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:

$$t = b \mid (a_1 : t_1, \ldots, a_n : t_n) \mid \{t\},$$

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 $comp(E) = \{r_1 : E_1, \ldots, r_n : E_n\}$ 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 $attr(E) = \{a_1, \ldots, a_m\}$ 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 $E = (\{r_1 : E_1, \ldots, r_n : E_n\}, \{A_1, \ldots, A_k\}, \{r_{i_1} : E_{i_1}, \ldots, r_{i_m} : E_{i_m}, A_{j_1}, \ldots, A_{j_\ell}\}) \in \mathcal{S}$:

- The representation type of E is the tuple type $t(E) = (r_1 : t(E_1), \ldots, r_n : t(E_n), A_1 : dom(A_1), \ldots, A_k : dom(A_k))$.
- The key type of E is the tuple type $k(E) = (r_{i_1} : k(E_{i_1}), \ldots, r_{i_m} : k(E_{i_m}), A_{j_1} : dom(A_{j_1}), \ldots, A_{j_\ell} : dom(A_{j_\ell}))$.

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 $(i = 1, \ldots, n)$ 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, $db(E_i) = \texttt{filter}(\varphi_i)db(E)$. For derived fragmentation, $db(E_i) = db(E) \ltimes db(E'_i)$ with primary fragmentation schema of E', i.e., $F_{E'} = \{E'_1, \ldots, E'_n\}$. 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 $\varphi_1 \equiv \text{level} \geq 300$ and $\varphi_2 \equiv \text{level} < 300$.
Fragment the database type LECTURE by semi-join with the fragments of PAPER we get:

$$\text{ADVANCED\_LECTURE} = \text{LECTURE} \ltimes \text{ADVANCED\_PAPER}$$

$$\text{BASIC\_LECTURE} = \text{LECTURE} \ltimes \text{BASIC\_PAPER}$$

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 $E_1 \cup \cdots \cup E_n$. Another round of query optimisation might shift the selection $\texttt{filter}(\varphi)$ and the projection $\texttt{map}(\pi_X)$ 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.).

$$\texttt{map}(\pi_X)(\texttt{filter}(\varphi)(db(E))) \qquad (*)$$

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 $\{E_1, \ldots, E_n\}$ of average sizes $s_1, \ldots, s_n$. If the network has a set of nodes $N = N_1, \ldots, N_k$ we have to allocate these fragments to one of the nodes, which gives rise to a mapping $\lambda : \{1, \ldots, n\} \to \{1, \ldots, k\}$, 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 $Q^m = \{Q_1, \ldots, Q_m\}$. 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 $(i = 1, \ldots, k)$.

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 $(i, j \in \{1, \ldots, k\})$.

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.

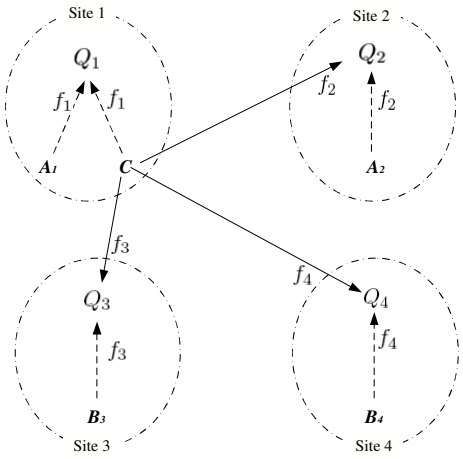

Figure 1: Scenario I - Primary Fragmentation Only

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:

$$cost_{\lambda_2}(Q) = s_{C_1} \cdot f_3 + s_{C_2} \cdot f_3 + s_{C_1} \cdot f_4 + s_{C_2} \cdot f_4$$

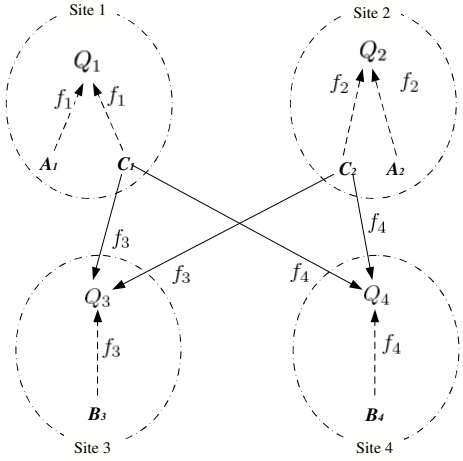

Figure 2: Scenario II - Primary and Derived Fragmentation on One Fragmentation Scheme

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:

$$cost_{\lambda_3}(Q) = s_{C_{2a3b}} \cdot f_2 + s_{C_{1a}} \cdot f_3 + s_{C_{1a}} \cdot f_4 + s_{C_{2a4b}} \cdot f_4$$

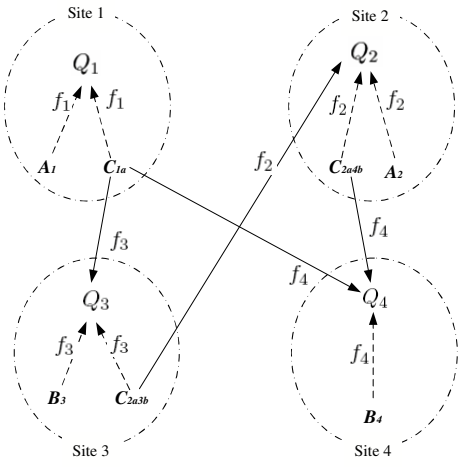

Figure 3: Scenario III - Primary and Derived Fragmentation on Two Fragmentation Schemata

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, $F_C = \{C_1, \ldots, C_m\}$ and $F'_C = \{C'_1, \ldots, C'_n\}$, 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: $Q^m = \{Q_1, \ldots, Q_m\}$ /* a set of global queries
E_d /* a type with a set of components and attributes
a set of network nodes $N = \{1, \ldots, k\}$
a set of fragmentation schemata resulting from primarily fragmentation
$E_i = E_i^1 \cup \cdots \cup E_i^k$
Output: derived horizontal fragmentation schema and fragment allocation schema
Begin
`for` each $h \in \{1, \ldots, k\}$ `let` $E_d^h = \emptyset$ `endfor`
`for each` related type $E_i \in \{E_1, \ldots, E_c\}$ `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, $\text{LECTURER}_{1D1T}, \text{LECTURER}_{2D2T}, \text{LECTURE-R}_{3D3T}$ 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.