Query Execution Strategies for Missing Data in Distributed Heterogeneous Object Databases*

Jia-Ling Koh and Arbee L.P. Chen

Department of Computer Science
National Tsing Hua University
Hsinchu, Taiwan 300, R.O.C.
Email : alpchen@cs.nthu.edu.tw

Abstract

The problem of missing data arises in the distributed heterogeneous object databases because of the missing attribute conflict and the existence of null values. A set of algorithms are provided in this paper for query processing when the predicates in global queries involve missing data. For providing more informative answers to the users, the maybe results due to missing data are presented in addition to the certain results. One algorithm is designed based on the centralized approach in which the data are sent to the same site for integration and then processing. On the other hand, for reducing the response time, the localized approaches evaluate the predicates in different component databases in parallel. The proposed algorithms are compared and discussed by the simulation result on the total execution time and response time.

1 Introduction

The rapid development of computer networks in recent years has made data sharing among multiple databases important. In a distributed heterogeneous system which consists of object databases, a global object schema created by integrating schemas of the component databases provides a uniform interface and high level location transparency for the users to retrieve data. A variety of approaches to schema integration have been proposed [1], [9], [15], [17]. In our previous work, we had presented a schema integration mechanism to achieve a global object schema for existing multiple object databases [13]. Moreover, a form of equation was defined to denote the mapping between the global and component schemas in [14]. One mechanism for processing the queries against the constructed global schema was also introduced in [14] and [4].

In the process of schema integration, the classes in the component schemas with the same semantics are integrated to construct a class in the global schema. In the following context, the constructed classes in the global schema are called *global classes*; the classes which are integrated to construct one global class are named *constituent classes* of the global class. The attributes of a global class are the set union of the attributes belonging to its constituent classes. The conflict of *missing attribute* arises when there exist semantically different attributes in the constituent classes [12]. The attributes appearing in the global class but not defined in constituent class **C** are named *missing attributes* of **C** (we also say **C** *holds* the missing attributes). The data for the missing attributes are called *missing data* and considered to be *null*. When the predicates of a global query involve the missing attributes of **C**, the objects in **C** are evaluated to be *maybe results* [7] if they satisfy the predicates except the ones involving the missing attributes. In other words, *maybe results* are produced due to the existence of missing data. In addition to the data for missing attributes, the null values originally existing in the component databases are also one kind of missing data.

A real-world entity is represented as an object in an object-oriented database. However, in a distributed heterogeneous object database system, the same real-world entity may exist in more than one object database with incompatible local object identifiers(**LOids**). We have discussed a strategy in [5] to find the objects, named *isomeric objects*, which are stored in different databases but representing the same real-world entity. The data for these isomeric objects need to be combined for providing complete information of a real-world entity represented in the distributed system. Therefore, one object evaluated to be a maybe result in a component database may be turned into a *certain result* when combined with the results from its isomeric objects in other component databases.

Many distributed optimization algorithms have been proposed. A set of rules were provided in [2] to transform the operations in the global query into the ones with less cost. A global query may also be decomposed into subqueries which can be executed in local sites in parallel [3]. [20] used semijoin to reduce the cost of data transmission. Determining the join sequence for reducing the transmission cost was studied in [6]. Query processing strategies for homogeneous distributed object database systems were discussed in [10]. None of the previous papers considered the missing data due to schema conflicts. The schema and domain incompatibilities in the multidatabase systems were considered in [11] and [16]. The approaches of partial values and probabilistic partial values were proposed to solve the

*This work was partially supported by the Republic of China National Science Council under Contract No. NSC 85-2213-E-007-024.

1063-6927/96 $5.00 © 1996 IEEE
Proceedings of the 16th ICDCS

schema integration problems including missing data in [8] and [18], respectively. However, the query processing algorithms were not discussed in these two papers.

In this paper, based on the object data model, the query processing is considered in particular for the situation when missing data caused by the missing attributes and null values are involved in the predicates of global queries. In addition to the certain results, the maybe results due to missing data are also provided. The concept of object isomerism is used in the query processing to turn the local maybe results into certain results. Therefore, the users can get more informative answers from the results. A set of algorithms are provided. One algorithm is designed based on the centralized approach in which the data are sent to the same site for integration and then processing. On the other hand, for reducing the response time, the *localized approaches* [3] are used to evaluate the predicates in different component databases in parallel. Finally, these algorithms are compared and discussed by the simulation result.

This paper is organized as follows. The next section clarifies the various kinds of missing data considered in this paper. The concepts of the centralized and localized approaches for processing the global queries involving missing data are also introduced by examples. Three algorithms are provided in Section 3. Section 4 presents the performance study for the algorithms. Finally, Section 5 concludes this paper with a discussion of the future work.

2 Missing Data and Global Query Processing

2.1 Missing Data and Missing Attributes

In this subsection, we start by giving an example to show the various kinds of missing data in a distributed heterogeneous object database system. Figure 1 shows the schemas of three databases: **DB1**, **DB2**, and **DB3**, which are located in different sites and used to store the personal information at the same school. The global schema shown in Figure 2 is constructed by integrating these three component schemas. Note that the schemas only include the class composition hierarchies without the class hierarchy. This paper focuses on the incompatibility of missing attributes among heterogeneous object databases. Therefore, the other conflicts among the object schemas are not discussed.

The missing data mainly come from the existence of missing attributes for the constituent classes. The missing attributes for a class can be divided into *primitive missing attributes* and *complex missing attributes* according to their types of being primitive or complex attributes. A complex missing attribute *MA* for a class **C** implies that the data for all nested attributes rooted at the domain class of *MA* are also missing for class **C**.

The other sources of the missing data are the original null values existing in the objects. If an object contains a null value for an attribute, the attribute is considered to be a missing attribute for the object. Null values may also occur in a complex attribute of an object,

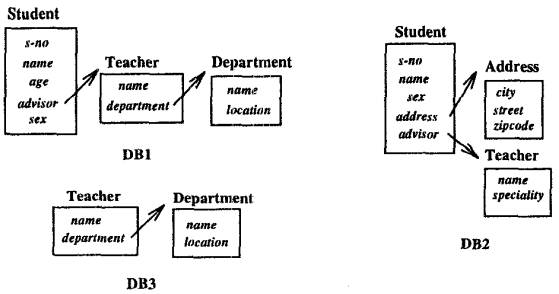

Figure 1: The component schemas

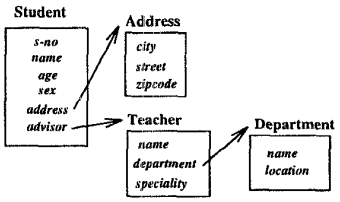

Figure 2: The constructed global schema

which results in a complex missing attribute for the object.

A predicate which involves a nested attribute of the range class is called a *nested predicate*. The nested attributes are represented by path expressions in a query. We consider the queries containing one range class, whose predicates contain nested predicates combined in conjunctive form. The range class is the *root class* of the composition hierarchy involved in the predicates; the other classes specified in the hierarchy are named *branch classes* for the query. The constituent classes of a root class and a branch class are called *local root classes* and *local branch classes*, respectively.

According to the constructed global schema shown in Figure 2, the query **Q1** is given as "Retrieve the name and the name of the advisor for the students living in Taipei, whose advisors are teachers in department of computer science and specialize in database." **Q1** is formulated in the form of SQL/X[19] an d shown in Figure 3(a). For **DB1**, the local root class **Student** has a complex missing attribute *address*; and *speciality* is a primitive missing attribute of the local branch class **Teacher**. On the other hand, the local branch class **Teacher** in **DB2** holds a complex missing attribute *department*.

2.2 A Centralized Scheme for Processing Missing Data

The basic schemes for processing missing data include a centralized approach and a localized approach, which are discussed in this and the next subsections, respectively. The query **Q1** given in subsection 2.1 is used to illustrate the main ideas of these approaches.

The attribute values of the object instances in the

Q1: *Select* **X**.*name*, **X**.*advisor.name*
From **Student X**
Where **X**.*address.city*=Taipei
and X.*advisor.speciality*=database
and X.*advisor.department.name*=CS

(a)

Q1': *Select* **X**.*Oid*, **X**.*advisor*, **X**.*name*, **X**.*advisor.name*
From **Student@DB1 X**
Where **X**.*advisor.department.name*=CS

Q1'': *Select* **X**.*Oid*, **X**.*advisor*, **X**.*name*, **X**.*advisor.name*
From **Student@DB2 X**
Where **X**.*address.city*=Taipei
and X.*advisor.speciality*=database

(b)

Figure 3: Example query **Q1** and its associated local queries

three component databases are shown in Figure 4(a), 4(b), and 4(c), respectively. The value denoted by the **bold** form is the **LOid** of an object. In this paper, we assume that the isomeric objects have been determined [5]. Each object in the distributed system is assigned a global object identifier (**GOid**), and the **GOids** for the isomeric objects are the same. The mappings among **LOids** and **GOids** are stored in the *GOid mapping tables* as shown in Figure 5.

As the name of centralized approach shows, all the objects in the local root classes and local branch classes are sent to the same site to process the predicates. In the example for processing **Q1**, the objects in all the classes shown in Figure 1 are sent to the global processing site. Consequently, the objects in the constituent classes for the same global class are integrated by using the outerjoin over their **GOids**. It is possible for an object with missing data to get the data from its isomeric objects. For example, object **s2'** can get value *31* for its missing attribute *age* from its isomeric object **s1**. Moreover, the **LOids** specified for the values of complex attributes are transformed to their **GOids**. The integrated result is shown in Figure 6. The global query is then processed against these integrated objects. Finally, the certain result (*Hedy, Kelly*) is obtained. and the maybe result is (*Tony, Haley*) because of the null values in *address* of Tony and *speciality* of Haley.

2.3 A Localized Scheme for Processing Missing Data

In the localized approach, the global query is decomposed into local queries against their associated local root classes. Assume an object o is located in component database **D**, the predicates involving missing attributes or null values of o are named *unsolved predicates* on o. The unsolved predicates which involves missing attributes are removed from the corresponding local query for component database **D**. The modified predicates are called *local predicates*, which can be

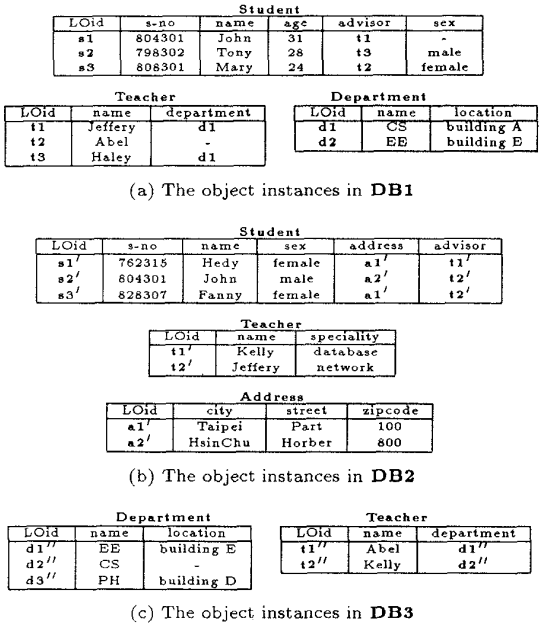

Figure 4: The object instances in component databases

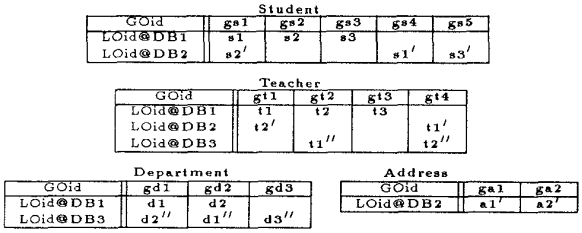

Figure 5: The GOid mapping tables

evaluated in component database **D**. Thus, the predicate evaluation is localized for achieving inter-site parallelism. On the other hand, the unsolved predicates can be evaluated if there exist isomeric objects of o, which contain the associated missing data. Such an isomeric object is called an *assistant object* for object o, which can be found by checking the **GOid** mapping tables and the other component schemas. Since each object o in the maybe results is produced due to the existence of missing data, there exists at least one unsolved predicate on o. o is defined to be *unsolved*. When only the local root class holds the missing attributes, the local maybe result o is turned into a certain result if o is solved. The rule for judging an object, which is unsolved, to become *solved* is as follows.

Certification Rule: An unsolved object o can be turned into a solved object if its assistant objects jointly satisfy all the unsolved predicates on o. Object o is eliminated when any of its assistant object violates an unsolved predicate.

Student

GOid	s-no	name	age	advisor	sex	address
gs1	804301	John	31	gt1	male	ga2
gs2	798302	Tony	28	gt3	male	-
gs3	808301	Mary	24	gt2	female	-
gs4	762315	Hedy	-	gt4	female	ga1
gs5	828307	Fanny	-	gt1	female	ga1

Teacher

GOid	name	department	speciality
gt1	Jeffery	gd1	network
gt2	Abel	gd2	-
gt3	Haley	gd1	-
gt4	Kelly	gd1	database

Department

GOid	name	location
gd1	CS	building A
gd2	EE	building E
gd3	PH	building D

Address

LOid	city	street	zipcode
ga1	Taipei	Park	100
ga2	Kaohsiung	Horber	800

Figure 6: The object instances for global classes when materized

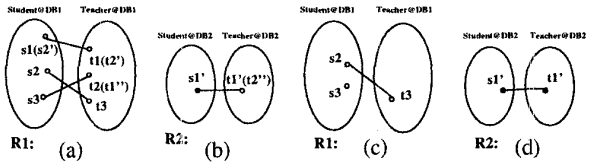

Figure 7: The local results for processing **Q2**

The process of applying the rule on an unsolved object is called to *certify* the unsolved object. The process to evaluate an unsolved predicate on an assistant object is called to *check* the assistant object.

However, additional processing is needed when the local branch classes hold missing attributes. It is necessary also to certify the objects in the nested complex attributes of the maybe results, which have the associated missing data. The local queries should include projecting the nested complex attributes which hold the missing attributes. For each maybe result o_m, the value for such a nested complex attribute is an object o_{nc}. There exists at least one unsolved predicate on the missing attributes of o_{nc}. o_{nc} is named an *unsolved item* of maybe result o_m. If o_{nc} satisfies the certification, it is turned into a *solved item*. Therefore, the assistant objects for the unsolved items need to be checked for certifying the unsolved items. The **LOids** of these assistant objects are then collected according to the classes to which they belong and sent to the corresponding component databases. Moreover, the associated unsolved predicates of the unsolved items are also sent with the **LOids** in order to check the assistant objects. The local maybe results and their unsolved items are sent to the global processing site, and wait for the checking results of assistant objects for certifying the unsolved items. When the certifications of unsolved items and unsolved maybe results complete, the local maybe result o_m can be turned into a certain result if o_m is solved and all its unsolved items become solved. However, o_m is eliminated from the results when any unsolved item of o_m is eliminated. In the other situations, o_m remains as maybe results.

Accordingly, the local queries for processing **Q1** are constructed as **Q1'** and **Q1''** shown in Figure 3(b). The maybe results obtained from **Q1'** are (**s1**, **t1**, John, Jeffery), (**s2**, **t3**, Tony, Haley), and (**s3**, **t2**, Mary, Abel) with unsolved predicates on missing attributes *address* and *advisor.speciality*, and an unsolved predicate on *advisor.department* for **s3**. (**s1'**, **t1'**, Hedy, Kelly) is maybe result obtained from **Q1''** with the unsolved predicate on missing attribute *advisor.department*. Let the local maybe results be named **R1** and **R2** and denoted by the object graphs shown as Figure 7(a) and 7(b), respectively. The blank circles represent the unsolved maybe results and the unsolved items. On the other hand, the black circles denote the solved maybe results and solved items. The **LOids** enclosed in the parentheses represent the assistant objects for the unsolved maybe results and unsolved items. For example, for the unsolved items in **Teacher@DB1**, **t2'** and **t1''** are the assistant objects for **t1** and **t2**, respectively. Therefore, the assistant object of **t1**, **t2'**, is sent to **DB2** with the predicate *"speciality=database"*. Similarity, **t1''** is sent to **DB3** for the unsolved item **t2** with the predicate on *department*. Note that no assistant object can provide the data of attribute *speciality* for object **t2**. **R1**, the local results from **DB1**, is then sent to the global processing site and wait for the checking results of **t2'** and **t1''**. Since there does not exist any assistant object satisfying the checking, the unsolved items **t1** and **t2** are eliminated. The other unsolved items remain unsolved because they do not have corresponding assistant objects for checking. Moreover, the unsolved maybe result **s1** is eliminated because its assistant objects are not obtained in the local results from **DB2**. The local results after certification are shown in Figure 7(c). The only one result in **R1** is a maybe result (*Tony, Haley*) identified by **s2**. The same processing is applied for **R2** for checking the assistant objects **t3''** and **t4**. The results after certification for **R2** are shown in Figure 7(d). Finally, the certain result (*Hedy, Kelly*) is obtained.

3 Processing Algorithms

From the discussion in the previous section, the following three phases are necessary for processing the queries involving missing data.

- *phase* **O**: this phase examines the **GOid** mapping tables to find the isomeric objects which can provide the missing data. In other words, this is the step for looking up the assistant objects. For the localized approach, the task for checking the assistant objects is also included in this phase.
- *phase* **I**: this phase integrates the information of one object and its isomeric objects. For the localized approach, it means the step for certifying the local maybe results and the unsolved items.
- *phase* **P**: this phase does the predicate evaluation. It is possible that the concerned predicates are local predicates when the localized approach is applied.

Note that phase **I** has to be executed after phase **O**.

In this section, the basic processing algorithms are provided by analyzing the combination of these three

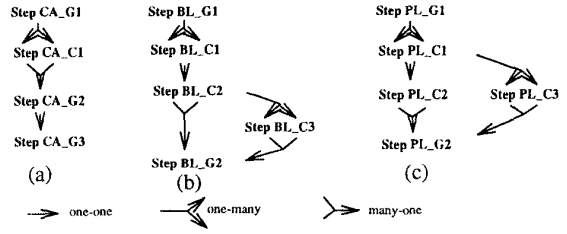

Figure 8: The executing flows of the processing algorithms

phases. Moreover, various algorithms are introduced when the auxiliary structure for storing object signatures is supported to aid the processing.

3.1 Centralized Approach (CA)

The centralized approach illustrated in subsection 2.2 follows the **O** → **I** → **P** order for executing the three necessary phases. The procedures executed in the global processing site and each component database are as follows.

global processing site

Step CA_G1: Send the request to component databases for retrieving the objects in the local root classes and local branch classes, then wait for the response.

Step CA_G2: To materialize each global class involved in the query, outerjoin is used over join attribute **GOid** to integrate the objects in the constituent classes (phase **O** and phase **I**).

Step CA_G3: Evaluate the predicates on the materialized global classes (phase **P**).

component database

Step CA_C1: When receiving the request from the global processing site, retrieve and send out all objects in the local root class and local branch classes of the query.

The executing sequence among these steps is shown in Figure 8(a). Since the optimization issue is concerned, the retrieved objects in step **CA_C1** are projected on the **LOids** and the attributes involved in the query before being transferred to the global processing site.

3.2 Basic Localized Approach (BL)

The basic localized approach is the approach illustrated in subsection 2.3 for explaining the concept of the localized approach. The execution order for the necessary phases is **P** → **O** → **I**. The tasks of the global processing site and the component databases are described as follows.

global processing site

Step BL_G1: For each component database containing the local root class, produce a local query against the local root class. The predicates remain unchanged at this step. Then send the local queries to the component databases and wait for the response.

Step BL_G2: According to the certification rule, certify the local results using the results of checking the assistant objects (phase **I**). Get the final certain results and maybe results.

component database

Case 1: receive the local query sent from the global processing site

Step BL_C1: Only evaluate the local predicates in the local query if the missing data is involved in the original predicates (phase **P**).

Step BL_C2: Examine **GOid** mapping tables to find the **LOids** of the assistant objects for the unsolved items of the local maybe results. The **LOids** of the assistant objects and the corresponding unsolved predicates are sent to the other associated component databases for further checking (part of phase **O**). The local results are then sent back to the global processing site.

Case 2: receive the request from other component databases for checking the assistant objects

Step BL_C3: Retrieve the objects for the **LOid** list of the assistant objects and evaluate the appended unsolved predicates. The **LOids** of the satisfied objects are sent back to the global processing site (phase **O**).

Figure 8(b) shows the whole executing sequence among these tasks.

3.3 Parallel Localized Approach (PL)

The execution order for the parallel localized approach is **O** → **P** → **I**. This approach makes the tasks for checking the assistant objects and evaluating local predicates to be executed in parallel in different component databases.

global processing site

Step PL_G1: This step is the same as step **BL_G1**.

Step PL_G2: This step is the same as step **BL_G2**.

component database

Case 1: receive the local query sent from the global processing site

Step PL_C1: For any object *o* in the local root class, retrieve the objects in the nested complex attributes of *o*, which have associated missing attributes. There exists at least one unsolved predicate for these retrieved objects. Check **GOid** mapping tables to find the **LOids** of the assistant objects for these objects. Send the **LOids** of the assistant objects and the corresponding unsolved predicates to the associated component databases (part of phase **O**).

parameters	description	setting
S_a	average size of attributes	32 bytes
S_{GOid}	size of **GOid**	16 bytes
S_{LOid}	size of **LOid**	16 bytes
S_s	size of object signatures	32 bytes
T_d	average disk access time	15 μs/byte
T_{net}	average network transfer time	8 μs/byte
T_c	average cpu processing time	0.5 μs/comparison
N_{iso}	average number of isomeric objects for the same real world entity	2

Table 1: The system parameters

Step PL_C2: This step is the same with step **BL_C1** (phase **P**). Then the local results are sent to the global processing site.

Case 2: receive the request from other component databases for checking the assistant objects

Step PL_C3: This step is the same with step **BL_C3** (phase **O**).

The difference between the parallel localized approach and the basic localized approach is the order for executing the tasks of Case 1 in the component databases. Moreover, step **PL_C2** and **PL_C3** are executed in parallel, which is shown in Figure 8(c).

4 Performance Study

In order to evaluate the performance of the proposed algorithms, a simulation experiment is implemented to estimate the total execution time and response time of the associated algorithms.

4.1 Simulation Setup

The simulation model consists of a number of component DBMSs connected by a communication network. There is a processor, a memory, and a hard disk in each component DBMS. Moreover, the GOid mapping table is replicated at each site. The mechanism used for managing the replicated data in the distributed environment can be applied to maintain the replicated GOid mapping tables.

The system parameters are given in Table 1. The database and query parameters are listed in Table 2. The default setting for the parameters is also shown in the table. In the experiment, the setting of the parameters is adjusted for observing the performance of the proposed algorithms. According to a setting range for the parameters, 500 sets of values for the parameters are generated. For each proposed algorithm, the total execution time and response time for executing the generated samples are computed and averaged to represent the times for this setting.

4.2 Simulation Result

- The average number of objects in each constituent class is adjusted:

  Figure 9(a) shows the total execution time for the three algorithms. The total execution time of algorithms **BL** and **PL** is shorter than that of algorithm **CA** in this situation. The main reason is that the local processing can eliminate the objects which do not satisfy the local predicates. Therefore, it reduces the time for transferring data and integrating data from component databases. Moreover, since both algorithms **BL** and **PL** are executed in component databases in parallel, their response time is much shorter than that of algorithm **CA**. Figure 9(b) shows the response time.

  On the other hand, the performance comparison between algorithms **BL** and **PL** is also shown in the figures. Since algorithm **PL** checks the assistant objects before the local predicates are evaluated, more objects need to be checked on the **GOid** mapping table. Moreover, more assistant objects need to be transferred and processed than algorithm **BL**. Although **PL** gains the benefit of parallel processing for phase **O** and phase **P**, the result shows that the benefit does not overcome its overhead. The performance of **BL** is better than **PL** in this situation.

- The number of component databases is adjusted: The total execution time for the three algorithms is shown in Figure 10(a). The ratio of objects which have isomeric objects will increase when the number of component databases increases. For this reason, the number of assistant objects which need to be checked will increase as the number of component databases increases. Moreover, the transfer time gets longer when more component databases transfer data simultaneously. Therefore, for algorithms **BL** and **PL**, the growing rate of the total execution time is higher than that of **CA** when the number of component databases is increasing. The total execution time of **PL** even passes that of **CA** as Figure 10(a) shows. However, the effect of parallel local processing in component databases makes the response time of algorithms **BL** and **PL** still shorter than that of **CA**. Their response time is shown in Figure 10(b).

- The selectivity of one local predicate is adjusted: $N_o^{i,k}$ is set from 1000 to 2000 in this experiment. The total execution time and response time are shown in Figures 11(a) and 11(b), respectively. The varying of the selectivity does not influence the total execution time and response time for algorithm **CA**. However, the times for **BL** and **PL** increase when the selectivity increases. For algorithms **BL** and **PL**, the selectivity decides the number of objects which satisfy the local predicates in component databases. The number of eliminated objects decreases as the selectivity increases. It implies that the time for transferring data and integrating data will increase. Moreover, for algorithm **BL**, the selectivity also influences the number of assistant objects to be checked. Therefore, the growing rate of the times for **BL** is higher than that of **PL** as the selectivity increases.

5 Conclusion

The problem of missing data arises in the distributed heterogeneous object databases because of the missing attribute conflict and the existence of null values. A set of algorithms are provided for query processing when the predicates in global queries involve missing data. The concept of object isomerism is used in the query processing to integrate the local data. For this reason, it is possible for the local maybe results due to missing data to turn into certain results. The centralized approach integrates the data sent from component databases first and then processes the predicates. On the other hand, the localized approaches process the local predicates, and the local results are integrated and certified. Therefore, the localized approaches achieve the inter-site parallelism.

The proposed algorithms are compared and discussed on the total execution time and response time according to the simulation result. Overall, when comparing algorithms **CA**, **BL**, and **PL**, the performance of algorithm **BL** is the best. The number of component databases is a major factor influencing the total execution time of algorithm **PL**. The performance of algorithms **BL** and **PL** gets worse if the selectivity of the local predicates increases. This effect on algorithm **BL** is more than that on **PL**.

In this paper, the predicates in the global queries are assumed to be combined in conjunctive form. The proposed algorithms will be extended in the future to process the global queries containing predicates in disjunctive form. Furthermore, the object signature can also be applied in the design of localized approaches for reducing the amount of data transfer. On the other hand, the global schema may contain multi-valued attributes whose values come from attributes in different component databases. The strategy for processing the global query involving such kind of multi-valued attributes is under investigation.

References

[1] E. Bertino, M. Negri, G. Pelagatti, and L. Sbampinato, Applications of Object-Oriented Technology to the Integration of Heterogeneous Database Systems, *Distributed and Parallel Databases*, 2 (4) (1994) pp.343-370.

[2] A.L.P. Chen, Outerjoin Optimization in Multidatabase Systems, *Proc. IEEE International Symposium on Databases in Parallel and Distributed Systems* (DPDS), 1990.

[3] A.L.P Chen, A Localized Approach to Distributed Query Processing, *Proc. International Conference on Extending Data Base Technology(EDBT)*, 1990.

[4] A.L.P. Chen, J.L. Koh, T.C.T. Kuo, and C.C. Liu, Schema Integration and Query Processing for Multiple Object Databases, *Integrated Computer-Aided Engineering: Special Issue on Multidatabase and Interoperable Systems*, 2 (1) (1995), *John Wiley & Sons.*

[5] A.L.P. Chen, P.S.M. Tsai, and J.L. Koh, Identifying Object Isomerism in Multiple Databases *Distributed and Parallel Databases*, 4 (2) (1996), *Kluwer Academic Publishers.*

[6] M.-S. Chen and P.S. Yu, A Graph Theoretical Approach to Determine a Join Reducer Sequence in Distributed Query Processing, *IEEE Trans. Knowledge and Data Engneering*, 6(1) (1994).

[7] E.F. Codd, Extending the Database Relational Model to Capture More Meaning, *ACM Trans. Database Systems*, 4 (1979) pp.397-434.

[8] L.G. DeMichiel, Resolving Database Incompatibility: An Approach to Performing Relational Operations over Mismatched Domains, *IEEE Trans. Knowledge and Data Engineering*, 1 (4) (1989) pp.485-493.

[9] W. Gotthard, P.C. Lockemann, and A. Neufeld, System-Guilded View Integration for Object-Oriented Databases, *IEEE Trans. Knowledge and Data Engineering*, 4 (1) (1992) pp.1-22.

[10] B.P. Jeng, D. Woelk, W. Kim, and W.L. Lee, Query Processing in Distributed ORION, *MCC Technique Report Number: ACA-ST-035-89*, (1989) pp.1-26.

[11] W. Kent, Solving Domain Mismatch and Schema Mismatch Problems with An Object-Oriented Database Programming Language, *Proc. Seventeenth International Conference on Very Large Data Bases*, (1991) pp.147-160.

[12] W. Kim, I. Choi, S. Gala, and M. Scheevel, On Resolving Schematic Heterogeneity in Multidatabase Systems, *Distributed and Parallel Databases*, 1 (3) (1993) pp.251-279.

[13] J.L. Koh and A.L.P. Chen, Integration of Heterogeneous Object Schemas, *Lecture Notes in Computer Sciences : Entity-Relationship Approach-ER '93*, Vol. 823, Springer-Verlang: Berlin, (1993) pp.297-314.

[14] J.L. Koh and A.L.P. Chen, A Mapping Strategy for Querying Multiple Object Databases with a Global Object Schema, *Proc. IEEE Fifth International Workshop on Research Issues in Data Engineering*, (1995) pp.50-57.

[15] M.P. Reddy, B.E. Prasad, P.G. Reddy, and A. Gupta, A Methodology for Integration of Heterogeneous Databases, *IEEE Trans.' Knowledge and Data Engineering*, 6 (6) (1994) pp.920-933.

[16] A. Sheth and V. Kashyap, So Far (Schematically) yet So Near (Semantically), *Proc. the IFIP Conference on Semantics of Interoperable Database Systems*, Lorne, Australia, Nov. (1993).

[17] S. Spaccapietra and C. Parent, View Integration: A Step Forward in Solving Structural Conflicts, *IEEE Trans. on Knowledge and Data Eng.*, 6(2) (1994) pp.258-274.

parameters	description	default setting
	for each global query	
N_{db}	number of component databases involved	3
N_c	number of global classes involved	$1 \sim 4$
	for each involved global class k	
N_p^k	number of predicates on the class	$0 \sim 3$
R_{ps}^k	selectivity of the predicates on the class	$0.45^{(N_p^k)^{1/2}}$
	for each constituent class of the involved global class k	
R_r^k	ratio of objects to be referenced	$0.5 \sim 1$
R_{iso}^k	ratio of objects having isomeric objects	$1 - 0.9^{(N_{db}-1)}$
	for each constituent class of the involved global class k in database i	
$N_o^{i,k}$	number of objects	$5000 \sim 6000$
$N_{qa}^{i,k}$	number of attributes involved in the subquery	$\max\{N_{pa}^{i,k}, N_{ta}^{i,k}\} \sim (N_{pa}^{i,k} + N_{ta}^{i,k})$
$N_{pa}^{i,k}$	number of attributes involved in the local predicates	$0 \sim N_p^k$
$N_{ta}^{i,k}$	number of attributes which are target attributes in the subquery	$0 \sim 2$
$R_{pps}^{i,k}$	selectivity of the local predicates on the class	$0.45^{(N_{pa}^{i,k})^{1/2}}$
$R_m^{i,k}$	ratio of objects which have missing data	1 if $(N_p^k - N_{pa}^{i,k}) > 0$ $0 \sim 0.2$ otherwise
$R_{as}^{i,k}$	selectivity of the predicates on the assistant objects	$0.55^{(N_p^k - N_{pa}^{i,k})^{1/2}}$
$R_{ss}^{i,k}$	selectivity of the predicates on the signatures of the assistant objects	$0.6^{(N_p^k - N_{pa}^{i,k})^{1/2}}$

Table 2: The database and query parameters

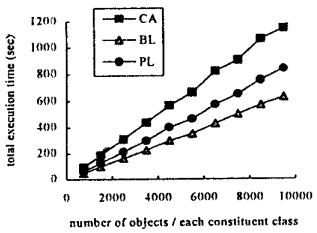

Figure 9 (a)

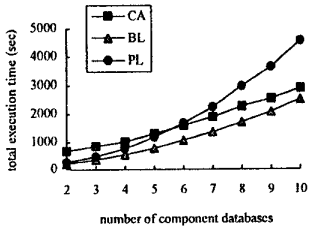

Figure 10 (a)

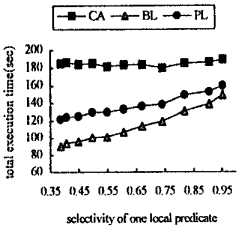

Figure 11 (a)

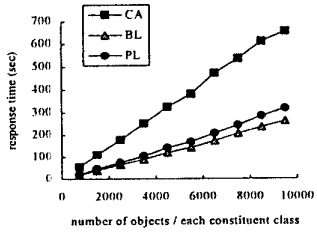

Figure 9 (b)

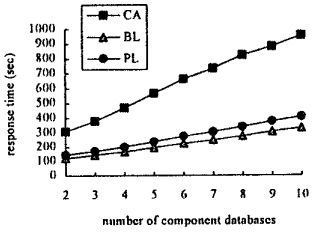

Figure 10 (b)

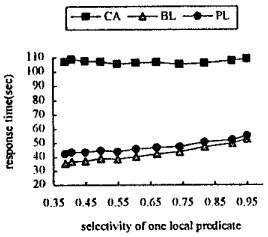

Figure 11 (b)

[18] F.S.C. Tseng, A.L.P. Chen, and W.P. Yang, Answering Heterogeneous Database Queries with Degrees of Uncertainty, *Distributed and Parallel Databases*, 1 (3) (1993) pp.281-302.

[19] UniSQL, Inc., UniSQL/X Database System User's Manual, Release 22.0, Austin, Texas, 1993.

[20] C. Wang, A.L.P Chen, and S.-C. Shyu, A Parallel Execution Method for Minimizing Distributed Query Response Time, *IEEE Trans. Parallel and Distributed Systems*, 3(3) (1992) pp.325-333.