DESIGN AND IMPLEMENTATION OF AN ER QUERY AND UPDATE INTERFACE

Bogdan Czejdo
Department of Mathematical Sciences
Loyola University
New Orleans, LA 70118

Abstract

In this paper we discuss a method for implementing an Entity-Relationship (ER) query and update interface called ERQUI. The approach is based on using a relational database management system as a major component of the ERQUI system architecture. During the process of manipulation of ER diagrams the information needed to generate an equivalent SQL query or update is accumulated in a specially designed database. The semantics of ER query and update operators is given.

1. Introduction

A variety of interactive graphical query and update systems were described in the literature [4, 5, 6]. Several systems have been based on the relational [2] and the Entity-Relationship (ER) model [1, 3, 7]. Although they can provide valuable assistance in formulating queries or updates, further research is generally suggested [4]. In this paper we will discuss an ER query and update testbed system called ERQUI that can be easily modified and therefore can be very helpful for experimenting with various database interfaces. Although during the presentation we basically assume the set of interface operators as described in [1], the ERQUI system allows us to conveniently define new operators and modify the existing ones.

First group of operators of the ERQUI system allows the user to create a convenient ER view of the database. The Hide operator removes from the terminal screen the icons corresponding to the specified object. This operator can be applied to an attribute, a relationship type or an entity type. When it is applied to an entity type, it additionally removes from the screen the icons corresponding to all relationship types defined on the specified entity type. The Display operator displays on the terminal screen the icons corresponding to the attributes of the specified objects that were hidden previously. The object can be an entity type or a relationship type. The Move operator re-positions the icon of a specified object. The object can be an entity or relationship type. The Rename operator allows a user to change the displayed name of a specified object. The object can be an attribute, an entity type or a relationship type.

The second group of operators is used to specify queries. The Select operator allows the user to identify the icons corresponding to the objects that are used in the process of data retrieval. This operator can be applied to an entity type, a relationship type or an attribute. The Restrict-Values operator allows a user to construct a condition that is used in the process of data retrieval. The argument list for this operator consists of three elements: an attribute, a boolean-valued operator and a constant from the domain of the attribute. When this operator is applied, the new condition icon is created and displayed on the terminal screen. The Duplicate operator allows the creation of a duplicate of an icon corresponding to the specified object. This operator can be applied to a relationship type or an entity type. When it is applied to a relationship type it creates a duplicate of the icon corresponding to the specified object. When it is applied to an entity type it creates a duplicate of the icon corresponding to the specified object and all displayed relationship types defined on the specified entity type.

The group of operators to specify updates includes the Delete, Modify and Insert operator. The Delete operator allows the user to specify the object which tuples are to be deleted in the process of database update. This operator can be applied to...
an entity type or a relationship type. When it is applied to one of these objects, the icon corresponding to the selected object is redrawn with a red line. The Modify operator allows the user to specify an attribute whose values are to be modified in the process of database update. The argument list for this operator consists of two elements: an attribute and a constant from the domain of the attribute. When this operator is applied, the new update icon is created and displayed on the terminal screen. The icon corresponding to the object the attribute belongs to is redrawn with a blue line. The Insert operator allows the user to construct an 'inserted_value' descriptor that is used in the process of database update. This operator can be applied to an entity type or a relationship type. Each new value for an attribute can be specified by pointing at an attribute icon and typing the new value. For each attribute of the object the new value descriptor is created and displayed on the terminal screen. For non-key attributes the new value can be null. In both cases the icon corresponding to a selected object is redrawn with a green line.

A user invokes the above described operations by using a pointing device to choose an operator from a menu including all operators described above. After choosing an operator the user either points to an element(s) of the diagram or enters text in the message area.

In this paper we discuss a method for implementing an ER query and update interface that includes the operators listed above. The approach is based on utilizing a relational database management system as a major component of an ER system architecture. During the process of manipulation of ER diagrams the information needed to generate an equivalent SQL query is accumulated in the specially designed database called ERR. The semantics of ER query and update operations is given in terms of ERR operations.

The paper is organized as follows. In the next section, an architecture of a prototype of the ERQUI system is described and the ERR database is presented. In Section 3, the component of the ERQUI system processing the menu operators is described. In Section 4, generation of an equivalent SQL expression for the underlying relational database is discussed. Section 5 contains the summary.

2. System Architecture and ERR Database

The ERQUI system presented in this paper can be easily modified. This is achieved by designing and using an auxiliary relational database called ERR describing the ER schema. The ERR relational database contains a basic information about entity types, relationship types and their attributes. It also includes the information about the position of entity and relationship type icons on the terminal screen, the description of selected attributes, the specification of conditions and update descriptors. The ERR database is one of the most important components of the ER query system architecture as shown in Figure 1.

The DBMS is a relational database management system accepting requests (queries or updates) in SQL for ERR or User Database. The Diagram Display Module is a simple subroutine to draw the ER diagram on the terminal screen according to the information stored in ERR database. The Menu Operators Module is another simple subroutine that accepts the user's graphical requests and generates the ERR database updates using SQL templates discussed in Section 3. The Execute Module constructs User Database queries or updates according to the information stored in the ERR database, which will be discussed in Section 4.

Let us first describe the ERR database in some detail. The relation Entity_type contains among other things the original name (ename) of the entity type, copy number (ecopy_number) that is initially equal to zero and the displayed name (alias) that is initially identical to ename. The position of the entity type icon on the terminal screen is described by eposition. The flag display_flag indicates whether the entity type is to be displayed on the screen. It can have two values 'Displayed' and 'Not_Displayed'. The flag select_flag indicates whether the icon corresponding to the entity type is to be darkened on the screen. It can have two values 'Selected' and 'Not_Selected'. The flag update_flag indicates whether the entity type is to be updated. It can have four values 'No_Update', 'Delete', 'Update' and 'Insert'. The Entity_type relation can be described by the following schema.

Entity_type(ename, ecopy_number, alias, position, display_flag, select_flag, update_flag)
Figure 1. System Architecture

The Relationship_type relation contains among other things the original name (rname) of the relationship type, ecopy_number, and the displayed name (alias) that is initially identical to rname. The position of the relationship type icon on the terminal screen is described by rposition. The flag rdisplay_flag indicates whether the relationship type is to be displayed on the screen. It can have two values 'Displayed' and 'Not_Displayed'. The flag rselect_flag indicates whether the icon corresponding to the relationship type is to be darkened on the screen. It can have two values 'Selected' and 'Not_Selected'. The flag rupdate_flag indicates whether the relationship type is to be updated (icon filled with dots). It can have four values 'No_Update', 'Delete', 'Update' and 'Insert'. The Relationship_type relation can be described by the following schema.

Relationship_type(rname, ecopy_number, alias, position, rdisplay_flag, rselect_flag, rupdate_flag)

To allow relationships among several entity types a separate relation Connects is introduced. The Connects relation lists entity type participating in each relationship type. This relation can be described by the following schema.

Connects(ename, ecopy_number, rname, rcopy_number)

The Attribute relation contains among others the original name (aname) of the attribute and the displayed name (alias) that is initially identical to aname. Each attribute belongs to an object with the name ername and ercopy_number. The flag display_flag indicates whether the attribute is to be displayed on the screen. It can have two values 'Displayed' and 'Not_Displayed'. The select_flag indicates whether the attribute is to be darkened on the screen (if it is displayed). It can have two values 'Selected' and 'Not_Selected'. The key_flag denotes the primary key of the entity type and can
be 'Key' or 'Not_Key'. A modified value for the attribute specified in one of the update operations is stored in update_value. The Attribute relation can be described by the following schema.

\[ \text{Attribute}(\text{aname}, \text{alias}, \text{ername}, \text{ercopy_number}, \text{display_flag}, \text{select_flag}, \text{key_flag}, \text{update_value}) \]

The Condition relation contains an entity or relationship type name (ername), ercopy_number, an attribute name aname (aname), a constant from the domain of the attribute (the same as the domain for values of the attribute) and an operator imposing a condition on the values of the attribute based on the constant.

\[ \text{Condition}(\text{ername}, \text{ercopy_number}, \text{aname}, \text{constant}, \text{operator}) \]

As an example let us consider an ER schema describing a University Database and containing three entity types FACULTY, STUDENT, COURSE and four relationship types Advises, IsTaking and Teaches. It can be represented by a diagram shown in Figure 2.

![ER diagram for the University Database](image)

Figure 2. ER diagram for the University Database.

Such a diagram can be described by the following database. The Entity_type relation contains three tuples:

\[ \{(\text{FACULTY}, 0, \text{FACULTY}, P1, \text{Displayed}, \text{Not_Selected}, \text{No_Update}),\]
\[ (\text{STUDENT}, 0, \text{STUDENT}, P2, \text{Displayed}, \text{Not_Selected}, \text{No_Update}),\]
\[ (\text{COURSE}, 0, \text{COURSE}, P3, \text{Displayed}, \text{Not_Selected}, \text{No_Update})\] \]

The Relationship_type relation contains three tuples:

\[ \{(\text{Advisor}, 0, \text{Advises}, P4, \text{Displayed}, \text{Not_Selected}, \text{No_Update}),\]
\[ (\text{IsTaking}, 0, \text{IsTaking}, P5, \text{Displayed}, \text{Not_Selected}, \text{No_Update}),\]
\[ (\text{Teaches}, 0, \text{Teaches}, P6, \text{Displayed}, \text{Not_Selected}, \text{No_Update})\] \]

The Connects relation contains six tuples:

\[ \{(\text{FACULTY}, 0, \text{Advises}, 0), (\text{FACULTY}, 0, \text{Advises}, 0),\]
\[ (\text{STUDENT}, 0, \text{IsTaking}, 0), (\text{COURSE}, 0, \text{IsTaking}, 0),\]
\[ (\text{FACULTY}, 0, \text{Teaches}, 0), (\text{COURSE}, 0, \text{Teaches}, 0)\] \]

The Attribute relation contains nine tuples:

\[ \{(\text{ssno, ssno, FACULTY}, 0, \text{Displayed}, \text{Not_Selected}, \text{Key}, \phi),\]
\[ (\text{name, name, FACULTY}, 0, \text{Displayed}, \text{Not_Selected}, \text{Not_Key}, \phi),\]
\[ (\text{salary, salary, FACULTY}, 0, \text{Displayed}, \text{Not_Selected}, \text{Not_Key}, \phi),\]
\[ (\text{idnum, idnum, STUDENT}, 0, \text{Displayed}, \text{Not_Selected}, \text{Key}, \phi),\]
\[ (\text{sname, sname, STUDENT}, 0, \text{Displayed}, \text{Not_Selected}, \text{Not_Key}, \phi),\]
\[ (\text{addr, addr, STUDENT}, 0, \text{Displayed}, \text{Not_Selected}, \text{Not_Key}, \phi),\]
\[ (\text{csnum, csnum, COURSE}, 0, \text{Displayed}, \text{Not_Selected}, \text{Key}, \phi),\]
\[ (\text{csname, csname, COURSE}, 0, \text{Displayed}, \text{Not_Selected}, \text{Key}, \phi),\]
\[ (\text{cshrs, cshrs, COURSE}, 0, \text{Displayed}, \text{Not_Selected}, \text{Key}, \phi)\] \]

The Condition relation is initially empty. It is assumed that each Pi describes the position of the corresponding object icon on the screen.

The above database should be modified when the
diagram is manipulated by the operators described in the next section. At each point of time the information necessary to properly display a diagram as well as the information necessary to properly formulate an equivalent query is reflected in the content of the ERR database.

3. The SQL Templates for the Menu Operators Module

The semantics of each operator can be defined by showing equivalent ERR database update operations. Most of the operators are partial, i.e. they are valid only if certain enabling conditions are satisfied. To simplify the discussion, the enabling conditions are not included in this paper.

3.1 Hide

When this operator is applied to an attribute with the name $1$, the following ERR update is performed:

Operation: Hide($1)
Action: UPDATE Attribute
SET display_flag = 'Not_Displayed'
WHERE aname = $1;

When this operator is applied to a relationship type with the name $1$ and the copy number $2$, the appropriate value of display_flag component of Relationship_type relation is changed to 'Not_Displayed'.

Operation: Hide($1, $2)
Action: UPDATE Relationship_type
SET display_flag = 'Not_Displayed'
WHERE mame = $1
AND rcopy_number = $2;

When this operator is applied to an entity type with the name $1$ and the copy number $2$, the appropriate value of display_flag component of Entity_type relation is changed to 'Not_Displayed'. In addition to that, the display_flag component for all relationship types in which $1$ participates are modified (an extended SQL is required to describe this operation conveniently).

Operation: Hide($1, $2)
Action: UPDATE Entity_type
SET display_flag = 'Not_Displayed'
WHERE ename = $1
AND ecopy_number = $2;

3.2 Display

This operator is applied to an object with the name $1$ and the copy number $2$. The object can be either an entity or relationship type. As a result of this operation the display_flag of each Attribute tuple corresponding to the object is updated to 'Displayed'.

Operation: Display($1)
Action: UPDATE Attribute
SET display_flag = 'Displayed'
WHERE ename = $1
AND ecopy_number = $2;

3.3 Move

When this operator is applied to an entity type with the name $1$ and the copy number $2$, the appropriate value of the position component of Entity_type relation is changed to $3$.

Operation: Move($1, $2, $3)
Action: UPDATE Entity_type
SET position = $3
WHERE ename = $1
AND ecopy_number = $2;

When this operator is applied to a relationship type with the name $1$ and the copy number $2$, the appropriate value of the position component of Relationship_type relation is changed to $3$.

Operation: Move($1, $2, $3)
Action: UPDATE Relationship_type
SET position = $3
WHERE mame = $1
AND rcopy_number = $2;

3.4 Rename

When this operator is applied to an entity type with the name $1$ and the copy number $2$, the appropriate value of the alias component of Entity_type relation is changed to $3$.

Operation: Rename($1, $2, $3)
Action: UPDATE Entity_type
SET alias = $3
WHERE ename = $1
AND ecopy_number = $2;
Operation: Rename($l, $2, $3)
Action: UPDATE Entity_type
        SET alias = $3
        WHERE ename = $1;
              AND ecopy_number = $2;

In the similar way the update for Attribute and Relationship_type can be defined.

3.5 Select

When the Select operator is applied to an attribute with the name $1 belonging to the object $2 (copy number $3), the appropriate value of the select_flag component of the Attribute relation is changed.

Operation: Select($l, $2, $3)
Action: UPDATE Attribute
        SET select_flag = 'Selected'
        WHERE aname = $1
              AND ername = $2;
              AND ercopy_number = $3;

When the Select operator is applied to an entity type with the name $1 (copy number $2), the appropriate value of the select_flag component of the Entity-type relation is changed.

Operation: Select($l, $2)
Action: UPDATE Entity-type
        SET select_flag = 'Selected'
        WHERE ename = $1
              AND ecopy_number = $2;

When the Select operator is applied to a relationship type with the name $1 (copy number $2), the appropriate value of the select_flag component of the Relationship type relation is changed. In addition to that, all entity types participating in the relationship are 'selected' (an extended SQL is required to describe this operation conveniently).

Operation: Select($l, $2)
Action: UPDATE Relationship_type
        SET select_flag = 'Selected'
        WHERE name = $1
              AND rcopy_number = $2;

        UPDATE Entity_type
        SET select_flag = 'Selected'
        WHERE [ename, ecopy_number] IN
              (SELECT ename, ecopy_number
               FROM Connects
               WHERE name = $1
              AND rcopy_number = $2);

3.6 Restrict_Values

The argument list for this operator consists of four elements: an entity or relationship type name $1 (copy number $2), an attribute name $3, a boolean-valued operator $4 and a constant $5. When this operator is applied, the new tuple is inserted into Condition relation.

Operation: Restrict_Values($l,$2,$3,$4,$5)
Action: INSERT INTO Condition
        VALUES ($1, $2, $3, $4, $5);

3.7 Duplicate

When the Select operator is applied to an object (an entity or relationship type) with the name $1 (copy number $2) and the position of the duplicate icon $3, the new tuple is inserted into the Entity-type or Relationship_type relation. In addition to that, when this operator is applied to an entity type, the new tuples are inserted into the Relationship_type relation for all displayed relationship types defined on the specified entity type.

3.8 Delete

When the Select operator is applied to an object (an entity or relationship type) with the name $1 (copy number $2), the appropriate value of update_flag component of the Entity-type or Relationship_type relation is changed. When the Select operator is applied to an entity type, the changes are as follows.

Operation: Delete($l, $2)
Action: UPDATE Entity_type
        SET update_flag = 'Delete'
        WHERE ename = $1
              AND ecopy_number = $2;

Similarly the semantics of the Update and Insert operators can be defined.

As an example let us consider a query: List the names of all students taking course 'Compilers'. Assuming that the diagram of Figure 2 is initially displayed, the query can be specified by the following sequence of user's actions:
1. Point at the operator 'Select'.
2. Point at the attribute 'sname' of the entity type 'FACULTY'.
   As a result of the above actions the icons corresponding to the attribute 'sname' and the entity type 'STUDENT' are darkened on the terminal screen.
3. Point at the relationship type 'IsTaking' ('Select' is still active)
   As a result of the above action the icon corresponding to the relationship type 'IsTaking' is darkened on the terminal screen.
4. Point at the operator 'R_Value'
5. Point at the attribute 'csname' of the entity type 'COURSE'.
6. Enter the remaining part of the condition ( = 'Compilers').
   As a result of the above actions the icon corresponding to the condition descriptor is displayed on the terminal screen adjacent to the attribute 'csname' icon.

The resulting diagram on the terminal screen is shown in Figure 3.

Such a diagram corresponds to an updated ERR database. The changes are specified below. The following two tuples of the Entity_type relation are changed:

(STUDENT, 0, STUDENT, P2, 'Displayed', 'Selected', 'No_Update')
(COURSE, 0, COURSE, P3, 'Displayed', 'Selected', 'No_Update')

The following tuple of the Relationship_type relation is changed:

(IsTaking, 0, IsTaking, P5, 'Displayed', 'Selected', 'No_Update')

The following tuple of the Attribute relation is changed:

(sname, sname, STUDENT, 0, 'Displayed', 'Selected', 'Not-Key', 0)

The Condition relation contains now one tuple:

(STUDENT, 0, sname, 'Compilers', '=?')

4. The SQL Templates for the Execute Module

The Execute operation is invoked to see the results of the user's query or to perform an update. In order to describe this operator a mapping of ER schema into an underlying relational schema should be specified [1]. Here, we assume that each entity and relationship type corresponds to a single relation (or a relational view). The relation corresponding to an entity type has the same attributes. The relation corresponding to a relationship type has several (usually two) attributes in addition to those in the relationship type. These additional attributes are foreign keys, namely the primary keys of the relations corresponding to entity types on which the relationship type is defined. As an example let us consider University Database of Figure 2. The ER schema used in this section, can be mapped into the following relational schema:

Faculty (ssno, name, salary)
Student (idnum, sname, addr)
Course (csnum, csname, cshrs)
Advises (ssno, idnum)
IsTaking (idnum, csnum)
Teaches (ssno, csnum)
For the given mapping of an ER schema into a relational schema, it should be possible to generate SQL expressions from any state of the ER diagram. Let us first describe the semantics of the execute operator when no update is present. The generated SQL query expression consists of three clauses: query-SELECT-clause, query-FROM-clause and query-WHERE-clause. The last clause is optional. To simplify the discussion it is assumed that in the rules specified below each entity and relationship type has the copy number 0 (original). The query-SELECT-clause can be obtained by creating a list containing attribute names from all 'Selected' attributes in the selected entity or relationship types.

Action: SELECT ername, aname 
FROM Attribute 
WHERE select_flag = 'Selected';

The query-FROM-clause can be obtained by creating a list containing names of all 'Selected entity and relationship types. 

Action: SELECT ename 
FROM Entity-type 
WHERE select_flag = 'Selected' 
UNION 
SELECT rname 
FROM Relationship-type 
WHERE select_flag = 'Selected';

The WHERE-clause can be obtained by combining all terms corresponding to join conditions and all terms specified by condition descriptors. The terms corresponding to join conditions are generated from the 'Selected' relationship types.

Action: SELECT Entity-type.ename, 
Attribute.aname, 
Relationship_type. rname, 
Attribute.aname 
FROM Entity-type, Connects, 
Relationship_type, Attribute 
WHERE Relationship_type.select_flag = 'Selected' 
AND Relationship_type.rname = Connects.rname 
AND Connects.ename = Entity_type.ename 
AND Entity_type.ename = Attribute.ename 
AND Attribute.key_flag = 'Key';

The terms corresponding to condition descriptors are generated from the Condition relation.

Action: SELECT ername, aname, operator, 
constant 
FROM Condition 

As an example let us consider the ERR database corresponding to query in Figure 3. The SQL expression generated for this query is:

SELECT sname 
FROM Student, IsTaking, Course 
WHERE Student.idnum = IsTaking.idnum 
AND Course.csnum = IsTaking.csnum 
AND csname = 'Compilers';

Let us now define the semantics of the Execute operator when the update flag is present. When the request to delete tuples from the database is specified (there is a red icon on the terminal screen), an equivalent SQL expression consists of two clauses: the delete-FROM-clause and the delete-WHERE-clause. The clauses are generated from the current diagram in the following way.

The delete-FROM-clause contains a single name of an entity or relationship type that has been indicated for deletion and can be obtained as follows.

Action: SELECT ename 
FROM Entity-type 
WHERE update_flag = 'Delete' 
UNION 
SELECT rname 
FROM Relationship-type 
WHERE update_flag = 'Delete';

If there is only one 'selected' entity type on the diagram then the delete-WHERE-clause can be obtained by appending all conditions using the AND separator similarly to the query-WHERE-clause.

If there is more than one 'selected' object on the diagram then the delete-WHERE-clause can be constructed by appending three components. The first component is a list of name(s) of a 'key' attribute(s) of the updated object that can be obtained as follows assuming that an entity type is the updated object.

Action: SELECT Entity_type.ename, 
Attribute.aname 

FROM Entity_type, Attribute
WHERE Entity_type.update_flag = 'Delete'
AND Entity_type.ename = Attribute.ername
AND Attribute.key_flag = 'Key';

If a relationship type is the updated object, then the list of 'key' attribute(s) consists of 'key' attribute(s) of the participating entity types and can be obtained as follows.

Action: SELECT Relationship_type. rname,
                        Attribute.aname
FROM Entity_type, Connects,
     Relationship_type, Attribute
WHERE Relationship_type.update_flag = 'Delete'
AND Relationship_type.mame = Connects.mame
AND Connects.ename = Entity_type.ename
AND Entity_type.ename = Attribute.ername
AND Attribute.key_flag = 'Key';

The second component delete-WHERE-clause is the token 'IN' and the third component is a SQL query expression. The query-SELECT-clause is identical to the first component. The query-FROM-clause and query-WHERE-clause are created according to the rules for generating an SQL query expression described above.

When the request to modify the database is specified (there is a blue icon on the terminal screen), an equivalent SQL expression consists of three clauses: modify-UPDATE-clause, modify-SET-clause and modify-WHERE-clause. The clauses are generated from the current diagram in the following way. The modify-UPDATE-clause contains a single name of an entity or relationship type that has been chosen for modification (blue icon). The modify-SET-clause can be obtained by appending the new value that was entered by a user. The modify-WHERE-clause is obtained similarly to the delete-WHERE-clause.

When the request to insert tuples into the database is specified (there is a green icon on the terminal screen), an equivalent SQL expression consists of two clauses: insert-INTO-clause and insert-VALUES-clause. The clauses are generated from the current diagram in the following way. The insert-INTO-clause contains a single name of an entity or relationship type that have been chosen for insertion (green icon). The VALUES-clause can be obtained by appending the new values that were entered by a user. If the tuple is inserted into a relationship type then the new values of 'key' attributes of participating entities need to be included in the insert-VALUES-clause.

As an example let us consider the query in Figure 3 and the corresponding content of the ERR database as shown in Section 3. The SQL expression for this query generated according to the rules discussed in this section is:

SELECT sname
FROM Student, IsTaking, Course
WHERE Student.idnum = IsTaking.idnum
AND Course.csnum = IsTaking.csnum
AND csname = 'Compilers';

5. Summary
In this paper we have presented a method for implementation of an ER graphical query and update interface. Using such interface the queries and updates are specified by manipulating ER schema diagrams displayed on the terminal screen. Diagrams are transformed until they represent a desired user query or update. During the process of query formulation the information needed to generate an equivalent SQL expression is accumulated in the specially designed relational database called ERR. We have described the basic ER operators and explained how they modify the ERR database.

A graphical query interface based on the described approach has been implemented using an IBM PC computer and a relational database management system. The scope of the language is equivalent to a large subset of SQL.

References
2. B. Czejdo, D.W. Embley, and M. Rusinkiewicz,