An Aspect of Query Optimization in Multidatabase Systems*
(Extended Abstract)

Chiang Lee *Chia-Jung Chen* *Hongjun Lu*

Database Laboratory
Institue of Information Engineering
National Cheng-kung University
Tainan, Taiwan
leec@dblab.iie.ncku.edu.tw

Department of Information Systems
and Computer Science
National University of Singapore
Singapore
luhj@iscs.nus.sg

Abstract

It is becoming increasingly important to manage databases as a repository resource and to allow application programs to access this resource in a heterogeneous distributed environment. Data requested by a query are sometimes available in multiple sites such that the query becomes executable in different ways. Under this circumstance, choosing the best execution plan [1] becomes an important task for optimizing the query execution. Until today, however, query optimization in multidatabase systems has not been discussed much in the literature. In this work, we utilize the knowledge of *scope relationship* of relations in multidatabases to identify the sites that will return the same results. Then, we propose a novel way of optimizing queries which takes advantage of the conflicts of schemas in searching for the execution plan with the least execution cost. We achieve the goal by first classifying various schema conflicts into different types. The costs of executing the same relational operation on relations of conflicting schemas are evaluated and a weight is assigned to each of the cases to reflect the complexity of executing the operation. As this method only involves simple iterative computations of the weights and the saving of a query execution time can be dramatic, the method developed here can be regarded as an effective way of optimizing query processing in a multidatabase environment.

1 Introdcution

It is becoming increasingly important to manage databases as a repository resource and allow application programs to access this resource in heterogeneous distributed environment. A multidatabase system is a system to meet this goal by providing uniform and integrated access to a distributed collection of existing databases[ACM90, She90, COM91, REC91, IMS91, IMS93].

A multidatabase system consists of a number of participating local database systems. In most cases, those local database systems store semantically related data. For instance, Figure 1 shows a university multidatabase system whose participating databases are from different departments, registrar's office, student union, etc. Each database has some information about students and teachers. An integrated schema (IS) that contains all the information of local schemas is also given in the figure. Note that the global schema can be integrated by using any existing method. Our optimization method is not affected by a different IS. We assume that in this integrated system a global query is issued against the IS. For simplicity, the schemas of all databases are represented in relational notations.

Assume that all students have to register at the registrar's office and all registered students are automatically members of the student union. Then, we can infer that the real-world entities referenced by tuples in *ROD.S_Female* and *ROD.S_Male* are equivalent to those referenced by *SUD.Student*. (For convenience, we use *DB.R* to denote the relation *R* of database *DB*.) In this case, we say that the *scopes* of these two relations are equivalent. A similar definition on the scope of a relation can also be found in [Wha92]. If some faculties of the EE department are also joint faculties of the CS department, then the scopes of the teacher relations of these two databases are overlapped. Other relationships are that, for example, the scope of the $CSD.Student$ relation is contained in the scope of the $SUD.Student$ relation, and contained in the union of $ROD.S\_Female$ and $ROD.S\_Male$, and the scope of $ROD.Student$ is equivalent to the union of the scope of $CSD.Student$ and the scope of $EED.Student$ (if CSD and EED are all the departments that the university has). Note

*This work was supported by the National Science Council under Grand# NSC82-0408-E-006-418

[1]By execution plan, we mean a description of relations and attributes of the databases to be accessed as well as the operations performed on these data in order to obtain the query result. The details of how to perform an operation and the schedule of executing the operations, as in traditional sense, are not the concern of this paper.

Registrar's Office Database (ROD)
S_Female(s#, name, GPA, address, advisor#)
S_Male(s#, name, GPA, address, advisor#)
T_Female(t#, name, office)
T_Male(t#, name, office)

Computer Science Dept. Database (CSD)
Student(s#, name, sex, address, advisor#)
Teacher(t#, name, sex, office)
Address(t#, street, city, state)

Electrical Engineering Dept. Database (EED)
Student(s#, name, female, male, address, advisor#)
Teacher(t#, name, office)

Student Union Database (SUD)
Student(s#, name, sex, advisor#)
Address(s#, street, city, state)
Teacher(t#, name, sex, office)

An Integrated Schema (IS)
Student(s#, name, sex, GPA, street, city, state, advisor#)
Teacher(t#, name, sex, office, address)

Figure 1 The schemas of databases in a university.

that the global schema can be integrated by using any existing method. Our optimization method is not affected by a different *IS*.

If the attributes inquired by a query are common to some databases, then there can be multiple ways to get the result from the databases. For instance, a global query is to find the names and addresses of all students. Then, the query can be processed in at least the following manners:

- Get all results from *SUD* in which a join of *SUD.Student* and *SUD.Address* is required (i.e., $SUD.Student \bowtie SUD.Address$).
- Get all results from *ROD* in which a union of the results of two projection queries is incurred (i.e., $(\Pi_{name,address} ROD.S\_Female) \cup (\Pi_{name,address} ROD.S\_Male)$).
- Get the union of the result of *CSD* and the result of *EED* (i.e., $(\Pi_{name,address} CSD.Student) \cup (\Pi_{name,address} EED.Student)$).

If the data are consistent in all databases (that is, the information of an entity is independent of the site where the data is stored), the results obtained in the above cases are the same. The costs of executing the query based on these execution plans are different, however, as indicated by the relational algebraic expressions in the paratheses of the above execution plans. The difference is caused by the representational conflicts of the data in these databases. Hence, choosing one execution plan that incurs the least processing cost becomes an important task of query optimization in such a multidatabase environment. This task is nontrivial while a query can be processed in multiple databases and various types of schema conflicts exist between these databases.

In the past, only a few papers discussing the optimization of query processing at the physical level were proposed [Du92, Lu93, Zhu94]. Their goal is to estimate the processing time for relational operations (mainly join) by figuring out the cost model of a given DBMS. Whether the join is performed by a hash, a sort-merge, or a nested loop algorithm, and whether a relation is indexed/clustered are the key factors considered in their research. Based on the estimated cost models of the participating databases in a multidatabases environment, the MDBS is able to determine the most suitable DBMS(s) to process the query. Comparing to their issues, the optimization issues studied here are at the semantic level. A query is optimized in our work by considering the schemas of the databases. The databases whose data representations best *match* the query in semantics are selected to process the query. To the best of our knowledge, there is not a similar work proposed in the literature. A systematic optimization method needs to be designed to resolve this new problem.

The organization of the extanded abstract is as follows. In Section 2, we classify the types of schema conflicts and describe the principle of our method. An example is used for illustration. In Section 3, we briefly conclude the paper.

2 The principle of our method

As demonstrated in the above example, a global query is translated into different forms based on different local schemas. In this section, we first introduce our classification of the types of schema conflicts. This classification is similar to those presented previously [Kim91, Lit91]. Then, the relationships between the types of conflicts and the query translation are discussed. Based on these discussions, our method is developed. The types of conflicts between databases are categorized as follows.

- Value-to-Value conflicts

  These conflicts occur when databases use different representations for the same data. There are three different aspects as to the representations of data: expressions, units, and precision. Examples of these conflicts are US dollars versus Japanese Yen, a score of 1 to 100 versus A to E, etc. Since our focus in this paper is to utilize the conflicts between the schemas of local databases for the optimization of query processing, this type of conflicts is not considered in the later optimization process.

- Value-to-Attribute conflicts

  These conflicts occur when the same information is expressed as values in one database and as an attribute(s) in another database. For example, the values of the attribute *sex* of *CSD.Student* are represented as attributes (*female* and *male*) in *EED.Student.* This type of conflicts is called a value-to-attribute conflict.

- Value-to-Table conflicts

  These conflicts occur when the attribute values in one database are expressed as tables in another database. For example, Figure 1 shows the relation schemas of *S_Female* and *S_Male* for female and male students, respectively. The same information is represented as values of *sex* of other databases. We refer to this type of conflicts as the value-to-table conflict.

- Attribute-to-Attribute conflicts

  These conflicts are caused by using different definitions for the semantically equivalent attributes in different databases. For instance, the *address* is one attribute in *CSD.Student.* It is however represented by three attributes *street*, *city*, and *state* in *IS.Student.* This type of conflicts is referred to as the attribute-to-attribute conflict.

- Attribute-to-Table conflicts

  These conflicts occur if an attribute of a database is represented as a table in another database. For example, *address* is an attribute in *CSD.Student.* It is however represented as a relation *Address* in *SUD.* This type of conflicts is termed the attribute-to-table conflict.

- Table-to-Table conflicts

  These conflicts are caused by representing the information of a set of semantically equivalent tables in a different number of tables in another databases. For example, the *IS.Student* has a table-to-table conflict with the *SUD.Student* and *SUD.Address.*

For an instance, a global query issued against the *IS* is to retrieve the name(s) of the student(s) who is a female and lives in the address "#1, Da-Shieh Road, Tainan, Taiwan". The SQL of the query is as follows.

```
select   name
from     Student
where    sex="female"
and      street="#1, Da-Shieh Road"
and      city="Tainan"
and      state="Taiwan"
```

To process this query in *ROD*, it is clear that the selection condition (sex="female") is not needed. The save of a selection operation is caused by the value-to-table conflict between the values of *sex* of the *IS.Student* and the tables *S_Female* and *S_Male* of *ROD.* In addition to the above difference, the selection conditions on *street, city,* and *state* are also combined as one condition on *address.* This change is caused by the attribute-to-attribute conflict between the databases. If this operation is performed in *SUD*, the selection condition on *sex* need not be changed as there is no conflict between the *sex* of *IS.Student* and the *sex* of *SUD.Student.* A join is incurred, however, in the *SUD* between the *Student* and the *Address* relations. This is due to an attribute-to-table conflict between the attributes $\{street, city, state\}$ of the *IS.Student* and the table *SUD.Address.*

From this discussion, we can envision that the type of added operations during translating a global query into a local query is related to the type of conflict between the integrated schema and the local schema. Not only selections and joins can be incurred, other operations such as projections and set operations can also be incurred in other types of schema conflicts. Operations can also be eliminated during the translation. It usually occurs when the local schema is expressed in a more "compact" way than the *IS.* A detailed study of the types of operations that can be added/removed during query translation has been conducted and the result is reported in [Lee94, Che94]. It is interesting that each type of schema conflicts results in a change of a specific type(s) of operations during the translation [Lee94]. We will illustrate the idea shortly by using some examples. As the schemas of databases are static information (irrelevant to the user query) and their differences can be compared prior to issuing a query, the site that gives the least execution cost can be found once the relations and attributes involved in the query are known. The translations of the global query into various forms of local queries based on the local schemas are in fact not needed.

The idea of our method is as follows. For each attribute $\mathcal{A}$ of the integrated schema, we use a weight to represent the extent of difficulty of performing in a local database a relational operation on the data corresponding to $\mathcal{A}$. These weights are retained in a *weight table.* An example weight table of the databases given in Figure 1 is shown in Figure 2. S, P, and J in the table represent the operations selection, projection, and join, respectively. For simplicity, the weights of the other operations are not shown. A weight of 0 in the entry $(DB_i, op_j, attr_k)$ indicates that the cost of performing the operation op_j on the data in DB_i corresponding to the attribute $attr_k$ of the *IS* is the same as performing the operation op_j on $attr_k$ of *IS.* A negative weight indicates that the cost of performing the corresponding local operation(s) in the local database is higher than the cost of performing the global operation. The greater the difference of the costs, the lower the weight. A positive weight indicates that the operation (op_j) is avoided if the query is executed in the local database DB_i, that is, a saving of an operation execution is obtained. An *a/b* (such as 0/-5) in an entry means that either *a* or *b* is chosen. The one to be chosen is determined by the rule:

Weights		ROD			CSD			EED			SUD		
		S	P	J	S	P	J	S	P	J	S	P	J
Student	s#	0	0	0	0	0	0	0	0	0	0	0	0
	name	0	0	0	0	0	0	0	0	0	0	0	0
	sex	+0.5	0	-10	0	0	0	0	0	-5	0	0	0
	GPA	0	0	0	X	X	X	X	X	X	X	X	X
	street	X	X	X	X	X	X	X	X	X	0/-5	0/-5	0/-5
	city	X	X	X	X	X	X	X	X	X	0/-5	0/-5	0/-5
	state	X	X	X	X	X	X	X	X	X	0/-5	0/-5	0/-5
	advisor#	0	0	0	0	0	0	0	0	0	0	0	0
Teacher	t#	0	0	0	0	0	0	0	0	0	0	0	0
	name	0	0	0	0	0	0	0	0	0	0	0	0
	sex	+0.5	0	-10	0	0	0	X	X	X	0	0	0
	address	X	X	X	X	0/-5	0/-5	X	X	X	X	X	X

Figure 2 A weight table for Figure 1.

If the queried attributes that are from the same global relation belong to the same local relation in a certain local database,
then a is chosen if the query is executed in this local database;
otherwise (the queried attributes are dispersed in two local relations), b is chosen.

Note that in this particular example a is always 0. It may be a different value in other cases. As the cost of executing a join (as well as a set) operation is greater than that of executing a selection (and projection) operation, we assume that the absolute weight of incurring/avoiding a join operation is larger than that of a selection operation. For illustration purpose, we simply assign 0.5 to a selection (projection) operation, and 5 to a join (set) operation. An exact value of the weight is dependent on factors such as processing strategies (hash join, sort-merge join, nested-loop join, etc.) for each relational operation of a DBMS. Since it is an issue independent of our concern in this extended abstract, the discussion is omitted here.

We use some examples to illustrate how the weights in the table are obtained.

Example 1
First, we see that the entry at (ROD, S, sex) is 0.5. The reason is explained as follows. In a value-to-table conflict, a selection query such as

```
select   name
from     Student
where    sex = 'female'
```

is translated to the following query if it is to be executed in ROD:

```
select   name
from     S_Female
```

Since a selection is avoided while executing the query in ROD, the weight is assigned to be +0.5.

Example 2
The entry at (ROD, J, sex) is -10. Let us use another example to explain the reason. A join query as follows

```
select   Student.name, Teacher.name
from     Student, Teacher
where    Student.sex=Teacher.sex
```

is issued against the IS. To process the query in ROD whose student and teacher tables have a value-to-table conflict with the sex attribute of the IS, the query is translated to

```
select   S_Female.name, T_Female.name
from     S_Female, T_Female
union
select   S_Male.name, T_Male.name
from     S_Male, T_Male
```

Note that there are two relations shown in the **select** clause without having a **where** clause. This indicates that a cross product is performed on the two relations. As the translated query involves two cross product operations and a union operation (totally three set operations versus one join operation in the original global query), the weight at this entry is assigned to be -5*2=-10.

Example 3
In this example, we see how the weight 0/-5 is assigned to the entry $(CSD, P, address)$. Assume that a global query is as follows.

```
select   address
from     Teacher
```

If it is to be executed in *CSD*, it is translated to

```
select   street, city, state
from     Address
```

As the projections in these two queries are both on one relation, the complexities of executing these two queries are about the same. Hence, the weight in this case is 0.

On the other hand, if a global query is as follows,

```
select   name, address
from     Teacher
```

then in order to be executed in *CSD* it is translated to

```
select   name, street, city, state
from     Teacher, Address
where    Teacher.t#=Address.t#.
```

In this case, an additional join must be involved (because the queried attributes belonging to the same global relation are from two local relations of the *CSD*). Hence, the weight is -5. In summary, the weight at the entry $(CSD, P, address)$ is 0/-5.

The weights in the other entries can be derived similarly based on the above discussion. An 'X' in the weight table indicates that either a weight is not applicable to that entry (because of a lack of the attribute in the local database, for example), or the operation is not executable for the data corresponding to the indicated attribute of the indicatetd database. For instance, as *GPA* is not an attribute of the student relation in *CSD*, *EED*, and *SUD*, the entries for those databases on the row *GPA* are 'X'. As for the 'X' in the entries of the rows *street*, *city*, and *state* at the S, P, and J columns of the *ROD*, *CSD*, and *EED* databases, respectively, is because the attribute *address* of those databases has an attribute-to-attribute conflict with the *street*, *city*, and *state* of *IS*. Such a conflict causes the operations S, P, and J on these attributes (e.g., select city="New York") unable to be executed on *address* in *ROD*, *CSD*, and *EED* as DBMSs do not have the intelligence to recognize whether a string, such as "New York", in an address string is a city, a state, or even a street name. (If some DBMSs do have the capability, then the weight in the weight table should be adjusted accordingly.)

The total weight of executing a global query in a local database is the sum of the weights of the attributes invovled in the global query. If multiple databases are involved in a global query, the total weight is the sum of the weights corresponding to all subqueries of the global query. Based on the weights, the optimization of global query processing at the schema level is formulated as follows.

Given the schemas of all databases and a global query, the query could be executed in multiple ways; each of the ways will return the equivalent result. Assume that ξ represents the set of execution plans to execute the query and e is one of them (i.e., $e \in \xi$). The set of databases involved in the execution plan e is DB_e. The global query is issued against the integrated schema (IS) and relations $R_1, \ldots, R_r$ of IS are accessed. Within a relation R_i, the attributes $attr_1, \ldots, attr_{a_i}$ are accessed. The operation performed on $R_i.attr_j$ is $op_{i,j}$. Assume that the weight of performing in the k-th database, db_k, the operation translated from $op_{i,j}$ on $R_i.attr_j$ of IS (i.e., the value at the entry $(db_k, op_{i.j}, R_{i,attr_j})$) is denoted as $\omega^{db_k}_{R_i.attr_j,op_{i,j}}$. Our target is to find $e' \in \xi$ such that

$$Weight(e') = \max_{\forall e \in \xi} \sum_{\forall db_k \in DB_e} \sum_{\forall R_i} \sum_{\forall attr_j} \omega^{db_k}_{R_i.attr_j,op_{i,j}}$$

where $Weight(\cdot)$ denotes the total weight of an execution plan.

This optimization process indicates that the plan causing the maximum total weight is chosen as the query execution plan. This plan is guaranteed to use the least number of costly relational operations. As the weights can be obtained by looking up the weight table, this optimization process only requires simple iterative computations. The cost is therefore insignificant comparing to that of executing a global query.

3 Conclusions

In a multidatabases environment, the techniques required for processing a query is quite different from those in a single database system. However, these issues have not attracted much attention from researchers in the past. In this work, we presented the idea of utilizing the information of heterogeneity among databases for query query optimization. We first presented in this extanded abstract the issues by using a university databases example. The priciple of our method is then introduced. Our basic idea is first to analyze the differences between the local schemas and the integrated schema (*IS*). Based on these differences, we assign weights to the entries in the weight table. When a global query is issued, we can optimize the query by computing the total weights of each execution plan. The plan that incurs the maximum total weight will be chosen as the optimal execution plan. Currently, we are developing a complete algorithm which considers the effect of all types of schema conflicts and all operations for optimizing global queries. A more accurate weight assignment policy that considers other factors such as the join algorithms and their costs is also under design. The finished work will be used as a module in a multidatabase query compiler developed in our laboratory.

References

[ACM90] ACM Computing Surveys, A Special Issue on Heterogeneous Database, vol.22, no.3, Sept. 1990.

[Ahm91] R. Ahmed, P.D. Smedt, W. Du, W. Kent, M.A. Ketbchi, W.A. Litwin, A. Rafii, and M.C. Shan, " The Pegasus Heterogeneous Multidatabase System", IEEE Computer, Vol. 24, No. 12, December 1991.

[Bre86] Breitbart, Y., P. Olson and G. Thompson, "Database integration in a distributed heterogeneous database system," Proc. IEEE International Conference on Data Engineering, 1986.

[Cha91] A. Chatterjee, and A. Segev, "Data Manipulation in Heterogeneous Databases", SIGMOD RECORD, Vol. 20, No. 4, December 1991.

[Che94] Chia-Jung Chen,"Reduction of Access Sites for Query Optimization in Multidatabase Systems," MS. Thesis, Institute of Information Engineering, National Cheng-Kung University, Tainan, Taiwan, July 1994.

[COM91] IEEE Computer, A Special Issue on Heterogeneous Distributed Database Systems, vol.24, no.12, Dec. 1991.

[Du92] W. Du, R. Krishnamurthy, and M. Shan, "Query Optimization in Heterogeneous DBMS", Proc. of VLDB, 1992.

[IMS91] The Proceedings of the First International Workshop on Interoperability in Multidatabase Systems, Ed. Y. Kambayashi, M. Rusinkiewicz, and A. Sheth, Kyoto, Japan, 1991.

[IMS93] The Proceedings of the International Workshop on Interoperability in Multidatabase Systems, Ed. H.-J. Schek, A. P. Sheth, and B. D. Czejdo, Vienna, Austria, 1993.

[Kim91] Won Kim and Jungyum Seo, "Classify Schematic and Data Heterogeneity in Multidatabase systems", IEEE Computer, Vol. 24, No. 12, December 1991.

[Kri91] Ravi Krishnamurthy, Witold Litwin, and William Kent," Language Features for Interoperability of Databases with Schematic Discrepancies", ACM SIGMOD Conference on Management of Data, 1991.

[Lee94] Chiang Lee and Chia-Jung Chen,"Reduction of Access Sites for Query Optimization in Multidatabase Systems," Technical Report TR-83-700, Institute of Information Engineering, National Cheng-Kung University, 1994.

[Lit90] W. Litwin, Leo Mark, and N. Roussopoulos, "Interoperability of Multiple Autonomous Databases", ACM Computing Surveys, Vol 22, Number 3, September 1990.

[Lit91] W. Litwin, M. Ketabchi, and R. Krishnamurthy, "First Order Normal Form for Relational Databases and Multidatabases", ACM SIGMOD RECORD, Vol. 20, No. 4, December 1991.

[Lu93] H. Lu, B.-c. Ooi, and C.-H. Goh, "Multidatabase Query Optimization: Issues and Solutions", IMS, 1993.

[REC91] ACM SIGMOD RECORD, A Special Issue on Semantic Issues in Multidatabase Systems, vol.20, no.4, Dec. 1991.

[She90] Amit Sheth and James A. Larson, "Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomus Databases", ACM Computing Surveys, Vol.22, Number 3, September 1990.

[Wha92] W. K. Whang, S. Chakravarthy, and S. B. Navathe," Heterogeneous Database: Inferring Relationships for Merging Component Schemas, and a Query Language", Tech. Report UF-CIS-TR-92-048, Department of Computer and Information Sciences, University of Florida.

[Zhu94] Q. Zhu and P.-A. Larson,"A Query Sampling Method for Estimating Local Cost Parameters in a Multidatabase System," Proceedings of the Data Engineering Conference, 1994, pp.144-153.