Translation of queries to account for direct communication between different DBMSs

by MEHDI OWRANG
American University
Washington, D.C.

and

L.L. MILLER
Iowa State University
Ames, Iowa

ABSTRACT

A translation process designed to translate queries between data models is examined. In particular, this paper concentrates on translating queries between the relational and network data models. Such a translation mechanism will enable the user to have access to the database resource in a distributed database for which different database management systems coexist. The translation process is described and examples illustrating the process are given. The discussion has been limited to operating in the environment where the source and target databases have the same semantics. Hypergraphs are used as the intermediate representation format for the query during the translation process. The semantics of both the source and target databases are described in terms of hypergraphs. In addition, the hypergraph is extended to incorporate the operations of the data manipulation language in order to define a general model for query translation. The quality of the translated query is examined.
INTRODUCTION

The problem of translation is not new. Each time a new generation of computer systems evolves, data from the old database must be regenerated and the old query set must be translated. In addition, there is a need for translation when we restructure the database. Such restructuring can be motivated by

1. A change of the use environment of a database management system (DBMS).
2. A change in DBMS.
3. Modification of the design for efficiency reasons.
4. A change in database semantics.

Su and Reynolds\(^6\) studied the problem of high-level sublanguage query conversion using the relational model\(^7\) with SEQUEL\(^3\) as the sublanguage, DEFINE\(^4\) as the data description language and CONVERT\(^5\) as the data translation language. They examined sublanguage query conversion where query modification is due to changes in the schema and sub-schemata. The changes they considered were

1. A large relation split into several relations.
2. Several relations combined into one.
3. Changes of the mathematical mapping between/among entities.
4. Adding or deleting entities and/or associations.

Their conversion algorithm is specific to the environment they studied and serves pedagogically when an attempt is made to extend their work into the general translation problem.

Other researchers, Katz and Wong,\(^8\) have studied the program conversion due to the changes in application programs that are caused by converting between database systems that support a different level of procedurality in the data sublanguage. In particular, they present an algorithm for mapping from the procedural operations of CODASYL DML\(^7\) into the nonprocedural relational calculus.\(^8\) Their mapping algorithm is restricted to only the conversion from CODASYL DML to relational calculus and therefore does not introduce a general model for translation.

The advent of lower-cost computer systems has paved the way for decentralization of computing resources. Decentralization has enabled designers to enhance local performance by allowing the freedom of design and software choice for each local system. Such freedom provides the necessary enhancement of local computing, but complicates the task of providing access to the database resource on a system-wide basis. In particular, it either forces the user to be aware of the location and format of all of the data in the system, or the computer software/hardware must have the ability to provide access to the local databases through a single data manipulation language. Systems, such as MULTIBASE\(^9\) provide such access but force the user to use both the data manipulation language (DML) of MULTIBASE and the local system to get the enhanced local performance.

What is needed is a model that can provide both the dynamic query translation necessary to allow the direct communication of different DBMS so the user only needs to be proficient in the DML of his/her local DBMS and the ability to provide general query translation to allow the extension of Su and Reynolds\(^1\) work. In recent work on database design, a number of authors\(^10-14\) have found the hypergraph to be a useful means of modeling relational database designs. When the hypergraph is extended to incorporate the DML operations, it is helpful in defining a general model for query translation.

In the present work, the space limitation forces us to examine the topic in narrower terms. In particular, we will concentrate on translating queries where the source hypergraph attempts to model the same semantics as the target hypergraph (as in dynamic query translation). We also restrict our discussion to the relational and network data models, although similar results are available for the hierarchical model as well.\(^15,16\)

REPRESENTATION IN HYPERGRAPH

This section examines the use of hypergraphs to model both the logical designs and the DML operations of the relational and network data models. A great deal has been written in recent years about the use of hypergraphs in representing the logical design of database schemes,\(^10-14\) and we assume that the reader is familiar with hypergraph concepts to the level of Fagin, Mendelzon, and Ullman.\(^13\) To be of use in translating queries, the hypergraph model has to be expanded to incorporate the operations of the DML. In the remainder of this section, the hypergraph representations for the relational and network data models are described.

Relational Data Model and Its Hypergraph Representation

Fagin, Mendelzon, and Ullman\(^13\) use the hypergraph to model the full join dependency which defines the universal relation (UR). For example, the well known supplier-parts database is defined by the dependency set \(\{\rightarrow_{C}[R_1, R_2, R_3], \ P\rightarrow C, \ C\rightarrow S, \ S\#P\rightarrow Q\}\), where \(R_1(S\#, P\#, Q), R_2(S\#, C), \) and \(R_3(C, S)\). The join dependency \((jd)\rightarrow_{C}[R_1, R_2, R_3] \) can be represented by the
hypergraph of Figure 1, where the nodes of the hypergraph are the attributes of the relation schemes and the edges are the relation schemes. Fagin, Mendelzon, and Ullman have shown that the semantics of any database can always be represented by such a full join dependency and a set of functional dependencies (fd). Ullman in his survey of universal relation assumptions denotes this as the URJD assumption.

In the following, we incorporate the DML commands of relational algebra into the model to fit the needs of the translation process. The inclusion of the three fundamental operators of relational algebra—join, project, and select—in the hypergraph model is described by the following set of actions.

**Join:** The natural join is introduced into the model by the creation of a new edge containing the combined attribute set of the two joined relations. The new edge is labeled as a join edge. To consider the more general question of the theta join requires an extension of the edge labeling process to describe the restrictions on theta.

**Project:** Projection is introduced into the hypergraph by inclusion of a new edge containing the projected attributes. The new edge is labeled as a projection edge.

**Select:** A selection operation on a relation creates a new relation of tuples which have been defined by the given condition. The condition involves a boolean combination of simple conditions. Simple conditions or combinations of conjunctive simple conditions can be indicated in the hypergraph by simply labeling the appropriate attribute nodes with the condition. Conditions formed by the disjunction of simple conditions require the insertion of a new edge which is labeled as an "OR" edge. The concept of projection (choosing data values from a record) is included in the hypergraph by using SELECT FROM record-name: list. The concept is introduced into the hypergraph by enclosing the relevant nodes in a projection edge.

The inclusion of an OUT OF THE BLUE FIND and RELATIVE FIND in the hypergraph model is implied by two paths—the positioning path and the answering path. The positioning path contains all of the selection criteria that is required to establish a starting position in the network database. A positioning path implies an OUT OF THE BLUE FIND operation and possibly a RELATIVE FIND if there is more than one in the positioning path.

An answering path contains all of the selection criteria and projection information required to generate the result of the query. To process the answering path, three paths are examined—the common path, the selection path, and the projection path. The common path consists of the nodes (owner or member) that are common to both the selection path and projection paths. This path is used to establish the initial position for processing the selection and projection paths. The common path will always imply a RELATIVE FIND operation.

A selection path contains all of the selection criteria required to satisfy the query. It implies a RELATIVE FIND operation. The projection path consists of all the projection fields and the required selection criteria for the records that make up the projection path. It implies a special RELATIVE FIND (FIND FIRST, an operation to allow backing up to the beginning of the named record type for the current path) and a RELATIVE FIND operation. (Example 2 illustrates the inclusion of a network DML query into the hypergraph model.)

In the next section the hypergraph models are used as the basis of a translation process between the two database models.
Translation of Queries to Account for Direct Communication Between Different DBMSs

QUERY TRANSLATION

We assume that we have two database designs. The source design attempts to retain the same semantics as the target design although they are supported by different database management systems and may be based on different data models. The data is stored in the format required by the target design, but the user views the data as though it were stored in the source design. As noted in the introduction, our motivation for examining this problem is to provide a local user in a distributed database environment with the ability to work with only one query language and to allow him/her to access the local DBMS directly.

The two designs are represented in the hypergraph format described in the previous section. To translate the query, we require three operations:

1. Map the source query into the hypergraph space of the source design.
2. Translate the resulting source query hypergraph into the hypergraph space of the target hypergraph.
3. Map the target query hypergraph to the target data manipulation language.

In the first operation we have the task of creating a source query hypergraph that has sufficient information content to provide a basis for translation. We have found four types of information to be useful. Naturally, the set of attributes that represents the result of the source query and the attributes used in search conditions are essential. In addition, we include information used in the network model to establish a position in the database. The hierarchical model operates in a similar manner.

The last type of information taken from the source query is navigational information, useful in navigating the target hypergraph. The path used in the source query is passed with the source query hypergraph. In target hypergraphs with ambiguous path selection, the source query path can be used to choose the correct path. For example, if the target database is relational and the hypergraph contains a cycle, we may have two paths connecting the relevant attributes. The two paths have somewhat different semantics which can be resolved in the translation process by falling back and using the user's interpretation of the semantics from the source query path. A similar approach is taken for navigation in the network model.

The format used to pass the four types of information to the translation process is a three edge hypergraph, and the path information passed as a set. The first edge of the source query hypergraph \( (P) \) is the set of attributes used to establish position. The second edge \( (C) \) is the set of attributes used in the search conditions, and the conditions are used to label the attributes in the manner described earlier. The final edge \( (R) \) is the set of attributes which represents the result of the source query. Examples 1 and 2 provide the source query hypergraphs for the network and relational queries, respectively. Note that the position edge \( (P) \) is always empty for relational source queries.

The translation process makes use of the source query hypergraph, the source path set, and the target hypergraph to...

EXAMPLE 1

Translation of a source query written for the network design of Figure 4 to the target query for the relational design of Figure 3. Query: Get the names of all students who made a grade of A in all offerings of 1985 CS courses after 2500.

Network Source Query:
FIND Course Record Within SS-Course SET where Course# = '2500'
Skip if fail
Repeat
FIND NEXT Course Record Within SS-Course SET where dept = 'CS'
Exit if status-check
Repeat
FIND NEXT Offering Record Within HASOFFER SET where date = '1985'
Exit if status-check
Repeat
FIND NEXT Student Record Within HASSTU SET where grade = 'A'
Exit if status-check
Select From Student: sname
end
end
end.

Source Query Hypergraph \( (Q_s): \)

\[
P:2500 \\ C:\text{CS} \quad \text{A} \quad \text{1985} \\ R: \text{sname}
\]

Target Query Hypergraph \( (Q_t): \)

Note: The value P:2500 denotes that the label (2500) is used in the position edge \( (P) \).

Target Query:
Select Course where dept = 'CS' giving \( T_1 \)
Select Offering where date = '1985' giving \( T_1 \)
Join \( T_1 \) and \( T_1 \) giving \( T_1 \)
Select Student where grade = 'A' giving \( T_1 \)
Join \( T_1 \) and \( T_1 \) giving \( T_1 \)
Find partition of \( T_1 \) where Course# \( \geq \) 2500 giving \( T_1 \)
Project \( T_1 \) over sname giving Result.
translate the source query into the target query hypergraph. When the source and target database designs have the same semantics (as in dynamic query processing), the source path set can be used directly to determine the path in the target hypergraph. For the case in which the target data model is relational, the source path is used to determine the join sequence required to incorporate the attributes used in the selection conditions and the projection attributes in the same relation. We assume that the join sequence produced is lossless. There are several methods that exist in the literature for the generation of a lossless join sequence, for example, $\gamma$-hypergraph,\textsuperscript{12} hinges,\textsuperscript{19} and maximal objects,\textsuperscript{14} and we will assume that one of the techniques is used in the remainder of this work. The edges forming the join set are used as the initial edges of the target query hypergraph. Join edges are added to the target query hypergraph to imply the order in which the join operations will take place. As a final step the labels from the nodes in the source query hypergraph are copied over to the nodes in the target query hypergraph. Labels from the position edge of the source query hypergraph (if any) are marked as position labels before they are copied to the target query hypergraph. The concept of positioning is handled in the relational environment in one of two ways. In cases where the positioning is such that it can be implemented by simply using the condition on a selection, relational algebra operators are sufficient to answer the query. In more complex cases, we may require a sort of the resulting tuples followed by a partition of the sorted set.

Example 1 illustrates the process of translating a query from a network DML to the relational environment. The mapping of the target query hypergraph into the appropriate DML is naturally language dependent. The target query hypergraph for the relational environment uses relational algebra as its basis to allow the mapping algorithm to be basically a translation from relational algebra to the appropriate DML.

Translation from the source query hypergraph into the environment of a target database in the network format requires the creation of a target query hypergraph made up of a positioning path and an answering path. The source query path set is used by the translation process to determine the segments required in the target query hypergraph. The position edge ($P$) is used in conjunction with the path information to create the positioning path. The selection and projection edges of the source query hypergraph, $C$ and $R$ respectively, are used by the translation algorithms to create the answering path required to generate the target query in the network DML. To enhance the process of generating the target query, the answering path is processed to create the common parent path, selection path, and projection path as described before.

To simplify our discussion in this presentation we make two assumptions concerning the nature of the source query. First we assume that the translation process considers a class of queries which are tree queries (or acyclic queries)	extsuperscript{95}—queries which do not have a cycle. We also assume that network source queries look at the data from some starting point (position) in the database and continue until the end of the database is reached. Such an assumption can be removed in the general case but requires unnecessary complexity in this discussion.

### Example 2

Example 2: Translation of a source query written for the relational design of Figure 3 to the target query for the network design of Figure 4. Query: Get the name of all students who make a grade of A in all offering CS courses. For all CS courses.

**Relational Source Query:**

Join Course and Offering giving $T_1$

Join Student and $T_1$ giving $T_2$

Select $T_2$ where dept='CS' and (offer#='2' OR location='Washington') giving $T_3$

Project $T_3$ over sname giving Result.

**Source Query Hypergraph ($Q_s$):**

$P = \{ name \}$

**Target Query Hypergraph ($Q_T$):**

**Target Query Answering Path:**

FIND Course Record Within SS-Course SET

Skip if fail

Repeat /* process common path */

FIND NEXT Course Record Within SS-Course SET where dept='CS'

Exit if status=check

Repeat

FIND NEXT Offering Record Within HASSOFRER SET where offer#='2' OR location='Washington'

Exit if status=check

/* process selection path */

FIND NEXT Teacher Record Within HASTEA SET where sex='male'

Skip if fail

FIND NEXT Student Record Within HASSSTU SET where grade='A'

Skip if fail

/* process projection path */

FIND FIRST Student Record Within HASSTU SET

Repeat

FIND NEXT Student Record Within HASSTU SET where grade='A'

Exit if status=check

Select From Student: sname

Exit

end.
Example 2 illustrates the translation process, where the source query is written for the relational design of Figure 3, and the target design is the network design shown in Figure 4.

CONCLUSION

A process for translation of queries between the network and relational data models has been presented and shown to produce valid queries. The translation scheme uses the hypergraph as an intermediate representation format between the two DMLs. The translation process requires three steps:

1. Map the source query into the hypergraph space of the source design.
2. Translate the resulting source query hypergraph into the hypergraph space of the target hypergraph.
3. Map the target query hypergraph to the target manipulation language.

The second step is the heart of the translation process and is model dependent, but not DML dependent. Steps one and three are language dependent, but based on the algorithms that we have developed to this point are not difficult to implement.

The work presented here represents the basic description of the translation process as it relates to the relational and network data models. Space limitations keep us from providing the algorithms used to create the structures given in Examples 1 and 2 (see Owrang for details). Currently we are working on expansion of the system to handle the cyclic queries, and work has been started on implementing the translation system. Note that the discussion has been limited to operating in an environment in which the source and target databases have the same semantics. The algorithms used in this translation process can be easily expanded to handle the case in which some changes have been made in the semantics, such as in the case in the work by Su and Reynolds.

REFERENCES


Figure 3—Hypergraph representation of relational course offering database

Figure 4—Hypergraph representation of network course offering database