Joint Conference of 1996 International Computer Symposium
December 19~21, Kaohsiung, Taiwan, R.O.C.

On Efficient Execution of Aggregate Functions for Resolving Data Inconsistency in Multidatabases *

Yungho Leu, Chunli Yang and Shyueming Tang

Department of Information Management
National Taiwan Institute of Technology
Taipei, Taiwan
yhl@cs.ntit.edu.tw

Abstract

In this paper, we address the issue of efficient execution of aggregate functions in resolving data inconsistency in multidatabases. We proposed two methods to optimize the execution of aggregate functions in the context of resolving data inconsistency. The first method is to localize the execution of selections by the help of key value transfer. Due to the reduction of the qualified tuples in the local databases, the communication overhead and the postprocessing at the query site are minimized. The second method is to maintain the maximal and minimal values of related attributes. This information helps the localization of the selection of some aggregate functions. The effectiveness of these methods is carefully studied and is measured in terms of the ratio of reduced data.

Keywords: *Data Inconsistency, Aggregate Functions , Query Optimization*

1 Introduction

In a multidatabase system, each preexisting local database is designed and maintained independently. Usually, a real world entity may be stored in more than one local databases at the same time. It is possible that a set of values of an attribute, each records the value of the attribute of an entity in its local database, do not match. This phenomenon is called *data inconsistency* in this paper. Data inconsistency may be categorized as a problem of semantic heterogeneity [1]. Resolving data inconsistency is crucial in building a multidatabase system.

To resolve data inconsistency, Chen et al. proposed to use probabilistic partial values [2] [3] . In their model, an attribute with inconsistent data is modeled as a discrete random variable. The attribute may take on any of the *possible* values with a predefined probability. They developed a set of extended relational operators for querying relations containing probabilistic partial values. Agarwal et al. proposed to use flexible relations [4] for resolving data inconsistency. They also developed a relational algebra for querying flexible relations which contain inconsistent data.

Yet another interesting approach was proposed by Dayal [5] [6], and is drawing a lot of attentions recently. Dayal proposed to integrate databases with inconsistent data in two steps. First, an outerjoin [7] is performed over a set of base relations (the relations corresponds to the same real world entity type). The second step is to apply aggregate functions over the attributes of the resultant relation to derive a semantically consistent global view.

For Dayal's approach to be really applicable, efficient execution of queries involving aggregate functions[1] is crucial. Dayal proposed several tactics for efficient execution of some aggregate functions. However, in our opinion, for most aggregate functions, their efficient execution strategies are not fully addressed.

In this paper, we propose two methods for the efficient execution of the aggregate functions. In the first method, we use the strategy of key value transfer to make it possible to distribute selection operations of a query to local databases. The second method is to maintain the current maximal and minimal values of an attribute. We then use these information to localize the execution of selection operations.

The rest of this paper is organized as follows. In Section 2, we briefly describe Dayal's approach. In Section 3, we propose two methods to optimize the execution of aggregate functions. In Section 4, we analyze the effectiveness of these methods. Finally, we draw some conclusions in Section 5.

*This work was supported by National Science Council, Republic of China, under Grant NSC85-2213-E-011-010

[1] It is noted that we study the execution of aggregate functions in the context of resolving data inconsistency, and should be distinguished from executing aggregate functions in SQL statements.

2 Resolving Data Inconsistency Using Aggregate Functions

In [5] [6], Dayal proposed to use aggregate functions including *max*, *min*, *sum*, *average*, *chooseany*, etc. to resolve data inconsistency. Here, we briefly restate his basic idea. First, the definitions of aggregate functions, taken literally from [5], are shown in Figure 1.

Suppose X_1 and X_2 be two inconsistent values and X is an aggregated value, then,

$$max : X = \begin{cases} X_1, & if\ X_2 = null \\ X_2, & if\ X_1 = null \\ max(X_1, X_2), & if\ X_1 \neq null\ and\ X_2 \neq null \\ null, & otherwise \end{cases}$$

min, sum and avg are defined in a similar way.

$$chooseany : X = \begin{cases} X_1, & if\ X_1 \neq null\ and\ X_2 = null \\ X_2, & if\ X_2 \neq null\ and\ X_1 = null \\ X_1\ or\ X_2, & if\ X_2 = X_1\ and\ all\ not\ null \\ null, & otherwise \end{cases}$$

Figure 1: Definition of aggregate functions

2.1 Global View Definition Using Aggregate Functions

For resolving data inconsistency, a global view is created based on the relations (i.e. base relations) of the local databases. First, an outerjoin [7] is applied on the base relations to derive an intermediate view. Then, appropriate aggregate functions are applied on the common attributes of the view to derived an integrated global view. The whole process can be illustrated by the following example.

Example 1: Suppose we have two base relations, $EMP1$ and $EMP2$, as shown in figure 2(a). Relation $EMP1$ consists of attributes ID, *name*, *age* and *salary*. While relation $EMP2$ consists of ID, *name*, *age* and *salary*. Note that these two relations possess some different attributes and, therefore, are not union compatible. To obtain a global view, we first apply an outerjoin on $EMP1$ and $EMP2$. The intermediate view is shown in figure 2(b). The next step is to define the attributes of the integrated global view in terms of aggregate functions. In this example, we use aggregate function *chooseany* for ID, *chooseall* for *name*, *max* for *age* and *sum* for *salary*.

The reasons are as follows. For simplicity, we assume that the same entity (e.g. an employee) has unique key value; and ID is the key attribute of the two base relations. For the ID attribute, no data inconsistency exists, therefore, *chooseany* is appropriate in this case. For the *name* attribute, it's possible that an employee uses different names in two local databases, and we are interested in both of his names (i.e., *chooseall*). For the *age* attribute, we usually choose the greater one (i.e., *max*) because some local database may forget to add one on an employee's age at the end of a year. For the *salary* attribute, we choose *sum* which reflects that an employee may have two jobs, each will earn him a salaries, and we are interested in the employee's total salary. The integrated global view is shown in figure 2(c).

EMP1

ID1	*name1*	*age1*	*salary1*
001	Tom	36	25,000
002	John	32	27,000
003	Mary	30	30,000
004	Jack	30	40,000
005	Remy	22	24,000

EMP2

ID2	*name2*	*age2*	*salary2*
001	W. Tom	35	30,000
003	L. Mary	31	30,000
004	A. Jack	28	38,000
009	W. Polo	26	27,000
011	W. King	36	33,000

(a) Two base relations

ID1	*name1*	*age1*	*salary1*	ID2	*name2*	*age2*	*salary2*
001	Tom	36	25,000	001	W. Tom	35	30,000
002	John	32	27,000	null	null	null	null
003	Mary	30	30,000	003	L. Mary	31	30,000
004	Jack	30	40,000	004	A. Jack	28	38,000
005	Remy	22	24,000	null	null	null	null
null	null	null	null	009	W. Polo	26	27,000
null	null	null	null	011	W. King	36	33,000

(b) Result of the outerjoin

ID	*name1*	*name2*	*age*	*salary*
001	Tom	W. Tom	36	55,000
002	John	null	32	27,000
003	Mary	L. Mary	31	60,000
004	Jack	A. Jack	30	78,000
005	Remy	null	22	24,000
009	null	W. Polo	26	27,000
011	null	W. King	36	33,000

(c) The integrated view (EMP)

Figure 2: Global view definition using aggregate functions

We deem that Dayal's approach is interesting because it allows a user (or a DBA) to define the most appropriate rules (for the need of the user) in resolving data inconsistency. Furthermore, aggregate functions are semantically rich and flexible when used in resolving data inconsistency.

2.2 Query Execution with Aggregate Functions

When a query is posed on a global view, the query must be modified and transformed into several subqueries. Each subquery contains only base relations or intermediate relations. Query execution on a global view containing aggregate functions should be optimized to prevent excessive communication overhead. Dayal proposed several tactics, such as *distribution of selection* and *semiouterjoin* [5] to reduce the communication overhead. A query containing aggregate functions can be executed efficiently using these tactics. For example, consider two base relations $EMP1(ID, weight1, age1)$ and $EMP2(ID, weight2, age2)$. A global view $EMP(ID, weight, age)$ is defined on $EMP1$ and $EMP2$, where the *age* attribute is defined to be $max(age1, age2)$. When a global query

$\sigma_{age>30}(EMP)$

is posed to the system. It can be transformed into two subqueries:

$\sigma_{age1>30}(EMP1)$ *and* $\sigma_{age2>30}(EMP2)$.

After the subqueries are performed by the local

databases, the results from the local databases are integrated into the final result. Note that in this example, instead of transferring all the tuples of both $EMP1$ and $EMP2$ to the query site, we distribute the corresponding selection formula to $EMP1$ and $EMP2$ respectively, and transfer only those tuples that are qualified for the query. It is to be noted that not all selection formulas can be safely distributed. Figure 3 summarizes the cases in which distribution of selection is possible.

aggregate function	selection formula		
	$A < c$	A=c	$A > c$
chooseany	S	S	S
max	N	T1	S
min	S	T2	N
sum	N	N	N
avg	N	N	N

$T1 : A_i \geq c$
$T2 : A_i \leq c$
$where\ i \in \{1, 2, ..., n\}$

Figure 3: Dayal's Rules for Distribution of the Selection Formula

In Figure 3, $A\Theta c$ is a selection formula, where A denotes an attribute name, Θ is comparison operator (including $<$, $=$, and $>$), and c is a constant. An "S" entry indicates that distribution of the selection formula can be performed safely without any change to the formula — a case called *full local selection* in [5]. For example, if we want to get the employees whose maximum ages are greater than c. We only need to select locally the employees whose ages are greater than c, and transmit the tuples to the query site. An "N" entry indicates that no distribution is possible. $T1$ and $T2$ means that distribution is possible only when the selection formula is changed according to $T1$ or $T2$ respectively — a case called *partial local selection* in [5].

Taking T1 for an example, assume that attribute A is defined based on A_1 in one site and A_2 in another site. Then, a query $\sigma_{A=c}$ can be transformed into $\sigma_{A_1 \geq c}$ and $\sigma_{A_2 \geq c}$.

The tactic of *distribution of selection* is beneficial because it can be implemented with little effort. However, in most cases, this tactic does not work; that is, no full or partial local selection is possible. For example, the selection formula:

$\sigma_{(weight1+weight2)/2>50}(EMP)$

cannot be distributed according to Figure 3. The only way of executing this selection is to transfer all the tuples of $EMP1$ and $EMP2$ from the corresponding local sites to the query site. The focus of this paper is to explore methods to make *distribution of selection* possible for all the entries in Figure 3.

3 Distribution of Selections with Aggregate Function

In this section, we present two methods to make distribution of selection possible for selections with all kinds of aggregate functions.

3.1 Method 1 : Key value transfer

As mentioned before, *data inconsistency* happens when two tuples match on their key attribute values but have conflicting values for some common non-key attributes. The main idea of *key value transfer* is to provide the query site with the key values of all tuples. These key values are used to prune the tuples that are not qualified for the selection condition.

For example, consider two base relations $EMP1$ and $EMP2$ as in Figure 2, where *age* is defined as *max(age1, age2)*. The selection formula

$\sigma_{age=30}(EMP)$

can be distributed as

$\sigma_{age1=30}(EMP1)$ *and* $\sigma_{age2=30}(EMP2)$,

if the key values of $age1 > 30$ from $EMP1$ and $age2 > 30$ from $EMP2$ are also transferred back to the query site. A tuple with $age1 = 30$ in $EMP1$ while has a corresponding tuple with $age2 > 30$ in $EMP2$ will be pruned by the key value of the corresponding tuple from $EMP2$. This is because the tuple with $age1 = 30$ in $EMP1$ is unqualified by its corresponding tuple which has $age2 > 30$ in $EMP2$, due to the semantics of the aggregate function *max*.

Figure 3 can be modified to Figure 4 using the *key value transfer* method. This method can be expressed as follows: Suppose that R_i contains all tuples selected from site i, and K_i contains all key values transferred from site i. Then the reduced relation R_i' of R_i can be derived by

$R_i' = R_i - Semijoin(R_i, \cup_{j \neq i} K_j)$,

where K_j contains key values from site j. R_i' is then used for performing outerjoin, aggregate functions, and selection if necessary.

aggregate function	selection formula		
	$A < c$	A=c	$A > c$
chooseany	S	S	S
max	T3	T1	S
min	S	T2	T4
sum	T5	T6	N
avg	T7	T8	N

T1: transfer tuples with $A_i = c$, and the key values of $A_i > c$
T2: transfer tuples with $A_i = c$, and the key values of $A_i < c$
T3: transfer tuples with $A_i < c$, and the key values of $A_i \geq c$
T4: transfer tuples with $A_i > c$, and the key values of $A_i \leq c$
T5: transfer tuples with $A_i < c$, and the key values of $A_i \geq c$
T6: transfer tuples with $A_i \leq c$, and the key values of $A_i > c$
T7: transfer tuples with $A_i < nc$, and the key values of $A_i \geq nc$
T8: transfer full tuples of $A_i \leq nc$, and the key values of $A_i > nc$
($i \in \{1, ..., n\}$, n is the number of sites)

Figure 4: Modified Rules for Distribution of the Selection Formula

Example 2: Consider relations $EMP1$ and $EMP2$ in Figure 2, and the execution of the query

$\pi_{id,name}\ (\sigma_{age<30}(EMP))$.

The selection formula $age < 30$ can be transformed by T3. Next, we can transform the global query into two local queries :

$Q1 : \pi_{id1,name1}\ (\sigma_{age1<30}(EMP1))$, *and*
$Q2 : \pi_{id2,name2}\ (\sigma_{age2<30}(EMP2))$,

and transfer the query results of each local execution

to the query site for final processing. Note that the key values of unqualified tuples should also be transferred to the query site. Given that John's $age1$ equals 29 and $age2$ equals 31, then max of $age1$ and $age2$ is 31, which is greater than 30, and John should be pruned from the query site. In our method, John is selected from $EMP1$ and pruned by the key values transferred from $EMP2$. If the key values is not transfered, John will be selected, which renders a wrong answer. Suppose R_i is the set of the qualified tuples from site i, and K_i is the set of key values of the unqualified tuples from site i (i=1,2), then the reduced results R_i' is:

$R_1' = R_1 - Semijoin(R_1 , K_2)$
$R_2' = R_2 - Semijoin(R_2 , K_1)$,
and the final processing is:
$EMP = (Outerjoin_{id}(R_1', R'2))$ *and*
$\pi_{id,name}\ (\sigma_{age<30}(EMP))$.
After the modification, T1 and T2 in Figure 4 are better than the original ones because the amount of data transferred is reduced enormously. T3 through T8 eliminate six "N" entries from Figure 3. However, T5, T6, T7 and T8 only work under the assumption that no negative attribute value exists. Meanwhile, T7 and T8 are not good enough because the number of sites n could be quite large. In this case, according to the selection formulas, there could be very small amount of tuples reduced by performing the local selection formulas. Therefore, we propose another method to further improve the localization of the selection formulas.

3.2 Method 2 : Maintaining the maximum and minimum value

If the maximum and minimum values of all numerical attributes are available, then the effectiveness of the local selection involving *sum* and *average* can be further improved.

After applying Method 2, the transformation rules for *sum* and *average* are shown in Figure 5.

aggregate function	selection formula		
	$A < c$	A=c	$A > c$
sum	T9	T10	T11
avg	T12	T13	T14

T9: *transfer tuples with* $A_i < max(c, c - A_{jmin})$
T10: *transfer tuples with* $min(c, c - A_{jmax}) \leq A_i \leq max(c, c - A_{jmin})$
T11: *transfer tuples with* $A_i > min(c, c - A_{jmax})$
T12: *transfer tuples with* $A_i < max(c, 2c - A_{jmin})$
T13: *transfer tuples with* $min(c, 2c - A_{jmax}) \leq A_i \leq max(c, 2c - A_{jmin})$
T14: *transfer tuples with* $A_i > min(c, 2c - A_{jmax})$
Note: *All rules have to transfer the key values of the unqualified tuples.*
$(i \in \{1,2\}, i \neq j)$

Figure 5: Transformation Rules of Method 2 for 2 Sites

In Figure 5, A_{imax} and A_{imin} denote, respectively, the maximum value and minimum value of attribute A at site i. The query site can get all A_{imax} and A_{imin} by sending a query to all sites, or by keeping the maximum values and the minimum values in the global data dictionary.

Using the information of A_{imax} and A_{imin}, Figure 4 can be improved. T11 and T14 replace the last two "N"s in Figure 4. Meanwhile, it is obvious that T12 and T13 are better than T7 and T8, respectively, because the transformed selection formulas become more selective (i.e., they can reduce more tuples from the base relations.). The proof of transformation rules T9 through T14 are shown in Appendix A . If no negative attribute value exists, rules T9, T10 and T11 can be written as:

T9: *transfer tuples with* $A_i < c$ $(= T5)$.
T10: *transfer tuples with* $c - A_{jmax} \leq A_i \leq c$.
T11: *transfer tuples with* $A_i > c - A_{jmax}$.

Example 3: Consider relations $EMP1$ and $EMP2$ in Figure 2, and the execution of the query
$\pi_{id,name}\ (\sigma_{salary>30000}(EMP))$.
Suppose that $max(Salary\ in\ EMP1)$ is 40000 and $max(Salary\ in\ EMP2)$ is 38000, we transform the global query into the following two local queries by applying rule T11:

$$Q1 : \pi_{id1,name1}\ (\sigma_{salary1>min(30000,60000-38000)}(EMP1))$$

$$Q2 : \pi_{id2,name2}\ (\sigma_{salary2>min(30000,60000-40000)}(EMP2)).$$

Finally, the query results of each local execution are transfered to the query site for final processing.

The transformation rules in Figure 5 can be generalized into n sites as follows:

T9: $A_i < max(c - \sum_{j\neq i} A_{jmin})$, *for any combination of* A_{jmin} $(A_i < c$, *if no negative attribute value exists)*
T10: $min(c - \sum_{j\neq i} A_{jmax}) \leq A_i \leq max(c - \sum_{j\neq i} A_{jmin})$, *for any combination of* A_{jmax} *or* A_{jmin} $(c - \sum_{j=1\ to\ n, j\neq i} A_{jmax} \leq A_i \leq c$, *if no negative attribute value)*
T11: $A_i > min(c - \sum_{j\neq i} A_{jmax})$, *for any combination of* A_{jmax} *(or* $A_i > c - \sum_{j=1\ to\ n, j\neq i} A_{jmax}$, *if no negative attribute value)*
T12: $A_i < max($*the number of* $A_{jmin} \times\ c - \sum_{j\neq i} A_{jmin})$, *for any combination of* A_{jmin}
T13: $min($*the number of* $A_{jmax} \times\ c - \sum_{j\neq i} A_{jmax}) \leq A_i \leq max($*the number of* $A_{jmin} \times\ c - \sum_{j\neq i} A_{jmin})$, *for any combination of* A_{jmax} *or* A_{jmin}
T14: $A_i > min($*the number of* $A_{jmax} \times\ c - \sum_{j\neq i} A_{jmax})$, *for any combination of* A_{jmax}
Note: *All rules have to transfer the key values of the unqualified tuples* $(i \in \{1,2,...,n\}$, *n is the number of sites)*

4 Effectiveness Analysis

So far, we have discussed how to improve the distribution of selection formulas involving aggregate functions. In this section, we discuss when the distribution of a selection formula is beneficial and how beneficial it is.

In evaluating the effectiveness, we define the *ratio of reduced data* at each site as the proportion of the reduced data (in terms of bytes) to the total amount of data in the base relation. In other words, the *ratio of reduced data* is one minus the ratio of transferred

data in a base relation. For simplicity, we assume that the values of an attribute A_i are uniformly distributed over its domain. That is, attribute A_i may take on any possible value from its domain with equal probability. In the following analysis, we further assume that the value of A_i ranges from 0 to 100. We also define *key ratio* of a base relation as the ratio of the length (in terms of bytes) of the key attributes to the length of a tuple in the relation (assuming that the base relation contains fixed-length tuple). The values of r are set to 0.1, 0.2 or 0.3 in different cases of our analysis.

4.1 Analysis of Method 1

The effectiveness of transformation rules listed in Figure 4 depends on the value c of the selection formula $A\Theta c$ as well as the *key ratio* , r. For example, if T1 is applied, all tuples for $A_i = c$ and the key values for $A_i > c$ are all transferred to the query site. The *ratio of reduced data* is $1 - r \cdot \frac{100-c}{100-0}$, where $\frac{100-c}{100-0}$ is the proportion of tuples with $A_i > c$, and $r \cdot \frac{100-c}{100-0}$ is the proportion (in terms of bytes) of keys to the tuples with $A_i > c$. It shows that the transformation is beneficial for $0 \leq c \leq 100$. Other rules can be analyzed in a similar way. The results of the analysis are shown in Table 1.

Transformation rule	Beneficial range	Ratio of reduced data
T1	$0 \leq c \leq 100$	$1 - r \cdot \frac{100-c}{100-0}$
T2	$0 \leq c \leq 100$	$1 - r \cdot \frac{c-0}{100-0}$
T3	$0 < c \leq 100$	$(1-r) \cdot \frac{100-c}{100-0}$
T4	$0 \leq c < 100$	$(1-r) \cdot \frac{c-0}{100-0}$
T5	$0 < c \leq 100$	$(1-r) \cdot \frac{100-c}{100-0}$
T6	$0 \leq c \leq 100$	$(1-r) \cdot \frac{100-c}{100-0}$
T7	$\frac{0}{n} < c \leq \frac{100}{n}$	$(1-r) \cdot \frac{100-nc}{100-0}$
T8	$\frac{0}{n} \leq c \leq \frac{100}{n}$	$(1-r) \cdot \frac{100-nc}{100-0}$

Table 1. Relationship between c and the ratio of reduced data for method 1

We sketch eight diagrams for T1 to T8 as shown in Figure 6. From Figure 6.(a) to 6.(f), it shows that transformation rules T1, T2, T3, T4, T5 and T6 are always beneficial. However, from Figure 6.(g) and (h), transformation rules T7 and T8 are beneficial only when c is less than 50. It turns out that transformation rule T7 and T8 are beneficial only when the value of c is small, i.e., when the value of $n \times c$ is within the domain of A_i (where n is the number of sites).

4.2 Analysis of Method 2

The effectiveness of transformation rules listed in Figure 5 depends not only on the value of c and the ratio r of key attribute, but also on the current maximum and minimum values of the attribute at other sites. Therefore, the analysis is more complicated. For simplicity, we consider only two sites, (n=2), and assume that the value of A_i ranges from 0 to 100. The *ratio of reduced data* for the rules are shown in Table 2.

	Ratio of reduced data
T9	$(1-r) \cdot \frac{100-c}{100-0}$, *when* $A_{jmin} \geq 0$
T10	$(1-r) \cdot (1 - \frac{min(A_{jmax}, c)}{100-0})$, *when* $A_{jmin} \geq 0$
T11	$(1-r) \cdot \frac{max(0, c-A_{jmax})}{100-0}$, *when* $A_{jmin} \geq 0$
T12	$(1-r) \cdot \frac{100-c}{100-0}$, *when* $c \leq A_{jmin}$ $(1-r) \cdot \frac{max(0, 100-(2c-A_{jmin}))}{100-0}$, *when* $c > A_{jmin}$
T13	$(1-r) \cdot (1 - \frac{c-max(0, 2c-A_{jmax})}{100-0})$, *when* $A_{jmin} \geq c$ $(1-r) \cdot (1 - \frac{min(100, 2c-A_{jmin}) - max(0, 2c-A_{jmax})}{100-0})$, *when* $A_{jmin} < c < A_{jmax}$ $(1-r) \cdot (1 - \frac{min(100, 2c-A_{jmin}) - c}{100-0})$, *when* $A_{jmax} \leq c$
T14	$(1-r) \cdot \frac{c-0}{100-0}$, *when* $c \geq A_{jmax}$ $(1-r) \cdot \frac{max(0, 2c-A_{jmax})}{100-0}$, *when* $c < A_{jmax}$

Table 2. Relationship between c and the ratio of reduced data for method 2

We sketch six diagrams, as shown in Figure 7, one for each of T9 to T14 respectively. Figure 7.(a) shows that the transformation rule T9 is always beneficial. Figure 7.(b) manifests that transformation rule T10 is always beneficial, and the ratio of the reduced data becomes a constant when c is greater than A_{jmax} (current max value at the other site). That is to say, the smaller the current A_{jmax} is, the larger the ratio of reduced data will be. Figure 7.(c) shows that transformation rule T11 is beneficial only when the value of c is greater than the current A_{jmax}. Figure 7.(d) shows that transformation rule T12 is beneficial only when the value of c is less than $\frac{1}{2}(100 + A_{jmin})$, and the ratio of reduced data becomes a constant when c is less than the current A_{jmin}. Figure 7.(e) shows that transformation rule T13 is always beneficial, and the ratio of reduced data is minimized when the value of c is between $\frac{1}{2}(100 + A_{jmin})$ and $\frac{1}{2}(0 + A_{jmax})$. Figure 7.(f) shows that transformation rule T14 is beneficial only when the value of c is greater than $\frac{1}{2}(0 + A_{jmax})$.

In general, the first method is effective when the aggregate function is a *max* or a *min*, and the second method is effective when the aggregate function is a *sum* or an *average*.

5 Conclusion and future research

In this paper, we proposed two methods to distribute selection formulas in a global query in which aggregate functions are used for resolving data inconsistency. Using our methods, global queries with aggregate functions can be executed efficiently through the localization of the selection formulas. The contributions of this paper are two folds. First, we extend Dayal's approach on localization of selection formulas such that, in most cases, the selection formulas involving aggregate functions can be executed efficiently. Second, we quantified the effectiveness of the transformation rules by the ratio of reduced data. This give us the clue about how effective a transformation rule is and when it should or should not be applied. This could be used as a starting point for developing a more comprehensive query optimization strategy.

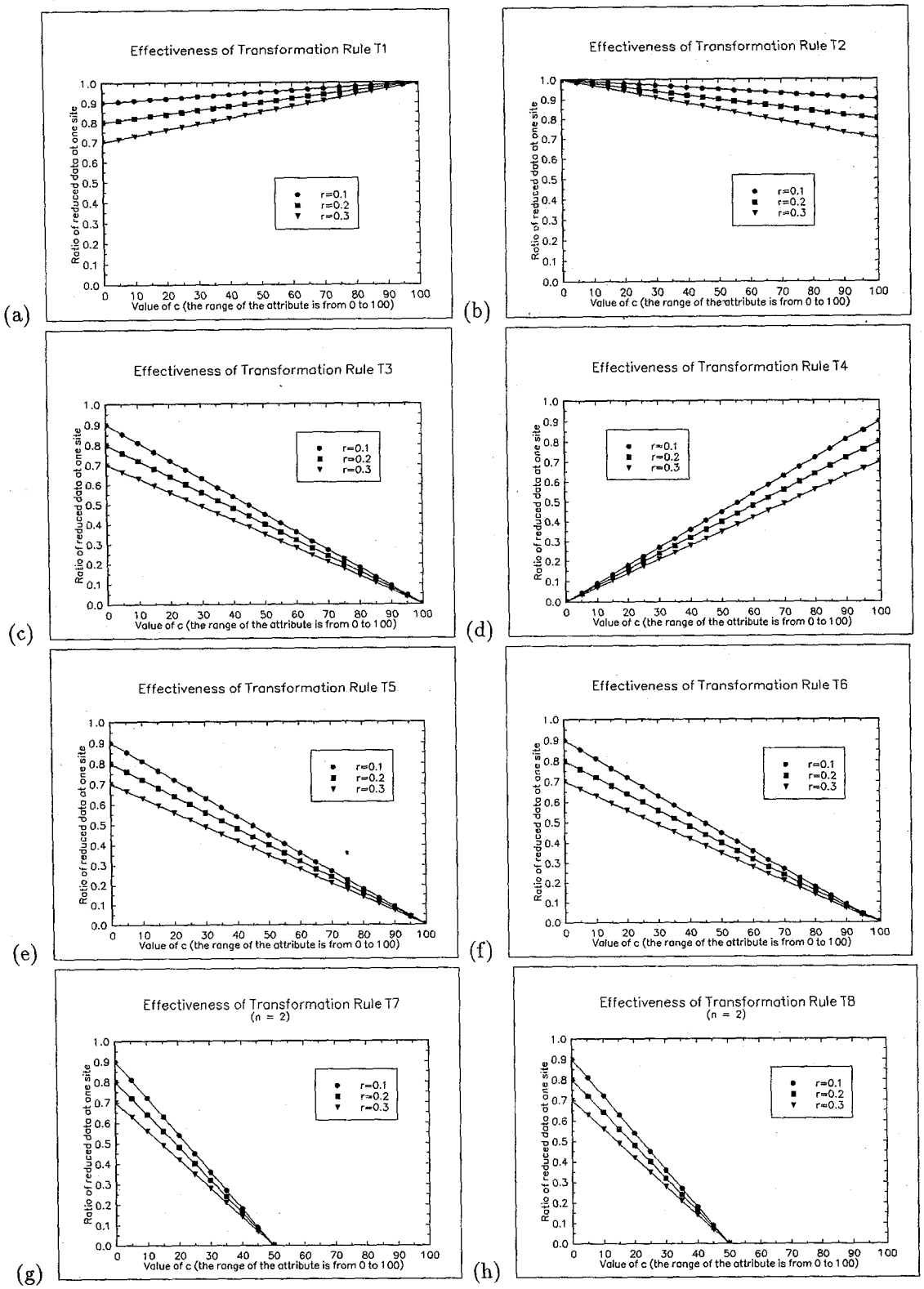

Figure 6: Effectiveness analysis for *method 1*

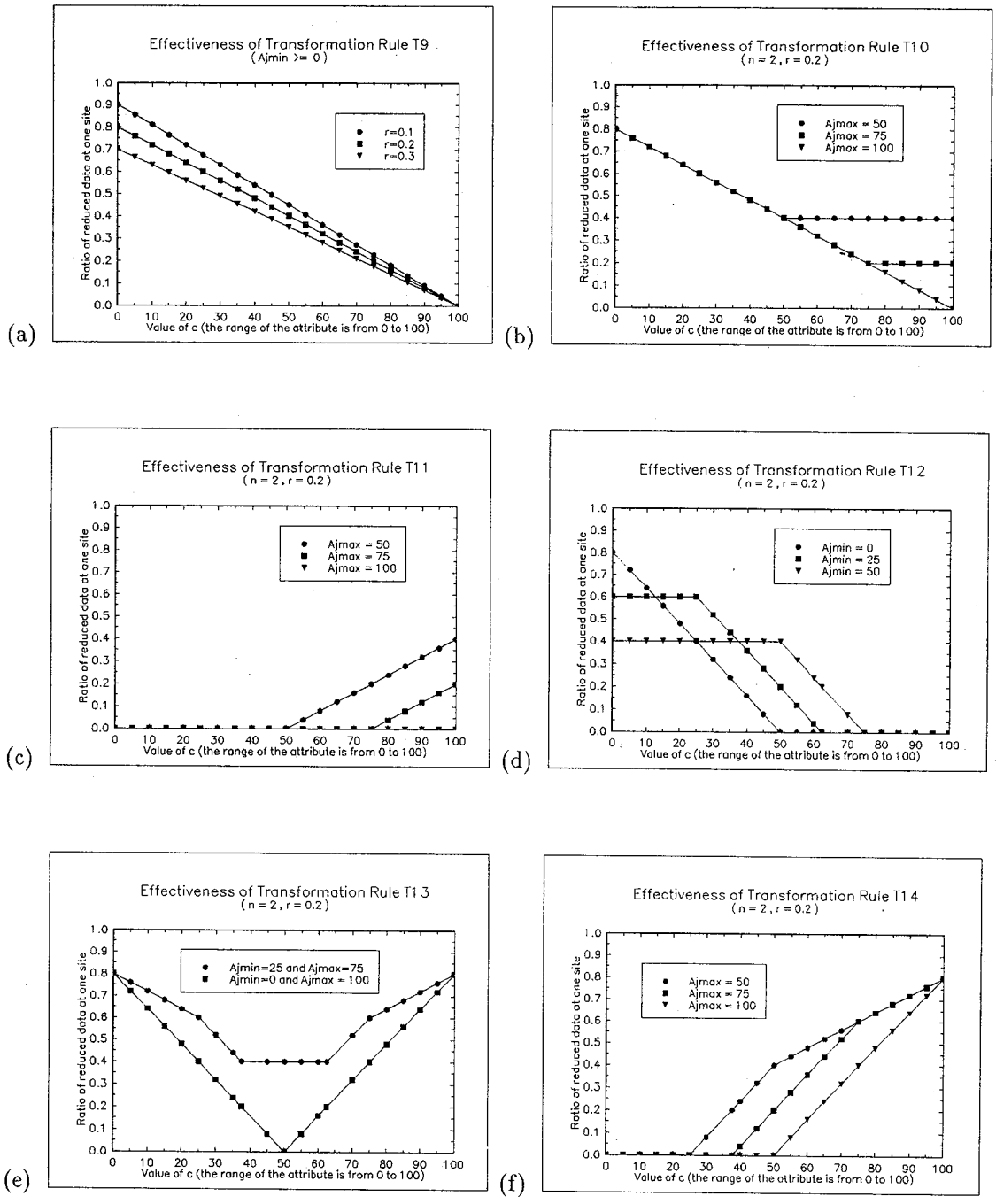

Figure 7: Effectiveness analysis for *method 2*

References

[1] **W. Kim and J. Seo**, "Classifying schematic and data heterogeneity in multidatabase systems," Computer, Dec., 1991, pp. 12-18.

[2] **F.S.C. Tseng, A.L.P. Chen and W.P. Yang**, "A probabilistic approach to query processing in heterogeneous database systems," Second International Workshop on Research Issues on Data Engineering:Transaction and Query Processing, 1992, pp. 176-183.

[3] **P.S.M. Tsai and A.L.P. Chen**, "Query uncertain data in heterogeneous databases," Third International Workshop on Research Issues on Data Engineering:Interoperability in Multidatabase Systems, 1993, pp. 161-168.

[4] **S. Agarwal, A. M. Keller, G. Wiederhold and K. Saraswat**, "Flexible relation: an approach for integrating data from multiple, possibly inconsistent databases," Proc. of 11th IEEE Int'l Conf. on Data Engineering, 1995, pp. 495-504.

[5] **U. Dayal**, "Processing queries over generalization hierarchies in a multidatabase system," Proc. VLDB Conference, 1983, pp. 342-353.

[6] **U. Dayal**, "Query processing in a multidatabase system," Query Processing in Multidatabase systems, 1985, pp. 81-108, edited by W. Kim, D. Reimer, and S. Batory.

[7] **C.J. Date**, "The outer join," Proc. Second International Conference on Databases, 1983.

Appendix A : The Proof of Transformation Rules of Method 2 (where i=1, j=2 or i=2, j=1)

T9 : When the aggregate function is a *sum* and the selection formula is $A < c$, we can transform the global selection $A < c$ to local selection $A_i < max(c, c - A_{jmin})$.

pf : (1) When the attribute exists at both sites:
$A < c \Rightarrow A_1 + A_2 < c \Rightarrow \begin{cases} A_1 < c - A_{2min} \\ A_2 < c - A_{1min} \end{cases} \Rightarrow A_i < c - A_{jmin}$
(2) When the attribute exists only at one site:
$A < c \Rightarrow A_i < c$
From (1) and (2), we have $A_i < max(c, c - A_{jmin})$.

T10 : When the aggregate function is a *sum* and the selection formula is $A = c$, we can transform the global selection $A = c$ to local selection $min(c, c - A_{jmax}) \leq A_i \leq max(c, c - A_{jmin})$.

pf : (1) When the attribute exists at both sites:
$A = c \Rightarrow A_1 + A_2 = c \Rightarrow \begin{cases} c - A_{2max} \leq A_1 \leq c - A_{2min} \\ c - A_{1max} \leq A_2 \leq c - A_{1min} \end{cases} \Rightarrow c - A_{jmax} \leq A_i \leq c - A_{jmin}$
(2) When the attribute exists only at one site:
$A = c \Rightarrow A_i = c \Rightarrow c \leq A_i \leq c$
From (1) and (2), we have $min(c, c - A_{jmax}) \leq A_i \leq max(c, c - A_{jmin})$.

T11 : When the aggregate function is a *sum* and the selection formula is $A > c$, we can transform the global selection $A > c$ to local selection $A_i > min(c, c - A_{jmax})$.

pf : (1) When the attribute exists at both sites:
$A > c \Rightarrow A_1 + A_2 > c \Rightarrow \begin{cases} A_1 > c - A_{2max} \\ A_2 > c - A_{1max} \end{cases} \Rightarrow A_i > c - A_{jmax}$
(2) When the attribute exists only at one site:
$A > c \Rightarrow A_i > c$
From (1) and (2), we have $A_i > min(c, c - A_{jmax})$.

T12 : When the aggregate function is an *avg* and the selection formula is A<c, we can transform the global selection A<c to local selection $A_i < max(c, 2c - A_{jmin})$.

pf : (1) When the attribute exists at both sites:
$A < c \Rightarrow \frac{1}{2}(A_1 + A_2) < c \Rightarrow \begin{cases} A_1 < 2c - A_{2min} \\ A_2 < 2c - A_{1min} \end{cases} \Rightarrow A_i < 2c - A_{jmin}$
(2) When the attribute exists only at one site:
$A < c \Rightarrow A_i < c$
From (1) and (2), we have $A_i < max(c, 2c - A_{jmin})$.

T13 : When the aggregate function is an *avg* and the selection formula is A=c, we can transform the global selection A=c to local selection $min(c, 2c - A_{jmax}) \leq A_i \leq max(c, 2c - A_{jmin})$.

pf : (1) When the attribute exists at both sites:
$A = c \Rightarrow \frac{1}{2}(A_1 + A_2) = c \Rightarrow \begin{cases} 2c - A_{2max} \leq A_1 \leq 2c - A_{2min} \\ 2c - A_{1max} \leq A_2 \leq 2c - A_{1min} \end{cases} \Rightarrow 2c - A_{jmax} \leq A_i \leq 2c - A_{jmin}$
(2) When the attribute exists only at one site:
$A = c \Rightarrow A_i = c \Rightarrow c \leq A_i \leq c$
From (1) and (2), we have $min(c, 2c - A_{jmax}) \leq A_i \leq max(c, 2c - A_{jmin})$.

T14 : When the aggregate function is an *avg* and the selection formula is A>c, we can transform the global selection A>c to local selection $A_i > min(c, 2c - A_{jmax})$.

pf : (1) When the attribute exists at both sites:
$A > c \Rightarrow \frac{1}{2}(A_1 + A_2) > c \Rightarrow \begin{cases} A_1 > 2c - A_{2max} \\ A_2 > 2c - A_{1max} \end{cases} \Rightarrow A_i > 2c - A_{jmax}$
(2) When the attribute exists only at one site:
$A > c \Rightarrow A_i > c$
From (1) and (2), we have $A_i > min(c, 2c - A_{jmax})$.