2015

On transformation of query scheduling strategies in distributed and heterogeneous database systems

Janusz R. Getta
University of Wollongong, jrg@uow.edu.au

- Handoko
University of Wollongong, h629@uowmail.edu.au

Publication Details

Getta, J. R. & Handoko, (2015). On transformation of query scheduling strategies in distributed and heterogeneous database systems. Lecture Notes in Computer Science, 9011 139-148.

On transformation of query scheduling strategies in distributed and heterogeneous database systems

Abstract
This work considers a problem of optimal query processing in heterogeneous and distributed database systems. A global query sub- mitted at a local site is decomposed into a number of queries processed at the remote sites. The partial results returned by the queries are in- tegrated at a local site. The paper addresses a problem of an optimal scheduling of queries that minimizes time spend on data integration of the partial results into the final answer. A global data model defined in this work provides a unified view of the heterogeneous data structures located at the remote sites and a system of operations is defined to ex- press the complex data integration procedures. This work shows that the transformations of an entirely simultaneous query processing strate- gies into a hybrid (simultaneous/sequential) strategy may in some cases lead to significantly faster data integration. We show how to detect such cases, what conditions must be satisfied to transform the schedules, and how to transform the schedules into the more efficient ones.

Keywords
query, transformation, heterogeneous, distributed, systems, strategies, database, scheduling

Publication Details
Getta, J. R. & Handoko, (2015). On transformation of query scheduling strategies in distributed and heterogeneous database systems. Lecture Notes in Computer Science, 9011 139-148.

On Transformation of Query Scheduling Strategies in Distributed and Heterogeneous Database Systems

Janusz R. Getta[1] and Handoko[1]

School of Computer Science and Software Engineering,
University of Wollongong, Australia
{jrg,h629}@uow.edu.au

Abstract. This work considers a problem of optimal query processing in heterogeneous and distributed database systems. A global query submitted at a local site is decomposed into a number of queries processed at the remote sites. The partial results returned by the queries are integrated at a local site. The paper addresses a problem of an optimal scheduling of queries that minimizes time spend on data integration of the partial results into the final answer. A global data model defined in this work provides a unified view of the heterogeneous data structures located at the remote sites and a system of operations is defined to express the complex data integration procedures. This work shows that the transformations of an entirely simultaneous query processing strategies into a hybrid (simultaneous/sequential) strategy may in some cases lead to significantly faster data integration. We show how to detect such cases, what conditions must be satisfied to transform the schedules, and how to transform the schedules into the more efficient ones.

Keywords: distributed heterogenous database systems, data integration, optimization of query processing

1 Introduction

Efficient data processing in the distributed and heterogeneous database systems is a critical factor for the successful implementations of global information systems. Performance of distributed applications strongly depends on the efficient algorithms that organize data processing in the distributed and heterogenous database systems. For instance, in the *MapReduce* programming model a user application that accesses data distributed over a number of remote sites simultaneously submits all of its sub-tasks to the remotes sites and later on integrates the partial results at a central site [4]. The simultaneous processing of sub-tasks makes *MapReduce* an efficient strategy when the amounts of processing and the amounts of data transmitted from the remote sites are more or less the same for all its sub-tasks. Unfortunately, a simultaneous processing strategy does not

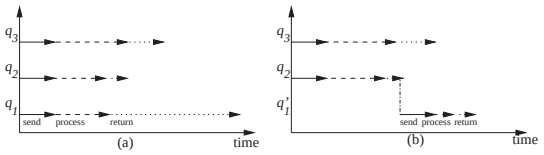

Fig. 1. Simultaneous (a) versus sequential (b) processing of tasks q_1 and q_2

provide the best performance when one of the sub-tasks returns significantly larger amounts of data and/or when transmission speed is significantly lower than for the other sub-tasks. For example, consider the time diagrams in Fig. 1 when a global query has been decomposed into the queries q_1, q_2, and q_3 simultaneously processed at the remote sites. In the first case (a) data transmission of the results of q_1 dominates the total processing time. However, if some of data obtained from the processing of q_2 can be used to modify q_1 into q_1' such that more processing can be done at a remote site then transmission of the results of q_1' may take less time despite that processing of q_1' follows processing of q_2, see case (b).

A partial order in which the individual queries are processed at the remote sites is called as a *query scheduling strategy*. In an entirely *sequential strategy* processing of a query at a remote site precedes processing of another query and the result of the first query can be used to modify the succeeding queries. In an entirely *simultaneous strategy* all queries are simultaneously submitted and processed at the remote sites. The efficiency of both strategies depends on the computational complexities of the individual tasks, computational power at a central and at the remote sites, amount of data transmitted over the networks, and data transmission speed of the networks used. Intuitively, a simultaneous strategy seems to be more efficient when the majority of query processing can be done at the remote sites and the amounts of data transmitted to a central site are small. A sequential strategy is more efficient when one or more tasks transmit the large amounts of data to a central site and it is possible to use the results of the other tasks to reduce the amounts of data to be transmitted later on. As usual the best solution is a hybrid one when some of the tasks are processed sequentially while the others simultaneously. Additional factors that significantly complicate data processing in distributed systems are a high level of autonomy and heterogeneity of the remote sites. The administrators of remote sites are usually very strict about performance and security of the managed systems and because of that they restrict external access to a read only mode without the rights to create and use the local data containers. It simply means that a central site cannot send a container with data to a remote site such that the container can be used for data processing there. Heterogeneity of remote sites means that organization of data, software and hardware used each at each site are different, which further limits any possible cooperation.

In this work we consider an environment of a heterogeneous and distributed database system where a user application issued at a *central site* accesses data

at the remote sites. Then, it brings the partial results from the remote sites to a central site to integrate it into the final outcomes. In this work, we do not impose any restrictions on the compatibility of structures and contents of data containers at a central and the remote sites and we do not impose any assumptions about any level of "cooperation" between the sites. The only assumption is that the remote sites "display" a unified view of their data containers to the external user applications and are able to process the queries over the unified view. A global query issued at a central site is transformed into a set of queries $q_1, \ldots, q_n$ such that that each one of the queries accesses data from only one remote site. An expression $e(q_1, \ldots, q_n)$ integrates the partial results returned from the remote sites. Objective of this work is to find the formal backgrounds for the algorithms that schedule processing of the queries $q_1, \ldots, q_n$ at the remote sites and minimize the total processing time of $e(q_1, \ldots, q_n)$. In particular, we attempt to answers the questions when a sequential strategy is possible, when it is more efficient than a simultaneous strategy, what transformations of the queries must be applied to find a sequential strategy, what hybrid (simultaneous/sequential) strategies are possible for a given global query, and how to evaluate hybrid strategies.

The paper is organized in the following way. The next section overviews the previous works related to data processing in distributed systems. Section 3 present a model of query processing in a distributed system. and section 4 defines a global data model. The transformations of data processing strategies are presented in section 5 and evaluation of the strategies is explained in section 6. Section 7 concludes the paper.

2 Previous work

Optimization of data processing in distributed systems has its roots in optimization of query processing in multidatabase and federated database systems [12]. One of the recent solutions to speed up distributed query processing in distributed systems considers the contents of cache in the remote systems and prediction of cache contents [11]. Wireless networks and mobile devices triggered research in mobile data services and in particular in location-dependent queries that amalgamate the features of both distributed and mobile systems [7]. An adaptive distributed query processing architecture is introduced at [14] where fluctuations in selectivity of operations, transmission speeds, and workloads of remote systems affect an order of distributed query processing.

In [15] the query sampling methods is used to estimated the query processing costs at the local systems. Query scheduling strategy in a grid-enabled distributed database proposed in [3] takes under the consideration so called "site reputation" for ranking response time of the remote systems. A new approach to estimation of workload completion time based on sampling the query interactions has been proposed in [1] and in [2]. Query monitoring can be used to collect information about expected database load, resource allocation, and expected size of the results [10].

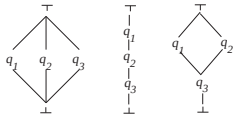

Fig. 2. The sample partial orders of processing the subqueries q_1, q_2, q_3

The reviews of research on query scheduling and data integration are included in [8], [16]. The implementations of experimental data integration systems based on application of ontologies and data sharing are described in [13] and [6].

3 Query processing in distributed systems

We consider a distributed and heterogeneous data base system where the data containers in the various formats like for example, relational, XML, object-relational, key-value, etc., are distributed over a number of highly autonomous remote sites. Each site "publishes" to all other sites a global view of data located at a site. A user application originated at a central site accesses data at the remote sites through a global query like $q(s_1{:}d_i, \ \ldots, \ s_n{:}d_n)$ where d_i is a data container located at a remote site s_i. A global query is decomposed into k queries $q_1, \ldots, q_k$ such that each query is processed at only one remote site and it is transformed into a *data integration expression* $e(q_1, \ldots, q_k)$. Let $Q = \{\top, \bot, q_1, \ldots, q_k\}$ be a set where $\top$ is a *start of processing* symbol and $\bot$ is an *end of processing* symbol. We define a partial order $P \subseteq Q \times Q$ such that $\langle q_i, q_j \rangle \in P$ if a query q_i is processed before a query q_j. Then, $< Q, P >$ is a lattice where $sup(P) = \top$ and $inf(P) = \bot$ that represents a partial order of processing the queries $q_1, \ldots, q_k$ at the remote sites.

For instance, the lattices given in a Fig. 2 represent an entirely simultaneous strategy, entirely sequential strategy, and hybrid strategy where the queries q_1 and q_2 are processed simultaneously before q_3.

4 Global data model

A *global data model* provides a unified view of data stored at the remote sites. It amalgamates the contradictory requirements of generality with the very precise specifications of the basic operations.

A *data object* is defined as a pair $\langle id, t \rangle$ where id is a unique object identifier at a given remote site and t is a *description* of the object. A *description* is defined as a mapping $t : S \rightarrow dom(A)$ where S is a set of *access paths* to the values of attributes, e.g. *address.street.house.flat*. Let $p.a$ be a path to a value of an attribute a. Then, a mapping t satisfies a condition $t(p.a) \in dom(a)$ where $dom(a)$ denotes a domain of attribute a. A set of all access paths S in a description of an object is called as a *schema of an object* and $dom(A) = \bigcup_{a \in A} dom(a)$. A *data container* d is a set of data objects. A *schema of a data container* is a union of all schemas of all objects included in the container.

A set of operations on data containers includes the unary operations of *selection* and app *extraction*, and binary operations of *union*, *composition*, *semi-* and *anti-composition*, and *substitution*.

Let d be a data container. An *access term* is defined as a triple *d.p.a*. A *selection condition* ϕ is defined as a well-formed formula of prepositional calculus built from the access terms, relational operators, Boolean operators (*and*, *or*, *not*), constants, and brackets. Additionally, all access terms in a *selection condition* must start from a name of the same data container.

Let d_i and d_j be the data containers. A unary *selection* operation σ on an argument d_i is defined as $\sigma_\phi(d_i) = \{\langle id, t\rangle : \exists\langle id_i, t\rangle \in d_i \; and \; eval(\phi, \langle id_i, t\rangle)\}$ where a function *eval* evaluates a selection condition ϕ against the contents of a data object $\langle id_i, t\rangle$ into *true* or $false$. Note, that in a result of *selection* each object in a result of selection obtains a new identifier.

Let S_i be a schema of a data container d_i and let $S \subseteq S_i$. A unary *projection* operation π of a data container d_i on a schema S is defined as $\pi_S(d_i) = \{\langle id, t\rangle$: $\exists\langle id_i, t_i\rangle \in d_i \; t = t_i[S]\}$ where $t_i[S]$ means restriction of a description t_i to the access paths in S.

A binary *union* operation $\cup$ on the arguments d_i and d_j is defined as $d_i \cup d_j = \{\langle id, t\rangle : \exists\langle id_i, t\rangle \in d_i \; or \; \langle id_i, t\rangle \in d_j\}$.

A *constructor* operation θ is defined as $\theta : d_i \times d_j \rightarrow d_{ij}$ such that $\theta(\langle id_i, t_i\rangle$, $\langle id_j, t_j\rangle) = \langle id_{ij}, t_{ij}\rangle$ where id_{ij} is an identifier of a new object and $t_{ij} = f(t_i, t_j)$ where f is an expression that combines the descriptions t_i and t_j into a description t_{ij} of a new object.

A *matching condition* ψ is defined as a well-formed formula of prepositional calculus built from the access terms, relational operators, Boolean operators (*and*, *or*, *not*), constants, and brackets. Additionally, a *matching condition* consists only of the comparisons between the access terms that related to the different data containers.

A binary *composition* operation $\otimes_{\psi\theta}$ on the arguments d_i and d_j is defined as $d_i \otimes_{\psi\theta} d_j = \{\langle id_{ij}, t_{ij}\rangle : \exists\langle id_i, t_i\rangle \in d_i \; and \; \exists\langle id_j, t_j\rangle \in d_j \; eval(\psi, \langle id_i, t_i\rangle, \langle id_j, t_j\rangle)$ $and \; \langle id_{ij}, t_{ij}\rangle = \theta(\langle id_i, t_i\rangle, \langle id_j, t_j\rangle)\}$.

We say that operation $d_i \otimes_{\psi\theta} d_j$ is *semi-reversible* if $\pi_{S_i}(d_i \otimes_{\psi\theta} d_j) \subseteq d_i$ and $\pi_{S_j}(d_i \otimes_{\psi\theta} d_j) \subseteq d_j$. In the rest of this paper we consider only the composition operations which are semi-reversible on the schemas of its both arguments.

A *semi-composition* operation $\oplus_\psi$ on the arguments d_i and d_j is defined as $d_i \oplus_\psi d_j = \{\langle id, t_i\rangle : \exists\langle id_i, t_i\rangle \in d_i \; and \; \exists\langle id_j, t_j\rangle \in d_j \; eval(\psi, \langle id_i, t_i\rangle, \langle id_j, t_j\rangle)\}$.

An *anti-composition* operation $\ominus_\psi$ on the arguments d_i and d_j is defined as $d_i \ominus_\psi d_j = \{\langle id, t_i\rangle : \exists\langle id_i, t_i\rangle \in d_i \; and \; \forall\langle id_j, t_j\rangle \in d_j \; not \; eval(\psi, \langle id_i, t_i\rangle, \langle id_j, t_j\rangle)\}$.

Consider a data container $d_j = \{\langle id_1, t_1\rangle, \ldots, \langle id_k, t_k\rangle\}$ and a matching condition $\psi(d_i.p_1.a_1, \ldots, d_i.p_m.a_m, d_j.p_1.b_1, \ldots, d_j.p_n.b_n)$. A *substitution* operator is denoted by $\psi \leftarrow d_j$ and it is defined as
$\psi(d_i.p_1.a_1, \ldots, d_i.p_m.a_m, d_j.p_1.b_1, \ldots, d_j.p_n.b_n) \leftarrow d_j =$
$\psi(d_i.p_1.a_1, \ldots, d_i.p_m.a_m, t_1(p_1.b_1), \ldots, t_1(p_n.b_n)) \; or \; \ldots or$
$\psi(d_i.p_1.a_1, \ldots, d_i.p_m.a_m, t_k(p_1.b_1), \ldots, t_k(p_n.b_n))$.

A *substitution* operator replaces all instances of access terms $d_j.s_1.b_1$, ..., $d_j.s_n.b_n$ in a matching condition ψ with the values of all attributes taken from all objects in a data container d_j and creates disjunction of all terms after the replacements. For example if $d_j = \{\langle id_1, t_1\rangle, \langle id_2, t_2\rangle\}$ and $t_1(s_1.name) =$ `James` and $t_2(s_1.name) =$ `Mary` then application of substitution operator $(d_i.s_1.name = d_j.s_1.name) \leftarrow d_j$ returns a matching formula $d_i.s_1.name =$ `James` *or* $d_i.s_1.name$ = `Mary`.

The following equations hold for any data containers d_i, d_j and any matching condition ψ.

$$d_i \oplus_\psi d_j = \sigma_{\psi \leftarrow d_j}(d_i) \tag{1}$$

$$d_i \ominus_\psi d_j = \sigma_{not(\psi \leftarrow d_j)}(d_i) \tag{2}$$

The equations listed above mean that the operations of *semi-* and *anti-composition* can always be replaced with a *filter* operation on the first argument while a *replacement* operation is applied to the second argument in a matching formula. If an expression $d_i \oplus_\psi d_j$ must be computed at a remote site that contains only a data container d_i and a data container d_j cannot be sent to the remote site then the computations of an expression with $\sigma_{\psi \leftarrow d_j}(d_i)$ replaces the computations of *anti-composition* at a remote site.

5 Transformations

We start from the simple transformations of simultaneous query scheduling strategies where two queries are simultaneously processed at the remote sites and their results are integrated with one of the arguments of *composition* operation. Next, we consider the complex transformations of the strategies where many queries are processed simultaneously and their results are integrated by an expression over many *composition* operations.

5.1 Simple transformations

We consider simple a data integration expression $q_i \otimes_{\psi\theta} q_j$ where q_i and q_j are the queries to be processed at two different remote sites. If we expect that transmission of the results of q_j will be significantly longer than transmission of the results of q_i then it is worth to change a simultaneous schedule into sequential where q_i is processed first and a part of it denoted by x will be involved in processing of q_j. To find x we rewrite a data integration expression into $q_i \otimes_{\psi\theta} (q_j \oplus_\psi x)$ where x is an unknown data container that must be sent to a remote site where q_j supposed to be processed.

We expect that a subexpression $(q_j \oplus_\psi x)$, when computed at a remote site, returns the results much smaller than the results of q_i. On the other hand, the results of the data integration expression must not change. Hence, to find x we solve an equation

$$q_i \otimes_{\psi\theta} q_j = q_i \otimes_{\psi\theta} (q_j \oplus_\psi x) \tag{3}$$

There exists many solutions of an equation (3) above, e.g. x equal to the results of q_j or any superset of the results of q_j satisfies the equation. We look for the smallest solution of the equation because we would like to minimize the amount of transmission to a remote site. An equation (3) can be transformed into an equivalent fixpoint equation and its fixpoint solution can be found using Kleene fix-point theorem [5].

$$x = x \cup \pi_{s_\psi}(q_i \otimes_{\psi\theta} q_j - q_i \otimes_{\psi\theta} (q_j \oplus_\psi x)) \cup (q_i \otimes_{\psi\theta} (q_j \oplus_\psi x) - q_i \otimes_{\psi\theta} q_j) \quad (4)$$

A projection π_{s_ψ} on a schema s_ψ of a matching condition ψ is necessary because a schema of a data container x does not need to include more attributes than it is used in a matching condition ψ. The solution of an equation (4) is obtained through the iterations starting from an empty data container $x_{(1)} = \emptyset$ and union of the results from each iteration. The smallest solution of the equation is equal to $x_{min} = \pi_{s_\psi}(q_i)$ which is consistent with our expectations. Then, the right hand side of equation (3) can be transformed into $q_i \otimes_{\psi\theta} (q_j \oplus_\psi \pi_{s_\psi}(q_i))$ and finally after application of equation (1) we obtain the final data integration expression $q_i \otimes_{\psi\theta} (\sigma_{\psi \leftarrow \pi_{s_\psi}(q_i)}(q_j))$. The expression is transformed into the following sequence of computations: $r_1 := q_i$; $r_2 := \sigma_{\psi \leftarrow \pi_{s_\psi}(r_1)}(q_j)$; $result := r_1 \otimes_{\psi\theta} r_2$.

If in the computations of data integration expression with *semi-composition* $q_i \oplus_\psi q_j$ the results of q_i are large and the results of q_j are small then it is possible to compute $r_1 := \pi_{s_\psi}(q_j)$ first and then apply an equation (1) to replace the composition with $result := \sigma_{\psi \leftarrow r_1}(q_i)$ computed at a remote site. In the opposite case we obtain $r_1 := \pi_{s_\psi}(q_i)$; $r_2 := \sigma_{\psi \leftarrow r_1}(\pi_{s_\psi}(q_j))$; $result := \sigma_{\psi \leftarrow r_2}(q_i)$.

If in the computations of *anti-composition* $q_i \ominus q_j$ the results of q_i are large and the results of q_j are small then it is possible to compute $r_1 := \pi_{s_\psi}(q_j)$ first and then apply an equation (2) to replace the composition with $result := \sigma_{not(\psi \leftarrow r_1)}(d_i)$ computed at a remote site. In the opposite case we obtain a plan: $r_1 := \pi_{s_\psi}(q_i)$; $r_2 := \sigma_{\psi \leftarrow r_1}(\pi_{s_\psi}(q_j))$; $result := \sigma_{not(\psi \leftarrow r_2)}(q_i)$.

5.2 Complex transformations

The simple transformations of two-argument data integration expressions described in the previous section can be systematically applied to find the complex transformations of n-argument data integration expressions. Consider a data integration expression $e(q_1, \ldots, q_n) = f(q_1, \ldots, q_k)\ \alpha_{\psi\theta}\ g(q_{k+1}, \ldots, q_n)$ where $\alpha \in \{\otimes, \oplus, \ominus\}$. Let $q_f = f(q_1, \ldots, q_k)$, $q_g = g(q_{k+1}, \ldots, q_n)$

Then, it is possible to determine whether a transformation from a simultaneous schedule to a sequential schedule is possible for q_f and g_g and if it is so, it is possible to find such transformation through systematic decomposition of the data integration expression into subexpression and find the transformations at each level of decomposition.

Without a significant loss of generality we consider an operation $\alpha_{\psi\theta}$ to be a composition $q_g \otimes_{\psi\theta} q_f$ and we transform a simultaneous schedule of processing q_g and q_f into a sequential one accordingly to the rules described earlier into $q_j := \pi_{S_\psi}(q_g)$ and $q_f := \sigma_{\psi \leftarrow q_j}(q_f)$. It means that in a sequential schedule an

entire expression $g(q_{k+1}, \ldots, q_n)$ must be computed before a modified expression $\sigma_{\psi \leftarrow q_j}(f(q_1, \ldots, q_k))$.

The further transformations depend on a distributivity of $f(q_1, \ldots, q_k)$ over an operation of selection and distributivity of $g(q_{k+1}, \ldots, q_n)$ over an operation of projection. If it is possible to transform $\pi_{S_\psi}(g(q_{k+1}, \ldots, q_n))$ into $q'_g \; \alpha_{\psi'\theta'} \; q''_g$ such that $q'_g = \pi_{S_\psi}(g'(q_{k+1}, \ldots, q_m))$ and $q''_g = \pi_{S_\psi}(g''(q_{m+1}, \ldots, q_n))$ then its is possible to transform again the computations of q'_g and q''_g from the simultaneous into the sequential ones.

In the same way if it is possible to transform $\sigma_{\psi \leftarrow q_g}(f(q_1, \ldots, q_k))$ into $q'_f \alpha_{\psi'\theta'} q''_f$ such that $q'_f = \sigma_{\psi \leftarrow q_g}(f'(q_1, \ldots, q_i)$ and $q''_f = \sigma_{\psi \leftarrow q_g}(f''(q_{i+1}, \ldots, q_k))$ then its is possible to transform again the computations of q'_g and q''_g from the simultaneous into the sequential ones.

A process described above is recursively applied to to each subexpression of data integration expression until the operations of selection and projection are directly applied to the arguments.

As a simple example consider a data integration expression $(q_1 \otimes_{\psi_1\theta} q_2) \oplus_{\psi_2} q_3$ where it is expected that the queries q_2 and q_3 return much smaller results than a query q_1. A simple transformation can be applied to change the processing of q_1 and q_2 from a simultaneous to a sequential one. It leads to a transformation of q_1 into $\sigma_{\psi_1 \leftarrow \pi_{S_{\psi_1}}(q_2)}(q_1)$. The second transformation can be obtained from the processing q_3 before q_1. An initial transformation $\sigma_{\psi_2 \leftarrow \pi_{S_{\psi_2}}(q_3)}(q_1 \otimes_{\psi_1\theta} q_2)$ applies to a result of composition of q_1 and q_2. Assuming, that in this case selection is distributive over composition we obtain the following second transformation $\sigma_{\psi_2 \leftarrow \pi_{S_{\psi_2}}(q_3)}(q_1)$. The outcomes of both transformation can be merged into a single expression $\sigma_{\psi_2 \leftarrow \pi_{S_{\psi_2}}(q_3) \; and \; \psi_2 \leftarrow \pi_{S_{\psi_2}}(q_2)}(q_1)$. It leads to a query scheduling strategy where q_2 and q_3 are simultaneously processed before q_1.

6 Evaluation of query scheduling strategies

If in the example above distributivity of selection over composition in $\sigma_{\psi_2 \leftarrow \pi_{S_{\psi_2}}(q_3)}$ $(q_1 \otimes_{\psi_1\theta} q_2)$ applies to both arguments of composition then is also possible to transform q_2 to $\sigma_{\psi_2 \leftarrow \pi_{S_{\psi_2}}(q_3)}(q_2)$. It means that it is possible to get more than one data integration plan where q_3 is processed before q_1 and q_3 is processed before q_2. To find an optimal processing plan we need information about the amounts of data to be transmitted over a network, transmission speed, amounts of time needed to process the queries at the remote sites and a cost function to calculate the total costs for each data integration plan represented by a lattice of queries. The total costs of processing a query q_i can be estimated as $t_i = t_{s_i} + t_{p_i} + t_{r_i}$ where t_{s_i} is time needed to send a query q_i to a remote site, t_{p_i} is time needed to process the query there, and t_{r_i} is time needed to transmit the results to a central site. Each one of the parameters depend on the information listed above. When the queries $q_1, \ldots, q_n$ are processed simultaneously then their total processing time is equal to $max(t_1, \ldots, t_n)$. If the queries are processed sequentially and the results of a query q_i are used to transform a query q_{i+1} the the total

processing time is equal to $t_1 + t'_2 + \ldots + t'_n$ where t'_i are the processing times of transformed queries. If a data integration strategy is represented by a lattice $< Q, P >$ then a cost formula is derived in the following way. Let p_i be a path from $\top$ to $\bot$ symbol in a lattice and passing through the nodes labeled with $q_{i_1}, \ldots q_{i_k}$. Then the costs of processing along a path p_i is equal to $t_{i_1} + \ldots + t_{i_k}$. The costs of processing along all paths $p_1, \ldots p_n$ from $\top$ to $\bot$ symbol in a lattice is equal to $max(t_{p_1}, \ldots, t_{p_n})$. Then, an equality $max(a+b, a+c) = a+max(b, c)$ can be use to simplify a cost formula. For example, a cost formula derived for a data integration schedule in Fig. (2, case 3) $max(t_1+t_3, t_2+t_3)$ can be simplified to $t_3 + max(t_1, t_2)$.

7 Summary and conclusions

This work is based on an observation that a significant difference between the amounts of data transmitted from the remote sites to a central site may have a negative impact on an overall time of data integration at a central site when a simultaneous query scheduling strategy is applied. Then, a transformation of a simultaneous strategy into a sequential or hybrid one speeds up data integration at a central site. This work shows when the transformations of query scheduling strategies are possible, how to perform it, when the transformations are beneficial, and how to evaluate the results.

Another interesting outcome of this work is a technique that embeds data into the queries through application of substitution operation. Substitution operation eliminates to some extent a problem of high level of autonomy of the remote sites that usually stop external users from transmitting data into the site and processing it there. A substitution operation allows for a safe processing of data obtained from another remote sites.

A system of operations proposed in this work allows for processing of any data containers as long as the access paths to the values of data items are provided and implemented by the owners of data. An interesting property of the system of operations is that it reduces to a standard relational algebra when the data containers include only homogeneous tuples or it reduces to XML algebra when the data containers include only XML documents, etc. For any structure of data objects included in data containers a system of operations needs the operations that select the objects that satisfy a given condition, operation that project the objects on a given sub-schema, union operation, operation that compares all pairs of objects and constructs new object from each pair, operation that compares objects from two containers and picks from one container the objects that match/do not match objects in the other container.

The transformation of query scheduling strategies mainly depend on the algebraic properties of a data integration expression and on the properties of composition operation. A composition operation must be semi-reversible such that it is possible to restore the subsets of the arguments from a result of an operation. A fixpoint equation (4) which is the basis for finding simple transformations is solvable when a function on its right hand side is monotonic. The complex trans-

formations are possible when the subexpressions of data integration expression are distributive over the operations of selection and projection.

References

1. Ahmad, M., Aboulnaga, A., Babu, S., Query interactions in database workloads.In: Proceedings of the Second International Workshop on Testing Database Systems, 1–6, (2009)
2. Ahmad, M., Duan, S., Aboulnaga, A., Babu, S., Predicting completion times of batch query workloads using interaction-aware models and simulation. In: Proceedings of the 14th International Conference on Extending Database Technology, 449–460, (2011)
3. Costa, R.L-C., Furtado, P., Runtime Estimations, Reputation and Elections for Top Performing Distributed Query Scheduling. In: Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, 28–35, (2009)
4. Dean, J., Ghemawat, S., MapReduce: Simplified Data Processing on Large Clusters. In: Proceedings of the 6th Symposium on Operating Systems Design and Implementation, (2004)
5. Granas, A., Dugundji, J., Fixed Point Theory. Springer-Verlag (2003)
6. Ives, Z.G., Green, T.J., Karvounarakis, G., Taylor, N.E., Tannen, V., Talukdar, P.P., Jacob, M., Pereira F., The ORCHESTRA Collaborative Data Sharing System. In: SIGMOD Record, (2008)
7. Ilarri, S., Mena, E., Illarramendi, A., Location-dependent query processing: Where we are and where we are heading. In: ACM Computing Surveys, vol. 42, no. 3, 1–73, (2010)
8. Lenzerini, M., Data Integration: A Theoretical Perspective. (2002)
9. Liu L., Pu, C., A Dynamic Query Scheduling Framework for Distributed and Evolving Information Systems. In: Proceedings of the 17th International Conference on Distributed Computing Systems, (1997)
10. Mishra, C., Koudas, N., The design of a query monitoring system. In: ACM Transactions on Database Systems, vol. 34, no. 1, 1–51, (2009)
11. Nam, B., Shin, M., Andrade H., Sussman, A.,Multiple query scheduling for distributed semantic caches. In: Journal of Parallel and Distributed Computing, vol.70, no. 5, 598–611, (2010)
12. Ozcan, F., Nural, S., Koksal, P., Evrendilek, C., Dogac, A.: Dynamic Query Optimization in Multidatabases. In: Bulletin of the Technical Committee on Data Engineering, vol. 20, no. 3, 38–45 (1997)
13. Thain, D. Tannenbaum, T., Livny, M., Distributed computing in practice: the Condor experience: Research Articles. In: Concurrency Computing: Practice and Experience. vol. 17, no. 2-4, 323–356, (2005)
14. Zhou, Y., Ooi, B.C., Tan, K-L., Tok, W. H., An adaptable distributed query processing architecture. In: Data and Knowledge Engineering,vol. 53, no. 3, 283–309, (2005)
15. Zhu, Q., Larson, P.A.: Solving Local Cost Estimation Problem for Global Query Optimization in Multidatabase Systems. Distributed and Parallel Databases, vol. 6, no. 4, 373–420 (1998)
16. Ziegler, P., Three Decades of Data Integration - All problems Solved ? In: 18th IFIP World Computer Congress, vol. 12, (2004)