_{l+1}}, \cdots, V^1_{q,v_m} \quad (2)$$

and Bob will have

$$V^2_{q,v_{l+1}}, \cdots, V^2_{q,v_m}, \quad (3)$$

where $dist(\vec{q}, \vec{v}_i) = V^1_{q,v_i} + V^2_{q,v_i}, \quad l < i \leq m$. Of course, Alice will have also the distance values for her own data. At this stage, Alice and Bob can perform the ADD VECTOR PROTOCOL with the values that determine $dist(\vec{q}, \vec{v}_i) = V^1_{q,v_i} + V^2_{q,v_i}, \quad l < i \leq m$. Alice receives these values shuffled by the permutation π that Bob knows. Alice finds among these values and her own $dist(\vec{q}, \vec{v}_i), 1 < i \leq l$, the k smallest. If any came from Bob's, she lets know the indexes j to Bob and Bob returns $\pi^{-1}(j)$ to Alice. Then, Alice broadcasts the indexes of all k-NN. Note that Alice learns all the distances from $\vec{q}$ to data points of Bob. We will discuss this issue later in Section 7.

Theorem 4.1 *The PP-k-NN protocol does not allow either Alice or Bob to learn each other's private data/vectors.*

Proof: In the first step of the PP-k-NN protocol, Alice obtains (2) and Bob obtains (3) as a result of the secure scalar product. The next step applies the secure add vector protocol (see Section 2.3) which allows Alice to learn the distance values. But, because the distances obtained by Alice were shuffled by Bob, Alice cannot learn the values in List (3). Clearly, Bob only learns the list of values in List (3) and the indexes of the k-NN. This, of course, is not enough to disclose Alice's private data. □

This protocol is efficient, because the secure scalar product can be implemented very efficiently in liner time on the dimension m of the data. The secure scalar product is executed at most m times for $O(mn)$ time. The secure add vectors depends on homomorphic encryption (which can be implemented with RSA). Because the add vector protocol for PP-k-NN operates here with at most m values, the complexity of this phase only depends on m and not the dimension of the data. The total complexity is $O(mn)$ time, which is linear in the input as per List (1).

5 The Ideal Case For Privacy Preserving k-NN queries

In this section we analyse what is the best possible security we can expect for a k-NN query. Assume Alice has a database $D_1 = \{\vec{a_1}, \cdots, \vec{a_m}\}$ and query vector $\vec{q}$, while Bob has database $D_2 = \{\vec{b_1}, \cdots, \vec{b_m}\}$. They want to compute privately

$$k\text{-NN}(\vec{q}, D_1, D_2) := \langle z_1, \cdots, z_k \rangle_{D_1 \bigcup D_2},$$

where $z_1, \cdots, z_k$ are the indexes of the vectors, which are k-NNs to the query point $\vec{q}$.

The literature of SMC has an ideal theoretical solution for this problem. In fact, it is always possible to securely compute $f(\vec{x}, \vec{y})$ for a polynomial-time f using private input $\vec{x}$ from Alice and private input $\vec{y}$ from Bob so that Alice learns nothing about $\vec{y}$ except what can be computed from $f(\vec{x}, \vec{y})$ and similarly Bob learns nothing about $\vec{x}$ except what can be inferred from $\vec{y}$ (Goldreich, Micali & Wigderson 1987). There is still a lot of interest in protocols for SMC because (1) the general solution requires f to be explicitly represented as a Boolean circuit of polynomial size and (2) the constants involved are not small, once the circuit is described the parties enter into a protocol holding shares of the inputs to gates and shares of the outputs of gates. In data mining settings this becomes impractical because even if represented as a circuit of polynomial size in its input, the input would represent the entire data sets of all the parties. Moreover, the community in the filed keeps discovering more efficient solutions for special cases of f.

If such an ideal (theoretically possible) protocol for k-NN were constructed from describing the function as a circuit, then every party obtains only the indexes $z_1, \cdots, z_k$. However, they also obtain what can be inferred from this. Assume $z_{i_1}, \cdots, z_{i_l}$ are indexes of vectors that belongs to Alice, and $z_{i_{l+1}}, \cdots, z_{i_k}$ are the those of vectors that belongs to Bob. If Bob constructs the $k-(l+1)^{th}$ order Voronoi diagram (Chazelle & Edelsbrunner 1987, Lee 1982) with his data, he can discover a cell/bounding box(BB) for the query vector $\vec{q}$. These cells are not regular, and in particular, the one for $\vec{q}$ could be extremely small even if the number $k-(l+1)$ is small. Naturally, the more of Bob's vectors among the k-NNs of $\vec{q}$, the more likely a more accurate cell/BB. Thus, in the ideal (theoretically possible) case, even Bob is able to discover a cell/BB where the query point $\vec{q}$ lies.

On the other hand, if Alice computes

$$d^a_{min} = \min_{a_i \in D_1 \text{ and } a_i \notin \{a_{z_{i_1}}, \cdots, a_{z_{i_l}}\}} (dist(\vec{q}, \vec{a_i})),$$

then Alice will know that all Bob's vectors that were included in k-NN are in the hyper-sphere centred by $\vec{q}$ with radius d^a_{min}.

6 Applications to Classification

Learning classifiers is a common machine learning or data mining task where from a training set of labelled cases, a classifiers is constructed to find the class of new instances. Famous approaches for this include Artificial Neural Networks, Decision Trees, Decision Lists and Support Vector Machines (Witten & Frank 2000). Recently, a solution for privately constructing k-NN classifiers in horizontally partitioned data has been proposed (Kantarcioğlu & Clifton 2004). This solutions is limited because it requires a non-colluding untrusted third party and is computationally costly — $O(P^2k^2)$ where P is number of parties.

Let us now see how we can use our PP k-NN protocol to solve this same problem far more efficiently and without a third party. We assume there are $P > 2$ parties and the query point $\vec{q}$ belongs to the first party. This is also an improvement from the previous solution that required the query point to be public (in our protocol, let the party that owns the query point be the first party). So, we are given $D_1, \cdots, D_P$ databases, where $D_i = \vec{p}^i_1, \cdots, \vec{p}^i_n$ are the attribute vectors for the i-th party. The t possible classes are identified by a set $L = l_1, \cdots, l_t$ of labels. Each point in $D_1 \cup \cdots \cup D_P$ has a label in L attached. The task is to find which of $l_j \in L$ corresponds to the query point $\vec{q}$ by taking a majority vote in the labels associated to the k-NNs of $\vec{q}$. Now, using our PP k-NN protocol (see Section 4), the parties can compute the k-NN of the query vector $\vec{q}$ (although Alice will also know all the distances). What we need now is to classify $\vec{q}$. The problem setting consists now of each party holding his/her training data vectors that are included in the global k-NN of the query point $\vec{q}$. Consequently, they know labels (classification) associated with their vectors. We want to compute which of the labels is the most present in the overall k-NN vectors. Assume the labels among the k-NN vectors are

$$\begin{aligned} &L^{k\text{-NN}} \\ &= \langle (1)_{i_1}, \cdots, l^{(1)}_{i_{p_1}}, l^{(2)}_{i_1}, \cdots, l^{(2)}_{i_{p_2}}, \cdots, l^{(m)}_{i_1}, \cdots, l^{(m)}_{i_{p_m}} \rangle \end{aligned}$$

where $p_1 + p_2 + \cdots + p_P = k$ and $l^{(j)}_{i_1}, \cdots, l^{(j)}_{i_{p_j}}$ are the labels of the vectors that the j-th party contributes to the k-NN. Then label for $\vec{q}$ will be

$$l^q = Majority(L^{k\text{-NN}}). \tag{4}$$

Hence, the challenge is how to compute (4) privately.

In fact, this is equivalent to finding the row with largest sum where each party owns a column, because $l^{(j)}_{i_1}, \cdots, l^{(j)}_{i_{p_j}}$ are the labels of the vectors that are known by the j-th party, he can compute sums of every label present in $l^{(j)}_{i_1}, \cdots, l^{(j)}_{i_{p_j}}$, if there are labels that were nor present, just put 0. This is the column known by the i-th party. For example, assume the j-th party labels are $(l_1, l_2, l_2, l_1, l_4, l_4, l_1)$, then he will have $(3, 2, 0, 2, 0, \cdots, 0)$ as a representative column.

Hence, every party will know its representative vector with length t. The protocol must add them and find the index of the maximum value. This will represent the class label that will be assigned to $\vec{q}$. We now use our index-version of the FINDING MAXIMUM IN THE SUM OF VECTORS protocol (see Section 2.4) to add all representative vectors, and then Alice (the first party) can find the maximum value and the permuted index where the maximum lies. Because Bob was the author of permutation π, then Bob can apply π^{-1} to discover the original index where the maximum value lies. This will represent the label's index in L, so $\vec{q}$ will be labelled.

Example: Assume there are three parties and after computation of representative vectors they have $R^{(1)} = (2, 0, 0, 3, 2, 0)$, $R^{(2)} = (0, 1, 2, 0, 4, 0)$, $R^{(3)} = (0, 0, 1, 3, 1, 1)$. When we apply the secure add vectors protocol Alice gets a permutation of $R = (2, 1, 3, 6, 7, 1)$, so the id that represents the maximum value here is 5, then Bob needs to find the real id of the maximum and broadcast it. Thus, label assigned for $\vec{q}$ will be $l_{\pi^{-1}(5)}$.

Our solution is less costly that the solution requiring a third party.

7 Private Combination of Local Associative Queries

Because the data is horizontally partitioned, every party owns part (a number of records) of the database. Our algorithm in Section 4 allows Alice to learn all the distances from its data point $\vec{q}$ to the data points of Bob. In this sections we discuss the rationale for this. First, note that if $\vec{q}$ were public (Kantarcioğlu & Clifton 2004), then each party could compute local k-NN and each would have a set of k distance values. The problem then reduces to find the k-richest millionaires and is solvable by repeated application of Yao's Millionaires Problem.

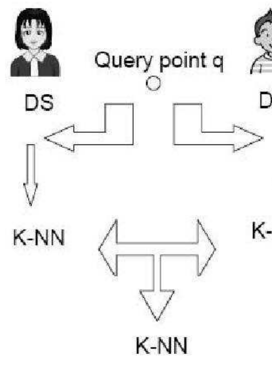


Figure 2: How local results combined to obtain result for joint database.

Figure 2 illustrates how the combining proceeds for two parties, but it is also applicable when more parties involved in the computation.

1. Alice and Bob use data structures (DS) to organise their data locally.
2. When the public query point $\vec{q}$ comes, both of them calculate the local k-NN for their data using their own data structures.
3. Then, k repetition of Yao, Millionaires problem determines the k smallest values for $dist(\vec{q}, \vec{v})$ among the parties.

Now, if we assume that $\vec{q}$ is private and owned by the first party, Alice, we have some complications if Bob learns $dist(\vec{q}, \vec{v})$ for vectors $\vec{v}$ in Bob's database (even when distances between the query point and Bob's vectors can be calculated using PP metrics as in Section 3). When the query point $\vec{q}$ is owned by Alice and Bob receives the results of the distance calculations, then Bob has the possibility to locate the query point $\vec{q}$. For instance, say Bob knows $dist(\vec{q}, \vec{v}_1) = d_1$ and $dist(\vec{q}, \vec{v}_2) = d_2$. From the first fact, Bob will know that $\vec{q}$ is in the hyper-sphere centred at $\vec{v}_1$ and with radius d_1. From the second fact, again $\vec{q}$ is in the hyper-sphere centred at $\vec{v}_2$ and with radius d_2. Hence, $\vec{q}$ in the intersections of this two hyper-spheres. With a third point $\vec{v}_3$, Bob will learn q is at the intersection of 3 hyper-spheres. This may provide significant information to identify values for $\vec{q}$, or very small regions where it lies. When the query point $\vec{q}$ owned by Alice and Alice receives the results of distance calculations, Alice learns only that Bob's vectors are in the hyper-sphere centred at $\vec{q}$ and the radius depending on the distances from $\vec{q}$. Even, if Alice knows all the distances from the query point $\vec{q}$ and indexes for Bob's vectors, Alice is unable to find the exact location of any Bob's vectors. Because all the hyper-spheres are centred at $\vec{q}$. These hyper-spheres either coincides with each other or do not intersect each other. That is the rationale why we allow the owner of the query point to learn the distance values.

8 The *SASH* Data Structure

In this section we show a strategy by which we limit the number of distances that one party learns when the k nearest neighbours to one of its data points is performed. The idea comes form approaches where the parties share a common data structure (Du & Zhan 2002, Amirbekyan & Estivill-Castro 2006). Not that when parties share a data structure, they share the nodes and the pointers among the nodes, but not the information stored at the nodes. If we want to share a search data structure, but preserve privacy we recommend the *SASH*. In this multi-level search structure queries are processed by recursively locating approximate neighbours within the sample, and then using the pre-established connections to discover neighbours within the remainder of the data set. We describe our privacy preserving version of the algorithms of the *SASH* data structure. We consider n objects for which a similarity measure $dist(u, v)$ exists between any two objects u and v. Since the *SASH* assumes only a metric $dist$, we can use any of the metrics between attribute-oriented vectors for which we have presented SMC protocols.

The directed edge-weighted graph that constitutes the *SASH* is shared by the parties. While each database object corresponds to a unique node, only the party that owns that record (vector) will know the data and the index of that vector, the others will only know who is the owner of the node. Thus, the following makes no distinction between a node and the database object to which it corresponds. Nodes are organised into a hierarchy of levels, ranging from a bottom level containing $\lfloor n/2 \rfloor$ nodes (the leaves), to a top level containing a single node (the root). With the possible exception of the top level, each level contains half as many nodes as the level below it, rounded down. The levels of the *SASH* are numbered from 1 (the top level) to h (the bottom level). Edges within the *SASH* link nodes from consecutive levels. Each node can have edges directed to at most p parent nodes as the level above it, and to at most c child nodes as the level below it. Every node except for the root must have at least one parent. In the discussion below, we will assume that each party stores (with each edge (u, v)) the data that enables them to compute securely k-NN regarding the distance $dist(u, v)$ (for example, in the case of Euclidean distance, the value $dist(u, v)$ may be partitioned among the parties as a sum of values add into to $dis(u, v)$). Every node v (other than the root) has an edge directed to one parent $g(v)$ that is designated as its guarantor. The guarantor of v must have v as one of its children; v is called the dependent of $g(v)$. The requirement that every node have a guarantor ensures that every node is reachable from the root.

The edges of the *SASH* heuristically minimise the distances between their endpoints. During the construction, each new node is attached to a small number of its near neighbours from the level above it. *SASH* edges do not induce a partition of the objects as lower levels. At the start of construction, the *SASH* is empty, and the parties pick a random and uniform order on the totality of the data to insert the database objects. As a result of this, the parties alternate being the party that owns the query point $\vec{q}$ and because the internal k-NN queries in the *SASH* are only for levels with restricted number of nodes, the party does not get to learn distances to all data points of the other parties.

Let $SASH_i$ denote the graph induced by the nodes from level 1 through i, for $1 \le i \le h$. $SASH_i$ is a *SASH* in itself. The construction of the entire *SASH* (that is, $SASH_h$) proceeds by iteratively constructing $SASH_1$, $SASH_2$, ..., $SASH_h$ in order. The following algorithm shows how to build securely $SASH_l$ given $SASH_{l-1}$, for $1 \le l \le h$, by adding edges between nodes of the current last two levels.

Algorithm Privacy Preserving ConnectSASHLevel(l):

1. If $l = 2$, then every node of level 2 will have the root node as its sole parent and guarantor, and the root node will have all nodes of level 2 as its children and dependents. This completes the construction of $SASH_2$.

2. Otherwise, for the remaining steps, we have $l > 2$. For each node v of level l, the parties choose a set of up to p near neighbours $P_i(v, p)$ from among the nodes of each level for $i = 1$ to l:

   (a) If $i = 1$, then $P_i(v, p)$ consists of a single node, the root.

   (b) Otherwise, $i > 1$. Let $P'_i(v)$ be the set of distinct children of the nodes of $P_{i-1}(v, p)$. Set $P_i(v, p)$ to be the p nodes of $P'_i(v)$ closest to v, according to the measure $dist$ using our privacy preserving k-NN operation. If $|P'_i(v)| < p$, then set $P_i(v, p) = P'_i(v)$.

3. Set the parents of v to the nodes in $P_{l-1}(v, p)$. Each element v at level l has up to p distinct parents associated with points in its vicinity.

4. Create the child edges for the nodes of level $l-1$, as follows:

   (a) For each node u of level $l - 1$, determine the list of distinct nodes $C(u)$ of level l that have chosen u as a parent.

   (b) Use our Privacy Preserving k-NN to find the c closest points to u among those in $C(u)$.

   (c) Set the children of u to be these c nearest neighbours.

5. For each node v of level l, determine whether it was accepted as a child of any node at level $l-1$. If yes, then the closest node that accepted it as a child becomes the guarantor $g(v)$ of v, and v becomes a dependent of $g(v)$. Otherwise, label v as an orphan node.

6. For each orphan node v at level l, a node at level $l - 1$ is needed to act as its guarantor. The node should be close as possible to v (in terms of the distance measure), and must be unencumbered; that is it must have fewer that the maximum allowed number c of children. Find a guarantor for v by successively doubling the size of the candidate parents set as follows:

   (a) Set $i = 1$

   (b) Compute $P_{l-1}(v, 2^i p)$ as in Step 2.

   (c) If $P_{l-1}(v, 2^i p)$ has no unencumbered node, increment i and go to 6b.

   (d) Otherwise, choose as the guarantor $g(v)$ the unencumbered node of $P_{l-1}(v, 2^i p)$ that is closest to v. Add v as a child and dependent of $g(v)$, and replace the parent of v furthest from v by $g(v)$.

This completes the construction of the privacy preserving $SASH_l$. Note that the information shared by the parties is all the edges (parent/child relationships) and results of k-NN queries that identify only owners of vectors but do not reveal the data associated with those vectors. It may be necessary to demonstrate to all other parties that all the local data is involved in the process. The *SASH* does not partition the search space, but a KD-Tree or an R-Tree does. Our case study is KD-Trees but the conclusions apply to R-Trees, since in fact, a node in an R-Tree is a bounding box for all data below that node.

9 Privacy Preserving KD-Trees

Multidimensional binary search trees (K-Dimensional search trees or KD-Trees)[3] are a generalisation of binary search trees for multidimensional points. A KD-Tree for a set of K-dimensional records is such that:

1. Each node contains a K-dimensional record and has an associated discriminant $j \in \{1, 2, ..., K\}$.
2. For every node with key $\vec{x}$ and discriminant j, any record in the left subtree with key $\vec{y}$ satisfies $y_j < x_j$ and any record in the right subtree with key $\vec{y}$ satisfies $y_j > x_j$.
3. The root node has depth 0 and discriminant 1. All nodes at depth d have discriminant $(d \texttt{ mod } K) + 1$.

A KD-Tree can be built by successive insertion into an initially empty KD-Tree. When inserting a key $\vec{x}$, we compare the key to be inserted with some key $\vec{y}$ at the root of some subtree: if $\vec{y}$ is at level j, we compare $x_{(j \texttt{ mod } K)+1}$ and $y_{(j \texttt{ mod } K)+1}$, and recursively continue the insertion in the left or the right subtree of $\vec{y}$, until a leaf (empty subtree) is found.

The construction of a shared K-dimensional search tree enables one party to learn bounding boxes for the other party's data. Without loss of generality, we will construct a secure solution for two parties and assume that $K = 2$ (we are in 2D). So, for the moment, the number P of parties is 2 (as well as the dimension m of the attribute vectors). Extensions to more parties and cases where each party holds more than two attributes are easy to infer once we present the initial case. So, in what follows we have a setting where the data points look like $\vec{p} = (p_1, p_2)$ where both p_1 and p_2 are known by only one party. Because we assumed two parties and the data is horizontally partitioned into two-dimensional domains for each party, we take it that the first party (Alice) knows a fraction of the total number of vectors while the other one (Bob) knows the other part (some number of vectors). All parties will know the structure of the tree; that is, all parties will know how many nodes are at each level and what records fall to the left or to the right on each internal node. The data at a node is only known by the party that owns the vector at that node. The other party only knows they are not the owner. As the tree is constructed, both parties hold an empty tree. A random order is jointly decided on the totality of the data. Both parties insert the first point, one will be the owner. Later points are included by the recursive insertion. As a data point arrives, there are two possibilities, the data point and the root of the subtree are owned by the same party. Then, that party performs the comparison and announces the result without any other part discovering here anything about the data values of the record being inserted or of the subtree's root. However, because data is partitioned horizontally, there would be cases where the vector being inserted and the subtree root node are own by different parties. In this case, they can use Yao's comparison protocol (Yao 1982) to compare the two values. Thus, parties never announce their values, only the outcome of comparisons. Once this secure comparison of levels is set in the KD-Tree, all other algorithms for associative queries can be set up in the privacy preserving KD-Tree.

[3]Here $K = m$ the dimension of the vectors, while k means the number of nearest neighbours.

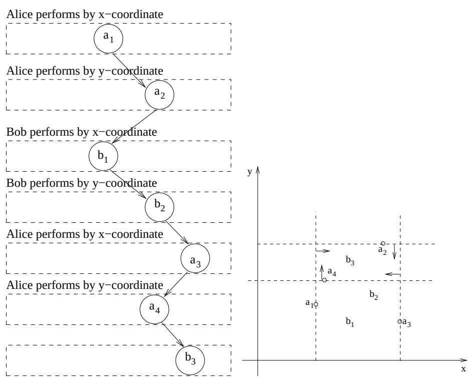

Alice performs by x−coordinate
a1
Alice performs by y−coordinate
a2
Bob performs by x−coordinate
b1
Bob performs by y−coordinate
b2
Alice performs by x−coordinate
a3
Alice performs by y−coordinate
a4
b3
y
x
a2
b3
a4
b2
a1
b1
a3

Figure 3: Example of partial 2D-tree structure and how Alice obtains range where $\vec{b_3}$ lies.

The KD-Tree is a sufficient data structure on small data sets that fit in RAM (main memory) and for which we are not performing many deletions and insertions of records (or locks for multi-user access or transactions) as it would be the case of a multi-dimensional database. Although large databases that use concurrent access and use external disks (I/O to store on disk besides RAM) require R-Trees or other partition-based data structure, we hope that the KD-Tree illustration shows how this data structures could be shared with some level of privacy for associative queries. However, as we mentioned before, when the data structure is shared, the tree structure is known to all parties involved. The question is: How much can be inferred from this kind of information? Unfortunately, quite rapidly one party can learn bounding boxes on the data of the other party. It is not uncommon that on average, with a tree of depth 7, one party would have compared values with 4 of its data points. All data of other parties that fall in the bounding box determined by those four points are known to be inside such bounding box. According to the algorithm's output both Alice and Bob know the structure shown in Figure 3, but they do not know the values of the each others points. Now let us see what Alice can infer from the output and her data about Bob's vector $\vec{b_3}$. Figure 3 shows how Alice obtains the approximate(range where the vector lies) location of $\vec{b_3}$. Here, because the KD-Tree constructed by arranging the values to the left or to the right, according to the secure comparison, Alice using her values and already constructed KD-Tree (she has as an output) detects the approximate location of the vector owned by Bob.

Of course with vectors of higher dimensions (more than 2) and more parties (more than 2), the level of privacy is higher. With D dimensions one party constructs a bounding box after having $D \times 2$ points in a path to some leaves.

10 Comparison and Analysis

While it is more effective and efficient that each party compute the k-NN on its data and then to merge them as in our privacy preserving k-NN algorithm, the *SASH* allows the parties to monitor the participation of other parties. We have assumed the semi-honest model, but if each party uses their own data structure to answer locally its k-NN query, in reality, one party may chose to keep away always some different sections of their data as they see fit. With a shared data structure, this is not possible. The data in the shared structure will be involved in many associative queries and many k-NN, perhaps the entire process of classification or clustering. Clearly, the methods are equivalent if one party decides from the beginning never to involve some set of records, but then, the results will not be accurate for the union of all records anyways.

We have shown that even in the ideal case, k-NN queries tend to disclose some information about the data of the parties involved. Even in the ideal case, bounding boxes or bounding regions are found for some of the data owned by other parties. From the perspective of Definition 2.1, the separate data structures and secure k-NN is the most secure option, but parties have less certainty that all others have contributed their entire data sets. The secure *SASH* allows parties to learn slightly more about the data values of other parties, but provides more assurances that all parties have contributed their data (otherwise the shared *SASH* can not be constructed) or that the data placed into the shared data structure is the effective union of the data. It does not enable to infer directly bonding boxes for other party's records. Partition data structures, like *KD*-Trees (or *R*-Trees) are the least private as they enable one party to immediately learn bounding boxes for the other's data. All these options incur a communication overhead. However, this is insignificant if we consider the overhead for the transmission of all data to a trusted server which performs the desired analysis on the join data.

10.1 Time Complexity Analysis

The complexity, of course, depends of the data structures, that every party uses locally and the number of distances calculated for local k-NN, plus after combining of the local k-NN query results the communication cost for sending the overall result to all parties. The cost of communication in our algorithms is much less that the communication cost when all parties migrate their data to a trusted party. For example, in the PP-k-NN protocol when $\vec{q}$ is public, one can easily see that the cost of communication is only applied to the vectors which are candidates for the k-NN query (see Figure 2), which is clearly cheaper than bringing all the data together.

In terms of CPU-time overhead, we have shown that all the algorithms induce only a linear time overhead. In all dictionary data structures, when the data is high-dimensional, the CPU-time costs are mainly those associated with metric evaluation. We have chosen *SASH*, because this data structure is very efficient in this manner. The approximate k-NN queries proceeds by choosing $P_i(q, k)$ at every level of the *SASH*, then combing them and choosing k closest to the query vector $\vec{q}$. In the *SASH* (Houle & Sakuma 2005, Houle 2003*b*) there are two strategies for queries: uniform and geometric. Their paper suggests that the geometric pattern improves both accuracy and search time. In geometric search instead of finding at every level $P_i(q, k)$, one looks for $P_i(q, k_i)$, where $k_i = \max\{k^{1-\frac{h-i}{\log_2 n}}, \frac{1}{2}pc\}$, for all $1 \leq i \leq h$. The upper bounds on the number of distance computations for the *SASH* (Houle & Sakuma 2005, Houle 2003*b*) are as follows.

$$\begin{aligned}\textit{SASH}\text{ construction: } & pcn \log 2n \\ \text{Approx. } k\text{-NN query (uniform): } & ck \log 2n \\ \text{Approx. } k\text{-NN query (geom.): } & \frac{k^{1+\frac{1}{\log 2n}}}{k^{\frac{1}{\log 2n}} - 1} + 2p^3 \log 2n\end{aligned}$$

Note that the cost of the geometric pattern query is less than uniform query. The cost of privacy would then add a multiplicative constant to these costs.

11 Final remarks

The development of privacy preserving versions of *SASH* and *KD*-Trees reveals that if the structure is shared between parties, it provides some unnecessary information (see Figure 3) and the information leak is worse as the data structure is organised around partitions of the space. That is, although the construction data structures is secure, its operation can disclose some private information. While these may seems unsatisfactory, the fact remains that the protocols and algorithms presented here are the most practical know to our knowledge. They allow some level of protection at essentially affordable cost (the CPU-time is affected only by a constant factor). Other methods are essentially theoretical because the calculations we want to perform here cannot be written into a small circuit with as many inputs as all the data points.

We have presented several privacy preserving metrics and based on this, an algorithms to obtain k-NN with some level of preserving privacy. This provide some practical methods for applications in classification and clustering with considerations to privacy.

References

Amirbekyan, A. & Estivill-Castro, V. (2006), Privacy preserving DBSCAN for vertically partitioned data, *in* 'IEEE Int. Conf. on Intelligence and Security Informatics, ISI 2006', Springer Verlag LNCS 3975, San Diego, CA, USA, pp. 141–153.

Ankerst, M., Kastenmueller, G., Kriegel, H. & Seidl, T. (1999), Nearest neighbor classification in 3D protein databases, *in* 'In Proc. 7th Int. Conf. on Intelligent Systems for Molecular Biology (ISMB-99)', pp. 34–43.

Ankerst, M., Kriegel, H. & Seidl, T. (1998), 'A multi-step approach for shape similarity in image databases', *IEEE Transactions on Data Engineering* **10**(6), 996–1004.

Atallah, M. & Du, W. (2000), Secure multi-party computational geometry, *in* 'Proc. of the 7th Int. Workshop on Algorithms and Data Structures', LNCS 2125, Springer Verlag, pp. 165–179.

Beaver, D. (1998), Server-assisted cryptography., *in* 'NSPW '98: Proc. of the 1998 workshop on New security paradigms', ACM Press, New York, NY, USA, pp. 92–106.

Bentley, J. (1975), 'Multidimensional binary search trees used for associative retrieval.', *Communications of the ACM* **18**(9), 509–517.

Berchtold, K. H. P. (1997), Similarity search in CAD database systems, *in* 'Proc. of ACM SIGMOD Int. Conf. on Management of Data', Tuscon, Arizona, pp. 564–567.

Berchtold, S., Keim, D. A. & Kriegel, H.-P. (1996), The X-tree: An index structure for higher dimensional data, *in* 'Proc. 22th VLDB Conf.', pp. 28–39.

Cachin, C. (1999), Efficient private bidding and auctions with an oblivious third party, *in* 'Proc. of the 6th ACM Conf. on Computer and Communications Security', SIGSAC, ACM Press, Singapore, pp. 120–127.

Chazelle, B. & Edelsbrunner, H. (1987), 'An improved algorithm for constructing k-th order voronoi diagrams', *IEEE Transactions of Computers* **C**(36), 1349–1354.

Cheng, X., Dolin, R., Neary, M., Prabhakar, S., Ravikanth, K., Wu, D., Agrawal, D., El Abbadi, E., Freeston, M., Singh, A., Smith, T. & Su, J. (1997), Scalable access within the context of digital libraries, *in* 'Proc. of the Int. Conf. on Advances in Digital Libraries, ADL', Washington, D.C., pp. 70–81.

Du, W. & Atallah, M. (2001), Privacy-preserving cooperative statistical analysis, *in* 'Proc. of the 17th Annual Computer Security Applications Conf. (ACSAC)', ACM SIGSAC, IEEE Computer Society, New Orleans, Louisiana, pp. 102–110.

Du, W. & Zhan, Z. (2002), Building decision tree classifier on private data, *in* V. Estivill-Castro & C. Clifton, eds, 'Privacy, Security and Data Mining', IEEE ICDM Workshop Proc., Volume 14 in the Conf. in Research and Practice in IT Series, ACS, Sydney, Australia, pp. 1–8.

Ester, M., Kriegel, H., Sander, S. & Xu, X. (1996), A density-based algorithm for discovering clusters in large spatial databases with noise, *in* E. Simoudis, J. Han & U. Fayyad, eds, 'Proc. of the 2nd Int. Conf. on Knowledge Discovery and Data Mining (KDD-96)', AAAI, AAAI Press, Menlo Park, CA, pp. 226–231.

Estivill-Castro, V. (2004), Private representative-based clustering for vertically partitioned data, *in* R. Baeza-Yates, J. Marroquin & E. Chávez, eds, 'Fifth Mexican Int. Conf. on Computer science (ENC 04)', SMCC, IEEE Computer Society Press, Colima, Mexico, pp. 160–167.

Faloutsos, C., Ranganathan, M. & Manolopoulos, Y. (May 1994), Fast subsequence matching in time-series databases., *in* 'In Proc. ACM SIGMOD Int. Conf. on Management of Data', Minneapolis, pp. 419–429.

Goldreich, O. (1998), 'Secure multi-party computation', Working draft, Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehevolt, Israel. www.citeseer.ist.psu.edu/goldreich98secure.html.

Goldreich, O., Micali, S. & Wigderson, A. (1987), How to play any mental game (extended abstract), *in* A. Aho, ed., 'Proc. of the 19th ACM Annual Symposium on Theory of Computing', ACM Press, New York, pp. 218–229.

Guttmann, A. (1984), R-trees: a dynamic index structure for spatial searching, *in* 'Proc. ACM SIGMOD Int. Conf. on Management of Data', pp. 47–57.

Houle, M. (2003*a*), Navigating massive data sets via local clustering, *in* 'KDD '03: Proc. of the ninth ACM SIGKDD Int. Conf. on Knowledge discovery and data mining', ACM Press, New York, NY, USA, pp. 547–552.

Houle, M. (2003*b*), SASH: a spatial approximation sample hierarchy for similarity, Technical Report RT-0517, IBM Tokyo Research Laboratory. 16 pages.

Houle, M. & Sakuma, J. (2005), Fast approximate similarity search in extremely high-dimensional data sets, *in* '21st Int. Conf. on Data Engineering ICDE', IEEE Computer Society, Tokyo, Japan, pp. 619–630.

Kahveci, T. & Singh., A. K. (2001), An efficient index structure for string databases, *in* 'VLDB Conf.', Rome, Italy, pp. 351–360.

Kantarcioğlu, M. & Clifton, C. (2004), Privatly computing a distributed k-nn classifier, *in* J.-F. Boulicaut, ed., '8-th European Conf. on Principles and Practice of Knowledge Discovery in Databases PKDD', Vol. 3202, Springer Verlag LNCS, pp. 279–290.

Katayama, N. & Satoh, S. (1997), The SR-tree: an index structure for high-dimensional nearest neighbour queries, *in* 'ACM SIGMOD Conf. on Management of Data', Tucson, USA, pp. 369–380.

Korn, F., Sidropoulos, N., Faloutsos, C., Siegel, E. & Proropapas, Z. (1996), Fast nearest neighbor search in medical image databases, *in* 'VLDB', Mumbai, India, pp. 215 – 226.

Lee, D. (1982), 'On k-nearest neighbors voronoi diagrams in the plane.', *IEEE Transactions of Computers* **C**(31), 478–487.

Mena, J. (2003), *Investigative Data Mining for Security and Criminal Detection*, Butterworth-Heinemann, US.

Morimoto, Y., Aono, M., Houle, M. & McCurley, K. (2003), Extracting spatial knowledge from the web, *in* 'Int. Symp.on Applications and the Internet (SAINT)', Orlando, USA, pp. 326–333.

Sakurai, Y., Yoshikawa, M., S., U. & Kojima, H. (2000), The A-tree: an index structure for high-dimensional spaces using relative approximation, *in* '26th VLDB Conf.', pp. 519–526.

Samet, H. (1984), 'The quad-tree and related hierarchical data structures', *ACM Computing Surveys* **16**(2), 187–260.

Subrahmanian, V. (1999), *Principles of mutimedia database systems*, Morgan Kaufmann Publishers, Inc., San Francisco, CA, USA.

Vaidya, J. & C. Clifton, C. (2002), Privacy preserving association rule mining in vertically partitioned data, *in* 'The Eighth ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining', SIGKDD, ACM Press, Edmonton, Canada, pp. 639–644.

Vaidya, J. & C. Clifton, C. (2003), Privacy-preserving k-means clustering over vertically partitioned data, *in* 'Proc. of the SIGKDD-ACM Conf. of Data Mining', ACM Press, Washington, D.C., US, pp. 206–215.

Witten, I. & Frank, E. (2000), *Data Mining — Practical Machine Learning Tools and Technologies with JAVA implementations*, Morgan Kaufmann Publishers, San Mateo, CA.

Yao, A. (1982), Protocols for secure computation, *in* 'IEEE Symposium of Foundations of Computer Science', IEEE Computer Society, pp. 160–164.

Yu, B., Ooi, C., Tan, K. & Jagadish, H. (2001), Indexing the distance: an efficient method to KNN processing, *in* '27th VLDB Conf.', Rome, Italy, pp. 421–430.

A Processing Model for the Optimal Querying of Encrypted XML Documents in XQuery

Tao-Ku Chang
Graduate Institute of Information and Computer Education, National Taiwan Normal University
Taipei, Taiwan
tkchang@ice.ntnu.edu.tw

Gwan-Hwan Hwang
Department of Computer Science and Information Engineering, National Taiwan Normal University Taipei, Taiwan
ghhwang@csie.ntnu.edu.tw

Abstract

XQuery is a powerful and convenient language that is designed for querying the data in XML documents. In this paper, we address how to optimally query encrypted XML documents using XQuery, with the key point being how to eliminate redundant decryption so as to accelerate the querying. We propose a processing model that can automatically and appropriately translate the XQuery statements for encrypted XML documents. Furthermore, we show that XML schema is significantly associated with queries over XML documents. The implementation and experimental results demonstrate the practicality of the proposed model.

Keyword: XML, XQuery, DSL, Security, Database.

1 Introduction

The XQuery language (Scott et al., 2005) proposed by W3C was designed to be broadly applicable across all types of XML data sources. Its mission is to provide flexible query facilities to extract data from real and virtual documents on the Web. XQuery uses an XML data model that can represent XML documents, sequences, or atomic elements (such as integers or strings). The concept of XQuery is depicted in Figure 1. *Q* represents an XQuery program that includes navigation in XML documents using XPath (Clark and DeRose, 1999), database statements (the so-called FLWOR expressions), construction of new XML elements, operations on XML Schema types, and function calls. The XQuery engine queries and formats data from an XML database that stores XML documents according to *Q*, with the resultant XML document being *R*.

XML is becoming a widespread data-encoding format for Web applications and services, which makes it important to secure XML documents in various ways. For example, we may need to sign and encrypt XML documents in order to ensure nonrepudiation and confidentiality (Schneier, 1995). Based on XML element-wise encryption (Maruyama and Imamura, 2000), the W3C's XML encryption working group (http://www.w3.org/Encryption/2001/Overview.html) delivered a recommendation specification for XML encryption (Imamura et al., 2002). The encrypted document specifies a process for encrypting data and representing the result in XML. The encrypted data may be arbitrary data, an XML element, or the content of an XML element. Figure 2 illustrates the concept of element-wise encryption. Only one element ("`Number`") of the original document is encrypted. This enables XML files to protect themselves because the sensitive data in XML are encrypted by particular keys.

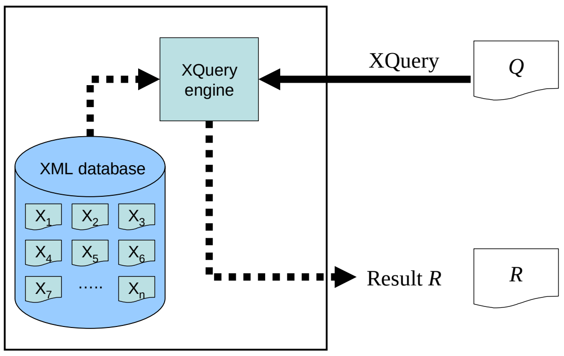
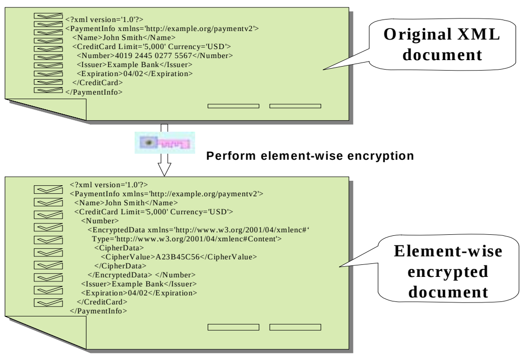


Figure 1: The data flow for querying XML documents

This paper addresses how to query data from these encrypted XML documents in XQuery. The intuitive, trivial method is to first decrypt the encrypted XML documents and then use an XQuery program to obtain the desired documents (see Figure 3). The drawback of this approach is that it is quite inefficient in certain situations because all of the encrypted elements in the queried XML document must be decrypted. According to its operational semantics, XQuery is normally used to obtain a small set of elements from the target XML documents. It is not theoretically necessary to decrypt all the encrypted elements in the target XML document – we only have to decrypt those elements that belong to the result elements of the issued query. It is obvious that a scheme that does not need to decrypt unwanted elements should be more efficient than a scheme that decrypts all the encrypted elements.

The first aim is to eliminate unnecessary decryption. According to the specification of W3C XML encryption (Imamura et al., 2002), the scopes of encryption could be "`element`", which encrypts a whole element (including the start/end tags), or "`content`", which encrypts the content of an element (between the start/end tags). Consider the XML document shown in Figure 4. The "`payer`" and "`cardinfo`" elements are encrypted as a whole; that is, their encryption scope is set to "`element`". In the encrypted XML document shown in Figure 5, the "`CipherData`" element contains the encrypted data of the "`payer`" and "`cardinfo`" elements, and is wrapped by the "`EncryptedData`" element. We see that the tag names of the "`payer`" and "`cardinfo`" elements disappear. Figure 5 indicates that once the encryption scope of an element is set to "`element`", its tag name cannot be examined unless we first decrypt the element. The type of encryption scope is helpful to data security because there is no clue about which element is encrypted. Figure 6 lists an XQuery program that is used to obtain the value of the "`cardinfo`" element from Figure 4. It is obvious that we cannot use this program to query the encrypted document shown in Figure 5; it appears that we have to decrypt the two encrypted elements before performing the query. However, since we only want to query one of them, one of the decryptions is redundant.


Original XML document
Perform element-wise encryption
Element-wise encrypted document

```
<?xml version='1.0'?>
<PaymentInfo xmlns='http://example.org/paymentv2'>
  <Name>John Smith</Name>
  <CreditCard Limit='5,000' Currency='USD'>
    <Number>4019 2445 0277 5567</Number>
    <Issuer>Example Bank</Issuer>
    <Expiration>04/02</Expiration>
  </CreditCard>
</PaymentInfo>
```

```
<?xml version='1.0'?>
<PaymentInfo xmlns='http://example.org/paymentv2'>
  <Name>John Smith</Name>
  <CreditCard Limit='5,000' Currency='USD'>
    <Number>
      <EncryptedData xmlns='http://www.w3.org/2001/04/xmlenc#'
       Type='http://www.w3.org/2001/04/xmlenc#Content'>
        <CipherData>
          <CipherValue>A23B45C56</CipherValue>
        </CipherData>
      </EncryptedData> </Number>
    <Issuer>Example Bank</Issuer>
    <Expiration>04/02</Expiration>
  </CreditCard>
</PaymentInfo>
```

Figure 2: Example of element-wise encryption

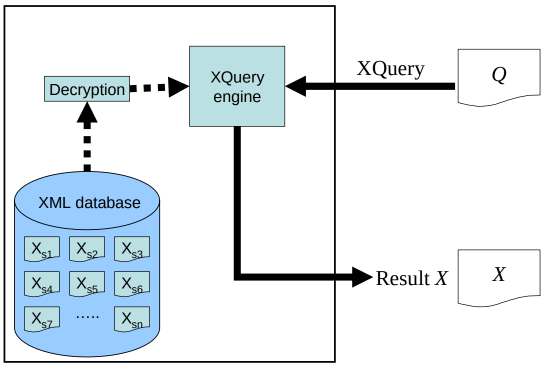


Figure 3: A trivial way to query encrypted XML documents

To improve the efficiency of decryption of encrypted XML documents in the query process, we should avoid performing unnecessary decryption. For the example shown in Figure 4, Figure 5, and Figure 6, it is obvious that some additional information is necessary to eliminate the redundant decryption because the encryption may break the structure of the XML document. Sometimes the structure information should be referred to during the query. As noted above, we use XML Schema (Fallside and Walmsley, 2004) that provides a means for defining the structure, content and semantics of XML documents to support it. It is usually used to validate XML documents but plays an important role in the XML queries in this research. We will illustrate it in Section 3. In this paper, we present the type of information required to eliminate redundant decryption and propose a processing model to automatically translate an XQuery program written by users to another one that can accurately locate the target elements that should be decrypted. The presented translation algorithm is optimal in terms of the computation required for decryption.

```
<?xml version='1.0'?>
<transactions>
  <transaction>
     <payer id = "M123456789">tony yao</payer>
     <price current="TWD">1350</price>
     <cardinfo>
        <cardtype>g</cardtype>
        <orgination>visa</orgination>
        <owner>tony yao</owner>
        <creditline>200000</creditline>
        <expiredate>12/01/2007</expiredate>
     </cardinfo>
  </transaction>
</transactions>
```

Figure 4: An XML document

```
<?xml version='1.0'?>
<transactions>
  <transaction>
     <EncryptedData
      Type='http://www.w3.org/2001/04/xmlenc#Element'
      xmlns='http://www.w3.org/2001/04/xmlenc#'>
        <CipherData>
          <CipherValue>
                  mrs79DfdL+ODXzur3DZXBDJx2EwRgz+MRP3Nv9T20J2L
                  ltPYthkSAG0zVoCt+GZhSdcf4T9xLp78tOxRN/PgmGo2
                  hLSO/3OtqTNukDooxPmA7sADaWiZOe6rbrNdFY5QgjBA
                  Z8TlnQ3SSBiSM11rygoDei4LTJEROcN6Lq5lL/c=
          <CipherValue>
        <CipherData>
     </EncryptedData>
     <price current="TWD">1350</price>
     <EncryptedData
      Type='http://www.w3.org/2001/04/xmlenc#Element'
      xmlns='http://www.w3.org/2001/04/xmlenc#'>
        <CipherData>
          <CipherValue>
                  h3IkkoyhsUL0uuC7MtSyw/xMfWlcKb144rH5EAQQ8vrj
                  rs3B1RwmIDF9lYBChHkfghk3eW4Jb6fQrnemykms7ZIA
                  y7dHpxL2lC7sJ0rX1UlDjzNoRHKVZo80IZzQ9yP/+mBl
                  br6C/mD5vE9aa2FEEAlFvdGxPeW62fKCD3ZM15kotIRw
                  yf5O+Ja1UJgLN2Juu5AQ3qkpScJBeocSeF207rveeCYP
                  yd+Nh/GrDFzjCndBOB1YV7RXXyUvaDu2PZ55OTwNufUQ
                  ggpvxpDZUZ7fSOkjzHrDN88ZwULKIf6aLBt1M=
          <CipherValue>
        <CipherData>
     </EncryptedData>
  </transaction>
</transactions>
```

Figure 5: An encrypted XML document

```
<transactions>
  {
   for $b in
     doc("example.xml")/transactions/transaction/cardinfo
     where $b/cardno = "1234-5678-8765-4321"
     return
       $b/cardno
  }
</transactions>
```

Figure 6: An XQuery to extract "`cardinfo`" from an XML file

The remainder of this paper is organized as follows: Section 2 presents the proposed processing model, Section 3 presents an algorithm for the transformation of XQuery statements for querying encrypted XML documents, Section 4 presents our implementation and experimental results, and Section 5 concludes the paper.

2 The Processing Model for Querying Encrypted XML Documents

Optimally querying the encrypted XML documents in XQuery requires information about security. Note that an optimal query is defined as that requiring minimal decryption for encrypted elements in the target XML documents. Generally speaking, the encryption and signature standards proposed by W3C offer a complete definition of the format for the encrypted XML document (Imamura et al., 2002). However, the language is not sufficiently powerful for the programmer to specify how to encrypt and sign his or her XML documents. To overcome this limitation, we previously proposed a security language that allows a programmer to specify the security detail of XML documents: the document security language (DSL) (Hwang and Chang, 2001, 2003, 2004, 2005). The DSL can be used to define how to perform encryption and decryption, and the embedding and verification of signatures. It offers a security mechanism that integrates *element-wise encryption* and *temporal-based element-wise digital signatures*. Also, because the syntax of the "`EncryptedData`" element in the XML encryption standard prevents its extension to handle attribute encryption, the DSL supports a type of element-wise encryption that is more general: the scope of encryption (or encryption granularity) can be a whole element, some of the attributes of an element, or the content of an element; where an attribute has two possible types of encryption: (1) to only encrypt its value and (2) to encrypt both its name and value (Chang and Hwang, 2003). The encrypted document produced by the *DSL securing tool* can be made compatible with the XML encryption and digital signature standard in cases where attribute encryption is not applied.

Figure 7 illustrates the relationship between XML, DSL, and the DSL securing tool. Figure 7A shows the process of encrypting and embedding digital signatures. The details of the encryption process and the digital signature itself are stored in a DSL document in D_P, D_T, and D_{Sig}: D_P is the security pattern definition that specifies the combination of security algorithms and encryption and decryption keys, D_T is the transformation description definition that specifies the actual data transformation of element-wise encryption, and D_{Sig} specifies how to embed digital signatures in the resulting XML document. The target XML document that is ready to be encrypted and signed is X. The DSL securing tool reads, parses, and analyzes D_P, D_T, D_{Sig}, and X, and then generates X_s and $D_{P'}$. X_s is still an XML document, but some of its elements contain ciphertexts that are translated by the DSL securing tool according to the encryption details recorded in D_P and D_T. In addition to the encrypted elements, X_s also contains signatures that are embedded by the DSL securing tool. Each signature signs a portion of the data in X. It should be noted that D_P and $D_{P'}$ may contain different information: D_P holds information describing how to encrypt X, whereas $D_{P'}$ should include details of how to decrypt X_s. In addition, we have developed a DSL editor with a graphical user-friendly interface to make it easier for users to generate DSL documents (Hwang and Chang, 2005).

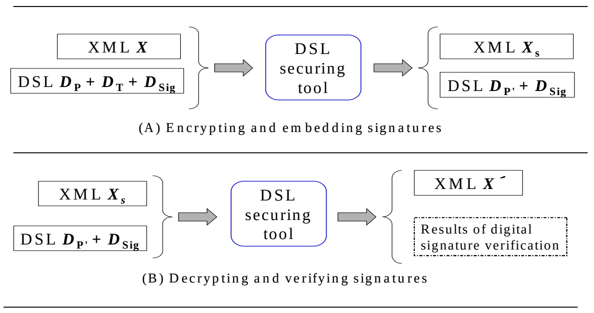

XML X
DSL D_P + D_T + D_Sig
DSL securing tool
XML X_s
DSL D_P' + D_Sig
(A) Encrypting and embedding signatures
XML X_s
DSL D_P' + D_Sig
DSL securing tool
XML X´
Results of digital signature verification
(B) Decrypting and verifying signatures

Figure 7: The operational model for securing XML documents

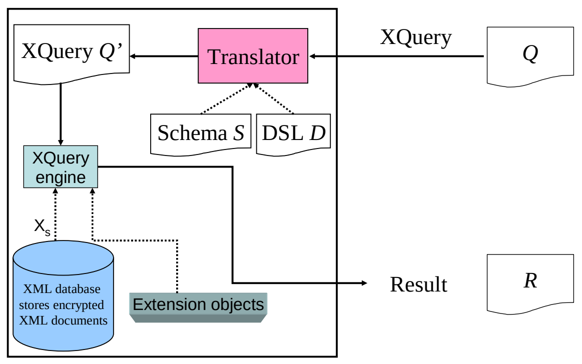


Figure 8: The processing model for querying encrypted XML documents

Figure 8 depicts the processing model we propose for the efficient querying of encrypted XML documents. Q is the original XQuery program. Note that Q is written to query data from the original XML document (i.e., the unencrypted document). D is a DSL document. The encrypted XML document X_s is encrypted according to D and is stored in the XML storage. Before Q is sent to the XQuery engine, the translator parses it and translates it into Q'. Q' is also an XQuery program, but some expressions in it are translated according to D and the XML Schema S (Fallside and Walmsley, 2004). In cases where the result document R contains some encrypted elements in X_s or the query needs to consult some encrypted element in X_s, Q' contains codes to invoke decryption functions that are the extension objects. Note that the XML Schema S may not be available; however, D is generally sufficient to generate an efficient XQuery Q'. In certain circumstances the information contained in S can be used to generate a more efficient query compared with a transformation obtained by only consulting D. The translation from Q to Q' is detailed in Section 3.

3 The Transformation Algorithm of XQuery Statements for Querying Encrypted XML Documents

Now we present our design of an algorithm that is used to transform the XQuery statements; that is, the design of the translator shown in Figure 8. We begin by considering the syntax of the XQuery statement. Each XQuery program contains one or more query expressions. The FLWOR expression is the most powerful of the XQuery expressions and is, in many ways, similar to the `SELECT-FROM-WHERE` statement used in SQL (ISO/IEC 9075-2, 2003). The formal grammar for a FLWOR expression in XQuery is defined in (Boag et al., 2005) as follows:

```
FLWORExpr ::= (ForClause | LetClause)
                          WhereClause?      OrderByClause?
                          return ExprSingle
```

The above BNF[1] form of the FLWOR expression is quite protean, being capable of generating a large number of possible query instances. The `ExprSingle` term following the "`return`" keyword can itself be replaced by another FLWOR expression, so that FLWOR expressions can be strung together ad infinitum. The replacement of an `ExprSingle` term by any other expression type is what makes XQuery composable and gives it its rich, expressive power. There are many expression types in XQuery, each of which can be plugged into the grammar wherever a more generic `ExprSingle` expression is called for.

In this paper, we focus on FLWOR expressions to implement the transformation algorithm, which is listed in Figure 9. In the following we use four examples to demonstrate this algorithm.

```
Algorithm: Transform a FLWOR expression for querying
encrypted XML documents
Input:
  Let F is a FLWOR expression of the form:
      FLWORExpr ::= (ForClause | LetClause)
      WhereClause?  OrderByClause?  return ExprSingle
  Let D is a DSL file
  Let S is an XML Schema
Output:
  N = A FLWOR expression
Begin_of_Algorithm
{
 ● Step 1:
    Let T_set represents the set of the path templates in the
    DSL file
    Let IF_set represents the set of the paths in ForClause
    Let IW_set represents the set of the paths in WhereClause?
    Let I_set = (IF_set ∪ IW_set)
    Let R_set represents the set of paths referred in
         ExprSingle. Note that if the ExprSingle is a FLWOR
         expression, we do not add the paths referred in the
         FLWOR expression to R_set
    BoundVariable_set = The bound variables in ForClause
    TargetXML_set = The file names of target XML documents
                    in doc function
    ForClause_String = The string of ForClause in F
    WhereClause_String = The string of WhereClause? in F
    ReturnClause_String = The string of "return" + ExprSingle
                           in F
    N = Null string

 ● Step 2:
    if Intersection(I_set, T_set)=∅[2] and
        Intersection(R_set, T_set)=∅
    {
     N = F
    }

    if Intersection(I_set, T_set)≠∅ and
        Intersection(R_set, T_set)=∅
    {
    P_set = XPath_Transformation (IF_set, T_Set, S);
    Scope_Array =
    Decryption_Scope (IF_Set, IW_set, R_set, T_set);

    For i = 1 to (the number of bound variables in
       BoundVariable_set)
    {
       BoundVariable_set_1(i) =
       BoundVariable_set(i) +"_1";
    }
    N = "for "
    For i = 1 to (the number of bound variables in
       BoundVariable_set)
    {
       N = BoundVariable_set_1(i) +" in
            doc("+TargetXml_set(i)+")"+P_set(i)+"\n";
    }
    For i = 1 to (the number of bound variables in
       BoundVariable_set)
    {
       N = N +"let " + BoundVariable_set(i)+
       "=decryption("+ BoundVariable_set_1(i)+",\""+
                      Scope_Array(i)+"\")"+"\n";
    }
    N = N + "return" + "\n";
    N = N + "if"
    For i = 1 to (the number of bound variables in
       BoundVariable_set)
    {
      N = N +" "+"(count("+BoundVariable_set(i)+")>0)"
      if BoundVariable_set(i) ≠ null
      {
         N = N + " and"
      }
    }
    N = N + " and "+WhereClause_string+"\n"+"then "+
         ReturnCluase_string+"\n"+"else ()"+"\n";
   }

  if Intersection(I_set, T_set)=∅ and
      Intersection(R_set, T_set)≠∅
  {
    Scope_Array =
    Decryption_Scope(IF_set, IW_set, R_set, T_set);
    For i = 1 to (the number of bound variables in
       BoundVariable_set)
    {
       BoundVariable_set_1(i) =
       BoundVariable_set(i) +"_1";
    }
    For i = 1 to (the number of bound variables in
       BoundVariable_set)
    {
      New_forClause =
       ForClause_String.replace(BoundVariable_set(i),
                                BoundVariable_set_1(i))
    }
    For i = 1 to (the number of bound variables in
       BoundVariable_set)
    {
       New_whereClause =
       WhereClause_String.replace(BoundVariable_set(i),
            BoundVariable_set_1(i))
    }
    N = N + New_forClause +"\n"
    N = N + New_whereClause +"\n"
    For i = 1 to (the number of bound variables in
       BoundVariable_set)
    {
       N = N +"let " + BoundVariable_set(i) +
       "=decryption("+ BoundVariable_set_1(i)+",\""+
                      Scope_Array(i)+"\")"+"\n";
    }
    N = N +"retrun"+"\n";

    N = N + "if"
    For i = 1 to (the number of bound variables in
      BoundVariable_set)
    {
      N = N +" "+"(count("+BoundVariable_set(i)+")>0)"
      if BoundVariable_set(i) ≠ null
      {
         N = N + " and"
      }
    }
    N = N + " and "+WhereClause_string+"\n"+"then "+
        ReturnCluase_string+"\n"+"else ()"+"\n";
  }

  if Intersection(I_set, T_set)≠∅ and
```

[1] See Fischer and LeBlanc (1991) for more information about the BNF representation. In this paper, all the nonterminal symbols are underscored.

[2] The symbol ∅ indicates the empty set.

```
      Intersection(R_set,T_set)≠∅
    {
      P_set = XPath_Transformation(IF_set, T_set, S)
      Scope_Array =
      Decryption_Scope(IF_set, IW_set, R_set, T_set);

      N ="for "
      For i = 1 to (the number of bound variables in
         BoundVariable_set)
      {
         N = BoundVariable_set_1(i) +" in
             doc("+TargetXml_set(i)+")"+P_set(i)+"\n";
      }
      For i = 1 to (the number of bound variables in
         BoundVariable_set)
         N = N +"let " + BoundVariable_set(i)+
         "=decryption("+ BoundVariable_set_1(i)+",\""+
                         scope_Array(i)+"\")"+"\n";
      }

      N = N + "return"+ "\n";
      N = N + "if"
      For i = 1 to (the number of bound variables in
        BoundVariable_set)
      {
        N = N +" "+"(count("+BoundVariable_set(i)+")>0)"
        if BoundVariable_set(i) ≠ null
        {
           N = N + " and"
        }
      }
      N = N + " and "+WhereClause_string+"\n"+"then "+
      ReturnCluase_string+"\n"+"else ()"+"\n";
     }
}
End_of_Algorithm

Procedure XPath_Transformation(IF_set,T_set,S)
Input:
  IF_set = A set of paths
  T_set = The set of path templates in the DSL file
  S = An XML Schema
Output:
  P_set = A set of paths
Begin
{
  For i = 1 to (the number of paths in IF_set)
  {
     if  (IF_set(i) ⊆ T_set) {
          Pt0 = A string in IF_set(i) from right to left until
          character is "/"
         Pt1  =  Delete Pt0 in IF_set(i) from right
         index = 0;
         If S is available {
             index = check-schema (IF_set(i), S)
         }
         if index >=1{
             P = Pt1 + "EncryptedData"+
                 "[" +index.toString()+"]"}
         else{
             P = Pt1 + "EncryptedData"
         }
     }
     else {P=IF_set(i)}
     Write P to P_set
  }
}
End

Procedure Decryption_Scope(IF_set,IW_set,R_set,T_set,S)
Input:
  IF_set = The set of the path in ForClause
  IW_set = The set of the path in WhereClause?
  R_set = The set of paths referred in ExprSingle. Note that
        if the ExprSingle is a FLWOR expression, we do not
        add the paths referred in the FLWOR expression to
        R_set
  T_set = The set of path templates in the DSL file
  S = An XML Schema
Output:
  Scope_Array = String Array
Begin
{
  for i = 1 to (the number of paths in IF_set)
  {
    scope = null string
    if (IF_set(i) ⊆ T_set) {
        scope = "all"
        Write scope to scope_Array
        Continue for loop
    }
    if  (IW_set ⊆ T_set) and ((IW_set ∩ IF_set(i) ≠∅){
        If S is available {
            index = check-schema (IW_set, S)
        }
        if (index >=1){
            scope = scope + "child:EncryptedData"+
                              "["+index.toString()+"]"
        }
        else{
            scope = scope + "child:EncryptedData"
        }
    }
    if  (R_set ⊆ T_set) and ((R_set ∩ IF_set(i) ≠∅) {
        If S is available {
            index = check-schema (R_set, S)
        }
        if (scope <> null){
             scope = scope + ";"
        }
        if (index >=1){
            scope = "child:EncryptedData"+
                      "["+index.toString()+"]"
        }
        else{
            scope = "child:EncryptedData"
        }
    }
    Write scope to Scope_Array
  }
}
End
```

Figure 9: Transformation algorithm

The first example demonstrates an XQuery program that queries some of the encrypted elements from the target XML document.

Figure **10**A lists a FLWOR expression that performs a simple search that returns the "`cardinfo`" element from the document `example.xml` (see Figure 4) where the value of "`/transactions/transaction/price`" is "`1350`". The XML document shown in Figure 5 is that encrypted according to the DSL document shown in Figure 11. The input includes a FLWOR expression, a DSL document, and an XML Schema. `Step 1` defines some variables: "*T_set*" represents the set of path templates in the DSL file, "*I_set*" represents the set of paths in "<u>`ForClause`</u>" and "<u>`WhereClause`</u>", and "*R_set*" represents the set of paths referred to in <u>`ExprSingle`</u>. Note that if <u>`ExprSingle`</u> is a FLWOR expression, we do not add the paths referred to in the FLWOR expression to "*R_set*". We present the situation in which <u>`ExprSingle`</u> is a FLWOR expression in the third example. In `Step 2`, we first compute the intersections of "*I_set*" and "*T_set*" and of "*R_set*" and "*T_set*". The intersection of "*I_set*" and "*T_set*" is not the empty set when the queried elements according to "<u>`ForClause`</u>" and "<u>`WhereClause`</u>" contain encrypted elements. Similarly, the intersection of "*I_set*" and "*R_set*" is not empty when the return elements contain encrypted elements. In this example there are two path templates in the DSL document (see Figure 11), and we have *T_set* = {"`/transactions/transaction/payer`," "`/transactions/transaction/cardinfo`"}, <u>`ForClause`</u> = "`for $b in doc("example.xml")/transactions/transaction`", <u>`WhereClause?`</u> = "`where $b/price=1350`", *I_set* = {"`/transactions/transaction`," "`/transactions/transaction/cardinfo/price`"}, <u>`ExprSingle`</u> = "`$b/cardinfo`", and *R_set* = {"`/transactions/transaction/cardinfo`"}. The intersection of "*I_set*" and "*T_set*" is not the empty set, whereas that of *R_set* and *T_set* is the empty set.

According to the algorithm listed in Figure 9, the translator then generates the transformed FLWOR expression. The "`ForClause`" and "`WhereClause?`" statements are changed to "`for $b_1 in doc("example.xml")/transactions/transaction`" and "`where $b_1/price=1350`", respectively. A "`LetClause`" statement ("`let $b = decryption($b_1,"child:EncryptedData[2]")`") is added after the "`ForClause`" and "`WhereClause?`" statements. Note that "`LetClause`" invokes a decryption function to decrypt the `$b_1` variable since it contains the encrypted elements that the original XQuery statement wants to query. Finally, we change `ExprSingle` to "`if (count($b) >0 then {$b/cardinfo} else ()`". The output FLWOR expression is listed in Figure 10B.

```
<transactions>
   {
   for $b in doc("example.xml")/transactions/transaction
   where $b/price=1350
   return
       $b/cardinfo
   }
</transactions>
```
(A) An input FLWOR expression

```
<transactions>
   {
   for $b_1 in doc("example.xml")/transactions/transaction
   where $b_1/price=1350
   let $b = decryption($b_1, "child:EncryptedData[2]")
   return
      if (count($b) >0
      then
        {
          $b/cardinfo
        }
      else ()
   }
</transactions>
```
(B) An output FLWOR expression

Figure 10: An XQuery to extract "`cardinfo`" from an encrypted XML file

```
<?xml version="1.0" ?>
<dsl:security_document
 xmlns:dsl="http://www.xml-dsl.com/2002/dsl" version="1.0">
    :
    :
   <dsl:template match="/transactions/transaction/payer">
      <dsl:value-of-encrypted-node scope="element"
            pattern="pattern1"/>
   </dsl:template>
   <dsl:template match="/transactions/transaction/cardinfo">
      <dsl:value-of-encrypted-node scope="element"
            pattern="pattern2"/>
   </dsl:template>
</dsl:security_document >
```

Figure 11: A DSL document

Figure 12A shows our second XQuery program, whose "`ForClause`", "`WhereClause?`", and `ExprSingle` expressions contain XPaths that point to encrypted elements. The program performs a search that returns the "`cardno`" element from the document `example.xml` (see Figure 4), where the value of "`/transactions/transaction/cardinfo/cardno`" is "`1234-5678-8765-4321`". In this example, we have *T_set* = {"`/transactions/transaction/payer`," "`/transactions/transaction/cardinfo`"}, *I_set* = {"`/transactions/transaction/cardinfo`," "`/transactions/transaction/cardinfo/cardno`"}, and *R_set* = {"`/transactions/transaction/cardinfo/cardno`"}.

The intersections of *I_set* and *T_set* and of *R_set* and *T_set* are not the empty set. According to the algorithm listed in Figure 9, "`ForClause`" is changed to "`for $b_1 in doc("example.xml")/transactions/transaction/EncryptedData[2]`". A "`LetClause`" statement ("`let $b = decryption($b_1,"all")`") is added after the "`ForClause`" statement. "`LetClause`" invokes a decryption function to decrypt the `$b_1` variable which represents the elements pointed at by the XPath `/transactions/transaction/EncryptedData[2]`. Finally, `ExprSingle` is modified by adding "`if (count($b) >0 and $b/cardno = "1234-5678-8765-4321" then $b/cardno else ()`". The output FLWOR expression is listed in Figure 12B.

```
<transactions>
   {
   for $b in doc("example.xml")/transactions/transaction/cardinfo
   where $b/cardno = "1234-5678-8765-4321"
   return
      $b/cardno
   }
</transactions>
```
(A) An input FLWOR expression

```
<transactions>
   {
   for $b_1 in doc("example.xml")/transactions/transaction/EncryptedData[2]
   let $b = decryption($b_1, "all")
   return
     if (count($b) >0 and $b/cardno = "1234-5678-8765-4321"
     then $b/cardno
     else ()
   }
</transactions>
```
(B) An output FLWOR expression

Figure 12: An XQuery to extract "`cardinfo`" from an encrypted XML file

Figure 13A is the third example, which is a more complicated XQuery program. The `ExprSingle` statement contains an FLWOR expression. The "`WhereClause?`" statement in the outer FLWOR expression contains encrypted elements. The FLWOR expressions `ExprSingle` and "`ForClause`" also contain encrypted elements. The transformation process occurs from outside to inside. We first transform the outer FLWOR expression: we have *T_set*={"`/transactions/transaction/payer`," "`/transactions/transaction/cardinfo`"} and *I_set*={"`/transactions/transaction`," "`/transactions/transaction/payer`"}. The inner FLWOR expression "`for $a in $b/cardinfo return $a`" will not be changed when transforming the outer FLWOR expression: thus we have *R_set*={"`/transactions/transaction/price`"}. After invoking the intersection function, the intersection of "*I_set*" and "*T_set*" is not the empty set whereas that of *R_set* and *T_set* is the empty set. The "`ForClause`" statement is changed to "`for $b_1 in doc("example.xml")/transactions/transaction`". A "`LetClause`" statement ("`let $b = decryption($b_1, "child:EncryptedData[1]")`") is added after "`ForClause`", which invokes a decryption function to decrypt the `$b_1` variable. Finally, we transform the `ExprSingle` into the following statements:

```
"if (count($b) >0 and $b/payer = "tony yao"
 then
 {
  <transaction>
     {
       $b/price
       for $a in $b/cardinfo return $a
     }
  </transaction>
 }
 else ()”.
```

After transforming the outer FLWOR expression, we should proceed to transform the inner FLWOR expression “`for $a in $b/cardinfo return $a`” to “`for $a_1 in $b/EncryptedData[2] $a = decryption($b_1,"all") if count($a)>0 then return $a else()`” according to the algorithm listed in Figure 9. The output XQuery program is listed in Figure 13B.

```
<transactions>
 {
   for $b in doc("example.xml")/transactions/transaction
   where $b/payer="tony yao"
   return
      <transaction>
         {
            $b/price
            for $a in $b/cardinfo
            return $a
         }
      </transaction>
 }
</transactions>
```

(A) An input FLWOR expression

```
<transactions>
 {
   for $b_1 in doc("example.xml")/transactins/transaction
   let $b = decryption($b_1, "child:EncryptedData[1]")
   return
      if (count($b))>0 and $b/payer="tony yao")
      then
      {
         <transaction>
            {
               $b/price
               for $a_1 in $b/EncryptedData[2]
               $a = decryption($b_1,"all")
               if count($a)>0
               then
                  return $a
               else()
            }
         </transaction>
      }
      else()
 }
</transactions>
```

(B) An output FLWOR expression

Figure 13: An XQuery to extract “`cardinfo`” from an encrypted XML file

It is essential to use the DSL in the proposed processing model because the translator must investigate the DSL document to determine which elements were encrypted. Although it is not compulsory to use XML Schema, it can be used to further reduce the times required for decryption. XML Schema is a DTD successor that expresses shared vocabularies and provides a guide for characterizing the structure, content, and semantics of an XML document. Furthermore, XML Schema offers (1) XML query validation, by exploiting the XML query language syntax to translate relative paths into absolute paths; and (2) identification of parent–child relationships, which improves the performance in solving XML queries for applications that require detection of these and other ancestor–descendant relationships.

In the following, we demonstrate that the XML Schema can be used to optimize the query. Figure 14 is an encrypted version of the XML document shown in Figure 4. Note that all child nodes of the transaction element are encrypted as a whole. If the user wants to obtain the value of the “`cardinfo`” element, s/he must write a “<u>`ForClause`</u>” statement such as “`$b in doc("example.xml")/transactions/transaction/EncryptedData`” in an XQuery program. However, there are three elements with tags named “`EncryptedData`”. These elements will be decrypted to check their tag names to identify which is the “`cardinfo`” element. We can use XML Schema to avoid the redundant decryption. Figure 15 lists the XML Schema of the XML document shown in Figure 4. The translator looks it up to determine that the “`cardinfo`” element is the third child element of the “`transaction`” element. Thus, the “<u>`ForClause`</u>” statement can be changed to “`doc("example.xml")/transactions/transaction/EncryptedData[3]`”, where the “`[3]`” means that only the third “`EncryptedData`” element needs to be decrypted.

```
<?xml version='1.0'?>
<transactions>
  <transaction>
    <EncryptedData
     Type='http://www.w3.org/2001/04/xmlenc#Element'
     xmlns='http://www.w3.org/2001/04/xmlenc#'>
      <CipherData>
        <CipherValue>
                mrs79DfdL+ODXzur3DZXBDJx2EwRgz+MRP3Nv9T20J2LltPY
                thkSAG0zVoCt+GZhSdcf4T9xLp78tOxRN/PgmGo2hLSO/3Ot
                qTNukDooxPmA7sADaWiZOe6rbrNdFY5QgjBAZ8TlnQ3SSBiS
                M11rygoDei4LTJEROcN6Lq5lL/c=
        <CipherValue>
      <CipherData>
    </EncryptedData>
    <EncryptedData
     Type='http://www.w3.org/2001/04/xmlenc#Element'
     xmlns='http://www.w3.org/2001/04/xmlenc#'>
      <CipherData>
        <CipherValue>
                CyT4UQrOQ1vijcGM8nbKsB1ckUTpBoNH1USfvHTiwhZjN/2+
                bAyEoqzU07IbYXTCKzslnymXivI7waPYZ76V97W2/JqYxRpv
                kBcml4MSulhbekSW+S//jRSjxPuk0FW1POaj7gF9lyWEN+F0
                VpNvqMLceZAVWB7TKTVRx8LGU5l0w=
        <CipherValue>
      <CipherData>
    </EncryptedData>
    <EncryptedData
     Type='http://www.w3.org/2001/04/xmlenc#Element'
     xmlns='http://www.w3.org/2001/04/xmlenc#'>
      <CipherData>
        <CipherValue>
                h3IkkoyhsUL0uuC7MtSyw/xMfWlcKb144rH5EAQQ8vrjrs3B
                1RwmIDF9lYBChHkfghk3eW4Jb6fQrnemykms7ZIAy7dHpxL2
                lC7sJ0rX1UlDjzNoRHKVZo80IZzQ9yP/+mBlbr6C/mD5vE9a
                a2FEEAlFvdGxPeW62fKCD3ZM15kotIRwyf5O+Ja1UJgLN2Ju
                u5AQ3qkpScJBeocSeF207rveeCYPyd+Nh/GrDFzjCndBOB1Y
                V7RXXyUvaDu2PZ55OTwNufUQggpvxpDZUZ7fSOkjzHrDN88Z
                wULKIf6aLBt1M=
        <CipherValue>
      <CipherData>
    </EncryptedData>
  </transaction>
</transactions>
```

Figure 14: An encrypted XML document

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
     <xs:element name="transaction">
```

```
            <xs:complexType>
                  <xs:sequence>
                        <xs:element ref="payer"/>
                        <xs:element ref="price"/>
                        <xs:element ref="cardinfo"/>
                  </xs:sequence>
            </xs:complexType>
      </xs:element>
      <xs:element name="transactions">
            <xs:complexType>
                  <xs:sequence>
                        <xs:element ref="transaction"/>
                  </xs:sequence>
            </xs:complexType>
      </xs:element>
</xs:schema>
```

Figure 15: An XML Schema

4 Implementation and Experimental Results

Many implementations of the XQuery engine exist. For example, Galax (http://www.galaxquery.org) is a lightweight and extensible implementation of XQuery 1.0. Since it closely tracks the definition of XQuery 1.0 as specified by the W3C, it also implements XPath 2.0, which is a subset of XQuery 1.0. Qexo (http://www.gnu.org/software/qexo/) is a partial implementation of the XQuery language that exhibits a good performance because a query is compiled down to the Java byte codes. Saxon (http://www.saxonica.com/) is a complete and conformable implementation of XSLT 2.0, XQuery 1.0, and XPath 2.0. We employ Saxon as the XQuery engine for executing XQuery programs. According to the processing model shown in Figure 8, we implement a translator that enables XQuery programs written by users to query data from encrypted XML documents according to the algorithm listed in Figure 9. We also implement extension objects to perform the decryption processes.

We have conducted experiments to evaluate the performance of querying data from encrypted XML documents. All of the experiments were performed on a PC with a 2.4-GHz Pentium 4 processor, 1024 MB of RAM, the MS Windows 2000 operating system, and Java Development Kit 1.4 (Sun Microsystems). The original XML document had 101 elements: a tree with one root node and its 100 child element nodes, in which each child node was associated with a text node which in turn comprised either 100 or 500 bytes. Table 1 lists the times required to decrypt the whole encrypted XML document and then to query target elements. The processing time increases dramatically with the number of encrypted elements because all encrypted elements need to be decrypted first. For comparison, Table 2 lists the times required to query encrypted documents using the XQuery statements generated by the algorithm listed in Figure 9. The algorithm ensures that only target elements are decrypted regardless of the number of encrypted elements. It is obvious that eliminating redundant decryption dramatically enhances the performance of the query process: increasing the number of encrypted elements in the target element has little effect on the time required to perform the query, which demonstrates the effectiveness of the processing model proposed in the paper.

Total elements in XML file	**Number of queried elements which are encrypted**	**Number of elements that are decrypted**	**Number of encrypted elements**	**Average time (in seconds)**	
				100 bytes*	**500 bytes***
101	**10**	**10**	**10**	**1.8984**	**3.7687**
101	**10**	**20**	**20**	**3.1155**	**6.7626**
101	**10**	**30**	**30**	**4.3640**	**9.8033**
101	**10**	**40**	**40**	**5.2296**	**12.7827**
101	**10**	**50**	**50**	**6.5156**	**15.7282**
101	**10**	**60**	**60**	**7.3671**	**18.6812**
101	**10**	**70**	**70**	**8.6720**	**21.4690**
101	**10**	**80**	**80**	**9.9843**	**24.8675**
101	**10**	**90**	**90**	**11.2171**	**27.3998**
101	**10**	**100**	**100**	**12.1735**	**29.9295**

*Number of bytes to be encrypted in an element

Table 1: The time required to obtain encrypted data by decrypting the whole XML document

Total elements in XML file	**Number of queried elements which are encrypted**	**Number of elements that are decrypted**	**Number of encrypted elements**	**Average time (in seconds)**	
				100 bytes*	**500 bytes***
101	**10**	**10**	**10**	**1.8937**	**3.7672**
101	**10**	**10**	**20**	**1.8968**	**3.7735**
101	**10**	**10**	**30**	**1.8921**	**3.7781**
101	**10**	**10**	**40**	**1.8984**	**3.7702**
101	**10**	**10**	**50**	**1.8077**	**3.7626**
101	**10**	**10**	**60**	**1.9157**	**3.7657**
101	**10**	**10**	**70**	**1.8469**	**3.7656**
101	**10**	**10**	**80**	**1.8531**	**3.7765**
101	**10**	**10**	**90**	**1.1987**	**3.7891**
101	**10**	**10**	**100**	**1.8938**	**3.7828**

*Number of bytes to be encrypted in an element

Table 2: The time required to query encrypted documents using the XQuery statements generated by our algorithm

5 Conclusion

In this paper we have presented a processing model for efficiently querying encrypted XML documents using XQuery. This model requires some documents for optimal querying, including a DSL that specifies how to encrypted the XML documents and the XML Schema of the original XML documents. We can use this model to optimally query the encrypted XML documents, in terms of the computation required for decryption during the query process. Moreover, the experimental results presented here demonstrate that XQuery programs that are transformed according DSL and XML Schema exhibit good performance.

References

Boag Scott, Chamberlin Don, Fernández Mary F., Florescu Daniela, Robie Jonathan and Siméon Jérôme (2005), "*XQuery 1.0: An XML Query Language W3C Working Draft.*" http://www.w3.org/TR/xquery/

Clark J. and DeRose S. (1999), "*XML Path Language (XPath) Version 1.0. W3C Recommendation*," http://www.w3.org/TR/1999/REC-xpath-19991116.xml

Schneier Bruce (1995), "*Applied Cryptography: Protocols, Algorithms, and Source Code in C*," 2nd Edition, published by John Wiley & Sons.

Maruyama Hiroshi and Imamura Takeshi (2000), "*Element-wise XML Encryption.*" http://www.alphaworks.ibm.com/tech/xmlsecuritysuite

"*XML Encryption WG*." http://www.w3.org/Encryption/2001/Overview.html.

Imamura Takeshi, Dillaway Blair, and Simon Ed (2002), "*XML Encryption Syntax and Processing. W3C Recommendation 10 December 2002*." http://www.w3.org/TR/xmlenc-core/

Hwang Gwan-Hwan and Chang Tao-Ku (2001), "*Document Security Language (DSL) and an Efficient Automatic Securing Tool for XML Documents*," International Conference on Internet Computing 2001, 24-28 June, Las Vegas, Nevada, USA, pp: 393-399

Hwang Gwan-Hwan and Chang Tao-Ku (2004). "*An operational model and language support for securing XML documents*," Computers & Security, Volume 23, Issue 6, September 2004, pp: 498-529.

Chang Tao-Ku and Hwang Gwan-Hwan (2003), "*Towards Attribute Encryption and a Generalized Encryption Model for XML*," International Conference on Internet Computing 2003, 23-26 June, Las Vegas, Nevada, USA, pp: 455-461.

Hwang Gwan-Hwan and Chang Tao-Ku (2005) "*DSL Editior*," http://www.xml-dsl.com/DSL_editor_detail.htm

Fallside David C. and Walmsley Priscilla (2004), "*XML Schema Part 0: Primer*," W3C Recommendation, 28 October 2004. http://www.w3.org/TR/xmlschema-0/

International Organization for Standardization, Information Technology- Database Language-SQL-Part 2: Framework (SQL/Framework), ISO/IEC 9075-2: 2003 and Information Technology- Database Language-SQL-Part 2: Foundation (SQL/Foundation), ISO/IEC 9075-2: 2003, http://www.iso.org.

Fischer Charles N. and LeBlanc Richard J Jr. (1991). "Crafting A Compiler with C". The Benjamin/Cummings Publishing Company, Inc.

Galax. Available from: http://www.galaxquery.org

Qexo. The GNU Kawa implementation of XQuery. Available from: http://www.gnu.org/software/qexo/

Saxon. Available from: http://www.saxonica.com/

Sun Microsystems, "*The Source for Java(TM) Technology*," http://java.sun.com

Computer Assisted Assessment of SQL Query Skills

Stijn Dekeyser **Michael de Raadt** **Tien Yu Lee**

Department of Mathematics & Computing
University of Southern Queensland
Queensland, Australia

{dekeyser, deraadt, leet}@usq.edu.au

Abstract

Structured Query Language (SQL) is the dominant language for querying relational databases today, and is an essential topic in introductory database courses in higher education. Even though the language is syntactically simple, relatively concise, and highly structured, students experience many difficulties while learning to express queries in SQL. In recent years a small number of software tools have been proposed to help students learn to write query statements and to assess their querying skills.

In this paper we compare and evaluate existing tools mainly from the perspective of database theory and practice, but also from a pedagogical perspective. Addressing the deficiencies and opportunities uncovered by the evaluation, we then introduce *SQLify*, a new tool that extends the current state of the art by incorporating semantic feedback, enhanced automatic assessment based on database theory, and peer review to arrive at a richer learning experience for students, as well as consistent assessment results and reduced marking for instructors.

Keywords Computer Assisted Learning and Assessment, SQL, Conjunctive Queries, Query Equivalence.

1 Introduction

Structured Query Language (SQL) is the dominant database language today, comprising commands to define relational schema objects (Data Definition Language DDL) as well as provisions to manipulate data (Data manipulation Language DML). In most introductory level database courses in higher education, learning to write DML query expressions in SQL receives a significant amount of attention. Students are not only taught to write syntactically correct statements, but more importantly to translate a natural language question into a semantically correct SQL expression. This learning process is often difficult.

Researchers [10, 11, 14, 17, 19] have identified several common problems that students encounter while learning SQL. We list some of these:

- It is a burden for students to memorize the database schema, possibly resulting in erroneous solutions due to incorrect table or attribute names. This burden also misleads the students to focus on low-level syntax at the expense of high-level query definition.
- Many students misunderstand the basic elements of SQL and first order logic and the relational data model in general. They have trouble grasping concepts such as joins, universal quantification, grouping and aggregation, and some set operations.
- Students may incorrectly perceive a query problem as being easy [19]. Thus, they need experience thinking about the semantics of questions and expressing them in SQL. Part of the problem is that many common, useful, simple to understand, and potentially easy to express queries are outside the bounds of relational completeness. Conversely, many queries that *can* be expressed with a relationally complete language are difficult to compose or comprehend.
- The declarative nature of SQL is rather difficult for many learners to grasp. It requires them to think sets rather than steps.

To overcome some of these problems, students are typically provided with a relational database management system in which they can experiment and increase their skills. However, this approach only provides students with immediate feedback regarding the syntactical correctness of their expressions[1]. This is insufficient to prepare them for assessment, where course instructors usually place more importance on semantical correctness.

From the perspective of students a simple environment is needed in which they can test query expressions and receive immediate feedback regarding both syntax and semantics.

On the other side of the teaching–learning divide, course instructors often desire a tool that helps them teach querying skills while also enforcing consistency in grading and helping to reduce their marking load, freeing them for more effective teaching tasks. For both parties, a tool that helps raise learning to higher orders of thinking (particularly evaluation) can improve educational outcomes [2].

Because relational query languages are not Turing complete, and because important subsets of these languages allow decidability of query equivalence, tools can be constructed that provide immediate syntactic and semantic feedback. In recent years a small number of tools that offer partial feedback have been

[1]Students may also get some idea of the semantic correctness of their queries from evaluation, but a correct query answer for one specific database instance may be misleading the student to think the query is correct in general. In addition, students don't always know what the correct answer for a given instance should be.

proposed. We evaluate a number of these in the next section.

Motivation. (1) To build on the current state of the art and improve the learning experience for students attempting to master SQL, while helping instructors in assessment. (2) To inform database researchers and teachers of educational SQL tools.

Contribution. This paper contains two distinct contributions. First, we compare and evaluate existing computer assisted SQL tutoring and assessment tools from the perspectives of Databases and Computer Science Education. Secondly, we propose *SQLify* as a tool building upon and enhancing these existing solutions, especially in the areas of improved computer assisted assessment and peer review.

Organization. In Section 2 we review existing tools used for teaching and assessing SQL statements. In Section 3 we then give a detailed description of *SQLify*, before presenting an example run-through of the system in Section 4. Some implementation details, especially concerning automatic query assessment, are given in Section 5. We conclude with a summary and future work in Section 6.

2 Evaluation of Existing Tools

In this section we review a number of tools that are well described in literature.

- *eSQL*, proposed in 1997 in [10] to help teach the concept of query processing. It is not used for query evaluation or assessment.
- *SQL-Tutor*, developed at the University of Canterbury, Christchurch, in 1998 [14]. It provides semantic feedback but is not used in assessment.
- *AsseSQL*, a tool created at the University of Technology in Sydney in 2004 [16]. It provides binary grading of queries submitted by students.
- *SQLator*, a tool created by the University of Queensland also in 2004 [17]. This tool is similar to *AsseSQL*.

A number of other tools and systems exist (e.g. [9] and [11]) but are only partially described in scientific research literature. Interestingly, however, very few papers discuss database teaching in general and most of them are not known to database researchers as they are published in computer science education literature.

We now offer a cross-disciplinary review of these tools, first evaluating the systems from a database theory and practice perspective, and then from a pedagogical perspective. The review is summarized in Figure 1.

2.1 Database Perspective

The tools we studied for this paper were primarily created and described from a computer science education point of view, with minor regard to relational database theory. None of them provide detailed implementation information.

Both *AsseSQL* and *SQLator* [16, 17] use heuristical methods to evaluate whether queries entered by students in a test are correct. This involves running the submitted query on a test database, and comparing the output with that of the query included in the definition of the question. In Relational Algebra terms, the condition that is tested is

$$(a(I) - s(I)) \,\cup\, (s(I) - a(I)) = \phi \qquad (1)$$

where a is the correct query as supplied by the course team, s is the query submitted by the student, and I is an instance of the database supplied by the course team. This test, however, is only approximate: it is possible for students to cheat by creating simple queries that produce the desired result, especially when they are shown the database instance. Sadiq and others in [17] report that their *SQLator* system appropriately marks a query as correct in 95% of cases when the test queries are relatively easy. This result is sufficient for a beginner's course on SQL, but may be more problematic for more advanced courses. Also, the success of the heuristic depends in part on the database instance used in the test; a badly designed instance reduces the level of correctness of this method.

In [16] Prior and Lister have therefore proposed extending their *AsseSQL* tool to run an additional test on a second database instance not known to the students. While this indeed increases the correctness of evaluation, it is still only a heuristic test.

In database theory it is well known that the class of Conjunctive Queries has the important property that it is decidable whether two queries are equivalent. The CQ class is a significant subset of SQL excluding the set operators and grouping statements. In the introductory *Database Systems* course at the University of Southern Queensland, more than 70% of the time spent on teaching SQL is reserved for such queries. Hence, for this type of query, a computer assisted assessment tool should be able to evaluate correctness of submitted queries with 100% accuracy. For queries that are not in CQ, a heuristic approach can still be used, but any automatic grading tool will need to flag such cases so that the lecturer can intervene appropriately. In Section 3 we detail use of query equivalence decidability results to improve the accuracy of computer-based assessment, and to allow automatic grading of reviews performed by students for their peers where possible.

The existing literature also does not address some practical considerations with regard to database systems. For example, the use of the *distinct* keyword or sorting in a query makes it impractical to test equivalence using only the heuristic described above[2]. Furthermore, both *AsseSQL* and *SQLator* seem vulnerable to SQL injection attacks. These include attempts to make unauthorised modifications to a database by taking advantage of the level of access provided by the interface. Care must be taken to check or rewrite a submitted query before it is evaluated by the database server.

None of the systems we reviewed have support for teaching or assessing Relational Algebra expression writing. We argue that adding such support is very valuable, for two reasons. Firstly, most introductory level relational database textbooks that we are familiar with (e.g. [5, 12, 20]) include the teaching of relational algebra, often *before* teaching SQL. They do so for a variety of reasons but mostly because students who understand the relational algebra are more likely to write better SQL queries: "*relational algebra is the key to understanding the inner workings of a relational* DBMS*, which in turn is essential in designing SQL queries*" [12]. Secondly, the techniques used

[2]CQ equivalence testing is also not sufficient for this purpose.

Feature	*eSQL*	*SQL-Tutor*	*SQLator*	*AsseSQL*	*SQLify*
Modelling of student to individualize instructional sessions	✗	✓	✗	✗	✗
Visualization of database schema	✗	✓	✗	✗	✓
Visualization of query processing	✓	✗	✗	✗	✓
Feedback on query semantics	✗	✓	✗	✗	✓[a]
Automatic assessment (using heuristics)	✗	✗	✓	✓[b]	✓[c]
Automatic assessment (using CQ query equivalence)	✗	✗	✗	✗	✓
Use of peer review for assessment	✗	✗	✗	✗	✓
Relational Algebra expressions support	✗	✗	✗	✗	✓[d]
Special treatment of DISTINCT and ORDER BY	✗	✗	✗	✗	✓
SQL-injection attack countermeasures	✗	✗	✗	✗	✓

Figure 1: Comparison of existing tools and *SQLify* detailed in the remainder of this paper. (a) in practice mode only. (b) on two instances (proposal only). (c) for queries not in CQ. (d) planned for next version.

in automated SQL teaching and assessment tools can be readily used for the relational algebra as well, requiring only a user-friendly (and, desirably, pedagogical) interface for entering relational algebra statements, and additional logic to convert students' algebra expressions into SQL, the latter of which is a well-documented procedure.

2.2 Pedagogical Perspective

eSQL [10] was one of the earliest tools proposed for teaching database concepts. This is a system similar to a normal query interface except that the response to a SELECT statement is not merely to show the result, but also to show a sequence of images giving a step-by-step account of how the query result is determined. Hence, *eSQL* visualizes query processing, at least at the conceptual level. This helps students develop a mental model and enhances students' understanding of the semantics of SQL. One of the steps being visualized is the creation of the cartesian product of the input tables. Since the number of rows in the intermediate result may be far too large to show, the system uses an ingenious algorithm to choose a sample row set for display.

The main contribution of *eSQL* is therefore pedagogical. The tool is not meant to analyze queries submitted by students, nor is it used in assessment.

SQL-Tutor [14] is a knowledge-based system that supports students in learning SQL. It focuses on the individualization of instructional sessions towards a particular student, by developing a model of the student's knowledge, learning abilities and general characteristics and tailoring instructional actions to the student's needs. *SQL-Tutor* is also an Intelligent Teaching System designed as a guided learning environment, which helps students in overcoming the difficulties in learning SQL. Students are given opportunities to discover things by themselves; they can learn by doing.

A major strength of *SQL-Tutor* is that this system gives meaningful feedback on the semantic correctness of queries in addition to feedback concerning syntax. Moreover, there are five levels of feedback in the system, yielding increasingly detailed information.

Another feature of *SQL-Tutor* is the visualization of the database schema. This removes cognitive load for students, allowing them to focus on higher level query definition problems instead of low-level syntax.

However, *SQL-Tutor* does not visualize the way a query is executed as *eSQL* does. In addition, the tool only focusses on helping students practice before assessment and does not help instructors in conducting assessment.

Turning to the systems which support assessment, both *SQLator* and *AsseSQL* [16, 17] apply only *binary* grading to queries submitted by students, and do not provide comments or suggestions for improvement. While Prior and Lister [16] argue the sufficiency of this *right-or-wrong* approach, binary grading does not correct students' misunderstandings or encourage further learning.

As well as giving students query problems to solve, *AsseSQL* also shows the desired result of the query they are to write. This is justified as an attempt to overcome students' poor English skills, but creates an unauthentic setting for student learning, as database programmers typically do not know the result of a query before submitting it to the server.

Both *AsseSQL* and *SQLator* create only a single channel of communication between the student and the instructor via the system. No other forms of communication (e.g., peer to peer) are mentioned as being part of these systems or used along-side these systems.

None of the tools we examined use *peer review* as part of learning and assessment. According to Saunders [18] peer learning is advantageous as "it offers the opportunity for students to teach and learn from each other, providing a learning experience that is qualitatively different from the usual teacher-student interactions". Peer review can take several forms. One form takes a student's submission and allows it to be reviewed by a number of student-peers, a process overseen by an instructor. Peer review has been successfully incorporated in the assessment of student work in various fields, including computing [6, 7, 13]. Peer review allows students to evaluate the work of others which requires higher order thinking skills [2] through evaluating the work of peers and reflecting on their own work. With peer review, students also receive feedback from more than one source enriching the learning experience for students. Receiving feedback from peers can encourage a community of learning [3] which can in turn further encourage higher order thinking. Peer review involves students in the assessment process, encouraging increased engagement in the course and ultimately improved learning outcomes [6]. Peer review, when used as an assessment tool, can also reduce the assessment workload of instructors.

3 *SQLify*

Having compared and evaluated existing computer assisted learning and assessment tools both from a Computer Science Education and a Database Theory point of view, we now turn to the description of *SQLify* (pronounced as *squalify*) which aims to improve existing solutions on several different fronts.

From the discussion in the previous section, it is clear that combining semantic feedback, an enhanced automatic assessment algorithm, and peer review will produce better outcomes for students and instructors alike. Specifically, the following requirements have driven the design of *SQLify*:

- Provide rich feedback to students in an automated and semi-automated fashion;
- Reduce the need for recall for students by presenting the relevant relational schema;
- Illustrate the execution strategy for queries submitted by students to deepen their understanding of database systems;
- Employ peer-review to enhance learning outcomes for students (through students conducting evaluations and receiving feedback from more sources);
- Use a query equivalence testing algorithm combined with peer review effectively to yield a wider range of final marks;
- Automatically judge the accuracy of reviews performed by students as part of their assignments;
- Reduce the number of necessary moderations conducted by instructors, freeing them for other forms of teaching;
- Increase the consistency of marks allocated to students.

Hence, the main focus of *SQLify* is computer assisted practice and assessment using a sophisticated automatic grading system in combination with peer review.

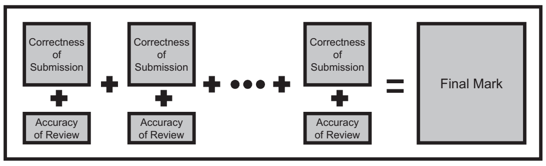
3.1 Use of *SQLify*

The *SQLify* system is intended to assess a student's query writing skills through an online interface in the context of assignments and preparing for assignments[3]. Student use of the system can be seen to fall into a series of phases.

1. Trial and submission
2. Reviewing peers' submissions
3. Receiving feedback and marks.

Students will submit solutions to a number of problems. The value of their submission will be judged by peers, the *SQLify* system and ultimately by the instructor (see Figure 2).

Students complete reviews of (usually two) other students submissions for which they are awarded marks. The accuracy of their submission determines the mark they receive for reviewing.



Correctness of Submission
Accuracy of Review
Correctness of Submission
Accuracy of Review
Correctness of Submission
Accuracy of Review
Final Mark

Figure 2: Components of a student's mark.

Finally the marks they received for submission and the accuracy of their reviews is summed to form a final mark.

The following subsections describe in detail the three phases mentioned above.

3.1.1 Trial and Submission

Students are able to develop and trial their query answers to a specific set of problems using *SQLify* and immediately see how the automatic grading system evaluates their work. The *SQLify* system will give one of (a limited set of) the levels of correctness shown in Figure 3. Students may practice query problems indefinitely prior to starting work on assignments. The mark they are shown during this trial period is not necessarily what they would receive from the instructor for the correctness of their submission in assignments; this is given later by the instructor under advisement of the student's peers and the *SQLify* system. When the student is happy with their work they may proceed to submitting query answers to assignment problems.

Students completing assignments using *SQLify* will typically be given a number of English-language problems (say three to five) that he or she would translate to SQL[4]. The problems are well defined descriptions of authentic, real world problems. Students' query answers are submitted through a web form; a screenshot of *SQLify* is shown in Figure 4.

[3]The system is not meant for use in the context of examination.
[4]Or Relational Algebra, as detailed in Section 6.

Level	Description	Students can use	System can use	Instructor can use	Example value
L0	Syntax, output schema, query semantics incorrect	✓	✓	✓	0%
L1	Syntax is correct, schema and semantics incorrect	✓	✓	✓	20%
L2	Syntax and schema correct, semantics are incorrect	✓	✓	✓	30%
L3	Syntax and schema correct, semantics largely incorrect			✓	40%
L4	Syntax and schema correct, semantics seem largely incorrect (not sure)	✓			70%
L5	Syntax and schema correct, semantics just adequate			✓	80%
L6	Syntax and schema correct, semantics seem largely correct (not sure)	✓	✓		90%
L7	Syntax, schema, and semantics are correct	✓	✓	✓	100%

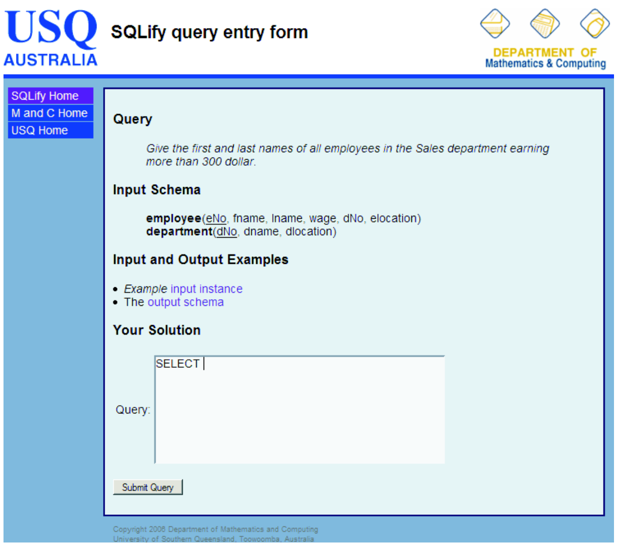
Figure 3: Levels implied by evaluation sentences. Different levels may be used by reviewing students, the *SQLify* system, and by instructors. Internal assessment values (last column) are *example* values for each level; they may be changed by instructors using *SQLify*.

The problems that students are asked to solve relate to differing database schema, so the student is presented with the correct schema (and a sample instance) for each individual problem. The student can also be supplied with hints and comments, and also with the desired output schema for the query (not the desired output instance), if so determined by the creator of the problem.

Once a query is submitted to the system it is checked for SQL injection attacks. First, tables referenced in the FROM clause of the submitted statement will need to appear in the source database schema, or the query will be rejected. Second, the WHERE clause will be analyzed and possibly rewritten using mainstream SQL injection countermeasures.

In *SQLify*'s assessment mode, students will not be notified if their submissions contain queries that are syntactically incorrect (although they should have been able to determine this themselves by trialing their submission in a database). Demonstrating understanding of the particular SQL syntax of the database system that is used by *SQLify* is a part of the assignment. Also, students may use any resources they like while answering the problems, including SQL language reference guides.

Students receive feedback about their submission in the final phase (see Section 3.1.3).



SQLify query entry form

Query

Give the first and last names of all employees in the Sales department earning more than 300 dollar.

Input Schema

employee(eNo, fname, lname, wage, dNo, elocation)
department(dNo, dname, dlocation)

Input and Output Examples

- *Example* input instance
- The output schema

Your Solution

Query: SELECT

Copyright 2006 Department of Mathematics and Computing
University of Southern Queensland, Toowoomba, Australia

Figure 4: *SQLify* screenshot of the query entry form.

3.1.2 Reviewing Peers' Submissions

After submitting, most students are able to immediately proceed to complete reviews allocated to them. A small pool of early-submitting students (usually four) must wait until enough submissions have accumulated before they can proceed to reviews. Such early-submitters are informed that they must wait and when this minimum pool has been reached the system will automatically allocate reviews to the initial pool and inform them by email that they may start reviews.

This single step submit-review process has been successfully applied [6] and has several advantages over a two step process (submit before deadline, review after first deadline and before a second deadline):

- only one deadline is needed,
- the majority of students are not required to return to the site for the sole purpose of completing reviews,
- students review the task they have just completed,
- students receive feedback from peers shortly after submission, and
- students can work ahead in the course.

The disadvantage of a *single phase* review allocation system is that it must distribute review allocations in a way that maintains anonymity. If students can predict who they will review, collusion between students is possible. This can be countered by complicating the review allocation process and keeping its workings secret, by requiring each submission to be reviewed by more than one peer, and by comparing the accuracy of a student's review to that suggested by the system.

When the system has allocated reviews to a student, reviewing can commence. The student is presented with a similar screen to what they used to input their query answer during the initial submission phase, but where they were previously able to enter their answer the system now shows a read-only query given by a peer. The reviewing student additionally sees the result of applying the query on the relevant database instance. The reviewing student then selects a level described by a sentence from the list shown in Figure 3 that best describes their assessment of the correctness of the query answer. The list of possible levels given in Figure 3 shows all available levels of which the reviewing student may choose levels marked with a tick in the column titled “Students can use”. No corresponding internal values are shown to the reviewing student. Reviewing students may express uncertainty by choosing a sentence that includes “not sure”. This allows the system to assign a wider range of marks to reviews, but is also used to flag potential problems that need to be moderated by an instructor.

By linking automatic assessment of queries with reviews given by students, it is not only possible to evaluate the correctness of queries, but also the accuracy of reviewers in judging that query. Students will review the work of two peers knowing that the accuracy of reviews they perform will also be assessed.

A student’s review accuracy should be marked high when the level they selected for a peer’s query answer is very similar to the level ultimately determined for that query answer by the instructor. Conversely, accuracy should be marked low when it differs greatly from the instructor’s correctness mark. Hence, the formula for marking accuracy of a review performed by a student is quite simple.

$$\text{accuracyMark} = 100 - |\ \text{correctnessMark} - \text{studentMark}\ |.$$

In other words, the mark given to a reviewer for the accuracy of their review depends on the difference to the correctness mark assigned by instructor. Note that this formula has the additional affect that when a student has signaled uncertainty (by picking level L4 or L6) they will not be awarded full marks for this review.

Giving fellow students a false high or low level evaluation which differs for the mark applied by an instructor will lose marks for the reviewing student.

As well as judging correctness levels for query answers, reviewing students are also required to leave a comment. Students are encouraged to give comments of praise or positive suggestions for improvement. This is arguably the most valuable part of the reviewing process for both the reviewer and the reviewee.

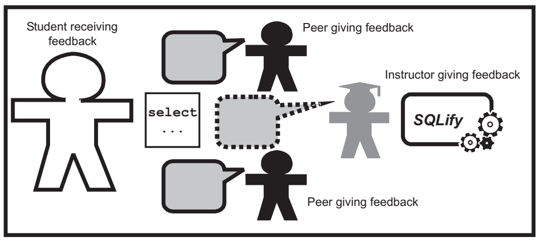


Figure 5: Feedback received by the student.

For the reviewer this is an opportunity to evaluate the work of a peer and in doing so, reflect on their own work. This requires higher order thinking skills [2] which will hopefully encourage greater learning outcomes.

For the reviewee receiving peer feedback means they will receive feedback from more sources than just the instructor or the system. The information contained in comments can encourage a more personal relationship among students (even anonymously) and between instructors and students, thus helping to form a community of learners.

For instructors, adding a comment allows elaboration on why a student may have lost marks and positive encouragement on their progress. The instructor may draw on a list of previously created comments to speed up the moderation process. This also provides consistency when multiple instructors are performing moderations.

Another benefit of this system is to allow students to flag peer reviews they believe to be incorrect for instructor intervention. Although quite often the instructor would be moderating such cases, this feature allows the student to express unhappiness with a review. This can remove some anxiety related to having their work assessed, in part, by algorithms and peers.

Note that the manner in which *SQLify* uses peer review is based on an existing, fully evaluated and successful peer-review system [6, 7].

3.1.3 Receiving Feedback and Marks

When all reviews of a student’s work are complete, the instructor allocates a mark for the student’s work based on the levels suggested by the *SQLify* system (which does so on the basis of its own algorithm and marks suggested by peers). Instructors must attend to submissions that have been assessed differently by each peer or by the system. Past experience [7] has shown that in at least half of normal submissions, peers alone are able to achieve non-conflicting reviews, so this means moderation is most likely to be unnecessary in these cases. Additionally, in many cases *SQLify* can determine a level for a solution with absolute certainty so this further eases the marking load of the instructor.

Past experience also suggests that it is important that students sense the instructor’s involvement in the assessment process [7]. They see the instructor as an authority and feel they deserve the attention of the instructor during the assessment process. It is possible for good students who produce excellent work, to be assessed equally by peers and the *SQLify* system. In such cases the instructor may elect to assign a mark based on the agreed standard of the work without performing moderation. If a student achieves this consistently through the semester, they may miss the instructor’s input in their assessment; they may then feel cheated by the assessment approach. It is possible to track how many times a student has been moderated by an instructor and set target levels of moderation at various points through the teaching period. This way each student can be satisfied with the attention they are receiving while still reducing the marking load on instructors.

One of the clearest benefits of using a single-step peer review system it that students receive feedback about their submission as soon as a peer has completed their review. Compared with a normal instructor marked assignment where students must wait until after the assignment deadline for feedback, previous use of the approach suggested here returns feedback to students within hours [7].

Once the peer review process is completed and the instructor has assigned marks to students, the *SQLify* system can calculate a final mark for each student.

The system suggests a final mark for a student’s assignment. It does so by summing both the correctness marks for each query answer and accuracy marks for the reviews conducted by that student (see details in

Section 3.2.1). The weighting of correctness and review accuracy for each problem in each assignment could be varied according to the effort for each. An example would be weighting the correctness marks to 70% of the entire assessment and review accuracy marks to 30%.

The instructor then chooses to accept or modify the suggested mark (see Section 3.2.2). Such marks may be released individually by the instructor or *en masse*.

3.2 Details of how a correctness mark is determined

The three phases described above result in a process of assigning marks to queries and reviews. Three variables are kept per submitted query answer for each student: a system correctness mark sys, and two correctness marks from peers $\mathsf{std1}$ and $\mathsf{std2}$. With the aid of *SQLify*, the instructor then uses these marks to determine an overall *correctness mark* for each query answer the student submits.

3.2.1 How the system determines its correctness mark (Determining values for sys)

The levels below are taken from Figure 3.

L0 The submission is syntactically incorrect
The submitted query is sent to the database engine which returns a syntax error. The system is *certain* that the query is syntactically incorrect, so the internal value for sys is L0.

L1,L2 The submission is syntactically correct
The query is accepted by the database, upon which the system checks whether the output schema is the same as the one produced by the solution query (supplied by the instructor). The system can determine this exactly, and assigns an internal mark of L1 for sys if the condition is not met, and L2 if the condition is satisfied.

L6 The submission produces a result that is probably correct but needs to be checked or compared with peer marks
The query passes the output schema test, and now undergoes examination of its semantics. If the query does not belong to the Conjunctive Query (CQ) class, only a heuristic approach is possible. If the heuristics determine that the query is correct, there is only a small chance that in fact the query is not semantically correct (see Section 2.1). Hence, the internal value for sys is set to L6 if the test is successful, and L2 if it is not.

L7 The submission is certainly correct
In case the query belongs to the CQ class, it is possible to algorithmically decide whether it is semantically equivalent to the set solution query. If it is, the internal value for sys is set to L7, otherwise sys will be reset to level L2.

As is clear, there is a significant gap between levels L2 and L6; levels L3 to L5 cannot be chosen by the system. This is because the *SQLify* system cannot determine how good or how bad a query is that has been proven to be semantically incorrect. Hence a combination of peer review and instructor intervention is used to come up with a wider range of accuracy marks. Thus, as well as enhancing the learning experience of students, the peer review process also plays a practical role in moderating the mark proposed by the system and in flagging possible problems to the instructor.

3.2.2 How the instructor determines a correctness mark for an answer

SQLify calculates a suggested correctness mark for each submitted answer by using the marks given in reviews by students when its own automatic assessment is not sufficient. In many cases the system can suggest a mark with great certainty which the instructor can accept. Instructor moderation is needed when the system is uncertain about the student's submission or if there are conflicts between peer reviews or between reviews and the mark of the system. The following procedure is used by the instructor to apply the mark suggested by *SQLify*.

$\mathsf{sys} \leq \text{L1}$
(The submission is incorrect)

In this case, the suggested mark will be same as sys, so possible internal values for the correctness mark are L0 or L1. The system is always right in these cases, so no student review marks need to be used to determine the correctness mark and no instructor intervention is necessary.

$\mathsf{sys} = \text{L2} \wedge \text{L2} \leq \mathsf{std1} \leq \text{L4} \wedge \text{L2} \leq \mathsf{std2} \leq \text{L4}$
(The submission is largely incorrect)

In this case, both reviewing students agree with the system that the submitted query is semantically incorrect while the syntax and output schema are correct. The suggested mark will be the average of both student reviews rounded up to the nearest level. So, possible internal values for the correctness mark are L2, L3 and L4 as chosen by the instructor.

$\mathsf{sys} = \text{L2} \wedge \neg(\text{L2} \leq \mathsf{std1} \leq \text{L4} \wedge \text{L2} \leq \mathsf{std2} \leq \text{L4})$
(There is a conflict between reviewers and the system)

In this case either one or both of the reviewing students disagree with the system. Hence, instructor intervention is appropriate to moderate the conflict. The correctness mark will be determined by the instructor and can be taken from L2, L3, L4, or L5 as suggested by the system. The correctness mark cannot be higher than L5 because *SQLify* has determined the query to be semantically incorrect. Also, the instructor can choose from the two additional levels L3 and L5 because he is more experienced than the reviewing students.

$\mathsf{sys} = \text{L6} \wedge (\mathsf{std1} \leq \text{L4} \vee \mathsf{std2} \leq \text{L4})$
(The system suggests the answer is probably correct but the reviewers disagree)

The system could only heuristically determine that the query semantics are likely to be correct, while at least one of the reviewing students believes that the query is incorrect. In this case, intervention is needed by an instructor, who may choose any of the suggested levels L0, L2, L6, and L7. The chance is rather low that the query is indeed incorrect, so the first two levels are only used when the instructor believes that the student may have attempted to cheat (L0) or only accidentally confused the system (L2).

$\mathsf{sys} = \text{L6} \wedge \mathsf{std1} \geq \text{L5} \wedge \mathsf{std2} \geq \text{L5}$
(The system thinks the query is probably correct and the reviewers agree)

Both students believe the query may be correct, and *SQLify* has also determined that this is likely. Intervention is not necessary as students are motivated to conduct accurate reviews. The system suggests a correctness mark of L7 (100%).

sys = L7
(The system indicates that the answer is certainly correct)

The system has incontrovertibly determined that the submitted query is correct, hence no student review marks are needed to determine the correctness mark, and no instructor intervention is necessary. The system strongly suggests a correctness mark of L7 (100%).

Based on the system's recommendation the instructor will set a correctness mark for each submitted answer. Each correctness mark is summed together with the marks calculated for the accuracy of reviews submitted by a student to form a final mark which can be released to the student.

4 Examples

To illustrate the workings of *SQLify*, two query problems are presented together with a description of how they are evaluated using *SQLify*. The assessment process to see how students' submissions are evaluated is followed through to a final mark.

4.1 Query Problems

The example problems make use of a database with the following schema[5].

employee(eNo, fname, lname, wage, dNo, eloc)
department(dNo, dname, dlocation)

The first query problem (QP1) is an example of a Conjunctive Query (a problem in class CQ). In this class it is possible to conclusively determine if a supplied query is correct without employing heuristic comparison.

Give the first and last names of all employees in the Sales department earning more than 300 dollars.

The instructor supplies a solution query that will be used by the system to test queries submitted by students.

```
SELECT fname, lname FROM employee E,
department D WHERE E.dNo = D.dNo AND
dname = 'Sales' AND wage > 300; (QP1)
```

The following are two queries submitted by students. They are both different to the solution presented by the instructor, but both can be proved to be semantically equivalent to the instructor's solution query and are therefore considered correct. Correctness marks given by the system and two peers are also shown.

Submitted query	sys	std1	std2
`SELECT fname, lname FROM employee JOIN department ON dNo WHERE dname = 'Sales' AND wage > 300;` (SA1)	L7	L6	L7
`SELECT fname, lname FROM employee E WHERE wage > 300 AND EXISTS (SELECT * FROM department D WHERE E.dNo = D.dNo AND dname = 'Sales');` (SA2)	L7	L7	L4

[5]Instructors submitting their own query problems can submit their own schemas and instances.

The following query is an incorrect query answer to the above problem (QP1).

Submitted query	sys	std1	std2
`SELECT fname, lname FROM employee E WHERE dname = 'Sales' AND wage > 300;` (SA3)	L2	L6	L4

The next problem (QP2) involves a query that is not in CQ class.

List all locations where there is either an employee or a department.

The following is an instructor's solution query for this problem.

```
(SELECT eloc FROM employee) UNION
(SELECT dlocation FROM department);
(QP2)
```

An incorrect solution to this problem is given next.

Submitted query	sys	std1	std2
`SELECT loc FROM employee, department WHERE loc = eloc OR loc = dlocation;` (SA4)	L2	L2	L3

4.2 Marking query correctness

When the system has evaluated a submitted query and peer reviews are complete for that query the system will recommend a mark to the instructor. The instructor can then assign a *correctness mark* for the query.

The table below shows, for each row, the correctness marks for a particular query submitted by a student, as given by the system itself (sys), and two peers reviewing the query answer (std1 and std2). In addition, a suggested mark is shown calculated by *SQLify* on the basis of sys, std1 and std2 using the procedure described in Section 3.2.2. Finally, the correctness mark assigned by the instructor is listed; this mark may or may not be the same as the suggested mark.

Student	Problem	Submitted query	System mark (sys)	Reviewer 1	Mark (std1)	Reviewer 2	Mark (std2)	Suggested mark	Correctness mark
1	QP1	SA1	L7	3	L6	5	L7	L7	L7
1	QP2	SA4	L2	4	L2	5	L3	L3	L3
				...					
4	QP1	SA2	L7	1	L7	3	L4	L7	L7
5	QP1	SA3	L2	1	L6	2	L4	L4	L4

The internal values corresponding to levels given in Figure 3 are not hard-coded into the system. The instructor using *SQLify* can set these values during use of the system. Hence the levels given in the last column will translate into different scores for queries as determined by the instructor.

4.3 Checking accuracy of reviews

The following table lists one row per peer review that is performed in the context of an assignment. The first row, for instance, shows that student 1 was a reviewer for a query (SA2) submitted by student 4 in answer to query problem QP1. Student 1 gave this query answer a correctness mark of L7. The correctness mark for the submitted query answer ultimately given by the instructor was also L7. Hence, the *accuracy mark* for this particular review is 100. For the next review performed by this student there is a difference between the correctness mark given by this student and the correctness mark set by the instructor. This difference causes their mark for accuracy to be reduced.

Reviewer	Reviewee	Problem	Submission	Reviewer's mark	Correctness mark	Difference	Accuracy mark
1	4	QP1	SA2	L7	L7	0%	100%
1	5	QP1	SA3	L6	L4	20%	80%
			...				

4.4 Calculating a final mark

The last table below summarizes the various marks that a particular student received for various query problems and for the reviews performed. A weighted final mark is given in the last row using the suggested weightings of 70% for correctness and 30% for accuracy of reviews.

Student: 1		
Correctness marks	QP1	100%
(Weight 70%)	QP2	50%
	QP3	70%
Review accuracy	QP1	100%
(Weight: 30%)	QP2	80%
	QP3	50%
Final Mark		74%

5 Implementation Aspects

The *SQLify* system has been implemented and is currently undergoing tests to prepare for first use in an undergraduate database systems course offered later this year.

The current version of the system was implemented in a standard LAMP (Linux, Apache, MySQL, PHP) environment and is integrated with university systems to manage assignment assessment for enrolled students. However, the tutoring part of *SQLify* is open for outside use, and can be accessed at [8].

While there are several implementation issues that are worthy of description, we here only focus on the process of testing query equivalence.

5.0.1 Heuristic Testing

When students submit a query for assessment, *SQLify* first rewrites it to counter SQL injection attacks and to align it with the actual database table names stored in MySQL. As will become clear, the rewriting is also useful for other reasons. Subsequently the system checks syntactic correctness and the correctness of the query's output schema. If either of these fail, evaluation stops and sys is set to either L0 or L1.

After rewriting, the heuristic test given in Formula 1 on page 2 is performed. The same measures used in *AsseSQL* and *SQLator* are taken to increase the reliability of the heuristic test.

If the heuristic test is negative, meaning that for the specific database instance or instances stored in MySQL the result of the query is different from that of the correct query (as supplied by the instructor), evaluation stops and the submitted query's sys value is set to L2.

In case the test was positive, but queries involve set operations, grouping, or aggregation, the analysis stops, and *SQLify* sets the sys variable to L6. Peer reviews and instructor input will then further refine the score for the submitted query.

5.0.2 CQ Equivalence Testing

If the submitted query is in the Conjunctive Query class automatic evaluation continues. This is the case if the query was able to be rewritten (in the previous phase) into the following SQL form:

$$\begin{array}{l}\text{SELECT } A_1, \ldots, A_n \\ \text{FROM table}_1, \ldots, \text{table}_t \\ \text{WHERE } \langle \textit{condition}_1 \rangle \text{ and} \ldots \text{and } \langle \textit{condition}_c \rangle\end{array}$$

where the *condition$_i$*, for $1 \leq i \leq c$, consists of only join conditions and equality comparisons, that is $X = Y$, where X is a variable (column) and Y is either a variable or constant.

Note that some SQL queries containing WHERE EXISTS subqueries can be rewritten in this form. Query SA2 given in Section 4 is such a case, and can be rewritten as QP1 which conforms to the SQL subset given above.

Testing the equivalence of conjunctive queries is a basic problem on which query optimizers are partly based. Since checking equivalence of two queries can be done by testing the mutual containment of the queries involved, the containment problem has been studied extensively by many researchers.

Under the set semantics, the containment of conjunctive queries using only equality tests was fully solved by Chandra and Merlin [4], using the concept of containment mapping. There has been extensive work on the testing of set containment of inequality conjunctive queries, and also on bag semantics with either equality or inequality tests.

Recently, the idea of using a finite set of canonical databases to represent an infinite set of databases is used by Penabad [15] to develop a general procedure, called Query Containment Checker (QCC), to test the containment problems of both equality and inequality conjunctive queries, under both set and bag semantics.

For *SQLify* we decided to implement an algorithm using tableaux representation of expressions [1] to test the equivalence of equality conjunctive queries under set semantics. The tableaux can be easily constructed from the previous query rewriting phase. We will leave all other fragments of the general containment problem as an extension for future versions of our system.

6 Conclusion and Future Work

In this paper a small set of existing tools used for teaching and assessing SQL writing skills was reviewed. The tools were evaluated both from Computing Education and Database perspectives, noting possible areas of enhancement.

Secondly, we proposed a comprehensive new tool for the teaching and assessment of SQL writing skills. Central to the system is the use of an intricate automatic grading system and peer review. The main reason for including peer review is to offer students a richer learning experience. Additionally, peer reviews assist in the assessment of assignments.

SQLify uses a relatively complex method to assign final grades to assignments, designed to (1) yield a much wider range of grades than simply *correct* or *incorrect*, (2) utilize database theory to arrive at a computer assisted assessment, (3) set high quality demands for student reviews, yielding a better learning environment, and (4) reduce the number of necessary interventions performed by course instructors.

Regarding future work, we are currently preparing *SQLify* for use in a live course by the end of 2006. We will then evaluate the usefulness of the system as perceived by students and instructors. Any change in student outcomes will be measured.

To evaluate relational algebra expressions two additions to the system (planned for the next version of *SQLify*) are needed: first, the implementation of an interface that helps students construct syntactically correct algebra expressions, and second the implementation of the algorithm that translates the submitted algebra expression to an equivalent SQL statement. The generated statement is then processed in the same way as a normal SQL statement.

References

[1] Serge Abiteboul, Richard Hull and Victor Vianu. *Foundations of Database Systems.* Addison Wesley, 1997.

[2] B. Bloom. *Taxonomy of Educational Objectives.* Edwards Bros., Ann Arbor, Michigan, 1956.

[3] C. Brook and R. Oliver. Online learning communities: Investigation a design framework. *Australian Journal of Educational Technology*, Volume 19, Number 2, pages 139–160, 2003.

[4] Ashok Chandra and Philip Merlin. Optimal implementation of conjunctive queries in relational data bases. In *Proceedings of the ninth annual ACM symposium on Theory of computing*, pages 77–90, Boulder, Colorado, 1977.

[5] Thomas Connolly and Carolyn Begg. *Database Systems – A Practical Approach to Design, Implementation, and Management.* Addison Wesley, fourth edition, 2005.

[6] Michael de Raadt, Mark Toleman and Richard Watson. Electronic peer review: A large cohort teaching themselves? In *Proceedings of the 22nd Annual Conference of the Australasian Society for Computers in Learning in Tertiary Education (ASCILITE'05)*, pages 159–168, Brisbane, December 2005.

[7] Michael de Raadt, Mark Toleman and Richard Watson. An effective system for electronic peer review. *International Journal of Business and Management Education*, Volume 13, Number 9, pages 48–62, 2006.

[8] Stijn Dekeyser, Michael de Raadt and Tien Yu Lee. *SQLify* project website. Technical report, 2006. `http://www.sci.usq.edu.au/projects/sqlify/`.

[9] Suzanne Dietric, Eric Eckert and Kevin Piscator. WinRDBI – a Windows-based relational database educational tool. In *Proceedings of SIGCSE '97*, pages 126–130, San Jose, California, March 1997.

[10] R. Kearns, S. Shead and A. Fekete. A teaching system for SQL. In *Proceedings of ACSE '97*, pages 224–231, Melbourne, July 1997.

[11] Claire Kenny and Claus Pahl. Automated tutoring for a database skills training environment. In *Proceedings of SIGCSE'05*, pages 59–62, St. Louis, Missouri, February 2005.

[12] Michael Kiefer, Arthur Bernstein and Philip Lewis. *Database Systems – An Application-Oriented Approach.* Addison Wesley, second edition, 2006.

[13] J. Kurhila, M. Miettinen, P. Nokelainen, P. Floreen and H. Tirri. Peer-to-peer learning with open-ended writable web. In *Proceedings of the 8th Annual Conference on Innovation and Technology in Computer Science Education*, pages 173–178, Thessaloniki, Greece, June 2003.

[14] Antonija Mitrovic. Learning SQL with a computerized tutor. In *Proceedings of SIGCSE'98*, pages 307–311, Atlanta, Georgia, February 1998.

[15] Miguel Penabad. *General Procedure to Test Conjunctive Query Containment.* Ph.D. thesis, Universidade da Coruña, 2002.

[16] Julia Prior and Raymond Lister. The backwash effect on SQL skills grading. In *Proceedings of ITiCSE'04*, pages 32–36, Leeds, UK, June 2004.

[17] Shazia Sadiq, Maria Orlowska, Wasim Sadiq and Joe Lin. SQLator—an online SQL learning workbench. In *Proceedings of ITiCSE'04*, pages 223–227, Leeds, UK, June 2004.

[18] D. Saunders. Peer tutoring in higher education. *Studies in Higher Education*, Volume 17, Number 2, pages 211–218, 2006.

[19] Ben Shneiderman. Improving the human factors aspect of database interactions. *ACM Transactions on Database Systems*, Volume 3, Number 4, pages 417–439, 1978.

[20] Abraham Silberschatz, Henry Korth and S. Sudarshan. *Database System Concepts.* McGraw-Hill, fifth edition, 2006.

Pruning SIFT for Scalable Near-Duplicate Image Matching

Jun Jie Foo **Ranjan Sinha**

School of Computer Science & IT
RMIT University, Melbourne, Australia, 3001
Email: {jufoo,rsinha}@cs.rmit.edu.au

Abstract

The detection of image versions from large image collections is a formidable task as two images are rarely identical. Geometric variations such as cropping, rotation, and slight photometric alteration are unsuitable for content-based retrieval techniques, whereas digital watermarking techniques have limited application for practical retrieval. Recently, the application of Scale Invariant Feature Transform (SIFT) interest points to this domain have shown high effectiveness, but scalability remains a problem due to the large number of features generated for each image. In this work, we show that for this application domain, the SIFT interest points can be dramatically pruned to effect large reductions in both memory requirements and query run-time, with almost negligible loss in effectiveness. We demonstrate that, unlike using the original SIFT features, the pruned features scales better for collections containing hundreds of thousands of images.

Keywords: near-duplicate image matching, near-replicate image retrieval

1 Introduction

On the web, we often find copies, versions, or even fragments of images that are scaled-down thumbnails, or variants of the same digital image that are, legally or otherwise, kept by different sources. These are often images that are almost identical but not recognised as such due to common image manipulations such as conversion to greyscale, change in color balance and contrast, rescaling, rotating, cropping, and filtering. Such image variants are commonly known as near-duplicate images [Jaimes, Chang & Loui 2002, Ke, Sukthankar & Huston 2004, Zhang & Chang 2004].

In most cases, the storage and retrieval of duplicate and near-duplicate images may be unnecessary and, in the context of collections derived from the web, may further present infringements of copyright. However, the detection of these near-duplicate images is a formidable task as two image versions are rarely identical. They may differ in filename, format, and size; simply saving an image may lead to bitwise differences due to the variations in the coding standards in different software.

While the detection of copied digital images have been extensively researched in the field of digital watermarking [Hartung & Kutter 1999, Johnson, Duric & Jajodia 1999, Kang, Huang & Shi 2002, Kang, Huang, Shi & Lin 2003], such methods are ill-suited for retrieval applications due to practicality issues [Lu & Hsu 2005, Qamra, Meng & Chang 2005]. Similarly, content-based retrieval techniques [Smeulders, Worring, Santini, Gupta & Jain 2000] — a well researched area — are unsuitable as they are designed for a much broader class of image matches with similar traits of colour, texture, or shapes; these techniques have been shown to have limited effectiveness for this task [Chang, Wang & Wiederhold 1998, Sebe, Lew & Huijsmans 1999, Luo & Nascimento 2003].

For the task of retrieval of near-duplicate images, Ke et al. [2004] have demonstrated near-perfect accuracy using PCA-SIFT local descriptors [Ke & Sukthankar 2004] on a moderate-sized image collection of about 20, 000 images, however, scalability and efficiency aspects were not discussed. Qamra et al. [2005] propose perceptual distance functions for near-duplicate retrieval using color and texture image features on large proprietary image collections; however, limited effectiveness is observed, and efficiency aspects were only briefly discussed. Lu & Hsu [2005] demonstrated an image hashing technique for the retrieval of near-duplicate images on a collection of 20, 000 images with limited effectiveness, wherein the number of alterations, though large, are limited in severity.

Thus far, in this domain, only Ke et al. [2004] have demonstrated a method that is indeed highly accurate for even relatively severe alterations. However, an inherent problem with applying SIFT [Lowe 2004] features (represented with PCA-SIFT local descriptors) is the large number of features — ranging from hundreds to thousands — it generates per image, whereby each feature is represented by a high dimensional feature vector [Ke & Sukthankar 2004]. Though these large set of image features are indexed using Locality-Sensitive Hashing [Gionis, Indyk & Motwani 1999, Ke et al. 2004], we show that such large numbers of features are impractical for moderate to large image collections.

The SIFT detector was originally designed for robust matching of even small occluded objects; hence, the quantity of features is crucial for such applications [Lowe 2004, Ke & Sukthankar 2004]. We hypothesize that only a small subset of features are required for near-duplicate image matching, as most images of interest in this domain contain high perceptual similarity to their originals [Qamra et al. 2005]. In this work, we propose a pruning strategy that reduces the number of SIFT features (consequently PCA-SIFT local descriptors), thus enabling these descriptors to scale well for a large collection containing hundreds of thousands of images. Such a pruning strategy simultaneously reduces the index size and query response time to 1/10 and 1/50, respectively, of the original approach, with little impact on effectiveness.

In the following section, we describe the distinctive local descriptors (mainly SIFT and PCA-SIFT). We discuss the Locality Sensitive Hashing index structure used in this work in Section 3. In Section 4, we introduce our approach of pruning the SIFT interest points and in Section 5, we describe the evaluation methodology and experimental setup. The results are presented and discussed in Section 6, followed by our conclusions in Section 7.

2 Distinctive Local Descriptors

Local descriptors computed for images have been demonstrated to be useful for object recognition [Lowe 2004], and robust image matching [Ke et al. 2004, Grauman & Darrell 2005]. Given an image, the idea is to detect image regions (centered around interest points) that possess properties invariant to geometric variation and photometric changes, so that distinctive local descriptors can be computed for each region. Popular region detectors include the Harris-Point, Harris-Laplace, Harris-Affine, Hessian-Laplace, and Hessian-Affine; popular descriptors include the SIFT (Scale Invariant Feature Transform) [Lowe 2004], and PCA-SIFT [Ke & Sukthankar 2004], to name a few. For a complete survey on region detectors and local descriptors, the reader is directed to the work of Mikolajczyk & Schmid [2003]

In this work, we use the SIFT (scale invariant feature transform) detector — which uses regions similar to Hessian-Laplace — as it has been demonstrated to outperform most existing detectors [Mikolajczyk & Schmid 2003]. We apply the PCA-SIFT descriptors on the SIFT interest points, instead of the original SIFT descriptors, as it has been reported to be both highly distinctive [Ke & Sukthankar 2004] and highly effective for near-duplicate image detection [Ke et al. 2004].

SIFT and PCA-SIFT descriptors

The Scale Invariant Feature Transform (SIFT) [Lowe 2004] devised for robust image feature detection is invariant to scale, rotation, and affine transforms. There are four major computational stages in SIFT for extracting a set of image features, namely the scale-space extrema detection, *keypoint* localization, orientation assignment, and generation of local descriptor.

In the first phase of the SIFT detector, the difference-of-Gaussian (DoG) function uses a Gaussian pyramid to identify any local peaks (keypoints) in various locations and scales. (This is achieved by finding the local scale-space extrema of the DoG). In the second phase, poorly localized and unstable keypoints below several threshold levels are rejected; mainly based on contrast level and ratio of principal curvature [Lowe 2004]. After all stable keypoints are identified, each keypoint is assigned a dominant orientation for rotation invariance in the third phase. Additional keypoints are generated if there are multiple orientations within 80% threshold of the dominant orientation; thus, there can be multiple keypoints with identical scale, location, but different orientation.

Instead of computing SIFT local descriptors, we use the PCA-SIFT descriptors which have been shown to be the most distinctive [Mikolajczyk & Schmid 2003] compared to other descriptors. To generate local descriptors, PCA-SIFT uses the same information as the original SIFT descriptor, that is, location, scale, and dominant orientations. PCA-SIFT concatenates the horizontal and vertical gradient maps for the 41×41 region — centered around the keypoint, rotated to align its orientation to a canonical direction — to produce a $2 \times 39 \times 39 = 3042$ element local descriptor (feature vector).

In cases where there are multiple dominant orientations, a separate vector is calculated for each. Each vector is then projected — using principal component analysis, a common technique for dimensionality reduction — to a low-dimensional feature space using a pre-computed eigenspace[1]. Ke et al. [2004] have empirically determined that $n = 36$ feature spaces for the local descriptor performs well for near-duplicate image retrieval; wherein any two PCA-SIFT local descriptors are deemed similar (a match) within an Euclidean distance (L_2-norm) of 3000. Hence, in this work, we use the same settings.

The number of PCA-SIFT local descriptors are dictated by the keypoints detected by SIFT, typically ranging from hundreds to thousands per image (depending on image complexity). The indexing of such a substantial number of PCA-SIFT local descriptors for large image collection is impractical and costly. We later show that the number of keypoints that SIFT generates can be significantly reduced by varying the threshold value in the second stage, resulting in great gains in efficiency with only slight loss in accuracy for matching near-duplicate images.

3 Indexing Local Descriptors

To index a set of 36-dimensional PCA-SIFT descriptors, we use the Locality Sensitive Hashing (LSH) index scheme [Indyk & Motwani 1998, Gionis et al. 1999, Datar, Immorlica, Indyk & Mirrokni 2004, Ke et al. 2004, Bawa, Condie & Ganesan 2005] for approximate nearest-neighbour matching in high-dimensional spaces.

Given a set of points (PCA-SIFT descriptors) P, the distance between two points in the approximate nearest neighbor search can be defined as:

$$d(q,p) \leq (1+r)d(q,P)$$

where q is the query point, and $d(q,P)$ is the distance of q to the closest point in P. The key idea of LSH is to use a family of hash functions to ensure that the probability of collision of two points is closely related to the distance between them.

A hash function can be defined as $g_i(p) = (h_1(p), \ldots, (h_k(p))$, for $i = 1, \ldots, l$, where k determines the probability of collision, and l determines the fraction of false negatives [Indyk & Motwani 1998]. More specifically, this family of hash functions is called $(r1, r2, p1, p2)$-sensitive [Gionis et al. 1999] for the distance between the elements in P for any $q, p \in P$:

- if $p \in \beta(q, r_1)$ then $Pr_H[h(q) = h(p)] \leq p_1$
- if $p \in \beta(q, r_2)$ then $Pr_H[h(q) = h(p)] \geq p_2$

where $\beta(q,r)$ denotes the set of elements within the distance of r to q. Additionally, the requirements of $p_1 > p_2$ and $r_1 < r_2$ has to be satisfied for locality-sensitivity.

The family of functions can be efficiently computed with a Hamming space H^d for d dimensions [Gionis et al. 1999], whereby each d-dimensional vector $p(x_1, \ldots, x_d)$ can be mapped to a Hamming cube $H^{d'}$ with $d' = Cd$ (where C denotes the largest coordinate in P), transforming vector p to a binary Hamming string p'.

[1]The eigenspace used in this work is provided by Ke & Sukthankar [2004].

Given a transformed vector of p', a hash $g_i(p)$, for $i = 1, \ldots, l$, can be obtained by a projection of vector p' onto the coordinate set I_i (where I consists of k elements that are sampled randomly with replacement from $\{1, \ldots, d'\}$) essentially hashing point p to bucket $g_i(p)$. As the number of buckets can be high depending on the cardinality of set P, a second level of standard hashing is used to map the contents of $g_i(p)$ to a hash table [Gionis et al. 1999]. Hence, there are a total of l LSH tables, each using the LSH functions from the $H^{d'}$ family. The size of each hash table M is determined using:

$$M = \frac{n}{\alpha B}$$

where n is the total number of points in a collection, and B, l, and k (critical for efficacy), are empirically determined to be of size 20, 20, and 450, respectively [Ke et al. 2004]; the utilization parameter α is selected to be 0.5.

All points sharing identical hash values (collisions) within a given hash table are estimated, by the Manhattan distance (L_1-norm embedded in the Hamming space), to be closer to each other than those that do not. Thus, the search space of an approximate nearest-neighbor match is greatly reduced to those that share identical hash values. To further eliminate the number of false positive matches, an additional verification step is required as the LSH index structure returns only approximate matches based on L_1 whereas the PCA-SIFT features require two vectors to be within an L_2 (Euclidean) norm of 3000 to be deemed a match [Ke & Sukthankar 2004]; hence, all keypoint matches are post-processed to discard false positive matches. A final filtering phase is applied using robust estimators such as RANdom SAmple Consensus (RANSAC) [Fischler & Bolles 1981] to geometrically verify that two images are indeed near-duplicates to ensure high precision in the returned answers. We apply the same filtering in our work.

For query evaluation, all candidate matches are returned by the LSH index for every query keypoint. Hence, each query image is treated as a *bag of points*, simulating a multi-point query evaluation. To do this efficiently, we use an identical framework as Ke et al. [2004], in that, we maintain two auxiliary index structures — File Table (FT) and Keypoint Table (KT) — to map SIFT keypoints (and PCA-SIFT descriptors) to their corresponding images; an entry in KT consists of the file ID (index location of FT) and keypoint information (x and y location, scale, orientation, and the PCA-SIFT local descriptor). The cost of query evaluation of a single image depends highly on the cardinality of the set of keypoints P in any given image. We observe an average of 1900 keypoints for a large crawled image collection, which implies 1900 point queries to the LSH index for the evaluation of a single image query; this is computationally intensive and highly impractical for online interactive querying on large image collections.

In the next section, we introduce our approach for pruning the SIFT interest points.

4 Reducing SIFT Interest Points

With the SIFT detector, the number of computed keypoints are typically in the order of 10^3 (this depends on the image content, size, and complexity). Such quantities of keypoints is often crucial for object-recognition to enable even small occluded objects to be reliably matched [Lowe 2004]. However, for retrieval tasks on large collections, such a large number of keypoints can reduce any efficient index structure to a sequential search.

We hypothesise that the application domain of near-duplicate image detection may not require the full set of keypoints, and often a small subset will suffice. The rationale being that the application of near-duplicate image matching is often targeted at images displaying high perceptual similarity [Qamra et al. 2005]; even cropped images (to a certain extent) will possess many similar traits (objects, subjects, texture, and shape) to the original image.

To reduce the number of SIFT interest keypoints, we vary the threshold applied to discard candidate local peaks, specifically, the low contrast (intensity) threshold. Earlier, a threshold value of 0.03 was empirically observed to work [Lowe 2004] in the domain of object recognition, but it typically yields a large number of keypoints. However, using geometric verification techniques such as RANSAC [Fischler & Bolles 1981], only a small number of keypoints — 3 to 5 — are required for reliable image matching [Lowe 2004, Ke et al. 2004].

There are two ways that this inclusion threshold value can be used to reduce the cardinality of the keypoint set:

1. Select top N most significant keypoints ranked by the contrast value.

2. Raise the inclusion threshold value to discard keypoints.

Both methods can be used to reduce the number of keypoints. The second method, however, yields an unstable number of keypoints as images with lower average intensity level may not have any detected keypoints. Thus, varying the inclusion threshold could yield undesirable results since no assumption can be made regarding intensity levels of any given image. Hence, we apply the former approach, which sets an "upper bound" on the number of keypoints selected in this phase. Images that do not have N detected keypoints are not pruned; we experiment with varying the number of N significant keypoints to determine the optimal value. Note that the term upper bound is used loosely, since some keypoints may share the same location and scale information with multiple orientation, resulting in approximately 15% additional keypoints generated in the subsequent phase [Lowe 2004].

This pruning strategy enables images to be indexed using only a small subset of keypoints (and PCA-SIFT local descriptors) and as observed with only slight loss in effectiveness during retrieval.

5 Evaluation Methodology

We now describe the series of experiments used to empirically demonstrate the effectiveness of our approach. First, we evaluate the effectiveness of keypoint reduction by varying the number of detected keypoints between 1900 (original number of keypoints as observed in our collection) and using a subset of 1000, 100 and 10 most significant keypoints. These values are henceforth referred to as *threshold value*.

Second, we report the percentage of keypoint matches between a query image and each of its image alterations; this percentage is relative to the keypoints in the query image. Two keypoints are deemed to be a match if the nearest-neighbors are within the L_2 norm of 3000. For an accurate evaluation, we use the sequential scan for the nearest-neighbor search on the collection of keypoints as the LSH index is an approximate nearest-neighbor algorithm. Due to the exhaustive computation involved with using several keypoint thresholds, we use 100 random query sets and evaluate the keypoint matches for each of its alterations (see below). All subsequent experiments are performed using the LSH index with 200 queries.

Third, we evaluate the effects of keypoint reduction at all threshold levels, 1000, 100, and 10 and compare them against the original approach in terms of retrieval effectiveness (wherein the original image is used to retrieve all altered versions). We apply the standard recall and precision metrics, which are defined as:

$$recall = \frac{\text{relevant images retrieved}}{\text{total relevant images in collection}}$$

$$precision = \frac{\text{relevant images retrieved}}{\text{total images retrieved}}$$

We also measure query run-time[2], and index size.

Fourth, we evaluate the effectiveness of our approach — against the original approach — using each altered image as a query to retrieve only its respective original image from which it is derived; for this purpose specifically, we do not consider other altered images that are relevant to a given query. In addition, using a more stringent evaluation, we assess the retrieval effectiveness of our approach using each altered image as a query to retrieve all other altered images that are derived from the same original image; this is also compared against the original approach.

Finally, we vary the threshold levels on both the query images and the indexed test images (images relevant to the query that are within the collection). For instance, we use a query image with a threshold value of 1000 to retrieve images that are indexed using a threshold value of 100 to study their effects on retrieval accuracy and query run-time. An identical framework to Ke et al. [2004] is used; the only difference is the amount of PCA-SIFT features used.

All experiments are run on a two-processor Xeon 3 GHz machine with 4 GB of main memory running Linux 2.4.

Image Collections

To create our test collection, we select 200 images at random from the crawled SPIRIT collection [Joho & Sanderson 2004] and perform 50 common alterations. We use the original image as a query whereby all altered images are considered relevant answers. Hence, each of the selected 200 images produce 50 altered versions, yielding 10,000 images. We then select 10,000 and 90,000 additional images from the SPIRIT collection to serve as noise, thereby creating two image collections — C_1 and C_2 — with aggregated sizes of 20,000 and 100,000 images, respectively. The list of alterations — similar to those of Ke et al. [2004] and Qamra et al. [2005] — is as follows:

1. `format change`: format change from .jpg to .gif (1)
2. `colorise`: each of the red, green, and blue channels are tinted by 10% (3)
3. `contrast`: increase and decrease contrast (2)
4. `severe contrast`: increase and decrease contrast 3× of original image (2)
5. `crop`: crop 95%, 90%, 80%, and 70% of image, preserve center region (4)
6. `severe crop`: crop 60%, 50%, and 10% of image, preserve center region (3)
7. `despeckle`: apply "despeckle" operation of ImageMagick (1)
8. `frame`: a frame size 10% of image is added using random colors (4)
9. `rotate`: rotate image (by 90^o, 180^o, and 270^o) about its center (3)
10. `scale-up`: increase scale by 2×, 4×, and 8× (3)
11. `scale-down`: decrease scale by 2×, 4×, and 8× (3)
12. `saturation`: alter saturation by 70%, 80%, 90%, 110%, and 120% (5)
13. `intensity`: alter intensity level by 80%, 90%, 110%, 120% (4)
14. `severe intensity`: alter intensity level by 50% and 150% (2)
15. `rotate+crop`: rotate image (by 90^o, 180^o, and 270^o), crop 50% in center region (3)
16. `rotate+scale`: rotate image (by 90^o, 180^o, and 270^o), decrease scale 4x (3)
17. `shear`: apply affine warp on both x and y axes using 5^o, and 15^o (4)

The number in parentheses is the number of instances for each alteration type.[3]

For both collections C_1 and C_2, all images are converted into greyscale[4] and resized to 512 pixels in the longer edge prior to feature extraction and indexing.

6 Results

We now present our results on the retrieval effectiveness of our keypoint pruning strategy on several image alterations and discuss the efficiency of this approach in terms of query run-time, and index size.

Retrieval Effectiveness

As shown in Table 1, the effects of keypoint reduction is presented for all threshold values on all 50 alterations; the percentages are averaged over 100 queries. Columns 1, 6, and 11 refer to the different alterations (indicated by *Alt*); the adjacent columns denote the percentage of keypoint matches within the L_2 norm of 3000. Columns 2, 7, and 12 (indicated by Default) indicate the original number of keypoints per image, which is observed to be an average of 1900 for our collection.

We experiment with threshold values of 1000, 100, and 10 (reflected in subsequent columns), reducing the average keypoints per image to 1059, 130, and 14, respectively. Table 1 shows that keypoint reduction on all threshold values have little effect on the percentage of keypoints matched within the threshold criterion for most image alterations. This implies that our pruning strategy is appropriate for near-duplicate image matching.

The variation between the percentage of matching keypoints of image alterations are expected as some alterations severely affect the local descriptors, yielding lower overall keypoint matches. These trends are relatively stable across all levels of reduction, which leads us to believe that a small subset of keypoints is sufficient for this application. This is an important finding as the same criterion of L_2-norm within 3000 is the basis by which the LSH index approximates matching keypoints.

For some alterations, the slight increase in the percentage of matching keypoints, for some threshold values as compared to the default, is explained by the fact that matched keypoints are relative to the number of detected keypoints in the image.

[2]Note that query run-time does not include feature extraction time for simplicity.

Table 1: Percentage (%) of keypoint matches within L_2 norm threshold at every level of reduction. Columns 1, 6, and 11 indicate the different alterations (Alt). Keypoint thresholds of 1000, 100, and 10 are used. Default indicates the original number of keypoints per image (average of 1,900).

Alt	Default	1000	100	10	Alt	Default	1000	100	10	Alt	Default	1000	100	10
1	84.4	87.2	90.2	88.9	18	46.3	50.5	56.6	55.2	35	20.9	21.3	23.2	19.3
2	78.2	81.3	85.1	83.8	19	22.7	23.8	31.7	32.5	36	3.2	2.5	0.3	0.6
3	81.0	84.1	87.1	85.7	20	7.8	7.9	12.2	14.3	37	52.3	52.7	61.0	62.4
4	68.5	72.8	76.5	73.5	21	4.6	3.7	3.6	5.0	38	47.3	54.2	46.4	33.9
5	67.9	69.9	73.6	71.1	22	59.3	62.5	66.1	58.6	39	42.6	46.0	47.4	41.6
6	40.5	45.7	49.6	42.7	23	60.2	62.8	65.3	58.6	40	37.1	34.9	32.0	29.4
7	36.3	42.1	47.0	38.2	24	59.2	62.0	64.5	56.8	41	18.0	17.1	19.2	16.4
8	31.6	37.3	42.8	35.2	25	80.8	84.0	87.9	86.4	42	20.4	17.9	17.6	14.7
9	26.4	31.0	36.4	32.3	26	82.4	85.5	88.8	86.7	43	17.7	16.7	18.7	15.7
10	59.0	59.8	71.4	70.4	27	84.3	87.1	90.9	89.5	44	18.1	16.6	19.7	18.1
11	73.2	78.5	84.5	84.4	28	83.8	86.8	90.2	90.4	45	18.3	16.7	17.9	15.3
12	38.8	37.0	42.7	38.2	29	81.9	85.0	88.4	87.6	46	17.5	16.1	19.3	18.5
13	32.7	31.6	36.2	30.4	30	68.5	70.5	76.6	75.5	47	43.9	47.3	53.1	41.2
14	31.8	30.5	33.3	14.7	31	74.6	77.5	81.8	80.4	48	12.1	12.1	9.0	6.9
15	36.1	35.0	41.6	38.3	32	74.6	79.0	82.7	80.2	49	35.7	35.4	34.8	15.1
16	46.6	51.4	57.6	53.7	33	65.8	72.7	73.8	65.5	50	19.4	17.4	9.5	5.3
17	49.1	53.2	54.6	51.5	34	22.3	24.9	29.6	25.6	-	-	-	-	-

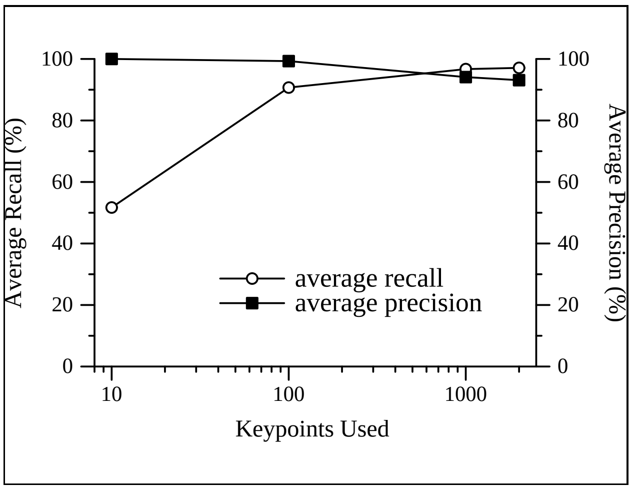


Figure 1: Average recall and precision (%) of retrieved answers using original images as queries on collection C_1 using 1900 (default), 1000, 100, and 10 keypoints; all values are averaged over 200 queries.

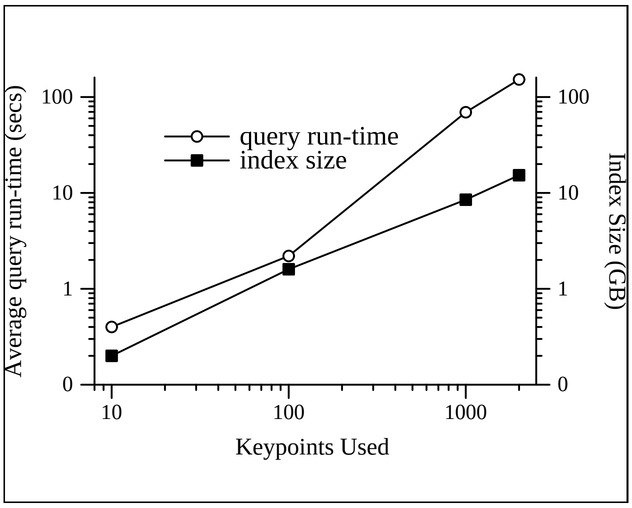


Figure 2: Average query run-time (secs) and Index sizes (GB) using 1900 (default), 1000, 100, and 10 keypoints on Collection C_1; all timings are averaged over 200 queries.

Figure 1 shows the average recall and precision of 200 queries on image collection C_1. The original image is used as a query to retrieve *all* its alterations; we apply the same variation of threshold values (1000, 100, and 10). Using a small threshold value of 100, we observe a high precision of 99.3% with only a slight drop in average recall of 90.7%. This implies that using roughly 100 keypoints, we retrieve 45 instead of 48 (default gives an average recall of 97.1%) on average. The average precision values for this threshold value also indicate that using less than 10% of the default number of keypoints, most of the relevant answers are retrieved with near-perfect accuracy. However, a threshold value of 10 results in a low average recall of 51.7%, indicating that too severe a reduction, results in considerable loss in effectiveness as almost 45% of the relevant answers are not retrieved.

Retrieval Efficiency

In Figure 2, we show the effect of varying threshold values on query evaluation time and memory requirements. With default settings, a single query evaluation on collection C_1 takes 152.4 seconds on average (over 200 queries). Using keypoint reduction with thresholds of 1000, 100 and 10, we observe timings of 69.4, 2.2, and 0.4 on average for a single query. This is a significant result, given that a threshold value of 100 also yields high effectiveness (average recall and precision of 90.7 and 99.3, respectively), with only a fraction of the speed of the original approach.

The on-disk memory requirements for the index (index size) of the same collection C_1 is also considerably reduced from 15.3 GB (default) to 8.5, 1.6, and 0.2 GB, for threshold values of 1000, 100, and 10, respectively. Note that the index size includes the Keypoint Table (KT) — wherein each entry is 92 bytes in length — and the LSH index; we do not include the File Table (FT) as this remains unchanged.

We do not stress on the timing patterns that are observed in our experiments as the existing implementation of the LSH index is not optimised in-memory; the timings were merely an indication of savings by the current framework. We believe an investigation

[3]All alterations are created using ImageMagick, `http://www.imagemagick.com`.

[4]The PCA-SIFT features are extracted from greyscale images.

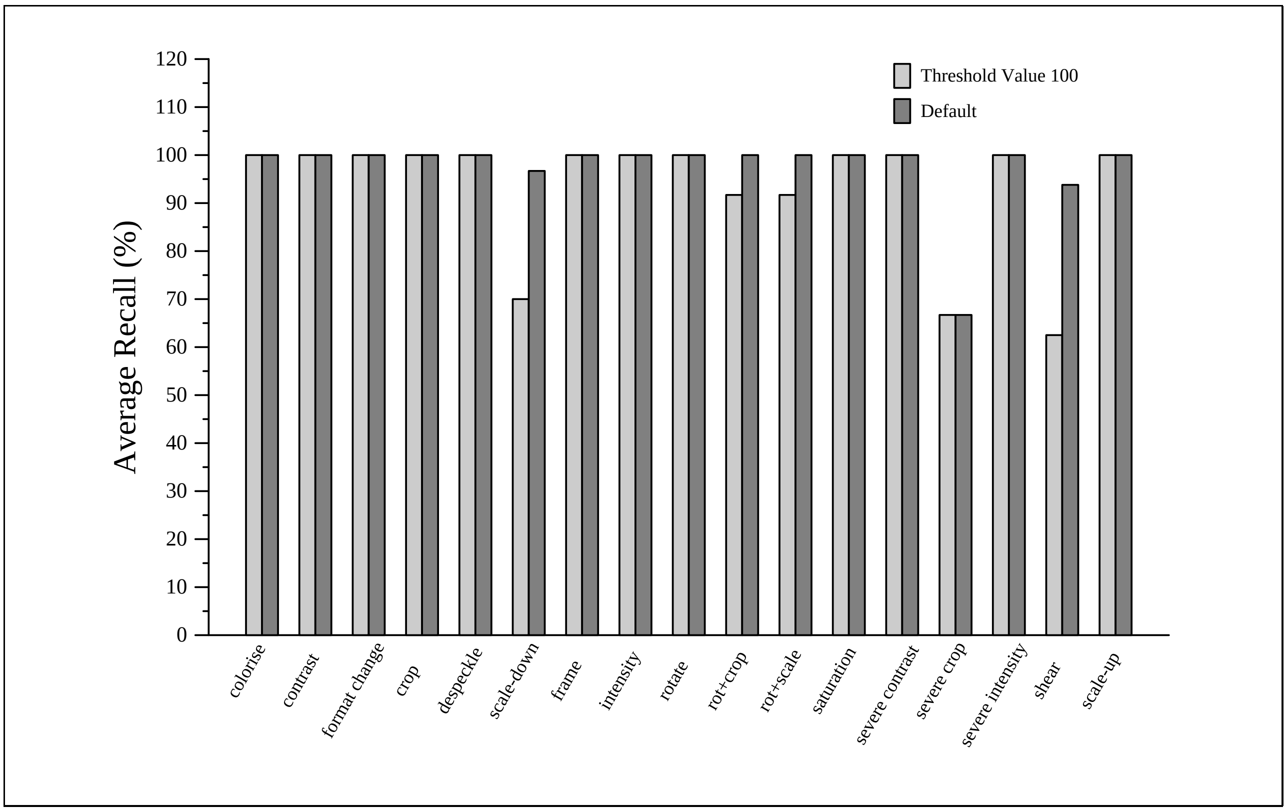


Figure 3: Average recall of each alteration over 20 queries — on Collection C_1 — to retrieve their respective original images using the original approach and threshold value of 100; each of the 17 alteration groups are described in Section 5, wherein there are 50 individual alterations in total.

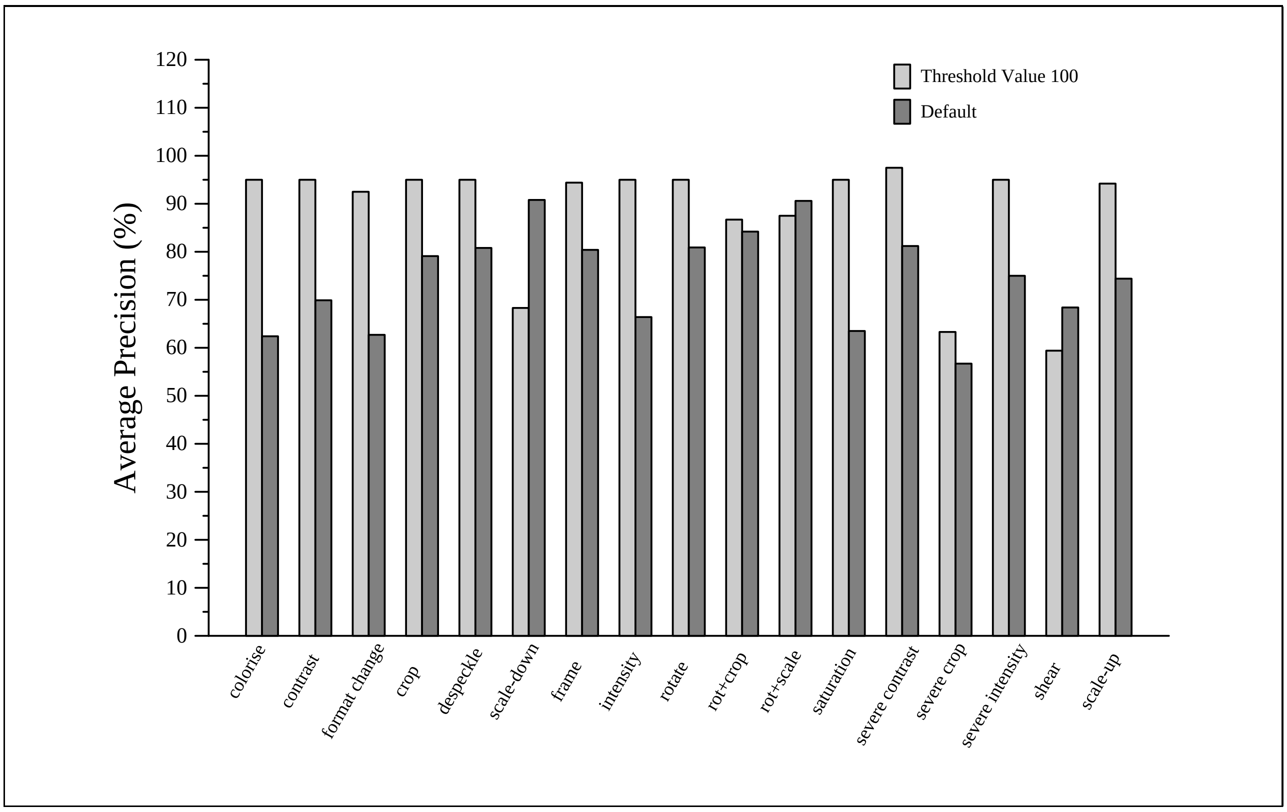


Figure 4: Average precision of each alteration over 20 queries — on Collection C_1 — to retrieve their respective original images using the original approach and threshold value of 100; each of the 17 alteration groups are described in Section 5, wherein there are 50 individual alterations in total.

Table 2: Average recall and precision (%) of 200 queries on Collection C_1 using four threshold values of 1000, 100, and 10; default indicates the original number of keypoints (1900 on average). Column 2 indicates the threshold value used for the query images.

Index KP	Query KP	Avg. Recall	Avg. Prec.	Avg. run-time	Candidate Set
Default	Default	97.1	93.1	152.4	62,549
	1000	96.7	93.9	102.1	31,890
	100	91.4	99.0	15.6	3,706
	10	56.1	100.0	1.0	395
1000	Default	96.9	93.4	91.5	41,772
	1000	96.7	94.1	69.4	27,634
	100	91.4	98.9	10.8	3,413
	10	56.2	100.0	0.7	372
100	Default	94.5	98.4	4.7	7,836
	1000	94.5	98.5	2.8	5,279
	100	90.7	99.3	2.2	2,647
	10	56.2	100.0	0.5	321
10	Default	68.4	100.0	0.9	920
	1000	68.4	100.0	0.7	634
	100	67.6	100.0	0.5	385
	10	51.7	100.0	0.4	265

on efficient in-memory data structures can further improve scalability on even larger image collections.

The threshold value of 100 provides the best tradeoff between effectiveness and efficiency and hence we apply this threshold value in further experiments.

Further Experiments

To investigate the effects of keypoint reduction on each image alteration, we use each altered version to retrieve its respective original image. We experiment with a threshold value of 100 and compare it to the original approach. Due to the large number of alterations, it is impractical to query using all 200 queries for each alteration; instead, we use 20 different queries for each alteration. For this experiment specifically, only the original images are indexed along with the noise collection of C_1; altered images are hence replaced by crawled images from the SPIRIT collection. To avoid confusion, we refer to this collection as C_1'

Figures 3 and 4 depict the retrieval effectiveness — average recall and precision, respectively — of each alteration on Collection C_1'. The 50 alterations are categorised into 17 groups as listed in Section 5. We observe a pleasingly high — relative to the original approach — average recall and precision across all alteration groups. For most alteration groups, the average recall is lossless, indicating that the original image is found with the altered query image using both threshold value of 100 and the original approach. An even more surprising result as shown in Figure 4 is that the same threshold value achieves a overall higher average precision than the original approach across almost all alterations, except for `scale-down`, `rotate+scale`, and `shear`. This indicates that our pruning strategy does not adversely affect accuracy but improves it; this can be explained the fact that our pruning strategy reduces the number of keypoints that are examined considerably, thereby also eliminating the number of potential false matches.

Figures 5 and 6 show the effectiveness — average recall and precision — of each alteration as queries in the retrieval of all other alterations on Collection C_1. We observe a lower average recall for a threshold value of 100 as compared to the original approach; however, note that this is a rather stringent evaluation measure given that a miss in one image alteration will be reflected in two groups. For instance, if a query image of alteration A does not retrieve an image of alteration B, a query the other way around would similarly fail. Nevertheless, for most alterations the discrepancies between them are relatively uniform, the only exceptions are `scale-down`, `severe contrast`, and `shear`, where we observe greater discrepancies. In contrast, Figure 6 shows a higher precision for all alterations using the threshold value of 100, indicating that the keypoint reduction slightly decreases completeness with no effect on the retrieval accuracy.

To further study the effects of keypoint reduction, we vary the threshold levels between the query and test images; thereby varying the quantity of indexed keypoints and query keypoints. This experiment allows us to determine whether using more keypoints for the query (on a smaller number of indexed keypoints) yields higher effectiveness. Table 2, shows the retrieval effectiveness of 200 queries on Collection C_1 using three different variations for thresholding the query and test images, mainly 1000, 100, and 10; default denotes the original number of keypoints (1900 on average). Columns 1 and 2 indicate threshold value used for the test (indexed), and query images, respectively. The last column indicates the average candidate keypoints that are retrieved from the LSH index during query evaluation.

As shown in this table, using the original number of keypoints to query for images using threshold value of 100 yields a high average recall and precision, of 94.5% and 98.4%, respectively, with a little under 5 seconds; this is similarly observed for the use of threshold value of 1000 for the query image, taking on average close to 3 seconds. Also note that the number of candidate keypoints that are examined are reduced considerably with threshold value of 100; from a set of $62,549$ to less than $10,000$. This is an important finding, given that with this threshold value, the loss of average recall is minimal while the savings in terms of index size and query run-time is significant. Note that similar trends are observed for other threshold values.

Finally, to evaluate the effectiveness of our approach on large collections, we experiment with a threshold value of 100 on Collection C_2 — containing $100,000$ images. As shown in Table 3, the average recall and precision are similar to those of Collection C_1. However, the average query run-time has increased considerably with default and threshold value of 1000 in the query images, as a result of a larger candidate keypoint set. In contrast, the threshold value of 100 for the query images is relatively unaffected judging by the slight growth in candidate keypoints. Furthermore, this threshold parameter requires only 6.2 seconds, indicating that it scales well for large image collections. We do not experiment on the original approach on Collection C_2 as the query run-time is impractical. Using this threshold value, we also observe an index size of 5.3 GB as compared to the estimated 76.5 GB using the original approach.

7 Conclusion

In this work, we have shown that the SIFT interest keypoints can be significantly pruned to reduce the amount of features that are indexed. Such pruning results in large reductions in both on-disk memory requirements and query evaluation time with only a slight loss in effectiveness.

To demonstrate the robustness of this approach,

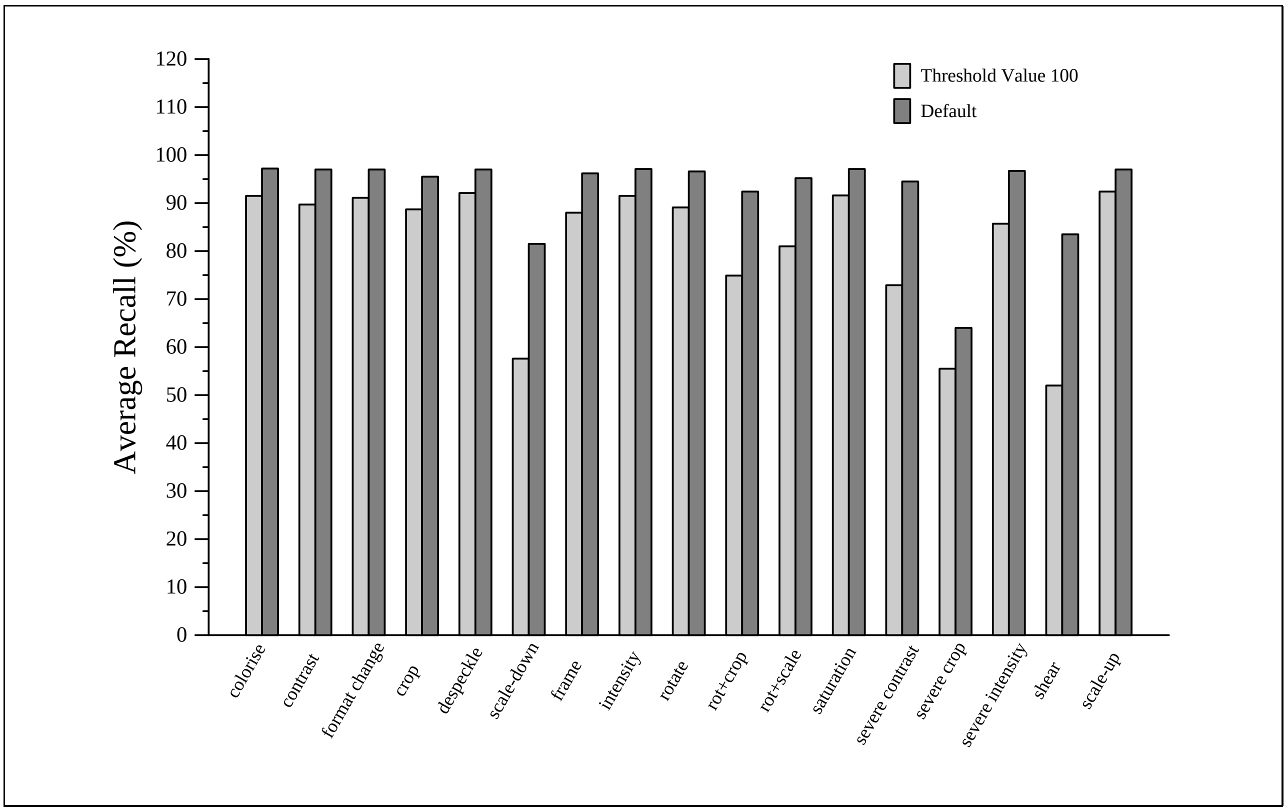


Figure 5: Average recall of each alteration over 20 queries — on Collection C_1 — to retrieve other altered images using the original approach and threshold value of 100; each of the 17 alteration groups are described in Section 5, wherein there are 50 individual alterations in total.

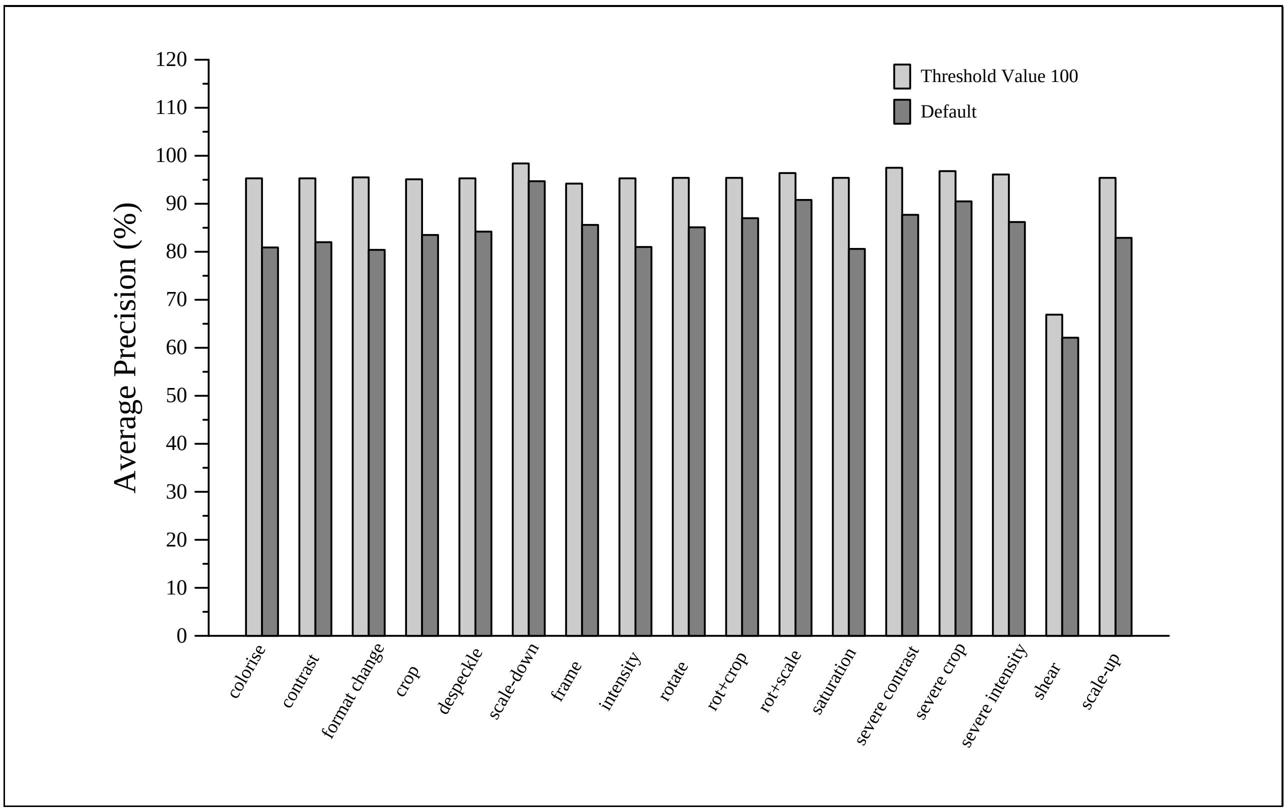


Figure 6: Average precision of each alteration over 20 queries — on Collection C_1 — to retrieve other altered images using the original approach and threshold value of 100; each of the 17 alteration groups are described in Section 5, wherein there are 50 individual alterations in total.

Table 3: Average recall and precision (%) of 200 queries on Collection C_2 using threshold value of 100. Column 2 indicates the threshold value used for the query images.

Index KP	Query KP	Avg. Recall	Avg. Prec.	Avg. run-time	Cand-idate Set
100	Default	94.5	98.2	40.8	19,215
	1000	94.5	98.4	30.7	10,728
	100	90.9	99.3	6.2	3,093
	10	56.9	100.0	0.5	367

we examine the effects of pruning on several kinds — including relatively severe ones — of image transformations. We show that our approach performs pleasingly well, with average recall close to the original approach; we also observe average precision to surpass the original approach.

Using parameters that show the best trade-off, our pruning method reduces the index size to approximately 1/10, and the query run-time to 1/50 of the original approach. Additionally, we demonstrate that, unlike the original approach, our method scales well for a large collection of $100,000$ images.

8 Acknowledgment

This work was supported by the Australian Research Council.

References

Bawa, M., Condie, T. & Ganesan, P. 2005, LSH forest: Self-tuning indexes for similarity search., *in* A. Ellis & T. Hagino, eds, 'WWW', ACM, pp. 651–660.

Chang, E., Wang, J. Z. & Wiederhold, G. 1998, RIME: A replicated image detector for the world-wide web, *in* 'Proc. SPIE Int. Conf. on Multimedia Storage and Archiving Systems III'.

Datar, M., Immorlica, N., Indyk, P. & Mirrokni, V. S. 2004, Locality-sensitive hashing scheme based on p-stable distributions., *in* J. Snoeyink & J.-D. Boissonnat, eds, 'Symposium on Computational Geometry', ACM, pp. 253–262.

Fischler, M. A. & Bolles, R. C. 1981, 'Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography.', *Commun. ACM* **24**(6), 381–395.

Gionis, A., Indyk, P. & Motwani, R. 1999, Similarity search in high dimensions via hashing, *in* 'Proc. VLDB Int. Conf. on Very Large Data Bases', Morgan Kaufmann, Edinburgh, Scotland, UK, pp. 518–529.

Grauman, K. & Darrell, T. 2005, Efficient image matching with distributions of local invariant features., *in* 'Proc. CVPR Int. Conf. on Computer Vision and Pattern Recognition', pp. 627–634.

Hartung, F. & Kutter, M. 1999, 'Multimedia watermarking techniques', *Proceedings IEEE (USA)* **87**(7), 1079–1107.

Indyk, P. & Motwani, R. 1998, Approximate nearest neighbors: Towards removing the curse of dimensionality., *in* 'Proc. STOC Int. Conf. on Theory of Computing', ACM Press, Dallas, Texas, USA, pp. 604–613.

Jaimes, A., Chang, S.-F. & Loui, A. C. 2002, Duplicate detection in consumer photography and news video., *in* 'Proc. MM Int. Conf. on Multimedia', pp. 423–424.

Johnson, N. F., Duric, Z. & Jajodia, S. 1999, On "Fingerprinting" images for recognition, *in* 'Proc. MIS Int. Workshop on Multimedia Information Systems', Indian Wells, California, pp. 4–11.

Joho, H. & Sanderson, M. 2004, 'The spirit collection: an overview of a large web collection.', *SIGIR Forum* **38**(2), 57–61.

Kang, X., Huang, J. & Shi, Y. Q. 2002, An image watermarking algorithm robust to geometric distortion., *in* 'Proc. IWDW Int. Workshop on Digital Watermarking', Springer, Seoul, Korea, pp. 212–223.

Kang, X., Huang, J., Shi, Y. Q. & Lin, Y. 2003, 'A DWT-DFT composite watermarking scheme robust to both affine transform and JPEG compression.', *IEEE Trans. Circuits and Systems for Video Technology* **13**(8), 776–786.

Ke, Y. & Sukthankar, R. 2004, PCA-sift: A more distinctive representation for local image descriptors., *in* 'Proc. CVPR Int. Conf. on Computer Vision and Pattern Recognition', IEEE Computer Society, Washington, DC, USA, pp. 506–513.

Ke, Y., Sukthankar, R. & Huston, L. 2004, An efficient parts-based near-duplicate and sub-image retrieval system, *in* 'Proc. MM Int. Conf. on Multimedia', ACM Press, New York, NY, USA, pp. 869–876.

Lowe, D. G. 2004, 'Distinctive image features from scale-invariant keypoints.', *Int. Journal of Computer Vision* **60**(2), 91–110.

Lu, C.-S. & Hsu, C.-Y. 2005, 'Geometric distortion-resilient image hashing scheme and its applications on copy detection and authentication.', *Multimedia Systems* **11**(2), 159–173.

Luo, J. & Nascimento, M. A. 2003, Content based sub-image retrieval via hierarchical tree matching., *in* 'Proc. MMDB Int. Workshop on Multimedia Databases', pp. 63–69.

Mikolajczyk, K. & Schmid, C. 2003, A performance evaluation of local descriptors., *in* 'Proc. CVPR Int. Conf. on Computer Vision and Pattern Recognition', pp. 257–263.

Qamra, A., Meng, Y. & Chang, E. Y. 2005, 'Enhanced perceptual distance functions and indexing for image replica recognition.', *IEEE Trans. Pattern Analysis and Machine Intelligence* **27**(3), 379–391.

Sebe, N., Lew, M. S. & Huijsmans, D. P. 1999, Multiscale sub-image search., *in* 'Proc. MM Int. Conf. on Multimedia', ACM Press, Orlando, FL, USA, pp. 79–82.

Smeulders, A. W. M., Worring, M., Santini, S., Gupta, A. & Jain, R. 2000, 'Content-based image retrieval at the end of the early years', *IEEE Trans on Pattern Analysis and Machine Intelligence* **22**(12), 1349–1380.

Zhang, D. & Chang, S.-F. 2004, Detecting image near-duplicate by stochastic attributed relational graph matching with learning., *in* 'Proc. MM Int. Conf. on Multimedia', pp. 877–884.

Optimizing XPath Queries on Streaming XML Data

Keerati Jittrawong **Raymond K. Wong**

University of New South Wales
and National ICT Australia
NSW 2052, Australia

wong@cse.unsw.edu.au

Abstract

XML stream processing has recently become popular for many applications such as selective dissemination of information. Several approaches have been proposed and most of them are based on the idea of finite automata. Different from these approaches, this paper presents a novel and efficient method for evaluating XPath with predicates on XML streaming data. For linear XPath expressions, our approach is at least as fast as the best method to date, i.e., with the cost of O(1) for each SAX event. For XPath with predicates, experiments have shown that our approach is efficient and scalable.

Keywords: DFA, Stream Data, XML, XPath Optimization.

1 INTRODUCTION

The popularity of XML as a standard for information representation and exchange has created a wave of new applications as well as challenges. In particular, due to the demands from sensor network applications [20], information dissemination [1], content based routing [28], and processing of scientific data [23], efficient evaluation of XPath expressions on XML stream data has attracted lots of attentions very recently. Most of these recent proposals are based on some form of finite automata and can be categorized into approaches either based on Nondeterministic Finite Automaton (NFA) or Deterministic Finite Automaton (DFA). In general, although DFA-based approach uses constant processing time independent of the XPath query workload, there are no space guarantees. Alternatively NFA-based approaches can guarantee space requirements, but they require more processing time. Example NFA-based approaches include XFilter, YFilter, and XTrie [1, 9, 4], and DFA-based approaches include Lazy DFA, and the XPush Machine [15, 17]. Those approaches represent XPath queries as finite automata, which can then be used to process against the incoming streaming XML data. Out of these approaches, the recent Lazy DFA [15] proposal addressed some of the above issues. First, it showed that the number of DFA states is small and will only be constructed lazily at run time. We observed from their result that the structure of input XML is usually small even though its schema may allow data instances with infinite structures. Based on their results and this observation, this paper proposes a novel approach by first extracting the structure of input XML documents and then using it for preprocessing XPath query workload. We term this special structure the *Structure Index*, due to the reason that the XPath queries are preprocessed according to the structure of the documents. While having better scalability, this approach is also based on the structure of the input XML documents, it thus will share some basic characteristics of the DFA-based approaches. Therefore, we also analyze and discuss the DFA-based approaches in detail in this paper.

Our contributions can be summarized as follows:

- We propose a novel approach based on the *Structure Index* which can be used to efficiently evaluate a large number of XPath queries on XML streaming data. We analyze the complexity of our approach both in time and space, and also verify it by experiments. After its short warm-up phase, it achieves constant processing time per SAX event independent of the query workload for linear XPath expressions.
- We propose a method called *Trigger Tree*, which is used to augment the Structure Index to efficiently evaluate XPath expressions with nested paths. This method is scalable with respect to both the number of nested paths and the number of value-based predicates when the queries have at least one value-based predicate. The central idea behind is to utilize the selectivity information of the value-based predicates.
- We provide detailed analysis for the DFA-based approaches, which has not been revealed in previous papers.

2 RELATED WORK

Related work on evaluating XPath expressions on XML data streams can be classified into DFA-based approaches and NFA-based approaches. We first discuss some recent NFA-based approaches below, followed by DFA-based approaches and other alternatives.

YFilter [9], a successor of XFilter [1], improves XFilter by using path sharing concept. It also separates the filtering problem of XPath expression into structure matching (using path expressions) and content matching (using value-based predicates). However, its performance still depends on the query workload.

XTrie [4] was designed to support large-scale filtering of streaming XML data. Unlike XFilter, XTrie supports complex XPath queries with predicates. Its basic idea is to use the *trie* to detect occurrences of substring matches for each event that it receives.

For DFA-based approach, the Lazy DFA [15], the XPush machine [17], and the recent work from Onizuka [26] (which is an improvement of the lazy DFA approach) are summarized below.

The XPush machine extends the lazy DFA approach to handle complex queries with nested paths using deterministic pushdown automaton in a bottom up fashion. Although this approach uses constant processing time in theory, it hardly achieves its theoretical performance due to its huge memory usage, and its efficiency in practice is about linear with the size of the query workload.

The recent work by Onizuka consists of two parts. The first part focuses on using lazy DFA with document-oriented XML, which usually has a complex schema. This class of data usually causes a problem in the lazy DFA approach because of its large data guide [14]. Onizuka clustered a large query workload into a number of smaller query workloads. Although this can reduce the soft upper bound of lazy DFA, it does not reduce the hard upper bound. Thus, the upper bound of lazy DFA still depends on a query workload and can be large. Therefore, this clustered lazy DFA approach can hardly achieve its stable phase, which can be observed from its experimental results.

Other approaches such as WebFilter [12] uses different approach based on the technique in [11], which treats an XML document as a set of attribute value pairs and an XPath expression as a collection of predicate pairs. [16] uses views of XML documents to speed up the processing time, which means that it needs to augment the query processor. Those approaches are interesting options and need further investigation to be compared with automata based approach.

3 THE STRUCTURE INDEX

In this section, we introduce the Structure Index, which is document structure derived from the input XML documents. It can be used to preprocess the structure navigation part of XPath queries. At runtime, the Structure Index can be used to efficiently find queries that match a given XML document by traversing its structure, and perform additional computation for predicate evaluation.

Example 3.1 Given two linear XPath queries:

```
Q1 = /a/*/c//d[text()="a2z"]
```

```
Q2 = //d[text()="t"]
```

This example will be used as an example in subsequent sections.

3.1 Document Structure

Document structure is a minimal structure of an XML document which is sufficient to answer structure navigation part of an XPath expression. It ignores element values, attribute values, duplicate elements, and document order since those information are not essential for structure navigation part. This is similar to an index structure for XML data such as DataGuides [14], the Index Fabric [7], and ViST [31]. However, those index structures have a goal to index the data to facilitate efficient query processing, while our document structure has a goal to extract the structure of data that can be used to preprocess queries. An example of an XML document and its document structure is shown in Figure 1.

```
<a m="1">
  <b>
    <c n="1">
      <d>
        <e>
          <f>3</f>
          <c/>
        </e>
      </d>
    </c>
  </b>
  <b>
```

(a)

```
<a>
  <@m/>
  <b>
    <c>
      <@n/>
      <d>
        <e>
          <f>
          </f>
          <c>
          </c>
        </e>
```

(b)

Figure 1. An XML document (a) and its document

3.2 The Structure Index and Preprocessing

Structure Index is constructed by representing each element or attribute node as a node in a Structure Index, called *Index Node*. The relationships between Index Nodes are the same as parent-child relationship in nodes of document structure. This index will be used (as an XML document tree) to preprocess the structural navigation part of XPath queries. It will attempt to find all possible ways that the structural parts of the queries can be matched with this XML document tree. Subsequently, each result from structure matching is stored at the Index Node that it matches. Figure 2 is a Structure Index resulting from document structure in Figure 1 (b), after queries from Example 3.1 are preprocessed. Each oval represents an Index Node where a text box shows queries that match at this Index Node. The algorithm to construct the Structure Index is described in Section 3.6.

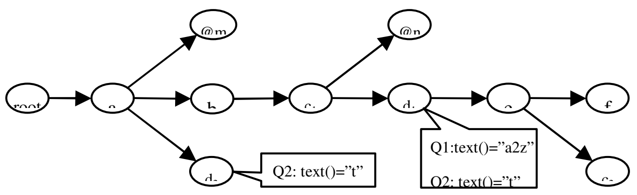


Figure 2. A Structure Index from document structure in Figure 1(b) after queries from Example 3.1 are

As a side benefit, the Structure Index also maintains containment and equivalence relationships between queries. These relationships could be used to easily find containment and equivalence for XPath queries, which has shown to be at worst coNP-complete in general in [19], between XPath queries with the given document and workload. As an example in Figure 2, it can be easily observed that query Q2 contains query Q1 under this structure domain since all nodes that the query Q1 match are also match by query Q2.

3.2.1 Preprocessing of Value-based Predicates

In a large query workload, there may be a large number of queries that have their structure navigation part matching at the same Index Node (as in d_1 node of Figure 2). Thus, this leaves a large number of value-based predicates to be evaluated at that node. A naïve solution is to evaluate those value-based predicates separately, which does not scale well. Here, we use *Atomic Predicate Index* from [17] to preprocess those value-based predicates. However, since we use the term value-based predicate, we slightly change its name to *Value-based Predicate Index.*

3.3 Processing an XML Stream with the Structure Index

In this section, we explain how a Structure Index can be used to process an XML stream using SAX parser. To process an XML stream with a Structure Index, we need a pointer to point to the current Index Node and a stack to maintain the previous Index Nodes. At this stage, we only interested in the start and end element events, and assume that character data is provided with an end element event. Attributes are simulated to be the same as elements except that its element name is added with a character `'@'`. Those tasks can be easily done as a mediation process between the SAX parser and the Structure Index. The processing of modified SAX events is described below.

Initially, the current Index Node is set to root. On a start element event, we push the current Index Node on the stack and lookup for a next Index Node with the name from the start element event. This takes only *O(1)* time, since a lookup is implemented as a hash table lookup. On an end element event, we evaluate the value-based predicate index of the current Index Node with a given character data. This takes only $O(\log_2 p)$ time, where *p* is a number of value-based predicates of that node, because the value-based predicate index is implemented as a binary search tree. All queries where their value-based predicates are evaluated to true, and queries without value-based predicates are considered as match queries. At the end, we pop the previous Index Node from the stack and set it as the current Index Node.

```
<a m=”1”>
  <b>
    <c n=”2”>
      <d>a<e>2</e>z</d>
    </c>
  </b>
  <d>char</d>
</a>
```

```
startElement(a)
  startElement(@m)
  endElement(”1”)
  startElement(b)
    startElement(c)
      startElement(@n)
      endElement(”2”)
      startElement(d)
        startElement(e)
        endElement(”2”)
      endElement(”a2z”)
      endElement(”a2z”)
```

(a) (b

Figure 3. An XML document (a) and its modified SAX

As a running example, we use the Structure Index in Figure 2 and modified SAX events in Figure 3 (b). At the beginning, a current Index Node start at root element Next, we see a `startElement(a)`. We push the current Index Node (root) to a stack, perform a lookup, and set the current Index Node to `a` Index Node. Next, we see a `startElement(@m)` and process it similarly to the previous event. Now, we see an `endElement(”1”)`. Since there is no queries match at this Index Node, we simply pop an Index Node from the stack (root) and set the current Index Node to that Index Node (root). The subsequent events are processed similarly until we reach an end element event of d_1 Index Node, an `endElement(”a2z”)`. At this Index Node, the current Index Node points to Index Node d_1, which we can see that both queries Q1 and Q2 match; thus, we need to evaluate the value-based predicate index of this node. After evaluating the value-based predicate index with data value `”a2z”`, the value-based predicate of query Q1 is evaluated to true, while the value-based predicate of query Q2 is evaluated to false. As a result, only query Q1 matches this document at this stage. Later on, we reach the end element of d_2 Index Node, which is an `endElement(”char”)`. Again, we evaluate the value-based predicate index of this node, but no value-based predicate is evaluated to true. At the end, only query Q1 matches with this XML document. This running example has shown that the Structure Index can be used to efficiently find match queries.

In summary, processing XML stream with a Structure Index takes *O(1)* for each start element event and takes $O(\log_2 p)$ for each end element event. This $O(\log_2 p)$ can be considered as a constant compare to the size of queries in a workload. Therefore, the processing time per SAX event of Structure Index for linear XPath queries is approximately a constant, independent of the query workload.

3.4 The Size of the Structure Index

As the Structure Index is document structure derived from the input XML documents, it can be observed that the size of the Structure Index depends on the schema of the input XML documents. A definition of document structure results in the data guide [14] of input XML data. Therefore, the upper bound of the size of the Structure Index is the size of DataGuides, and does not depend on the size of a query workload. An empirical observation in [15] reveals that the size of DataGuides is usually small in real data regardless of its schema for data-oriented XML, because real data tends not to exploit all possible patterns allowed by its schema. On the other hand, for datasets where its data guide is large, the size of the Structure Index can be large. This affects the performance of the Structure Index significantly both in time (due to its Index Node construction), and space (due to the space usage of Index Nodes). This becomes the same characteristic as with lazy DFA.

Additionally, to validate the size of the Structure Index, we also show, in Figure 4, the number of Index Nodes in Structure Indexes of three different datasets, which are Protein [27], NASA [23] and NITF [8] datasets which is used in [15] and [9]. Table 1 shows some characteristics of the schemas of these three datasets.

Table 1. Characteristics of the schemas of three XML datasets.

Dataset	Number of element names	Number of attributes	Recursive schema
Protein	64	13	No
NASA	142	197	Yes
NITF	123	510	Yes

For each dataset, we extract the first 200 XML documents from the real dataset. In comparison, we also try to generate synthetic data to be similar to the real data. Since the Protein dataset has a maximum depth of 7, we generate its synthetic using D=7 and RP=2. NASA has a maximum depth of 8, so we generate its synthetic data using D=8 and RP=2. However, we do not have real data for NITF, thus, we generate it using the same parameter as in synthetic NASA dataset. Table 2 show some characteristics of three XML datasets. The results are shown in Figure 4.

Table 2. Characteristics of three XML datasets.

Dataset	Number of elements	Number of attributes	No. of elements and attributes
Protein – Real	19336	1254	20590
Protein – Synthetic	19521	8891	28412
NASA – Real	39574	4152	43762
NASA – Synthetic	25860	25703	51563
NITF – Synthetic	7652	29365	37017

From Figure 4, it can be seen that there is a huge difference in the number of Index Nodes between real and synthetic data. In this result, the number of Index Nodes of synthetic data continues to increase since its upper bound is large, and it cannot reach its upper bound. In contrast, the upper bound of real dataset is very small. After a short period, the number of Index Nodes almost reaches the upper bound, and becomes almost constant. In addition, it can be seen that synthetic data of the Protein dataset has the same characteristic as real data because its schema is non-recursive. Thus, its synthetic data cannot be much different from its real data

Nevertheless, another major space usage in the Structure Index is the XPath expression table, which contains pointers to the XPath expressions that kept at each Index Node (for fast maintenance of the index in case of document updates). The size of this table grows linearly with the size of queries in a workload. However, this XPath expression table is used to improve the performance of Index Node construction, and can be removed to save more space.

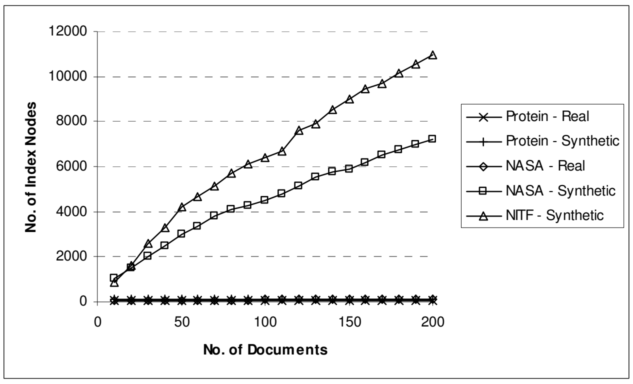


Figure 4. Number of Index Nodes of three different datasets for real data and synthetic data.

3.5 Warm-up and Stable Phase

The processing of Structure Index can be divided into two phases, which are *warm-up phase* and *stable phase*. The warm-up phase is a phase where most of Index Nodes of Structure Index are constructed. This Index Node construction accounts for most of the processing time in the warm-up phase. However, this processing overhead only occurs at the beginning of the process. After it reaches its stable phase, there is virtually no processing overhead because construction of Index Nodes does not occur or rarely occurs. At this stage, the processing time becomes approximately constant as analyzed previously. Nevertheless, under certain circumstance where data is synthetic and their schemas are recursive, it is possible that its stable phase will not be reached, and the processing overhead from constructing an index is unavoidable. However, this situation hardly occurs in practice for data-oriented XML. To verify this analysis, we show the number of Index Node construction, using the same datasets from previous section, in Figure 5. The reported number is an average number for groups of ten XML documents.

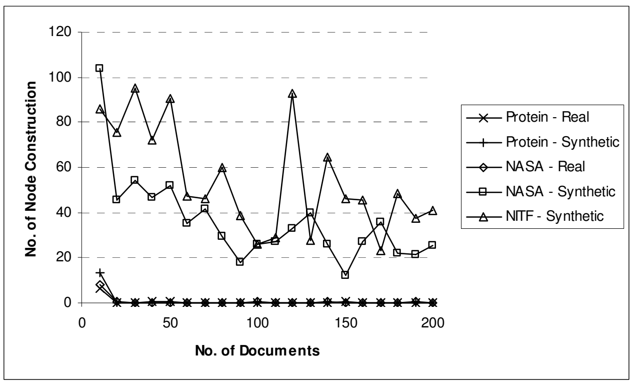


Figure 5. The number of Index Node construction from three different datasets for real and synthetic data.

From Figure 5, it can be seen that, for real data, major constructions of Index Nodes occur at the very beginning of the processing, and becomes zero, or a few after that. On the other hand, for synthetic data, constructions of Index Node are unavoidable because of its large upper bound. Again, synthetic data of protein dataset has the same characteristics as real data because of its non-recursive schema.

3.6 Maintaining the Structure Index

3.6.1 Structure Update

Structure update occurs when the Structure Index encounters unknown document structure and needs to construct Index Nodes for this new document structure. This process can be viewed as a learning process of the Structure Index. An algorithm to construct a new Index Node is given in Algorithm 3.1.

Algorithm 3.1 An algorithm to construct an Index Node

```
Method UPDATE-STRUCTURE (P, t)
Input:  P is a parent Index Node where an update occur
        t is a name of found element or attribute.
Data:   XPT is an XPath expression table of Index Node P
Output: N is a new Index Node which is a child of P
1   Create new Index Node N;
2   Find XPath expression in XPT that has a name test t;
3   for (each expression e with name test t)
4       INSERT-EXPR (N, e);
5   Find XPath expression in XPT that has a name test *;
6   for (each expression e with name test *)
7       INSERT-EXPR (N, e);
8   p = parent of P;
9   while (p is not the root node)
10      Find XPath expression in XPT of p that has a
        name test t with a descendant axis;
11      for (each expression e with name test t)
12          INSERT-EXPR (N, e);
13      Find XPath expression in XPT of p that has a
        name test * with a descendant axis;
14      for (each expression e with name test *)
15          INSERT-EXPR (N, e);
16      p = parent of p;
17  set N as a children of P;
18  return N;
```

```
Method INSERT-EXPR (N, e)
Input:  N is an Index Node
        e is a name of found element or attribute.
Data:   XPT is an XPath expression table of Index Node N
        MQ is a list of XPath queries that match at Index
        Node N
        VPI is a value-based predicate index for XPath
        queries that its structure navigation part match at
        Index Node N
1   c = child of the main path of e;
2   if (c is NULL)
3       Insert c in MQ;
4   else if (c is a valued-based predicate)
5       Insert c in VPI;
6   else
7       Insert c in XPT;
8   for (each predicate p of e)
9       if (p is a valued-based predicate)
10          Insert p in VPI;
11      else
12          Insert p in XPT;
```

Intuitively, this algorithm simply evaluates queries with the Structure Index. In this algorithm, structure update occurs at an Index Node *P* with a new element or attribute *t*. In each Index Node, important information are kept, which are *XPT*, an XPath expression table of XPath fragments of queries that match at this node, *MQ*, a list of queries that match at this node, and *VPI*, a value-based predicate index which is used to evaluate value-based predicates of queries that match at this node. Then, a new Index Node is constructed, and all XPath fragments that have node test *t* and * need to insert in this new Index Node. In addition, we also need to look for XPath fragments that have node test *t* and * at each ancestor, because the descendant axis can match at any level. Alternatively, we can insert XPath fragments with descendant axis at all of its descendant nodes, but this requires more space, and the former method makes space usage of the Structure Index less sensitive to the descendant axis.

3.6.2 Query Update

Query update occurs when there is an update in the query workload. There are two types of query update, insertion and deletion.

Query insertion can be viewed as evaluating one query with one XML document, since the Structure Index is simply a structural part of XML documents. Generally, query insertion is very efficient because the Structure Index is usually very small in data-oriented XML. Additional work is required to insert value-based predicates into a valued-based predicate index at each corresponding Index Node. This is also very efficient because valued-based predicate index is implemented as a binary search tree.

Query deletion can be done by traversing the whole Structure Index and removing the query from each Index Node. This is also efficient because the Structure Index is usually very small in data-oriented XML.

3.7 Nested Paths

In this section, we extend the Structure Index to handle more complex XPath queries, queries with nested paths. This kind of queries is more complex and quite different from the problem of linear XPath queries. One general approach that can be used to deal with this problem is to decompose an XPath query with nested paths as multiple linear XPath queries, and combine results from each linear XPath query to answer the query with nested paths. This approach is demonstrated in Example 3.2.

Example 3.2 Given the XPath query from Example 1.1:

```
Q0 = /a/*[c/@n>1][//d=3]//c
```

It can be decomposed into the following linear XPath queries:

```
Q3.0 (main) = /a/*//c
Q3.1 (nested) = /a/*/c/@n>1
Q3.2 (nested) = /a/*//d=3
```

However, this approach becomes complicated since each nested path must match under the same element of a document. To avoid this problem, we use a slightly different approach by viewing an XPath query with nested paths as a tree of linear queries.

In addition, from an empirical observation with data-oriented XML, a major difference between queries in the query workload is not in their structure navigation part, but in their predicate evaluation part. This is because possible patterns of linear XPath queries without predicates is quite limited. For example, consider different the XPath queries in the form:

```
//keyword[text()="database"]
//keyword[text()="XML"]
//keyword[text()="XPath"]
...
```

Those queries have the same structure navigation part, but are different in their predicate evaluation part. This kind of query is likely to happen and the domain values which can be used in the predicate evaluation part is very large. As a result, in a large query workload, there will be a large number of value-based predicates to be evaluated, and only a small amount of those value-based predicate will evaluate to true. Therefore, we take advantage of this observation by using the concept of trigger, where each query has its *Trigger Tree*, and the processing of a trigger occurs only when value-based predicates are evaluated to true. This Trigger Tree allows a processing of arbitrary nested paths and can be extended to support an XPath query with boolean connectors. This approach to handle XPath queries with nested paths are further described in the following subsections.

3.7.1 Query Tree

A query tree is a tree that represents an XPath query. A query tree of the query from Example 3.2 is shown in Figure 6. It should be noticed that both nested paths `c/@n>1` and `//d=3` are represented as `c/@n[text()>1]` and `//d[text()=3]` respectively since those forms can also represent main path with value-based predicate, and it evaluate to the same result.

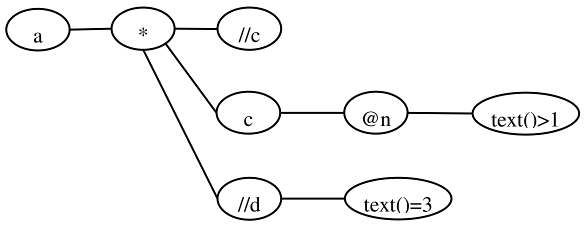


Figure 6. A query tree represents a query from Example 3.2.

3.7.2 Trigger Tree

A Trigger Tree is a tree of trigger node that represents a structure of an XPath query. A query tree of a query from Example 3.2 is shown in Figure 7.

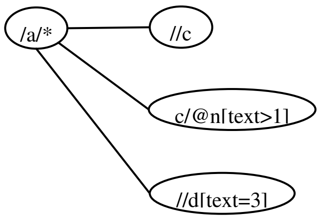


Figure 7. A trigger tree of a query from Example 3.2.

Later on, a Trigger Tree is preprocessed with the Structure Index. This is illustrated in Figure 8 where the Structure Index from document structure in Figure 1 (b) is preprocessed with a Trigger Tree in Figure 7.

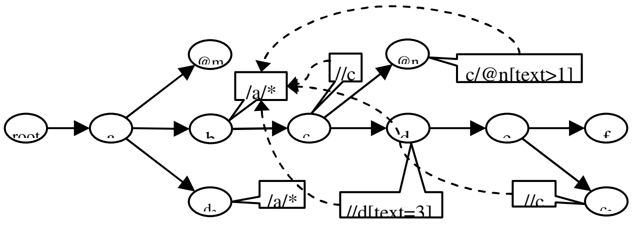


Figure 8. A Structure Index from document structure in Figure 1(b) after a Trigger Tree in Figure 7 is preprocessed.

3.7.3 Trigger Tree at Run time

At run time, input XML documents are processed similarly to the processing described in Section 3.3. Additional work is that when a nested predicate matches, it needs to trigger its parent; and if all children of the parent match, the parent becomes true, and the following condition is checked. If this trigger node (parent) also has a parent, again, it needs to trigger its parent. If it does not have a parent, it means that this node is a root node of a Trigger Tree and a query corresponding to this node match with this document. Nevertheless, extra work to clean up a Trigger Tree, and some duplicate triggers has to be handled carefully.

4 EXPERIMENTS

This section presents various experiments of the Structure Index. Our execution environment consists of a Pentium III 600 Mhz processor with 512MB memory running JVM 1.4.1 on Linux kernel 2.4.22. The SAX parser we used is Xerces2 Java Parser 2.5.0 [2] in non-validating mode.

The NASA XML dataset [23] is chosen as an experiment dataset because it is a data-oriented XML with recursive schema. We used the first 200 XML documents concatenated into a single file. We use a modified version of the XPath generator in [9] to generate synthetic XPath queries in distinct mode. The modification is to generate value-based predicates using data values from the NASA XML dataset.

All the numbers reported are averages of this dataset unless stated otherwise. The probability of wildcard (*) and descendant axis (//) is set to 20%. All reported processing times are in milliseconds, including document parsing time.

4.1 Experiment 1: Linear XPath Queries

The first experiment shows the performance of the Structure Index when the query workload consist of linear XPath queries. We varied the size of a query workload from 1,000 to 1,000,000. The processing time reported in the graph is the processing time for each XML document. The reported time is an average time for each ten XML documents.The results are shown in Figure 10.

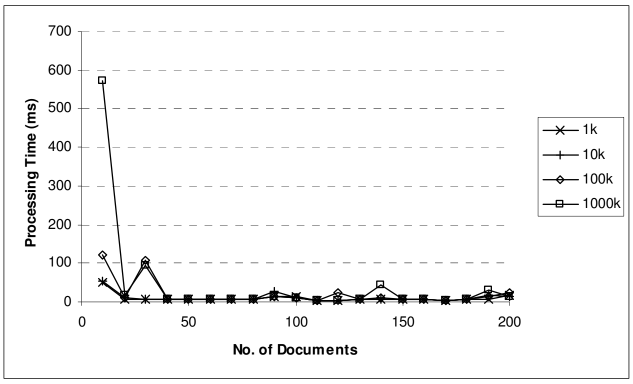


Figure 9. The processing time of real NASA data for 1k, 10k, 100k, 1000k queries.

This experimental result has shown that, after the short warm-up phase, our approach is extremely efficient. It uses constant processing time independent of the query workload. A little fluctuation comes from a few Index Node constructions that rarely occur after the short warm-up phase. Also, it should be noted that the processing time of the warm-up phase depends on the size of queries. This result comes from the construction cost of Index Nodes, which is more expensive when the query workload is large.

In comparison, we also use synthetic NASA data from Section 3.4. To ease the comparison, we plotted one result from real NASA dataset at the query size of 100,000. We ran the experiment with the same above setting. The results are shown in Figure 11.

From this result, there is a huge difference between the processing time of synthetic data and real data as can be compare in the case of 100,000 queries. Synthetic data takes much more processing time, and depends on the query workload, because it cannot achieve its stable phase.

In brief, for real data-oriented XML, the processing time of the Structure Index is about constant independent of the query workload, while, for synthetic data, it cannot achieve its stable phase and results in decreased performance.

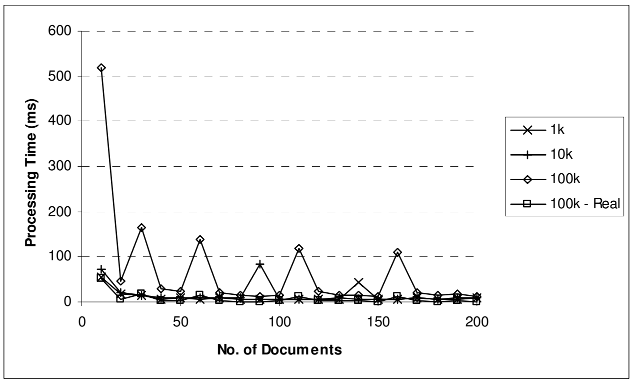


Figure 10. The processing time of synthetic NASA data for 1k, 10k, 100k queries.

4.2 Experiment 2: XPath Queries with Nested Paths

In this experiment, we evaluate the performance of the Structure Index with Trigger Trees when the query workload consists of XPath queries with nested paths. We varied the size of a query workload from 10,000 to 50,000. The probability of equal, range, and not equal operators are set to 80%, 10% and 10% respectively. All reported time is the processing time in stable phase. In the first experiment, we show that the processing time of Trigger Tree depends largely on selectivity of value-based predicates. All queries have 3 nested paths. We vary percentage of nested paths without value-based from 0, 25, 50, 75, and 100 percent. The result is shown in Figure 12.

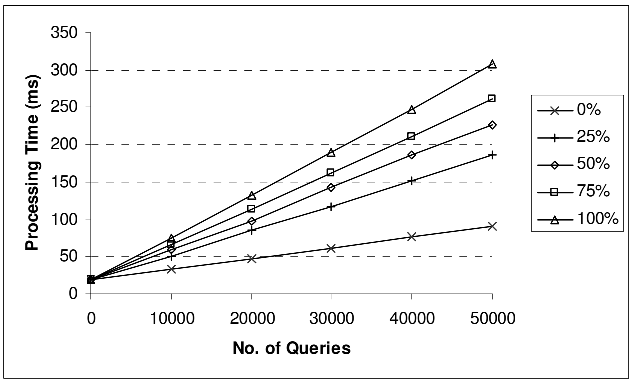


Figure 11. A comparison of the processing time for queries with nested paths with/without value-based predicates.

It can be seen from this figure that the scalability depends on the percentage of nested paths without value-based predicates, where pure nested paths with value-based predicates (0%) has the best scalability, and pure nested paths without value-based predicates (100%) has the worst scalability. Also, the scalability of pure nested paths with value-based predicates is quite distinct from the others. Obviously, this comes from the selectivity of value-based predicates because a nested path with value-based predicate has a much higher selectivity. To gain more understanding on this result, the number of triggers is shown in Figure 13.

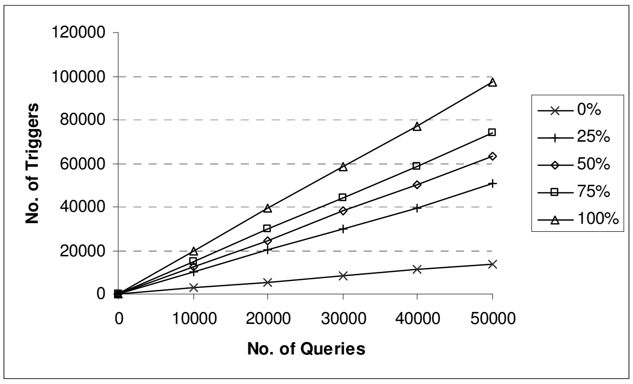


Figure 12. A comparison of the number of triggers for queries with nested paths with/without value-based predicate.

This figure is very similar to the previous figure. This result verifies that the processing time of Trigger Tree depends primarily on the number of triggers. In addition, when all nested paths have value-based predicates (0%), Trigger Tree is very efficient. This result leads to the concept of post-processing where less selective value-based predicates are processed after more selective value-based predicates has been satisfied.

4.3 Experiment 3: XPath Queries with Nested Paths using Post-processing Concept

This experiment demonstrates the performance of the Structure Index with Trigger Tree using post-processing concept, when the query workload consists of XPath queries with nested paths. The first result in Figure 14 shows the processing time of a Trigger Tree using post-processing concept.

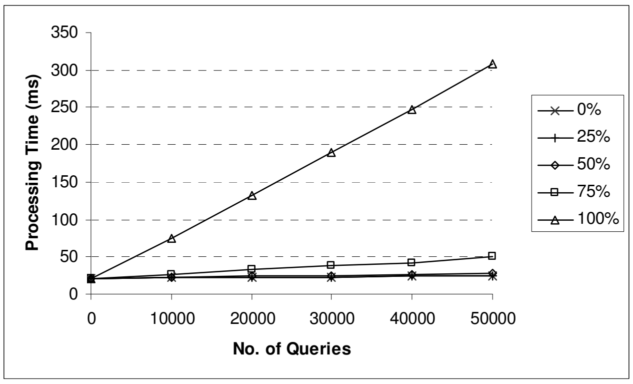


Figure 13. A comparison of the processing time for queries with nested paths with/without value-based predicates using post-processing concept.

From this figure, it can be seen that post-processing significantly improves the performance because it post process value-based predicates with low selectivity. Nevertheless, for the case of pure nested paths without value-based predicates (0%), the result is the same as in the previous experiment. In this case, post-processing cannot help because all its paths have low selectivity. To gain more understanding, we also show the number of triggers in Figure 15.

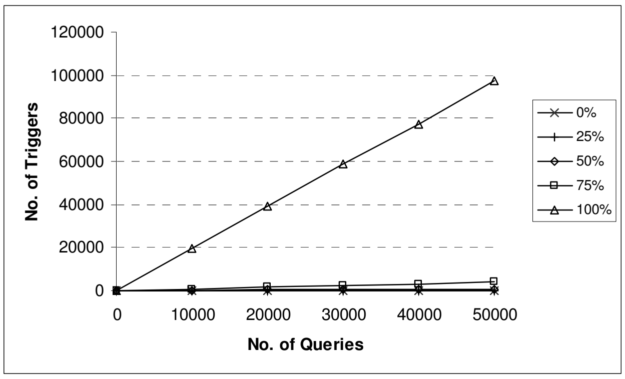


Figure 14. A comparison of the number of triggers for queries with nested paths with/without value-based predicates using post-processing concept.

As predicted, post-processing greatly reduces the number of triggers. In fact, the number of triggers is even less than the number of triggers of pure nested paths with value-based predicates from the previous experiment. This comes from that post-processing also postpone range and not equal operators since it has lower selectivity than an equal operator. Again, the number of trigger of pure nested paths without value-based predicates far exceeds the others.

Next, we show the performance of post-processing concept while varying the number of nested paths from 2, 5, and 8 nested paths. In this experiment, we vary the number of queries from 20000 to 100000. We set the percentage of nested paths without value-based predicate to 67 percent in all queries since nested path tends to end with value-based predicates. Other parameters are the same as in the experiment 2. The result is shown in Figure 16.

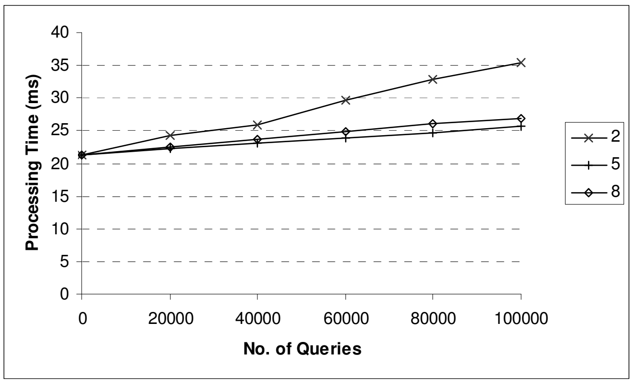


Figure 15. The processing time for queries with nested paths varying number of nested paths.

From this result, the performance of Trigger Tree with post-processing is highly scalable to a large number of nested paths. In fact, queries with 2 nested paths use more processing time than the others. Again, this comes from the selectivity of the value-based predicates since more nested paths means more chance for a query to has more selective value-based predicates. However, queries with 8 nested paths also use more processing time than queries with 5 nested paths. This comes from the number of decomposed linear queries, where 5 nested paths decompose to 600,000 linear queries, and 8 nested paths decompose to 900,000 linear queries. Thus, there is more chance for value-based predicates of queries with 8 nested paths to be evaluated to true, while queries with 5 nested paths is the most balance between its selectivity, and the number of decomposed linear queries.

5 CONCLUSIONS

In this paper, we proposed a novel approach called *Structure Index* for evaluating a large number of XPath queries on XML streaming data. Our focus is on data-oriented XML which has many practical applications. Our approach is based on the document structure, which can be easily derived from the input XML documents. After that, XPath queries are preprocessed and annotated into the Structure Index to facilitate efficient matching against future input XML documents. We analyzed the efficiency of the Structure Index both in time and space, and also by experiments. After a short warm-up phase, it achieves constant processing time, *O(1)*, per SAX event independent of the query workload for linear XPath queries. In addition, we proposed a method called *Trigger Tree* to efficiently evaluate XPath queries with nested paths. This method is scalable in both the number of nested paths and the number of value-based predicates when the queries have at least one value-based predicate. For space usage, the space efficiency of the Structure Index depends on the size of the structure of the input XML, which is usually very small for data-oriented XML. The maintenance cost (query update) of our approach is also very efficient because it also depends on the size of (small) document structure.

6 REFERENCES

[1] M. Altinel, and M. Franklin. Efficient Filtering of XML Documents for Selective Dissemination of Information. In *Proceedings of VLDB*, pages 53-64, Cairo, Egypt, September 2000.

[2] The Apache Software Foundation. Xerces2 Java Parser. http://xml.apache.org/xerces2-j.

[3] S. Babu, and J. Widom. Continuous Queries Over Data Streams. *SIGMOD* Record, 30(3):109-120, September 2001.

[4] C. Chan, P. Felber, M. Garofalakis, and R. Rastogi. Efficient Filtering of XML Documents with XPath Expressions. In *Proceedings of the International Conference on Data Engineering*, 2002.

[5] J. Chen, D. DeWitt, F. Tian, Y. Wang. NiagaraCQ: A Scalable Continuous Query System for Internet Databases. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, pages 379-390, Dallas, TX, May 2000.

[6] Y. Chen, S. B. Davidson, Y. Zheng. Validating Constraints in XML. Technical report, *University of Pennsylvania*, 2002. Technical Report MS-CIS-02-03.

[7] B. F. Cooper, N. Sample, M. Franklin, G. R. Hjaltason, and M.Shadmon. A fast index for semistructured data. In *VLDB*, pages 341-350, September 2001.

[8] R. Cover, The SGML/XML Web Page. http://www.w3.org/TR/xslt, 1999.

[9] Y. Diao, M. Altinel, M. Franklin, and P. Fischer. Path Sharing and Predicate Evaluation for High-Performance XML Filtering. In *Proceedings of ACM Transactions on Database Systems*, December 2003

[10] Y. Diao, P. Fischer, M. Franklin, and R. To. Yfilter: Efficient and scalable filtering of xml documents. In *proceeding of the International Conference on Data Engineering*, 2002.

[11] F. Fabret, A. Jacobsen, F. Llirbat, J. Pereira, K. Ross, D. Shasha. Filtering Algorithms and Implementations for Very Fast Publish/Subscribe Systems. In *Proceedings of ACM SIGMOD*, pages 115-126, Santa Barbara, California, May 2001.

[12] F. Fabret, A. Jacobsen, F. Llirbat, J. Pereira, D. Shasha. Webfilter: A High-throughput XML-based Publish and Subscribe System. In *Proceedings of The VLDB Journal*, pages 723-724, 2001.

[13] L. Fegaras, D. Levine, S. Bose, V. Chaluvadi. Query Processing of Streamed XML Data. Submitted, November 2001.

[14] R. Goldman and J. Widom. DataGuides: enabling query formulation and optimization in semistructured databases. In *Proceedings of Very Large Data Bases*, pages 436-445, September 1997.

[15] T. Green, G. Miklau, M. Onizuka, D. Suciu. Processing XML Streams with Deterministic Automata. Submitted to *The 9th International Conference on Database Theory*, Siena, Italy, 8-10 January 2003.

[16] A. Gupta, A. Y. Halevy, D. Suciu. View Selection for Stream Processing. Submitted to *Fifth International Workshop on the Web and Databases (WebDB 2002)*, Madison, Wisconsin, June 2002.

[17] Ashish Gupta, Dan Suciu. Stream Processing of XPath Queries with Predicates. In *Proceeding of ACM SIGMOD Conference on Management of Data*, 2003.

[18] Z. Ives, A. Halevy, and D. Weld. An XML query engine for network-bound data. In *Proceedings of Very Large Data Bases*, December 2002.

[19] Z. Ives, A. Halevy, and D. Weld. Efficient Evaluation of Regular Path Expressions on Streaming XML Data. Technical report, *University of Washington*, 2000. Technical Report UW-CSE-200-05-02.

[20] S. Madden and M. J. Franklin. Fjording the stream: An architecture for queries over streaming sensor data. In *Proceedings of the 2002 Intl. Conf. on Data Engineering*, Feb 2002.

[21] G. Miklau, and D. Suciu. Containment and equivalence of an XPath fragment. In *Proceedings of the ACM SIGMOD/SIGART*, pages 65-76, June 2002.

[22] T. Milo, and D. Suciu. Index Structures for Path Expressions. In *ICDT '99, 7th International Conference*, Jerusalem, Israel, January 10-12, 1999, pages 277-295, 1999.

[23] NASA's astronomical data center. ADC XML resource page. http://xml.gsfc.nasa.gov/.

[24] B. Nguyen, S. Abiteboul, and G. Cobena. Monitoring XML Data on the Web. In *Proceedings of the 2001 ACM SIGMOD International Conference On Management of Data*, pages 437-448, May 2001.

[25] D. Olteanu, H. Meuss, T.Furche, and F. Bry. XPath: Looking forward. In *Proceedings of Workshop on XML Data Management (XMLDM)*, LNCS. Springer, 2002.

[26] M. Onizuka. Light-weight XPath Processing of XML Stream with Deterministic Automata. In *Proceedings of CIKM*, 2003.

[27] Protein Information Resources. PIR International Protein Sequence Database. http://pir.georgetown.edu.

[28] A. Snoeren, K. Conley, and D. Gifford. Mesh-based Content Routing Using XML. In *Proceedings of the 18th Symposium on Operating Systems Principles*, 2001.

[29] E. Viglas, and J. Naughton. Rate-based Query Optimization for Streaming Information Sources. In *Proceedings of SIGMOD*, 2002.

[30] W3C (1999) XML path language (XPath) 1.0. http://www.w3.org/TR/xpath

[31] H. Wang, S. Park, W. Fan, and P. S. Yu. ViST: A Dynamic Index Method for Querying XML Data by Tree Structures. In *Proceedings of SIGMOD*, 2003.

Interoperability for Geospatial Analysis: a semantics and ontology-based approach

Zarine Kemp[1], Lei Tan[1] and Jacqueline Whalley[2]

[1]Computing Laboratory, University of Kent
Canterbury, Kent CT2 7RY, U.K.

[2]School of Computer and Information Sciences, Auckland University of Technology,
Private Bag 92006, Auckland 1020, New Zealand

Z.Kemp@kent.ac.uk

Abstract

Information extraction and integration from heterogeneous, autonomous data resources are major requirements for many spatial applications. Geospatial analysis for scientific discovery involves identification of relevant information resources, extraction and fusion of requisite subsets of the information, application of spatial analytical techniques and visualization of the results in an appropriate form. The motivating application domain underlying this research is marine environmental management, although the principles discussed apply to a wide range of scientific disciplines. The research discussed in the paper focuses on integration of data sources, data exploration and interactive data analysis. A knowledge base is used to capture the semantics of the spatial, temporal and thematic dimensions at a domain level, and the context-aware framework exploited to meet the requirements of a varied and distributed user community with differing objectives.

Keywords: information fusion, geospatial analysis, knowledge base, ontologies, visualization.

1 Introduction

Information technologies such as the Internet and Grid computing have revolutionized the way that data resources are discovered and shared. In application domains dependent on geospatial and scientific information, reuse, sharing and dissemination of data is a major requirement. These information repositories are maintained by autonomous organizations, are heterogeneous in structure and semantics and are used by researchers and decision-makers in various contexts and from different perspectives. Interoperability of data and services underpins the next phase of the World Wide Web. Research in distributed databases, integration of structured and semi-structured data and technologies for mediator and information brokers have enabled *syntactical* and *structural* heterogeneities to be overcome. Issues relating to *semantic* heterogeneity are also being tackled using metadata, ontologies and thesauri to express salient concepts and knowledge within a domain of discourse.

In this paper we describe the architecture and framework of a system for environmental information systems. We suggest that in the context of geospatial information systems a data integration approach based on a global monolithic view of data, and a foundational ontology, is not an appropriate solution. We propose an architecture that provides interoperability, querying and analysis capabilities for a community of researchers while maintaining the autonomy of participating data sources. The middleware framework uses an adaptable, scalable knowledge base to accommodate semantic heterogeneity and provide analysis services.

The next subsection describes a motivating application and the data sources in the test bed. Section 2 discusses system requirements and related work. Section 3 presents the system architecture and details of the knowledge base. Section 4, illustrates the interaction model using example queries and section 5 concludes the paper.

1.1 Motivating Application

The system discussed in this paper is based on a platform for marine research and decision support but the requirements and principles are equally applicable to a wide range of application areas. It is intended as a research hub for a community of scientists who pool their information resources and use analytical and visualization tools for monitoring and understanding the marine ecosystem. For example, users may wish to retrieve detailed information about the fishing industry, study phenomena such as algal blooms, explore the changes in biodiversity in a particular part of the ecosystem, retrieve applicable legislation or investigate the effects of anthropogenic activities on particular marine species.

We discuss, briefly, the content and structural characteristics of the data sets in the research test bed emphasizing the geo-referenced attributes of the information stores.

Industrial activity data: the two main activities are fisheries and aggregate dredging for the building industry.

Management of fishing activities is regulated by the Common Fisheries Policy (CFP) legislation of the European Union using sea areas defined by the International Council for the Exploration of the Sea

(ICES). Quotas are allocated by country, species and marine area; these are *ICES Divisions* defined in *vector* format. Data relating to fishery harvesting activities are held in national databases by haul, spatial reference, date/time and species/weight. The spatial reference type in this case is by *ICES statistical rectangle*, a standard grid defined for all EU waters, forming *a hierarchical subdivision* of the quota divisions (illustrated in Figure 2).

Aggregate dredging: these are *vector-defined areas* where licences have been granted for extraction of material from the seabed. Environmental impact assessment reports and research papers may be associated with these activities.

Research data: Annual surveys are conducted by research centres based in different countries. The data sets consist of environmental information such as sea surface temperature, salinity, seabed type and biomass abundances by species. The location of sampling stations (*geo-referenced point*) is stored with the primary data sets to enable variables to be subsequently interpolated over the spatial extent of interest using an appropriate interpolation technique.

Ad hoc surveys: for example of benthic fauna provide data sources at fine spatial resolutions and are stored as *point samples* in the database.

Other related data: Legislation applicable to activities, species and habitats in marine environments. The statutes refer to areas directly using geographic coordinates or, indirectly, by reference to habitats for endangered species.

Base data of the geography of the research area including coastlines, ports and rivers are held in *vector format* using standards such as ESRI .shp files (ESRI).

In addition to the data sources, marine researchers and managers subscribe to domain-related ontologies. We have included two typical global ontologies: an ontology of marine species which consists of a tree-structured biological taxonomy and a more complex marine habitat multilevel classification that is becoming a European standard (Connor *et al* 2004).

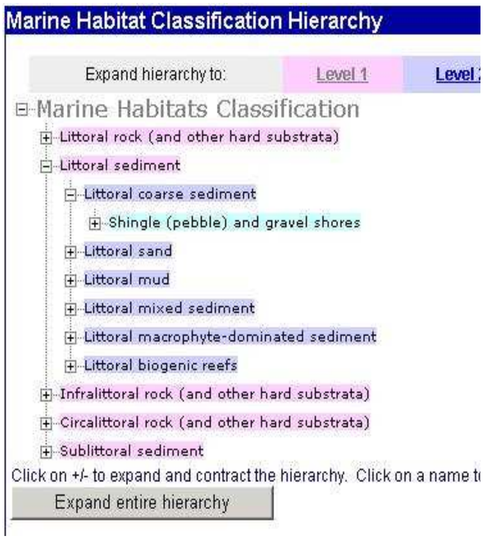
2 Requirements and Related Work

The primary role of the middleware is to provide the abstractions and services that enable the development and deployment of user-level applications in a heterogeneous, distributed, computing environment. It must also be geared to the geospatial requirements of marine environment research communities as described by Tsontos and Kiefer (2003). From the computational perspective, the system should:

- support discovery of, and access to distributed, heterogeneous information sources
- provide tools for representation, manipulation and visualization of spatiotemporal and scientific data
- be adaptable to enable data sources, semantic information and services to evolve according to the requirements of the research community.

Halevy *et al* (2003) discussed the notion of *interoperability* across the *structure chasm,* that is, over sources that encompass *structured* and *unstructured* data. More recently, the notion of *dataspaces* has been proposed as a data management abstraction with associated *DataSpace Support Platforms* (DSSPs) to provide the required services (Franklin *et al* 2005). The middleware described in this paper encompasses several requirements of dataspaces using capabilities of extended database management systems. Interoperability is based on XML-based mediation techniques for data sets in relational or object-relational databases (Wiederhold 1999). The knowledge base enables links between data sources to be represented and supports keyword-based information retrieval and querying.

A major characteristic of computation in the geographic sciences domain is the pervasiveness of the *space-time-theme* composite. Understanding phenomena in geo-scientific domains requires queries and analyses to be predicated in terms of these three dimensions. Theories underlying these dimensions and their representation in data management systems have been discussed by researchers including Buckland and Lancaster (2004) and Smith and Mark (1998). A consequence of this is that interoperability platforms have to incorporate an understanding of, and mappings between, different conceptual views of space. Details of these are beyond the scope of this paper but reference may be made to international standards for geospatial data such as ISO 19115 (2003) and OGC (2003) and various classifications of space such as the object and field view space or the vector-raster views of space, reflecting alternative conceptual spatial representations. In the marine domain classifications of space may also involve complex hierarchies such as the Joint Nature Conservation Committee habitat classification (Connor *et al*, 2004). Figure 1 illustrates a small subset of this classification.


Marine Habitat Classification Hierarchy

Expand hierarchy to: Level 1 Level

- Marine Habitats Classification
  - Littoral rock (and other hard substrata)
  - Littoral sediment
    - Littoral coarse sediment
      - Shingle (pebble) and gravel shores
    - Littoral sand
    - Littoral mud
    - Littoral mixed sediment
    - Littoral macrophyte-dominated sediment
    - Littoral biogenic reefs
  - Infralittoral rock (and other hard substrata)
  - Circalittoral rock (and other hard substrata)
  - Sublittoral sediment

Click on +/- to expand and contract the hierarchy. Click on a name t

Figure 1. JNCC habitat classification (subset)

Similarly, multiple representations also arise in the case of the temporal and thematic dimensions. The middleware platform in this research uses ontologies in the knowledge base to provide mappings between alternative spatiotemporal classifications as discussed in Kemp and Frank (2005).

Users in a research community use their expertise and experience to guide and inform the data they collect and the analyses they undertake. Bouquet *et al* (2004) suggest that application domain knowledge may be included by 'contextualizing ontologies'. We propose extending global ontologies to incorporate community (local) or regional context using the knowledge base, as illustrated in the following example.

A marine research community in the UK may include data on fishing activity which refers to regulatory areas for quota allocations and recording of catch statistics. These areas are specific instances of a generic vector-defined marine area feature. Extending the spatial ontology to include this contextual knowledge makes this semantic information explicit, facilitates data interoperability and enables users to query and analyse the information in a meaningful way. The map in Figure 2 shows a spatial containment hierarchy of marine areas referred to in section 1.1. It shows ICES Divisions (large non-uniform spaces), ICES rectangles (cells within each division) and the fine scale research grid (Eastern English Channel).

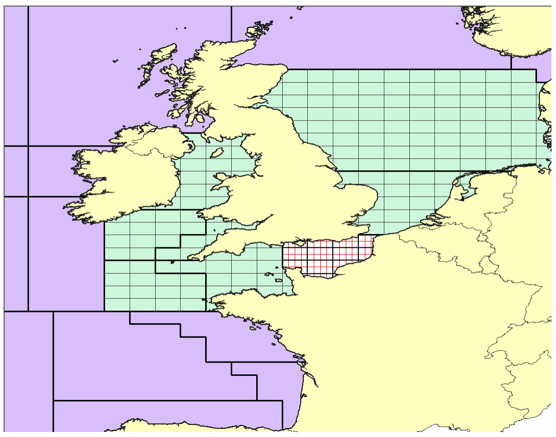


Figure 2: A nested hierarchy of marine areas

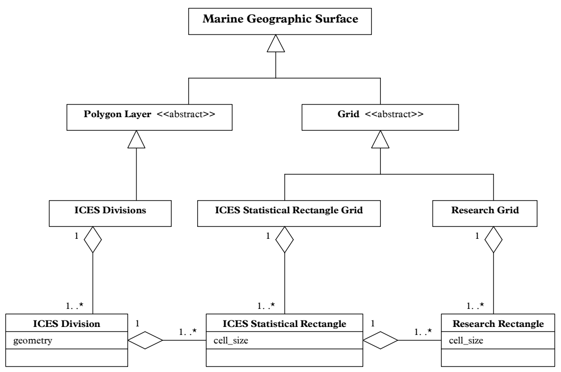

Marine Geographic Surface
Polygon Layer <<abstract>>
Grid <<abstract>>
ICES Divisions
ICES Statistical Rectangle Grid
Research Grid
1
1..*
ICES Division
geometry
ICES Statistical Rectangle
cell_size
Research Rectangle
cell_size

Figure 3. Part of the spatial ontology

Figure 3 shows part of the spatial ontology in the knowledge base (using the OGC standard) which has been *extended* at the lowest level to include additional semantics of marine space of relevance to the research community.

Ontologies have been used in many applications to enable shared concepts to inform the research. Gangemi *et al* (2002) describe a detailed ontological framework for fisheries. This FAO (United Nations Food and Agriculture Organisation) initiative provides a platform for unifying different thesauri, topic trees and taxonomies to provide a formal, integrated ontological framework for the fishery application domain. The semantic framework from this initiative is similar to the ontology of the thematic dimension included in our knowledge base. In our system, spatial conceptualizations have to be integrated with other dimensions, temporal and thematic. Wadsworth *et al* (2005) identify a problem with using ontologies, namely that of reconciling, alternative overlapping conceptualizations. They describe a semantic-statistical methodology for quantifying overlaps to resolve the problem.

In many information systems, the ontologies, i.e. the semantics underlying retrieval and querying of data are completely hidden from the user. However, In some situations scientists need to traverse the concept hierarchies to enable them to specify the parameters for tasks that constitute the workflow. In this system we have enabled a navigational form of querying where some level of semantic information has to be identified. A simple example involves querying a data set using a variable that is an element of a hierarchical classification. The global taxonomy of marine species is such a hierarchical classification. If the user wishes to query a data set at the 'genus' level, then the user can indicate that aggregation level for the analysis. The system deduces that the concept 'species' in the data sources is subsumed under the more general category 'genus' and returns the aggregated values as required. The design of the knowledge base also enables representation of the semantics of different types of relationships.

3 Architecture of the Framework

3.1 Overview

In this section we present an overview of the prototype system that was implemented for this research. Much of the functionality provided by the framework consists of dynamic composition of data and services. Typical services consist of:

- flexible extraction of subsets from the heterogeneous resources, dependent on user-specified parameters
- data discovery at different levels of abstraction
- sub-sampling, reclassification and re-gridding of extracted data, if required
- processing data by applying computational models
- visualization of output in textual, tabular, graphic or cartographic format.

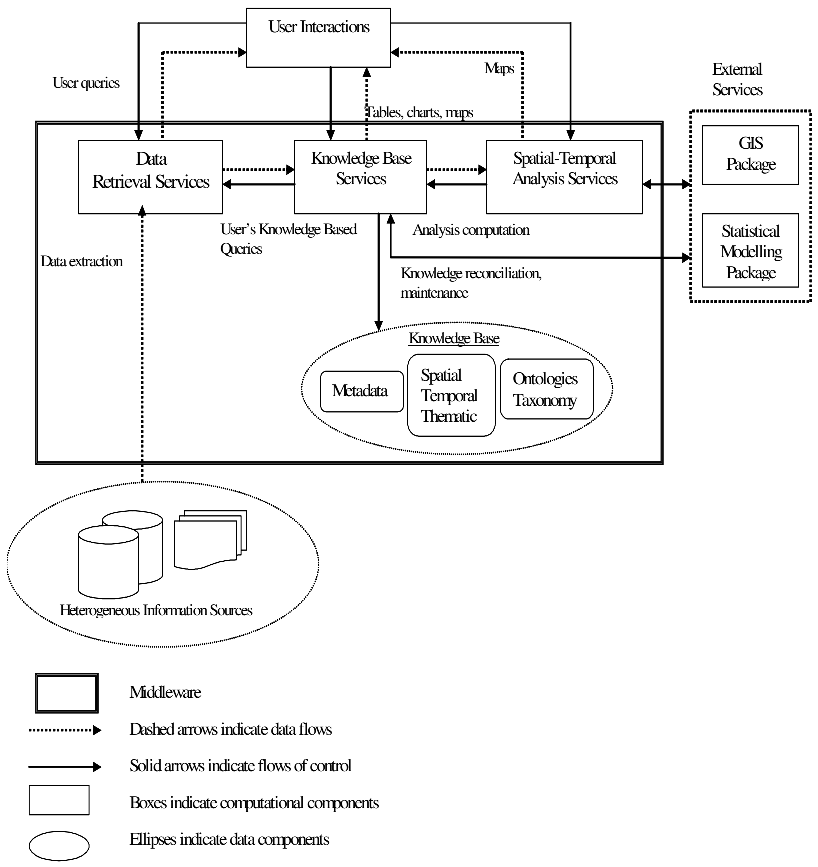

User Interactions
User queries
Maps
Tables, charts, maps
External Services
Data Retrieval Services
Knowledge Base Services
Spatial-Temporal Analysis Services
GIS Package
Statistical Modelling Package
User's Knowledge Based Queries
Analysis computation
Knowledge reconciliation, maintenance
Data extraction
Knowledge Base
Metadata
Spatial Temporal Thematic
Ontologies Taxonomy
Heterogeneous Information Sources
Middleware
Dashed arrows indicate data flows
Solid arrows indicate flows of control
Boxes indicate computational components
Ellipses indicate data components

Figure 4. Architecture of the framework – system components interaction

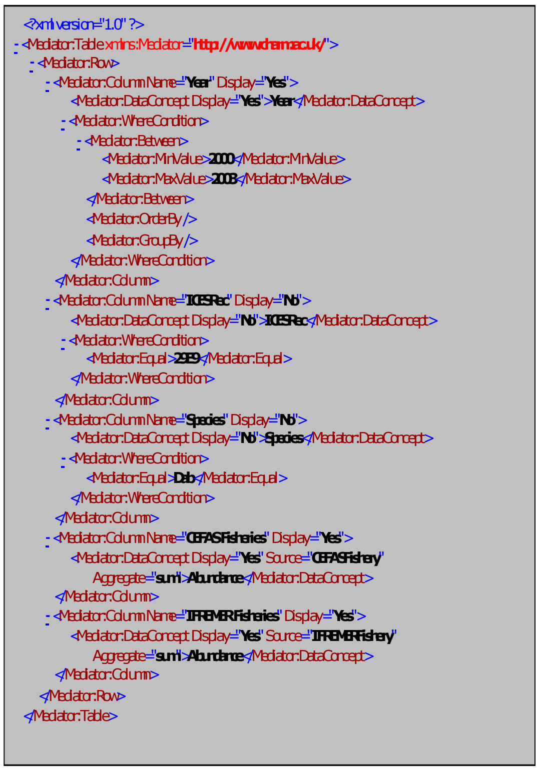
The diagram in Figure 4 provides a high-level view of the system architecture showing the interaction between the computational and data components of the system:

- The *Data Retrieval Services* provide the functionality for querying different data sources. Users' queries are accepted and mediator services used to retrieve and combine the required information from the data sources.
- The *Knowledge Base Services* provide more advanced searching and functionality. These services include ontology-based searching, linking data sources using thesauri and spatial-temporal indexing.
- The *Analysis Services* provide computational capabilities including linking to external software packages for enviromental analyses using the data extracted by the *Knowledge Base Services*. The external capabilities include a GIS (Geographical Information System) and a statistical modelling package.
- *User Interaction Services* provide users with a graphical interface to enable specification of parameters and present results in appropriate formats.

The system components identified in Figure 4 have been implemented as interacting packages. Each package consists of services that implement the functionality provided by that package. The main system components include: the QueryGUI package that encompasses the interaction model of the framework, the MediatorWrapper package that provides the functionality for linking disparate data sources as required by specific tasks and the XMLDataLink package that provides the utility classes to interpret XML documents.

3.2 User Interaction Services

In the system framework, an XML format file is designed to represent users' queries, and data access services are implemented as a set of Java objects. An example XML query file is shown in Figure 5. The XML files are parsed according to the DOM (Document Object Model) standard using the JAXP (Java API for XML Parsing) (JAXP, DOM , Ungerer and Goodchild 2003). The links to relational databases are implemented by the JDBC to ODBC driver provided in the Java library. Thus the data sources can by distributed over a network environment.



```
  <?xml version="1.0" ?>
- <Mediator:Table xmlns:Mediator="http://www.cham.ac.uk/">
  - <Mediator:Row>
    - <Mediator:Column Name="Year" Display="Yes">
        <Mediator:DataConcept Display="Yes">Year</Mediator:DataConcept>
      - <Mediator:WhereCondition>
        - <Mediator:Between>
            <Mediator:MinValue>2000</Mediator:MinValue>
            <Mediator:MaxValue>2003</Mediator:MaxValue>
          </Mediator:Between>
          <Mediator:OrderBy />
          <Mediator:GroupBy />
        </Mediator:WhereCondition>
      </Mediator:Column>
    - <Mediator:Column Name="ICESRec" Display="No">
        <Mediator:DataConcept Display="No">ICESRec</Mediator:DataConcept>
      - <Mediator:WhereCondition>
          <Mediator:Equal>29E9</Mediator:Equal>
        </Mediator:WhereCondition>
      </Mediator:Column>
    - <Mediator:Column Name="Species" Display="No">
        <Mediator:DataConcept Display="No">Species</Mediator:DataConcept>
      - <Mediator:WhereCondition>
          <Mediator:Equal>Dab</Mediator:Equal>
        </Mediator:WhereCondition>
      </Mediator:Column>
    - <Mediator:Column Name="CEFASFisheries" Display="Yes">
        <Mediator:DataConcept Display="Yes" Source="CEFASFishery"
            Aggregate="sum">Abundance</Mediator:DataConcept>
      </Mediator:Column>
    - <Mediator:Column Name="IFREMERFisheries" Display="Yes">
        <Mediator:DataConcept Display="Yes" Source="IFREMERFishery"
            Aggregate="sum">Abundance</Mediator:DataConcept>
      </Mediator:Column>
    </Mediator:Row>
  </Mediator:Table>
```

Figure 5. XML file representing a query.

The QueryGUI package provides the functionality for user interaction. Users interact with the user interface of package to submit queries to the underlying data sources via the MainFrame object. The AddQueryButton adds a new QueryPanel to the MainFrame. The RemoveButton will remove the last QueryPanel from the MainFrame. The user can add as many QueryPanels to the MainFrame as required. The QueryPanel enables users to specify the data sources, concepts (i.e. variables) and logical conditions involved in a query. Options are also provided to so that users can specify the style and format of the displayed results.

3.3 Data Retrieval Services

The main purpose of this interface is to transform the XML query to the underlying data source query language. Each information source implements a wrapper interface called *DataSource* that must be registered with the *Mediator*. Wrappers consist of the structure specification of a data source and an understanding of the transformation between XML documents and the underlying data.

The MediatorWrapper package implements part of the wrapper-mediator methodology for interoperable data sources as described by Wiederhold (2000). When the user clicks on the "Execute" button on the user interface, the query is generated as an XML file and sent to the *Mediator*. The *Mediator* interprets the XML file, sends it to the relevant data source wrapper according to the content of the XML file, and combines the results returned from each data source for the user. The mediator does not have a global mediated schema that is shared by all the participating data sources. It is only aware of the data sources that are registered currently. It also gets relevant information from the XML query file to rearrange and merge the results returned from the data sources. The results returned from each data source are *ResultSet* type objects; they are combined into an *ArrayList* of *ArrayLists* object by the *combineResult2* method in the *Mediator* class.

The *XMLDataLinkUtility* package provides the utility classes that can be used by other classes, such as the *XMLDataUtility*; it contains methods to interpret XML documents. The *DataConcept* class represents the thematic concept on which a user's query is based, such as 'year', 'species', 'abundance'. There are classes in this package to map the terminologies used by users in the *DataConcept* objects to the equivalent terms used in the individual data sources. The *DataLink* classes also have the necessary knowledge of the underlying data sources, schemas or structures to form a query. If the underlying data source is a relational database, the associated *DataLink* object holds information about relation names and attribute names in each relation, and the links between relations. If the underlying data source is in XML format, the *DataLink* object is aware of the document type definition associated with each XML file.

3.4 Spatiotemporal Analysis Services

The Spatial-Temporal Analysis Services add computational capabilities for the specialized analytical tasks required by marine scientists. The example modelling and simulation task used in the prototype involves calculating Habitat Suitability Indices (HSI) for different marine locations for a species. These indices are then used to arrive at a classification of habitats for that species, presented to the user in cartographic format.

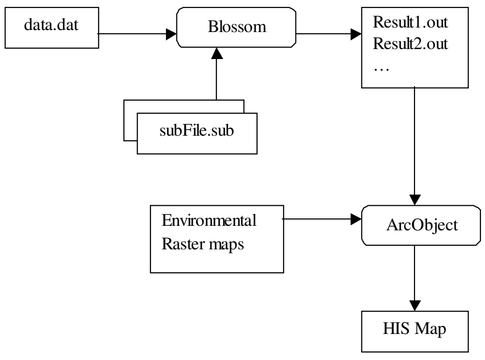


Figure 6. HSI Workflow diagram

Figure 6 illustrates the workflow involved in the analysis process in this component. The user interacts with the main interfaces to specify parameters for the analysis: environmental variables, temporal range and species. The software extracts relevant data from the data source, generates an ASCII text file in the format required by the statistical package Blossom (2005). The application software interacts with Blossom commands to carry out the analyses. The output returned by the Blossom package is interpreted and used to produce Habitat

Suitability maps. Several map generating and manipulation functionalities are used in producing the final and intermediate maps: interpolation methods are used to generate raster maps for the environmental variables, and map algebra techniques are used to combine the separate environmental maps into the Habitat Suitability Index maps.

3.5 Knowledge Base Services

The Knowledge Base, discussed further in the subsection 3.5.1, maintains a repository of ontologies that represent users' understanding of relevant domain concepts. The metamodel incorporates semantics and links between the global ontologies (biological taxonomies, habitats etc.), ontologies for dimensions space, time and theme and the underlying data collections. The Knowledge Base Services provide the functionality that enables users to express queries in terms that are relevant to them. The code in this component is similar, at the design level, to the services in the *QueryGUI* package (section 3.2).

The user interaction model is an important aspect of this component. Its interface includes browsing capabilities for two reasons: first, it reveals to users the classification schemes and hierarchies available to allow for semantics-based querying (illustrated in section 4.1) and second, it enables them to identify the level of the hierarchy required. This selection prompts the system to generate aggregate variable values at the required level of abstraction. This feature is illustrated in the example in section 4.2.

Another important aspect of the Knowledge Base services functionality is to resolve semantic differences between data sources. The case study in section 4.3 serves to illustrate this feature using the example of inconsistency in the classification of the environmental variable *sediment.* The knowledge services use a reconciliation method to reclassify the hierarchy in the data resource and map it to the global domain classification for marine habitats.

3.5.1 The Knowledge Base

Georeferenced digital libraries and web-based search engines as described in Janee and Frew (2002) are frequently underpinned by carefully curated ontologies and gazetteers. In the case of our system framework, the data schemas, diverse ontologies, classifications, taxonomies and thesauri all represent relevant information. Unifying a wide collection of semantic fragments into a definitive well-crafted knowledge base is a major challenge as discussed by Frank and Kemp (2001). Another characteristic of the diverse data resources is that they overlap in their content to varying degrees. In order to accommodate the structural and semantic diversity and to provide links between the data sources in the application and the scientific domain related concepts we provide layered conceptual *domain knowledge model*. In addition to articulating the semantics at various levels of abstraction, our framework encapsulates the associated services required for interoperability in multidimensional, hierarchical information spaces (Kemp and Lee 2000). Zaslavsky *et al* (2003) describe a similar system based on the Open GRID Services Architecture as a community cyberinfrastructure.

The knowledge base consists of a layered structure that hides the complexity and diversity of the information resources from end users. It consists of three types of objects: metadata objects, dimension ontologies and global or domain ontologies.

The *metadata layer* consists of metadata objects (one per information source) that encapsulate collection or document level information about each data source conforming to standards prevailing in the marine geoscience community. They contain administrative and access information, details of the provenance of the data sources, lineages and approximations of the spatial and temporal extents of the underlying data. These coordinates enable quick 'first pass' searches over the data sets available in the information base. Metadata objects also include information on the format/data type of the spatial and temporal attributes in the collection to determine the appropriate level of spatial integration when data are extracted from more than one resource. Many geoscientific data portals provide metadata views for tasks such as data discovery, access services and indication of fitness-for-use. Our knowledge base enables each metadata object to be linked with one or more of the types in the dimension ontologies to enable access to a range of spatial and temporal services.

The components of the *dimension level metadata* objects perform two functions. They articulate the domain concepts that enable users to specify the spatial, temporal and thematic parameters relevant to queries. They also provide links to the underlying information sources to enable transparent interoperability over the different data sets. As most queries in environmental analysis examine attributes with reference to the space-time-theme composite, three ontologies have been provided at this level: the spatial hierarchy, the temporal hierarchy and the thematic classification. Figures 2 and 3 (in section 2) illustrate part of the spatial ontology specialized by community-related context. Each spatial class in the ontology (Figure 3) is represented by its type_name, textual label, textual description and structural and functional specification. The classes in the bottom layer of the ontology instantiate the aggregation semantics. Thus, for example, a particular ICES Division may be identified by its code (VIId), its complete or part textual description (Eastern English Channel), its defining coordinates (MBR: minimum bounding rectangle) and by direct interaction at the user interface. The ontology specification also enables the aggregation of aspatial attributes of spaces *contained_in* the specified area.

Similarly, the temporal hierarchy can provide several perspectives on time. For example, a linear temporal view enables investigation of phenomena using operators based on temporal logic such as *overlap, touch, disjoint* and so on. An alternative classification may be based on seasons as shown in Figure 7.

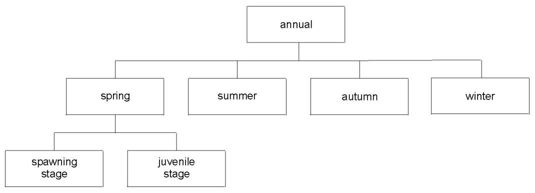


Figure 7. Temporal seasonal classification

The seasonal classification is relevant for the framework and could be implemented using either lookup tables or functions depending on the spatial extent of the analysis. In the current implementation, temporal attributes are identified simply as *time points* and *temporal intervals* with associated operators.

The thematic ontology consists of a hierarchy of textual terms classified according to the main categories of information sources in the testbed: 'fisheries', 'legislation', 'research', 'benthos' and 'dredging'. Concept terms link to other domain related concepts, using *hierarchical* (*broader/narrower term*), *associative* and other relationships. Some of these concepts are also linked to the global ontologies in the knowledge base. For example, linking the concept 'species' to the global taxonomy for marine species, provides access to the semantics of the biological taxonomy.

The *reference ontology layer* contains the global semantics applicable to the marine domain. Information at this level refers to global repositories such as those supported by the Global Biodiversity information Facility (GBIF), Taxonomic Databases Working Group (TDWG) and the Ocean Biodiversity Information System (OBIS). In our prototype system we have included a structured biological taxonomy of the species that occur in the marine area of interest and are used for various tasks such as providing the infrastructure for associating different common names for scientifically identified species, aggregating data at various levels of the hierarchy and indirectly enabling the underlying data sets to be linked for ad hoc analysis. A more complex example of an ontology at this level is the detailed classification of marine habitats (illustrated in Figure 1). This ongoing European initiative on defining a classification for marine spaces, starts with fairly coarse classifications based on a few major physical parameters and proceeds through successive levels of refinement to include topographic features and biotic communities associated with the ecological units. In the current version of the project the main use of this ontology is to enable users to define habitat suitability indexes for relevant species depending on the variables available in the underlying data sets.

4 Geospatial information retrieval: case studies

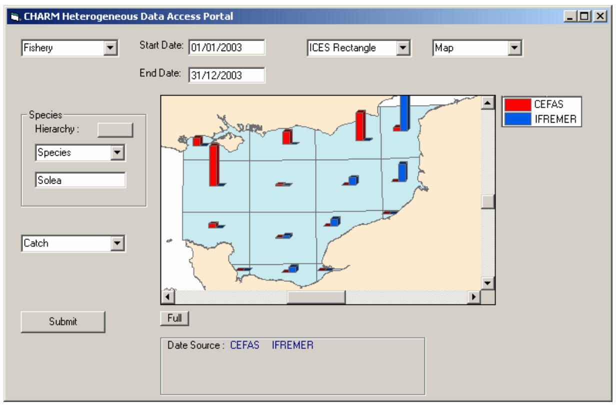
In this section the capabilities of the research framework are illustrated using typical queries and analysis tasks.

4.1 Interoperability of data sources

The first example illustrates retrieval of data from two heterogeneous fishery databases. The query parameters specify retrieval of:

- Theme: *fishery*, subtheme *catch*
- Theme: species *Solea solea*
- Time: *temporal interval* (calendar dates)
- Space: *ICES rectangle* (the second level of the spatial hierarchy, illustrated in Figure 2)
- Display mode: *map*

Figure 8 illustrates the output showing the requied variables from two separate national databases.

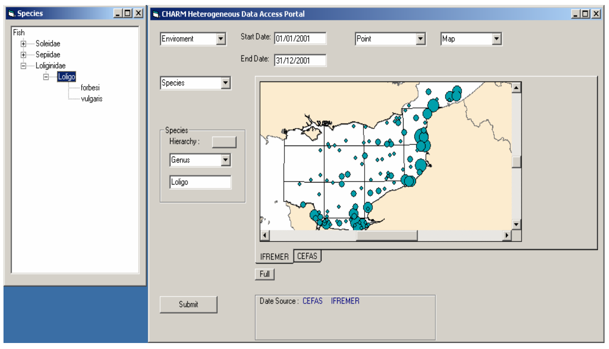

CHARM Heterogeneous Data Access Portal

Fishery
Start Date: 01/01/2003
End Date: 31/12/2003
ICES Rectangle
Map
Species
Hierarchy :
Species
Solea
Catch
🟥 CEFAS
🟦 IFREMER
Date Source : CEFAS IFREMER

Figure 8. Data from heterogeneous databases

4.2 Aggregation of data at user-specified level of concept hierarchy

This example illustrates the interaction of an information source with one of the semantic hierarchies in the knowledge base, the biological taxonomy. The user has navigated the taxonomy and selected the genus *Loligo* as the aggregation level for thematic information to be retrieved. The abundance values refer to the data collected in annual research surveys. Figure 9 shows the abundance of this genus (all species aggregated), from the identified input source, in cartographic format. In this case, the visualization of the *spatial* dimension is in *point* form, with the size of the icons of the sampling locations reflecting the relative values of the abundance. The ICES grid is superimposed on the map for visual reference, for example for industrial catch of the same genus.

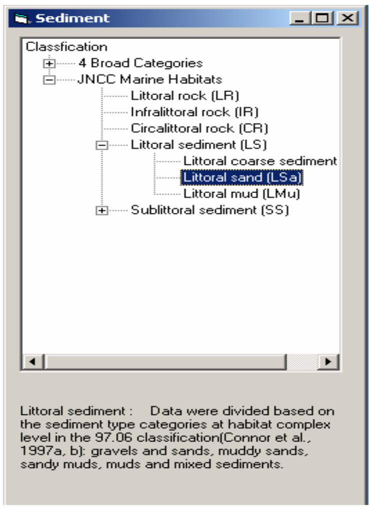

Species
Fish
Soleidae
Sepiidae
Loliginidae
Loligo
forbesi
vulgaris

CHARM Heterogeneous Data Access Portal
Enviroment
Start Date: 01/01/2001
End Date: 31/12/2001
Point
Map
Species
Species
Hierarchy :
Genus
Loligo
IFREMER
CEFAS
Date Source : CEFAS IFREMER

Figure 9. Aggregated abundance of selected genus

4.3 Visualization based on ontology-defined classification

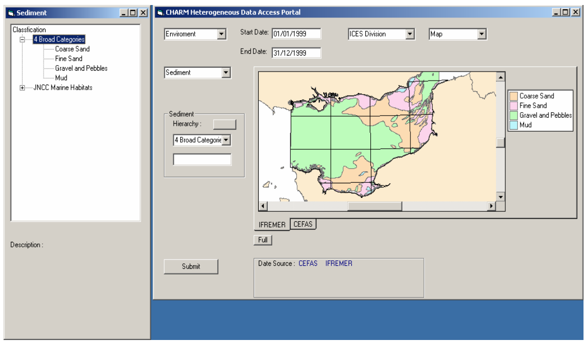
This example illustrates the use of a domain level (global) ontology, the habitat classification (Connor *et al,* 2004), to reclassify an alternative classification in one of the data sources. The ontology includes seabed sediment classes in its definition of marine areas at level 2 (see Figure 1). The first frame, Figure 10 (A), shows a small subsection of this classification where the class 'Littoral sediment' (LS) is further subdivided into three subclasses, 'Littoral coarse sediment' (LCS), 'Littoral sand' (LSa) and 'Littoral mud' (Lmu). This particular research survey database uses an alternative sediment classification system (Larsonneur *et al* 1979), which contains four subclasses at this level: 'Coarse sand', 'Fine sand', 'Gravel and pebbles' and 'Mud' as shown in Figure 10 (B). When this data set is used locally, this classification is appropriate. However, when it is integrated with data sets from other national data sets, the global ontology is used to *reclassify* it to achieve semantic consistency. Figure 10 (C) shows the same data in map form where the original categories 'Fine sand' and 'Coarse sand' have been merged for equivalence with the ontological class 'Littoral sand'. Thus the framework enables individual databases to maintain local heterogeneity and also provides a reclassification service, when required, for global interoperability.


Sediment

Classfication
- 4 Broad Categories
- JNCC Marine Habitats
  - Littoral rock (LR)
  - Infralittoral rock (IR)
  - Circalittoral rock (CR)
  - Littoral sediment (LS)
    - Littoral coarse sediment
    - Littoral sand (LSa)
    - Littoral mud (LMu)
  - Sublittoral sediment (SS)

Littoral sediment : Data were divided based on the sediment type categories at habitat complex level in the 97.06 classification(Connor et al., 1997a, b): gravels and sands, muddy sands, sandy muds, muds and mixed sediments.

Figure 10(A) Global ontology


Sediment

Classfication
- 4 Broad Categories
  - Coarse Sand
  - Fine Sand
  - Gravel and Pebbles
  - Mud
- JNCC Marine Habitats

Description :

CHARM Heterogeneous Data Access Portal

Enviroment | Start Date: 01/01/1999 | End Date: 31/12/1999 | ICES Division | Map

Sediment

Sediment — Hierarchy : 4 Broad Categorie

Coarse Sand, Fine Sand, Gravel and Pebbles, Mud

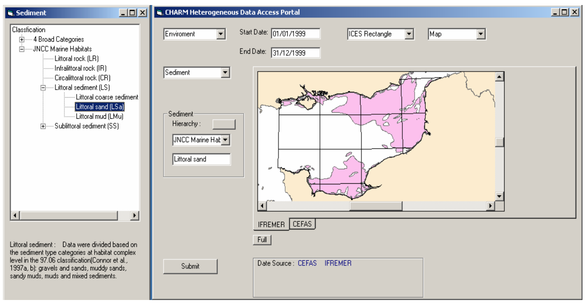
IFREMER | CEFAS

Full

Submit

Date Source : CEFAS IFREMER

Figure 10(B) Local classification (4 sub classes)


Sediment

Classfication
- 4 Broad Categories
- JNCC Marine Habitats
  - Littoral rock (LR)
  - Infralittoral rock (IR)
  - Circalittoral rock (CR)
  - Littoral sediment (LS)
    - Littoral coarse sediment
    - Littoral sand (LSa)
    - Littoral mud (LMu)
  - Sublittoral sediment (SS)

Littoral sediment : Data were divided based on the sediment type categories at habitat complex level in the 97.06 classification(Connor et al., 1997a, b): gravels and sands, muddy sands, sandy muds, muds and mixed sediments.

CHARM Heterogeneous Data Access Portal

Enviroment | Start Date: 01/01/1999 | End Date: 31/12/1999 | ICES Rectangle | Map

Sediment

Sediment — Hierarchy : JNCC Marine Hat — Littoral sand

IFREMER | CEFAS

Full

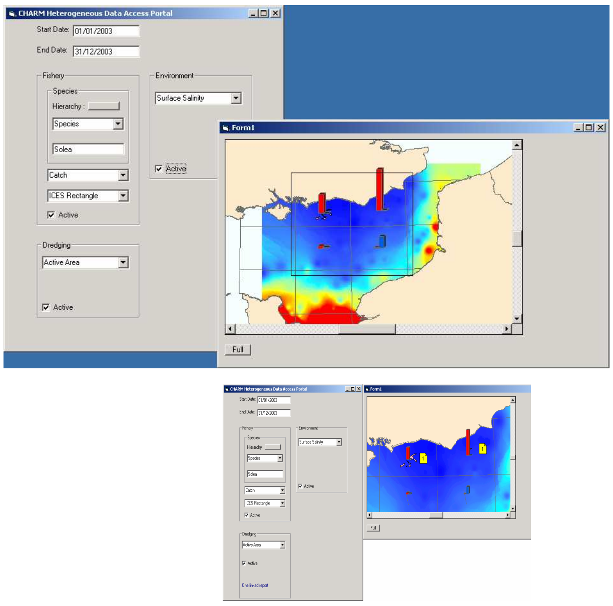
Submit

Date Source : CEFAS IFREMER

Figure 10(C). Reclassification of sediment types (3 subclasses)

4.4 User-specified spatial search and multiple thematic retrieval

This example illustrates the discovery of multi-theme data and related information. User interaction in this example starts with an interactively specified rectangle of interest as shown in the upper window of Figure 11.


CHARM Heterogeneous Data Access Portal

Start Date: 01/01/2003

End Date: 31/12/2003

Fishery — Species — Hierarchy : — Species — Solea — Catch — ICES Rectangle — ☑ Active

Environment — Surface Salinity — ☑ Active

Dredging — Active Area — ☑ Active

Figure 11. Multi-theme output and retrieval of linked information

The system reveals the data sources relevant to the search space and enables the user to refine the query by specifying required parameters. In this example, the environmental variable *surface salinity* is displayed as a raster map (interpolated from point samples in the research database), overlaid with information relating to *fishery catch* data for species *Solea solea* by *ICES rectangles*. Unstructured information in the user-specified search rectangle referring to active dredging areas is also displayed in the lower window. The knowledge base enables the system to discover that the dredging areas have *association* links with text documents (research reports). The existence of the documents is indicated on the map using document shaped icons which also indicate the number of relevant documents discovered (1 in this case). Clicking on the hyperlinks displays the contents of the documents.

5 Conclusion and future work

Initial results of this research are promising. Flexible and open-ended support for scientists and decision-makers can be provided by enabling interoperability across dispersed heterogeneous information sources coupled with appropriate metadata and semantic knowledge.

The future of the World Wide Web will involve scientific domains with a large number of existing metamodels and ontologies (Costello and Vanden Berghe 2006). There will also be increasing requirements to extend or contextualize existing ontologies and map between different ontological specifications. A related requirement is to enable the ontological resources (knowledge bases) to evolve and be easily updated, as data sets and metadata models are added to the resources for a research community (Reinoso-Castillo *et al* 2003). In our system, the collection level metadata and the ontologies are part of the knowledge base which functions as a community resource at a central hub. This makes it easier to extend the range and type of information resources and related semantics available to researchers in a scientific domain.

There are several interesting directions for future work.

Ontologies and reasoning: Investigation of formal ontology specification languages to enable reasoning with multiple ontologies in complex scientific domains.

Real-time response: Many environmental monitoring tasks such as early warning systems for natural hazards require real-time responses. Wiederhold (2000) has suggested that real-time simulations should be an integral part of decision-making frameworks. A future extension of this research will consider the design and performance implications of this requirement.

Platform for creating and maintaining the knowledge base: It would be useful to include an interface to enable users to discuss and update the knowledge model as a collective activity. This could be similar to a 'blackboard' component in decision support systems. Such a facility would encourage cooperative decision-making and thus be of interest in areas such as the marine domain where there exist recognized conflicts of interest between user groups such as the fishing industry and marine biologists.

Ontologies and reasoning: Investigation of formal ontology specification languages to enable reasoning with multiple ontologies in complex scientific domains.

Distributed framework: The design of the framework and the underlying technologies assume that the data, users and computational framework are distributed. To achieve a disciplined model for the framework, standards such as WSDL, UDDI and SOAP should also be investigated as well as the promise of the GRID architecture for a federated infrastructure as discussed by Watson (2005).

6 References

Blossom: Blossom Statistical Software. US Geological Survey. http://www.fort.usgs.gov/products/software/blossom/blossom.asp Accessed 8 Aug. 2006.

Bouquet, P., Giunchiglia, F., van Harmelen, F., Serafini, L. and Stuckenschmidt, H. (2004):Contextualizing ontologies. *Journal of Web Semantics* 1(4): available online: http://www.websemanticsjournal.org/ps/pubs/2004-20

Buckland, M. and Lancaster, L. (2004): Combining Place, time and Topic. *D-Lib Magazine* **10**(5), ISSN 1082-9873.

CFP: The European Union Common Fisheries Policy.http://ec.europa.eu/dgs/fisheries/index_en.html

Accessed May 2006.

Connor, D.W., Allen, J.H., Golding, N., Howell, K.L., Lieberknecht, L.M., Northen, K.O. and Reker, J.B., (2004): The Marine Habitat Classification for Britain and Ireland, Version 04.05 JNCC, Peterborough, ISBN 1 861 07561 8.

Costello, M. and Vanden Berghe, E. (2006): 'Ocean Biodiversity Informatics': a new era in marine biology research and management. *Marine Ecology Progress Series* **316**: 203-214.

DOM: Document Object Model, http://www.w3.org/DOM/

ESRI: Environmental Systems Research Institute, http://www.esri.com/ Accessed Jan. 2006.

Frank, R. and Kemp, Z. (2001): Ontologies for knowledge discovery in environmental information systems. Proc. *Workshop on Complex reasoning on geographical data (CRGD),* Raffaeta, A. and Renso, C. (Eds). December, 2001, pp 15-30, Paphos, Cyprus, 2001.

Franklin, M., Halevy, A. and Maier, D. (2005): From Databases to Dataspaces: A New Abstraction for Data Management. *ACM SIGMOD* **34**(4): 27-33.

Gangemi, A., Fisseha, F., Pettman, I., Pisanelli, D.M., Taconet, M. and Keizer, J. (2002): A Formal Ontological Framework for Semantic Interoperability in the Fishery Domain, *Proc. ISCW 2002, International Semantic Web Conference*, June 9-12, Sardinia, Italy.

GBIF: Global Biodiversity Information Facility. http://www/gbif.org/ Accessed Aug. 2006.

Halevy, A., Etzioni, O., Doan, A., Ives, Z., Madhavan, J., McDowell, L. and Tatarinov, I. (2003): Crossing the Structure Chasm. *Proc. Conference on Innovative Database Research (CIDR 2003),* Asilomar, CA, USA. January, 2003).

ICES: International Council for the Exploration of the Sea, http://www.ices.dk/ Accessed May, 2006.

ISO: International Standards Organization. http://www.iso.org/ Accessed Jul. 2006.

Janeé, G. and Frew, J., (2002): The ADEPT Digital Library Architecture, *ACM / IEEE Joint Conference on Digital Libraries*, July 2002, Portland, Oregon.

JAXP: Java API for XML Processing. http://java.sun.com/webservices/jaxp/ Accessed June 2005.

Kemp, Z. and Lee, H. (2000): A Multidimensional Model for Exploratory Spatiotemporal Analysis. *Proc. 5th International Conference on GeoComputation*, Carlisle, B. and Abrahart, R. (Eds). University of Greenwich, UK.

Kemp, Z. and Frank, R. (2005): Knowledge representation and semantic interoperability in marine information systems. In *GIS/Spatial Analyses in Fishery and Aquatic Sciences (Vol. 2)*. Nishida, T., Kailola, P. and Hollingworth, C. (Eds). Fishery-Aquatic Research Group, Saitama, Japan. 735 pp. ISBN: 4-9902377-0-6.

Larsonneur, C., Vaslet, D., J.-P. Auffret. (1979): Les Sédiments Superficiels de la Manche, Carte Géologique de la Marge Continentale Française. *Bureau des Recherches Géologiques et Minières, Ministère de Industrie, Service Géologique National*, Orléans, France.

OBIS: Ocean Biogeographic Information System. .http://www.iobis.org/ Accessed Aug. 2006.

OGC: Open geospatial Consortium. http://www.opengeospatial.org/ Accessed Jul. 2006.

Reinoso-Castillo, J., Silvescu, A., Caragea, D., Pathak, J., and Honavar, V. (2003): Information extraction and integration from heterogeneous, distributed, autonomous information sources – A federated ontology-driven, query-centric approach. In *IEEE International Conference on Information Integration and Reuse*, Las Vegas, Nevada, 2003.

Smith, B. and Mark, D. (1998): Ontology and Geographic Kinds. *Proc. 8th International Symposium on Spatial Data Handling, SDH '98*, Poiker, T. amd Chrisman, N. (Eds). Vancouver, Canada, July 1998.

TDWG: Taxonomic Databases Working Group. http://www.nhm.ac.uk/hosted_sites/tdwg/ Accessed Nov. 2005.

Tsontos, V. and Kiefer, D. (2003): The Gulf of Maine biogeographical information system project: developing a spatial data management frameworkin support of OBIS. *Oceanologica Acta* **25** (2003): 199-206.

Ungerer, J. M. and Goodchild, F. M. (2002): Integrating spatial data analysis and GIS: a new implementation using the Component Object Model (COM), *International Journal of Geographic Information Science*, IJGIS, **16**(1): 41-53.

Wadsworth, R., Balzter, H., Gerard, F. George, C., Comber, L. and Fisher, P. (2005): Quantified Conceptual Overlaps: their use for reconciling inconsistent data sets using Siberian land cover as an example. *Proc. GISPlanet 2005 Conference*, 30 May - 4 June 2005, Estoril, Portugal.

Watson, P. (2005): Databases in Grid Applications: Locality and Distribution. In *Databases: Enterprise, Skills and Innovation*. Jackson, M. Nelson. D. and Stirk, S. (Eds). Lecture Notes in Computer Science 3567, Springer.

Wiederhold, G. (1999): Mediation to Deal with Heterogeneous Data Sources. In *Interoperating Geographic Information Systems*, Vckovski, A. Brassel, K. and Schek H-J. (Eds). Springer Lecture Notes in Computer Science 1580.

Wiederhold, G. (2000): Information Systems that Really Support Decision-making, *Journal of Intelligent Information Systems*, **14**(2) 85-94.

Zaslavsky, I., Baru, C., Bhatia, K., Memon, P., Velikhov, P. and Veyster, V. (2003): Grid-enabled mediation services for geo-spatial information. *Proceedings of NG21-03, Workshop on Next Generation Geospatial Information,* Cambridge, Massachusetts, USA, October 19-21, 2003.

Selectivity Estimation by Batch-Query based Histogram and Parametric Method

Jizhou Luo[1] **Xiaofang Zhou**[2] **Yu Zhang**[2] **Heng Tao Shen**[2] **Jianzhong Li**[1]

[1]Harbin Institute of Technology, China
[2]University of Queensland, Australia
{luojizhou,ljzh}@hit.edu.cn {zxf,zhang,shenht}@itee.uq.edu.au

Abstract

Histograms are used extensively for selectivity estimation and approximate query processing. Workload-aware dynamic histograms can self-tune itself based on query feedback without scanning or sampling the underlaying datasets in a systematic and comprehensive way. Dynamic histograms allocate more buckets not only for the areas with most skewed data distribution but also according to users' interest. However,it takes long time to 'warm-up' (i.e., a large number of queries need to be processed before the histogram can provide a satisfactory coverage and accuracy). Thus, it is less effective to adapt with workload pattern changes. In this paper, we propose a novel online query scheduling algorithm which can significantly reduce the warm-up time for dynamic histograms. A parametric method is proposed to remedy the problem of inaccurate query selectivity estimation for the areas with poor histogram coverage. Experimental results demonstrate a significant effectiveness and accuracy improvement of our approach.

1 Introduction

Most commercial database systems maintain histograms for the purpose of selectivity estimation and approximate query processing (Lim, Wang & Vitter 2003). Typically, the process of building histograms involves sorting and partitioning the data into buckets, based on scanning or sampling the data. Histograms built in such a way are often called *static* histograms in a sense that, once being built, they will remain unchanged even when the underlying data distribution is changed over time. This type of histograms needs to be rebuilt periodically or when the error of selectivity estimation reaches a pre-specified threshold. In order to reduce the cost of building and maintaining histograms for very large datasets, self-tuning histograms have attracted growing attention recently(Aboulnaga & Chaudhuri 1999, Bruno, Chaudhuri & Gravano 2001, Srivastava, Haas, Markl, Megiddo, Kutsch & Tran 2006, Loannidis 2003). Self-tuning histograms are built based on query feedback. It provides an inexpensive way to construct histograms for large datasets with a low up-front overhead. While such histograms can adapt well to the changes of the underlying data distribution, they typically assume that user queries follow a stable workload pattern. That is, newly arrived queries are more likely to visit the areas where previous queries just visited, and the ares visited by more queries in the past are likely to be visited more frequently in the future. This type of 'workload-aware' histograms concentrates on building high quality histograms for these 'hot' areas, at the expense of poor quality or even no coverage for other areas. For many database applications where the query workload patterns do change over time, this type of histograms may not work well. It is important to investigate self-tuning histograms that can adapt to the changes of both data distribution and workload patterns.

In this paper, we consider an online query scheduling method to speed up accuracy convergence process such that a histogram can reach a satisfactory average selectivity estimation accuracy more rapidly when workload pattern changes. This is achieved by giving a higher priority to the queries in the areas which are 'hot' (according to recently arrived and executed queries) and skewed (according to recent query feedback). On the other hand, there are always some queries which do not follow the current workload pattern. Estimation accuracy for the queries to these areas can be poor for self-turning histograms, adversely affecting the performance of the entire system, often measured by the average estimation inaccuracy. This problem is more serious at the beginning of workload pattern changes. Instead of using uniform distribution assumption, a novel parametric selectivity estimation method is proposed in this paper for queries in the areas with poor or no histogram coverage. The information required for applying parametric estimation is simple and online maintainable. Our work in this paper is based on $STHoles$ histograms(Bruno et al. 2001), but provides a significant improvement for convergence speed and overall estimation accuracy for queries with changing workload patterns.

The rest of the paper is organized as follows. Section 2 introduces notations, definitions and the basic idea of $STHoles$. An online query scheduling method is introduced in Section 3, and selectivity estimation for range queries using the novel parametric method in Section 4. A performance evaluation is reported in Section 5, with related work reviewed in Section 6.

Supported by ARC under Grant No.DP0663272; the National Natural Science Foundation of China under Grant No.60273082.

We conclude this paper in Section 7.

2 Preliminaries

As we consider only range queries in this paper, we can assume that there is only one relation of n attributes. We also assume that an attribute contains only countable value coded as integers. Let $R = (X_1, \dots X_n)$ be the schema of a relation, whose domain is $\mathcal{D} = (\mathcal{D}_1 \times \dots \mathcal{D}_n)$. Let r be an instance of R, and $\mathcal{V}_i \subseteq \mathcal{D}_i$ be the set of X_i values in r.

A hyperrectangle B in $\mathcal{D}$ is defined as $B = \{(x_1, \dots x_n) \in \mathcal{D} | v_l^i < x_i < v_u^i, i = 1..n\}$. B can be represented using two n-dimensional points $v_l = (v_l^1, \dots v_l^n)$ and $v_u = (v_u^1, \dots v_u^n)$, where $v_l^i = \min_{v \in B} v^i$ and $v_u^i = \max_{v \in B} v^i$. Thus, B can be denoted as $B =< v_l, v_u >$. The volume of $B(v_l, v_u)$ is $vol(B(v_l, v_u)) = \prod_{i=1..n}(v_u^i - v_l^i)$. The topological relationships we are interested between two hyperrectangles B and B' are disjoint ($B \cap B' = \emptyset$), overlapping ($B \cap B' \neq \emptyset$) and containment (B is nested in B', denoted as $B \subseteq B'$). A containment tree can be constructed for a set of hyperrectangles B_i, $1 = 1..w$, such that 1) the root node is the entire space of interest, 2) a child node is nested inside a parent node, and 3) no sibling nodes are nested among each other.

A histogram $\mathcal{H}$ for relation r is a collection of hyperrectangle and frequency pairs. That is, $\mathcal{H} = \{(B_i, f_i) | i = 1..m\}$, where f_i is the expected number of r tuples inside B_i (the points on a hyperplane of a hyperrectangle is assigned to only one hyperrectangle following the open-close hyperplane convention). The estimated frequency of r tuples in B_i is also denoted as $\mathcal{H}(B_i)$, and B_i is referred to as a bucket. For different types of histograms, all buckets in a histogram of r may form a partition, a cover, or a partial cover of $\mathcal{D}$ or $\mathcal{V}$.

For a hyperrectangle $B_q =< v, v' >$, a range query $q(B_q)$ against r retrieves all $\{x \in r | x \in B(v, v')\}$. B_q is called the query region of q. The collection of query result is the query feedback of q, and the number of tuples in the result collection is denoted as $act(r, q)$. We use $est(\mathcal{H}, q)$ to denote the estimated number of tuples returned from executing q against r according to histogram $\mathcal{H}$. If data distribution inside any bucket is assumed to be uniform, this estimation can be done in a straightforward way:

$$est(\mathcal{H}, q) = \sum_{B_i \in \mathcal{H}} \frac{vol(B_i \cap B_q)}{vol(B_i)} \mathcal{H}(B_i)$$

Next we introduce *STHoles* histograms(Bruno et al. 2001), which is a representative method to build and maintain histograms using query feedback. To simplify our discussion and for better visualization, we set $n = 2$ hereafter. However, our discussions are applicable to higher dimensions.

There are many different ways to construct the initial histogram. Assuming that initially the histogram is empty. Once a query is executed, the query feedback becomes available and can be used to identify any skewed data distribution by checking the precise number of tuples returned for the query in the areas overlapping with other existing buckets. Once a significant variation of data distribution is found for any part of a bucket, a 'hole' is created to give a more accurate expected frequency for that part using a new bucket, and the frequency of the existing bucket affected will be adjusted with the new information. Some sophisticated algorithms are proposed in (Bruno et al. 2001) to determine bucket shapes and perform bucket mergers. This approach does not require a systematic/comprehensive scan of the underlying dataset, but allocates buckets according to user query feedback. This approach is called 'workload-aware', as the refinement of a histogram is directed by user's collective interest (as per queries). It can also adapt to the changes of data distribution, as such changes are eventually reflected in query feedback that leads to changes of the histogram.

Assume bucket B in a *STHoles* histogram contain a number of holes (i.e., a set of buckets $B_1, \dots B_k$, where these child buckets do not overlap by definition). The net volume and net frequency of B can be defined as

$$vol_{net}(B) = vol(B) - \sum_{i=1}^{k} vol(B_i)$$

and

$$f_{net}(B) = f(B) - \sum_{i=1}^{k} f(B_i)$$

The density of a bucket B is defined as

$$d(B) = \frac{f_{net}(B)}{vol_{net}(B)}$$

Define a workload W as a sequence of $|W|$ queries. A common metric to measure the performance of a histogram is the average absolute error over W:

$$E(r, \mathcal{H}, W) = \frac{1}{|W|} \sum_{q \in W} |est(\mathcal{H}, q) - act(r, q)|$$

In *STHoles*, a histogram may not fully cover the entire data space, thus there may not exist any buckets for certain areas. To make robust comparison across different datasets, a normalized absolute error metric is used in (Bruno et al. 2001) by introducing $est_{uni}(r, q)$ which is the estimated result size by assuming uniform data distribution for the areas where no histograms are available. That is,

$$E'(r, \mathcal{H}, W) = \frac{E(r, \mathcal{H}, W)}{E_{uni}(r, W)}$$

where

$$E_{uni}(r, W) = \frac{1}{|W|} \sum_{q \in W} |est_{uni}(r, q) - act(r, q)|$$

The *STHoles* approach makes an explicit assumption that workload pattern is stable. Therefore, histogram performance is evaluated by using a training workload to create a histogram, and then to use a validation histogram to compute the error metric, where the validation workload has the same distribution pattern as to that of the training workload (in terms of both coverage and query foci areas). For many database applications, queries do follow some workload patterns within a period, but the patterns may change over time. Not being able to divide queries into training and validation queries, the order

Notation	Description
id	query ID
B	query area
hot, *gap*	hotness scale and gap value
est, *rst*	the estimated and actual number of tuples
children	pointers to children nodes
parent	pointer to the parent node
next	pointer to the next execution candidate node

Table 1: A table of attributes.

of executing queries becomes important as the feedback from a query can impact on other queries. In other words, this problem is an online query scheduling problem with a continuous stream of range queries targeting variable query regions. We approach this problem from two aspects: 1) a fast-convergence online scheduling algorithm to give higher priority for the queries in the areas where more queries need to be observed to improve the overall quality of the histogram; and 2) a novel estimation method using concise and online maintainable information to estimate the selectivity for queries in an area with no or poor histogram coverage.

3 Online Query Scheduling

In this section, we propose a scheduling algorithm based on both the recent past query history and the queries in the buffer (i.e., arrived but yet to be processed). For each query q, we define two measures first: hotness scale and the gap value.

The *hotness scale* of q in relation to a set Q of queries is defined as $hot(q, Q) = \sum_{i=1}^{k} S_i \times n_i$, where $\{S_i | i = 1..k\}$ is a set of disjoint rectangles inside $q.B$ generated by overlaying all $q'.B$ together, $q' \in Q$, and n_i is the number of queries in Q whose query regions cover S_i (because each S_i is obtained by a complete overlay of all query regions in Q, the case for a query in Q to partially cover any S_i does not exist). Intuitively, a query is hotter if it is in a larger area that many other queries also are interested in. Obviously, the hotness scale changes when new queries arrive.

For two regions B_1 and B_2, $B_1 \subseteq B_2$, their density difference can be measured using a *gap* value, which is defined as

$$gap(B_1, B_2) = \frac{d(B_1)}{d(B_2)} = \frac{f_{net}(B_2)}{vol_{net}(B_2)} \times \frac{vol_{net}(B_1)}{f_{net}(B_1)}$$

Clearly, a *gap* value falls into range $(0, +\infty)$, where $gap = 1$ means that there is no difference between the density of B_1 and B_2; $0 < gap < 1$ means that the inner bucket B_1 has a lower density (and a smaller value of *gap* indicates a larger density difference); and $gap > 1$ means that the inner bucket B_1 has a higher density (and a larger value of *gap* indicates a larger density difference).

3.1 Query Tree Construction

Assume that continuously arriving query stream is stored in a buffer. The buffer size is limited but large enough so we do not need to consider the problem of query buffer overflow. Further, we assume that a query has a deadline for it to be executed, but in general queries can be deferred for execution within the specified time limit in favor of a better overall system performance. This is a common assumption for batch query processing. Once a new query arrives, the scheduling algorithm must compute its hotness and gap values, and update the changes of these values in relation to other yet-to-execute queries in the buffer. The queries in the buffer need to be organized in such a way that updates when new queries arrive and the algorithm to select queries for execution can be efficiently supported. We propose to organize the current queries according to their natural containment relationship. Table 1 lists the attributes recorded for each node in the tree.

```
Algorithm 1: InsertQuery
input: $q$: arriving query, $QT$: root of a query tree
output: $QT$: query tree updated with $q$
01. $t \leftarrow QT$; $q.hot \leftarrow vol(q)$;
02. while $t \neq null$ {
03.     if $q.B \subseteq t.B$ {
04.         $t.hot \leftarrow t.hot + q.hot$;
05.         $t \leftarrow$ the first unvisited child of $t$;
06.     } else { // $q$ will be inserted here
07.         for each child node $q' \in t.children$ {
08.             if $q'.B \subseteq q.B$ // replace a child
09.                 $q.hot \leftarrow q.hot + q'.hot$;
10.                 let $q'$ be a child of $q$, and $q$ a child of $t$;
11.             else { // exchange hotness with a sibling
12.                 $q.hot \leftarrow q.hot + vol(q.B \cap q'.B)$;
13.                 $q'.hot \leftarrow q'.hot + vol(q.B \cap q'.B)$;
14.             }
15.         }
16.         $t \leftarrow$ the next node to visit according to DFS;
17.     }
18. }
19. return $QT$;
```

The root node of the tree is initialized to cover the entire space. The algorithm to insert a new query q into the tree is simple: a depth-first search (DFS) is performed on the current query tree to find all queries whose query regions fully contain $q.B$, and the hotness value for all the nodes traversed during the search will be updated. Algorithm 1 is the sketch of this algorithm. If $q.B$ is nested inside $t.B$, t is a node in the tree, $t.hot$ will be increased by $q.hot$ and the search will continue for all of children nodes of t (lines 3-5). If $q.B$ is not nested inside a node t, then q will be inserted as a child of t, and the search of this branch of the tree is completed and moved to another part of the query tree according to DFS until no more nodes to search (lines 7-13). When q is to be inserted as a child of t, there are two different cases: 1) a child node q' of t are nested inside $q.B$. In this case, q' will become a child of q and pass its hotness value to q before q is inserted as a child of t (lines 8-10); 2) a child node q' of t is not nested in $q.B$, in this case, the volume of overlapping areas of q and $q'.B$ will be added to the hotness value of both q and q' (lines 11-13). Note that a query may be nested inside more than one sibling nodes; in this case, the search will be followed along all these subtrees. That is, a query can be inserted into multiple places in the tree. It is not difficult to see that the above algorithm maintain the hotness scale values for all nodes in the tree after a new query is inserted. The reason for a query q to be inserted into multiple places in the tree is that all nodes that contain $q.B$ must be prossed to get its share of increased hotness scale resulting from q.

3.2 Query Scheduling

In order to minimize the average error for a sequence of queries, a query is selected to execute such that its feedback can benefit the process of tuning the histograms most for the remaining and future queries. Before discussing the scheduling algorithm, we explain two observations.

observation 1. *In order to make a histogram more adaptive to workload pattern changes, one should test the queries on hot spots first. If two spots' hotness degrees are identical, one should test the queries which are as close as possible to those areas where data is more disproportionately distributed. This strategy can speed up accuracy convergence of the histogram.*

observation 2. *If the density of bucket B_1 is much smaller than that of bucket B_2, then the data distribution in bucket B_2 is more interesting than that in bucket B_1. So, it is rationale to choose a child query of B_2 to execute in the next step in order to speed up the convergence of the histogram. Otherwise, the data distribution in bucket B_1 is more interesting; thus it is better to choose a sibling query of B_2 to execute in the next step in order to speed up the convergence.*

Based on these two observations and using the query tree data structure, a simple scheduling algorithm can be used to select the top k queries from the query tree according to their hotness values. These k queries will be further selected to choose the one with the smallest *gap* value. Figure 1 illustrate the rational behind this heuristics of choosing the smallest *gap* value. This can be understood by considering the following three cases:

Case 1: if $g_1 < g_2 < 1$, this means that $q_1.parent$ is more dense than $q_1.parent.parent$ comparing with its counterpart of $q_2.parent$. Based on the above heuristics rule, q_1 is selected to execute.

Case 2: if $g_2 > g_1 > 1$, this means that $q_2.parent$ is less dense than $q_2.parent.parent$ comparing with its counterpart of $q_2.parent$. So q_1 has a higher priority.

Case 3: if $g_2 > 1 > g_1$, it is straightforward to tell $q_1.parent$ is much more dense than $q_1.parent.parent$ comparing with its counterpart of $q_2.parent$. q_1 will be executed first.

The above three cases all suggest to select a query with the smallest *gap* value. Note that a gap value is only known after some queries are executed. When a new *query* is inserted, its gap value is inherited from its parent, and the gap value for the root is defined as 1. Once a query q is executed, $q.rst$ is set to the number of tuples in the query result, and for each children nodes q' of q, $q'.gap = d(q.parent.B)/d(q.B)$. In other words, the gap value of a node is the ratio of the densities of its parent and grand-parent nodes. This is important because the gap value of a node is used to prioritize query execution (i.e., its own density is unknown), so its gap value indicates if its parent node is in an area with skewed data distribution.

Once q is selected, it is not removed from the query tree immediately; rather, $q.est$ is set to the current estimated selectivity of q (when q is inserted into the query tree, $q.est = q.rst = 0$). The reason of doing so is that executed queries are still useful for identifying the areas with large hotness and *gap* values. These executed queries, as explained next, will only be deleted when the query buffer is about to run out for new arrivals. $q.rst$ is set to the actual number of tuples in the result of q after its execution. It is mentioned before that a query may be inserted into more than one place in the tree. Multiple instances of the same query are treated as different queries in the scheduling algorithm, except that they will be executed only once and are deleted together.

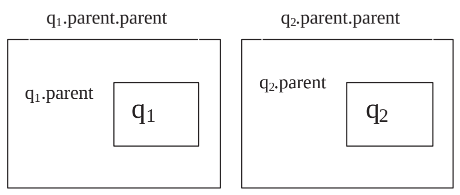


Figure 1: Scheduling order with *gap* value

3.3 Discussions

Building Histograms The scheduling algorithm proposed in this paper is an orthogonal improvement to the *STHoles* idea. With the optimized order of query execution, the basic ideas and algorithms in *STHoles* are still applicable here. A noticeable difference between our method and *STHoles* is how the entire data space is approximated in the histogram. In *STHoles*, the histogram is initialized to be empty; in our method, the initial histogram is a single bucket that covers that entire space of interest with uniform distribution. A uniform distribution about a very large area can lead to a considerably worse overall estimation accuracy error; thus, *STHoles* avoids this problem by making an assumption that a consistent workload pattern means that, after a training period, no new queries will retrieve data from previously unvisited region. For the query stream environment considered in this paper, this assumption can no longer be made. Further, consider the case where a query to a large region is followed by a number of queries targeting some small areas. This is equivalent to the problem of how to deal with the entire space. We address this problem by a novel parametric selectivity estimation method we propose in this paper. Another difference comparing to *STHoles* is that a different merger algorithm is used in our approach, as will be discussed later.

Query Deletion Once a query is executed, it needs to be removed from the query tree. Otherwise the query buffer will reach its limit and new queries can not be stored for optimization. However, if a query is deleted from the query tree immediately after execution (and its children nodes will be 'promoted' one level up), then the query tree tends to be a flat structure and becomes ineffective to trace the users' interests. This may also cause the histogram oversensitive to users' interest change (such as a small variation of workload pattern), as buckets are moved around too quickly and affected too much by the new arrivals without the balancing factor of recent past queries. To overcome this problem, we store the executed queries in another First-In-First-Out queue of a pre-determined size. After query q is executed, q remains in the query tree but is also pushed into the queue. The oldest query will be removed from the queue when the queue is full, and only at this time, q is deleted from the query tree.

Starvation Using the scheduling algorithm descried above, some queries may be forced to wait for a long time or even never get executed (because they are from a 'cold' area for example). While such waiting has no negative impact on the error metric we try to minimize, they can cause problems from user's side. It also has a side effect that the available size of the query buffer is reduced by the queries that cannot be cleared for too long. This problem can be solved by recording, for each query q, a pair of time values $q.t_1$ and $q.t_2$, $q.t_1 \leq q.t_2$, where t_1 is its arrival time and t_2 is the maximum tolerate time which can be a default time set by the system (i.e., 3 minutes after arrival) or a time set by the user. These two values can be used to let the queries that have been waiting for too long or approaching the deadline 'jump the queue'. Using feedback-based histograms, however, means that these time-activated queries may not in the areas with good histogram coverage, thus may have a negative impact on the error metric measure. This problem is addressed in the next section using the parametric selectivity estimation method.

4 Selectivity Estimation

Traditionally, the data in a bucket is assumed to be uniform. Selectivity estimation, therefore, can be done easily as mentioned before. However, the whole idea of $STHoles$ is built on the observation that data inside a bucket may not be uniform (thus, 'holes' are opened inside a bucket once a significant variation of data distribution is found in a bucket). An interesting case is how to estimate the selectivity for a region inside a large bucket which consists of a number of hole buckets. We argue that if the query region is close to those hole buckets in a large bucket, a better estimation can be obtained by considering the frequency of those hole buckets. The situation of estimating selectivity for a query in an area which has no histogram is in the same category. We propose a parametric method to estimate the selectivity for areas close to the child buckets.

```
Algorithm 2: EstInBucket
input: $B$: a bucket of $\mathcal{H}$; $q$: a query covered by $r(B)$
//$r(B)$ is the rectangle bounded by the boundary of $B$
output: $S$: selectivity estimation of $q$
1. $E \leftarrow 0$;
2. For each child bucket $B'$ of $B$;
3.     $q_B \leftarrow q \cap r(B)$;
4.     if $q_B \neq \emptyset$ then $E \leftarrow E + EstInBucket(q_B, B')$;
5.     $q \leftarrow q \setminus q_B$;
6. if $cr(B) < threshold$ then $E \leftarrow E + \int\int_q \varphi(x,y)\,\mathrm{d}x\mathrm{d}y$;
7. else $E \leftarrow E + \frac{vol(q)}{vol_{net}(B)} * f(B)$;
8. return $E$;
```

Given a user query q, its selectivity is estimated by a recursive algorithm $EstInBucket(q, B)$, as shown in Algorithm 2. $EstInBucket(q, B)$ recursively estimates the selectivity of the intersection area between q and each child bucket $B' \in child(B)$, where $child(B)$ consists of all the child buckets of B (lines 2-5). If the cover ratio C_r is less than a pre-specified threshold, a density function $\varphi(x, y)$ is proposed to capture the data distribution around each child bucket (line 6). In other words, we use child buckets' information to estimate selectivity in bucket B. However, if the cover ratio C_r is greater than the threshold, we assume all the tuples in B are uniformly distributed and estimate the selectivity as $f_{net}(B)$ times the ratio of the volume of q falling in B over $vol_{net}(B)$(line 7). We defer the discussion on density function $\varphi(x, y)$ to the next subsection.

4.1 Density Function

In this subsection, we deduce the density function $\varphi(x, y)$ used in Algorithm 2. Our idea is based on the following observation.

observation 3. *The difference between the data distribution in the area around a bucket and that in the bucket is unlikely to change dramatically.*

This observation is based on the fact that bucket boundaries are determined by user queries, rather than some optimal separation based on data distribution. This observation implies that information about data distribution in each bucket can be used to approximate the data distribution of the area nearby. Without a uniform distribution assumption, an appropriate density function needs to be identified to describe data distribution.

Define a *barycenter* (i.e., center of mass) of a bucket as the point in the bucket where the frequency of the bucket can be viewed as concentrated on that point. If data are uniformly distributed within the area bounded by the boundary of bucket, this barycenter should be the center of the bucket. However, when data are not uniformly distributed, i.e., the area has been separated by hole buckets, it is nontrivial to compute the barycenter, as the center moves with respect to each insertion, deletion and frequency update of a child bucket. To determine the barycenter $(\bar{x}_{B_i}, \bar{y}_{B_i})$ for bucketB_i in $\mathcal{H}$, we use the following propositions.

Proposition 1. *Suppose bucket A is the only child bucket of bucket B, $d(A)$ and $d(B)$ are their densities under uniformly distribution assumption respectively, then the barycenter of bucket B can be calculated as follows:*

$$\bar{x}_B = \frac{1}{2f(B)} \sum_{X \in \{A,B\}} (d(X) - d(P(X)))F_1(X)$$

$$\bar{y}_B = \frac{1}{2f(B)} \sum_{X \in \{A,B\}} (d(X) - d(P(X)))F_2(X)$$

where $P(X)$ is the parent of X, $d(P(B)) = 0$ and
$F_1(X(v_l, v_u)) = (v_u^2 - v_l^2)([v_u^1]^2 - [v_l^1]^2)$,
$F_2(X(v_l, v_u)) = (v_u^1 - v_l^1)([v_u^2]^2 - [v_l^2]^2)$.

Proof. According to the formula calculating the barycenter of an area in the plane with density function $\rho(x, y)$, we have

$$\begin{aligned}\bar{x}_B &= \frac{1}{f(B)} \int\int_B x\rho(x,y)\,\mathrm{d}x\mathrm{d}y \\ &= \frac{1}{f(B)}(\int\int_B xd(B)\,\mathrm{d}x\mathrm{d}y + \int\int_A x(d(A) - d(B))\,\mathrm{d}x\mathrm{d}y)\end{aligned}$$

The result of the expression is what we expected. The calculation of $\bar{y}_B$ is similar. □

Proposition 2. *Let B as a bucket in $\mathcal{H}$, A as an arbitrary hole bucket of B, $d(A)$ as density of A, and $p(A)$ as the parent of bucket A. By setting $d(P(B)) = 0$, the barycenter of B can be computed as follows under the uniformly distribution assumption:*

$$\bar{x}_B = \frac{1}{2f(B)} \sum_A (d(A) - d(P(A)))F_1(A)$$

$$\bar{y}_B = \frac{1}{2f(B)} \sum_A (d(A) - d(P(A)))F_2(A).$$

Proof. By proposition 1, we can use the induction on the number of levels in the bucket-tree rooted at B to verify this proposition easily. □

Once the barycenters of buckets are calculated, we can use a parametric function to model the data distribution around these buckets as follows. Let B be a bucket of histogram $\mathcal{H}$, point $(\bar{x}_B, \bar{y}_B)$ be its barycenter, ρ_B be the density of point $(\bar{x}_B, \bar{y}_B)$ in B, and u_B be the shortest distance from $(\bar{x}_B, \bar{y}_B)$ to the boundary of B. According to Observation 3, barycenter's density is supposed to be maximal and the density around this point decreases continuously in the manner characterized by a parametric function $g_\theta(t)$, where θ is the parameters to be determined and t represents the distance from the query to $(\bar{x}_B, \bar{y}_B)$. Several issues need to be considered. Firstly, the valid radius of parametric function $g_\theta(t)$ must be determined (within and only within this radius the barycenter of this bucket can be used for estimation). Secondly, the number of parameters in function $g_\theta(t)$ must be determined. And thirdly, the type of function should be determined. we discuss each of them below.

Let us consider the radius of the parametric function fist. Since the histogram is built from query feedback, the frequencies in each bucket and all its child buckets are known. That is, the data density of bucket B can be computed precisely and is not affected by any other buckets. Thus, the valid radius of each parametric function $g_\theta(t)$ cannot exceed the distance dis_B from $(\bar{x}_B, \bar{y}_B)$ to its nearest barycenter. Hence, we get have the following condition (1).

$$g_\theta(dis_B) = 0 \tag{1}$$

Each parametric function can only be used to estimate the data distribution around its owner bucket. When a query is far away from all buckets, there is no buckets to be used for selectivity estimation. In this case, we follow the assumption of uniform density. The idea above is based on the principle "more information we known, better accuracy we can get."

Next, let us determine the number of parameters in function $g_\theta(t)$. Except condition (1), we only know the average data density d_B in bucket B. The exact density at the barycenter is not available. Generally, the influence form one barycenter reduces with the distance. Denote the density of point p, which is the nearest point on the boundary of B to its barycenter $(\bar{x}_B, \bar{y}_B)$, as $d(B)$. We get condition (2)

$$g_\theta(u_B) = d(B) \tag{2}$$

Thus, we have only two conditions to determine a parametric function $g_\theta(t)$ and the number of parameters needed for this function is at most 2.

It is more complex to select the type of the parametric function. A polynomial function with only two parameters such as $at + b$, $at^2 + bt$, $at^2 + b$, can be used. When the data distribution is known, a more accurate function can be applied. For example, we can choose ae^{-bt} as the function if we know that data follows Gaussian distribution approximately. After the function type is determined, function parameters can be determined by using condition (1) and condition (2). Denote the obtained function for bucket B as $g_B(t), t \in [u_B, dis_B]$. This function can be easily extended to be defined for the whole space as below.

$$G_B(t) = \begin{cases} d(B) & t < u_B \\ g_B(t) & u_B < t \leq dis_B \\ 0 & otherwise \end{cases} \tag{3}$$

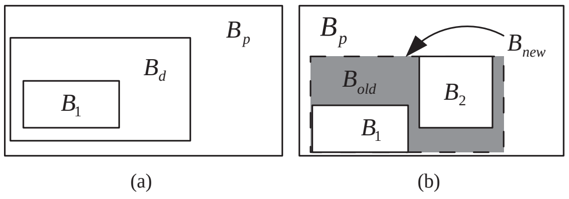


Figure 2: Buckets Update

If a query point p is covered by more than one bucket, its density estimation is a combination from the influences from all these buckets. Thus, this density function is

$$\varphi(x, y) = \sum_{B \in child(B)} G_B(dis((x, y), (\bar{x}_B, \bar{y}_B))).$$

4.2 Dynamic Maintenance of Histogram

With more buckets added, a histogram will reach the size of available memory allocated. In order to be adaptive to the changes of underlying data distribution and user's interest, some existing buckets need to be merged, by considering two different cases: parent-child buckets merge (where a parent bucket merges with one of its child bucket) and sibling-sibling merge (where two buckets under the same parent bucket merge).

For the first case, we scan all the buckets' gap value and find out the minimum gap value. Denote the bucket with minimum gap value G_{min} as B_{min}. Data density in B_{min} is the closest to that of its parent bucket. This means least penalty will be caused if B_{min} is merged with its parent bucket. Figure 2 (a) illustrates an example. After B_1 is created in B_d, the actual density of B_d is similar to B_p. Thus, B_d needs to be merged with B_p.

For the other case, to compare two sibling buckets (B_1 and B_2)on the same level under parent bucket B_p, we find the smallest hyperrectangle B_{new} that encloses both B_1 and B_2 (Bruno et al. 2001). Figure 2 (b) illustrates an example. In general, B_{new} contains old part B_{old} (the shadowed region in the example) from B_p. As we are able to calculate the area of B_{old}, we can easily get the frequency of B_{old} approximately by using

$$f(B_{old}) = f(B_p)\frac{V(B_{old})}{V(B_p)}$$

In the mean while, B_p becomes $B_{p'}$, volume and frequency are updated respectively. Having $B_{p'}$ and B_{new}'s volume and frequency, we get a new gap value G_{new} for B_{new}.There are two drawbacks in sibling-sibling merge. On one side, the identification of B_{new} is very expensive because we must test if any two sibling buckets forms as the configuration in figure 2 (b) without any other sibling bucket falling into the shadowed region. On the other side, $f(B_{old})$ is an estimation number, which may lead to an misleading.

Based on the above discussion, we give parent-child merge a higher priority than sibling merge. That is, we determine a parent-child merge candidate bucket B_{min} globally and then to determine wether there is an better sibling merge candidate bucket B_{new} such that $G_{new} < G_{min}$. If so, we will merge some two sibling buckets into B_{new}; otherwise, we merge B_{min} with its parent.

After two buckets are merged, the barycenter of each bucket needs to be updated, as well as the gap values of the buckets surrounding the merged bucket. Suppose the barycenter of a bucket B is $(\bar{x_B}, \bar{y_B})$, and a new query causes updating the histogram by inserting a new bucket B_I somewhere in bucket B and deleting a bucket B_D in bucket B_1. It is possible that $B = B_1$ (see Figure 2). Now we need calculate the barycenter of the updated bucket B and B_1. Since if $B = B_1$, we can finish this update by first calculating the updated barycenter for a insertion in B following by calculating the updated barycenter for a deletion in B. So, we only consider the update caused by a single operation, insertion or deletion.

Proposition 3. *Let B is a bucket of histogram $\mathcal{H}$, $d(B)$ is the density of B and $(\bar{x_B}, \bar{y_B})$ is the barycenter of B. If a bucket B_I with density $d(B_I)$ is insert into B and causes the density of B changed as $d'(B)$, then the barycenter of B can be updated as follows.*

$$\begin{aligned}\bar{x} = \bar{x} + (d'(B) - d(B))[\sum_{X \in child(B)} F_1(X) - F_1(B)] \\ + (d(B_I) - d(P(B)))F_1(B_I)\end{aligned}$$

$$\begin{aligned}\bar{y} = \bar{y} + (d'(B) - d(B))[\sum_{X \in child(B)} F_2(X) - F_2(B)] \\ + (d(B_I) - d(P(B)))F_2(B_I)\end{aligned}$$

Proof. Calculate the barycenter of B before and after insertion with proposition 2, and subtract the former from the latter them. □

Proposition 4. *Let B is a bucket of histogram $\mathcal{H}$, $d(B)$ is the density of B and $(\bar{x_B}, \bar{y_B})$ is the barycenter of B. If a child bucket B_D of B with density $d(B_D)$ is merged with bucket B and causes the density of B changed as $d'(B)$, then the barycenter of B can be updated as follows.*

$$\begin{aligned}\bar{x} = \bar{x} + (d'(B) - d(B))[F_1(B) \\ - \sum_{X \in child(B) \setminus \{B_D\}} F_1(X)] - (d(B_D) - d'(B))F_1(B_D)\end{aligned}$$

$$\begin{aligned}\bar{y} = \bar{y} + (d'(B) - d(B))[F_2(B) \\ - \sum_{X \in child(B) \setminus \{B_D\}} F_2(X)] - (d(B_D) - d'(B))F_2(B_D)\end{aligned}$$

The proof of this proposition is similar as above.

5 Experimental Evaluation

In this section, we compare the performance of our proposed methods with *STHoles* in the context of changing data distribution and changing workload patterns. In addition to comparing their estimation accuracy, two other impotent factors are also considered: convergence speed and memory sizes. We call our technique *BQHist* hereafter.

In our experiment, a real-world dataset is used: the State Regional Ecosystem (SRE) dataset with 398, 464 polygons representing different species' coverage. Synthetic datasets of 2- and 3-dimensions are also used: one with Gauss distribution and one with logistic distribution. The Gauss distribution data consists of a number of Gaussian bells with a predetermined standard deviation. The total number of tuples is 1,600,000. A Zipf distribution regulates the number of tuples in each Gaussian bell. The logistic dataset is generated in a similar way.

5.1 Estimation Accuracy

For different datasets, we examine estimation accuracy of our algorithm using parametric or uniform estimation at different cover ratios, as detailed in different types of workload. The capacity of the query buffer is set to 200 throughout the experiment (i.e., at any time no more than 200 queries can be cached in the buffer for scheduling).

Workload 1: This workload pattern is designed to compare estimation accuracy. It consists of 1050 queries, with 50, 100, 300 and 600 queries to retrieve 2%, 1%, 0.4% and 0.3% of the whole dataset respectively. These queries are distributed uniformly in the region, and their combined query areas cover 50% of the total area.

Result: Figure 3 shows the accuracy after 250 queries are executed on SRE dataset. *BQHist* with uniform estimation performs at least 30% more accurate than *STHoles*. This figure also shows that *BQHist* with uniform estimation performs better than *BQHist* with parametric estimation, which is consistent with what we discussed before, as this workload distribution is not skewed much.

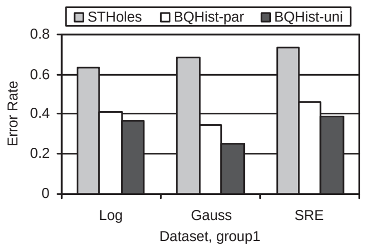


Figure 3: Accuracy of different estimation methods

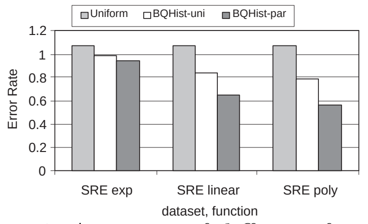


Figure 4: Accuracy of different functions

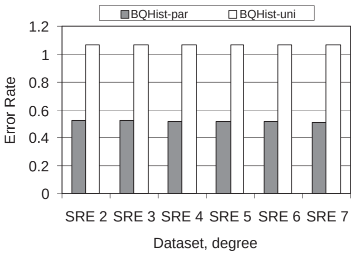


Figure 5: Accuracy of different polynomial powers

Workload 2: This workload is designed to examine the cases where query areas are concentrated to a

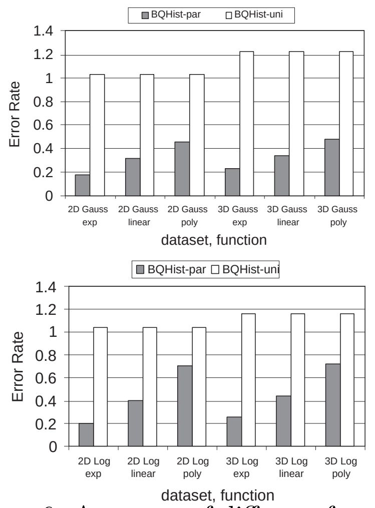


Figure 6: Accuracy of different functions

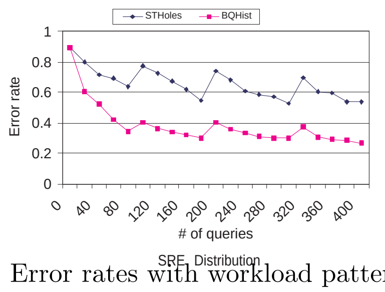


Figure 7: Error rates with workload pattern change

small area. It still uses 1050 queries but the total area covered by these queries occupy just 10% of the total area. Further, these queries are nested to 10 levels, with 10 top-level queries retrieving 3% of tuples from the dataset, and each query is followed by two child queries which retrieve 10% of the data in its parent.

Result: Figure 4 shows the normalized absolute errors of using different parametric functions and uniformity based estimation on SRE dataset. This figures clearly shows that the *BQHist* can achieve a much higher accuracy comparing to the uniformity-based estimation, for all the three sample functions used; and for the case of polynomial function, a 50% improvement has been achieved. The reason here is that SRE is not a highly skewed dataset, which means that the frequencies of the areas around a buckets do not change dramatically from that in the bucket. A polynomial function reflects this behavior well, as its value does not drop quickly in a short distance away from the starting point. A uniformity-based estimation, on the other side, results in an underestimation around a bucket's child buckets and an overestimation at the area far from the child buckets.

Figure 5 shows that the differences among polynomial's powers (on SRE dataset) do not have a significant impact on estimation accuracy. This insensitivity is a nice property, as these powers are typically not easy to determine. Figure 6 shows the normalized absolute errors of using different parametric functions on the four synthetic datasets. Since the distribution of the synthetic data is known in prior, one can predict which function has better performance, as the more similar they are, the better accuracy the result is. Because exponential function is the same as Gauss distribution, it leads to the best accuracy. Since logistics distribution is the most similar among three functions, it delivers good results as well. However, the polynomial function is constantly robust across all the tests with error rate less than 50% of uniform distribution error rate. Furthermore, we notice that with the number of dimension increasing to there, the error rates only change slightly.

5.2 Changing Workload Patterns

Workload 3: This workload is designed to test the impact of changing workload patterns. Four clusters of queries are used, where the areas for the queries in one cluster do not overlap with the queries from another cluster. For each cluster, 400 queries with a uniform selectivity of 0.25% are randomly distributed in the area of the cluster. During the tests, only 100 queries are used from each cluster (i.e., after these 100 queries in a cluster are processed, the queries from another cluster are used straightaway). The normalized error is calculated with the previously tested queries.

Result: Figure 7 shows the results on SRE (experimental results on other datasets, which confirm a similar trend, are omitted due to space limit). It is clear that *BQHist* results in a lower error rate than *STHoles* consistently, especially when the workload pattern moves from one cluster to another.

5.3 Convergence Speed and Cache Capacity

Workload 4: This workload is designed to test the convergence speed and the impact of cache capacity (as it can be argued that the additional memory required by *BQHist* can be used by *STHoles* to improve its performance by using more buckets). 50 queries from Workload 1 and other two group of queries are used as training queries, and 1050 validation queries following the same distribution as the training queries are used to measure the normalized absolute error. The initial queries of both sequential and batched order are the same. We use three groups of 1050 queries with 25% cover ratio distributed in the region following the Gaussian, Zipfian (with the z parameter, which indicates distribution skewness, set to 2) and unform distribution. The selectivity ranges from 1% (50 queries), 0.25% (200 queries), to 0.05% (800 queries).

Result: Figure 8 shows the results on different datasets, methods and cache capacities. In general, the error rates drop sharply in around first 300 queries when system is able to cache all the queries(*BQHist-full*). When the *QT* tree is flat (i.e., too many query nodes are on top level), *BQHist* becomes less effective (shown in the second part of Figure 8) as the gap value's function is not fully utilized. However, the result is still about 20% better than *STHoles*. The performance of *BQHist*-300 is between *STHoles* and *BQHist-full*, which indicates that a larger cache leads to a better performance. Figure 9 demonstrates that *BQHist* is at least 10% more accurate than *STHoles* with randomly distributed queries.

Workload 5: A scheduling algorithm is proposed in subsection 3.2 to select the top k queries from the query tree according to their hotness value. This workload is designed to test the impact of k values on our the performance of our method. In the first experiment, 400 queries in a single cluster from Workload 3 are used. In the second experiment, queries from all Workload 3 clusters are involved.

Result: Recall that these top k queries are selected according to the hotness value in the query

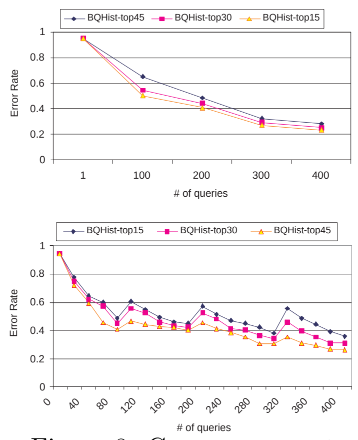


Figure 8: Convergence rates

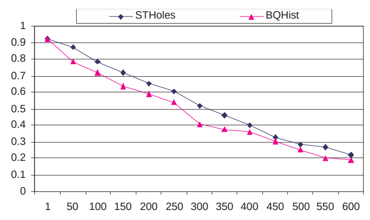


Figure 9: Convergence for random queries

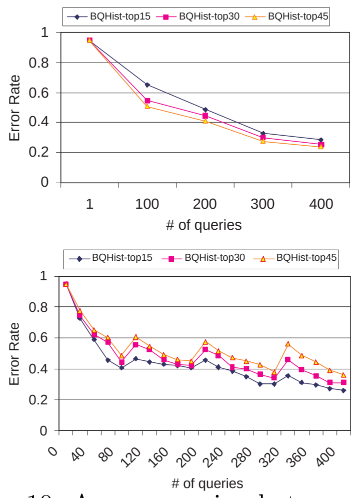


Figure 10: Accuracy using hotness and gap

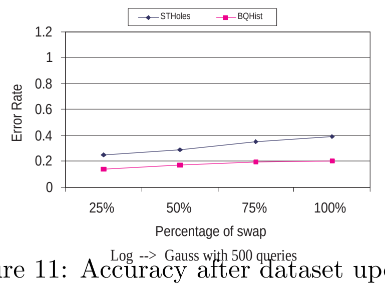


Figure 11: Accuracy after dataset updates

tree. Thus, a larger k value deemphasize the importance of the hotness value, giving more room to select query execution according to their gap values. The first figure in Figure 10 shows that the convergence speed increases when the k value decreases (as all queries are from the same cluster, thus it is more important to cover disproportional area than hot areas). In other words, the gap value is a better performance indicator than the hotness indicator. On the contrary, the second figure shows that a bigger k value can make our method more adaptive to workload pattern changes, since the hot spots, which change when workload patterns change, have a better chance to be visited than those disproportional areas.

5.4 Effect of Updates

One of the benefits of building histograms based on batch query feedback is that such a histogram keeps updating when the data distribution changes. This final set of tests evaluates how *BQHist* adapts to data distribution changes.

Workload 6: In this experiment, we progressively swap the Gaussian and Logistics datasets and test the accuracy. We start with the Gaussian dataset and test with 350 and 500 queries respectively. Then we swap a certain percentage of data with the Logistic dataset and test with another 350 and 500 queries respectively. Finally we use validation queries to test the accuracy. To simulate the reality that queries keep coming in, we set up a query pool with 2100 queries with random order. For *STHoles*, we choose the first 350 and 500 queries as validation queries. For *BQHist*, we organize the first 1050 queries in the *QT* tree for the first half of workload, but we only use 350 and 500 queries among them respectively. Then, we add another 1050 queries in *QT* and test the same amount as first half in second half workload.

Result: Figure 11 shows the accuracy of histogram adapts to data distribution changing from the Gaussian to Logistic datasets. One can observe that *BQHist* is more adaptive than *STHoles* with data distribution update, as its line is smoother than *STHoles* ins both cases.

6 Related Work

Several selectivity estimation techniques have been proposed in the past, including sampling (Wu, Agrawal & Abbadi 2002, Lipton, Naughton & Schneider 1990, Wu, Agrawal & Abbadi 2001), histograms (Aboulnaga & Chaudhuri 1999, Bruno et al. 2001, An, Yang & Sivasubramaniam 2001, Chen & Roussopoulus 1994, Donjerkovic, Ioannidis & Ramakrishnan 2000, Ioannidis & Poosala 1995, Jin, An & Sivasubramaniam 2000) and parametric technique (Konig & Weikum 1999). The tuple sampling technique (Lipton et al. 1990, Wu et al. 2001) summarizes a relation by taking uniform samples from the tuples in the relation. When a query is posed to an estimator, the estimator considers the sample size and sampling result and produces results. Parametric technique, also known as the curve-fitting technique, approximates data distributions using distribution functions with a limited number of parameters. In (Chen & Roussopoulus 1994, Sun, Ling, Rishe & Deng 1993), a general polynomial function and least squares fitting are used to choose its coefficients, while (Konig & Weikum 1999) represents the distribution as a linear combination of some mathematical functions. The coefficients of the functions are adjusted using feedback information. The main problem with this approach is that usually it is very difficult to find a function to describe arbitrary data distribution.

Histograms can be constructed either statically or dynamically (see (Loannidis 2003) for a survey). Static histograms have been well studied in literature, such as wavelet based histograms (Matias, Vitter & Wang 1998) and the V-*optimal*(F, F) (Poosala, Ioannidis, Haas & Shekita 1996) and V-*optimal*(V, F) (Jagadish, Koudas, Muthukrishnan, Poosala, Sevcik & Suel 1998) family of histograms. After histograms are built using such static approaches, buckets and frequencies remain fixed regardless of any changes in the dataset. The histograms are rebuilt when the error of selectivity estimation reaches some threshold. On the other side, dynamic histograms can capture the changes in the data distribution. This type of histograms works well for close to uniform tuple density by considering query workload information and query execution feedback to progressively refine histogram buckets. Buckets with non-uniform density can be detected and split into smaller and more accurate buckets. Adjacent buckets of similar data distribution can also be detected and merged to recuperate space for more critical regions. *STGrid* histograms use query workloads to refine a grid-based histogram structure (Aboulnaga & Chaudhuri 1999). A problem of this method is that the grid partitioning strategy can be too rigid, as data distributions generally form clusters which leads to many not-so-useful buckets. In (Lim et al. 2003), SASH is proposed to use a two-phase method to automatically build and maintain an optimal set of histograms using query feedback information. A sampling-based approach for incremental maintenance of approximate histograms is reported in (Gibbons, Mattias & Poosala 1997). In (Thaper, Guha, Indyk & Koudas 2002), dynamic histogram on data stream is addressed.

(Bruno et al. 2001) proposed a histogram called *STHoles*. It is a 'workload aware' histogram technique allows nested histogram buckets to capture regions with nearly uniform data distribution. To construct an *STHoles* histogram, one can start with an empty histogram or a single bucket histogram that covers the entire domain. The actual result of each query in the workload is intercepted from the query processor and is used to refine the histogram. By computing the overlapping regions of a query with each histogram bucket, one can refine the histogram by 'drilling holes' or zooming into the buckets that cover the query region. If the total number of buckets exceeds the fixed storage constraint, one can merge adjacent similar buckets, which results in the smallest penalty on the query estimation accuracy.

7 Conclusions

In this paper, we have proposed two effective methods to make dynamic histograms adaptive to changing workload patterns.We have addressed the problems in *STHoles* which is a representative self-tuning histograms by using an online scheduling algorithm that orders query execution such that histograms built from the feedback can adapt much more rapidly but not oversensitively to the changes of the underlying workload patterns. An selectivity estimation algorithm has also been proposed to improve estimation accuracy for queries in the areas close to but not covered by high quality buckets, using a technique based on parametric approximation. Both the online scheduling algorithm and selectivity estimation algorithms can be incrementally maintained. Our experiments have demonstrated that our methods can improve consistently the range query selectivity estimation accuracy by nearly 50%, comparing *STHoles* which is a highly effective and practical histogram method. Our future work include investigating the problem of non-aligned window query selectivity estimation, and extending our techniques to join selectivity estimation.

References

Aboulnaga, A. & Chaudhuri, S. (1999), Self-tuning histograms: Building histograms without looking at data, *in* 'SIGMOD'.

An, N., Yang, Z. Y. & Sivasubramaniam, A. (2001), Selectivity estimation for spatial joins, *in* 'ICDE'.

Bruno, N., Chaudhuri, S. & Gravano, L. (2001), Stholes: A multidimensional workload-aware histogram, *in* 'SIGMOD'.

Chen, C. M. & Roussopoulus, N. (1994), Adaptive selectivity estimation using query feedback, *in* 'SIGMOD'.

Donjerkovic, D., Ioannidis, Y. & Ramakrishnan, R. (2000), Dynamic histograms: Capturing evolving data sets, *in* 'SIGMOD'.

Gibbons, P., Mattias, Y. & Poosala, V. (1997), Fast incremental maintenance of approximate histograms, *in* 'VLDB'.

Ioannidis, Y. E. & Poosala, V. (1995), Balancing histogram optimality and practicality for query result size estimation, *in* 'SIGMOD'.

Jagadish, H. V., Koudas, N., Muthukrishnan, S., Poosala, V., Sevcik, K. C. & Suel, T. (1998), Optimal histograms with quality guarantees, *in* 'VLDB'.

Jin, J., An, N. & Sivasubramaniam, A. (2000), Analyzing range queries on spatial data, *in* 'ICDE'.

Konig, A. C. & Weikum, G. (1999), Combining histograms and parametric curve fitting for feedback-driven query result-size estimation, *in* 'VLDB'.

Lim, L., Wang, M. & Vitter, J. S. (2003), Sash: A self-adaptive histogram set for - dynamically changing workloads, *in* 'VLDB'.

Lipton, R. J., Naughton, J. F. & Schneider, D. A. (1990), Practical selectivity estimation through adaptive sampling, *in* 'SIGMOD'.

Loannidis, Y. (2003), The history of histograms (abridged), *in* 'VLDB'.

Matias, Y., Vitter, J. S. & Wang, M. (1998), Wavelet-based histograms for selectivity estimation, *in* 'SIGMOD'.

Poosala, V., Ioannidis, Y. E., Haas, P. J. & Shekita, E. J. (1996), Improved histograms for selectivity estimation of range predicates, *in* 'SIGMOD'.

Srivastava, U., Haas, P. J., Markl, V., Megiddo, N., Kutsch, M. & Tran, T. M. (2006), Isomer: Consistent histogram construction using query feedback, *in* 'ICDE'.

Sun, W., Ling, Y., Rishe, N. & Deng, Y. (1993), An instant and accurate size estimation method for joins and selections in an retrival-intensive environment, *in* 'SIGMOD'.

Thaper, N., Guha, S., Indyk, P. & Koudas, N. (2002), Dynamica multidimensional histogram, *in* 'SIGMOD'.

Wu, Y. L., Agrawal, D. & Abbadi, A. E. (2001), Applying the golden rule of sampling for query estimation, *in* 'SIGMOD'.

Wu, Y. L., Agrawal, D. & Abbadi, A. E. (2002), Query estimation by adaptive sampling, *in* 'ICDE'.

A Heuristic Approach to Cost-Efficient Derived Horizontal Fragmentation of Complex Value Databases

Hui Ma, Klaus-Dieter Schewe, Qing Wang

Massey University, Department of Information Systems
& Information Science Research Centre
Private Bag 11 222, Palmerston North, New Zealand,
Email: h.ma|k.d.schewe|q.q.wang@massey.ac.nz

Abstract

Derived horizontal fragmentation is one of the main database distribution design techniques. Unlike primary horizontal fragmentation, the decision of derived horizontal fragmentation is not straightforward. In the literature, in the context of the relational model, derived horizontal fragmentation of a member relation is achieved by performing semijoins with fragments of one of its owner relations, which is chosen in an ad hoc manner without evaluating the system performance. Similar approaches are found in the literature for the object oriented data model. We note that fragmentation and allocation are often considered separately, disregarding that they are using the same input information to achieve the same objective, i.e. improve the overall system performance. This paper addresses derived horizontal fragmentation and allocation simultaneously in the context of complex data model. The core of the paper is a heuristic approach to derived horizontal fragmentation, which uses a cost model and is targeted at globally minimising costs.

Keywords. Derived horizontal fragmentation, distribution design, cost model, heuristic procedure, complex value databases

1 Introduction

There are two types of horizontal fragmentation, primary and derived. In the literature often when horizontal fragmentation is discussed, it is mainly primary horizontal fragmentation (Zhang 1993)(Ra 1993)(Ra & Park 1993)(Shin & Irani 1991)(Bellatreche, Karlapalem & Li 1998). Derived horizontal fragmentation seems have not received the same interests as primary horizontal fragmentation, even though the impact of using it to improve the system performance is compatible to other distribution design techniques.

Unlike primary horizontal fragmentation, which is performed using predicates of queries accessing the data, the decision of derived horizontal fragmentation is not straightforward. In the literature derived horizontal fragmentation is firstly discussed in the context of the relational datamodel (RDM) and then discussed in the context of the object oriented datamodel (OODM). When derived horizontal fragmentation (DHF) is discussed in the RDM it refers to horizontal fragmentation defined on a member relation of a link according to fragmentation of one of its owner relations (Özsu & Valduriez 1999)(Ceri, Negri & Pelagatti 1982). However, when there is more than one owner relation of a member relation it is not clear which owner relation should be chosen even with the criteria given in (Özsu & Valduriez 1999). Further, it is not discussed how to deal with owner relations if their member relations have been horizontally fragmented using predicates but the owner relations do not have predicates defined on it. Later, when DHF is discussed in the context of OODM, it refers to fragmentation of an non-leaf class fragmented on fragmentation of a leaf class (Karlapalem & Navathe 1994)(Bellatreche, Karlapalem & Simonet 1997)(Bellatreche, Karlapalem & Basak 1998). Again, the restriction that derived fragmentation can only be performed on a non-leaf class based on a leaf class is not reasonable. With the existence of these deficiencies of derived horizontal fragmentation further research is essential. In the meantime, with the current popularity of web information systems that often support web-based database application, including object-oriented database, object-relational database or databases based on the eXtensible Markup Language (XML), there are needs of efficient and effective database design technique on the common aspect of these models, complex data model. In this article we discuss derived horizontal fragmentation on complex datamodel with the aim of solving the exiting problems of derived horizontal fragmentation.

The aim of database distribution design is to improve the performance of applications accessing the database. Therefore, a cost model should be employed to evaluate the total query costs of the global queries while making decisions on derived horizontal fragmentation. (Ceri et al. 1982) indicated that the important parameter needed for horizontal fragmentation is the number of accesses performed by the applications to different portions of data. However, the parameter is not used while performing derived horizontal fragmentation. Further, DHF is performed in an ad hoc way without considering how it will affect the resulting system performance. Only at the later stage of allocation, system performance is evaluated. We argue that once the decision on derived horizontal fragmentation has been made, the possibilities of minimising total query costs are restricted at the stage of allocation. In this paper, we address the problem to design derived horizontal fragmentation and to allocate fragments in a way such that the overall performance of the distributed database system is better than the one of an equivalent centralised one. That is, we first develop a query cost model for complex value databases. Then we present a heuristic approach to minimise query costs for the case of derived horizontal fragmentation. We show that the minimisation of transportation costs is decisive, and that can be achieved by refining derived horizontal fragmentation using all the candidate fragmentation schemata. The work in this article is to extend the work in (Ma, Schewe & Wang n.d.) and in (Ma, Schewe & Wang 2006).

In Section 2 we present the basic definitions of a complex value datamodel that is adapted from the Higher-Order Entity Relationship model (HERM) from (Thalheim 2000). For this model we also briefly describe a general query algebra following the general approach in (Schewe 2001) and algebraic query optimisation, for which we adapt the techniques from the RDM and from (Kirchberg, Riaz-ud-Din, Schewe & Tretiakov 2006). In addition, we briefly review the definition of horizontal fragmentation and its impact on query trees. In Section 3 we then discuss a cost model. In Section 4 we present some problems of the existing derived horizontal fragmentation approaches in the literature followed by a heuristic approach that solves the problem. Besides discussing the correctness and complexity of the heuristic we also show an experimental evaluation of the proposed heuristic. We conclude with a short summary in Section 5.

2 Complex Value Databases

In this section we briefly present the basic definitions of a complex value datamodel following the work in (Thalheim 2000). Furthermore, for this model, we briefly present a generic query algebra and discuss heuristic algebraic query optimisation.

2.1 The Datamodel

In order to define complex values we use a type system, which can be defined using abstract syntax as:

$$t = b \mid (a_1 : t_1, \dots, a_n : t_n) \mid \{t\},$$

with b as an arbitrary collection of *base types*, $(\cdot)$ and $\{\cdot\}$ as constructors for records and finite sets, respectively. Base types include *BOOL*, *OK PIC MPIC*, *CARD* and *INT*, *DATE*.

On the basis of this type system we can define database schemata, which are sets of database types. A *database type of level k* has a name E and consists of a set $comp(E) = \{r_1 : E_1, \dots, r_n : E_n\}$ of components with pairwise different role names r_i and database types E_i on levels lower than k with at least one database type of level exactly $k-1$, a set $attr(E) = \{a_1, \dots, a_m\}$ of attributes, each associated with a data type $dom(a_i)$ as its domain, and a key $id(E) \subseteq comp(E) \cup attr(E)$. We shall write $E = (comp(E), attr(E), id(E))$. A *database schema* is a finite set $\mathcal{S}$ of database types such that for all $E \in \mathcal{S}$ and all $r_i : E_i \in comp(E)$ we also have $E_i \in \mathcal{S}$. That is, schemata are closed under component references.

Given a database schema $\mathcal{S}$ we associate two types $t(E)$ and $k(E)$ – called *representation type* and *key type*, respectively – with each $E = (\{r_1 : E_1, \dots, r_n : E_n\}, \{A_1, \dots, A_k\}, \{r_{i_1} : E_{i_1}, \dots, r_{i_m} : E_{i_m}, A_{j_1}, \dots, A_{j_\ell}\}) \in \mathcal{S}$:

- The representation type of E is the tuple type $t(E) = (r_1 : t(E_1), \dots, r_n : t(E_n), A_1 : dom(A_1), \dots, A_k : dom(A_k))$.
- The key type of E is the tuple type $k(E) = (r_{i_1} : k(E_{i_1}), \dots, r_{i_m} : k(E_{i_m}), A_{j_1} : dom(A_{j_1}), \dots, A_{j_\ell} : dom(A_{j_\ell}))$.

Finally, a *database db* over a schema $\mathcal{S}$ is an $\mathcal{S}$-indexed family $\{db(E)\}_{E \in \mathcal{S}}$ such that each $db(E)$ is a finite set of values of type $t(E)$ satisfying the following two conditions:

- whenever $t_1, t_2 \in db(E)$ coincide on their projection to $id(E)$, they are already equal;
- for each $t \in db(E)$ and each $r_i : E_i \in comp(E)$ there is some $t_i \in db(E_i)$ such that the projection of t onto r_i is t_i.

Example 2.1. The following database schema is adapted from the ODIN system (Feyer, Kao, Schewe & Thalheim 2000):

Department = (∅, {name, homepage, contact}, {name})
Lecturer = ({in:Department}, {name, position, homepage, email}, {name, in:Department})
Paper = (∅, {no, kind, name, level, description, regularity, points}, {no})
Prerequisite = ({of:Paper, for:Paper}, ∅, {of:Paper, for:Paper})
Lecture = ({goal:Paper}, {semester, schedule, literature, comment}, {goal:Paper, semester})
Teach = ({who:Lecturer, for:Lecture}, ∅, {who:Lecturer, for:Lecture})

Recollect that in (Ceri, Navathe & Pelagatti 1983) member relation and owner relation are defined as the relation at the tail of a join link and a relation at the head of the link, respectively. In our complex data model the link between a member database type and a owner database type is simply a link between a database type and one of its component. For example, if $E_1 \in comp(E_2)$, then E_1 is the owner database type and E_2 is the member database type.

2.2 Query Algebra and Heuristic Query Optimisation

As query algebra for complex databases has been discussed in (Ma, Schewe & Wang 2006), we do not repeated them here. As derived horizontal fragmentation will involve semijoin, we now define semijoin in this section. With the existence of the *join types* $t_1 \bowtie_t t_2$ the join over t can be defined as

$$C_1 \bowtie_t C_2 = \{z : T_{C_1} \bowtie_t T_{C_2} \mid \exists z_1 \in C_1 . \exists z_2 \in C_2 . \\ \pi_{t_1}(z) = z_1 \wedge \pi_{t_2}(z) = z_2\}.$$

Simijoin is performed by applying join operation first and then applying projection operation:

$$C_1 \ltimes_t C_2 = \{z : \pi_{C_1}(T_{C_1} \bowtie_t T_{C_2}) \mid \exists z' \in T_{C_1} \bowtie_t \\ T_{C_2} . \wedge z = \pi_{t_1}(z')\}.$$

To discuss heuristic query optimisation we need to employ a query tree, which are drawn from a query algebra in the same way as for the relational data model. Furthermore, using the same heuristics as for the RDM, we can rearrange a query tree in a way that (if possible) we first apply structural recursion operations $src[e, g, \sqcup]$ on the sets of input database, i.e. on some $db(E)$. In particular, we first apply selections and projections, the operations that can be expressed by structural recursion (Schewe 2001), on some $db(E)$.

Projection is a special case of `map`. To define `map`, we first consider a function $f : t \to t'$ for arbitrary types t and t'. We then "raise" f to a function $\texttt{map}(f) : \{t\} \to \{t'\}$ by applying f to each element of a set. Obviously, we have $\texttt{map}(f) = \texttt{src}[\emptyset, \texttt{single} \circ f, \cup]$.

Next, considering a function $\varphi : t \to BOOL$, we define selection as an operation $\texttt{filter}(\varphi) : \{t\} \to \{t\}$, which associates with a given set the subset of all elements "satisfying the predicate" φ, i.e. elements that are mapped to $\mathbf{T}$. Then we may write

$\texttt{filter}(\varphi) = \texttt{src}[\emptyset, \texttt{if\_then\_else} \circ (\varphi \times \texttt{single} \times (\texttt{empty} \circ \texttt{triv})), \cup]$

with the function $\texttt{if\_then\_else} : BOOL \times t \times t \to t$ with $(\mathbf{T}, x, y) \mapsto x$ and $(\mathbf{F}, x, y) \mapsto y$.

2.3 Horizontal Fragmentation

Let us now define operations for horizontal fragmentation. Similar to the RDM horizontal fragmentation exploits the fact that each database type E defines a set $db(E)$ in a database db, thus can be partitioned into disjoint subsets.

There are two types of Horizonal fragmentation: primary horizontal fragmentation and derived fragmentation. Primary horizontal fragmentation refers to the fragmentation on database types using predicates (Ma 2003). Derived horizontal fragmentation refers to performing fragmentation on database types R using semijoins with fragments of its component database types R' or a database type having R as a component, i.e., $R' \in comp(R)$ or $R \in comp(R')$. As in any database db the database type E is associated with a finite set $db(E)$, we obtain an easy generalisation of relational horizontal fragmentation. For this let E be some database type. Take boolean valued functions φ_i $(i = 1, \dots, n)$ such that for each database db we obtain

$$db(E) = \bigcup_{i=1}^{n}(db(E_i)), \qquad 1 \leq i \leq n.$$

with disjoint sets $db(E_i)$. For primary fragmentation, $db(E_i) = \mathtt{filter}(\varphi_i)db(E)$. For derived fragmentation, $db(E_i) = db(E) \ltimes db(E'_i)$ with primary fragmentation schema of E', i.e., $F_{E'} = \{E'_1, \dots, E'_n\}$. There is always a remainder $db(E_{n+1}) = db(E) - (db(E) \ltimes db(E')$. We then replace E in the schema by $n+1$ new database types E_i, all with the same definition as E. Note that the biggest number for n is the number of network nodes k, i.e., $n + 1 \leq k$. Note that we do not restrict ourself to perform derived horizontal fragmentation on a database type using horizontal fragmentation schema of only its components as in (Özsu & Valduriez 1999), (Baião, Mattoso & Zaverucha 2000). This extension makes derived horizontal fragmentation to be applied in more general cases.

In the complex data model introduced in Section 2, database types are on different levels. If a type is derived fragmented with the fragmentation schema of a type of its components, then the resulting fragments will be disjoint. If a type is fragmented using the fragmentation schema of a type which has it as a component, then the properties of disjointness cannot be guaranteed. However, if an extra procedure of removing overlaps is employed, disjointness can be guaranteed.

Example 2.2. Take the schema from Example 2.1 and fragment the database type PAPER into two new instances ADVANCED_PAPER and BASIC_PAPER using $\varphi_1 \equiv$ level ≥ 300 and $\varphi_2 \equiv$ level < 300.
Fragment the database type LECTURE by semi-join with the fragments of PAPER we get:

$$\textsc{Advanced\_Lecture} = \textsc{Lecture} \ltimes \textsc{Advanced\_Paper}$$

$$\textsc{Basic\_Lecture} = \textsc{Lecture} \ltimes \textsc{Basic\_Paper}$$

Note that database type LECTURE is derived fragmented using the fragmentation schema of its component PAPER. Therefore the disjointness criteria is satisfied because the inclusion constraints between a database type and its component, i.e., t[PAPER](LECTURER) = t(PAPER).

Horizontal fragmentation corresponds to replacing E in the query tree by some union $E_1 \cup \dots \cup E_n$. Another round of query optimisation might shift the selection $\mathtt{filter}(\varphi)$ and the projection $\mathtt{map}(\pi_X)$ inside the newly introduced union, but the "upper part" of the query tree would not be affected. Therefore, in order to optimise horizontal fragmentation, it is decisive and sufficient to consider subqueries of the form the following (Ma et al. n.d.).

$$\mathtt{map}(\pi_X)(\mathtt{filter}(\varphi)(db(E))) \qquad (*)$$

3 A Cost Model

Taking the same cost model as in (Ma, Schewe & Wang 2006) we now analyse the query costs in the case of derived horizontal fragmentation. Size calculation for leaves and nodes are discussed in (Ma, Schewe & Wang 2006). For the convenience of discussion we briefly present the cost model in the following. The major objective is to base the fragmentation decision on the efficiency of the most frequent queries. As a general pragmatic guideline we follow the recommended rule of thumb to consider only the 20% most frequent queries, as these usually account for most of the data access (Özsu & Valduriez 1999).

Assume fragmentation of type E results in a set of fragments $\{E_1, \dots, E_n\}$ of average sizes $s_1, \dots, s_n$. If the network has a set of nodes $N = N_1, \dots, N_k$ we have to allocate these fragments to one of the nodes, which gives rise to a mapping $\lambda : \{1, \dots, n\} \to \{1, \dots, k\}$, which we call a *location assignment*. This decide the allocation of leaves of query trees, which are fragments. For each intermediate node v in each relevant query tree, we must also associate a node $\lambda(v)$, i.e., $\lambda(v)$ indicating the node in the network that the intermediate query result corresponding to v will be stored at.

Given a location assignment λ we can compute the total costs of query processing. Let the set of queries be $Q^m = \{Q_1, \dots, Q_m\}$. Query costs are composed of two parts: *storage costs* and *transportation costs*: $costs_\lambda(Q_j) = stor_\lambda(Q_j) + trans_\lambda(Q_j)$.

The storage costs give a measure for retrieving the data back from secondary storage, which is mainly determined by the size of the data. The storage costs of a query Q_j depend on the size of the intermediate results and on the assigned locations, which decide the storage cost factors. It can be expressed as

$$stor_\lambda(Q_j) = \sum_h s(h) \cdot d_{\lambda(h)},$$

where h ranges over the nodes of the query tree for Q_j, $s(h)$ are the sizes of the involved sets, and d_i indicates the storage cost factor for node N_i $(i = 1, \dots, k)$.

The transportation costs provide a measure for transporting between two nodes of the network. The transportation costs of query Q_j depend on the sizes of the involved sets and on the assigned locations, which decide the transport cost factor between every pair of sites. It can be expressed by

$$trans_\lambda(Q_j) = \sum_h \sum_{h'} c_{\lambda(h')\lambda(h)} \cdot s(h').$$

Again the sum ranges over the nodes h of the query tree for Q_j, h' runs over the predecessors of h in the query tree, and c_{ij} is the transportation cost factor for data transport from node N_i to node N_j $(i, j \in \{1, \dots, k\})$.

Furthermore, for each query Q_j we assume a value for its frequency f_j. The total costs of all the queries in Q^m are the sum of the costs of each query multiplied by its frequency:

$$\sum_{j=1}^{m} cost_\lambda(Q_j) \cdot f_j.$$

In general, the distribution could be called optimal if we find a fragmentation and allocation schema such that the resulting total query costs are minimal. As this problem is practically incomputable, we suggest to use a heuristic instead.

4 A Heuristic Method for Derived Horizontal Fragmentation and Allocation

In the following we first present an example to show the problems of existing approaches of derived horizontal fragmentation. We will attempt to solve the problem by first analysing the cost model and then propose a heuristic method based on the result of the analysis. We will show how the heuristic method is applied with a simple example. Then we will prove that the proposed heuristic is correct with regard to the criteria of correctness of fragmentation in (Özsu & Valduriez 1999).

4.1 An Motivating Example

Assume there are three relations A, B, C in a database. Relation C is accessed by four queries $Q_1, \ldots, Q_4$ with different frequencies $f_1, \ldots, f_4$. Relation A is accessed by Q_1 and Q_2. Relation B is accessed by Q_3 and Q_4. Relation A and B have been horizontally fragmented using predicates. Relation A has been fragmented into A_1, A_2 which are allocated to site 1 and 2 respectively, i.e., $\lambda(A_1) = 1$, $\lambda(A_2) = 2$. Relation B has been fragmented into two fragments, B_3 and B_4 which are allocated to site 3 and 4, respectively. Assume that there is no predicate defined on C, which is accessed by all four queries, and $\text{MAX}(f_1, f_2, f_3, f_4) = f_1$, performing only primary horizontal fragmentation we will have Scenario I depicted in Figure 1, in which C is allocated to site 1, i.e., $\lambda_1(C) = 1$ according to cost optimisition rule. In figures below, all remote transactions and their frequencies are depicted with solid lines and values on the lines, while local transactions are depicted in dashed lines with their frequencies marked on the lines.

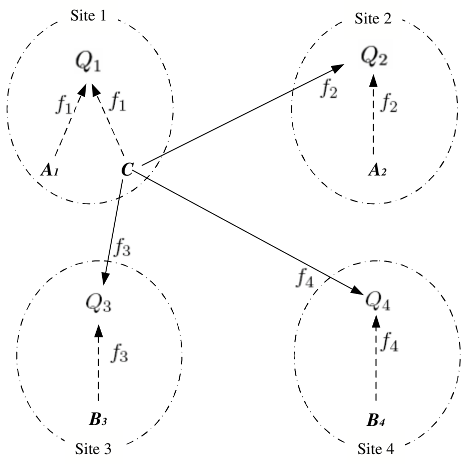


Figure 1: Scenario I - Primary Fragmentation Only

Assuming that the transportation cost factors among all sites are same, the total query costs of Scenario I is computed as:

$$cost_{\lambda_1}(Q) = s_C \cdot f_2 + s_C \cdot f_3 + s_C \cdot f_4$$

When a member relation has more than one owner relation, there will be more than one possible derived horizontal fragmentation schemata. In this case it is recommended to choose a fragmentation that is used in more applications (Özsu & Valduriez 1999). Assume $f_1 + f_2 > f_3 + f_4$, relation C is therefore derived horizontally fragmented by semi-join with fragments of A. The resulting fragmentation and fragment allocation is depicted in Scenario II in Figure 2. The total query costs for Scenario II are:

$$cost_{\lambda_2}(Q) = s_{C_1} \cdot f_3 + s_{C_2} \cdot f_3 + s_{C_1} \cdot f_4 + s_{C_2} \cdot f_4$$

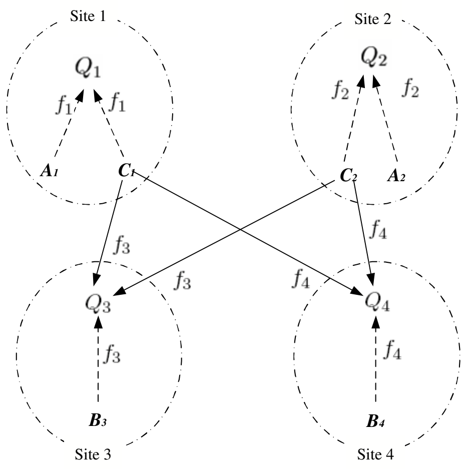


Figure 2: Scenario II - Primary and Derived Fragmentation on One Fragmentation Scheme

However, performing derived fragmentation based on fragmentation schema of relation A can only improve the performance of the queries which access both the member relation C and owner relation A. However, the performance of the queries that access the member relation C together with another owner relation B cannot be improved. That is, the chance of optimising the system performance is restricted, if the fragmentation of C is only based on fragmentation of one owner relation. Now we look at what happens if we take into consideration fragmentation of both owner relations. In this case, we have two fragmentation schemata for C, i.e., $F_C = \{C_{1a}, C_{2a}\}$ and $F'_C = \{C_{3b}, C_{4b}\}$ with:

$$C_{1a} = C \ltimes A_1, C_{2a} = C \ltimes A_2$$

$$C_{3b} = C \ltimes B_3, C_{4b} = C \ltimes B_4$$

Applying intersection operation on $C_{ia} \in F_C, (1 \leq i \leq 2)$ and $C_{jb} \in F'_C, (3 \leq j \leq 4)$ we get the following finer fragmentation:

$$C_{1a3b} = C_{1a} \cap C_{3b}, C_{1a4b} = C_{1a} \cap C_{4b},$$

$$C_{2a3b} = C_{2a} \cap C_{3b}, C_{2a4b} = C_{2a} \cap C_{4b}$$

Assuming $f_1 > f_3, f_1 > f_4, f_2 < f_3, f_2 > f_4$, with the cost model introduced in 3 we get the optimized allocation of the four atom fragments as following:

$$\lambda(C_{1a3b}) = 1, \lambda(C_{1a4b}) = 1,$$

$$\lambda(C_{2a3b}) = 3, \lambda(C_{2a4b}) = 2$$

Scenario III in Figure 3 depicts the finer derived horizontal fragmentation and fragment allocation.

In the case of Scenario III, the total query costs are:

$$cost_{\lambda_3}(Q) = s_{C_{2a3b}} \cdot f_2 + s_{C_{1a}} \cdot f_3 + s_{C_{1a}} \cdot f_4 + s_{C_{2a4b}} \cdot f_4$$

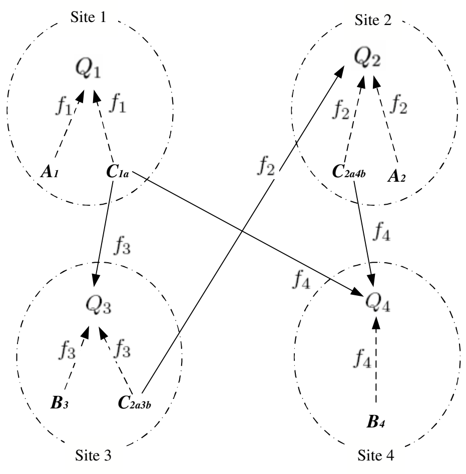


Figure 3: Scenario III - Primary and Derived Fragmentation on Two Fragmentation Schemata

Comparing with the costs in Scenario I and Scenario II we get:

$$\begin{aligned} & cost_{\lambda_1}(Q) - cost_{\lambda_2}(Q) \\ = \; & s_C \cdot f_2 + s_C \cdot f_3 + s_C \cdot f_4 - (s_{C_1} \cdot f_3 + s_{C_2} \cdot f_3 \\ & + s_{C_1} \cdot f_4 + s_{C_2} \cdot f_4) \\ = \; & s_C \cdot f_2 + s_C \cdot f_3 + s_C \cdot f_4 - (s_{C_1} + s_{C_2}) \cdot f_3 \\ & -(s_{C_1} + s_{C_2}) \cdot f_4 \\ > \; & 0 \end{aligned}$$

We can conclude that the derived fragmentation indeed can further reduce the total query costs and therefore should be employed while doing database distribution design.

$$\begin{aligned} & cost_{\lambda_2}(Q) - cost_{\lambda_3}(Q) \\ = \; & s_{C_1} \cdot f_3 + s_{C_2} \cdot f_3 + s_{C_1} \cdot f_4 + s_{C_2} \cdot f_4 \\ & -(s_{C_{2a3b}} \cdot f_2 + s_{C_{1a}} \cdot f_3 + s_{C_{1a}} \cdot f_4 + s_{C_{2a4b}} \cdot f_4) \\ = \; & s_{C_2} \cdot f_3 + s_{C_2} \cdot f_4 - (s_{C_{2a3b}} \cdot f_2 + s_{C_{C2a4b}} \cdot f_4) \\ > \; & 0 \end{aligned}$$

The above formula proves that $cost_{\lambda_2}(Q) > cost_{\lambda_3}(Q)$. This result shows that a finer derived fragmentation approach can lead to better system performance than derived fragmentation based on fragmentation schema of one owner relation.

4.2 Some Terms

We now define some terms to facilitate our discussion of derived horizontal fragmentation. Let $db(E_{ji}) = \{t | t \in \texttt{filter}(\varphi_j)(db(E_i))\}$ denote the set of tuples of E_i accessed by query Q_j.

As we do not restrict derived fragmentation to be performed on a member relation only according to one of its owner relation, in the complex data model we introduce terms: *target type* and *related type*, which have broader meanings. A *target type* is a database type to be derived fragmented using semijoin with fragments of other database types, either at its lower level or its high level.

A *related type* of a *target* type is a type that has been horizontally fragmented and are accessed by queries together with the target type.

The *request* of a fragment E_i^k at a site h over a network is the sum of frequencies of all the queries that are issued at site h and access E_i^k:

$$request_h(E_i^k) = \sum_{j=1, \lambda(Q_j)=h, db(E_{ji}) \cap db(E_i^k) \neq \emptyset}^{m} f_j$$

The *affinity* between a target type E_d and one fragment E_i^k of its related type is the sum of frequencies of all the queries Q_j accessing E_d and the fragment E_i^k together at site h:

$$aff_h(E_d, E_i^k) = \sum_{j=1, \lambda(Q_j)=h, db(E_{ji}) \cap db(E_i^k) \neq \emptyset, db(E_{ji}) \cap db(E_d) \neq \emptyset}^{m} f_j$$

When there is more than one fragmentation schema of a given target type, we define *atom derived horizontal fragments*, or *atom fragments* in short, as the intersections of fragments of one fragmentation schema with fragments of another fragmentation schema. For example, $F_C = \{C_1, \dots, C_m\}$ and $F'_C = \{C'_1, \dots, C'_n\}$, then $C_i \cap C'_i$ is an atom fragment.

According to our discussion of how horizontal fragmentation affects query costs, the allocation of fragments to network nodes, following the cost minimisation heuristics, already determine the location assignment provided that an optimal location assignment for the queries was given prior to the fragmentation. Horizontal fragmentation will only change the subqueries in the form of (*). In this case we evaluate a derived horizontal fragmentation by comparing the total query costs before and after the fragmentation. After the derived horizontal fragmentation the instance of a database type will be replaced by a set of atom fragments which are allocated to network nodes that lead to the least query costs.

Taking the cost model introduced in Section 3 we now investigate how total query costs are affected by derived horizontal fragmentation. Assume there are two related types A and B, one target type C. Let $C_{iai'b}$ be the atom fragment, λ_1 indicate a distribution design without derived horizontal fragmentation, λ_2 indicate a distribution design with derived horizontal fragmentation and fragment allocation, t_j be the optimal allocation of the root of subqueries in the form (*), x indicate the site that C is allocated before it is derived horizontally fragmented, y denote an optimal allocation of atom fragment $C_{iai'b} = C_{ia} \cap C_{i'b}$. As the transportation costs dominate the total query costs, we get the following formulae:

$$\begin{aligned} & cost_{\lambda_1}(Q^m) - cost_{\lambda_2}(Q^m) \\ = \; & \sum_{j=1}^{m} cost_{\lambda_1}(Q_j) \cdot f_j - \sum_{j=1}^{m} cost_{\lambda_2}(Q_j) \cdot f_j \\ = \; & \sum_{j=1}^{m} (\sum_{h_1} \sum_{h'_1} c_{\lambda(h'_1)\lambda(h_1)} \cdot s(h'_1)) \cdot f_j \\ & - \sum_{j=1}^{m} (\sum_{h_2} \sum_{h'_2} c_{\lambda(h'_2)\lambda(h_2)} \cdot s(h'_2)) \cdot f_j \\ = \; & \sum_{j=1}^{m} c_{t_j x} \cdot s_C \cdot f_j - \sum_{j=1}^{m} (\sum_{i=1}^{k} \sum_{i'=1}^{k} s_{C_{iai'b}} \cdot c_{t_j y}) \cdot f_j \end{aligned}$$

In order to maximize the value of $cost_{\lambda_1}(Q^m) - cost_{\lambda_2}(Q^m)$ we need to minimise the value of $\sum_{j=1}^{m} (\sum_{i=1}^{k} \sum_{i'=1}^{k} s_{C_{iai'b}} \cdot c_{t_j y}) \cdot f_j$. For a single atom fragment $C_{iai'b}$, we need to minimise the value of $\sum_{j=1}^{m} c_{t_j y} \cdot$

f_j. This leads to a heuristic which allocates atom fragments $C_{iai'b}$ to a site that accesses it most often by queries Q_j together with a related fragment, either A_i or $B_{i'}$. The optimal allocation is $y = t_j$ in which case $c_{t_j y} = 0$. That is, we allocate the atom fragment $C_{iai'b}$ to a site that request it most often. This will maximize the local data availability for the most frequent queries. The accesses of $C_{iai'b}$ by queries are either with A_i, $B_{i'}$ or no of them. Therefore the difference between $aff(C, A_i)$ $aff(C, B_{i'})$ will reflect the local *request* at site i and i', and indicate the difference between $request_i(C)$ and $request_{i'}(C)$. In other words, we should allocate an atom fragment to the same site of the related fragment which has the highest *affinities* with the target type. In the following section we present a heuristic procedure for derived horizontal fragmentation.

4.3 Heuristics for Derived Horizontal Fragmentation

We perform derived horizontal fragmentation with the following steps. Read and write queries are not distinguished because replication is not considered at this stage.

1. Take the most frequently used 20% queries Q^m.
2. Process primary horizontal fragmentation using the heuristic in (Ma, Schewe & Wang 2006) to get a set of primary horizontal fragmentation schemata.
3. Get a set of target types that have not been fragmented primarily but are accessed together with some related fragments.
4. For each of the target database types find the set of queries that accessed both the target type and corresponding related types and get the frequencies of each query.
5. For each of the target type get the *request* of each fragment of the related data types.
6. Use fragmentation schemata of each of related types to perform derived fragmentation of the target type. Allocate resulting fragments to the same site of the corresponding related fragment involved in the semi-join. Remove overlaps between each pair of the resulting fragments. A overlap part is allocated to the same site as the related fragment that are requested the most by queries.
7. If there are more than one derived fragmentation schema from step 3 perform derived fragmentation refinement by performing intersection between every pair fragments from two different schemata to get a set of atom fragment.
8. Allocate atom fragment to the same site of the related fragments that have the highest affinity with the target type.

This procedure is formally described by the algorithm below.

Algorithm 1 (Derived Horizontal Fragmentation and Fragment Allocation).

```
Input: $Q^m = \{Q_1, \dots, Q_m\}$ /* a set of global queries
       $E_d$ /* a type with a set of components and
          attributes
       a set of network nodes $N = \{1, \dots, k\}$
       a set of fragmentation schemata resulting
          from primarily fragmentation
          $E_i = E_i^1 \cup \dots \cup E_i^k$
Output: derived horizontal fragmentation schema
          and fragment allocation schema
Begin
  for each $h \in \{1, \dots, k\}$ let $E_d^h = \emptyset$ endfor
    for each related type $E_i \in \{E_1, \dots, E_c\}$ do
      $E_{di}^h = E_d \ltimes E_i^h$
    endfor
    for each fragments $E_{di}^h$ /* remove overlaps
      for each fragments $E_{di}^{h'}$ do
        $E_{di}^{hh'} = E_{di}^h \cap E_{di}^{h'}$
        choose $y$ such that $request(E_i^y) =$
              $\min\{request(E_i^h), request(E_i^{h'})\}$
        $E_{di}^y = E_{di}^y - E_{di}^{hh'}$ /* remove intersection
              from the smaller request node
      endfor
    endfor
    for each $E_{di}^h$ do
      for each $E_{di'}^{h'}$ do
        $E_{dii'}^{hh'} = E_{di}^h \cap E_{di'}^{h'}$
        $aff_h(E_d, E_x^z) =$
            $\max\{aff_h(E_d, E_i^k), aff_h(E_d, E_{i'}^{k'})\}$
        $E_d^z = E_d^z \cup E_{dij}^{hh'}$
      endfor
    endfor
  endfor
```

Example 4.1. Taking again the database schema in Example 2.1, we now assume for a target database type LECTURER, there are two related database types, DEPARTMENT and TEACH, each of which has been horizontally fragmented into three fragments that are allocated to network notes, 1, 2, 3, respectively, i.e., $F_{\text{DEPARTMENT}} = \{\text{DEPARTMENT}_1, \text{DEPARTMENT}_2, \text{DEPARTMENT}_3\}$, $F_{\text{TEACH}} = \{\text{TEACH}_1, \text{TEACH}_2, \text{TEACH}_3\}$. To perform derived horizontal fragmentation of type LECTURER we go through the following procedure:

1. With each related type semijoin is performed to get a set of horizontal fragments. Remove the overlap between each pair of fragments.

   Type LECTURER can then be derived horizontally fragmented according to the fragmentation schemata of DEPARTMENT and TEACH. The fragments resulting from semijoin with fragments of DEPARTMENT are:

   $\text{LECTURER}_{1D'} = \text{LECTURER} \ltimes \text{DEPARTMENT}_1$,
   $\text{LECTURER}_{2D'} = \text{LECTURER} \ltimes \text{DEPARTMENT}_2$,
   $\text{LECTURER}_{3D'} = \text{LECTURER} \ltimes \text{DEPARTMENT}_3$,
   $\text{LECTURER}_{4D'} = \text{LECTURER} - (\text{LECTURER} \ltimes \text{DEPARTMENT})$

   Because DEPARTMENT is a component of LECTURER. Therefore, the above fragments are disjoint. Also, all objects for DEPARTMENT in LECTURER must be in one of fragments of DEPARTMENT. Hence, $\text{LECTURER}_{4D'}$ is always a empty set. Therefore, we can directly get the following disjoint fragments:

   $\text{LECTURER}_{1D}, \text{LECTURER}_{2D}, \text{LECTURER}_{3D}$.

   Similarly, fragment LECTURER by performing semijoin with fragments of TEACH we get

   $\text{LECTURER}_{1T'} = \text{LECTURER} \ltimes \text{TEACH}_1$,
   $\text{LECTURER}_{2T'} = \text{LECTURER} \ltimes \text{TEACH}_2$,
   $\text{LECTURER}_{3T'} = \text{LECTURER} \ltimes \text{TEACH}_3$,
   $\text{LECTURER}_{4T'} = \text{LECTURER} - (\text{LECTURER} \ltimes \text{TEACH})$

   Because LECTURER is a component of TEACH there might be overlaps between the above fragments. Removing overlap we get the following disjoint fragments:

$\text{Lecturer}_{1T}, \text{Lecturer}_{2T}$,
$\text{Lecturer}_{3T}, \text{Lecturer}_{4T}$.

2. Perform intersection between all fragments resulted from semijoin with Department and fragments resulted from semijoin with fragments of Teach, we have 12 intersections:

   $\text{Lecturer}_{1D1T} = \text{Lecturer}_{1D} \cap \text{Lecturer}_{1T}$,
   $\text{Lecturer}_{1D2T} = \text{Lecturer}_{1D} \cap \text{Lecturer}_{2T}$,
   $\text{Lecturer}_{1D3T} = \text{Lecturer}_{1D} \cap \text{Lecturer}_{3T}$,
   $\text{Lecturer}_{1D4T} = \text{Lecturer}_{1D} \cap \text{Lecturer}_{4T}$,

   $\text{Lecturer}_{2D1T} = \text{Lecturer}_{2D} \cap \text{Lecturer}_{1T}$,
   $\text{Lecturer}_{2D2T} = \text{Lecturer}_{2D} \cap \text{Lecturer}_{2T}$,
   $\text{Lecturer}_{2D3T} = \text{Lecturer}_{2D} \cap \text{Lecturer}_{3T}$,
   $\text{Lecturer}_{2D4T} = \text{Lecturer}_{2D} \cap \text{Lecturer}_{4T}$,

   $\text{Lecturer}_{3D1T} = \text{Lecturer}_{3D} \cap \text{Lecturer}_{1T}$,
   $\text{Lecturer}_{3D2T} = \text{Lecturer}_{3D} \cap \text{Lecturer}_{2T}$,
   $\text{Lecturer}_{3D3T} = \text{Lecturer}_{3D} \cap \text{Lecturer}_{3T}$,
   $\text{Lecturer}_{3D4T} = \text{Lecturer}_{3D} \cap \text{Lecturer}_{4T}$.

3. For each of the intersections we decide its allocation based on the allocation of corresponding related fragments and their frequencies. Two situations may occur.

   - Intersections resulting from fragments at the same site.

     Among the above intersections, $\text{Lecturer}_{1D1T}, \text{Lecturer}_{2D2T}$, Lecture -$\text{r}_{3D3T}$ are resulted from intersections of the fragments at the same network node. Therefore we allocate them at the same site of this related fragments.

     $\lambda(\text{Lecturer}_{1D1T}) = 1$,
     $\lambda(\text{Lecturer}_{2D2T}) = 2$,
     $\lambda(\text{Lecturer}_{3D3T}) = 3$,

   - Intersections resulting from fragments at different sites.

     In this case the affinities between related types, involved in the intersections, and the target type will be used to decide the allocation of the atom fragment. The related fragment that have highest affinity with the target type will decide the allocation of the atom derived fragment.

     For example, the allocation of Lecturer_{1D2T} will be decided by the values of $aff(\text{Lecturer}, \text{Department}_1)$, $aff(\text{Lecturer}, \text{Teach}_2)$. If $aff(\text{Lectur-er}, \text{Department}_1) = \text{MAX}\{(aff(\text{Lectu-rer}, \text{Department}_1), aff(\text{Lecturer}, \text{Te-ach}_2)\}$ then allocate C_{1D2T} to site 1. Otherwise allocate it to site 2.

Example 4.2. Looking back at the example in 4.1 there are four atom fragments, $C_{1a3b}C_{1a4b}, C_{2a3b}, C_{2a4b}$. To allocate this atom fragments we compare affinities. For example, to allocate C_{1a3b} we compare affinities $aff(C, A_1), aff(C, B_3)$. As $aff(C, A_1) = f_1$ and $aff(C, B_3) = f_3$ and $f_1 > f_3$. Hence, we allocate C_{1a3b} to site 1.

In the same way we have:
$\lambda(C_{1a4b}) = 1$ because $aff(C, A_1) = f_1$, $aff(C, B_4) = f_4$ and $f_1 > f_4$
$\lambda(C_{2a3b}) = 3$ because $aff(C, A_2) = f_2$, $aff(C, B_3) = f_3$ and $f_3 > f_2$.
$\lambda(C_{2a4b}) = 2$ because $aff(C, A_2) = f_2$,$aff(C, B_4) = f_4$ and $f_2 > f_4$.

The allocation resulted from the heuristic using affinities is the same as the optimised allocation in the example in 4.1.

4.4 Discussion

In this section we will prove that the proposed approach for derived horizontal fragmentation is correct with regard to the criteria in (Özsu & Valduriez 1999). In addition we will analysis the complexity of the proposed approach.

Let C be an instance of a target database type, A and B be the instances of two related types, which are fragmented as $F_A = \{A_1, \ldots, A_m\}$, $F_B = \{B_1, \ldots, B_n\}$. Database type C have common attribute between A, and B. That means C is either a component of A and B of have A or B as a component. The following shows that criteria of fragmentation, disjointness, reconstruction, completeness, are satisfied.

- **Completeness:** As shown in the definition of derived horizontal fragmentation, there always is a remainder which contains all instances of C which do not match instance in A or B. In other words, if an object can not be selected using semi-join with fragments of A or B, it will be in the remainder fragment.

- **Disjointness:** To check disjointness between each pair of atom derived fragments $C_{iajb} \cap C_{i'aj'b}$ we can check the following three situations: $i = i'$, $j = j'$ and $i \neq i' \wedge j \neq j'$.

  For the first two situations the prove of disjointness are straightforward. Because C_{jb} and $C_{j'b}$ are disjoint, the disjointness between C_{iajb} and $C_{iaj'b}$ is guaranteed.

  $$\begin{aligned} C_{iajb} \cap C_{iaj'b} &= (C_{ia} \cap C_{jb}) \cap (C_{ia} \cap C_{j'b}) \\ &= C_{ia} \cap (C_{jb} \cap C_{j'b}) \\ &= \emptyset \end{aligned}$$

  Similarly, because C_{ia} and $C_{i'a}$ are disjoint C_{iajb} and $C_{i'ajb}$ is disjoint.

  $$\begin{aligned} C_{iajb} \cap C_{i'ajb} &= (C_{ia} \cap C_{jb}) \cap (C_{i'a} \cap C_{jb}) \\ &= C_{jb} \cap (C_{ia} \cap C_{i'a}) \\ &= \emptyset \end{aligned}$$

  For general cases $i \neq i' \wedge j \neq j'$.

  $$\begin{aligned} C_{iajb} \cap C_{i'aj'b} &= (C_{ia} \cap C_{jb}) \cap (C_{i'a} \cap C_{j'b}) \\ &= (C_{ia} \cap C_{i'a}) \cap (C_{jb} \cap C_{j'b}) \\ &= \emptyset \end{aligned}$$

- **Reconstruction:** The formulae below show that the union of all atom derived fragments reconstruct the original instance C.

  $$\begin{aligned} \bigcup_{i=1}^{m}\bigcup_{j=1}^{n} C_{iajb} &= \bigcup_{i=1}^{m}\bigcup_{j=1}^{n}(C_{ia} \cap C_{jb}) \\ &= \bigcup_{i=1}^{m}(C_{ia} \cap \bigcup_{j=1}^{n} C_{jb}) \\ &= \bigcup_{i=1}^{m}(C_{ia} \cap C) \\ &= C \cap \bigcup_{i}^{m} C_{ia} \\ &= C \cap C \\ &= C \end{aligned}$$

The complexity of this approach is higher than the traditional approaches using fragmentation of one owner relation but the improvement of system performance make the pay on complexity worthwhile. Lets c be the number of related types of a given target type, n be the number of records of the target type, m be the average number of records of instances of related types, k be the number of network nodes. The complexity of our approach, which deals with derived fragmentation and allocation, is $O(k^c + c \cdot m \cdot \log(m) + n \cdot \log(n) + c \cdot k^2)$ for derived fragmentation procedure, including performing semi-join, removing overlap and fragment. For example, if there are two related types for a target type, the complexity of the traditional approach is $O(m \cdot \log(m) + n \cdot \log(n) + k^2)$ while our approach is $O(2 \cdot m \cdot \log(m) + n \cdot \log(n) + 3 \cdot k^2)$. The complexity, for the one time design procedure, does not change very much while the system performance can be indeed improved, for the long term using of the system.

4.5 Experimental Evaluation of the Heuristics

We present here some experiments that have been conducted to verify the algorithm, *DR_Frag_Alloc*, proposed above. We used the same testbed as in (Ma, Schewe & Wang 2006). The testbed has been designed with a database schema $\mathcal{S}$, which was populated with records to get $db(\mathcal{S})$. Then we assumed from four sites over a network there are 30 queries, which were the 20% most frequently queries or used by most critical transactions. These 30 queries were designed by applying the similar pattern of queries as in OO7 project (Carey, DeWitt & Naughton 1993). According to the well-known 20/80 rule, the system performance is assessed by the total query costs of these 30 queries. Some of the types were accessed by queries with predicates while other types, *target types*, were accessed by queries though joining with *related types* or directly. To test the heuristic we designed queries such that there were two target types, each of which had two related types. The types that have predicates defined on had been horizontally fragmented using the heuristics proposed in (Ma, Schewe & Wang 2006). With these fragmented related types we performed derived horizontal fragmentation on the target types using the following two approaches:

- case I: using the algorithm 1 introduced above.
- case II: using the traditional approach based on fragments of one owner type. (Özsu & Valduriez 1999).

We ran the tests on three different instances of different sizes. Comparing the results we use the following table:

case	I	II
Instance 1	$21079 \cdot 10^6$	$21152 \cdot 10^6$
Instance 2	$78111 \cdot 10^6$	$78456 \cdot 10^6$
Instance 3	$136565 \cdot 10^6$	$136724 \cdot 10^6$

The experimental results showed that the total query costs for case I is smaller than for case II on all three different database instances. This means that our heuristic approach for derived horizontal fragmentation can lead to better system performance than using the traditional approaches according to fragmentation of one owner relation. This valid our proposed heuristic approach in this article. Further, we observed that the time to process the tests using our approach was of similar length as the time using the traditional approach. Furthermore, even though the improvement of performance is not significant, considering that the two target types are of small sizes, each of which only has 0.5% of the total number of the tuples of the database instances, and that the queries accessing the target types only count for about 12% total query costs, we can expect better performance improvement for some other database instances and queries.

5 Conclusion

In this paper we presented a heuristic approach to derived horizontal fragmentation for complex datamodel. The work in this paper complement the work in (Ma, Schewe & Wang 2006) to provide a complete design procedure of horizontal fragmentation, including primary horizontal fragmentation and derived horizontal fragmentation, for complex datamodel. The major objective is to provide a tractable approach to minimising the query processing costs by performing horizontal fragmentation and fragment allocation simultaneously.

The next step of our work is to integrate the handing of horizontal fragmentation, which has been discussed in (Ma, Schewe & Wang 2006) and in this paper, and vertical fragmentation, which has been discussed in (Ma, Schewe & Kirchberg 2006), with the consideration of the requirement of global optimisation.

References

Baião, F., Mattoso, M. & Zaverucha, G. (2000), Horizontal fragmentation in object dbms: New issues and performance evaluation, *in* ‘Proc. The 19th IEEE International Performance, Computing and Communications Conference’, IEEE CS Press, Phoenix, pp. 108–114.

Bellatreche, L., Karlapalem, K. & Basak, G. (1998), Horizontal class partitioning for queries in object oriented databases, Technical report, HKUST-CS98-6.

Bellatreche, L., Karlapalem, K. & Li, Q. (1998), Complex methods and class allocation in distributed object-oriented database systems, *in* ‘Object Oriented Information Systems’, pp. 239–256.

Bellatreche, L., Karlapalem, K. & Simonet, A. (1997), Horizontal class partitioning in object-oriented databases, *in* ‘DEXA ’97: Proceedings of the 8th International Conference on Database and Expert Systems Applications’, Springer-Verlag, London, UK, pp. 58–67.

Carey, M. J., DeWitt, D. J. & Naughton, J. F. (1993), ‘The OO7 benchmark’, *SIGMOD Record (ACM Special Interest Group on Management of Data)* **22**(2), 12–21.

Ceri, S., Navathe, S. & Pelagatti, G. (1983), ‘Distribution design of logical database schemes’, *IEEE Trans. Software Eng* **SE-9**(4), 487–503.

Ceri, S., Negri, M. & Pelagatti, G. (1982), Horizontal data pertitioing in database design, *in* ‘Proc. the ACM SIGMOD International Conference on Management of Data’, pp. 128–136.

Feyer, T., Kao, O., Schewe, K.-D. & Thalheim, B. (2000), Design of data-intensive web-based information services, *in* Q. Li, Z. M. Ozsuyoglu,

R. Wagner, Y. Kambayashi & Y. Zhang, eds, 'Proceedings of the 1st International Conference on Web Information Systems Engineering (WISE 2000)', IEEE Computer Society, pp. 462–467.

Karlapalem, K. & Navathe, S. B. (1994), Materialization of redesigned distributed relational databases, Master's thesis, Hong Kong University of Science and Technology, Hong Kong.

Kirchberg, M., Riaz-ud-Din, F., Schewe, K.-D. & Tretiakov, A. (2006), 'Towards algebraic query optimisation for xquery', *LNCS Journal on Data Semantics VII* . to appear.

Ma, H. (2003), Distribution design in object oriented databases, Master's thesis, Massey University.

Ma, H., Schewe, K.-D. & Kirchberg, M. (2006), A heuristic approach to vertical fragmentation incorporating query information, *in* O. Vasilecas, J. Eder & A. Caplinskas, eds, 'Proceedings of the 7th International Baltic Conference on Databases and Information Systems', IEEE, pp. 69–76.

Ma, H., Schewe, K.-D. & Wang, Q. (2006), A heuristic approach to cost-efficient fragmentation and allocation of complex value databases, *in* G. D. J. Bailey, ed., 'Proc. ADC 2006', Vol. 49 of *CRPIT*, Hobart, Australia.

Ma, H., Schewe, K.-D. & Wang, Q. (n.d.), 'Distribution design for higher-order data models', *Data and Knowledge Engineering* . to appear.

Özsu, M. T. & Valduriez, P. (1999), *Principles of Distributed Database Systems*, Alan Apt, New Jersey.

Ra, M. (1993), Horizontal partitioning for distributed database design, *in* M. Orlowska & M. Papazoglou, eds, 'Advances in Database Research', World Scientific Publishing, pp. 101–120.

Ra, M. & Park, Y.-S. (1993), Data fragmentation and allocation for pc-based distributed database, *in* 'The Third International Symposium on Database Systems for Advanced Applications', Taejon, South Korea.

Schewe, K.-D. (2001), On the unification of query algebras and their extension to rational tree structures, *in* M. Orlowska & J. Roddick, eds, 'Proc. Australasian Database Conference 2001', Australian Computer Society, pp. 52–59.

Shin, D.-G. & Irani, K. B. (1991), 'Fragmenting relations horizontally using a knowledge-based approach', *IEEE Trans. Softw. Eng.* **17**(9), 872–883.

Thalheim, B. (2000), *Entity-Relationship Modeling: Foundations of Database Technology*, Springer-Verlag.

Zhang, Y. (1993), On horizontal fragmentation of distributed database design, *in* M. Orlowska & M. Papazoglou, eds, 'Advances in Database Research', World Scientific Publishing, pp. 121–130.

Condensative Stream Query Language for Data Streams

Lisha Ma[1] **Werner Nutt**[2] **Hamish Taylor**[1]

[1]School of Mathematical and Computer Sciences
Heriot-Watt University, Edinburgh, UK
[2]Faculty of Computer Science
Free University of Bozen-Bolzano, Italy

Abstract

In contrast to traditional database queries, a query on stream data is continuous in that it is periodically evaluated over fractions (sliding windows) of the data stream. This introduces challenges beyond those encountered when processing traditional queries. Over a traditional DBMS (Database Management System), the answer to an aggregate query is usually much smaller than the answer to a similar non-aggregate query making query processing *condensative*. Current proposals for declarative query languages over data streams do not support such condensative processing. Nor is it yet well understood what query constructs and what semantics should be adopted for continuous query languages.

In order to make existing stream query languages more expressive, a novel stream query language CSQL (Condensative Stream Query Language) are proposed over a sequence-based stream model (Ma & Nutt 2005). It is shown that the sequence model supports a precise tuple-based semantics that is lacking in previous time-based models, and thereby provides a formal semantics to understand and reason about continuous queries. CSQL supports sliding window operators found in previous languages and processes a declarative semantics that allows one to specify and reason about the different meanings of the frequency by which a query returns answer tuples, which are beyond previous query languages over streams. In addition, a novel condensative stream algebra is defined by extending an existing stream algebra with a new frequency operator, to capture the condensative property. It is shown that a condensative stream algebra enables the generation of efficient continuous query plans, and can be used to validate query optimisation. Finally, it is shown via an experimental study that the proposed operators are effective and efficient in practice.

Keywords: Data Stream, Stream Query Language, Window Aggregation, Sequence Model, Stream Algebra, Condensative Queries.

1 Introduction

Stream data can be characterised as an unbounded list of data elements that arrive in an order and at a rate over which a Data Stream Management System (DSMS) has no control. These can be found in a large variety of applications as diverse as stock ticker processing, network traffic analysis, intrusion detection, sensor monitoring, web tracking and personalisation.

Compared to a traditional database architecture in which data is persistent and queries can be considered to be dynamic, in these new applications data can be considered active whereas queries are persistent or long-standing. As a result, traditional relational database technology designed to support applications over relatively static data has not proved entirely suitable for data stream applications.

A new research area, *data stream processing*, has generated considerable interest from both industry and research community seeking to bridge the gap between current technology and the needs of these emerged applications. This has resulted in more challenging requirements being generated for the data stream field with increasing interaction among different academic fields and a growing demand for information sharing to address many related research problems in semantics, query processing and runtime management. Among all the interesting issues, one major challenge is the development of techniques for providing continuously updating answers to aggregate queries over potentially unbounded streams. A general approach to addressing this challenge is by means of both window queries and a centralised query processing approach. Window queries add window clauses to continuous queries and allow aggregate queries to be evaluated over a segment of the input data stream rather than over the entire stream. There has been a surge of interest in the research issues involved in developing algorithms for windowed aggregate queries (Arasu & Widom 2004*b*, Chandrasekaran & Franklin 2002, Cranor, Johnson, Spataschek & Shkapenyuk 2003, Dobra, Garofalakis, Gehrke & Rastogi 2002, Dobra et al. 2002, Gilbert, Kotidis, Muthukrishnan & Strauss 2001, Li, Maier, Tufte, Papadimos & Tucker 2005).. Driven by different purposes, a number of Data Stream Management Systems (DSMSs) have been developed (Babcock, Babu, Datar, Motwani & Widom 2002, Abadi, Carney, Çetintemel, Cherniack, Convey, Lee, Stonebraker, Tatbul & Zdonik 2003, Yao & Gehrke 2003, Cranor et al. 2003, Chen, DeWitt, Tian & Wang 2000, Chandrasekaran, Cooper, Deshpande, Franklin, Hellerstein, Hong, Krishnamurthy, Madden, Raman, Reiss & Shah 2003) and several stream query languages (Arasu, Babcock, Babu, McAlister & Widom 2004, Dobra et al. 2002, Hammad, Franklin, Aref & Elmagarmid 2003, Arasu & Widom 2004*a*) have been recently proposed.

However, current techniques have two serious limitations. First, most current stream languages do not have the necessary language constructs to support *condensative* query processing over streams in the manner of traditional DBMSs (Database Management Systems). Aggregate query processing is called condensative in a traditional DBMS since the answer to an aggregate query is usually much smaller than the answer to its non-aggregate counterpart. It is crucial that a declarative stream language supports condensative query processing over distributed data streams due to the large amount of data that is usually involved and the limited storage capacity that is usually available, such that aggregation will *not* be performed over a window where queries return new results whenever the window content changes. Second, the focus of previous work has mainly been on query execution while fundamental questions in connection with data models and formal semantics for queries have not yet been thoroughly addressed. The lack of this makes it difficult to reason about aggregate queries and compare different languages within a uniform semantics. Moreover, it makes it difficult to define a clear and sensible semantics for distributed data stream processing, as is usually the case when dealing with data streams.

Prior Work. Prior work in this area has been focused on operations and system architectures for data stream processing, to augment existing technologies and build new systems for stream-based applications, e.g.,(Babcock et al. 2002, Abadi et al. 2003, Yao & Gehrke 2003, Cranor et al. 2003, Chen et al. 2000, Sullivan 1996, Chandrasekaran et al. 2003). Among the few established works on studying stream query languages, CQL (Continuous Query Language) (Arasu & Widom 2004*a*) is one of the most powerful relation-based languages. It is used in the STREAM system, and is proposed with full semantics over a time-based model. CQL provides advanced windowing capabilities, and it is even possible to PARTITION a window on an attribute and specify the width of a window (e.g. ROWS 100 or RANGE 100 MINUTES). However as the order of tuples is not uniquely defined in a time-based model, there is no clear semantics for continuous queries involving tuple-based sliding windows or moving aggregation. Another expressive language that supports condensation is StreaQuel, which is implemented in TelegraphCQ (Chandrasekaran et al. 2003). In StreaQuel, each query definition is followed by a for-loop construct that specifies (1) the set of windows over which the query is to be executed, and (2) how often the query should be run. Recently, Li et al. (Li et al. 2005) have proposed a similar, but more declarative way to define windowed aggregate queries. Their window definition has three parameters: RANGE specifies the window size, SLIDE the window movement, and WATTR the granularity, that is, whether RANGE and SLIDE are defined in terms of timestamps or sequence numbers of tuples. All these patterns were defined with respect to window identifiers. Such semantics define a function to identify uniquely each window extent for a given window aggregate query; also, they require an inverse function that, for each tuple, it determines the extents of the window to which the tuple belongs.Window identifier semantics was implemented in an extended version of the Niagara Query Engine (Chen et al. 2000) for evaluating aggregate window queries over data streams. In all three languages frequency and window length have to be defined in terms of the same granularity. Besides frequency can not exist independently without a window expression and has to be defined in a fixed place within a query due to the limited semantics to interpret different meanings of frequency, e.g., the frequency over the input stream cannot be differentiated from the frequency over the result stream.

A host of research exists on tackling aggregate query evaluation over data streams. Widely differing approaches employing, e.g., hashing, sampling, sketches and wavelets, just to name a few, have been proposed in the literature (Golab & Özsu 2003, Babcock et al. 2002, Carney, Çetintemel, Cherniack, Convey, Lee, Seidman, Stonebraker, Tatbul & Zdonik 2002, Yao & Gehrke 2003, Arasu & Widom 2004*b*, Chandrasekaran & Franklin 2002, Cranor et al. 2003, Dobra et al. 2002, Gilbert et al. 2001, Li et al. 2005). All those approaches, however, are workload-driven, as opposed to being user-driven. In more detail, it is desirable to give the user considerable freedom in how windows are defined and handled by the system. In other words, the user should be in a position to declare window semantics at the language level, rather than rely on the system to deduce tradeoffs between accuracy and resource consumption, as is usually the case. This is what where termed condensative query evaluation: the user should be able to declare how frequently sampling should take place and the system should perform aggregation based on the user's instructions.

This paper introduces a Condensative Stream Query language (CSQL) over the sequence model. CSQL aims to extend the expressiveness of stream query languages along the dimension of answer frequency, which is a live issue for continuous queries and for which no analogy exist in classical databases. It is shown that CSQL supports sliding window operators found in previous languages, and a frequency operator. It is also shown that CSQL is capable of expressing useful sample queries such as queries with user-defined sampling, mixed jumping windows, and nested aggregation, which are beyond previous query languages over streams. More specifically, the main contributions are as follows.

1. Sampling and jumping windows are incorporated in a declarative fashion into a novel Condensative Stream Query Language (CSQL), along with its formal semantics and syntax over the sequence model (Ma & Nutt 2005). It is shown that CSQL can specify window queries found in (Arasu & Widom 2004*a*, Carney et al. 2002, Chandrasekaran et al. 2003, Chen et al. 2000, Yao & Gehrke 2003); also it supports (1) a way to express frequency requirements on both base and derived streams (2) user-defined group-based sampling (3) a mixed jumping window over streams (4) various forms of aggregation, e.g., nested aggregation. None of them can be realised in previous stream query languages.

2. A *condensative stream algebra* is introduced extending the existing stream algebra with a new kind of operator, called a frequency operator as well as showing its concrete semantics. Approaches to optimisation for the condensative model such as splitting, interleaving and compositional operations are discussed. Furthermore as an independent operator, the frequency operator can be easily pushed down in a stream algebra to avoid unnecessary computation based on *local* and *non-local* semantics. This allows the possibility of splitting aggregate query processing techniques into two levels, namely, tuple sampling and aggregation evaluation, which provides a flexible mechanism to interact with different advanced aggregate operators.

3. Finally, these ideas have been implemented in a prototype query engine. In order to demonstrate the efficiency gained by a pushed down frequency operator for a jumping window query over an ordered sequence stream, it is compared with the window Id approach presented in (Li et al. 2005). The experimental results show that a pushed down frequency operator is effective and outperforms the window Id approach and that a con-

densative stream algebra can be reasonably optimised for continuous queries with a frequency-based equivalence. It is also shown how to evaluate "mixed jumping window" queries, which cannot be handled by existing approaches.

Organisation. The remainder of the paper is organised as follows. Section 2 reviews sequence databases and the time-based data model. Sections 3, 4 and 5 introduce the semantics of the CSQL language. Language syntax is presented in section 6. Algebraic rules for the operators and example CSQL queries are given in section 7 and 8, respectively. The main algorithms for CSQL are given in section 9, while experimental results are presented in section 10. Section 11 concludes the paper and discusses future work.

2 Background

In this section two previous data models are reviewed, from which the proposed model draws some features.

2.1 Sequence Database Model

The SEQ sequence model and algebra were introduced by Seshadri et al. in (Seshadri, Livny & Ramakrishnan 1995). They define a sequence as an ordering function from the integers (or another ordered domain such as calendar dates) to the items in the sequence. The SEQ model separates the data from the ordering information and can deal with different types of sequence data by supporting an expressive range of sequence queries.

Some operators, such as selection, projection, various set operations, and aggregation (including moving windows) are carried over from the relational model. A number of operators for manipulating sequences have also been developed. The SEQ model has been implemented in SRQL (Sorted Relational Query Language) (Ramakrishnan, Donjerkovic, Ranganathan, Beyer & Krishnaprasad 1998), in which sequences are implemented as logically or physically sorted multisets (relations) and the language attempts to exploit the sort order.

2.2 Time-based Stream Model

In a data stream model, data items appear in a time-varying, continuously arriving, and append-only format.

A formal time-based stream model has been defined in (Arasu & Widom 2004*a*) and a declarative Continuous Query Language (CQL), including a formal semantics, has been defined. CQL has been implemented in the *STREAM* system at Stanford. The core of the model is as follows: Let $\mathcal{D}_r$ be the set of all tuples that satisfy the schema r. Let $\mathcal{T}$ be the set of all timestamps. Then a *stream* s with schema r is a subset

$$s \subseteq \mathcal{D}_r \times \mathcal{T},$$

such that for every $\tau \in \mathcal{T}$ the bag $\{\!\{e \mid \langle e, \tau\rangle \in s\}\!\}$ is finite. $\mathcal{P}(\mathcal{D}_r)$ denotes the set of all subsets of $\mathcal{D}_r$. A *time-dependent relation* R for the schema r is a mapping:

$$R\colon \mathcal{T} \to \mathcal{P}(\mathcal{D}_r),$$

such that each set $R(\tau)$ is finite. With these definitions, a stream can be transformed into a time-varying relation and vice versa.

3 Sequence Model

In a time-based model the order of tuples is not uniquely defined. This drawback leads to ambiguous semantics for continuous queries involving tuple-based sliding windows or moving aggregation. To address this issue, features are drawn from sequence databases and is constructed a sequence dependent model for streams. In this section the model and its formal semantics are introduced. As will be seen in the next two sections, the model is more expressive than the existing time-based model in (Arasu & Widom 2004*a*). The section concludes by describing the common properties that need to be shared by different data stream models, and that the new model aims to encode. An abstract *relational stream* is required to have the following characteristics:

- A stream consists of tuples;
- A stream has a relational schema and all its tuples comply with that schema;
- A stream develops over time. Therefore it is assumed that there is a set $\mathcal{T}$ to represent the time domain, such as wall-clock time or the natural numbers. A *timestamp* is any value from $\mathcal{T}$. Timestamps are linearly ordered.

3.1 Relational Schema

A relation schema has the form:

$$r\langle a_1 \colon T_1, \ldots, a_k \colon T_k\rangle,$$

where r is a relation symbol, $a_1, \ldots, a_k$ are attributes, and $T_1, \ldots, T_k$ are types as in SQL. A timestamp is not included as a default attribute in the schema in case there is a need to separate various different timestamps associated with a tuple such as the tuple's birth time or its arrival time.

3.2 Time Domain

There is no restriction on whether the time domain has to use wall clock time or the natural numbers. However it is still required that the general properties of the time domain be defined. A time domain should be *ordered.* Let $R \subseteq X \times X$ be an ordering (i.e, R is reflexive, antisymmetric and transitive). For every ordering R the strict version R' of R is defined by

$$xR'y \text{ iff } xRy \text{ and } x \neq y.$$

A binary relation is

Linear: if for all $x, y \in X$ where $x \neq y$ either $xR'y$ or $yR'x$

Dense: if for all $x, y \in X$ where $xR'y$, there is a $z \in X$ such that $xR'z$ and $zR'y$

Discrete: if for every two elements $x, y \in X$ where $xR'y$, there are only finitely many elements z between them, i.e, there are only finitely many z such that $xR'z$ and $zR'y$

If a linear ordering R' is discrete, then for every element $x \in X$, either at least one element y is such that $xR'y$, or there is no element y such that $xR'y$. A time ordering is required to have following properties:

1. Any two distinct timestamps must be comparable. This means, the ordering should be *linear.*
2. The ordering should not be *dense*, but *discrete.*

A time domain with these properties is essentially identical with the integers or an interval of the integers. A window of length n consists of the starting point plus the next $(n-1)$ elements. If the time domain has a first element and is not bounded, then it can be represented by the natural numbers.

3.3 Local and Non-Local Semantics

A stream operator is a function Ω that takes a stream s as input and outputs a stream Ωs. Stream operators that apply to a stream can be *local* or *non-local.* Suppose there is an operator $\mathcal{Q}$ that transfers a data stream s from a sequence model to a time based model $\mathcal{Q}s$, and $\mathcal{Q}s(t)$ represents a bag of the tuples that have timestamp t.

Definition 1 Ω *is local if:*

$$\mathcal{Q}s_1(t) = \mathcal{Q}s_2(t) \textit{implies } (\Omega(\mathcal{Q}s_1))(t) = (\Omega(\mathcal{Q}s_2))(t),$$

otherwise it is non-local.

Most relational operators are local such as selection σ, and most stream operators are non-local such as sliding window operators $\mathcal{W}$.

4 Stream Operators

It is shown that queries expressible in a time-based model can also be specified in this model. Furthermore, as will be seen in section 8, the model and operators are capable of expressing queries found in practice that are beyond previous models and languages. Next, some typical stream operators are introduced in this model.

4.1 Selection Operator

First the conditions for a selection operator are defined. A *term* is either an attribute name or a value constant. An *atomic condition* is an expression of the form

$$t_1 \ \rho \ t_2,$$

where t_1, t_2 are terms and ρ is a comparison like "$<$", "$\leqslant$", "$=$", "$\geqslant$", or "$>$". Arbitrary conditions can be built up from atomic conditions using the boolean connectives "$\neg$", "$\vee$", or "$\wedge$". Conditions are denoted by the letter C.

For every condition C a selection operator σ_C is defined. Intuitively, $\sigma_C(s)$ is the subsequence of tuples with index j of stream s, where $j \in \mathbb{N}$. $\sigma_C(s)$ is defined for an arbitrary stream s recursively by saying what it is the tuple $\sigma_C(s)(j)$ for an arbitrary number j. The set of indices I_1 is defined as

$$I_1 = \Big\{ k \in \mathbb{N} \ \Big| \ s(k) \text{ satisfies } C \Big\}.$$

If $I_1 \neq \emptyset$, then let $n_1 = \min I_1$ and define $\sigma_C(s)(1) := s(n_1)$, otherwise let $\sigma_C(s) = \perp$. Now, suppose n_j is defined for some $j \in \mathbb{N}$. Then let

$$I_{j+1} = \Big\{ k \in \mathbb{N} \ \Big| \ s(k) \text{ satisfies } C \text{ and } k > n_j \Big\}.$$

Again, if $I_{j+1} \neq \emptyset$, then let $n_{j+1} = \min I_{j+1}$ and define $\sigma_C(s)(j{+}1) := s(n_{j+1})$, otherwise let $\sigma_C(s)(j+1)$ be undefined. Also, $\sigma_C(s)(j+1)$ is undefined if n_j is undefined.

4.2 Sliding Window Operators

W_t will be used to denote a time-based window, and W_n will be used to denote a tuple-based sliding window.

Time-based Sliding Window

A time-based sliding window W_t is bounded by its temporal size t even though it is not known exactly how many tuples there are within the window size. However it slides whenever the time slot increases. The sliding rate will depend on the time granularity. $s^\tau(k)$ is introduced to denote the timestamp for tuple $s(k)$. More formally, the output stream $W_t s$ is defined as a sequence of sets $W_t s(j)$ for a given j in stream s. $W_t s(j)$ is not defined if $s(j)$ is not defined, otherwise

$$W_t s(j) = \Big\{ s(k) \ \Big| \ s^\tau(k) + t \geqslant s^\tau(j) \text{ and } k \leqslant j \Big\}.$$

Tuple-based Sliding Window

A tuple-based sliding window will slide whenever a new tuple arrives. So, for every $n \in \mathbb{N}$, a tuple-based sliding window W_n over stream s produces a sequence of sets

$$W_n s(j) = \Big\{ s(k) \ \Big| \ k \geqslant \max\{0, j-n\} \text{ and } k \leqslant j \Big\}.$$

4.3 Frequency Operator

The frequency operator F will pick the stream tuple based on a defined frequency. Depending on how the frequency is set, different types of frequency operators can be defined. Basically, the parameters can be set either by a physical bound (tuple-based) or a logical bound (time-based). In order to separate the different bounds, F_n and F_t are used to denote a tuple-based frequency operator and a time-based frequency operator respectively.

Tuple-based Frequency Operator

For every natural number $n \in \mathbb{N}$ a tuple-based frequency operator F_n selects every n-th tuple of a stream. Formally:

$$F_n s(j) = s(n \times j).$$

Time-based Frequency Operator

For every time instance t a time-based frequency operator F_t selects tuples with timestamp $j \times t$ as a stream $F_t s$, where $j \in \mathbb{N}$.

If there is no tuple with timestamp $j \times t$, then the last tuple is output within that time slot. $F_t s(j)$ is a subsequence of tuples with order j over order n_j of stream s, where $j \in \mathbb{N}$. Then if $s(n_j) \neq \emptyset$ let

$$n_j = \ \max \Big\{ k \in \mathbb{N} \ \Big| \ (j-1) \times t \leqslant s^\tau(k) \leqslant j \times t \Big\},$$

otherwise it is undefined. Now, $F_t s(j) = s(n_j)$, for all $j \in \mathbb{N}$ if n_j is defined, otherwise $F_t s(j) = \perp$.

4.4 Jumping Windows

Sometimes, the window needs to jump rather than slide. This can be achieved by applying a frequency operator to a sequence of sets instead of posing a frequency to a sequence of tuples. Such a kind of window is called a jumping window. Depending on how the frequency length is defined. Jumping windows are categarised into two different types: tuple-based jumping windows and time-based jumping windows.

Tuple-based Jumping Window

For every number $n \in \mathbb{N}$, and a sequence of sets Ws that are produced by the sliding window operators W_n or W_t, a tuple-based jumping window $F_n(Ws)$ selects every n-th set of Ws as follows:

$$(F_n(Ws))(j) = Ws(n \times j).$$

Time-based Jumping Window

For a sequence of sets Ws that are produced by a sliding window operator W applied to the stream s, A time-based jumping window $F_t(Ws)$ is obtained by selecting a subsequence $Ws(n_j)$ ($j \in \mathbb{N}$) of $Ws(n)$. Intuitively, $F_t(Ws)(j)$ is the first window that contains an element with a timestamp that is greater or equal to $t \times j$. Formally, for an arbitrary stream s and a window operator W, $F_t(Ws)$ is defined recursively by saying what it is the set $F_t(Ws)(j)$ for an arbitrary number j. The set of indices I_j is defined as

$$I_j = \Big\{ i \in \mathbb{N} \;\Big|\; \exists k.\, s(k) \in Ws(i) \wedge s^{\tau}(k) \geq j \times t \Big\}.$$

If $I_j \neq \emptyset$, then $n_j := \min I_j$, and $F_t(Ws)(j) := Ws(n_j)$, otherwise let $F_t(Ws)(j) = \perp$.

Time-based Jumping Operator

For every time instance t, and a stream Ws that is produced by applying a sliding window operator W to a stream s. A time-based jumping window $F_t(Ws)$ is obtained by selecting a bag of tuples for every multiple of t from stream Ws. $(F_t(Ws))(\tau)$ is defined for an arbitrary time t, where $t \in \mathcal{T}$ as:

$$(F_t(Ws))(\tau) = \begin{cases} Ws(\tau) & \text{if } \tau = j \times t \text{ for some } j \in \mathbb{N} \\ \emptyset & \text{otherwise} \end{cases}$$

5 Condensative Stream Queries

A condensative stream query Q, in essence, is a traditional SPJ query augmented with frequency predicates (not necessarily have to be an aggregate query). Conceptually such queries have the "canonical" form of Eq. 1 in terms of relational algebra:

$$Q = \pi_* \mathcal{F}_{(p_1,\ldots,p_n)} \sigma_{\mathcal{B}(c_1,\ldots,c_m)} (R_1 \times \ldots \times R_h) \qquad (1)$$

That is, upon the product of the base relations ($R_1 \times \ldots \times R_h$), two types of operations performed with projected attributes (as indicated) are returned as the results.

Filtering: a *Boolean function* $\sigma_{\mathcal{B}(c_1,\ldots,c_m)}$ filters the results by the *selection* operator $\sigma_{\mathcal{B}}$ *(e.g.,* $\mathcal{B} = c_1 \wedge c_2 \wedge c_3$), and

Frequency: a *Frequency function* $\mathcal{F}(p_1, \ldots, p_n)$ picks up the results from the base relations.

The goal is to support such condensative stream queries efficiently. Condensative stream models boolean filtering, *i.e.,* $\sigma_{\mathcal{B}(c_1,\ldots,c_m)}$ as a *first-class* construct in query processing. With algebraic support for optimisation, Boolean filtering is virtually never processed in the canonical form (of Eq. 1). Consider, for instance, $\mathcal{B} = c_1 \wedge c_2$ for c_1 as a selection over R and c_2 as a join condition over $R \times S$. The algebra framework supports *splitting* of selections (*e.g.,* $\sigma_{c_1 \wedge c_2}(R \times S) \equiv \sigma_{c_1}\sigma_{c_2}(R \times S) \equiv \sigma_{c_1}(R \bowtie_{c_2} S)$) and *interleaving* them with other operators (*e.g.,* $\sigma_{c_1}(R \bowtie_{c_2} S) \equiv \sigma_{c_1}(r) \bowtie_{c_2} S$). Their algebraic equivalences enable query optimisation to transform the canonical form into efficient query plans by splitting and interleaving.

Such algebraic support, *splitting* and *interleaving* for optimisation, are completely inherited for frequency, *i.e.,* $\mathcal{F}(p_1, \ldots, p_2)$. Moreover, the support can be *compositional.* Suppose there is a frequency function $\mathcal{F} = p_1 \wedge p_2 \wedge p_3$, for p_1, p_2, p_3 as a frequency condition over R_1, R_2, R_3 respectively. p_1, p_2, p_3 are either all time-based or all tuple-based. Suppose $p_3 \mathsf{mod} p_2 = 0, p_3 \mathsf{mod} p_1 = 0, p_2 \mathsf{mod} p_1 = 0$, then the frequency functions are *compositional* (*e.g.,* $\mathcal{F}_{p_1}\mathcal{F}_{p_2}\mathcal{F}_{p_3}(R_1 \times R_2 \times R_3) \equiv \mathcal{F}_{p_1}\mathcal{F}_{p_2}\mathcal{F}_{p_2}(R_1 \times R_2 \times R_3) \equiv \mathcal{F}_{p_1}\mathcal{F}_{p_1}\mathcal{F}_{p_1}(R_1 \times R_2 \times R_3) \equiv \mathcal{F}_{p_1}(R_1) \times (R_2 \times_{\mathcal{F}_{p_1}} R_3)$. When queries are nested, frequency functions can be compositional even when the frequencies involved do not have the same granularity. (*e.g,* for a self-join query $(R_1 \times_{\mathcal{F}_{p_1}} R_2) \times_{\mathcal{F}_{p_2}} (R_1 \times_{\mathcal{F}_{p_1}} R_2)$, the inner frequency condition p_1 has the priority to synchronise the outer frequency condition p_2, $(R_1 \times_{\mathcal{F}_{p_1}} R_2) \times_{\mathcal{F}_{p_2}} (R_1 \times_{\mathcal{F}_{p_1}} R_2) \equiv (R_1 \times_{\mathcal{F}_{p_1}} R_2) \times_{\mathcal{F}_{p_1}} (R_1 \times_{\mathcal{F}_{p_1}} R_2)$).

Finally relational algebra's *pushing down* optimisation is extended into a stream algebra. The operators which can be used in a stream algebra are categorised by local and non-local semantics defined in definition 1 as such semantics assist the pushing down optimisation approach. An operator can be easily pushed down if it is a local operator such as a time-based frequency operator or a selection operator, otherwise not, *i.e.,* suppose there is a time based frequency function $\mathcal{F}$, then $\mathcal{F}(R_1 \times R_2 \times R_3) \equiv \mathcal{F}(R_1) \times \mathcal{F}(R_2) \times \mathcal{F}(R_3)$.

6 Syntax of CSQL

CSQL is a stream language that adds additional language patterns to SQL to support a stream processing ability. The core syntax of CSQL can be described with a context-free grammar.

```
string: represents for any valid string
number: represents any valid number
asterisk: represents *
<Query>⟶<Select><From> | <Select><From><Where> |
          <Select><From><Where><GroupBy>
<Name>⟶string | <Name>.<Name> |asterisk|
          <Name>AS<Name>
<Attribute List>⟶<Name> | <Name>(,<Name>)*
<Granularity>⟶Milliseconds|Seconds|Minutes|Hours|Tuples|
               Millisecond|Second|Minute|Hour|Tuple
<Length>⟶number
<Frequency>⟶[<Fre>Partitioned By <Attribute List>]|
             [<Fre>]
<Fre>⟶Frequency<Length><Granularity>
<Range>⟶Range<Length><Granularity>
<Compare>⟶> | < | >= | <= | = | <>
<Clause>⟶<Name><Compare><Name> |
          <Name><Compare>number
<op>⟶ and|or
<Condition>⟶<Clause> | <Clause> (<op><Clause>)*
<term>⟶COUNT|SUM|AVG|MAX|MIN
<Aggregation>⟶<term> (<Name>)|
               <Aggregation> (,<Aggregation>)*
<Select>⟶ SELECT <SelectTerm> |<Select><Frequency>
<SelectTerm>⟶<Aggregation> | <Attribute List> |
              <SelectTerm> (,<SelectTerm>)*
<From>⟶ FROM <FromTerm>
<LeftBracket>⟶(
<RightBracket>⟶)
<FromTerm>⟶<Name> | <Name><Frequency> |
            <Name> [<Range>]|
            <Name> [<Frequency>, <Range>]|
            <LeftBracket><Query><RightBracket>AS
            <FromTerm> |
            <FromTerm> (,<FromTerm>)*
<Where>⟶ WHERE <Condition>
<GroupBy>⟶GROUP BY <Attribute List>
```

7 Example Scenario and Queries

In practice there has been an increasing need for aggregate queries. To illustrate this, consider a scenario of a tracing system to study the behavior of wild animals, which collects distributed sensor measurements. One of the sensors records the pulse of an animal. Upon every heart beat of an animal it will send out a tuple with a timestamp and the animal's ID. The schema of the relation for these measurements has the form

$$\mathsf{Pulse}\langle\mathsf{Id}, \mathsf{Timestamp}\rangle$$

The other type of sensors report on an animal's blood pressure and body temperature regularly, for example every (full) second. It has a core relation

$$\mathsf{BodyCondition}\langle\mathsf{Id}, \mathsf{Species}, \mathsf{BTemp}, \mathsf{BloodP}, \mathsf{Timestamp}\rangle$$

In these two relations: Id is the unique number of each animal, Species represents the type of animal, Timestamp represents the timestamp, BloodP is the blood pressure of the animal, and BTemp is the animal's temperature. For ease of presentation, it is assumed that tuples arrive in the order of their timestamp attribute. Here are four queries with requirements on how often to evaluate them.

1. **Simple sampling query:** For every 100 tuples, report all horses' body condition records.
2. **Latest result query:** Report the latest results of measurement on blood pressure and body temperature for each animal at the rate of one reading every minute, and evaluate the query for every 100 arriving tuples.
3. **Aggregate query:** For each animal, what is the pulse rate per minute? It is supposed that the user wants to know the result for every 10 tuples.
4. **Nested aggregate query:** For each animal, what is the average pulse rate per hour? It is supposed that the user wants to make use of the answers to the first query and expects a result every minute.

8 CSQL vs. Condensative Stream Algebra

The Condensative Stream Query Language (CSQL) is similar to SQL but extended with operators such as those discussed in Section 4, as well as a mechanism for directly submitting plans in a stream algebra that underlies the language.

In the CSQL language, a frequency operator can be expressed by adding to a range variable of a stream, say S, the expression [Frequency F], where F denotes an interval length. The length is either defined in terms of a number of tuples as [Frequency n Tuples] ("every n tuples") or in terms of a time period, e.g. as [Frequency t Minutes] ("every t minutes"). The operator picks tuples based on the predefined frequency length from the base stream. For group-based sampling [Frequency F Partitioned By $A_1, \ldots, A_k$] is used. The operator will partition S into different substreams based on the grouping attributes $A_1, \ldots, A_k$, then for each substream the operator picks tuples based on the predefined frequency. The frequency is separated for an input stream and a result stream by putting the frequency expression in either the FROM clause or the SELECT clause.

The frequency operator can be combined with sliding window operators when the window needs to move much faster. Such a kind of window is called a *jumping window*. When Frequency = 1 Tuple, it is equivalent to a normal sliding window. Depending on how the frequency length is defined, *tuple-based* and *time-based* jumping windows can be distinguished. Instead of computing the answer whenever a new tuple arrives, the frequency operator requires a computation only after an interval of the frequency length. This means that, the operator will take a "nap" between any two computations. A jumping window has two parameters:

The window size W. All of the tuples that arrive from the start during a period of W, or within W tuples have to be stored for a computation.

The length of the "nap" F. A new window is only output after the nap is over.

A jumping window is always defined by a sliding window operator, followed by a frequency operator expression, such as [Range W, Frequency F]. The semantics supports the mixing of tuple-based frequencies with time-based window bounds and vice versa. Such windows are called "mixed jumping windows".

To enable frequency-based query processing and optimisation, the relational algebra (Kießling 2002) is extended into a stream algebra (Babcock et al. 2002) by substituting relations for streams, where the operators, and algebraic laws "respect" and take advantage of the "compositional" property introduced in section 5. Such a stream algebra is extended by adding the new operator frequency operator $\mathcal{F}$ and the sliding window operator $\mathcal{W}$, and so generalise the existing relational operators (e.g., $\pi, \mathcal{A}, \sigma, \mathcal{G}$ in figure 1) to be "frequency-based". CSQL is shown to be declarative by expressing the frequency in different places within a query. It is also shown how those differences affect the stream algebra in figure 1 and 2 respectively. Consider the first query.

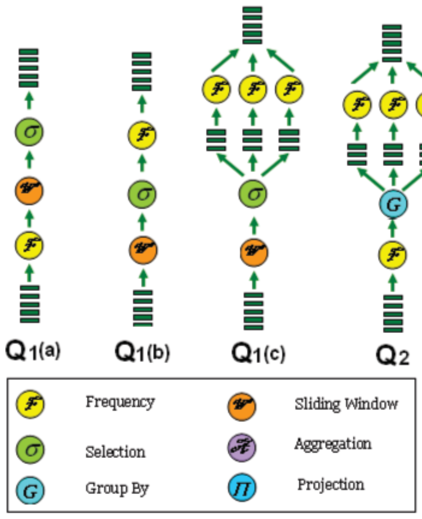


Figure 1: Stream Algebra for Example Queries 1 & 2

Query 1

"For every 100 tuples, report all horses' body condition records.". The query can be interpreted as for every 100 tuples over the base stream:

$Q_1(a)$:

```
SELECT   *
FROM     BodyCondition AS B
         [Range 1 Minute, Frequency 100 Tuples]
WHERE    B.Species = 'horse'
```

It can also be understood as for every 100 tuples over an answer stream (a stream full of animal's pulse rate per minute).

$Q_1(b)$:

```
SELECT   * [Frequency 100 Tuples]
FROM     BodyCondition AS B [Range 1 Minute]
WHERE    B.Species = 'horse'
```

Thirdly it can be understood as for every 100 tuples over a substream w.r.t to each animal.

$Q_1(c)$:

```
SELECT   *
         [Frequency 100 Tuples Partitioned By B.Id ]
FROM     BodyCondition B [Range 1 Minute]
WHERE    B.Species = 'horse'
```

This shows how CSQL supports different meanings of frequency. Figure 1 and 2 give more intuitive interpretation for these three queries using stream algebra.

Query 2

Report the latest results of measurement on blood pressure and body temperature for each animal at the rate of one reading every minute, and evaluate the query for every 100 arriving tuples.

```
SELECT   * [Frequency 1 Minute Partitioned By B.Id]
FROM     BodyCondition B
         [Frequency 100 Tuples Partitioned By B.Id]
```

This query will take all the last 100 tuples, and then group the stream into substreams with equality of grouping attribute "Id". Finally it returns the relation that contains all the latest results from each substream within a minute.

Query 3

"For each animal, what is the pulse rate per minute?", and supposing the user wants to know the result for every 10 tuples.

```
SELECT     P.Id, COUNT(*)
           [Frequency 10 Tuples Partitioned By P.Id]
FROM       Pulse P
           [Range 1 Minute, Frequency 1 Tuple]
GROUP BY   P.Id
```

This mixed jumping window query will evaluate the query with sliding semantics Frequency = 1 Tuple. It will count the last minute's worth of Pulse tuples for each animal after receiving every incoming tuple. It can be easily optimised by using the default jumping semantics Range = Frequency without losing the precision of the result. Then the frequency operator will evaluate the relational query over the window, at the end of the nap period. The frequency operator sitting in the SELECT clause will sample the resulting substream for each animal picking one tuple out of every ten tuples.

Query 4

"For each animal, what is the average pulse rate per hour?", and supposing that the user wants to make use of the answers to the first query and expects a result every minute.

```
SELECT   PR.Id, AVG(PR.Rate)
         [Frequency 1 Minute Partitioned By PR.Id]
FROM     (SELECT   P.Id, COUNT(*)
         [Frequency 10 Tuples Partitioned By P.Id]
         AS Rate
          FROM     Pulse P [Range 1 Minute,
                            Frequency 1 Tuple]
          GROUP BY   P.Id )
         AS PulseRate PR
         [Range 1 Hour, Frequency 1 Tuple]
GROUP BY   PR.Id
```

This query contains two nested frequencies, but only the outer query determines how often the inner query is evaluated. The inner query or a similar query can also be registered with a frequency that is an integer fraction of 10 tuples as an independent view, then this existing inner query can be used to answer a new query.

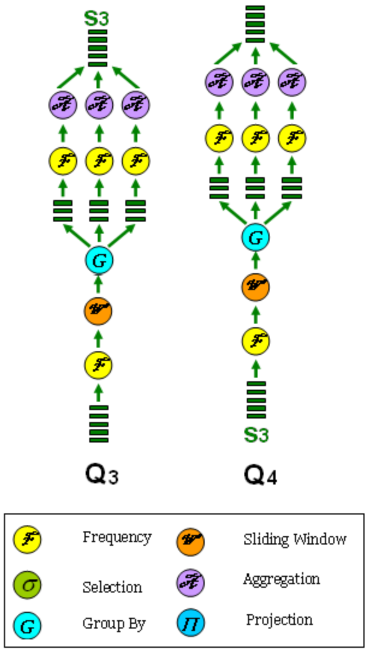


Figure 2: Stream Algebra for Example Queries 3 & 4

9 Frequency Algorithms

Some aggregation algorithms are provided from the implementation for the CSQL language below. The algorithms are categorised based on the frequency declaration. The main optimisation in the algorithm is to find the right window for each tuple (one tuple can belong to more than one window). As the window can be constructed incrementally within a sequence model, any time-based declaration (window or frequency) may lead to the next tuple *jump*ing over some empty windows. A variable *jumpto* is therefore defined to calculate the starting bound of a target window. Time-based declarations also require two pointers to mark the current insertion and deletion points. Based on different updating requirements (lazy or eager), the old tuple can be either deleted whenever the new tuple is inserted or the old tuple deleted when the timestamp changes. Note that a different buffer is used for different window, e.g., a circular buffer with fixed size w_n when the window length is in sequence number and a linked list when the window length is in the time range w_t.

Algorithm 1 Non-grouping Aggregation with Time-based Frequency

```
Require: frequency length is in time range, non-grouping aggregate query
  Input: tuple, query starting time $q$, current loop index $i$, frequency length $f_t$, window length $w_t$
  Output: answer over windowed stream $W_s$
  if $w_t$ is in time range & $w_t \leq f_t$ then
    $jumpto = f_t * i - w_t + q + 1$;
    {$jumpto$: sliding window bound}
    while $t_\tau \geq jumpto + w_t$ do
      i++;
      $jumpto = f_t * i - w_t + q + 1$;
      {$t_\tau$: tuple's timestamp}
    end while
    while $t_\tau < jumpto$ do
      discard current tuple;
      get next tuple;
    end while
    while $t_\tau < jumpto + w_t$ do
      insert tuple into the buffer;
      get next tuple;
    end while
    perform moving aggregation over the buffer;
    i++;
  else if $w_t$ is in time range & $w_t > f_t$ then
    while $t_\tau > f_t * i + q$ do
      i++;
      $jumpto = f_t * i - w_t + q + 1$;
    end while
    while $t_\tau \leq f_t * i + q$ & $t_\tau \geq f_t * i - w_t + q$ do
      insert tuple into the buffer;
      $\tau_O = \tau_{head}$;
      {$\tau_{head}$: timestamp of first tuple in buffer}
      $\tau_N = \tau_{tail}$;
      {$\tau_{tail}$: timestamp of last tuple in buffer}
      get next tuple;
    end while
    perform aggregation over the buffer;
    update the buffer by condition: $\tau_N - \tau_O \leq w_t$;
    i++;
  end if
```

Algorithm 2 Non-grouping Aggregation with Mixed-jumping Window

```
Require: frequency length is in time range, non-grouping aggregate query, $w_t$ is in sequence number
  Input: tuple, query starting time $q$, current loop index $i$, frequency length $f_t$, window length $w_t$
  Output: answer over windowed stream $W_s$
  while $t_\tau > f_t * i + q$ do
    i++;
  end while
  while $t_\tau \leq f_t * i + q$ do
    insert tuple into the buffer;
    get next tuple;
  end while
  perform moving aggregation over the buffer;
  i++;
```

Algorithm 3 Aggregation with Partitioned Tuple-based Frequency

```
Require: partitioned tuple-based frequency
  Input: tuple, query starting time $q$, current inner loop index $i$ for each group, frequency length $f_n$, window length $w_n$ or $w_t$
  Output: answer over windowed stream $W$
  if window length is in sequence number $w_n$ & $w_n \leq f_n$ then
    $jumpto = f_n * i - w_n + 1$;
    while $t_n \leq jumpto$ do
      discard current tuple;
    end while
    while $t_n \leq f_n * i$ do
      insert tuple into the buffer;
      get next tuple;
    end while
    perform aggregation over the buffer;
    i++
  else if window length is in sequence number $w_n$ & $w_n \geq f_n$ then
    while $t_n \leq f_n * i$ do
      insert tuple into the buffer;
      get next tuple;
    end while
    perform aggregation over the buffer;
    i++;
  else if window length is in time range $w_t$ then
    while $t_n \leq f_n * i$ do
      insert tuple into the buffer;
      $\tau_O = \tau_{head}$;
      $\tau_N = \tau_{tail}$;
      update the buffer by condition: $\tau_N - \tau_O \leq w_t$;
      get next tuple;
    end while
    perform aggregation over the buffer;
    i++;
  end if
```

10 Experimental Study

To evaluate the effectiveness of proposed semantics, the conceptual operators were implemented in a prototype query engine and a preliminary experimental study is conducted. The framework was implemented in Java and experiments were executed on a Pentium IV 2.4Ghz with 512M of RAM. Wall clock timings are used and execution time is calculated by measuring the average cost of 10000 answer tuples. Streaming behavior was simulated by using a pull-based execution model: the more effective the algorithm, the more tuples it is able to process. Since a frequency operator typically spends its time sampling and aggregating, there is a clear division of work. The interest is in showing how it is possible to optimize the sampling cost in such an environment, as the aim is to treat the efficiency of the aggregation algorithm as an orthogonal issue. Therefore, the same aggregation was used in all experiments, and the execution time is calculated as the sum of scanning the input stream and producing the aggregate. As a result, any performance gain observed will be due to the efficiency of the sampling methodology, which is directly tied to how well the semantics of the operators can be implemented.

Experiments are divided into two parts. Firstly, the performance of the pushed down frequency operators are evaluated in contrast to the window identifier approach for a tuple-based jumping window (`AVG`) query. Note that one tuple may be in the contents of multiple windows.

The efficiency of evaluating queries without or with a `GROUP BY`-clause is shown in figures 3 and 4 re-

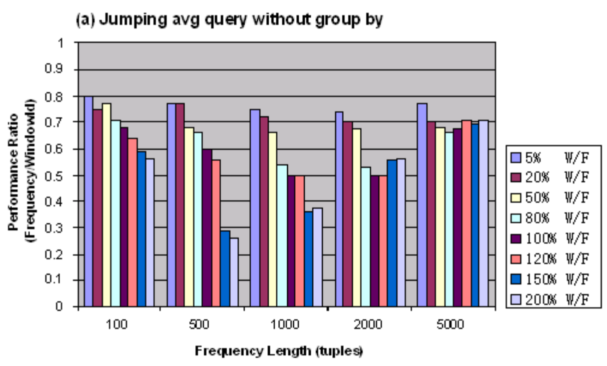


Figure 3: Cost Ratio of Frequency vs. Window Id Approach (a).

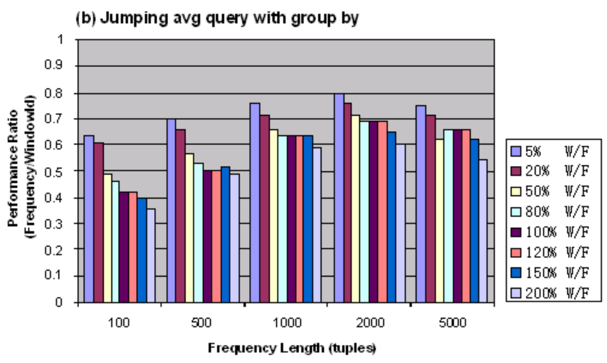


Figure 4: Cost Ratio of Frequency vs. Window Id Approach (b).

spectively. The horizontal axis is the frequency length measured in tuples. The vertical axis is the execution time using a pushed down frequency operator expressed as a fraction of the execution time of the window identifier approach. The window length is represented as a percentage of the frequency length. For example a 30% W/F ration for a frequency length F of 1000 tuples will evaluate the query over a window length bounded by 300 tuples. As an independent operator, the frequency operator can be easily pushed down in a query plan to avoid unnecessary computation. This allows aggregate query processing to be split in two levels: (1) tuple sampling, and (2) aggregation evaluation; this modelling provides a flexible mechanism to interact with different advanced aggregate operators. Our experiment showed that pushing down the frequency operator is an effective technique and it significantly outperforms the window identifier approach. Secondly, the efficiency of processing

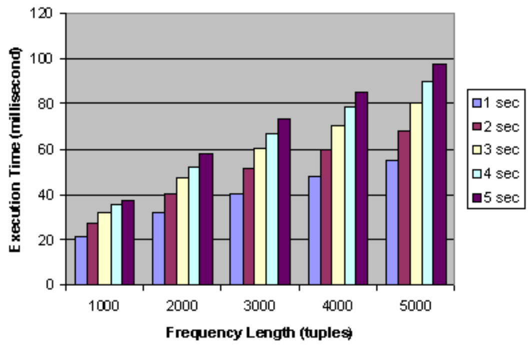


Figure 5: Performance of Mixed Jumping Window.

mixed jumping window queries was evaluated. That cannot be handled by existing approaches. Figure 5 shows the upper bound performance time of a mixed jumping window (AVG) query that has a frequency length specified on a tuple basis and a window length specified on a time basis. The horizontal axis is the frequency length (measured in tuples) where the window length is measured in seconds. The vertical axis is the total execution time (per answer tuple appearing in the average) measured in milliseconds. Note that performance can be further improved if a more efficient aggregation algorithm is employed.

11 Conclusion

This paper has studied stream queries from a theoretical angle. More specifically, it has incorporated sampling and jumping in a declarative fashion in a query language, CSQL. Furthermore a formal semantics has been introduced on both new data model and along with a novel frequency operator for extending stream query languages with more expressibility, allowing e.g., for user-defined sampling and condensative query processing.

In the future there is the prospect of handling more complex queries, which require the automatic construction of distributed query plans, based on techniques for answering continuous queries using continuous views. A key operation in creating such plans is to determine whether a query is contained in another query. While this problem has been thoroughly investigated for queries over static databases, it is still open for continuous queries. The algebraic perspective of the model is also an interesting issue to be investigated in the future by defining extensions of relational operators to handle data stream constructs, and to study further the resulting "stream algebra" and other properties of these extensions.

Furthermore, little work considers data stream processing over different forms of stream data such as tuple-like data or token-like data. Though a data stream is commonly recognised as continuously arriving data usually at a high rate, an individual data stream item can appear in various forms such as tuple-like data (relational tuples or instances of objects) or token-like data (XML files), or numerical data (treated as a special case of tuple-like data). In relation-based models (e.g. STREAM (Babcock et al. 2002)), items are transient tuples stored in virtual relations, possibly horizontally partitioned across remote nodes. In object-based models (e.g. COUGAR (Yao & Gehrke 2003) and Tribeca (Sullivan 1996)), sources and item types are modelled as (hierarchical) data types with associated methods. Semi-structured data models for data streams have just recently been introduced (Koch, Scherzinger, Schweikardt & Stegmaier 2004, Florescu, Hillery, Kossmann, Lucas, Riccardi, Westmann, Carey & Sundararajan 2004). In an XML context, a "tuple" is often specifically referred to as a set of cells, where each cell contains a set of XML nodes (e.g. XML trees). Each data stream item is a token such as a start tag, an end tag or a PCDATA item. The work published here relates well to previous work in selectively extracting data for XML files (Fan & Ma 2006), which can lead to a promising bridge effort between querying relational data streams and XML streams.

Finally, as CSQL is a declarative stream language which is able to describe how to transform streams into smaller result streams, it will be useful for queries in large distributed applications. Such a foundation is surely key to develop a general-purpose well-understood distributed query processor for distributed data streams.

References

Abadi, D. J., Carney, D., Çetintemel, U., Cherniack, M., Convey, C., Lee, S., Stonebraker, M., Tatbul, N. & Zdonik, S. B. (2003), 'Aurora: a new model and architecture for data stream management.', *VLDB Journal* **12**(2), 120–139.

Arasu, A., Babcock, B., Babu, S., McAlister, J. & Widom, J. (2004), 'Characterizing memory requirements for queries over continuous data streams.', *ACM Transactions on Database Systems* **29**(1), 162–194. ACM Press.

Arasu, A. & Widom, J. (2004*a*), 'A denotational semantics for continuous queries over streams and relations.', *SIGMOD Record* **33**(3), 6–11. ACM Press.

Arasu, A. & Widom, J. (2004*b*), Resource sharing in continuous sliding-window aggregates., *in* 'Proceedings of International Conference on Very Large Databases (VLDB)', Morgan Kaufmann, pp. 336–347.

Babcock, B., Babu, S., Datar, M., Motwani, R. & Widom, J. (2002), Models and issues in data stream systems., *in* 'Proceedings of ACM Symposium on Principles of Database Systems (PODS)', ACM Press, pp. 1–16.

Carney, D., Çetintemel, U., Cherniack, M., Convey, C., Lee, S., Seidman, G., Stonebraker, M., Tatbul, N. & Zdonik, S. B. (2002), Monitoring streams - a new class of data management applications., *in* 'Proceedings of International Conference on Very Large Databases (VLDB)', Morgan Kaufmann, pp. 215–226.

Chandrasekaran, S., Cooper, O., Deshpande, A., Franklin, M. J., Hellerstein, J. M., Hong, W., Krishnamurthy, S., Madden, S., Raman, V., Reiss, F. & Shah, M. A. (2003), TelegraphCQ: Continuous dataflow processing for an uncertain world., *in* 'Proceedings of the Biennial Conference on Innovative Data Systems Research (CIDR)', ACM Press, pp. 269–280.

Chandrasekaran, S. & Franklin, M. J. (2002), Streaming queries over streaming data., *in* 'Proceedings of International Conference on Very Large Databases (VLDB)', Morgan Kaufmann, pp. 203–214.

Chen, J., DeWitt, D., Tian, F. & Wang, Y. (2000), NiagaraCQ: A scalable continuous query system for internet databases., *in* 'Proceedings of ACM SIGMOD Conference on Management of Data (SIGMOD)', ACM Press, pp. 379–390.

Cranor, C., Johnson, T., Spataschek, O. & Shkapenyuk, V. (2003), Gigascope: A stream database for network applications., *in* 'Proceedings of ACM SIGMOD Conference on Management of Data (SIGMOD)', ACM Press, pp. 647–651.

Dobra, A., Garofalakis, M. N., Gehrke, J. & Rastogi, R. (2002), Processing complex aggregate queries over data streams., *in* 'Proceedings of ACM SIGMOD Conference on Management of Data (SIGMOD)', ACM Press, pp. 61–72.

Fan, W. & Ma, L. (2006), Selectively storing XML data in relations., *in* 'Proceedings of International Conference on Database and Expert Systems Applications (DEXA)', Lecture Notes in Computer Science, Springer Verlag, pp. 518–526.

Florescu, D., Hillery, C., Kossmann, D., Lucas, P., Riccardi, F., Westmann, T., Carey, M. J. & Sundararajan, A. (2004), 'The BEA streaming xquery processor.', *The VLDB Journal* **13**(3), 294–315.

Gilbert, A. C., Kotidis, Y., Muthukrishnan, S. & Strauss, M. (2001), Surfing wavelets on streams: One-pass summaries for approximate aggregate queries., *in* 'Proceedings of International Conference on Very Large Databases (VLDB)', Morgan Kaufmann, pp. 79–88.

Golab, L. & Özsu, M. T. (2003), 'Issues in data stream management.', *SIGMOD Record (ACM Special Interest Group on Management of Data)* **32**(2), 5–14. ACM Press.

Hammad, M. A., Franklin, M. J., Aref & Elmagarmid, A. K. (2003), Scheduling for shared window joins over data streams., *in* 'Proceedings of International Conference on Very Large Databases (VLDB)', Morgan Kaufmann, pp. 297–308.

Kießling, W. (2002), Foundations of preferences in database systems., *in* 'Proceedings of International Conference on Very Large Databases (VLDB)', Morgan Kaufmann, pp. 311–322.

Koch, C., Scherzinger, S., Schweikardt, N. & Stegmaier, B. (2004), FluXQuery: An optimizing XQuery processor for streaming XML data., *in* 'Proceedings of International Conference on Very Large Databases (VLDB)', Morgan Kaufmann, pp. 1309–1312.

Li, J., Maier, D., Tufte, K., Papadimos, V. & Tucker, P. A. (2005), Semantics and evaluation techniques for window aggregates in data streams., *in* 'Proceedings of ACM SIGMOD Conference on Management of Data (SIGMOD)', ACM Press, pp. 311–322.

Ma, L. & Nutt, W. (2005), Semantics of stream operators for condensatively querying data streams., *in* 'IEEE International Conference on e-Business Engineering (ICEBE)', IEEE Computer Society, pp. 518–526.

Ramakrishnan, R., Donjerkovic, D., Ranganathan, A., Beyer, K. S. & Krishnaprasad, M. (1998), SRQL: Sorted relational query language., *in* 'Proceedings of International Conference on Scientific and Statistical Database Management (SSDBM)', IEEE Computer Society, pp. 84–95.

Seshadri, P., Livny, M. & Ramakrishnan, R. (1995), SEQ: A model for sequence databases., *in* 'Proceedings of IEEE International Conference on Data Engineering (ICDE)', IEEE Computer Society, pp. 232–239.

Sullivan, M. (1996), Tribeca: A stream database manager for network traffic analysis., *in* 'Proceedings of International Conference on Very Large Databases (VLDB)', Morgan Kaufmann, p. 594.

Yao, Y. & Gehrke, J. (2003), Query processing in sensor networks., *in* 'Proceedings of the Biennial Conference on Innovative Data Systems Research (CIDR)', ACM Press, pp. 233–244.

Incremental Mining for Temporal Association Rules for Crime Pattern Discoveries

Vincent Ng*, Stephen Chan, Derek Lau, Cheung Man Ying

Department of Computing
The Hong Kong Polytechnic University,
Hong Kong, China

cstyng@comp.polyu.edu.hk

Abstract

In recent years, the concept of temporal association rule (TAR) has been introduced in order to solve the problem on handling time series by including time expressions into association rules. In real life situations, temporal databases are often appended or updated. Rescanning the complete database every time is impractical while existing incremental mining techniques cannot deal with temporal association rules. In this paper, we propose an incremental algorithm for maintaining temporal association rules with numerical attributes by using the negative border method. The new algorithm has been implemented to support the discoveries of crime patterns in a district of Hong Kong. We have also experimented with another real life database of courier records of a shipping company. The preliminary results show a significant improvement over rerunning TAR algorithm..

Keywords: Temporal Association Rules, Incremental Mining, Crime Analysis.

1 Introduction

Association rule plays an important role in data mining and has been becoming applicable in many areas. Since the pioneering works of Agrawal, Imielinski and Swami (1993) and Agrawal and Srikant (1994), it has led to many proposals of mining of association rules such as fast mining approaches, updating approaches, and various formations of rule patterns such as temporal patterns discussed in Wang, Yang and Muntz (1999, 2001). In general, most transactional database systems accumulate small size of incremental datasets and are appended into the main databases regularly. In such situations, incremental mining becomes a necessity. Incremental mining algorithms (Cheung, Han, Ng, Fu and Fu 1996, Cheung, Han, Ng, Wong 1996, Ayad, El-Makky and Taha 2001, Cheng, Yan, and Han 2004, Ayan, Tansel and Arkun 1999, Chang, Yang 2003, Cheung, Lee and Kao 1997) have been developed and are mostly focused on minimizing the number of database rescanning.

Currently there is no updating algorithm available for temporal association rules. In our work, we have developed a new algorithm call *ITAR* for incremental mining of temporal association rules by employing the skeleton of *ICAP* (Ayad, El-Makky and Taha 2001). For the rest of the paper, section 2 provides a review of related works *ITAR*. Section 3 gives a description of the incremental mining problem of temporal association rules. Section 4 presents the proposed algorithm *ITAR* in details. The section afterwards describes how the algorithm has been adopted for crime pattern discovery. Another real life dataset has been used to experiment with the new algorithm and the results are presented in section 6. Section 7 concludes our work.

2 Related Works

2.1 Temporal Association Rules with Numerical Attributes

The definition of temporal association rules is provided in the works of Wang, Yang and Muntz (1999, 2001) together with the introduction of the TAR (Temporal Association Rule) algorithm. It deals with objects in a database, *DB*. Each object O_i has a unique ID and a set of numerical attributes $A_1, A_2, \ldots, A_n$. The *DB* is viewed as a set of sequences of snapshots S_1, S_2, …, S_t and each snapshot has objects consisting of attributes with their numerical values.

Evolutions are defined to represent the temporal changes of attribute value of some object histories. Given an attribute A_i, an attribute evolution $E(A_i)$ of length m describes the value change of A_i over m consecutive snapshots. $E(A_i) = (A_i\ ?\ [l_1, u_1])\ ?\ (A_i\ ?\ [l_2, u_2])\ ?\ \ldots\ ?\ (A_i\ ?\ [l_m, u_m])$ represents the changes of A_i over m consecutive snapshots where $[l_j, u_j]$ is the interval of values of A_i, and is the union of a sequence of adjacent base intervals. For example, an employee's salary changes from an interval of $40,000 and $45,000 to an interval of $50,000 to $55,000, and then to an interval $60,000 to $65,000. This example represents an evolution that across 3 snapshots regardless of it is consecutive or not. The attribute value is presented in the interval form instead of distinct values.

A temporal association rule can be discovered from base cubes as long as it achieves the minimum *support* and *strength*. Let A_1, A_2, …A_n be a subset of attributes. A temporal association rule, R, of length m is defined as:

- $E(A_1) \cap E(A_2) \cap \ldots \cap E(A_{k-1}) \cap E(A_{k+1}) \cap \ldots \cap E(A_n) => E(A_k)$, and
- $E(A_i)$ is an evolution of length m with attributes $A_i (1 <= i <= n)$.

In TAR, three thresholds, which are support, density and strength, are introduced to select interesting temporal association rules. The support of a temporal rule $E(A_1) \cup E(A_2) \cup \ldots \cup E(A_{k-1}) \cup E(A_{k+1}) \cup \ldots \cup E(A_n) \Rightarrow E(A_k)$ is the support of an evolution conjunction $E(A_1) \cup E(A_2) \cup \ldots \cup E(A_n)$, that is the number of object histories of length m which follow this evolution conjunction. The density is the ratio of the number of objects to total number of objects, which will make a base cube considered "dense". The base cube is defined as evolution cube which consist of a set of attributes and whose values fall into base interval regarded as non-distinguishable. The density is introduced to ensure that a cube will not be included as part of a rule if there is no enough evidence to show that the rule holds for the cube. The strength of an association rule is $\text{Support}(E(A_1) \cup E(A_2) \cup \ldots \cup E(A_n)) / \text{Support}(E(A_1) \cup E(A_2) \cup \ldots \cup E(A_{k-1}) \cup E(A_{k+1}) \cup \ldots \cup E(A_n))$. The strength indicates the degree of dependence between attributes $(A_1, A_2, \ldots, A_n)$ of the rule.

The TAR algorithm has two phases. The first phase is to find all base cubes, *BaseCube,* which satisfy the density threshold after partitioning each attribute domain into *b* base intervals. Based on dense base cubes, the second phase finds all valid temporal rule sets. In our work, we focus on refining phase 1. It is because updating original database by appending incremental database affects the densities and the distributions of existing discovered base cubes, i.e. *BaseCube*.

2.3 Incremental Mining of Association Rules

The problem of incremental mining was solved by the FUP algorithm (Cheung, Han, Ng, Fu and Fu 1996). Let L^{DB} be a set of frequent itemsets in a transactional database *DB*. Let L^{db} be a set of frequent itemsets in an incremental database *db* that is going to be appended to DB then eventually form *DB+*. With the same support threshold, it is required to discover L^{DB+}. The output of L^{DB+} is a new set of frequent itemsets as the result of incremental mining. The itemsets of winners and losers are defined below.

- Winners: infrequent itemsets of *DB* that become frequent in *DB+* after adding the increment data to the database.
- Losers: frequent itemsets of *DB* that become infrequent *DB+* after adding the increment data to the database.

Let *X* be an itemset and superscript *DB* or *db* present the type of database. In order to discover the winners and losers from itemsets, itemsets can be classified into four types from L^{DB} and L^{db} as shown in Figure 1.

	$X \in L^{db}$	$X \notin L^{db}$
$X \in L^{DB}$	1	2
$X \notin L^{DB}$	3	4

Figure 1. Itemset updating scenarios.

Scenarios (1) and (4) in Figure 1 are easy to handle because itemsets belong either (1) and (4) represent itemsets definitely become winners and losers, respectively. Itemsets in scenario (2) and scenario (3) are the uncertainty areas because itmesets can become either winners and losers. Scenario (2) would be the easier one because support counts of itemsets in L^{DB} are already available and itemsets can become winners as long as updated support counts are equal or exceed the minimum support. Scenario (3) would require rescanning the original database since the support counts of itemsets in original database *DB* were unknown.

The major drawback of the *FUP* is that it was built on the *Apriori* algorithm with small modifications. Many passes of checking against the original database would usually be required. To remedy this problem, a concept of negative border introduced by Toivomen in 1996 can indicate the need for an update of rules. Informally, the negative border of a set of frequent itemsets *L* consists of all the itemsets that are not in *L* but have all their subsets in *L*. The itemsets in negative border for subsequent incremental mining have potential to become a winner as long as it achieves the minimum support threshold. On the other hand, if no winner appeared in the negative border, there is no need for checking further in the original database. The negative border, therefore, can provide a signal of the need for updating the database. It has been also used in *ICAP* (Ayad, El-Makky and Taha 2001) and *IUS* (Zheng, Xu, and Ma 2002) to improve the efficiency of speed and minimize the use of resources during mining.

3 Incremental Mining of Temporal Association Rules

3.1 Incremental Mining Problem

The problem of incremental mining of temporal association rules is mostly identical to the one in general association rules. The idea is to discover new sets of dense base cubes by reallocating the average density in hyercubes of evolution spaces when there is another set of objects *db* added into *DB.*

Let $Dense^{DB} = \bigcup_{i=1}^{m} \bigcup_{j=1}^{T} BaseCube\,(i, j) = \{bc \mid bc >= density\}$ be a set of dense base cubes from all sliding windows of *DB* that satisfies the density threshold where *m* is width of window and *T* is the number of attributes. An incremental database *db* can be added to *DB* and the $Dense^{DB}$ is needed to be updated.

After updating, all winners are updated to $Dense^{DB+}$ and losers are updated to temporal negative border *TNBd*. The negative border, *TNBd*, keeps all losers that were the winners in previous *DB*. The concept of temporal negative border here is dissimilar to original negative border. It aims to avoid generating un-necessary powersets of base cube from the winners. Further details of temporal negative border and its comparison will be presented in Section 3.3. Before continuing our discussion, the notations used in this paper are shown in Figure 2.

DB	The original database
db	The incremental database
DB+	The updated database ($DB+ = DB \cup db$)
$\|DB\|$, $\|db\|$, $\|DB+\|$	Number of objects in *DB*, *db*, and *DB+*, respectively
T^{DB}, T^{db}, T^{DB+}	Set of snapshots in *DB*, *db*, and *DB+*, respectively
T_s^{DB}, T_e^{DB}, T_s^{db}, T_e^{db}, T_s^{DB+}, T_e^{DB+}	Starting snapshot and ending snapshot in DB, db, and DB+.
O^{DB}, O^{db}, O^{DB+}	Set of objects in *DB*, *db*, and *DB+*, respectively
bc	A base cube of the set of attribute evolution follows the same temporal sequence
$Dense^{DB}$, $Dense^{db}$, $Dense^{DB+}$	Dense base cubes for *DB*, *db*, and *DB+*, respectively
BaseCube(i,m)	Level-wise regional base cube having *i* distinct attributes and length *m* of evolutions
db^n	nth incremental database for mining
m	Size of sliding window
b	Number of intervals for attributes
min_sup	Minimum support threshold
e	Density threshold
W(j,m)	Sliding window with width *m* and starting snapshot is *j*

Figure 2: Notations used.

3.2 Different Cases in Updating Temporal Association Rules

We adopt the similar approaches to handle INSERT and APPEND cases as in *IncSpan* (Cheng, Yan, and Han 2004). INSERT case represents pure object growth in a database regardless to snapshot domain coverage. New objects in *db* might introduce new timestamps (snapshots) of attributes. Also, *db* might introduce itemsets with new atomic items which were not existed in *DB*. The APPEND case presented in *IncSpan* adds itemsets at the end of sequence of an existing sequence, where $T > T_e^{DB}$. In terms of snapshot coverage, APPEND case can also include previous snapshots, where $T_s^{DB} <= T <= T_e^{DB}$ and $T \notin T^{DB}$ and $T < T_s^{DB}$ and $T \notin T^{DB}$. After updating, T_s^{db+} and T_e^{db+} will be updated. The coverage of the *db* in *DB* could be classified into 3 types of coverage, total overlapping, partial overlapping and non-overlapping.

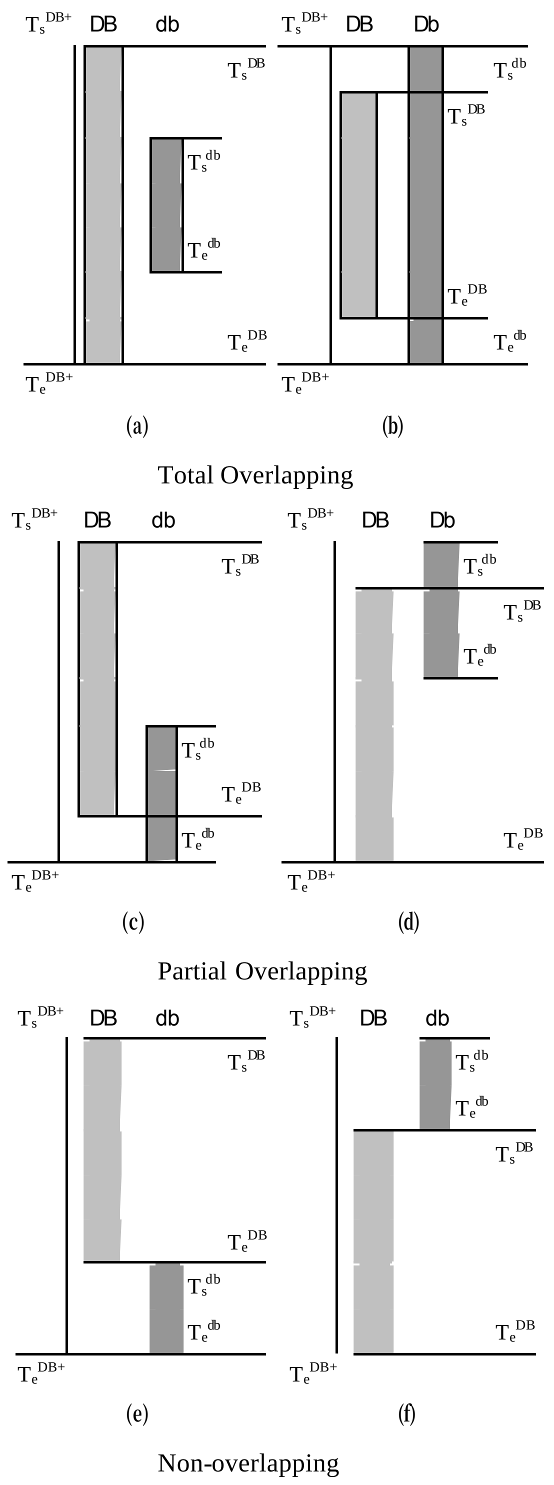


Figure 3. Snapshot Coverage for APPEND

Total overlapping is that the domain of snapshots of either *DB* or *db* fully covers each other. Case (a) in Figure 3 does not have any new snapshot to be introduced into *DB*, where T^{db} ? T^{DB}, and $T_s^{DB} = T_s^{db} = T_e^{db} = T_e^{DB}$. Case (b) in Figure 3 introduces new snapshots which beyond both sides of domain, where T^{DB} ? T^{db}, and $T_s^{db} = T_s^{DB} = T_e^{DB} = T_e^{db}$. The domain of snapshots in *DB+* will be updated to same as *db*.

Partial overlapping is that some snapshots of *db* cover the domain snapshots in DB. Case (c) in Figure 3 represents that *db* introduces new subsequent snapshots at the end point of domain where $T_s^{DB} = T_s^{db} = T_e^{DB} = T_e^{db}$ and the updated domain will be $T_s^{DB+} = T_s^{DB}$, $T_e^{DB+} = T_e^{db}$ in $DB+$. Case (d) in Figure 3 has new snapshots from db complementing aforetime side where, $T_s^{db} = T_s^{DB} = T_e^{db} = T_e^{DB}$, and the updated domain will be $T_s^{DB+} = T_s^{db}$, $T_e^{DB+} = T_e^{DB}$

Non-overlapping is that there is no overlapped snapshot in *db*. Case (e) in Figure 3 is similar to the APPEND case in *IncSpan* where new subsequent snapshots attributes are to be introduced and $T_s^{DB} = T_e^{DB} < T_s^{db} = T_e^{db}$ and updated domain for $DB+$ will be $T_s^{DB+} = T_s^{DB}$, $T_e^{DB+} = T_e^{db}$. Case (f) in Figure 3 is the inverse version of case (e) where $T_s^{db} = T_e^{db} < T_s^{DB} = T_e^{DB}$, and updated domain for $DB+$ will be $T_s^{DB+} = T_s^{db}$, $T_e^{DB+} = T_e^{DB}$.

Besides timestamp considerations, new objects may be added during updating. There can be five different cases of temporal database insertion. In order for better explanation, we illustrate with a sample database. Given a database with one object, and let $I = \{A_1, A_2, A_3, ... , A_6\}$ be its set of literals. The object consists of set of items with timestamp as $\{\{T_1: A_1, A_2, A_3\}, \{T_2: A_2, A_3, A_5\}, \{T_3: A_1, A_3, A_5, A_6\}, \{T_4: A_1, A_2, A_3\}\}$. The object could be transformed to be a set of atomic items $\{A_1T_1, A_2T_1, A_3T_1, A_2T_2, A_3T_2, A_5T_2, A_1T_3, A_3T_3, \ldots, A_3T_4\}$. Therefore, the set of atom items will be extended if the timestamp of new data is growing.

- *Case 1: Updating values of existing atom items:* Atomic items are modified for an existing object. For example, atomic items A_1T_1=2 and A_2T_2=4 are to be replaced by A_1T_1=3 and A_2T_2=5, where $T_s^{DB} <= T <= T_e^{DB}$. After updating, the average density of base cubes might be affected because the new values of relevant items might be fall into other interval.

- *Case 2: Inserting items into existing snapshots:* In this case, un-existed items X ? I are to be supplemented. For example, items A_4T_1, A_2T_3, and A_4T_3 with their values, which were not existed in *DB* and where A_i ? I and $T_s^{DB} <= T <= T_e^{DB}$, are intersect into the existing histories. Therefore, the object is updated to $\{A_1T_1, A_2T_1, A_3T_1, \underline{A_4T_1}, A_2T_2, A_3T_2, A_5T_2, A_1T_3, \underline{A_2T_3}, A_3T_3, \ldots, A_3T_4\}$. After updating, the average density of base cubes would be affected because new atomic items were inserted into *DB*.

- *Case 3: Appending subsequent snapshots:* Introduces new timestamp items where A_i ? I and $T > T_e^{DB}$. For example, $\{\{T_5: A_2, A_3, A_4\}, \{T_6: A_4, A_6\}\}$ insert into subsequent object history.

- *Case 4: Inserting new items into existing snapshots:* From case 1 to case 3, the items are only the member of existing literals. However, *db* can introduce some new literals such as in set $J = \{A_7, A_8, ..., A_{10}\}$, where J n I = Ø. This is similar to case 2 but incremental data has new literals. Incremental database *db* consists of $\{\{T_1: A_7, A_8,\}, \{T_4: A_8, A_9\}\}$, where $T_s^{DB} <= T <= T_e^{DB}$ and A_i ? J.

- Case 5: *Appending subsequent snapshots with new literals of item:* This is similar to case 3 but incremental data is new literals of item. Incremental database *db* consist of $\{\{T_5: A_7, A_8,\}, \{T_8: A_8, A_9\}\}$, where $T_n > T_e$ and A_i ? J.

  In our work, we focus on case 3, case 4 and case 5 in which new objects are inserted or appended in the incremental database.

3.3 Temporal Negative Border

Since mining temporal association rules is restricted by sliding windows and temporal sequence, the original concept of negative border when applied directly would introduce the following problems:

- The power set, *P(R)* where R is a temporal association rule, generation approach by the original negative border for the base cubes is not practical because the set of literals for obtaining atomic items in base cubes can become very large since temporal evolution is continuous in nature.

- Constructing level-wise dense base cube *Dense* of every slide window lattice employs the *AprioriGen* method. However, it requires additional computation effort to construct the set of minimal itemsets *X*? *R* not in *Dense*. Computation effort includes time for processing, and memory and I/O usage for generating and storing all power set of rules *P(R)* from frequent itemsets to negative border by the *AprioriGen* method.

- The updating problem of temporal rules has a fundamental difference from generic association rules in which the set of literals $I = \{i_1, i_2, \ldots, i_3\}$ was well established and predictable. In temporal association rule mining, snapshot evolution is continuously growing and the set of literals becomes large. Consequently, the size of negative border becomes relatively large.

Hence, we define a variation of negative border, called the *temporal negative borde (TNB)r*, to simplify the computation problem above. The TNB of set of base cubes, $Dense^{DBn}$ which follow the past object histories in evolution cubes, is referred to as *TNBd(Dense)*. That is $TNBd(Dense^{DBn})$ = {bc | bc $\notin$ $Dense^{DBn}$ and bc ? $\exists \bigcup_{i}^{n-1} Dense^{dbi}$, where 1 <= i < n }. In other words, it says to include all the losers in the last incremental mining if these base cubes have been winners before.

Suppose *DB* is the initial database and subsequent incremental database are represented as db^i, where 1 <= *i* <= *n*. The negative border in the first mining is empty since there was no existing negative border. From second mining iteration onward, the negative border would keep previous losers. There are 3 sets of dense bases, $Dense^{DB}$, $Dense^{db}$, and $TNBd(Dense^{DB})$ to be processed for each mining iteration. The $TNBd(Dense^{DB})$ in second mining is still empty. $TNBd(Dense^{DB+})$ is formed from the losers of *DB*. Since the losers from all $Dense^{db}$ are never to be considered, for any base cube, *bc*, in $Dense^{DB}$, it must be winners of $Dense^{db}$. Therefore, the losers from $Dense^{DB}$

must also be the winners of some previous *db*. There are several benefits in using the temporal negative border:

- It is able to eliminate un-necessary sets of base cubes. It would not generate powersets that do not exist in *db* nor *DB*.

- With using the new negative border, the rescanning of the complete database can be at most one time. The new base cubes having new snapshot(s) or existing snapshot(s) not occurred in previous *Dense* are discovered.

- It handles the problem of the INSERT and APPEND cases. In temporal association rules, the INSERT case introduces object histories of new objects. It can be easily handled based on the property of support counts between winners and losers as well as the intersections mapping concept in *ICAP* (Ayad, El-Makky and Taha 2001, Cheng). The APPEND case involves new discovered base cubes in $Dense^{db}$, where base cubes were not previously discovered both in $Dense^{DB}$ and *TNBd*. Attached snapshots in such base cubes are either new ($T >= T_e^{DB}$ or $T <= T_s^{DB}$) or existing timestamps ($T_s^{DB} <= T <= \mathrm{T}_\mathrm{e}^{\mathrm{DB}}$). New discovered base cube will be required an original database rescan. In our algorithm, it can deal with the INSERT case and the hybrid of INSERT and APPEND case since *db* only contains new objects.

4 Incremental TAR Algorithm

Our Incremental Temporal Association Rule (*ITAR*) algorithmas shown in Figure 4 has two assumptions. The first one is the incremental database *db* only contains new objects with their details which were not existed in *DB*. Second, if a base cube *bc* in *db* satisfied the density threshold before the database is updated, *bc* will satisfy the density threshold in *DB+* as well; otherwise, *bc* is never to be considered. No matter *bc* was already in either $Dense^{DB}$ or $TNBd(Dense^{DB})$.

The ITAR algorithm takes input as the set of dense base cubes $Dense^{DB}$ of the original database *DB* and the negative border $TNBd(Dense^{DB})$. Incremental database *db* is to be mined separately using *TAR* algorithm that delivers a new set of dense base cubes $Dense^{db}$. After the merging of three sets of dense base cubes, five different groups from the sets can be identified and each group will be respectively processed. The proposed algorithm can be summarized in the following steps:

1. Use the original *TAR* algorithm to discover dense base cubes (all level-wise base cubes of all sliding window) denoted as $Dense^{db}$ in the incremental database *db*.
2. With the discovered $Dense^{db}$, all intersections of $Dense^{db}$, $Dense^{DB}$, and $TNBd(Dense^{DB})$ could be identified. After accumulating each base cube's density, winners will be fall into $Dense^{DB+}$ meanwhile losers will be fall into new negative border, $TNBd(Dense^{DB+})$.
3. After step 2, the remaining bases cubes would be filtered from the sets of intersections. Remaining base cubes in $Dense^{DB}$ are required to check their densities again for the updated object counts. On the other hand, remaining base cubes in $TNBd(Dense^{DB})$ will automatically be fall into $TNBd(Dense^{DB+})$ since such base cubes are not the winners in current mining.
4. All base cubes of $Dense^{DB}$ and $TNBd(Dense^{DB})$ have been offset. The remainders in $Dense^{db}$ represent those were not occurred in both $Dense^{DB}$ and $TNBd(Dense^{DB})$ but are new discovered base cubes from *db*. It requires rescanning the original database to updated densities for remainders. If a base cube *bc* satisfies the density threshold, *bc* will fall into $Dense^{DB}$; otherwise, it will fall into $TNBd(Dense^{DB+})$.

ITAR Algorithm:

```
Function ITAR ($Dense^{DB}$, $TNBd(Dense^{DB})$, $db$)
1. Compute $Dense^{db}$ using TAR algorithm
2. $Dense^{DB+} = \emptyset$ ; $TNBd(Dense^{DB+}) = \emptyset$
3. $|DB+| = |DB| + |db|$ // Accumulate the number of object to DB+
4. For each base cube bc in $Dense^{DB} \cap Dense^{db}$ do // (1)
5.         bc.count of DB+ = bc.count of DB + bc.count of db
6.         $Dense^{DB+} = Dense^{DB+} \cup \{bc\}$
7. For each base cube bc in $TNBd(Dense^{DB}) \cap Dense^{db}$ do // (2)
8.         bc.count of DB+ = bc.count of DB + bc.count of db
9.         if bc.count of DB+ >= |DB+| / b x e then
10.                $Dense^{DB+} = Dense^{DB+} \cup \{bc\}$
11.        else    $TNBd(Dense^{DB+}) = TNBd(Dense^{DB+}) \cup \{bc\}$
12. For each base cube bc in $Dense^{DB}$ do // (3)
13.        if bc.count of DB+ >= |DB+| / b x e then
14.                $Dense^{DB+} = Dense^{DB+} \cup \{bc\}$
15.        else    $TNBd(Dense^{DB+}) = TNBd(Dense^{DB+}) \cup \{bc\}$
17. For each base cube bc in
            $TNBd(Dense^{DB}) \{bc \mid bc \notin Dense^{db}\}$ do // (4)
18.        $TNBd(Dense^{DB+}) = TNBd(Dense^{DB+}) \cup \{bc\}$
19. if $Dense^{db} \{bc \mid bc \notin TNBd(Dense^{DB})$
            and $bc \notin Dense^{DB}\} = \emptyset$ then // (5)
20.        For each O in DB do
21.                For all cube bases bc in O do
22.                        if bc ? $Dense^{db}$ then
23.                bc.count of DB+ = bc.count of DB+ ++
24.                For each bc in $Dense^{db}$ do
25.                        if bc.count >= |DB+| / b x e then
26.                        $Dense_1^{DB+} = Dense_1^{DB+} \cup \{bc\}$
27.                        else
28.               $TNBd(Dense^{DB+}=) = TNBd(Dense^{DB+}) \cup \{bc\}$
29. Return $Dense^{DB+}$ and $TNBd(Dense^{DB+})$
```

Figure 4: A high level description of algorithm ITAR

The idea of *ITAR* is to merge the dense base cubes between the intersections of $Dense^{DB}$, $Dense^{db}$, and $TNBd(Dense^{DB})$. During step 4, rescanning the original database is required if and only if the remaining base cubes do not belong to any intersections from *db*.

The original negative border (Toivonen 1996) made use of the idea in the *Apriori* algorithm to generate all candidate itemsets where negative border of *L* consists of minimal itemsets *X* not in *L*. Our variation of negative border does not posses the same property. The proposed temporal negative border avoids the power set of base cube generation. A smaller negative border is exploited to improve mining performance by cutting down redundant or insignificant base cube checking. For example, suppose a base cube $bc = \{E(A),E(B),E(C)\}$ was never a winner previously but subsets ($\{E(A)\}$, $\{E(B)\}$, $\{E(C)\}$, $\{E(A),E(B)\}$, …, $\{E(B),E(C)\}$) of *bc* were the winners. The original negative border would hold *bc* which might be already by power set. However, temporal negative border never keeps it.

5. Crime Pattern Discoveries

We have adopted the new algorithm, ITAR, to support a crime pattern analysis for a district in Hong Kong for better computational performance. In the developed system, an end-user can make a request for mining of the temporal association rules through the interface of the system. The interface will accept a user request and then the corresponding query is generated for collecting corresponding crime data from the database. The query specifies the day range (week, month), attributes and grid size, and counts of crime incidence to be utilized. The query can also filter crime data with missing values. In order to generalize the coordinates of crime data, coordinates are divided into grids for better spatial mining results. Figure 11 shows a system architecture of the crime pattern analysis system.

Crime information extraction is done by *crime extractor*, which collects and analyses user requests, collects corresponding crime data from the database, processes the data and then passes the data to *dense base cube generator* to create base cubes for mining. The results of the *dense base cube generator* is used for examining crime trends, generating spatial-temporal associations, or mining incrementally. The mining results are returned to the end-user through a graphical interface. The back-end database stores the crime data for analysis. Dense base cubes generated from the *dense base cube generator* are also stored in the database to prepare for incremental mining. When the system requires mining incrementally, the original dense base cubes can be retrieved from the database.

Crime trend discovery is done by *crime trend generator*, which collects dense base cubes generated from *dense base cube generator*, analyses snapshots in dense base cubes and outputs the trend results. In order to generate a dense base cube, the density of the conjunction of snapshots in the cube must be greater than the density threshold. The density thresholds can control the generation of trend, so that if there are enough records or evidence to show that a trend exists, the trend will be discovered.

Incremental base cube discovery is done by *incremental base cube generator*, which collects original dense base cubes generated from *dense base cube generator* or the database, mines dense base cubes from incremental database, merges the dense base cubes of incremental database to cubes of original database and passes the updated dense base cubes to the *spatial-temporal association rules generator* to generate the rules.

Since continuous temporal change of value of crime attributes is mined, the crime attributes must be in numerical form. However, there is no numerical attribute, except coordinates, in the crime data provided. The offence and mo (modus operandi, which means committing crime method) attributes of crime data are categorized into different levels of seriousness, with values from 1 to 10. Figure 5 and Figure 6 shows some examples of categorizing of offence and mo respectively.

Offence	Seriousness
Affray	1
Breach of condition of stay	2
Resisting arrest	3
Lending at excessive rate	4
Criminal damage	5
Robbery, deception	6
Burglary	7
Criminal intimidation	8
Assault	9
Arson, child abuse	10

Figure 5. Examples of Categorizing of Offence

MO	Seriousness
Loitering – causing others to be concerned	1
Loitering – with intent to commit an arrestable offence	2
Theft – steal from locker	3
Theft – pickpocket – cut open	4
Deception – modelling school	5
Robber – use knife, fruit knife 20cm long	6
Wounding – minor dispute	7
Possession of dangerous drug –stop and search	8
Possession of dangerous drug -operation	9
False imprisonment – escort back and detain to settle debt	10

Figure 6. Examples of Categorizing of MO

Although the crime data is stored in a transaction database, it can be viewed as a sequence database, which records a set of snapshots that capture crimes happened at different locations in different time intervals (week or month). Each base snapshot generated from crime data records the values of a set of crime attributes in base intervals. For example, a base snapshot may record that there are ten crimes with offence=4 happened at grid (x=4, y=8) in January. When association relationship (crime1$\Rightarrow$crime2) between two crimes happened in different locations are tried to discover, base snapshots must record attributes of the two crimes. However, there is no single crime record consists of two crimes happened at two different locations, so single crime record cannot be used to generate base snapshot with two crimes. In order to grouping two crime records together to generate the base snapshot, we assume that when two crimes happened in the same time interval (week or month), there is correlation relationship between the crimes. For example, a base snapshot may record that there are ten crime incidence happened in January, one of the crimes with offence=4 happened at grid (x=4, y=8), another crimes with offence=8 happened at grid (x=10, y=14).

The spatial-temporal crime analysis system has been developed by the JAVA 2 programming language (JDK 1.5.0_06). Eclipse 3.0 Software Developer's Kit was used as development platform for debugging and building the system. MySQL server 5.0 was used as database of the system. Figure 7 shows the interface of the system.

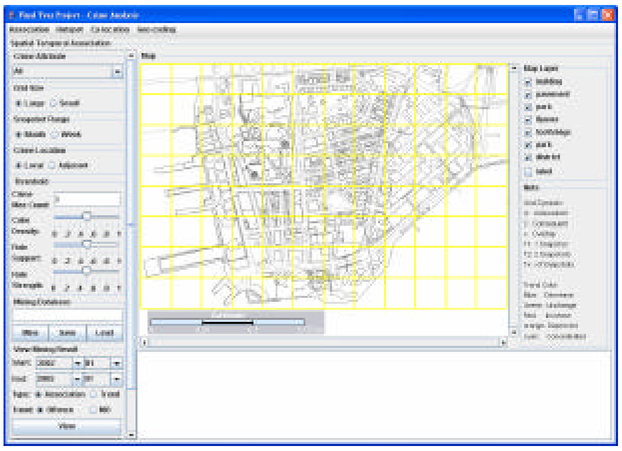


Figure 7. Crime Pattern Analysis System

6 Experiments

6.1 Environment and Configuration

For the verification of the performance of the ITAR algorithm, we have tried it with a real data set. The data used here is a transactional database with customers shipment details for the whole year of 2002 obtained from a shipping company. There are a total of 40,000 customer transactions to represent the daily histories of shipment services used. Each transaction has a customer ID, and a shipment product with its total volume of the same day. The snapshot was taken once a month, from Jan 2002 to Dec 2002. With the real dataset, a synthetic temporal database for the experiments is generated by means of SQL with grouping and aggregation techniques. Each element of the database contains an object with its identifier and object histories of 12 snapshots (months). The properties of the synthetic temporal database are summarized and listed in Figure 8.

Number of objects in temporal database	40,000
Number of attributes	5
Period	1 Jan 2002 to 31 Dec 2002
Snapshots	Each month represent 1 snapshot

Figure 8. Synthetic dataset properties for experiments

Furthermore, the synthetic temporal database with 40,000 objects has also been randomly partitioned into 20 sets of small and equal size datasets as incremental databases. Each dataset consists of 2,000 objects (5 percents of the original database) with their object histories. Meanwhile, we used these small datasets to generate another 20 sets of accumulated datasets for different mining iterations. For example, the accumulated dataset for n^{th} iteration incremental mining has the 1^{st} to n^{th} small datasets. There are two types of experiments. The first type is to compare the mining speed between *ITAR* and *TAR*. The second type is to verify the resultant temporal rules generated by the two algorithms respectively.

6.2 Mining Performance

In executing ITAR, the number of base intervals for attributes in the dataset, the width of sliding window, and the density threshold are the important parameters that can determine required runtime.

We divided the set of experiments that used different parameters: attribute intervals from 10 to 40, the widths of sliding windows from 3 to 5, and density thresholds from 0.5 to 2. In each experiment, we fixed the values for two parameters and adjusted the remaining one. The Accumulated ITAR (AITAR) algorithm is implemented as the TAR algorithm was applied to the complete database every time when there is an update on the database. Figure 9 shows the results of *ITAR* algorithm under different setups of parameters. In Figure 10, it shows the ratio of mining efficiency between *ITAR* and *TAR*. Second, it shows the ratio of mining efficiency between accumulated *ITAR* (AITAR) and *TAR*.

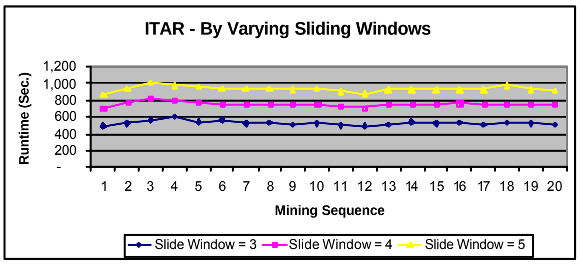
In both Figures 9 and 10, the performance of *ITAR* with different parameters is very stable and showing good performance. In contrast, the performance of *TAR* shows a linearly growth along the mining sequence.

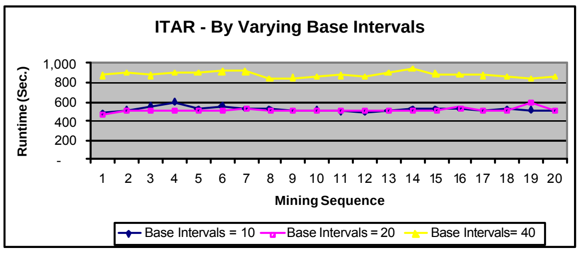
7 Conclusion

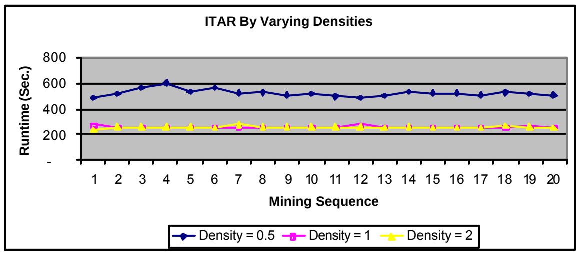
Researches in temporal association rules mining have been developing for many years. Due to the re-scanning issue in *TAR* in order to handle dataset updating, an incremental algorithm *ITAR* (Incremental *TAR*) is proposed in this paper. The algorithm applies the techniques of incremental mining on the *TAR* to update the dense base cubes after the update of the original database by adding new objects with temporal histories. Due to the nature of the temporal association rules mining model, a variation of negative border is proposed for *ITAR* in order to simplify the use of the original. Temporal negative border proposed in this paper only keeps all past winners which are becoming losers instead of comprising the power set of dense base cubes. Therefore, the number of losers of base cubes held by temporal negative border can be minimized. The ITAR algorithm has been adopted in a crime pattern analysis system. Furthermore, two sets of experiments have been conducted to measure the relative performance of the new algorithm compared to the *TAR* algorithm. The results reflected that the exhibitions of *ITAR* benefit a significant efficiency against *TAR* in term of efficiency ratio. Moreover, using temporal negative border enables the avoidance of any missing base cube when using ITAR.

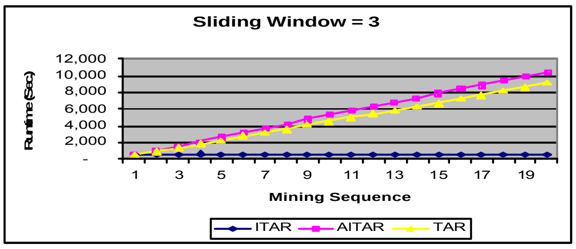
Acknowledgement

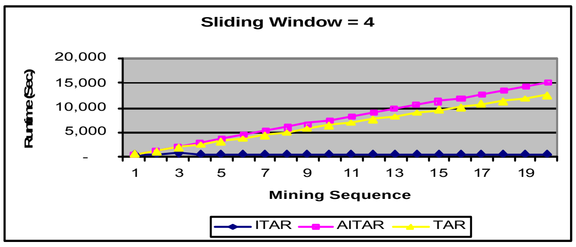
The work of the authors is supported in part by the CERG RGC Grant of research project code PolyU 5106/05E (BQ938).

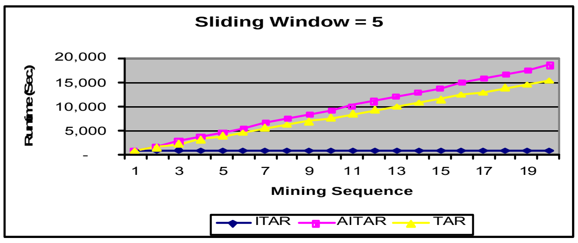


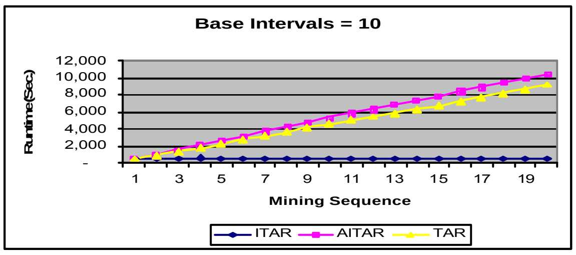


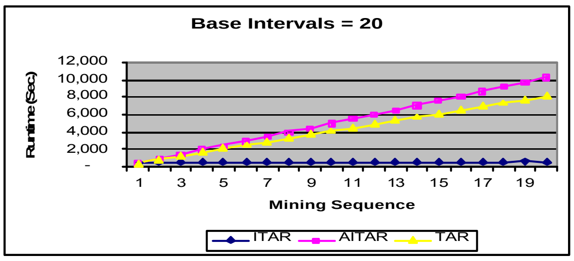


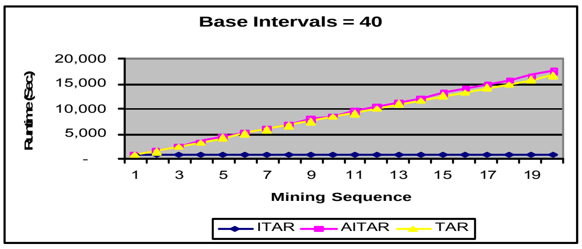












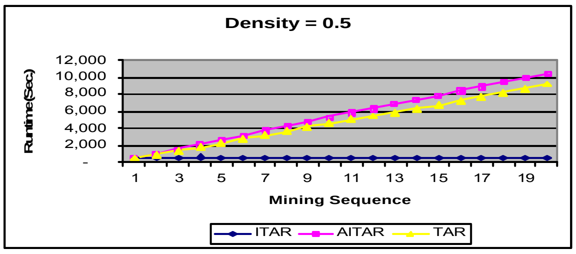
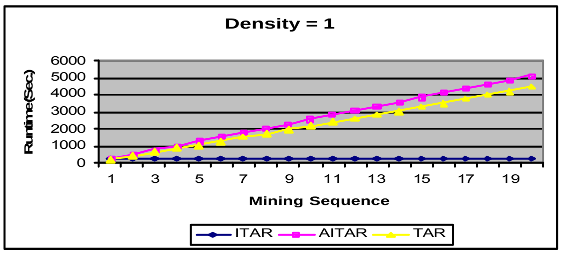


Figure 9. Mining Efficiency of ITAR comparison by different parameters



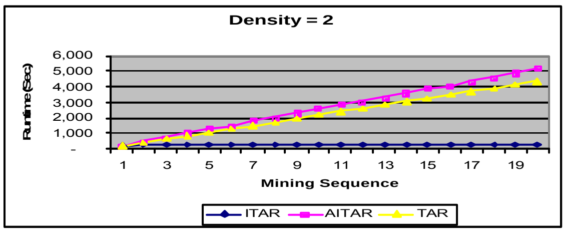


Figure 10. Mining Efficiency comparisons (by Varying Density thresholds)

8 References

Agrawal, R., Imielinski, T. and Swami, A. (1993): Mining association rules between sets of items in large databases. *Proc. ACM SIGMOD International Conference on Management of Data*, Washington DC, USA, **22**:207-216, ACM Press.

Agrawal. R., and Srikant. R. (1994) Fast algorithms for mining association rules. Proc. of the 20th Int' l Conf. on Very Large Database, Santiago, Chile, Sept. 1994, pp. 487-499.

Ayad. A., El-Makky. N., and Taha. Y. (2001) Incremental Mining of Constrained Association Rules. In First International SIAM Conference on Data Mining (SDM01), Chicago, April 2001. http://www.siam.org/meetings/sdm01/pdf/sdm01_01.pdf.

Ayan, N.F., Tansel, A.U., and Arkun, E. (1999) An efficient algorithm to update large itemsets with early pruning. The 5th Conference on Knowledge Discovery and Data Mining (SIGKDD99), AMC Press, San Diego, USA, pp. 439-450.

Cheng. H., Yan. X., and Han. J. (2004) “IncSpan: Incremental Mining of Sequential Patterns in Large Database”, Proc. 2004 Int. Conf. on Knowledge Discovery and Data Mining (KDD'04), Seattle, WA, Aug. 2004, pp. 527-532.

Chang, C.H., Yang S.H. (2003) Enhancing SWF for incremental association mining by itemset maintenance. The 7th Pacific-Asia Conference on Knowledge Discovery and Data Mining, PKADD 203, LNAI 2637, Springer, Seoul, Korea, pp 301-312.

Cheung. D.W., Han. J., Ng. V., Fu. A., and Fu. Y. (1996) A fast distributed algorithm for mining association rules. In Proc. 1996 Int. Conf. Parallel and Distributed Information Systems, pp. 31-44, Miami Beach, FL. Dec 1996

Cheung. D.W., Han. J., Ng. V., and Wong. Y. (1996) Maintenance of discovered association rules in large databases: An incremental updating technique. In Proc. 1996 Int. Conf. Data Engineering (ICDE' 96), pp. 106-114, New Orleans, LA, Feb. 1996

Cheung. D.W., Lee, S.D., Kao, B. (1997) A general incremental technique for maintaining discovered association rules. The 5th International Conference on Database Systems for Advanced Applications, Melbourne, Australia, pp. 185-194.

Toivonen, H. (1996) Sampling large database for association rules. In 22nd International Conference on Very Large Database (VLDB' 96), pp. 135-145, Mumbay, India, September 1996.

Wang. W., Yang. Y., and Muntz. R. (1999) Temporal Association Rules with Numerical Attributes. NCLA CSD Technical Report 990011, 1999.

Wang. W., Yang. Y., and Muntz. R. (2001) TAR: temporal association rules on evolving numerical attributes,. Proceedings of the 17th IEEE International Conference on Data Engineering, pp. 283-292, 2001.

Zheng. Q., Xu. K., and Ma. S. (2002) The Algorithm of Updating Sequetial Pattern, 2002, http://xxx.arxiv.org/ftp/cs/papers/0203/0203027.pdf.

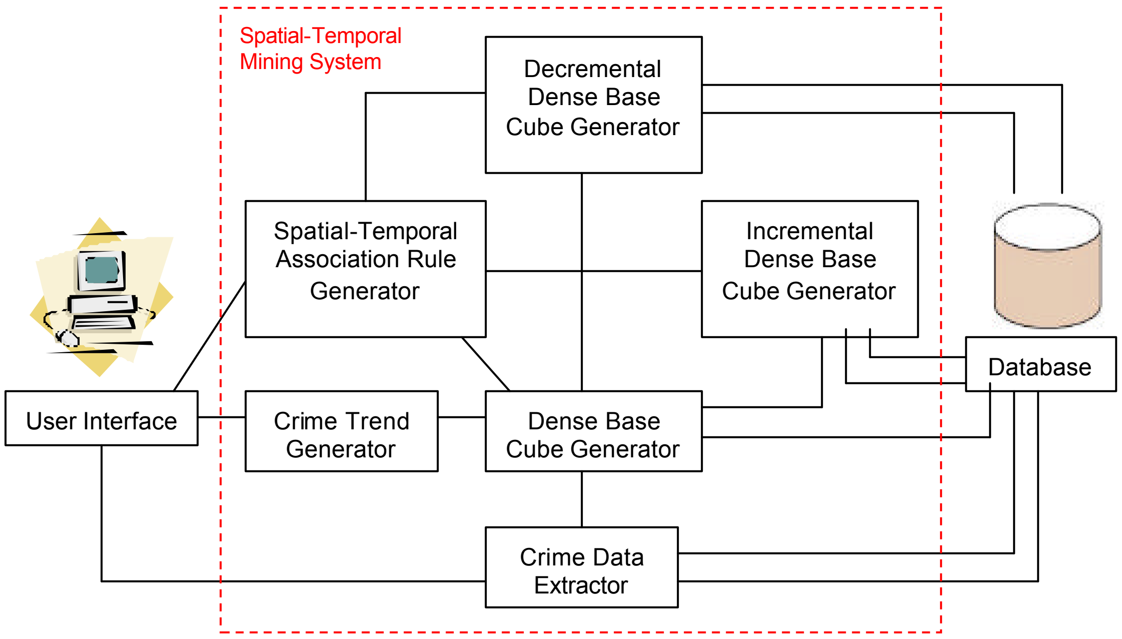

Spatial-Temporal
Mining System
Decremental Dense Base Cube Generator
Spatial-Temporal Association Rule Generator
Incremental Dense Base Cube Generator
Database
User Interface
Crime Trend Generator
Dense Base Cube Generator
Crime Data Extractor

Figure 11. Overview of the Crime Pattern Analysis System

Opinion Search in Web Logs

Deanna J. Osman **John L. Yearwood**

School of Information Technology & Mathematical Sciences
University of Ballarat,
Ballarat, Victoria,
Email: d.osman@ballarat.edu.au
Email: j.yearwood@ballarat.edu.au

Abstract

Web logs (blogs) are a fast growing forum for people of all ages to express their feelings and opinions on topics of interest. The entries are often written in informal language without the structure found in newswire or published articles. One blog entry may contain many topics, these topics may express an opinion or a fact on a particular topic.

This research is in contrast to work on opinion detection which has been carried out on more formally authored texts and on segments that are either whole documents or sentences. Whole web logs are divided into topics using a simple text segmentation approach. Similarity scores are used to distinguish where topic changes occur. The results are compared to human-evaluated topic changes and the most accurate algorithm is used in the remainder of the research.

Words within each topic-block are allocated weightings depending on their opinion-bearing strength. Two approaches of using these weights, the sum and the maximum, are used to determine whether the topic-block is opinion-bearing or non-opinion-bearing. The opinion-bearing topic-blocks are rated by human evaluators as either opinion-bearing or non-opinion-bearing with precision of 67% for approach A and 70% for approach B. These results are compared with two approaches on published text to identify the difference between web logs and published articles.

Keywords: Opinion Retrieval, Opinion Identification, Web Logs, Blogs

1 Introduction

Web logs (blogs) are a fast growing phenomenon on the World Wide Web (Web) as they allow people to publish their thoughts and opinions on any topic they choose. In July 2006 a blog tracking company, Technorati, Inc., reported that it was tracking 48.8 million blogs worldwide (*Currently tracking 48.8 million sites and 2.7 billion links* 2006), up from 4.2 million in October 2004 (Rosenbloom 2004). This number has grown to 57.4 million in October 2006.

Blogs are an informal form of communicating, usually the audience of the blog is known to the author. Blogs created by organisations such as newspapers or those addressing political or popular topics receive thousands of hits per day, however most blogs have a far smaller audience (Nardi, Schiano, Gumbrecht & Swartz 2004). The author of a blog is free to write informally and use any language required to express their thoughts and opinions.

The length of blogs varies from one paragraph to multiple paragraphs, the paragraphs within the blogs also vary considerably. The content within the blog is generally considered to be of high quality as the reputation of the author will encourage people to read the blogs on a regular basis.

On the other hand, newswire articles are written using a formal structure, using proper english without slang and word abbreviation. The length of the article is dictated by the space available, however one paragraph articles are uncommon. These articles have trained, experienced authors and an editor to ensure high quality writing techniques are used and words are not used out of context or with ambiguous meaning.

A search engine that retrieves only opinion-bearing blogs on a particular topic would allow a person to retrieve specific feedback on that topic. The parts of the blog that are not relevant to the query are removed, allowing only the text specifically relating to the topic to be retrieved.

The remainder of this paper is organised as follows. Section 2 presents related work on opinion retrieval and identification, text segmentation and passage retrieval. Section 3 describes the process followed throughout this research, with section 4 detailing the results of this research. Section 5 concludes this work and discusses future research.

2 Related Work

Major areas of this research require the discussion of the characteristics of blogs, investigation of opinion retrieval, text segmentation and passage retrieval research. The section discusses the work that relates to this research.

2.1 Characteristics of Blogs

Writing blogs is a form of communication of personal thoughts and opinions (Nardi et al. 2004, Leshed & Kaye 2006) intended as a form of personal communication. Blogs form a community by linking to similar blogs, many people who write blogs regularly read other blogs (Rosenbloom 2004).

Blogs are an information resource that can be read by anyone on the web, as such the reputation of products, organisations and companies can be discussed. This discussion can alter these reputations by spreading quickly around the web (Nanno, Fujiki, Suzuki & Okumura 2004).

Blogs are moving ahead of many mainstream journals and portals as information sources due to the expanding communities being linked together. A blog is a quick and cheap alternative to hiring expensive writers to maintain a web site (Searls & Sifry 2003).

2.2 Opinion Retrieval

Opinion retrieval aims to retrieve opinion-bearing text without distinguishing between positive and negative opinions (Eguchi & Lavrenko 2006). Opinion retrieval and opinion extraction has been researched at the sentence level (Riloff & Wiebe 2003), and at the word level (Yu & Hatzivassiloglou 2003, Kim & Hovy 2005, Hatzivassiloglou & McKeown 1997).

Evaluation of research in opinion retrieval relies on a corpus with annotation of opinions (Ku, Liang & Chen 2006). News articles and blog articles are used to create a corpus that has been tagged using XML-like tags to identify opinions within the documents (Ku et al. 2006).

The Wall Street Journal collection is used extensively for opinion identification as it contains a large number of editorials and letters to the editor with the assumption that they are opinion pieces (Yu & Hatzivassiloglou 2003, Kim & Hovy 2005). Terms are extracted from this document collection for classification as either opinion-bearing or non-opinion-bearing (Yu & Hatzivassiloglou 2003, Kim & Hovy 2005, Hatzivassiloglou & McKeown 1997). A high precision and recall (F-measure of 97%) was obtained using this method (Yu & Hatzivassiloglou 2003).

The Wall Street Journal term list (Yu & Hatzivassiloglou 2003) was expanded by using synonyms and antonyms (Kim & Hovy 2005) and a strength score was allocated to each word. Two methods of identifying opinion-bearing sentences were tested by Kim & Hovy (2005), the sum of the opinion score for an entire sentence and a high scoring word appearing within the sentence, with the latter being used in the testing. When applied to the MPQA corpus (Kim & Hovy 2005), the expanded list improved precision while the original list achieved higher recall. These two methods of applying the weights are tested in this research at a topic-block level instead of the sentence level. The list of opinion-bearing weightings list was extended by Hatzivassiloglou & McKeown (1997) by adding labels indicating the polarity of the adjective, which will be included in future research on opinion retrieval.

Subjective adjectives are extracted from 1,001 sentences from the Wall Street Journal Treebank Corpus by using annotators to indicate the subjective elements within each sentence (Wiebe 2000). This is improved using distributional similarity, which measures the similarity between words, using the broad-coverage parser (Lin 1994) and WordNet to expand the list.

Subjective language is used to express an opinion (Wilson & Wiebe 2003), this is called the private state. Annotators were asked to mark opinion/emotion-bearing sentences, and also to mark the private state within the sentences, within 13 English-language versions of foreign news documents. The documents annotated consisted of objective articles, news articles and editorials.

Subjective patterns are used to identify opinion-bearing phrases within text (Riloff & Wiebe 2003). These patterns contain word combinations that, when present, indicate the tone of the selected text. The subsumption hierarchy was used to develop a general pattern for expression of opinions within text (Riloff, Patwardhan & Wiebe 2006). These patterns include a flexible element that allows the same phrase to be expressed with extra words, this research extends the work reported by Riloff & Wiebe (2003).

A combination of Usenet newsgroup messages and Wall Street Journal data was used to create a corpus for annotators to indicate the subjectivity of individual sentences (Wiebe, Wilson, Bruce, Bell & Martin 2004). This allowed the generation of a subjective features list. Unique words (found once in the corpus) were found to be an indicator of an opinion being expressed.

The research reviewed above highlights the need for a document collection with annotations on whether the document is opinion-bearing or non-opinion-bearing.

Documents classified as opinion-bearing are considered to contain mostly opinion-bearing sentences (Yu & Hatzivassiloglou 2003). Four steps to determining the nature of an opinion are proposed by Ku, Lee, Wu & Chen (2005). The first of these steps is to determine the topics and sub-topics of the text. Then sentences relevant to the topic are retrieved and the sentence is examined for an opinion. Finally, the opinions are summarised and returned as the response to a query. This research modifies these four steps by adding an initial document retrieval step and dividing the documents into topic-blocks instead of determining the topics, as illustrated in Figure 1.

A system that uses a URL as input, reads the document and highlights the opinion-bearing sentences within the document (Wilson, Pierce & Wiebe 2003). This system uses manually identified features from a variety of sources (Levin 1993, *Framenet* 2002), combined with automatic information extraction techniques (Riloff & Jones 1999, Thelen & Riloff 2002) to identify opinion-bearing sentences. Automatic information techniques will be applied to the opinion retrieval approach developed in this research.

2.3 Text Segmentation and Topic Structuring

TextTiling determines the change in topic within a document. A document is divided into a block of text and adjacent blocks are compared and allocated a similarity score (Hearst 1997). The scores are graphed and the valleys within the graph indicate a change in the topic.

Different approaches to dividing full-length text into smaller segments to determine when a topic change occurs are investigated by Hearst & Plaunt (1993). A document can be divided using orthographically marked segments (paragraphs), even-sized blocks (30 words) or TextTiling.

These segments are compared to the adjacent segment to determine the similarity score, a drop in the score indicates a change in the topic. Once the document has been divided into topic-blocks, these topic-blocks are considered independent documents.

This research is used as a basis when applying topic segmentation to web logs. An alternative to topic segmentation is passage retrieval, where the document is divided into overlapping fixed-length blocks of text.

2.4 Passage Retrieval

Passage retrieval uses fixed-length passages that overlap, which are indexed and used to match a query (Kaszkiel & Zobel 1997). Query terms are more likely to be in close proximity due to the passage being shorter than an entire document. One difficulty with passage retrieval is that it creates a large number of passages to be indexed, therefore processing time is increased. Passage retrieval can be used in question/answering systems. Applying passage retrieval to web logs was not researched in this research, however future research will consider passage retrieval as an alternative to text segmentation and topic structuring.

3 Methodology

Opinion retrieval is an area covered well in scholarly research, however opinion retrieval within web logs research is not prevalent. This research aims to build an approach to retrieving opinions from within a corpus of blogs. This research utilises the process of document retrieval twice to retrieve opinions that match a query. Initially, to retrieve documents matching a query, creating a mini-corpus, this mini-corpus is indexed and used for document retrieval in the latter part of the process.

Figure 1 illustrates the process of finding opinion-bearing blogs that match the query for topic ‘A’ (Eg: “opinions on climate change”). The first step is to retrieve all blogs that match the query, creating a mini-corpus (corpus ‘A’). Corpus ‘A’ is expanded by dividing the blogs into topic-blocks, which are then treated as individual documents within corpus ‘A’.

The documents within corpus ‘A’ are identified as being opinion-bearing or non-opinion-bearing. The non-opinion-bearing documents are removed from corpus ‘A’. The remaining opinion-bearing documents are indexed to enable the retrieval of all opinions matching the query.

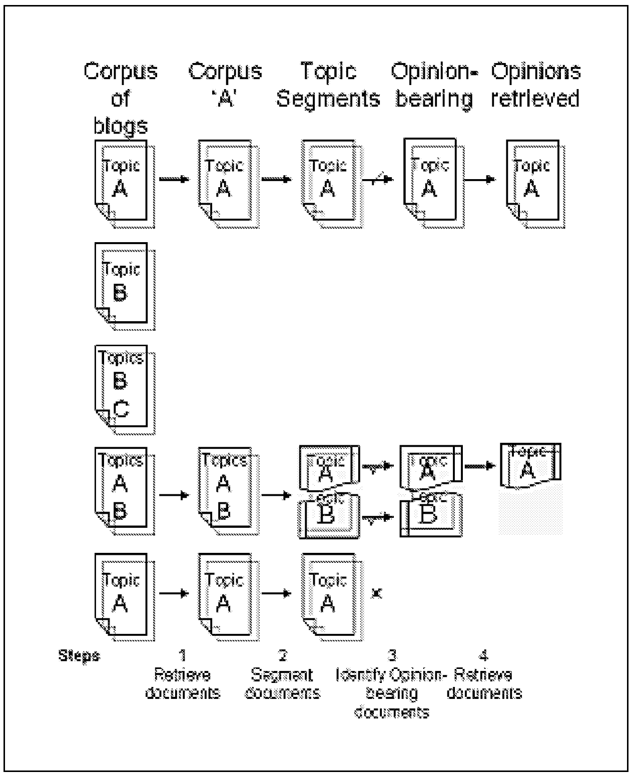


Figure 1: Process of retrieving opinions. Steps: 1 – Retrieve documents matching query. 2 – Segment documents into topic-blocks. 3 – Identify opinion-bearing topic-blocks. 4 – Retrieve opinion-bearing topic-blocks matching query

Document retrieval is considered to be well researched and developed, leaving two main areas of development, topic segmentation and opinion identifying, to complete this research.

3.1 Topic Changes Segmentation

An experiment has been developed to determine the most accurate algorithm to use when automatically dividing a large corpus of web logs into individual topics. One document containing all web logs is divided into smaller segments and these segments are compared to their adjacent segment to calculate a similarity score (Hearst & Plaunt 1993, Hearst 1997).

3.1.1 Dataset

The dataset was constructed by combining 11 web logs, collected from the World Wide Web, into one continuous text document (totalling 3,506 words). The web logs all contained a general topic of Skype with different areas within Skype being discussed in each web log.

The blogs ranged from 63 to 1140 words in total with some blogs containing sub-headings. The dataset used by Hearst (1997) was 12 magazine articles (length between 1,800 and 2,500 words) which were evaluated using paragraph breaks to indicate the new sub-topic.

3.1.2 Dividing the Document into Segments

The informal nature of the language, punctuation and structure of a web log meant that dividing the document into paragraphs, as used by Hearst & Plaunt (1993), created segments ranging from one word to 179 words. This approach was considered not applicable to web logs.

The document was divided into four different segment sizes using either orthographically marked segments or even-sized blocks (Hearst & Plaunt 1993). The orthographical dividing was done using single sentence and three sentence blocks. Even-sized blocks of 30 words and 60 words were also created.

3.1.3 Numerically Representing Segments

Vectors were created to represent each term within the dictionary of the text document, each segment was represented using one of three variations of the vector space model.

- Binary – If a term appeared in the segment, a ‘1’ was placed in the vector, otherwise a ‘0’ appeared. This did not account for the number of times a term appeared within the segment.
- Term Frequency – The number of times a term appeared in the segment represented the term within the vector. Term Frequency was considered important when comparing the vocabulary used within each segment to the adjacent segment.
- tf.idf – This determines the overall importance of the terms in the segments in relation to the entire corpus. The formula used in this research is:

$$w_{ij} = tf_{ij} \cdots log\frac{N}{df_j}$$

3.1.4 Calculating Similarity Scores

Similarity scores were calculated using segment comparison and block comparison (Hearst 1997). In segment comparison, adjacent segments were compared to calculate a similarity score[1]. Figure 2 (Hearst 1997) illustrates block comparison for binary vectors which uses two segments compared to the adjacent two segments.

Block 1 – A:2*1, B:1*1, C:2*1, D:1*1, E:1*1
Block 2 – A:1*1, B:1*1, C:1*1, D:1*1, E:2*1

The similarity score between block 1 and block 2 is 8. The blocks are moved along one segment so that block 1 contains segments two & three and block 2

[1]Similarity scores are calculated by taking the inner product of adjacent vectors

contains segments 4 & 5. The blocks are continuously moved along until all gaps have been allocated a similarity score. This calculation method was then applied to term frequency and tf.idf.

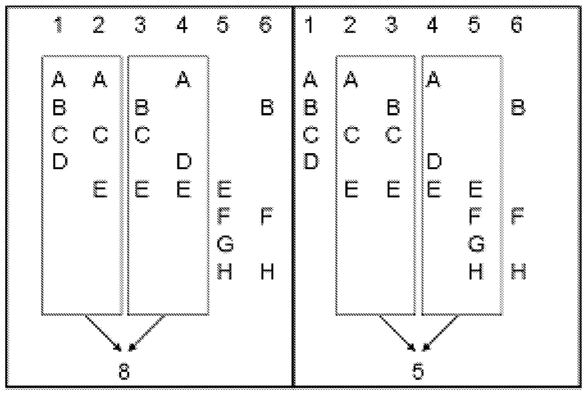


Figure 2: Illustration of block comparison technique

These calculations created six sets of similarity scores. In addition to these sets of similarity scores, an extra two sets were calculated using normalisation of tf.idf to account for the varying lengths of sentences.

The similarity scores calculated using segment comparison showed many small variations in topics. Although the block comparison did not have as many small variations, smoothing[2] was applied to the block comparison scores. Four sets of graphs were created comparing the four segment size scores, these graphs used the binary, term frequency, tf.idf and tf.idf(N) similarity scores.

3.1.5 Evaluation

Human evaluators were asked to read the document and mark the start of each new topic within the document. The topic changes marked by the evaluators were added to the graphs to enable analysis of which calculation method is the most accurate. The results of this evaluation are discussed in section 4.1.

3.2 Opinion Identification

The results Topic Change experiment (discussed in section 4) were used to determine the algorithm for dividing the blogs into the topic-blocks, which will be used for opinion identification. A larger data set is used for the remainder of this research.

3.2.1 Dataset

The process of combining the web logs into one continuous document and dividing the document into topics for opinion identification was tested on 200 web logs, which were collected from the world wide web and stored in one continuous text document. The TREC Blogs06 data will be used in future research using the approach developed in this research.

The TREC Blogs06 test collection is a collection of web logs collected over an 11-week period and is made up of 'Feeds', 'Permalinks' and 'Homepages'. The total size of the collection is 23GB compressed (>150GB uncompressed) including spam and possibly non-blogs. 'Feeds' are XML-based files which contain a machine-readable version of the blog, 'Permalinks' are a type of permanent link that can be used to find the blog in the future, while 'Homepages' are the files that are browsed on the world wide web.

[2]Smoothing is used to reduce the small variations in the similarity scores. The average of the current, previous and next score is calculated using the following formula: $s_i = \frac{(s_{i-1})+(s_i)+(s_{i+1})}{3}$

3.2.2 System

A dictionary was created for the corpus using the words contained in the blogs without stemming. Stopping was performed to remove non-content-bearing words. The dataset was divided into single sentences (segments) and the Vector Space Model (Salton 1989) was used to create a vector, containing Term Frequencies, to represent each segment.

The block method, explained in section 3.1.4, was used to calculate similarity scores. Smoothing was applied to the similarity scores and the corpus was divided into topic-blocks using the dips in the scores, creating 975 topic-blocks.

To determine which topic-blocks were opinion-bearing, a list of opinion-bearing words with weights was requested from Yu & Hatzivassiloglou (2003). The weights were applied to the words within each topic-block to identify which of the topic-blocks were opinion-bearing. Two approaches to identify opinion-bearing topic-blocks, as developed by Kim & Hovy (2005), were used:

- **Approach 1**: Sum the weights for all words used in the topic-block.

  Each word within the topic-block was compared to the list of weighted words. If the word was present in the weights list, the weight was recorded against the topic-block. Once all words within the topic-block had been compared to the weights list, the sum of the weights was recorded for that topic-block.

- **Approach 2**: Identify topic-blocks containing words with the highest weight.

  This approach was similar to Approach 1, except the highest weight was recorded against the topic-block instead of the sum of all the weights. This approach aimed at finding the topic-block with the strongest opinion-bearing word.

3.2.3 Evaluation

Two evaluators were asked to assess lists of the top 100 opinion-bearing topic-blocks. The topic-blocks in the lists were identified by the sum of all of the opinion-bearing weights (Approach 1) and the topic-blocks with the highest individual opinion-bearing weight (Approach 2) respectively. The evaluators were given the instructions to mark a topic-block as an opinion if any part of the topic-block expressed an opinion. The results of this evaluation are discussed in section 4.2.

4 Results and Discussion

The research discussed in section 3 has resulted in the following findings. The topic segmentation results are discussed in section 4.1 while the opinion retrieval results are discussed in section 4.2.

4.1 Topic Change Identification

A document was created containing the 11 blogs used in the topic segmentation experiment with the titles of the blogs and the headings within the blogs in normal font. This removed any visual indication of where one

Table 1: Agreement Between Evaluators

Evaluators	Agreement	Kappa
Initial evaluation of topic changes		
a & b	94%	0.76 κ
a & c	82%	0.36 κ
b & c	82%	0.39 κ
Re-evaluation of topic changes		
a & b	95%	0.80 κ
a & c	87%	0.57 κ
b & c	87%	0.58 κ

blog finished and the next blog started. Three human evaluators were asked to indicate the position of topic changes within the document, this did not necessarily mean at the end of a sentence or paragraph. The evaluators commented that the document was difficult to mark topic changes as some topics changed gradually and some sentences contained topics of their own.

Agreement between two of the evaluators was high, while the third evaluator marked many topic changes which were within one sentence of the other evaluators. The three evaluators were asked to re-examine their topic changes where two evaluators agree and the third evaluator was within one sentence. The evaluators agreed that the topic changes were ambiguous and could be marked at the next or previous sentence.

Table 1 shows the agreement and Kappa values between evaluators of the first examination and the modified values after re-examination. Topic-blocks ranged from one sentence up to 12 sentences, smaller than the topic-blocks found by Hearst (1997), which spanned much larger blocks of text.

Using the topic changes determined by the agreement of two or three evaluators, the original graphs were marked with a vertical line to enable the researcher to visually determine the agreement between evaluators and the computed topic changes. Figure 3 is an example of the graphs produced comparing the similarity scores and the topic changes marked by the evaluators (vertical lines).

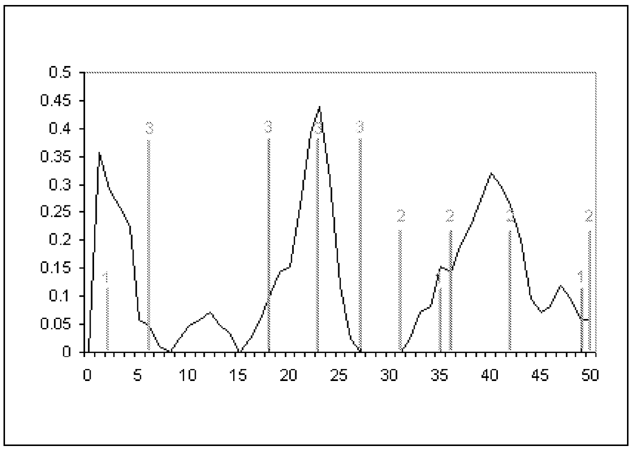


Figure 3: Example of graphed Similarity Scores with vertical lines indicating human evaluators topic changes. Note: The length of the line indicates the number of evaluators marking the segment as a topic change.

The graphs show that the topic changes indicated by the similarity scores often disagree with the topic changes indicated by the evaluators, however it is common that the topic changes are within one sentence of agreeing. This highlights the difficult task of dividing blogs into topic-blocks because the topics change slowly over several sentences, or the topic changes temporarily for one or two sentences and then changes back to the original topic.

In the example graph shown (Figure 3) three evaluators marked a topic change at the end of segment six, while the graph shows a topic change at the end of segment eight. The two segments in question here are a one word sentence and a segment containing the heading of the web log. The document given to the evaluators did not indicate whether text was a heading, or where one web log ended and the next one started. In this particular example whether the topic change is marked at the end of segment six, seven or eight would not change the content of the topic-block created.

The following advantages (A) and disadvantages (D) of each segment size were found:

- **Single sentences**:
  - (A) Topic changes invariably change at the end of a sentence.
  - (D) The similarity scores changed dramatically causing the graphs to have many small spikes.
- **Three sentence blocks, 30-Word blocks & 60-Word blocks**:
  - (A) Graphs were smoother (less spikes).
  - (D) The evaluator's topic-change often fell within the block.

The similarity score graphs showed no noticeable variation in the pattern between using the binary representation, term frequency or tf.idf. With the ambiguous nature of topic change within blogs in mind, and the belief that topics won't change mid-sentence, single sentence segments were used in combination with term frequency to calculate similarity scores for the remainder of the research.

4.2 Opinion Identification

Topic segmentation was applied to 200 blogs to divide them into 975 topic-blocks. A list of words (Yu & Hatzivassiloglou 2003), weighted by their strength of opinion-bearing, was used to allocate weights to the topic-block using two approaches: Approach 1 – the sum of the weights & Approach 2 – the highest weight of a single word within the topic-block.

The top 100 opinion-bearing blogs from each approach were given to evaluators, who marked the topic-blocks as opinion-bearing or non-opinion-bearing. The evaluators generally agreed upon the whether the topic-blocks were opinion-bearing or non-opinion bearing, figure 4 shows the comparison between the proposed approaches and the results of the human evaluators.

- Human Evaluation – Sum of Weight

  The evaluators were given a list of top 100 topic-blocks that scored the highest sum of the opinion-bearing weights. The agreement between the evaluators was 92% with a Kappa score of .81κ. Overall, the precision of the top 100 topic-blocks was 67%. A limitation of this approach is that it is not normalised to account for the varying sizes of the topic-blocks (larger topic-blocks have more terms which are allocated weights).

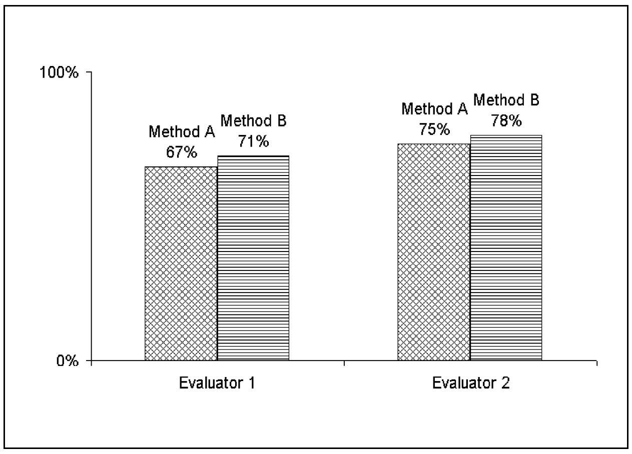


Figure 4: Agreement between proposed opinion identification approaches and human evaluators

- Human Evaluation – Highest Weight

  The evaluators were given a list of top 100 topic-blocks that contained the highest single weight of the opinion-bearing weights. The agreement between the evaluators was 91% with a Kappa score of .76κ. Overall, the precision of the top 100 topic-blocks was 70%.

The opinion identification precision was higher for the topic-blocks that contained a single word with the highest opinion weight (70%) compared to the sum of the weights for a topic-block (67%). This is in line with results found by Kim & Hovy (2005), who found that using the single opinion-bearing term with the highest weight produced better results. However, the precision of this research was lower than the precision of previous research conducted using formal, structured documents (Kim & Hovy 2005, Yu & Hatzivassiloglou 2003).

5 Conclusions and Future Work

This preliminary research implemented methodologies used in previous research (Hearst & Plaunt 1993, Hearst 1997, Yu & Hatzivassiloglou 2003, Kim & Hovy 2005) on formal, structured documents and combined them to test the algorithms on unstructured, informal blogs. The accuracy of the topic segmentation results were not as high as expected.

However, using the topic segmentation algorithm without human intervention worked well in the ensuing research. The opinion identification research resulted in a precision result lower than that of previous research (Kim & Hovy 2005, Yu & Hatzivassiloglou 2003), which may highlight a need for refinement of the adjective list in future research.

Two methods were applied to opinion identification, Method A (sum of opinion-bearing weights) and Method B (the highest weight within a document). Opinion identification within topic-blocks resulted in precision of 67% for Method A and 70% for Method B.

Topic segmentation was originally created using formal, structured documents to divide blogs into topic-blocks. The original research (Hearst 1997) used a single large document to test the algorithm, while the blogs used in this research were much shorter in length and varied topics more frequently.

This type of topic segmentation does not encompass the structure of a blog where the text within the blog may change topic several times and the feedback/comments will relate to an earlier topic, which may lead to future work on finding topic blocks within a blog and combining them. The nature of blogs and the structure will be researched to assist in the development of a topic segmentation algorithm that deals specifically with blogs.

The topic-segmentation of blogs was not as precise as expected due to evaluators marking topic changes either before or after the blog title, or blogs changing to the original topic within one or two sentences. An alternative to topic-segmentation to be considered in future work is passage retrieval (Kaszkiel & Zobel 1997), however passage retrieval spans over orthographic features which may be inappropriate to opinion retrieval.

The topic segmentation algorithm was then used in the opinion identification stage of the research without human intervention and the evaluators of the opinion identification task commented on the rarity of topic-blocks having multiple topics. This showed that the topic segmentation algorithm worked well, although there were small discrepancies in the topic change boundaries noted by the human evaluators.

The entire data-set used in the opinion identification section of the research has not been fully evaluated, this removes the ability to calculate recall on the results. However, using evaluators to determine whether the top 100 topic-blocks were opinion-bearing enabled precision to be calculated.

The low number of evaluators may have contributed to the high precision of the opinion identification task. Future work will involve TREC Blogs06 data which will be evaluated prior to the research commencing.

The algorithm for dividing documents into topic-blocks is computationally expensive which may cause concerns in overall retrieval times for the entire opinion retrieval process using TREC data. An important step in the process described in Figure 1 is the initial document retrieval, creating a mini-corpus, which is used in topic segmentation. Future research will investigate document retrieval models to determine which is most efficient for this research. Alternatively, consideration will be given to whether topic segmentation is included in the final opinion retrieval process. Using an entire web log as a document or using headings and titles to indicate topic changes will be researched as alternatives.

These results will be used to develop an approach to retrieving opinion-bearing topics from within a blog. The TREC Blogs06 data will be used in this research. The process discussed in section 3 and illustrated in figure 1 will be implemented.

The proposed steps are to index the dataset and retrieve documents matching a query (creating a mini corpus). Segment the blogs in the mini corpus into topic-blocks (expanding the mini corpus), identify the topic-blocks as opinion-bearing or non-opinion-bearing, and remove the non-opinion-bearing topic-blocks. Index the remaining opinion-bearing topic-blocks and retrieve the topic-blocks that match the query in the expanded mini corpus.

References

Currently tracking 48.8 million sites and 2.7 billion links (2006).
*http://www.technorati.com/about/

Eguchi, K. & Lavrenko, V. (2006), Sentiment retrieval using generative models, *in* 'Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP 2006)', pp. 345–354.

Framenet (2002).
*http://www.icsi.berkeley.edu/ framenet/

Hatzivassiloglou, V. & McKeown, K. (1997), Predicting the semantic orientation of adjectives., *in* 'ACL', pp. 174–181.

Hearst, M. A. (1997), 'Texttiling: segmenting text into multi-paragraph subtopic passages', *Comput. Linguist.* **23**(1), 33–64.

Hearst, M. A. & Plaunt, C. (1993), Subtopic structuring for full-length document access, *in* 'SIGIR 93: Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval', ACM Press, New York, NY, USA, pp. 59–68.

Kaszkiel, M. & Zobel, J. (1997), Passage retrieval revisited, *in* 'SIGIR '97: Proceedings of the 20th annual international ACM SIGIR conference on Research and development in information retrieval', ACM Press, New York, NY, USA, pp. 178–185.

Kim, S.-M. & Hovy, E. (2005), Automatic detection of opinion bearing words and sentences, *in* 'Natural Language Processing - IJCNLP 2005', Springer, New York.

Ku, L.-W., Lee, L.-Y., Wu, T.-H. & Chen, H.-H. (2005), Major topic detection and its application to opinion summarization, *in* 'SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval', ACM Press, New York, NY, USA, pp. 627–628.

Ku, L.-W., Liang, Y.-T. & Chen, H.-H. (2006), Tagging heterogeneous evaluation corpora for opinionated tasks, *in* 'LREC 2006'.

Leshed, G. & Kaye, J. J. (2006), Understanding how bloggers feel: recognizing affect in blog posts, *in* 'CHI '06: CHI '06 extended abstracts on Human factors in computing systems', ACM Press, New York, NY, USA, pp. 1019–1024.

Levin, B. (1993), *English Verb Classes and Alternations: A Preliminary Investigation*, University of Chicago Press, Chicago.

Lin, D. (1994), Principar-an efficient, broad-coverage, principle-based parser, *in* 'The 15th International Conference on Computational Linguistics Proceedings COLING '94', pp. 482–488.

Nanno, T., Fujiki, T., Suzuki, Y. & Okumura, M. (2004), Automatically collecting, monitoring, and mining japanese weblogs, *in* 'WWW Alt. 04: Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters', ACM Press, New York, NY, USA, pp. 320–321.

Nardi, B. A., Schiano, D. J., Gumbrecht, M. & Swartz, L. (2004), 'Why we blog', *Commun. ACM* **47**(12), 41–46.

Riloff, E. & Jones, R. (1999), Learning dictionaries for information extraction by multi-level bootstrapping, *in* 'AAAI '99/IAAI '99: Proceedings of the sixteenth national conference on Artificial intelligence and the eleventh Innovative applications of artificial intelligence conference innovative applications of artificial intelligence', American Association for Artificial Intelligence, Menlo Park, CA, USA, pp. 474–479.

Riloff, E., Patwardhan, S. & Wiebe, J. (2006), Feature subsumption for opinion analysis, *in* 'Conference on Empirical Methods in Natural Language Processing (EMNLP-2006)'.

Riloff, E. & Wiebe, J. (2003), Learning extraction patterns for subjective expressions, *in* 'Conference on Empirical Methods in Natural Language Processing (EMNLP-03). ACL SIGDAT', pp. 105–112.

Rosenbloom, A. (2004), 'Introduction: The blogosphere', *Commun. ACM* **47**(12), 30–33.

Salton, G. (1989), *Automatic Text Processing - The Transformation, Analysis, and Retrieval of Information by Computer*, Addison-Wesley, Reading, Mass.

Searls, D. & Sifry, D. (2003), 'Building with blogs', *Linux J.* **2003**(107), 4.

Thelen, M. & Riloff, E. (2002), A bootstrapping method for learning semantic lexicons using extraction pattern contexts, *in* 'EMNLP '02: Proceedings of the ACL-02 conference on Empirical methods in natural language processing', Association for Computational Linguistics, Morristown, NJ, USA, pp. 214–221.

Wiebe, J. (2000), Learning subjective adjectives from corpora, *in* 'Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-2000)', AAAI Press, Menlo Park, California, pp. 735–741.

Wiebe, J., Wilson, T., Bruce, R., Bell, M. & Martin, M. (2004), 'Learning subjective language', *Comput. Linguist.* **30**(3), 277–308.

Wilson, T., Pierce, D. R. & Wiebe, J. (2003), Identifying opinionated sentences, *in* 'NAACL '03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology', Association for Computational Linguistics, Morristown, NJ, USA, pp. 33–34.

Wilson, T. & Wiebe, J. (2003), Annotating opinions in the world press, *in* 'Proceedings of the 4th SIGdial Workshop on Discourse and Dialog (SIGdial-03)', pp. 13–22.

Yu, H. & Hatzivassiloglou, V. (2003), Towards answering opinion questions: separating facts from opinions and identifying the polarity of opinion sentences, *in* 'Proceedings of the 2003 conference on Empirical methods in natural language processing', Association for Computational Linguistics, Morristown, NJ, USA, pp. 129–136.

Distributed Text Retrieval From Overlapping Collections

Milad Shokouhi **Justin Zobel** **Yaniv Bernstein**

School of Computer Science and Information Technology
RMIT University,
PO Box 2476V, Melbourne, 3001, Australia
Email: {milad,jz,ybernste}@cs.rmit.edu.au

Abstract

In standard text retrieval systems, the documents are gathered and indexed on a single server. In distributed information retrieval (DIR), the documents are held in multiple collections; answers to queries are produced by selecting the collections to query and then merging results from these collections. However, in most prior research in the area, collections are assumed to be disjoint. In this paper, we investigate the effectiveness of different combinations of server selection and result merging algorithms in the presence of duplicates. We also test our hash-based method for efficiently detecting duplicates and near-duplicates in the lists of documents returned by collections. Our results, based on two different designs of test data, indicate that some DIR methods are more likely to return duplicate documents, and show that removing such redundant documents can have a significant impact on the final search effectiveness.

Keywords: Distributed information retrieval, duplicate and near-duplicate detection, similarity measurement, search engines

1 Introduction

In standard text retrieval systems, the collection of documents is indexed at a single location and made available to users through a search interface. Such centralized information retrieval (IR) is efficient and effective when the documents can be gathered together, but such consolidation is not always possible. On the web, for example, many documents are not crawlable and can only be accessed by that particular site's search interface. Such documents form the *hidden web*, which has been reported to be many times larger than the fraction of the web that can be crawled (Bergman 2001). Also, crawling the web can be prohibitively slow; some documents are held on servers whose delivery speed makes crawling impractical.

A solution is provided by distributed information retrieval (DIR), in which documents held on multiple servers are made available through a single interface. Distributed search over multiple collections—also known as federated search—has been an active area of research for over a decade. In addition to making hidden-web documents centrally accessible, there are other gains; for example, with DIR the costs of crawling can be avoided, and distributed indexes can be updated more easily than the alternatives.

In DIR, queries are submitted to a central component, known as the *broker*. For efficiency, the broker sends the query to only a limited number of collections—creating the problem of collection selection. In order to select suitable collections for each query, the broker should acquire information about each collection in advance—creating the problem of collection representation. A collection's *representation set* is the broker's knowledge about each collection. Queries are compared with representation sets, and collections that are more likely to return relevant documents are selected (Callan, Lu & Croft 1995, Gravano, Garcia-Molina & Tomasic 1999). The selected collections return their answers to the broker, which merges the results and presents them to the user (Callan et al. 1995, Si & Callan 2003*b*)—creating the problem of result merging.

In *cooperative* environments, collections provide the broker with comprehensive information about their indexes, typically comprised of data such as lexicon statistics and collection size, allowing the broker to effectively select suitable collections for a query (Allan et al 2003, Meng, Yu & Liu 2002). In *uncooperative* environments, collections do not provide such information. To identify likely collections, the broker must first create sample surrogates by downloading a limited number of documents from collections (Callan, Connell & Du 1999, Callan & Connell 2001, Ipeirotis & Gravano 2004, Ipeirotis & Gravano 2002). In *semi-cooperative* environments, the broker has incomplete knowledge about the collections. However, it might receive supplementary information such as document scores from collections, allowing more accurate results merging (Callan et al. 1995, Si & Callan 2003*b*).

Many collection representation techniques (Callan et al. 1999, Callan & Connell 2001, Ipeirotis & Gravano 2004, Ipeirotis & Gravano 2002, Shokouhi, Scholer & Zobel 2006), collection selection techniques (Callan et al. 1995, Gravano et al. 1999, Nottelmann & Fuhr 2003, Si & Callan 2003*a*, Si & Callan 2004, Yuwono & Lee 1997, Zobel 1997), and result merging techniques (Callan et al. 1995, Kirsch 2003, Si & Callan 2003*b*) have been investigated. However, in almost all reported experiments, it is assumed that collections do not overlap, so that any given document is only available in one collection.

In practice, however, this assumption is frequently valid. For example, many research articles are indexed by both the ACM portal[1] and the IEEE Xplore[2] digital libraries. Therefore, a broker that sends queries to both online collections may receive many duplicate documents. Detecting such duplicates is far from straightforward, as they may be delivered from different URLs and have slight changes, such as in their metadata or presentation, that do not

[1] http://portal.acm.org
[2] http://ieeexplore.ieee.org

alter the content but mean that the documents are not identical. Retrieval of all candidate documents by the broker to check for duplication imposes significant expense—in standard DIR the broker need not inspect the documents—while, as we show, failure to eliminate duplicates can significantly degrade the quality of answers returned.

Although distributed search among overlapped collections has been suggested as one of the major issues in DIR (Allan et al 2003, Meng et al. 2002), the problem has not been thoroughly explored. We have previously proposed that *grainy hash vectors* (GHVs) can be used for removing duplicate and near-duplicate documents from result lists (Bernstein, Shokouhi & Zobel 2006). However, these experiments used a centralized index. Moreover, in this work the performance of DIR techniques in the presence of duplicates and near-duplicates is not comprehensively investigated.

In this paper, we use GHVs on distributed testbeds to remove duplicate and near-duplicate documents from the results. We also analyze the behavior of well-known DIR collection selection and result merging strategies on overlapped collections. As a basis for this work, we propose two mechanisms for creating experimental testbeds from standard test collections. One, based on windowing, is somewhat artificial but allows precise setting of the level of duplication, thus allowing us to explore the impact of parameters in a controlled way; the other, based on query result sets, generates collections with some consistency of topic. To our knowledge, such a study on overlapped collections has not previously been undertaken.

In our experiments, we find that GHVs are effective at identifying duplicates in answer sets, and that removal of such duplicates can have a substantial impact on effectiveness. We also find that relative performance of the collection selection and result merging methods depends to some extent on the testbed used, demonstrating that care is required in the development of resources and interpretation of outcomes for such research questions.

2 Distributed information retrieval

In DIR, the broker receives each query, distributes it to selected servers, and merges results to give a consolidated list to return to the user. Before the broker can begin processing queries, it must gather a representation set for each collection. In cooperative protocols such as STARTS (Gravano, Chang, Garcia-Molina & Paepcke 1997), collections provide the broker with comprehensive information about their indexes including the lexicon statistics and the number of documents. However, access to such a representation set for each collection may not be possible; in uncooperative environments, the broker must download a limited number of documents from each collection and use them as a representation set.

Query-based sampling (QBS) was proposed by Callan et al. (1999) for uncooperative environments. In this approach, an initial query is selected from a list of frequent keywords and is submitted to a collection. The top N documents returned by the collection are downloaded by the broker and the next query is picked, usually at random, from the contents of these documents. This process continues until a sufficient number of documents (k) has been downloaded. Callan et al. (1999) claimed that $N = 4$ and $k = 300$ are suitable values; in contrast, we have suggested that the rate of discovery of new terms in the downloaded documents is a suitable indicator of when sampling should stop (Shokouhi et al. 2006). In this paper, we use the values suggested by Callan et al. (1999), to make our results comparable to related work in this area (Nottelmann & Fuhr 2003, Si & Callan 2003*b*, Si & Callan 2003*a*, Si & Callan 2004).

Once representation sets are created, the broker can use them to pass queries to collections that are deemed as likely to contain relevant documents. In some collection selection techniques, the broker uses variants of standard IR measures to compute the similarity of a query with collection representation sets. CORI (Callan et al. 1995), GlOSS (Gravano et al. 1999) and CVV (Yuwono & Lee 1997) are a few well-known examples of such approaches. Among these, CORI was reported to produce the highest precision (Craswell, Bailey & Hawking 2000, Powell & French 2003, Rasolofo, Abbaci & Savoy 2001).

In CORI, the belief $P(t|c)$ for observing a query term t in collection c is computed as below:

$$P(t|c) = d_b + (1 - d_b) \times T_c \times I_c$$

$$T_c = d_t + (1 - d_t) \cdot \frac{\log(f_{t,c} + 0.5)}{\log(max_c(f_t) + 1.0)}$$

$$I_c = \frac{\log(\frac{N+0.5}{cf})}{\log(N + 1.0)}$$

where $f_{t,c}$ is the document frequency of t in c and cf is the number of collections containing the term t. d_t is the minimum term frequency component while d_b is the minimum belief component when t occurs in c. d_t and d_b are respectively set to 0.4 and 0.5 by default. N represents the total number of collections while $max_c(f_t)$ is the maximum document frequency in c. The final weight of a collection c for query Q is calculated by summing up the computed beliefs $P(t|c)$ for all query terms $t \in Q$.

Although more recent studies suggest that CORI is generally worse than other methods on many testbeds (D'Souza, Zobel & Thom 2004, Si & Callan 2003*a*), it is still used frequently for collection selection research (Si & Callan 2004, Si & Callan 2005, Avrahami, Yau, Si & Callan 2006). Thus we use CORI as one of our collection selection methods in this paper.

In recent years, new collection selection algorithms have been shown to produce better results than CORI (Hawking & Thomas 2005, Si, Jin, Callan & Ogilvie 2002, Si & Callan 2003*a*, Si & Callan 2004, Si & Callan 2005) on some testbeds. In HARP (Hawking & Thomas 2005), the anchor texts available in a large crawled repository are used to create the representation sets, which for each collection consists of the anchor texts of URLs available in the crawled data that are targeting the collection. This is similar to Q-pilot (Sugiura & Etzioni 2000).

ReDDE (Si & Callan 2003*a*) ranks collections according to the estimated number of relevant documents. The broker creates a central index of all sampled documents, then each submitted query is evaluated on this index before being sent to collections. The number of relevant documents in each collection is estimated from the contribution of collections in the top-ranked documents. UUM (Si & Callan 2004) estimates the probability of relevance of documents inside each collection using training queries and their related relevance judgments. UUM was reported to produce better results than ReDDE (Si & Callan 2004).

RUM (Si & Callan 2005) is a variant of UUM that also considers the search effectiveness of collections. As in ReDDE, RUM maintains a central index of all sampled documents on the broker. In the training stage, the documents returned by collections for queries are compared to those ranked by the central index. In addition, the broker downloads a few top-ranked documents from each collection and calculates their weights in the central index. Based on the weights of downloaded documents for the training queries, the broker then approximates the search effectiveness of each collection. Si & Callan (2005) suggest that RUM can slightly outperform UUM.

RUM and UUM have significant drawbacks: both require a set of training queries and relevance judgments. RUM also downloads documents from each collection for the training queries. In practical situations, relevance judgments may be costly and downloading documents might be infeasible. Therefore, we use ReDDE (Si & Callan 2003*a*) as a practical representative of recent collection selection algorithms.

Selected collections return their answers to the broker. The broker then merges the results and represents them to the user.

The document score values reported by collections are not comparable as they are computed by different retrieval models and rely on inconsistent lexicon statistics. The goal of result merging algorithms is to calculate a global score for each document that is comparable to the scores of documents returned by other collections.

In CORI result merging (Callan et al. 1995), the global score D_G of a document d returned by a collection (c) is calculated according to its normalized document score (D') and collection score (C'). The former is the *collection-specific* weight of d reported by its original collection (c) and the latter is the weight of c calculated by the broker.

$$C' = \frac{(C - C_{min})}{(C_{max} - C_{min})}$$

$$D' = \frac{(D - D^c_{min})}{(D^c_{max} - D^c_{min})}$$

$$D_G = \frac{D' + 0.4 \times D' \times C'}{1.4}$$

where D^c_{min} and D^c_{max} are respectively the minimum and maximum document scores reported by collection c, and C_{min} and C_{max} are the minimum and maximum weights that can be assigned to any collection by the broker during collection selection.

SSL (Si & Callan 2003*b*) uses a semi-supervised learning method to create a model for each collection that maps document scores into global scores. SSL creates a central index of all sampled documents. For a query, some of the documents returned by any collection c might be already available in the central index. SSL compares the central weights of such overlap documents with the reported scores from c and then approximates the weights of other documents that are not available in the central index.

When collections use an identical retrieval model, all overlap documents can be used to train a single linear model that maps collection-specific scores into approximated global scores. In such a scenario, for an overlap document d returned from any selected collection c, two scores are available: D is the score reported by the original collection and D_S is the weight of document computed by the central, sample-based index.

Using the values for D and D_S, SSL trains a linear model that converts the reported scores by collections to their approximated central scores, with parameters a and b used to combine scores as follows:

$$D_G = a \times D + b \times D_S \times C$$

where C is the weight of collection c from which d is drawn. Given this information, SSL can accurately approximate the central scores of documents. The central index can be assumed to be representative of the global information, so the weights of documents in the central index are likely to be representative of their global scores.

For simplicity, we assume that all collections in our experiments are using INQUERY (Callan, Croft & Harding 1992) as their retrieval model. Thus, we use SSL single-model as one of our merging methods.

CORI and SSL are intended for semi-cooperative environments where document scores are broadcast by collections. In the absence of document scores, they calculate the pseudoscores of documents as described by Si et al. (2002). Other merging strategies (Kirsch 2003, Si et al. 2002) typically follow the same strategy but are not as well-known as CORI and SSL. Therefore, in our experiments we use CORI and SSL for merging.

DIR merging is not equivalent to data fusion or the merging problem in metasearch (Croft 2000, Fox & Shaw 1994, Lee 1997). In data fusion, different ranking functions are applied to the same collection (Meng et al. 2002). In the presence of multiple collections, queries are usually sent to all of them without collection selection. Therefore, in data fusion or metasearch merging, collection representation sets are not usually required. Also, documents are merged solely based on their ranks or reported scores by collections.

3 Overlapping collections

Distributed retrieval from overlapped collections has been described as one of the current challenges in IR (Allan et al 2003, Meng et al. 2002). However, all of the methods discussed above assume that collections do not have overlapped documents or that the number of duplicates is negligible.

Metasearch engines such as ProFusion (Gauch, Wang & Gomez 1996), MetaCrawler (Selberg & Etzioni 1997), and Grouper (Zamir & Etzioni 1999) remove duplicate documents from the results by aggregating those that point to the same URL. The rank of a duplicate document in the final result is calculated according to its position in the ranklists of different search engines. Typically, this involves use of methods such as CombSUM or ComMNZ (Fox & Shaw 1994), or other similar approaches (Lee 1997). Some researchers argue that such methods cannot be defined in the context of DIR and should be described in the broader category of metasearch (Si & Callan 2003*b*).

These metasearch methods have two major drawbacks; they cannot detect and remove near-duplicates and they are unable to distinguish exact duplicate documents with different URLs (mirror URLs). Moreover, a recent study (Wu & McClean 2006) suggests that when the rate of overlap between the final results of collections is less than 60%, the performance of such methods significantly degrades.

To our knowledge, the only discussion of removal duplicates and near-duplicates during merging in DIR is that given by Bernstein et al. (2006). This approach is discussed in detail in the following sections.

Duplicate documents can also be avoided by using an overlap-aware collection selection method. Hernandez & Kambhampati (2005) introduced COSCO, which considers the rate of overlap among collections during collection selection. For a query, COSCO does not select two collections that are likely to return many identical documents. The approach, although interesting, has defects. COSCO requires a large number of training queries to learn the rates of overlap between collections for each topic. In addition, COSCO was not tested for detection of near-duplicate documents, which, as discussed below, is a much more challenging problem (Zobel & Bernstein 2006).

We argue that removal of duplicates during merge is more appropriate because overlap-aware collection selection methods may be lossy. That is, ignoring a collection that includes unvisited relevant documents can affect the final precision. In contrast, if duplicate removal occurs during merging, all relevant documents can be retrieved by the broker. We now explore methods for detection and removal of duplicates.

4 Detection of duplicates and near-duplicates

There are many sources of duplication in text collections. For example, a crawl of documents harvested from the web may yield duplicates due to factors including URL aliasing; copies of the same document held at several mirrors; each author of a paper placing a copy on their website; republication of news stories by multiple online newspapers; and commercial websites presenting local copies of public documents.

Exact copies are easy to detect, but many duplicates are not exact copies. A newspaper that republishes a newswire article may edit it to reflect local knowledge, and the page context such as advertising and 'related story' links is likely to be different. Some stories are regularly recycled, such as birthday notices and Groundhog Day features. Policy documents and legislation may differ little from jurisdiction to jurisdiction, as policymakers adopt models from elsewhere. Ultimately, the question of whether two documents are duplicates is highly application-dependent—for example, in some contexts two documents that differ only in publication date may be regarded as having a significant difference. However, we have found that, in the context of search, mechanisms for detecting duplication such as those described in this section are consistent with user judgements as to whether documents are duplicates or near-duplicates. (Bernstein & Zobel 2005, Zobel & Bernstein 2006).

In the context of search, duplicate detection can take place during either indexing or retrieval. At indexing time, duplicate detection involves processing the entire collection to find pairs of documents that appear to be duplicates or near-duplicates (Manber 1994, Brin, Davis & García-Molina 1995, Broder, Glassman, Manasse & Zweig 1997, Fetterly, Manasse & Najork 2003, Bernstein & Zobel 2004). In a DIR system, such an approach is unlikely to be feasible, particularly in uncooperative environments. Thus we must focus on methods that can be used to eliminate duplicates from answer lists. However, we wish to avoid adding significant costs to the query evaluation mechanism. A DIR duplicate-detection mechanism that involved fetching all of the answer documents from each collection is not attractive.

As we have described elsewhere (Bernstein et al. 2006), the most suitable methods described in previous literature are based on computing a brief descriptor of each document. In a semi-cooperative environment, each collection could return these descriptors, allowing efficient duplicate detection. We now summarize our previous analysis of prior methods.

A descriptor-based approach that is superficially attractive is to use deterministic term extraction to identify terms in each document that are deemed likely to be indicative of duplication (Chowdhury, Frieder, Grossman & McCabe 2002, Ilyinski, Kuzmin, Melkov & Segalovich 2002, Cooper, Coden & Brown 2002, Conrad, Guo & Schriber 2003, Kolcz, Chowdhury & Alspector 2004). The assumption is that near-duplicates will share an exact set of such terms, so that hashing them will produce a descriptor that can be matched extremely fast. This term extraction process must take place at index time; otherwise, query processing would be much more expensive.

However, such approaches can only succeed if they are effective at identifying indicative terms. A problem is that a common design principle in these approaches is to select terms that are significant within the document, that is, are relatively rare across the corpus. In DIR, there are no cross-corpus statistics, and even in centralized retrieval the corpus is not static; thus the same term extraction method will produce different term sets in different collections. It is not at all clear that such term extraction methods can be reliable in the absence of global statistics. Another problem is that it is not known how robust these techniques are, as a single difference between the term sets means that near-duplication is not detected. A proposed solution is to return multiple hashes per document (Pugh & Henzinger 2003, Kolcz et al. 2004), but in the absence of global statistics it is still unclear that this can be effective.

The other widely-investigated approach to duplicate detection is to use *chunks* (Hoad & Zobel 2003), an approach that has been proposed for a variety of applications (Manber 1994, Lyon, Malcolm & Dickerson 2001, Brin et al. 1995, Conrad et al. 2003, Broder et al. 1997, Bernstein & Zobel 2004, Bernstein & Zobel 2005). In chunking, each document is parsed into strings of text, each typically of some fixed length or some fixed number of words. A pair of documents is deemed to be duplicated if they share a sufficient number of chunks. Chunk-extraction can be selective or exhaustive; some methods use only a few chunks per document, while in others every word occurrence is the start of a new chunk. Given a set of chunks, the *resemblance* between two documents (Broder et al. 1997) can be defined as:

$$R(d, d') = \frac{|\hat{d} \cap \hat{d}'|}{|\hat{d} \cup \hat{d}'|}$$

where $\hat{d}$ is the set of chunks extracted from document d.

A complete set of chunks is not a particularly useful document descriptor in DIR. However, the method of Fetterly et al. (2003), which we call *minimal-chunk sampling*, can be used for chunk selection. By use of an appropriate class of hash functions, by use of ρ functions from this class, chunks can be sampled in an unbiased way. Each chunk is hashed with each function, and for each of the ρ functions the smallest observed hash value yielded by any chunk is kept. These hash values can be used as proxies for chunks in determining resemblance. Fetterly et al. (2003) use $\rho = 84$, giving 84 32-bit hashes, a total of 336 bytes. For DIR, this is a significant data volume to transmit per document. It is therefore attractive to seek a more compact alternative.

5 Grainy hash vectors

A grainy hash vector (GHV) (Bernstein et al. 2006) is a form of minimal-chunk sampling method. However, it has features derived from deterministic term extraction techniques, and the vectors are only a few bytes, making it suitable for DIR. Bernstein et al. (2006) list the major benefits of GHVs as below:

- Efficiency; the hash vectors are designed to fit into a single machine word of 32 or 64 bits.
- Theoretical foundation; GHVs are based on mathematical principles that are amenable to analysis.

- Fast comparison; thousands of document vectors can be compared in a few milliseconds using bit-parallelism techniques.
- Robust comparison; GHVs do not change significantly in the presence of small differences in documents.

A GHV of n bits consists of ρ w-bit hashes and is represented as follows:

$$\rho(n,w) = \left\lfloor \frac{n}{w} \right\rfloor$$

Each of the w-bit hashes is produced by a separate minimal-chunk sampling technique. Therefore, for a 64-bit GHV with $w = 2$, there are 32 2-bit hashes that are merged into the vector. The value of w should be a factor of n, to avoid having unused bits.

For small values of w, there is a significant probability of collision between the outputs of different minimal hashes. For $w = 2$, for example, it is likely that most of the minimal hashes will be the bit pair 00 (hash values for different chunks are sorted and the 2 least significant bits of the smallest hash value are selected). On the other hand, using large values for w may be computationally expensive, and reduces the number of hashes per vector. Therefore, GHV initially uses minimal sampling to produce ρ 32-bit hashes for each document. Then the least significant w bits of each hash value are used to give the GHV.

For a pair of documents with resemblance r ($0 \leq r \leq 1$), the probability of having the same hash value at any given position of their GHVs is:

$$\phi(r;w) = r + (1-r)(2^{-w})$$

If two documents have resemblance r, they will at least have r of their hash vectors in common. Thus, the probability of a hash match between their hash vectors is r. The second component calculates the probability of having the same hash value while the two source chunks are different.

Assuming that for each bit the in a vector the probability of being 0 or 1 is independent the other bits, we can conclude that the number of matches between two vectors with resemblance r follows a binomial distribution $Bi(\rho(n,w), \phi(r;w))$, and thus

$$P(X=k) = \binom{\rho}{k} \phi^k (1-\phi)^{\rho-k}$$

where $P(X = k)$ is the probability of having k matches between the vectors of two documents with resemblance r.

For each query, collections send a GHV per document they return to the broker. The broker detects and removes duplicates or near-duplicates according to the number of mismatches between any given pair of vectors. If $w = 1$, this amounts to counting the number of bits that match in the vectors; if the number is sufficiently high, the documents are deemed to be likely to be duplicates or near-duplicates.

A critical question, then, is what is the minimum number of mismatches between two GHVs if the documents they represent are not near-duplicates? That is, a threshold for the number of mismatches needs to be set to minimise the number of false misses. (False misses are much more acceptable than false matches in this application; the former means that duplicate information is presented, while the latter means that novel information is lost.) Bernstein et al. (2006) compared the accuracy of GHVs with the duplicate detection method DECO (Bernstein & Zobel 2004), for different values of n and w. They found that GHVs can effectively detect duplicates or near-duplicates using $n = 64$, $w = 2$, and a threshold of 8 mismatches, without significant false misses. We apply the suggested values by Bernstein et al. (2006) and use 32 minimal-chunk hashes to create 32 2-bit hash values for each document.

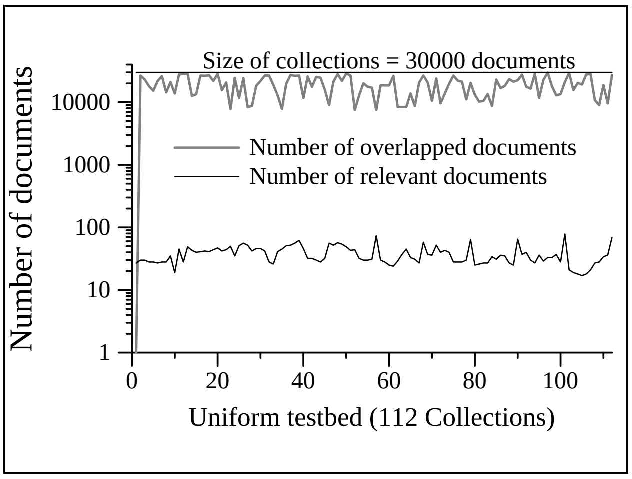


Figure 1: *Collection sizes, distribution of relevant documents and the amount of overlap between consequent collection pairs in the uniform testbed.*

We now compare the performance of collection selection and result merging algorithms on collections that overlap. We also analyze the impact of removing duplicates and near-duplicates from the list of results in terms of search effectiveness, using GHVs to determine whether documents are likely to be duplicates.

6 Testbeds

In current well-known DIR testbeds such as trec123-100col-bysource or trec4-kmeans,[3] the degree of overlap between collections is intentionally set to zero. Therefore, current testbeds are not suitable for evaluating DIR methods in the presence of overlap. We create three testbeds using the documents available in the TREC GOV data (Craswell & Hawking 2002). These testbeds are as follows:

uniform-112col-dups (uniform): This testbed is comprised of 112 collections, each containing 30 000 documents, created using a sliding window on the TREC GOV documents. The first 30 000 documents comprise the first collection. Then a random percentage R ($R \geq 25\%$) is picked. The second collection is created from the last $R\%$ of the first collection and the next documents from the GOV corpus to a total size of 30 000 documents. The rest of the collections are generated in the same sliding window manner.

Figure 1 shows the distribution of relevant documents for TREC topics 551–600 among the collections in this testbed. As was expected, relevant documents are spread uniformly among collections. The figure also shows the number of documents in each collection that are shared with the previous collection. The rate of overlap varies from 25% to 99%.

skewed-115col-dups (skewed): Collection selection algorithms show variable performance on different testbeds. Some approaches such as CORI are found to be less effective when the distribution of collection sizes is skewed (D'Souza et al. 2004, Si & Callan 2003*a*). To create such a testbed, we adapt the approach Si & Callan (2003*a*) used for deriving

[3]Available at `www.cs.cmu.au/~callan/Data`.

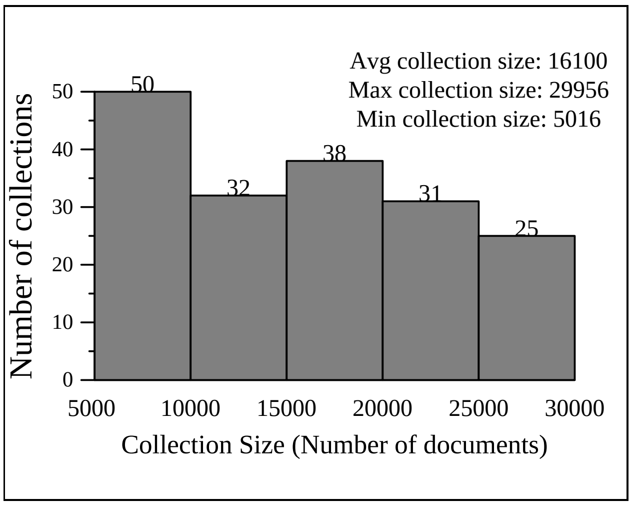


Figure 2: *Collection sizes in the Qprobed testbed.*

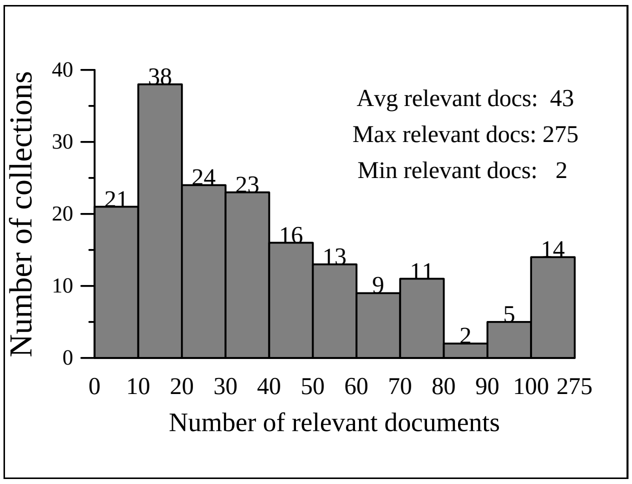


Figure 3: *Distribution of relevant documents in the Qprobed testbed.*

the so-called *representative testbed* from the trec123-100col-bysource documents. Every tenth collection in the uniform testbed starting from the first collection is collapsed into a single large collection. The same procedure is repeated starting with the second and third collections and another two large collections are created. The testbed thus contains the uniform testbed plus an additional three large collections.

Qprobed-176col-dups (Qprobed): 176 collections have been generated by passing 200 probe queries to an index of the TREC GOV documents. Queries are the most frequent single terms in a query log supplied by a major search engine, of queries with a highly ranked answer in the `.gov` domain. For each query, a random number of results between 5 000 and 30 000 are extracted and gathered as a collection. Queries that return less than 5 000 documents are discarded. The average size of collections in this testbed is 16 100 documents while the largest and smallest collections contain 29 956 and 5 016 documents. The distribution of collection sizes and the number of relevant documents among collections are depicted in Figures 2 and 3. Considering Figure 3, for example, there are 50 collections that have between 5 000 and 10 000 documents, and there are 38 collections that contain between 10 and 20 relevant documents each for TREC topics 551–600.

The degree of overlap among collections in this testbed is diverse. Figure 4 shows that there are

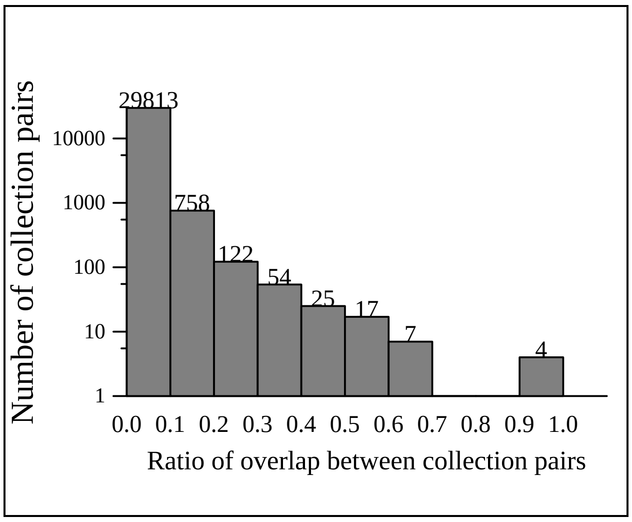


Figure 4: *Degree of overlap between collections in the Qprobed testbed.*

29 813 collection pairs that have less than 10% overlap while the rate of overlap for four collection pairs is close to 100%.

The ideal testbed for these experiments would be complete crawls of websites from the hidden web, but by construction such crawls are not easily available. Another good testbed would be crawls of sites known to contain high levels of overlap. There are such sites in the TREC GOV data (in which we believe more than half the pages are duplicates or near duplicates), but identifying them is far from straightforward. It is for these reasons that other ways of forming testbeds are of interest. The uniform and skewed collections are somewhat artificial, but they allow exploration of the effectiveness of DIR methods as a function of the extent of overlap. The Qprobed testbed consists of collections with some degree of internal consistency, and the collections are comprised of documents selected by a method that is fully independent of the techniques we are exploring.

In the next section we investigate the effectiveness of different DIR algorithms on the discussed testbeds.

7 Experimental results

On each testbed, we explore three combinations of collection selection and result merging algorithms. These are CORI-CORI, ReDDE-CORI, and ReDDE-SSL where the first part specifies the collection selection method and the last part represents the merging algorithm used for the experiments.

We use query-based sampling to gather 300 documents from each collection. The sampled documents from each collection are used as its representation set. In addition, aggregating all representation sets creates a central sample index that is used by SSL result merging (Si & Callan 2003*b*). All experiments reported in this paper use the Lemur toolkit.[4] Methods are compared according to their precision at the top n returned documents ($P@n$). For each query, at most 1 000 answers are returned.

We use a 64-bit GHV with $w = 2$ and a threshold of 8 mismatches out of possible 32, as recommended by Bernstein et al. (2006). Collections provide the broker with a GHV for each answer they return. They also publish the document scores of their returned answers that then will be used by the broker for result

[4]`http://www.lemurproject.org`

merging. That is, we assume a semi-cooperative environment.

Duplicate and near-duplicate documents in an answer list can be regarded as redundant and irrelevant, as they do not add to the user's knowledge. We use GHV to identify duplicate and near-duplicate documents in the final merged results. Then we compare the search effectiveness of two scenarios where in the first one, duplicates are considered as irrelevant and in the second set duplicates are removed. In the first scenario, the broker removes those documents that GHV detects as redundant from the list of results. The upper tables for each combination contain the precision values for this scenario.

In the second scenario, the returned duplicate and near-duplicate documents are considered as irrelevant. To avoid bias, we do not use GHV to judge whether documents are duplicates and near-duplicates; instead, the broker uses a list of near-duplicate documents in the TREC GOV that are identified by DECO (Bernstein & Zobel 2004) duplicate detection method.[5]

One might expect that the lists of duplicate documents detected by GHV and DECO would be significantly different. However, our investigations show that 93% of documents that are identified as near-duplicate by GHV are also detected by DECO, while, 72% of all documents that are identified by DECO are detected by GHV. The high rate of accuracy and coverage of GHVs for detecting near-duplicate documents is consistent with the reported values by Bernstein et al. (2006) on a centralized index. On average, over all experiments, use of GHVs detects 9 near-duplicates per query while this number is 12.5 for DECO. Considering that 1 000 documents are collected per query, the difference of 2.5 documents has negligible impact on the final effectiveness. Also, this is further evidence that GHVs avoid false misses.

Note that DECO removes exact-duplicate pairs according to their document identifiers. All exact-duplicates are detected and removed by GHV as there is no mismatch between their vectors.

We use the t-test to measure the statistical significance of difference between results in the presence and absence of duplicates. The differences at 0.95, 0.99 and 0.999 confidence intervals are respectively specified with $*$, $\dagger$, and $\ddagger$. The baseline for all experiments against which significance is measured is the effectiveness of retrieval when the returned duplicates and near-duplicates are considered as irrelevant. The differences are measured against the scenario in which redundant documents are removed by GHV.

Table 1 shows the effectiveness of different combinations on the uniform testbed. Cutoff (CO) values indicate the number of collections selected for each query. Comparing the first two sets of experiments (CORI-CORI and ReDDE-CORI) suggests that CORI and ReDDE collection selection methods have similar performance on the uniform testbed. Considering that all collections in this testbed contain an identical number of documents, the results are consistent with the previous observations of Si & Callan (2003*a*), which suggest the performance of CORI and ReDDE are similar when the distribution of collection sizes is not skewed. As in experiments on collections that do not overlap (Si & Callan 2003*b*), SSL merges results more effectively than does CORI on the uniform testbed.

The impact of removing duplicates and near-duplicates becomes more apparent as the cutoff value grows. In addition, the effectiveness of ReDDE-SSL improve more significantly than the other combinations when duplicates are removed. This suggests that selecting more collections and using ReDDE-SSL combination increase the likelihood of finding duplicate documents in the final results. The probable explanation is that ReDDE-SSL is more effective than the other methods at giving the same document the same score when it is present in multiple collections. That is, if one copy of a duplicate document is fetched, then under ReDDE-SSL the other copy is also likely to be fetched. (The number of duplicates and near-duplicates returned by different combinations is shown in Table 4 and is discussed later.)

Table 1: *The impact of removing duplicate and near-duplicate documents. Results are obtained by running the TREC topics 551–600 (title) on the "uniform" testbed. CO represents the cutoff value.*

	P@5	**P@10**	**P@15**	**P@20**
Duplicates Removed (CORI-CORI)				
CO3	0.0840	0.0620	0.0507	0.0470
CO5	0.0840	0.0700	0.0587	0.0510
CO10	0.1760*	0.1240†	0.1093*	0.0970†
CO20	0.1959†	0.1449†	0.1252†	0.1153†
Duplicates Irrelevant (CORI-CORI)				
CO3	0.0800	0.0600	0.0493	0.0460
CO5	0.0800	0.0680	0.0573	0.0500
CO10	0.1560	0.1120	0.1027	0.0890
CO20	0.1592	0.1306	0.1129	0.1031
Duplicates Removed (ReDDE-CORI)				
CO3	0.0907	0.0542	0.0431	0.0354
CO5	0.1250	0.0729	0.0583	0.0521
CO10	0.1583	0.1083	0.0958	0.0857
CO20	0.1625	0.1333†	0.1139*	0.1083†
Duplicates Irrelevant (ReDDE-CORI)				
CO3	0.0907	0.0542	0.0431	0.0354
CO5	0.1208	0.0708	0.0583	0.0479
CO10	0.1500	0.0979	0.0917	0.0833
CO20	0.1542	0.1187	0.1056	0.1010
Duplicates Removed (ReDDE-SSL)				
CO3	0.1167	0.0667	0.0528	0.0490
CO5	0.1583	0.1062*	0.0861	0.0719
CO10	0.1875*	0.1479*	0.1347*	0.1177*
CO20	0.2042*	0.1979*	0.1667‡	0.1458†
Duplicates Irrelevant (ReDDE-SSL)				
CO3	0.1167	0.0667	0.0514	0.0479
CO5	0.1417	0.0979	0.0847	0.0708
CO10	0.1759	0.1375	0.1264	0.1135
CO20	0.1792	0.1792	0.1500	0.1323

Table 2 shows similar results on the skewed testbed. For small cutoff values, ReDDE significantly outperforms CORI, because the three largest collections contain many relevant documents that get high ranks by ReDDE for the majority of queries. The better performance of ReDDE is not surprising as it is designed for situations where the distribution of collection sizes is skewed (Si & Callan 2003*a*). SSL is again the dominant merging algorithm and produces better results than CORI.

As in previous experiments on the uniform testbed, removing duplicate documents changes the final precision more significantly for larger cutoff values. The gaps are also similarly larger when ReDDE is used for collection selection and SSL is used for result merging, for the reasons given above.

However, the results on the Qprobed testbed, shown in Table 3, are rather different. The CORI-CORI combination produces the greatest effectiveness when duplicate documents are removed. Comparing the results of ReDDE-CORI and ReDDE-SSL combinations suggests that the performance of SSL

[5]The list of near-duplicate documents detected by DECO is available at: `http://www.cs.rmit.edu.au/~ybernste`

Table 2: *The impact of removing duplicate and near-duplicate documents. Results are obtained by running the TREC topics 551–600 (title) on the "skewed" testbed. CO represents the cutoff value.*

	P@5	**P@10**	**P@15**	**P@20**
Duplicates Removed (CORI-CORI)				
CO3	0.0880	0.0660	0.0573	0.0520
CO5	0.0800	0.0680	0.0613	0.0530
CO10	0.1680†	0.1220*	0.1133*	0.0970†
CO20	0.1840†	0.1380‡	0.1253†	0.1140‡
Duplicates Irrelevant (CORI-CORI)				
CO3	0.0840	0.0640	0.0560	0.0510
CO5	0.0760	0.0660	0.0600	0.0520
CO10	0.1440	0.1120	0.1053	0.0900
CO20	0.1520	0.1200	0.1133	0.0990
Duplicates Removed (ReDDE-CORI)				
CO3	0.1625	0.1312	0.1139*	0.1052
CO5	0.1625	0.1417	0.1208*	0.1125
CO10	0.1667	0.1437	0.1250*	0.1177*
CO20	0.1583*	0.1417*	0.1278*	0.1208†
Duplicates Irrelevant (ReDDE-CORI)				
CO3	0.1583	0.1271	0.1097	0.1010
CO5	0.1583	0.1375	0.1167	0.1083
CO10	0.1583	0.1354	0.1194	0.1125
CO20	0.1542	0.1333	0.1208	0.1135
Duplicates Removed (ReDDE-SSL)				
CO3	0.1625	0.1417†	0.1097*	0.1000*
CO5	0.1708*	0.1458*	0.1278†	0.1115†
CO10	0.1917*	0.1583†	0.1444†	0.1260†
CO20	0.1833†	0.1646†	0.1556‡	0.1437‡
Duplicates Irrelevant (ReDDE-SSL)				
CO3	0.1458	0.1208	0.1014	0.0906
CO5	0.1542	0.1354	0.1111	0.1010
CO10	0.1750	0.1312	0.1250	0.1104
CO20	0.1583	0.1417	0.1306	0.1135

Table 3: *The impact of removing duplicate and near-duplicate documents. Results are obtained by running the TREC topics 551–600 (title) on the "Qprobed" testbed. CO represents the cutoff value.*

	P@5	**P@10**	**P@15**	**P@20**
Duplicates Removed (CORI-CORI)				
CO3	0.1959	0.1776†	0.1578†	0.1378*
CO5	0.2163	0.1857†	0.1592†	0.1439†
CO10	0.2204†	0.1898†	0.1660‡	0.1469†
CO20	0.2571†	0.2143‡	0.1782‡	0.1612‡
Duplicates Irrelevant (CORI-CORI)				
CO3	0.1878	0.1612	0.1442	0.1296
CO5	0.2000	0.1653	0.1401	0.1235
CO10	0.1918	0.1633	0.1333	0.1204
CO20	0.2204	0.1776	0.1429	0.1255
Duplicates Removed (ReDDE-CORI)				
CO3	0.1750	0.1500	0.1194	0.1000*
CO5	0.2000	0.1729	0.1403*	0.1167*
CO10	0.2125	0.1938*	0.1597†	0.1406†
CO20	0.2083	0.1958	0.1681†	0.1469†
Duplicates Irrelevant (ReDDE-CORI)				
CO3	0.1667	0.1396	0.1139	0.0938
CO5	0.1875	0.1583	0.1292	0.1094
CO10	0.2000	0.1771	0.1389	0.1271
CO20	0.1958	0.1812	0.1444	0.1302
Duplicates Removed (ReDDE-SSL)				
CO3	0.1625	0.1396	0.1083	0.0958
CO5	0.2125*	0.1542†	0.1319†	0.1146*
CO10	0.2458†	0.1958†	0.1611†	0.1427‡
CO20	0.2292†	0.2021‡	0.1736‡	0.1542‡
Duplicates Irrelevant (ReDDE-SSL)				
CO3	0.1625	0.1313	0.1028	0.0885
CO5	0.1917	0.1333	0.1153	0.1042
CO10	0.2042	0.1687	0.1403	0.1167
CO20	0.1750	0.1500	0.1319	0.1156

and CORI merging methods is similar. Therefore, the high effectiveness of CORI-CORI is largely due to its effective collection selection method. We observed similar trends for CORI and ReDDE on a similar testbed (trec4-kmeans (Xu & Callan 1998)) in our preliminary experiments. We believe that ReDDE is not as effective as CORI on testbeds where the distribution of collection sizes in not skewed and the documents within each collection have similar topicality.

These results illustrate the importance of using diverse testbeds in such experiments; the uniform and skewed results are not predictive of the results on a collection created with another method.

As in experiments on the other testbeds, removal of duplicates may drastically improve the final precision. The difference can be significant at the 0.999 confidence interval for larger cutoff values.

The number of duplicate and near-duplicate documents detected by GHVs for different combinations is presented in Table 4. In the uniform and skewed testbeds, the number of near-duplicates detected climbs significantly for CORI-CORI while it remains near constant for the other combinations. In the Qprobed testbed, the number of near-duplicates in the result declines by selecting more collections, which is possibly an artefact of the way the testbed was constructed.

Comparing the exact-duplicate (ED) numbers for ReDDE-CORI and ReDDE-SSL suggests that SSL is more likely to return exact-duplicates in the final results than CORI. Using CORI and ReDDE collection selection algorithms leads to similar number of exact-duplicates on the uniform and skewed testbeds. However, collections selected by CORI return more exact-duplicates on the Qprobed testbed.

The coverage values in Table 4 represent the fraction of unique documents in the testbeds that are being searched on different cutoff points. The coverage values grow linearly for all combinations in the uniform and Qprobed testbeds. This implies that collections selected by CORI and ReDDE contain similar number of documents.

In the skewed testbed, the coverage values for ReDDE collection selection increases very quickly for small cutoff values while it grows linearly for CORI. This is due to the fact that the three largest collections in the skewed testbeds get high ranks by ReDDE for the majority of queries. Therefore, they are very likely to be selected by the broker in the top three or five collections. However in CORI, the three largest collections do not have any advantage over the other collections to be selected.

8 Conclusions

We have investigated the problem of distributed information retrieval on collections that overlap, evaluating the effectiveness of several collection selection and result merging algorithms. Our experiments are broadly consistent with previous observations on the traditional, disjoint DIR testbeds: ReDDE is a better collection selection algorithm than CORI when the distribution of collection sizes is skewed, and SSL is a more effective result merging method than CORI. However, on a testbed where documents in each col-

Table 4: *Number of duplicates, near-duplicates and coverage values obtained by collection selection and result merging combinations on the three testbeds. For all experiments, the TREC topics 551–600 (title) have been used and the numbers are averaged over all queries. CO is the cutoff value. For each query, 1 000 answers are collected. ND and ED stand for near-duplicate and exact-duplicate respectively.*

	ND	ED	coverage	ND	ED	coverage	ND	ED	coverage
	CORI-CORI			**ReDDE-CORI**			**ReDDE-SSL**		
Uniform									
CO3	8	53	0.06	9	19	0.06	10	53	0.06
CO5	10	78	0.11	9	66	0.11	11	77	0.11
CO10	11	116	0.21	8	110	0.21	9	118	0.21
CO20	15	191	0.38	11	174	0.38	11	195	0.38
Skewed									
CO3	9	56	0.06	11	106	0.31	10	143	0.31
CO5	11	82	0.11	9	140	0.35	11	193	0.35
CO10	13	122	0.23	12	163	0.42	10	229	0.42
CO20	15	211	0.42	9	211	0.54	11	298	0.54
Qprobed									
CO3	15	136	0.06	8	76	0.04	8	83	0.04
CO5	9	218	0.11	6	145	0.07	6	158	0.07
CO10	5	340	0.21	5	241	0.14	6	264	0.14
CO20	5	451	0.38	6	335	0.27	4	371	0.27

lection may share topicality, CORI seems to be a better option than ReDDE for collection selection.

For this work we introduced three testbeds created from documents available in the TREC GOV corpus. The different testbeds yield results that are somewhat inconsistent, demonstrating that design of experiment is critical to achieving robust results in this area—conclusions based solely on one testbed might not generalise.

We have shown that removing duplicates and near-duplicates can significantly improve the final search effectiveness. We used grainy hash vectors to detect duplicate and near-duplicate documents in the final list of results on the broker. Grainy hash vectors successfully identified 72% of all near-duplicate documents in the results. The accuracy of GHVs is consistent with the values we previously reported on a centralized index (Bernstein et al. 2006). These results demonstrate that duplicate removal need not be expensive, and can greatly enhance the quality of results returned by a search engine.

References

Allan et al, J. (2003), 'Challenges in information retrieval and language modeling: report of aworkshop held at the center for intelligent information retrieval, University of Massachusetts Amherst, september 2002', *SIGIR Forum* **37**(1), 31–47.

Avrahami, T., Yau, L., Si, L. & Callan, J. (2006), 'The FedLemur: federated search in the real world', *Journal of the American Society for Information Science and Technology* **57**(3), 347–358.

Bergman, M. (2001), 'The deep Web: Surfacing hidden value', *Journal of Electronic Publishing* **7**(1).

Bernstein, Y., Shokouhi, M. & Zobel, J. (2006), Compact features for detection of near-duplicates in distributed retrieval, *in* 'Proceedings of String Processing and Information Retrieval Symposium (to appear)', Glasgow, Schotland.

Bernstein, Y. & Zobel, J. (2004), A scalable system for identifying co-derivative documents, *in* 'Proceedings of String Processing and Information Retrieval Symposium', Padova, Italy, pp. 55–67.

Bernstein, Y. & Zobel, J. (2005), Redundant documents and search effectiveness, *in* 'Proceedings of 14th ACM CIKM Conference on Information and Knowledge Management', Bremen, Germany, pp. 736–743.

Brin, S., Davis, J. & García-Molina, H. (1995), Copy detection mechanisms for digital documents, *in* 'Proceedings of ACM SIGMOD international conference on Management of Data', San Jose, California, pp. 398–409.

Broder, A. Z., Glassman, S. C., Manasse, M. S. & Zweig, G. (1997), 'Syntactic clustering of the web', *Computer Networks and ISDN Systems* **29**(8-13), 1157–1166.

Callan, J. & Connell, M. (2001), 'Query-based sampling of text databases', *ACM Transactions on Information Systems* **19**(2), 97–130.

Callan, J., Connell, M. & Du, A. (1999), Automatic discovery of language models for text databases, *in* 'Proceedings of ACM SIGMOD International Conference on Management of Data', Philadelphia, Pennsylvania, pp. 479–490.

Callan, J., Croft, W. B. & Harding, S. M. (1992), The INQUERY retrieval system, *in* 'Proceedings of third International Conference on Database and Expert Systems Applications', Valencia, Spain, pp. 78–83.

Callan, J., Lu, Z. & Croft, W. B. (1995), Searching distributed collections with inference networks, *in* 'Proceedings of 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval', Seattle, Washington, pp. 21–28.

Chowdhury, A., Frieder, O., Grossman, D. & McCabe, M. C. (2002), 'Collection statistics for fast duplicate document detection', *ACM Transactions on Information Systems* **20**(2), 171–191.

Conrad, J. G., Guo, X. S. & Schriber, C. P. (2003), Online duplicate document detection: Signature reliability in a dynamic retrieval environment, *in* 'Proceedings of 12th ACM CIKM Conference on Information and Knowledge Management', New Orleans, Louisiana, pp. 443–452.

Cooper, J. W., Coden, A. R. & Brown, E. W. (2002), Detecting similar documents using salient terms, *in* 'Proceedings of 11th ACM CIKM Conference on Information and Knowledge Management', McLean, Virginia, pp. 245–251.

Craswell, N., Bailey, P. & Hawking, D. (2000), Server selection on the World Wide Web, *in* 'Proceedings of Fifth ACM Conference on Digital Libraries', San Antonio, Texas, pp. 37–46.

Craswell, N. & Hawking, D. (2002), Overview of the TREC-2002 Web Track, *in* 'Proceedings of TREC-2002', Gaithersburg, Maryland.

Croft, B. (2000), 'Combining approaches to information retrieval', *Advances in information retrieval, chapter 1* pp. 1–36.

D'Souza, D., Zobel, J. & Thom, J. (2004), Is CORI effective for collection selection? an exploration of parameters, queries, and data, *in* 'Proceedings of Australian Document Computing Symposium', Melbourne, Australia, pp. 41–46.

Fetterly, D., Manasse, M. & Najork, M. (2003), On the evolution of clusters of near-duplicate web pages, *in* 'Proceedings of first Latin American Web Congress', IEEE, pp. 37–45.

Fox, E. & Shaw, J. (1994), Combination of multiple searches, *in* 'Proceedings of TREC-1994', NIST Special Publication, Gaithersburg, Maryland, pp. 105–108.

Gauch, S., Wang, G. & Gomez, M. (1996), 'ProFusion: Intelligent fusion from multiple, distributed search engines', *Journal Universal Computer Science* **2**(9), 637–649.

Gravano, L., Chang, C. K., Garcia-Molina, H. & Paepcke, A. (1997), STARTS: Stanford proposal for Internet metasearching, *in* 'Proceedings of ACM SIGMOD International Conference on Management of Data', Tucson, Arizona, pp. 207–218.

Gravano, L., Garcia-Molina, H. & Tomasic, A. (1999), 'GlOSS: text-source discovery over the Internet', *ACM Transactions on Database Systems* **24**(2), 229–264.

Hawking, D. & Thomas, P. (2005), Server selection methods in hybrid portal search, *in* 'Proceedings of 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval', Salvador, Brazil, pp. 75–82.

Hernandez, T. & Kambhampati, S. (2005), Improving text collection selection with coverage and overlap statistics, *in* 'Proceedings of 14th International Conference on World Wide Web', Chiba, Japan.

Hoad, T. C. & Zobel, J. (2003), 'Methods for identifying versioned and plagiarised documents', *Journal of the American Society for Information Science and Technology* **54**(3), 203–215.

Ilyinski, S., Kuzmin, M., Melkov, A. & Segalovich, I. (2002), An efficient method to detect duplicates of web documents with the use of inverted index, *in* 'Proceedings of 11th International Conference on World Wide Web', Honolulu, Hawaii.

Ipeirotis, P. G. & Gravano, L. (2002), Distributed search over the hidden Web: Hierarchical database sampling and selection., *in* 'Proceedings of 28th International Conference on Very Large Data Bases', Hong Kong, China, pp. 394–405.

Ipeirotis, P. G. & Gravano, L. (2004), When one sample is not enough: improving text database selection using shrinkage, *in* 'Proceedings of ACM SIGMOD International Conference on Management of Data', Paris, France, pp. 767–778.

Kirsch, T. (2003), 'Document retrieval over networks wherein ranking and relevance scores are computed at the client for multiple database documents', *U.S. Patent 5,659,732* .

Kolcz, A., Chowdhury, A. & Alspector, J. (2004), Improved robustness of signature-based near-replica detection via lexicon randomization, *in* 'Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining', Seattle, WA, pp. 605–610.

Lee, J. (1997), Analyses of multiple evidence combination, *in* 'Proceedings of the 20th annual international ACM SIGIR conference on Research and development in information retrieval', Philadelphia, Pennsylvania, United States, pp. 267–276.

Lyon, C., Malcolm, J. & Dickerson, B. (2001), Detecting short passages of similar text in large document collections, *in* 'Proceedings of Conference on Empirical Methods in Natural Language Processing', Philadelphia, Pennsylvania.

Manber, U. (1994), Finding similar files in a large file system, *in* 'Proceedings of USENIX Winter Technical Conference', San Fransisco, CA, pp. 1–10.

Meng, W., Yu, C. & Liu, K. (2002), 'Building efficient and effective metasearch engines', *ACM Computing Surveys* **34**(1), 48–89.

Nottelmann, H. & Fuhr, N. (2003), Evaluating different methods of estimating retrieval quality for resource selection, *in* 'Proceedings of 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval', Toronto, Canada, pp. 290–297.

Powell, A. L. & French, J. (2003), 'Comparing the performance of collection selection algorithms', *ACM Transactions on Information Systems* **21**(4), 412–456.

Pugh, W. & Henzinger, M. H. (2003), 'Detecting duplicate and near-duplicate files (United States Patent 6,658,423)'.

Rasolofo, Y., Abbaci, F. & Savoy, J. (2001), Approaches to collection selection and results merging for distributed information retrieval, *in* 'Proceedings of 10th ACM CIKM International Conference on Information and knowledge management', Atlanta, Georgia, pp. 191–198.

Selberg, E. & Etzioni, O. (1997), 'The MetaCrawler architecture for resource aggregation on the web', *IEEE Expert* **12**(1), 8–14.

Shokouhi, M., Scholer, F. & Zobel, J. (2006), Sample sizes for query probing in uncooperative distributed information retrieval, *in* 'Proceedings of of Eighth Asia Pacific Web Conference', Harbin, China, pp. 63–75.

Si, L. & Callan, J. (2003*a*), Relevant document distribution estimation method for resource selection, *in* 'Proceedings of 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval', Toronto, Canada, pp. 298–305.

Si, L. & Callan, J. (2003*b*), 'A semisupervised learning method to merge search engine results', *ACM Transactions on Information Systems* **21**(4), 457–491.

Si, L. & Callan, J. (2004), Unified utility maximization framework for resource selection, *in* 'Proceedings of 13th ACM CIKM Conference on Information and Knowledge Management', Washington, D.C., pp. 32–41.

Si, L. & Callan, J. (2005), Modeling search engine effectiveness for federated search, *in* 'Proceedings of 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval', Salvador, Brazil.

Si, L., Jin, R., Callan, J. & Ogilvie, P. (2002), A language modeling framework for resource selection and results merging, *in* 'Proceedings of 11th ACM CIKM International Conference on Information and Knowledge Management', New York, NY, pp. 391–397.

Sugiura, A. & Etzioni, O. (2000), Query routing for web search engines: architectures and experiments, *in* 'Proceedings of the 9th international World Wide Web conference on Computer networks', North-Holland Publishing Co., Amsterdam, The Netherlands, pp. 417–429.

Wu, S. & McClean, S. (2006), 'Result merging methods in distributed information retrieval with overlapping databases', *Journal of Information Retrieval (In press)* .

Xu, J. & Callan, J. (1998), Effective retrieval with distributed collections, *in* 'Proceedings of 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval', Melbourne, Australia, pp. 112–120.

Yuwono, B. & Lee, D. L. (1997), Server ranking for distributed text retrieval systems on the internet, *in* 'Proceedings of the Fifth International Conference on Database Systems for Advanced Applications (DASFAA)', World Scientific Press, Melbourne, Australia, pp. 41–50.

Zamir, O. & Etzioni, O. (1999), Grouper: a dynamic clustering interface to web search results, *in* 'Proceedings of 8th International Conference on World Wide Web', Toronto, Canada, pp. 1361–1374.

Zobel, J. (1997), Collection selection via lexicon inspection, *in* P. Bruza, ed., 'Proceedings of the Australian Document Computing Symposium', pp. 74–80.

Zobel, J. & Bernstein, Y. (2006), The case of the duplicate documents: Measurement, search, and science, *in* 'Proceedings of of Eighth Asia Pacific Web Conference', Harbin, China, pp. 26–39.

Building a disordered protein database: A case study in managing biological data

Arran D. Stewart **Xiuzhen Zhang**

School of Computer Science and IT, RMIT University
Email: {arstewar, zhang}@cs.rmit.edu.au

Abstract

A huge diversity of biological databases is available via the Internet, but many of these databases have been developed in an ad hoc manner rather than in accordance with any data management principles. In addition, in the area of disordered protein databases, many of the databases have not been made publicly available. This poses challenges to researchers, since reliable protein databases are required in order to test and measure the accuracy of protein structure prediction software. In this paper, we describe our work developing a disordered protein database using data from the protein secondary structure database DSSP-cont. In particular, we discuss the way in which we have addressed the issues of data cleaning, query processing and interoperability. This research is a pilot study in managing biological data.

Keywords: disordered proteins, biological data management

1 Introduction

The number of biological databases available via the Internet is large and constantly increasing. The number of publicly available databases that appear each year is so many that the Nucleic Acids Research Journal dedicates an entire issue to them annually (Srdanovic et al. 2005), and this listing does not include many more databases which are developed by researchers but not made publicly available.

These biological databases vary greatly in size and complexity (Luscombe et al. 2001). On the one hand one can find extremely large, "industrial scale" databases like the Protein Data Bank (PDB, http://www.rcsb.org/pdb/) (Berman et al. 2000), which stores data on the position of atoms within protein structures as determined by such techniques as X-ray crystallography or Nuclear Magnetic Resonance (NMR) spectroscopy. The PDB has a complicated, multi-tier infrastructure, an integrated workflow system, and dedicated staff to maintain and develop it. At the other extreme are small, temporary collections of flat files put together by individual bioinformatics researchers or small teams.

The multiplicity of biological databases has attracted considerable attention from members of the database research community. Some have called for more research into the data management issues of biological databases—for instance, Gupta (2004), Jagadish & Olken (2003) and Wooley & Lin (2005). A number of important issues have been recognized, such as the complexity of queries and data models in biological databases (discussed further in section 2).

In this paper we report on how we have addressed data management issues in building a disordered protein database using data from the DSSPcont (http://cubic.bioc.columbia.edu/services/DSSPcont/) protein secondary structure database. We also suggest practical implementation tips for bioinformaticists working on structural biology database projects. We provide a more detailed explanation of what disordered protein segments are later in section 2.1. Roughly, however, they can be described as highly flexible regions of protein which do not adopt a regular, stable structure under normal physiological conditions. Our aim in the present research is to develop a reliable database of protein structural data which users can query in order to identify protein segments that meet particular criteria for being disordered.

In particular, we discuss the following issues:

- **Data cleaning and transformation.** High quality data is crucial to the success of any database. One important data quality issue is that of data semantics. For instance, the concept of solvent accessibility (SA) has been widely used to compute the hydrophobic contribution to the stability of proteins, and is used for defining protein disorder in our study. Raw SA scores derived from databases such as DSSPcont require normalization before they can be used as a reliable measure of protein flexibility. However, this normalization has been performed in a number of different ways in the biological literature. We propose normalization of SA according to the method of Ahmad et al. (2003) to avoid semantic disparity and ensure the reliability of our database.

- **Incremental query processing.** Queries run on biological databases are usually complex—the information needed is typically defined by compound criteria rather than a single condition, and are often performed on sequences or more complex data structures. In extracting disordered segments from DSSPcont, for instance, our criteria for identifying segments as disordered are based on a combined set of parameters. Furthermore, biologists often want to examine how changes in parameters affect the result data. We therefore design our system in such a way that queries are processed incrementally to improve efficiency.

- **Data interoperability and storage.** The text-based FASTA format is widely used for data exchange in bioinformatics. However interoperability considerations, and the ability to store metadata about sequences in a structured way, make XML a preferable option. We discuss how data is organized in our system.

Our prototype disordered database can be used to compare the performance of different disorder prediction models. Since it represents a set of disordered segments collected according to a clearly-defined definition of disorder, by comparing the properties of our collection with others compiled according to different criteria, our database can also be used to examine the relationships between different types of disorder. The database is currently in use by staff at the Walter and Eliza Hall Institute of Medical Research (WEHI, `http://www.wehi.edu.au/`) for analyzing the properties of existing disordered proteins. It is intended that a Web-based interface be developed to make the database available for public use.

1.1 Related work

As noted previously, several studies have included general discussions of various data management issues for biological databases, and the need for increased effort in this area from the database community—for instance, Gupta (2004), Jagadish & Olken (2003) and Wooley & Lin (2005). However, general guidelines on how to resolve such issues have not yet been developed. Although not a general solution, our work is a pilot study in applying general data management principles to building a specific purpose database.

Bry & Kroger (2003) have examined in detail various biological databases and describe the development of data management technology in such databases. They noted that developers of biological databases have typically taken an ad hoc approach to data management issues.

Herbert et al. (2004) proposed a framework for managing and integrating evolutionary experimental data, and discussed data cleaning as a necessary step in data integration. They used a conceptual model to resolve mismatches and achieve uniformity among different terms. In contrast, our approach is to physically clean the data itself.

Much work on protein disorder has focused on developing models which can predict disorder from protein sequences (e.g., (Linding et al. 2003a), (Linding et al. 2003b), and (Cheng et al. 2005)). Building such models and evaluating their accuracy requires reliable training databases of known disorder in existing proteins. However, the training databases used in these research efforts are generally not intended for re-use and are not made publicly available. Data management issues are not discussed in any of these works.

DisProt (`http://www.disprot.org/`) (Vucetic et al. 2005) is the only publicly available disordered protein database. It contains records of protein segments that have been reported in the biological literature as showing disorder in experiments. Since different definitions of disorder have been adopted by experimenters, and since several distinct types of protein disorder are believed to exist, the use of DisProt's data is not suitable for studies that require a single, consistent definition of disorder. Additionally, the exact extent of a disordered segment in DisProt will have been determined by experimenters, and thus contains a subjective element. For our purposes, it is preferable that disordered segments be identified by objectively applied criteria. Data management issues are not discussed in this work.

2 Background

Although highly diverse, biological databases do have some similarities. For instance, those available via the World Wide Web typically follow a 3-tier architectural model (Smail-Tabbone et al. 2005). At the bottom level is the actual database, at the top is a web interface, and in between is a software layer which mediates between the two. The middle layer turns requests made via the Web interface into actual database queries, and presents database results in HTML format.

Biological databases typically store data of a complex and often hierarchical nature (Brusic et al. 1998). The data may include sequences (for instance, DNA or protein sequences), graphs (for instance, metabolic pathways) or spatial information (for instance, 3D protein structures). It is possible to store such structures in a relational DBMS (RDBMS) by decomposing them into tables, but the result is often an overly complex relational schema (Bry & Kroger 2003). As a result, the use of an RDBMS has been described as being a "clumsy and awkward" way of manipulating complex biological data (Wooley & Lin 2005).

Consequently, biological databases have typically taken alternative approaches to data management. The earliest approach (and still a common one) was to simply store data as text-based flat files (Nelson et al. 2003). Some biological databases adopted a format known as ASN.1, originally developed for describing telecommunications protocols (Bry & Kroger 2003). More recent approaches have been the use of object-oriented or object-relational DBMSs (Jagadish & Olken 2003), or XML- or other semi-structured DBMSs (Shui et al. 2003).

The sorts of queries made on biological databases can typically be expressed succinctly in English, but lead to complex processing requirements. Singh (2003) gives several illustrative examples: "find all genes that are structurally similar to a given gene and express similarly over a specific DNA microarray dataset"; "find all proteins that are structurally similar to a given protein, used in a given pathway, and are expressed similarly as another given protein in a given experiment"; and "find all protein pairs that are less than 30% similar at a string level, share a given active site, and co-occur in some metabolic pathway". All of these queries would lead to complex processing of non-atomic data types.

Our area of research—disordered protein databases—is a good example of the challenges faced. As will be seen in the next section, protein data is complex and hierarchical in nature, and querying it requires more complex processing than can be expressed in relational queries.

2.1 Proteins and their structure

Proteins are complex biological molecules made up of long sequences of small molecules—amino acids or residues—linked together in a chain. A single protein can contain a number of chains of amino acids (Branden & Tooze 1991); a protein will typically contain hundreds of amino acids, and some contain thousands.

Proteins have several levels of structure (Baxevanis & Ouellette 2005). At the level of *primary structure*, a protein chain can simply be viewed as a string of letters, with each letter representing one of 20 naturally-occurring amino acids (listed in Table 1).

For instance, the primary sequence of the human myoglobin protein, used to carry oxygen in the muscles, begins "GLSDGEWQLVLNVWGKVEA".[1] However, each amino acid has chemical and physical properties which cause it to interact with nearby

[1]This sequence was obtained from the SWISS-PROT database, at `http://ca.expasy.org/sprot/`, accession number P02144.

Amino acid	Abbreviation	Amino acid	Abbreviation
Alanine	A	Leucine	L
Arginine	R	Lysine	K
Asparagine	N	Methionine	M
Aspartic acid	D	Phenylalanine	F
Cysteine	C	Proline	P
Glutamic acid	E	Serine	S
Glutamine	Q	Threonine	T
Glycine	G	Tryptophan	W
Histidine	H	Tyrosine	Y
Isoleucine	I	Valine	V

Table 1: Naturally occurring amino acids. Source: Lesk (2002).

```
HEADER  OXYGEN TRANSPORT                19-FEB-91    2MM1               2MM1  2
COMPND  MYOGLOBIN MUTANT WITH LYS 45 REPLACED BY ARG AND CYS 110         2MM1  3
COMPND  2 REPLACED BY ALA (K45R, C110A MUTANT)                           2MM1  4
SOURCE   HUMAN (HOMO $SAPIENS) RECOMBINANT FORM EXPRESSED IN             2MM1  5
SOURCE  2 (ESCHERICHIA $COLI)                                            2MM1  6
AUTHOR  S.R.HUBBARD,W.A.HENDRICKSON,D.G.LAMBRIGHT,S.G.BOXER              2MM1  7
REVDAT  1 15-JAN-93 2MM1 0                                               2MM1  8
JRNL     AUTH S.R.HUBBARD,W.A.HENDRICKSON,D.G.LAMBRIGHT,S.G.BOXER        2MM1  9
JRNL     TITL X-RAY CRYSTAL STRUCTURE OF A RECOMBINANT HUMAN             2MM1 10
JRNL     TITL 2 MYOGLOBIN MUTANT AT 2.8 ANGSTROMS RESOLUTION             2MM1 11
JRNL     REF J.MOL.BIOL. V. 213 215 1990                                 2MM1 12
JRNL     REFN ASTM JMOBAK UK ISSN 0022-2836 070                          2MM1 13
REMARK  1                                                                2MM1 14
REMARK  2                                                                2MM1 15
REMARK  2 RESOLUTION. 2.8 ANGSTROMS.                                     2MM1 16
REMARK  3                                                                2MM1 17
REMARK  3 REFINEMENT. BY THE RESTRAINED LEAST SQUARES PROCEDURE OF J.    2MM1 18
REMARK  3 KONNERT AND W. HENDRICKSON (PROGRAM *PROLSQ*. THE R            2MM1 19
REMARK  3 VALUE IS 0.158.                                                2MM1 20
REMARK  3                                                                2MM1 21
...
SEQRES  1 153 GLY LEU SER ASP GLY GLU TRP GLN LEU VAL LEU ASN VAL        2MM1 43
SEQRES  2 153 TRP GLY LYS VAL GLU ALA ASP ILE PRO GLY HIS GLY GLN        2MM1 44
SEQRES  3 153 GLU VAL LEU ILE ARG LEU PHE LYS GLY HIS PRO GLU THR        2MM1 45
...
ATOM    1 N   GLY    1      -5.817 17.320 15.842 1.00 18.38              2MM1 66
ATOM    2 CA  GLY    1      -4.704 17.705 14.942 1.00 17.81              2MM1 67
ATOM    3 C   GLY    1      -3.356 17.578 15.656 1.00 17.07              2MM1 68
ATOM    4 O   GLY    1      -3.081 16.578 16.353 1.00 17.43              2MM1 69
ATOM    5 N   LEU    2      -2.521 18.595 15.488 1.00 16.23              2MM1 70
ATOM    6 CA  LEU    2      -1.187 18.608 16.091 1.00 15.37              2MM1 71
ATOM    7 C   LEU    2      -1.322 18.457 17.619 1.00 14.87              2MM1 72
```

Figure 1: The PDB file format. Extracts from a sample PDB file for the human myoglobin protein.

amino acids, and result in protein chains folding into repetitive structures such as helices or sheets (Baxevanis & Ouellette 2005, Berg et al. 2002). These local, repetitive structures are known as the protein's *secondary structure*.

The repetitive structures are a compact configuration for the protein chain, and this helps the protein achieve stability. The main forms of secondary structure are *alpha helices* and *beta strands* (which can also join together to form *beta sheets*). Some researchers consider alpha helices and beta strands/sheets to be the only "ordered" secondary structures (for instance, see (Uversky 2002)), whereas others include less common structures such as "3/10 helices" (for instance, (Linding et al. 2003a)). Helices other than alpha helices are rarely observed in proteins except at the ends of protein chains. Hence, we have adopted the approach of considering only alpha helices and beta strands/sheets as "ordered".

The ordered secondary structures (helices and sheets) are typically linked to each other by unstructured lengths of protein that are typically classified as "coils", "loops" and "turns". In normal circumstances, the secondary structures of proteins further fold up into a complex three-dimensional structure, specific to each protein (Berg et al. 2002, Luscombe et al. 2001); this three-dimensional arrangements of the protein chains is known as the protein's *tertiary structure* (Branden & Tooze 1991). The final shape a protein assumes depends on the exact sequence of amino acids which compose it, as well as on the properties of its environment (for instance, a protein may fold differently depending on whether it is an acid or alkaline environment, or whether the ambient temperature is warm or cool) (Raven & Johnson 1989).

One of the central tenets of structural biology is that the function of a protein is determined by its three-dimensional structure. However, it has recently been recognized that unstructured or disordered regions of protein also play critical roles in protein function (Dunker et al. 2002, Dyson & Wright 2005, Linding et al. 2003a). Although no single consistent definition has yet been developed, loosely speaking, disordered protein segments are highly flexible regions of protein whose structures are difficult to investigate experimentally. Segments of protein which form stable three-dimensional structures are described as *ordered*, and those that do not as *disordered* (Cheng et al. 2005). It is also believed that there are several different types of protein disorder (Dunker et al. 2002, Linding et al. 2003a).

```
#   RESIDUE AA ... ACC  G   H   I   T   E   B   S   L ...

31    71 A G  ...   0   0   0   0   0   0   0   0 100 ...
32    72 A V  ...   0   0   0   0   0   0   0 100   0 ...
33    73 A R  ...  36   0   0   0  10   0   0  90   0 ...
34    74 A V  ...   0   0   0   0   0   0  87  13   0 ...
35    75 A D  ...   5   0   0   0   0   0   0   0 100 ...
36    76 A L  ...  14   0   0   0   0   0   0   0 100 ...
37    77 A G  ...  19   0   0   0   0   0   0   0 100 ...
38    78 A E  ...  50   0   0   0   0 100   0   0   0 ...
```

Figure 2: Extracts from a sample DSSPcont file.

2.2 The Protein Data Bank and derived databases

The main source of protein structural data is the PDB. The PDB contains data on over 32,000 proteins (over 21 gigabytes of data), and new proteins are added weekly. The primary format for exchange of PDB data is a text-based flat-file format. Each file contains data on the coordinates of the atoms in one protein, and details of the experimental procedures which created that data (Baxevanis & Ouellette 2005, Berman et al. 2000, Cohen 2004). The files consist of a *body*, containing the 3D coordinates of all atoms in the protein (usually about 2000 atoms per protein), and a *header* containing metadata about the protein and how the coordinates were derived. Figure 1 shows extracts from a PDB file.

The DSSP database[2] is derived from PDB data (Carter et al. 1983). It uses atomic coordinate data from PDB to derive a description of a protein's secondary structure. Each amino acid in a protein is described as being part of various sorts of helix, sheet, or loop. A later database which extends the approach of DSSP is the DSSPcont database[3] (Carter et al. 1983). Whereas DSSP assigns amino acids to a set of discrete structural categories (a process called "structural assignment"), DSSPcont performs *continuous* structural assignment—for each structural category, a *probability* is given that the amino acid falls into that category. DSSPcont is particularly useful in a study of disordered proteins, because the continuous assignment process captures flexibility in the positions of residues.

At the time of initial development of our database, the DSSPcont database contained 20,216 text-based flat files. Each file in the database records information on one protein; the files are named using the PDB accession number for the protein, with a ".dsspc" extension. A single protein may consist of multiple protein chains, each identified by a single-letter abbreviation ("A", "B", and so on), and chains are separated by a delimiter record in which the amino acid type is marked as "!" (an exclamation mark). The total set of files is 1815 MB in size.

For each residue, DSSP and DSSPcont assign it to one of eight categories of structure, as shown in Table 2. Such a detailed classification is more than we need, however, since our aim is simply to divide residues into the two categories of "structured" or "unstructured". For our purposes, only three of the categories are considered "structured", namely alpha helices, beta strands, and beta sheets. All the other types of structure are considered "unstructured".

[2]`http://swift.cmbi.ru.nl/gv/dssp/`

[3]Available at `http://cubic.bioc.columbia.edu/services/DSSPcont/`.

3 Extracting Disordered Protein Segments

In the present research, we aim to develop a database of protein structural data (ultimately derived from PDB data), and query it to locate disordered protein segments.

At the time of writing, we have developed a prototype which implements the bottom (database) level of the system, and processes queries made on the data. The top (presentation) level remains to be implemented.

As noted previously, it is generally accepted that a low degree of secondary structure combined with high flexibility implies disorder. Our query can be expressed as follows: given a threshold T_u of secondary-level unstructuredness, and a threshold T_f of flexibility, which residues in a set of protein chains exceeded both these thresholds? That is, which residues r have unstructuredness $r_u \geq T_u$ and flexibility $r_f \geq T_f$?

The DSSPcont database was selected as the most easily usable source of reliable structure and flexibility data. For any specified amino acid, the DSSPcont format provides us with information on its propensity for secondary structure or lack of structure, and a measurement of its SA.

3.1 Data cleaning and transformation: computing relative solvent accessibility

Solvent accessibility (SA) measures the proportion of an amino acid's surface which is exposed to the solvent surrounding a protein (for instance, water). The interior of a protein is densely packed; thus, SA is low for amino acids in the interior of a protein, and high for those on the surface. SA is measured in units of square Ångstrøms. [4] In general, the higher the SA of a region of residues, the more flexible it is.

DSSPcont provides raw SA measurements for amino acids. However, since amino acids vary in size, this figure is not comparable between different amino acids: a certain number of square Ångstrøms might constitute a very small proportion of a large amino acid, but a large proportion of a small one. We therefore normalize the SA of amino acids: the *relative* solvent accessibility (RSA) of an amino acid is the ratio of absolute solvent accessibility to the maximum observed accessibility for the amino acid type.

Different researchers have developed different methods of determining the maximum surface area, which give slightly different measurements (Richardson & Barlow 1999). Since we consistently adopt the values observed by Ahmad et al. 2003, all the RSA values we calculate are comparable. However, they are not comparable with RSA values obtained by different normalization methods. If one wants to compare RSA values between our data set and another which uses a different normalization method, RSA values for one or the other of the data sets would need to be recalculated.

[4]One Ångstrøm equals 0.1 nanometres.

Structure	Label	Structure	Label
Alpha helix	A	Extended beta strand	E
3/10 helix	G	Turn	T
Pi helix	I	Bend	S
Beta bridge	B	Other/loop	L

Table 2: Structural categories used by DSSPcont.

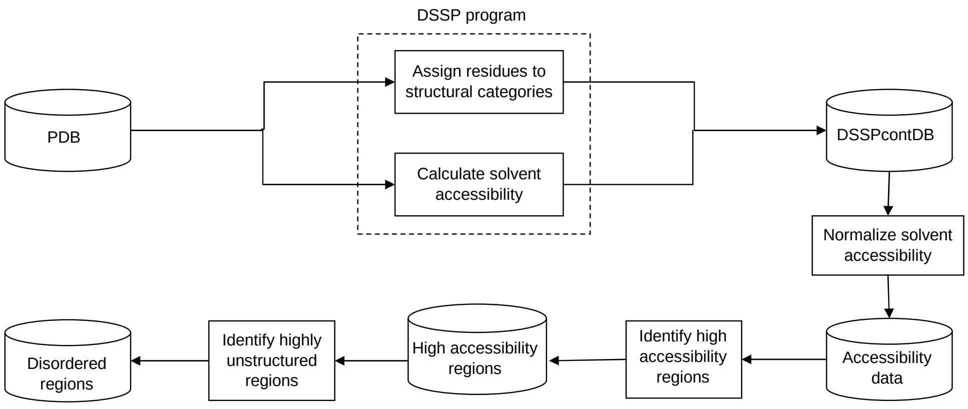

DSSP program
Assign residues to structural categories
Calculate solvent accessibility
PDB
DSSPcontDB
Normalize solvent accessibility
Disordered regions
Identify highly unstructured regions
High accessibility regions
Identify high accessibility regions
Accessibility data

Figure 3: Processing of data

Although the files in the PDB are generally of high quality, some forms of "noise" and bad data do occur. One form of bad data we detect is residues that have been marked as "X" for "unknown" in the DSSPcont data files—is detected, and these residues are "sequestered": they are not included in any statistical analyses of the data (but can be inspected separately). Other more subtle forms of noise can also occur, and represent a possibility for future work in this area.

Two conflicting requirements in relation to data noise are that although users typically want to remove or lessen the effects of noise, they often also want access to the original experimental data on which a database was based, and to have good records of the "provenance" (or history) of the data (Jagadish & Olken 2003). Our approach to resolving these conflicting requirements is to retain and make available the original DSSPcont data, but to allow "noise" to be screened out at the query stage.

3.2 Incremental query processing: creating a RSA repository

As discussed previously our queries to DSSPcont for disordered residues are defined by two criteria: (1) a high level of unstructuredness for a residue, and (2) and a high level of RSA . The level of unstructuredness for a residue r is the probability that the residue is *not* part of an alpha helix or beta strand. By referring to Table 2, we can see that this is therefore the sum of the probabilities that the residue is part of a 3/10 helix (G), pi helix (I), turn (T), bend (S) and "other/loop" (L):

$$unstructuredness = P(G) + P(I) + P(T) + P(S) + P(L)$$

We wish to identify regions where the unstructuredness exceeds a threshold for unstructuredness α—$unstructuredness \geq \alpha$—as well as having a high level of RSA, $RSA(r) \geq \beta$, where β is the threshold for RSA .

One simple way to identify disordered residues is to iterate over all the residues in the DSSPcont database, calculate the RSA and unstructuredness for each residue and identify which residues meet both criteria. A notable problem with the approach is the the large number of costly disk reads (recall that there are 20,216 files in the DSSPcont database). Thus, we only perform calculation of RSA once, creating an RSA repository with RSA values for the entire DSSPcont database.

On the other hand, to compare the disordered regions obtained by our approach with those by other approach such as DisProt, we need to run many disorder-extraction queries with different parameter settings for unstructuredness and flexibility. In order to amortize space and time costs over multiple queries, we therefore process queries incrementally—caching the result of the first step for use in later queries. For instance, one might wish to hold the RSA threshold constant at, say, 25%, and use 5 different parameters for the threshold of unstructuredness to see what effect this would have. Firstly, regions are identified which meet the 25% RSA threshold and the results are cached. Then, when looking for regions which also meet the unstructuredness criteria, we need only consider the high-RSA regions, rather than iterating over the whole DSSPcont database. Of course, it is also possible to detect regions in the opposite order—unstructured regions first, and then high-RSA regions within them—but for brevity, we restrict ourselves here to discussing the case in which we first identify high-RSA regions, and then disordered regions within those high-RSA regions.

3.3 The complete procedure

Our complete procedure for extracting disordered segments from DSSPcont is shown in Figure 3. The process of creating the disordered database is as follows:

1. Firstly, normalize absolute SA for all residues in the DSSPcont database and store these in a repository of RSA data. This repository is used in the next stage for identifying high-accessibility regions.

2. Identify high-RSA regions, by iterating over the DSSPcont database and referring to the repository of RSA data. The algorithm for identifying high-RSA regions is shown in Figure 4.

3. Iterate over the set of high-RSA regions and identify sub-regions of those regions which addition-

```
Input:
  • A database $RSADB$ of amino acid records
  • A minimum RSA threshold $\beta$
  • A minimum length threshold $T_{length}$
Output:
  • A set $HighRSA$ of regions of high solvent accessibility
Method:
1. for each protein chain $C$ in $RSADB$:
2.    for each residue $r$ in $C$:
3.       if $RSA_r \geq \beta$ and $StartPos$ has not been set:
4.          $StartPos \leftarrow r$
5.       if $RSA_r < \beta$ and $StartPos$ has been set:
6.          $EndPos \leftarrow r$
7.          $length \leftarrow EndPos - StartPos + 1$
8.          if $length > T_{length}$:
9.             add $\langle StartPos, EndPos \rangle$ to $HighRSA$
```

Figure 4: Algorithm: Locate regions of high relative solvent accessibility

```
Input:
  • A set $HighRSA$ of regions of high solvent accessibility
  • A set $DSSPcont$ of mappings from $ResidueID$ to structural assignments
  • A minimum unstructuredness threshold $\alpha$
  • A minimum length threshold $T_{length}$
Output:
  • A set $DisorderedRegions$ of disordered regions
1. for each region $R$ in $HighRSA$:
2.    for each residue $r$ in $R$:
3.       From $DSSPcont$, look up $P(G)_r$, $P(I)_r$, $P(T)_r$, $P(S)_r$ and $P(L)_r$
4.       Calculate the degree of unstructuredness of $r$:
            $unstructuredness_r \leftarrow P(G)_r + P(I)_r + P(T)_r + P(S)_r + P(L)_r$
5.       if $unstructuredness_r \geq \alpha$ and $StartPos$ has not been set:
6.          $StartPos \leftarrow r$
7.       if $unstructuredness_r < \alpha$ and $StartPos$ has been set:
8.          $EndPos \leftarrow r$
9.          $length \leftarrow EndPos - StartPos + 1$
10.         if $length > T_{length}$:
11.            Add $\langle StartPos, EndPos \rangle$ to $DisorderedRegions$
```

Figure 5: Algorithm: Locate disordered regions

ally are highly unstructured (that is, have a low level of secondary structure). The algorithm for identifying disordered regions is shown in Figure 5.

In step 1, since all residues in the DSSPcont data set must be iterated over, the time taken to create the RSA repository will be proportional to the number of residues in the DSSPcont database, that is, the algorithm is of $O(n)$ time complexity. The space required is likewise proportional to the number of residues. Once RSA values have been calculated for all residues, we then have enough information to detect disordered regions. We also allow a user to specify a minimum length of disordered region they wish to detect. This can help save on disk usage. It is less likely that a very short region will provide statistically useful information; if users are only interested in, say, regions over 20 residues length, we can save space by ignoring all regions less than that length

In step 2, to detect high-RSA regions, we iterate over the DSSPcont database, this time using our database of RSA values to look up the RSA for each residue. When the RSA rises above the threshold, we record this as being the possible start of a high-RSA region. When the RSA falls below the threshold again, we have reached the end of a possible region. If the length of the region exceeds the user-specified length threshold, we add it to the result set of high-RSA regions. In step 3, the algorithm for identifying disordered regions iterates over all residues in the previously detected high-RSA regions, and similarly to the previous algorithm, flags any sub-regions which exceed the unstructuredness threshold. Both algorithms are straight forward, and are of $O(n)$ complexity, where n is the number of residues under consideration.

3.4 Implementation

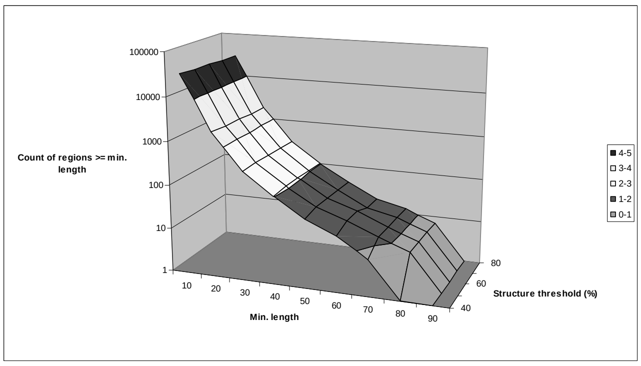
The Perl and Python languages are the most frequently used in the bioinformatics field (Cohen 2004) and are therefore a natural choice for developing data processing routines for our database. We have made use of Python for our data processing routines, and some use of Perl for data analysis.

Parsers exist for the DSSP format, but none for the newer DSSPcont format. We therefore created a parser for the DSSPcont format by adapting algorithms used in the BioPython (Chapman & Chang 2000) (see `http://www.biopython.org/`) and BioPerl (Stajich et al. 2002) (see `http://www.bioperl.org/`) open source bioinformatics toolkits.

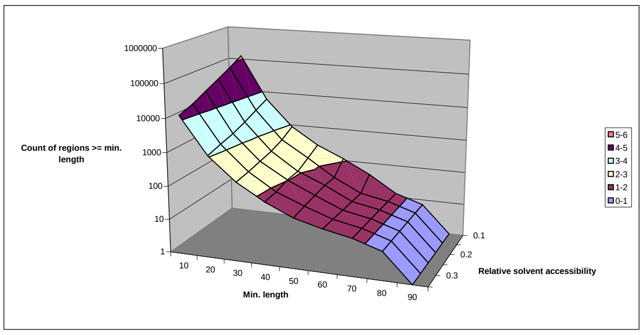
The DSSPcont format contains a range of header information (see Figure 2) not used in the present research application, which we therefore ignore. This includes fields such as "AUTHOR" (identifying the authors of the research which solved the structure of the relevant protein) and statistics of various sorts

```
>sp|P02144|MYG_HUMAN Myoglobin - Homo sapiens (Human).
GLSDGEWQLVLNVWGKVEADIPGHGQEVLIRLFKGHPETLEKFDKFKHLKSEDEMKASED
LKKHGATVLTALGGILKKKGHHEAEIKPLAQSHATKHKIPVKYLEFISECIIQVLQSKHP
GDFGADAQGAMNKALELFRKDMASNYKELGFQG
```

Figure 6: Example of FASTA file format. Source: SWISS-PROT database, at `http://ca.expasy.org/sprot/`.



(a) Disordered regions detected as unstructuredness threshold is varied.



(b) Disordered regions detected as RSA threshold is varied.

Figure 7: Disordered protein regions under different parameters

on the chemical bonds and structures within the protein (for instance, the total number of various types of hydrogen bonds). The data of interest to us is the residue records—one line of text per amino acid which list each residue in the protein, and give columns containing the structural data. Each line is 169 characters long, and data columns are identified by their position in the line for instance, the amino acid type of a residue is specified in the 13th character, and the SA in positions 34 to 37.[5]

There are some challenges in parsing DSSPcont data and integrating the results with other data sources. For instance, the ID numbers allocated to each residue in a protein do not correspond to those used by many other formats (although fortunately, DSSPcont also records PDB residue ID numbers). Additionally, the DSSPcont records contain discontinuities where the end of a protein chain is reached or where a region of missing structural data in the PDB is encountered, and these must be detected and handled by the parsing routine.

The output of normalizing solvent accessibility for all residues in the DSSPcont database (step 1 of the complete procedure for extracting disordered regions) is an index which can be used to look up the RSA for any specified residue in the DSSPcont data set, where a residue is specified by a protein ID, a chain ID and a residue position number. These identifiers are based on the PDB system—the protein ID is the PDB accession number, the chain ID is the PDB chain ID, and so on. As with any index, our index can be modelled as a set of mappings from input (in this case, a residue ID—a tuple $\langle \mathit{ProteinID}, \mathit{ChainID}, \mathit{ResiduePosition} \rangle$) to output (the RSA value). The function was implemented in Python; internally, the index is stored as a set of serialized Python data structures, since the code for reading and writing these to and from disk has been optimized to be very fast. The RSA data for each protein is serialized as a single file, within which the RSA for a specific amino acid can be looked up in constant time.

Both the algorithm for identifying high-RSA regions (Figure 4) and the algorithm for identifying disordered regions (Figure 5) output sets of regions. Internally, our implementations of these algorithms store the sets as serialized Python data structures. For each protein, a serialized file contains a list of the detected regions for that protein. Fully specifying the location of a region requires specifying its start and end—thus, a region consists of a pair of residue IDs.

4 Managing the Disordered Protein Database

As discussed in Section 3, our disordered protein region extraction procedure are run with different unstructuredness and flexibility parameter settings so as to examine how these parameters affect the regions extracted and to verify the feasibility of our definition of disorder. The results are shown in Figure 7.

Figure 7(a) shows the effect on the number of regions found as we vary the level of unstructuredness required. It would appear that varying this threshold causes some variation, but has not nearly so pronounced an effect as does length. This seems to be because although DSSPcont uses a "continuous" method of assigning structure categories to residues, many residues show only either a very high chance of belonging to a category, or a very low one. That is, given a particular category (say, "alpha helix"), it appears that most residues are either given a high (often 90% to 100%) chance of belonging to that category, or a very low one (say, 0% to 10%), with few residues falling into intermediate probabilities.

Figure 7(b) shows the effect on the number of regions found as we vary the RSA threshold. For long regions (greater than about 65 residues in length), varying the adsAcRSA threshold does not seem to have a great impact on the number of regions detected, whereas for shorter regions, varying the adsAcRSA threshold from 0.4 to 0.1 can result in more than 10 times as many regions being detected. This suggests that where very long (> 65 residue length) regions occur at all, they have a very high adsAcRSA (> 0.4), whereas shorter regions have a lower and more variable adsAcRSA.

A question that is left to answer is how to store and manage the extracted disordered protein regions. This question is complicated by the complexity of our data. Considering the difficulty of applying generic Database Management Systems (DBMSs) and the

[5] `http://cubic.bioc.columbia.edu/services/DSSPcont/DSSPcont.html`.

characteristics that the data is used, we propose to manage the database with our custom data management tools. Specifically, data is stored internally as flat text files and output is in XML and text-based FASTA format for compatibility and interoperability reasons.

Generic DBMSs are in general not suitable for managing disordered segments. This is partly due to the fact that the data they contain is complex in structure. With relational DBMSs, representing the position and structural information of disordered segments is difficult and awkward. Alternatively, simply storing the data as unstructured binary or text-based fields (that is, BLOBs, or Binary Large OBjects, and CLOBs, or Character Large OBjects) means additional routines must be written for interpreting and using these fields.

Additionally, as with many biological databases, the patterns of access in our database differ significantly from most transactional databases. New data is required to be added at a reasonably slow rate (when compared with transactional systems), and once added, is only retrieved by users rather than edited. Only a small number of users require write access to the system. As a result, features of generic DBMSs such as concurrency control, transaction-based processing, and so on may not be needed (Han & Kamber 2001).

On the other hand, historically, many biological databases adopted the approach of storing their data as text-based flat files, and writing customized routines to parse and manage this data (Nelson et al. 2003). Because flat files are so widely used in the bioinformatics world, supporting them, at least as a format which can be exported, is important. Many bioinformatics analysis tools which operate on sequence data only accept data in the "FASTA" format. Figure 6 shows an example of the format, which is very simple. FASTA files begin with a "comment" line which starts with the ">" character, and are followed by 60-character-width lines containing the actual sequence. Thus, rather than using a generic DBMS, we use the operating system file system directly, and define our own routines for storing and managing data in FASTA format—effectively creating a custom DBMS. Due to the effort involved in implementing all the features (such as transactions, for instance) typically found in generic DBMSs, our system is far simpler.

XML (Bray et al. 2000) is becoming a widely used format for the storage and interchange of biological data (Srdanovic et al. 2005). Considering the interoperability of our data in future applications, we also support managing data in the XML format.

5 Conclusions

We have presented a prototype system for building a disordered protein database from PDB data using the derived secondary structure database DSSPcont. We have discussed the data management issues that arise in building the system, and in particular, we have proposed techniques for data cleaning, incremental query processing and data storage and analysis. We have also discussed implementation considerations on the management of protein sequence data. Work is underway to develop tools for analyzing the protein disorder databases and to make our prototype system publicly available.

Our system has demonstrated that our approach for building biological databases is feasible. Our system has also reflected on that managing biological data should follow the "business rules" in biology. The experimental nature of the biological rules leads to the uncertainty in data and calls for specialized tools for managing different types of biological data. Indeed our work is an initial part of a bioinformatics project focusing on the study of disordered proteins. Our future work will focus on developing a suite of tools for querying and mining the disordered database.

Acknowledgements

The authors would like to thank Dr. Zhi-Ping Feng from The Walter and Eliza Hall Institute of Medical Research for helpful discussions.

References

Ahmad, S., Gromiha, M. A. & Sarai, A. (2003), 'Real value prediction of solvent accessibility from amino acid sequence', *Proteins* **50**(4), 629–635.

Baxevanis, A. D. & Ouellette, B. F. F. (2005), *Bioinformatics: A Practical Guide to the Analysis of Genes and Proteins*, John Wiley & Sons, Hoboken, NJ.

Berg, J. M., Tymoczko, J. L. & Stryer, L. (2002), *Biochemistry, Fifth Edition: International Version*, W. H. Freeman.

Berman, H. M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T. N., Weissig, H., Shinyalov, I. N. & Bourne, P. E. (2000), 'The Protein Data Bank', *Nucleic Acids Research* **28**, 235–242.

Branden, C. & Tooze, J. (1991), *Introduction to Protein Structure*, Garland Publishing, NY.

Bray, T., Paoli, J., Sperberg-McQueen, C. M. & Maler, E. (2000), *Extensible Markup Language (XML) 1.0 Second Edition W3C Recommendation*, Technical Report REC-xml-2001006, World Wide Web Consortium.

Brusic, V., Wilkins, J. S., Stanyon, C. A. & Zeleznikow, J. (1998), Data learning: Understanding biological data, *in* G. Merrill & D. K. Pathak, eds, 'Knowledge Sharing Across Biological and Medical Knowledge Based Systems: Papers from the 1998 AAAI Workshop', AAAI Press, pp. 12–19.

Bry, F. & Kroger, P. (2003), 'A computational biology database digest: Data, data analysis, and data management', *Distributed and Parallel Databases* **13**(1), 7–42.

Carter, P., Andersen, C. A. F. & Rost, B. (1983), 'DSSPcont: continuous secondary structure assignments for proteins', *Nucleic Acids Research* **31**(13), 3293–3295.

Chapman, B. & Chang, J. (2000), 'Biopython: Python tools for computational biology', *SIGBIO Newsletter* **20**(2), 15–19.

Cheng, J., Sweredoski, M. & Baldi, P. (2005), 'Accurate prediction of protein disordered regions by mining protein structure data', *Data Minining and Knowledge Discovery* **11**(3), 213–222.

Cohen, J. (2004), 'Bioinformatics—An introduction for computer scientists', *ACM Computing Surveys* **36**(2).

Dunker, A. K., Brown, C. J., Lawson, J. D., Iakoucheva, L. M. & Obradovic, Z. (2002), 'Intrinsic disorder and protein function', *Biochemistry* **41**(21), 6573–6582.

Dyson, H. J. & Wright, P. E. (2005), 'Intrinsically unstructured proteins and their functions', *Nature Reviews* **6**, 197–208.

Gupta, A. (2004), 'Life science research and data management', *SIGMOD Record* **33**(2), 12–14.

Han, J. & Kamber, M. (2001), *Data Mining: Concepts and Techniques*, Morgan Kaufmann Publishers, San Francisco, CA.

Herbert, K. G., Gehani, N. H. & Piel, W. H. (2004), 'BIO-AJAX: an extensible framework for biological data cleaning', **33**(2), 51–57.

Jagadish, H. V. & Olken, F. (2003), 'Database management for life science research: Summary report of the workshop on data management for molecular and cell biology at the National Library of Medicine, Bethesda, Maryland, February 2–3, 2003', *OMICS* **7**(1), 131–137.

Lesk, A. M. (2002), *Bioinformatics*, Oxford University Press, Oxford, UK.

Linding, R., Jensen, L. J., Diella, F., Bork, P., Gibson, T. J. & Russell, R. R. (2003a), 'Protein disorder prediction: implications for structural proteomics', *Structure* **11**(11), 1453–1459.

Linding, R., Russell, R. B., Neduva, V. & Gibson, T. J. (2003b), 'GlobPlot: Exploring protein sequences for globularity and disorder', *Nucleic Acids Research* **31**(13), 3701–3708.

Luscombe, N. M., Greenbaum, D. & Gerstein, M. (2001), 'What is bioinformatics? A proposed definition and overview of the field', *Methods of Information in Medicine* **40**, 346–358.

Nelson, M. R., Reisinger, S. J. & Henry, S. G. (2003), 'Designing databases to store biological information', *BIOSILICO* **1**(4), 134–142.

Raven, P. H. & Johnson, G. B. (1989), *Biology*, Times Mirror/Mosby College Publishing, St Louis, Missouri.

Richardson, C. J. & Barlow, D. J. (1999), 'The bottom line for prediction of residue solvent accessibility', *Protein Engineering* **12**(12), 1051–1054.

Shui, W. M., Wong, R. K., Graham, S. C., Lee, L. K. & Church, W. B. (2003), A new approach to protein structure and function analysis using semi-structured databases, *in* Y.-P. P. Chen, ed., 'First Asia-Pacific Bioinformatics Conference (APBC2003)', Vol. 19 of *Conferences in Research and Practice in Information Technology*, Australian Computer Society, Inc., Adelaide, Australia, p. ???

Singh, A. K. (2003), 'Querying and mining biological databases', *OMICS: A Journal of Integrative Biology* **7**(1), 7–8.

Smail-Tabbone, M., Osman, S., Messai, N., Napoli, A. & Devignes, M.-D. (2005), Bioregistry: A structured metadata repository for bioinformatic databases, *in* 'CompLife 2005 (First International Symposium on Computational Life Science, Konstanz, Germany, September 25-27, 2005)', Vol. 3695, Springer, Berlin, pp. 46–56.

Srdanovic, M., Schenk, U., Schwieger, M. & Campagne, F. (2005), 'Critical evaluation of the JDO API for the persistence and portability requirements of complex biological databases', *BMC Bioinformatics* **6**(1), 5.

Stajich, J. E., Block, D., Boulez, K., Brenner, S. E., Chervitz, S. A., Dagdigian, C., Fuellen, G., Gilbert, J. G., Korf, I., Lapp, H., Lehvaslaiho, H., Matsalla, C., Mungall, C. J., Osborne, B. I., Pocock, M. R., Schattner, P., Senger, M., Stein, L. D., Stupka, E., Wilkinson, M. D. & Birney, E. (2002), 'The Bioperl toolkit: Perl modules for the life sciences', *Genome Research* **12**(10), 1611–1618.

Uversky, V. N. (2002), 'What does it mean to be natively unfolded?', *European Journal of Biochemistry* **269**, 2–12.

Vucetic, S., Obradovic, Z., Vacic, V., Radivojac, P., Peng, K., Iakoucheva, L. M., Cortese, M. S., Lawson, J. D., Brown, C. J., Sikes, J. G., Newton, C. D. & Dunker, A. K. (2005), 'DisProt: A database of protein disorder', *Bioinformatics* **21**(1), 137–140.

Wooley, J. & Lin, H., eds (2005), *Catalyzing Inquiry at the Interface of Computing and Biology*, The National Academies Press, Washington.

Efficient Similarity Search by Summarization in Large Video Database

Xiangmin Zhou, Xiaofang Zhou and Heng Tao Shen

School of Information Technology and Electrical Engineering
University of Queensland,
St. Lucia, QLD 4072 Australia,
Email: {emily,zxf,shenht}@itee.uq.edu.au

Abstract

With the explosion of video data, video processing technologies have advanced quickly and been applied into many fields, such as advertisements, medical etc.. To fast search these video data, an important issue is to effectively organize videos by data compacting and indexing. However, practically, many useful distances for video comparison are suitable to human perception, but non-metric. Therefore, traditional high dimensional data structures can not be utilized to index videos directly when non-metric measures are applied. In this paper, we propose a compact video representation based on global summarization, by which each video in database is mapped into a digital string(a series of cluster id). Consequently, the inter-frame similarity measure is transformed into inter-cluster comparison. Then, we propose an efficient index strategy based on sequence decomposition and reconstruction, by which the spatial index methods can be utilized with non-metric measures for video similarity search. We employ an optimal B^+-tree with an inverted list attached, for quickly identifying similar clusters and locating potentially similar videos respectively. Finally, a clustering based query summarization technique is proposed, which can greatly reduce the IO and CPU cost in the query processing by batch mapping.

Keywords: Video Summarization, Decomposition and Reconstruction, Batch Query Mapping, Multi-Symbol Representation.

1 Introduction

Recently, content-based video search has attracted much attention, due to its wide applications in many areas such as advertising, news video broadcasting and personal video archive. In a video search system, a user typically wants to retrieve videos which are most similar to a video example or the video in his mind.

For *efficient* video similarity search in large video database, there are two typical approaches: compact video representation and effective video index. The first approach is to reduce the computational cost of similarity measure between two videos by representing them in a much simpler way. As a video sequence typically consists of a large number of frames which are high dimensional vectors, and inter-video similarity is generally measured by identifying the similarity match for every single frame in each sequence, which results in high computational cost for large video databases. The second approach is to reduce the number of sequence comparisons in large video database by indexing videos efficiently. For a video database which contains large numbers of videos, identifying similar videos by exhaustive scanning all videos is apparently undesirable. We need to index videos (or their summaries) in an efficient way, thus limiting the search space and reducing the number of inter-video comparisons. However, the existing similarity search methods have not addressed both of two issues with a non-metric distance function.

In (Shen, Ooi & Zhou 2005), the authors proposed ViTris for video summarization, and optimal B^+-tree for indexing, however, they neglected the temporal ordering in a video sequence, thus possibly compromising the effectiveness of similarity search. In (Lee, Chun, Kim, Lee & Chung 2000), the authors proposed to summarize each video sequence by several Minimum Bounding Rectangles (MBRs), and to use R-tree (Guttman 1984) for indexing. However, this video representation method was customized for mean Euclidean distance measure, which preserves the temporal order in a video sequence only in a strict manner. Gaps within video sequences cannot be dealt with well.

In this paper, we propose a summarization based video similarity search approach, which has the following main features: Firstly, we symbolize each video sequence as a digital string. This video symbolization makes each video be represented compactly, moreover, it facilitates video sequence matching. Secondly, we propose an index strategy based on sequence decomposition and reconstruction, which can use traditional index structure to manage video summaries for efficient video search with non-metric measures. We perform the video sequence matching by using an edit distance variant, which is a non-metric measure. It takes into consideration not only the temporal ordering inherent in video sequences, but also the inter-frame similarity between two compared sequences. And finally, we employ some optimizations to improve the search performance and effectiveness, which includes novel query summarization method and multi-symbol representation method.

The rest of the paper is organized as follows. The related work to video search is presented in Section 2. Section 3 describes our approach, including video representation, indexing, and similarity search. Section 4 presents our performance study, and Section 5 concludes the whole paper.

2 Related Work

In general, the method used to address the video sequence matching problem depends on the technique used to represent the sequences (Adjeroh, Lee,& King 1999). One popular video representation technique is to represent each video sequence as key frames. This type of representation is mainly used for video browsing (Chang, Sull & Lee 1999, Zhu, Wu, Fan, Elmagarmid & Aref 2004). Several video representations (Lee et al 2000, Cheung & Zakhor 2003, Shen et al 2005) are proposed in recent years for effective video similarity search. We only briefly review a few representatives.

Lee et al. (Lee et al 2000) proposed to partition each video sequence into subsequences, and represent each subsequence by a Minimum Boundary Rectangle(MBR). Accordingly, the query processing is based on these MBRs, instead of scanning data elements of entire sequences. Thus, it is fast and needs small storage overhead compared with a sequential scan. To index these MBRs, a R-tree structure is utilized. The main problem with this representation is that, it can not be used for similarity measure with local time shifting. As such, some similar videos may be considered as dissimilar due to frame insertion or deletion.

In (Cheung et al 2003), authors proposed a video representation named *Video Signature* (or ViSig). Its basic idea is to summarize each video with a small set of its sampled frames, called video signature, and estimate the percentage of similar frames shared by two sequences by computing the percentage of similar ViSig frame pairs. In our previous work (Shen et al 2005), we introduced a video representation model called ViTri. A sequence is first summarized into a small number of clusters which contain similar videos, and each cluster is modelled as a hypersphere in a high-dimensional space, which is represented by a video triplet (or ViTri). As such, the inter-video similarity is measured by the inter-cluster similarity of them, which is in turn estimated by the number of similar frames shared among the clusters (Shen et al 2005). To further facilitate the search process, ViTris are indexed by an optimal B$^+$-tree. However, both of these two video representation methods do not consider the temporal ordering in similarity measure.

We improve the previous work for content-based video search by symbolizing video sequences, which reduces the dimensionality of frame and keep the temporal order in video sequence. Many high-dimensional indexing methods have been proposed (Bohm, Berchtold & Keim 2001). In our work, for non-metric measure, we propose a sequence decomposition and reconstruction based indexing strategy to index the clusters and symbol segments(a Triplet representing a series of identical symbols) by employing an index structure which is closely related to iDistance (Jagadish, Ooi, Tan, Yu & Zhang 2005) or similar indexing methods. In these methods, high-dimensional data are first transformed into single dimensional data, and single-dimensional indexing methods such as B$^+$-tree are then used. For example, in iDistance, by selecting a reference point and computing the distance of each high-dimensional point to the reference point, a high-dimensional point is transformed into a single-dimensional value.

3 Our Approach

In this section, we describe in detail our similarity search approach, which includes the following aspects: how a video sequence is represented, how sequences are decomposed and reconstructed, how indexes are constructed, and finally, give a query video, how similarity search is performed and optimized.

3.1 Video Representation

As mentioned earlier, each video sequence consists of a large number of frames, which results in high computational cost in similarity search. Thus, we need to represent video sequences in a compact and effective way.

Based on the observation, that nearby frames in a video are very similar, and also similar videos usually share similar frames, we can summarize a video sequence by the clusters that contain similar frames. Specifically, suppose each video frame in a database is represented as a high-dimensional image feature vector, we employ a hierarchical clustering method (e.g., k-means) and perform clustering over all frames in the database. As such, the whole database is formed into a set of clusters, each containing similar frames with inter-frame distances (measured by Euclidean distance) less than or equal to ϵ (called *clustering threshold*). We represent a cluster C by $< id, O, r, N >$, where

1. id is the cluster identifier;
2. O is the cluster centre which indicates the position of the cluster in the original high dimensional space;
3. r is the radius of boundary hypersphere of the cluster.
4. N is the number of frames in the cluster.

Suppose each video frame is represented by its cluster id, we can symbolize each video sequence as a digital string which consists of the cluster ids of its video frames. This video symbolization makes each video be represented compactly, moreover, it facilitates video similarity matching with consideration of temporal ordering.

3.2 Two-layer Video Indexing

To facilitate video search, we need to organize video data by indexing. Since video sequences can not be indexed directly by traditional index structures when a non-metric distance measure is utilized. We adopt video decomposition strategy before video data are indexed. In a video sequence, a series of consecutive frames that are mapped into the same symbol consists of a *video segment*, which can be represented by $< vid, pos, len >$, where vid is the identifier of the video where the symbol appears; pos is the first position of the symbol in the video; len is the length of the segment that shows the frequency of the symbol appearance in the video.

We build index on compact video representations. Our index structure includes two parts: (1) An optimal B$^+$-tree for indexing clusters which contain similar video frames in the database; (2) An inverted file for indexing the decomposed video segments. As B$^+$-tree is used for indexing one-dimensional data, we first map each cluster center, O, a high-dimensional vector, into a one-dimensional value based on O's distance to a selected reference point, and then use this one-dimensional value as a key in B$^+$-tree. For selecting the reference point, we borrowed the method employed in (Shen et al 2005), where an optimal reference point lies on the line identified by the first principle component and out of its variance segment.

Figure 1 shows the index structure. Each leaf node in the structure may have several entries, and each entry contains a key with a cluster and a pointer to

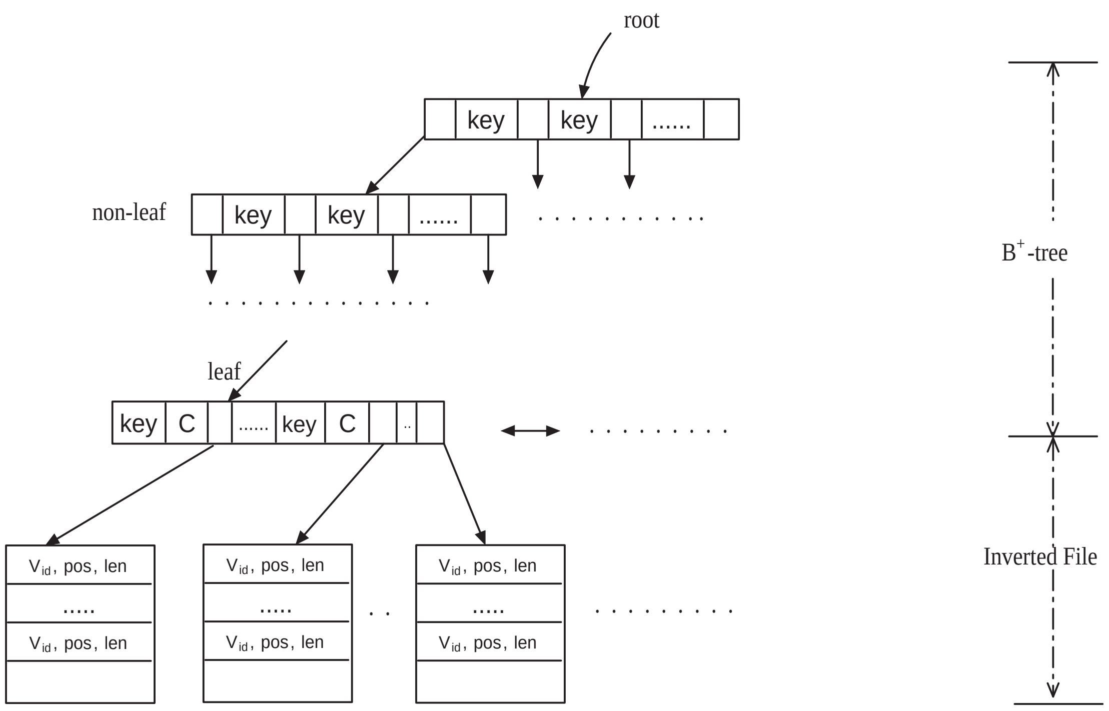


Figure 1: The two-layer index structure.

the corresponding position of the inverted file. The inverted file is used for locating a video segment, i.e., which video sequence a symbol appears in.

3.3 Similarity Search

Given a query video, we need to search the database for most similar videos. During a search, three steps are mainly involved: query mapping, sequence reconstructing, and sequence matching. In the following, we describe each step in detail.

Query mapping For each frame in a query video, we first map it to a set of symbols which are most similar to itself and whose distances from it are within a given similarity threshold ϵ. With our index structure, we can easily identify these symbols. According to the triangular inequality, these symbols' key values should be between $|x - \epsilon|$ and $|x + \epsilon|$, where x is the distance of the query frame to the selected reference point. However, in this process, a frame may not be mapped into any cluster. It shows that this frame is dissimilar with any video frame in the database. In this case, this frame is represented by a special symbol '$-$', which is dissimilar with any symbol. Further, according to these symbols (except '$-$'), the potentially similar video segments can be retrieved by the inverted file. By this process, any dissimilar video segment is filtered out. Thus, we can symbolize the query video with the potentially similar videos identified.

To further reduce the number of similar candidates, we adopt the following two filtering strategies: (1) filtering small symbol segments; (2) filtering common symbol segments. The former one is to filter out the candidates that has low similarity with the query. If only few small segments belong to a certain video, these segments will be filtered out from the candidate set. The latter one is to filter out the discriminative segments. If most of the video candidates include the segments consisting of the same symbol, these segments can not contribute to the final query results. These symbol segments will be thrown off.

Sequence reconstructing For video segments obtained by query mapping, we need to recompose symbol sequences according to the location information of each similar segment. To ensure the proper similar results, this process has to confirm to the following rule, ie., the similarity between a reconstructed symbol sequence and the query should be the same as that between its original one and the query. However, there may be cases in which some frames in a similar sequence are not similar to any frame in a query at all, and thus these frames cannot be identified. Therefore, it is impossible to achieve the completely same symbol sequences with the original ones by reconstruction.

Since the identified symbols have the same similarity with query as '$-$', by sequence reconstruction, we represent these unidentified frames as '$-$', and facilitate future similarity matching. For example, suppose a query video $Q :< f_1 f_2 f_3 >$ is mapped to $< 223 >$ ('2' and '3' are cluster *ids*), and also suppose, after query mapping by the index, two video sequences in the database are possibly similar to Q, S_1 represented as $< 11222 >$, and S_2 represented as $< 3344 >$. As no symbol in Q is similar to '1' and '4', S_1 and S_2 will be instead represented as $< - - 222 >$ and $< 33 - - >$ respectively.

Sequence Matching By the above steps, only possibly similar videos are obtained and reconstructed. To refine the results and decide the final query results, we adopt Probability-based Edit Distance, or PED, as a distance measure for sequence matching.

Edit distance is widely used in string matching. Though it is possible for us to use it directly for the similarity measure between video symbol sequences, better performance can be achieved by PED, which takes into consideration not only the temporal ordering inherent in video sequences, but also the inter-symbol similarity of two different symbols in symbol sequences.

For a video symbol sequence $S < s_1, s_2, ..., s_m >$, its PED to a query symbol video $Q < q_1, q_2, ..., q_n >$ is computed as follows:

$$PED(S,Q) = \begin{cases} 0 & m = n = 0 \\ \text{m} & m > 0 \; and \; n = 0 \\ \min\{PED(\text{S}^{\text{m}-1}, \text{Q}^{\text{n}-1}) + p, \\ \quad PED(\text{S}^{\text{m}}, \text{Q}^{\text{n}-1}+1), \\ \quad PED(\text{S}^{\text{m}-1}, \text{Q}^{\text{n}}+1)\} & otherwise \end{cases} \quad (1)$$

Here, p is decided by the probability distance d between s_i and q_j. We measure d by $\frac{|C_i - C_j|}{|C_i|}$, where C_i is the cluster which contains s_i, C_j is that covers q_j, and $|C_i|$ represents the number of video frames in C_i; and $|C_i - C_j|$ represents the number of frames which are in C_i but not covered by C_j. Note that this measurement of the similarity degree is probability-based. Given a T (called *probability threshold*), if d is less than T, these two symbols are similar, then $p = 0$; else, $p = 1$.

The whole similarity search process is shown in Figure 2

```
Procedure KNNSimilaritySearch.
input:  Q - query video sequence
        k - the number of similar videos
output: similar videos
1. {C_k} ←QuerySummarization(Q)
2. for each cluster C_k ∈ {C_k}
3.     Let the representative frame be f_ki
4.     ref← GetB+treeRefpoint
5.     dist=Dist(ref,f_ki)
6.     {minKey,maxKey}←MappingkeyRange
7.     {C_i}←SearchB+tree(minKey,maxKey)
8.     Mappingframe(f_ki)
9.     for each frame f_kj ∈ C_k (f_kj ≠ f_ki)
10.        for each cluster C_i ∈{C_i}
11.         if(|Dist(ref, f_ki)-Dist(ref, C_i)| < ε)
12.             dist=Dist(f_ki, C_i)
13.             Mapping to video segments
14. Filtering video segments
15.     IgnoreCommenSymbolSegments
16.     ThrowOffSmallSegments
17. Reconstruct candidate sequences
18. MatchingQueryAndCandidates
19. return results
```

Figure 2: : The similarity search algorithm.

3.4 Optimizations

Some optimizations are employed to reduce the information loss in video representation and improve the performance of similarity search.

3.4.1 Multi-Symbol Representation

To capture more information from original high dimensional space and improve the query result of similarity search, we proposed multi-symbol representation for query video. By this representation, each frame in a video is mapped to a set of clusters that cover it. Thus, each frame corresponds to a set of symbols. And then, the whole sequence is transformed into a symbol set sequence, denoted as $(\text{S}_1^Q, S_2^Q, ...S_n^Q)$.

For all the clusters containing a certain frame, each of them is similar with the symbolized frame to different extent. To obtain the optimal performance, the symbols in a certain symbol set is accessed orderly. Here, they are sorted by the distances between this frame and its cluster centers in the original space. Among them, the cluster having the smallest Euclidean distance to the frame is also called *principal cluster*. Accordingly, the id of the principal cluster is the *principal symbol*. The inter-symbol similarity is decided by the the probability distance between the principal symbols.

Given two sequences and the inter-frame similarity threshold, ϵ, in order to judge the inter-frame similarity of two sequences in the original high dimensional space, a multi-symbol match method is employed to compare a specific symbol in the symbol sequence data with a symbol set of the query sequence. This symbol will be compared with each in symbol set orderly. If two symbols are same or completely dissimilar, the comparison process will be stopped. Otherwise, the comparison operation continues over the next one in its symbol set, until the completely similar or dissimilar symbol is obtained or all symbols in the symbol set is visited. If all the symbols in symbol set are overlapped with one in the symbol sequence data, the probability distance is applied.

This multi-symbol match is based on the following theory.

Theorem 1. *Given the symbol set of a frame f, namely $s =< c_1, c_2, ...c_n >$, and a symbol c', if c_m is similar with c', then it is impossible for c' to be dissimilar with another symbol c_n in the symbol set, where c_m and c_n are two different symbols in the symbol set.*

Proof. Since c_m and c_n are different symbols of f, f falls in the intersection part of them.

Suppose c_m be completely similar with c', all the relations of symbol set are shown in Figure 3, we have f similar with c'. Then

$$d(f, c') \leq \frac{\epsilon}{2}$$

Suppose on the contrary that c_n and c' are dissimilar completely. according to the rule of dissimilarity between symbols, any frame f' in the cluster of c_n, will have a distance more than $\frac{\epsilon}{2}$+ r' from c'. Since frame f is in the cluster of c_n, we have

$$d(f, c') > \frac{\epsilon}{2} + r'$$

These two assumptions contradict each other, thus would not exist in the meanwhile. □

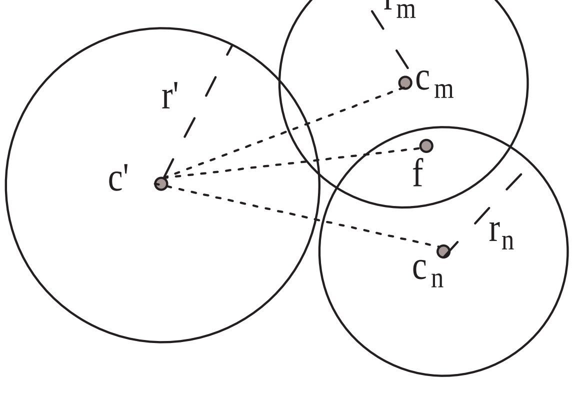


Figure 3: Similarity between symbol and symbol set

Figure 4 summarize the algorithm for the similarity measure between a symbol and a symbol set.

```
Matching Video Symbol and Symbol Set.
1. Given video symbol c' and symbol set s, here
   s=< $c_1, c_2, ...c_n$ >
2. for(All symbols in the symbol set s, $c_i$ )
3   If( $c_i$ and c' are (dis)similar completely)
4         GetInterSymbolSimilarity;
5         StopCheckingSymbolSet;
6   else
7         CheckNextSymbolInSymbolSet;
8   if(Not found completely (dis)similar pair)
9         MeasurePDistToPrincipleSymbol;
```

Figure 4: : Similarity Measure between Symbol and Symbol Set.

3.4.2 Batch Query Mapping

For mapping a query video to the corresponding symbols and further retrieving the potential similar segments, the naive method is to map each video frame one by one. As a query sequence may also consist of a large number of frames, naive method is costly. In order to improve the query mapping and the similarity search, based on observation 1, we proposed batch query mapping that performs a batch of individual frame mappings by the index.

Observation 1. *In a video, there exists a lot of very similar frames, which share the same query space in the process of mapping, thus accessing the same disk blocks. Batch processing for them can avoid the unnecessary IO access.*

We propose a query summarization method based on *local clustering*, which perform clustering over query frames by their inter-frame similarity. Since nearby frames in a video are usually quite similar. By doing so, a query sequence is formed into a small number of query clusters which contain similar query frames, where the inter-frame distance in the same cluster is no more than the similarity threshold, ϵ.

Due to the small number of the query clusters, the number of disk re-accesses is reduced greatly. For each frame cluster, with radius r_1, the cluster center is chosen as a representative. The symbolization for the frames in the whole cluster is completed by finding the first symbol, with radius R_1, of this representative, and then obtaining all neighboring cluster spaces with distances no more than R_1+r_1 from the first cluster center. Meanwhile, the distance between each cluster in query space and the representative is used to perform the filtering based on triangular inequality, reducing the CPU cost of mapping. Consequently, both IO and CPU cost for query mapping are reduced greatly, thus enhancing the performance of similarity search.

4 Performance Study

In this section, we present an empirical study to evaluate the proposed approach over large real video data sets. The evaluation is based on two aspects: (1) the effectiveness of video symbolization; (2) the efficiency of video indexing.

4.1 Setup

The experiments are conducted over two real datasets from 6000 15s video clips and 896 10s clips , which are recorded by using Virtual Dub at PAL frame rate of 25fps (Shen et al 2005). Each frame is represented as a 64-d vector in the RGB color space. In the same

Para	Description
ϵ	Clustering threshold
T	Probability threshold
K	Number of most similar sequences

Table 1: Parameters used in the experiments.

set of tests, the lengthes of video sequences used are similar.

For the effectiveness, we evaluate symbolization representation by comparing with key frames approach, where key frames are obtained by shot boundary detection method (Nagasaka & Tanaka 1992), to test the information loss. As the original data has no information loss, we examine the effectiveness by employing the video symbolization and the original videos and see how close the results generated by the PED over symbol sequences are to those by the edit distance (ED) in the original space. ED in original space is different from PED in how to decide the p. If the Euclidean distance between two frames is more than ϵ, p=1; otherwise, p=0. The set of query results by ED in the original space is denoted as *rel*, and that from summarization method is *ret*, the Accuracy of matching is defined as:

$$Accuracy = \frac{|rel \bigcap ret|}{|rel|} \qquad (2)$$

For the efficiency, we compare our proposed two-layer indexing method with optimal reference based B^+-tree, the high performance of which has been proved in (Shen et al 2005), and sequential scan as a base. By varying the size of video dataset, the efficiency of two-layer video indexing and ViTris indexing are compared. We measure the search efficiency in terms of IO and CPU cost. IO cost is evaluated by the number of page accesses, and the size of each page is set to be 4k. For CPU, we use the number of distance calculations, since it dominates the time of CPU cost and is also objective. All the experiments were performed on a Sun Enterprise E420(4*450MHz CPU's with 4GB RAM). Table 1 summarizes the parameters used in the experiments.

4.2 Effectiveness of Symbolization

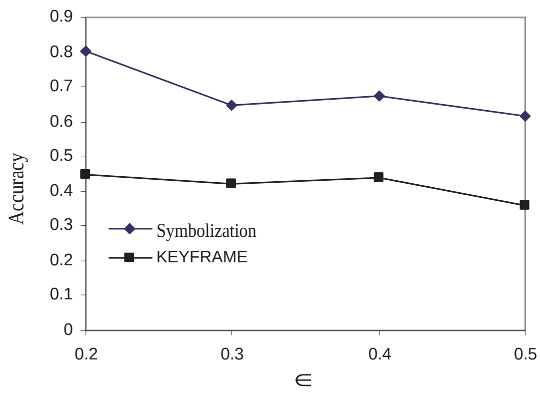


Figure 5: Accuracy By ϵ Varying in KNN Search on 15s Clips

To examine the effectiveness of symbolization, we need to estimate the value of optimal probability threshold of similar symbols, T, in probability measure. According to our analysis, T=0.5 is the optimal symbol similarity threshold.

We evaluate our representation approach by turning the ϵ value, for testing the information loss of it with clustering threshold. Figure 5 shows the test results. We fixed T to 0.5, and changed ϵ from 0.2 to 0.5. Obviously, the accuracy of our method is much better than that of key frames representation. Meanwhile, with the increasing of ϵ, the effectiveness tends to degrade for both methods, because of the more information lose for them. For the symbol sequence similarity search, since the overlap extent is small for small ϵ, inter-symbol similarity is very close to inter-frame similarity, leading to high effectiveness of it. With the ϵ increasing, due to the high overlapping, the stability of inter-symbol comparison degrades accordingly, causing the degradation of effectiveness.

4.3 Efficiency of Batch Mapping

During query mapping, a query video which consists of large numbers of frames may need to be summarized first to reduce the cost. In this set of experiments, we compare the performance of batch mapping with that of the naive mapping approach.

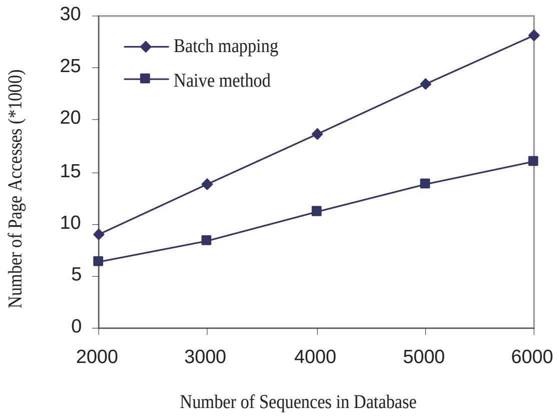


Figure 6: IO in Batch Mapping

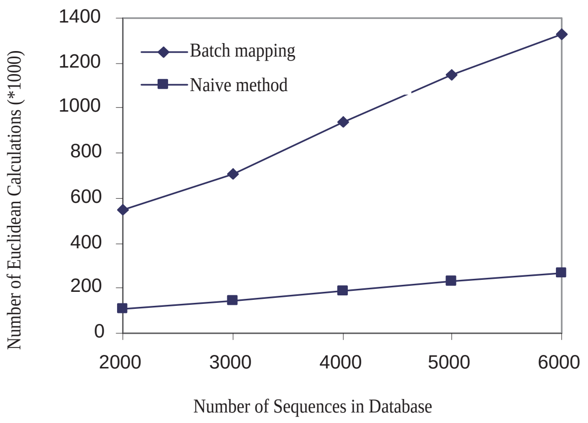


Figure 7: CPU in Batch Mapping

Figure 6 and 7 show the IO cost and CPU cost respectively. As shown in the figures, the batch mapping improves the performance a lot, and both IO cost and CPU cost are reduced greatly. For naive approach, each query frame needs to be mapped to the index space one by one. Also, the same index space may be accessed repeatedly as nearby frames in a query video are usually similar and share the same index space, thus incurring much more IO and CPU costs. When the batch mapping is used, similar frames are clustered together, thus query mapping has less IO costs. Meanwhile, in the batch mapping, the filtering based on triangular inequality is performed, accordingly, the CPU cost of batch mapping is also reduced greatly.

4.4 Efficiency of Two-layer Video Indexing

In this part, we performed experiments to test the two-layer index method by comparing with ViTris indexing and the sequential scan as a base. The IO cost and CPU cost are shown in the Figures 8 and 9.

Obviously, compared with ViTris indexing and sequential scan, the filtering ability of two-layer index has been improved greatly. Since two-layer index gives up a large number of candidates, in which only very few similar query symbols appear or only common symbols which are not discriminative, the number of IO and that of probability edit distance calculations are reduced noticeably, up to half. At the same time, it is clear that the filtering ability is more efficient for small K, with the increasing of K, becoming a bit weaker, but still keep high efficiency.

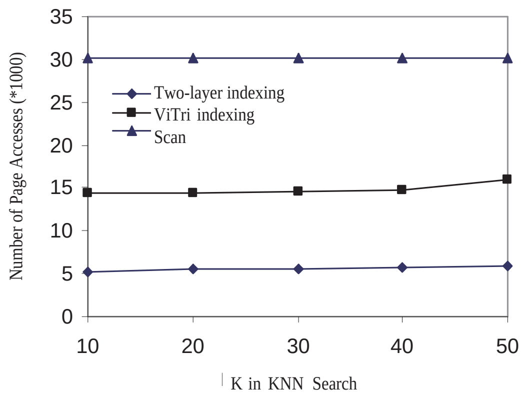


Figure 8: Effect of K in KNN Search

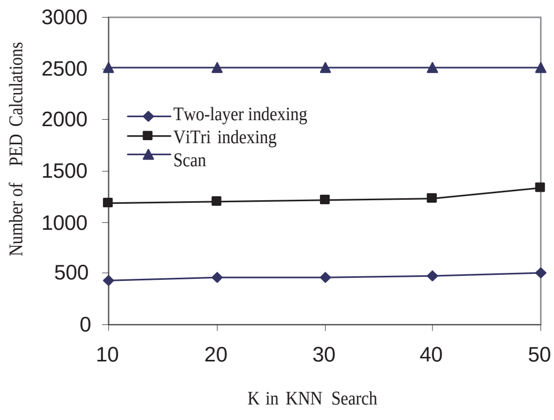


Figure 9: Effect of K in KNN Search

4.5 Effect of Dataset Size

We compare the efficiency of our two-layer video indexing with ViTris indexing in case of varying the number of 15s video clips varying from 2500 to 5000. Figure 10 and 11 show the IO cost and the number of frame comparisons in two-layer video indexing by comparing with that of B$^+$-tree based ViTri indexing.

We can see that, with the increasing of the data size, the IO and CPU costs increase for both of them. For another, obviously, the two-layer video index needs far less IO and frames comparison than ViTri indexing, since two-layer video index performs high dimensional distance calculations only in the process of frame mapping, while not in similarity measure. Moreover, query summarization based batch mapping can filter out most of the unconcerned symbols, thus reducing the cost of two-layer video indexing greatly. Comparing with B$^+$-tree based ViTri indexing, for the same video dataset, the cost of two-layer video indexing is improved one order of magnitude.

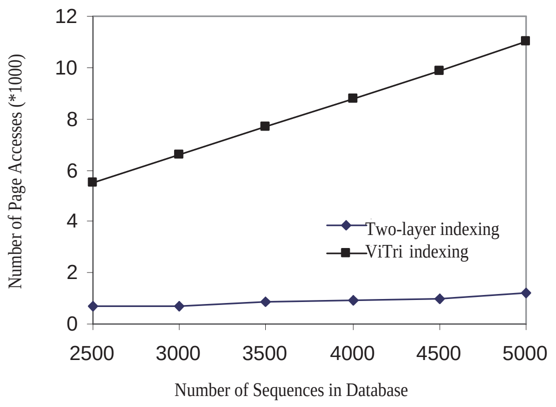


Figure 10: Efficiency by varying dataset size

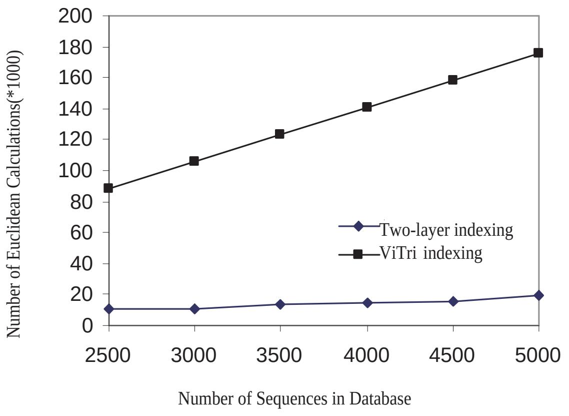


Figure 11: Efficiency by varying dataset size

5 Conclusions

In this paper, we have proposed a summarization based video similarity search approach, which has the following features: by video symbolization, each video sequence is symbolized as a digital string, which facilitates similarity matching with consideration of temporal ordering; by decomposition and reconstruction based video indexing strategy, video data are indexed and similar videos can be efficiently retrieved with non-metric similarity measure. Further, optimizations have been employed to further improve the search performance. We have done extensive performance study and our results have shown that our approach is very efficient, while keeping high search accuracy.

6 Acknowledgments

This research was supported by ARC grant DP0663272 and an Endeavour IPRS.

References

Adjeroh, D. A., Lee, M. C. & King, I. (1999), A distance measure for video sequences, *in* 'Computer Vision and Image Understanding', Vol. 75, pp. 25–45.

Bohm,C., Berchtold,S. & Keim, D. (2001), Searching in High-dimensional Spaces: Index Structures for Improving the Performance of Multimedia Databases,*in* 'ACM Computing Surveys', Vol. 33, pp. 322-373.

Chang, H. S., Sull, S. & Lee, S. U. (1999), Efficient video indexing scheme for content-based retrieval, *in* 'IEEE Transactions on Circuits and Systems for Video Technology', Vol. 9, pp. 1269-1279.

Cheung, S-C. S. & Zakhor, A. (2003), Efficient video similarity measurement with video signature, *in* 'IEEE Transactions on Circuits and Systems for Video Technology', vol. 13,2003, pp. 59-74.

Guttman, A. (1984), R-Trees: A Dynamic Index Structure for Spatial Searching, *in* 'Proceedings of the ACM SIGMOD International Conference on Management of Data', ACM Press, Boston, Massachusetts, USA, pp. 47-57.

Jagadish, H. V., Ooi, B. C., Tan, K-L., Yu, C. & Zhang, R. (2005), iDistance: An adaptive B+-tree based indexing method for nearest neighbor search, *in* 'ACM Transactions on Data Base Systems (TODS)', Vol. 30 ACM Press, New York, NY, USA, pp. 364–397.

Kim, S. H. & Park, R-H (2002), An efficient algorithm for video sequence matching using the modified Hausdorff distance and the directed divergence, *in* 'IEEE Transactions on Circuits and Systems for Video Technology', Vol. 12 , pp. 592-596.

Lee, S-L., Chun, S-J., Kim, D-H., Lee, J-H.& Chung, C-W. (2000), Similarity Search for Multidimensional Data Sequences, *in* 'Proceedings of the International Conference on Data Engineering', Vol. 12 , pp. 599-608.

Nagasaka, A. & Tanaka, Y. (1992), Automatic Video Indexing and Full-Video Search for Object Appearances, *in* 'Proceedings of the IFIP TC2/WG 2.6 Second Working Conference on Visual Database Systems II', Vol. 12 ,North-Holland, pp. 113–127.

Shen, H. T., Ooi, B. C. & Zhou, X. (2005), Towards Effective Indexing for Very Large Video Sequence Database, *in* 'Proceedings of the ACM SIGMOD International Conference on Management of Data', pp. 730-741.

Zhu, X., Wu, X., Fan, J., Elmagarmid, A. K. & Aref, W. G. (2004), Exploring video content structure for hierarchical summarization, *in* 'Multimedia System', Vol. 10, pp. 98-115.

Author Index

Recent Volumes in the CRPIT Series