https://doi.org/10.48009/2_iis_2006_263-268

AVOIDANCE OF CYCLICAL REFERENCE OF FOREIGN KEYS IN DATA MODELING USING THE ENTITY-RELATIONSHIP MODEL

Ben B. Kim, Seattle University, bkim@seattleu.edu

ABSTRACT

The entity-relationship (ER) model is clearly established as a conceptual model of choice when building relational database systems. Once the ER model is built, it can be mapped algorithmically to the relational model and consequently to the data definition language (DDL) of SQL. Algorithmic mapping of the ER model to the relational model can produce foreign keys referencing each other's tables cyclically. This cyclical reference of foreign keys generates irresolvable deadlocks when enforcing the referential integrity rule using the cascading option. However, it is not obvious for database designers while building an ER model. In this article, we provide a set of rules of ER modeling for avoiding cyclical references of foreign keys.

Keywords: Database Design, Cyclical Reference, Foreign Keys, Entity-Relationship Model, Relational Model

INTRODUCTION

Since Chen [1] developed the Entity-Relationship (ER) model, it has become a widely accepted way of designing a conceptual model for database systems. Developmental stages of database systems design involve requirements analysis, conceptual design using ER models, logical design using the relational models, and physical design and implementation as shown in Figure 1 [3]. The ER model provides an architectural tool for designing relational database systems [2] without concerning the underlying syntactical details of any particular database management system (DBMS). The ER model is a formal model based on mathematical set theory. Thus, it can be algorithmically mapped to other formal models such as the relational model and consequently SQL.

Many Computer-Aided Software Engineering (CASE) tools provide this feature of algorithmic mapping of the ER model to the relational model and SQL. This algorithmic mapping saves the drudgery of programming in SQL for data architects. While mapping the ER model to the relational model, foreign keys can be defined on the tables in a cyclical reference to each other. This cyclical reference of foreign keys generates irresolvable deadlocks when enforcing the referential integrity rule using the cascading option. However, it is not obvious for database designers while building an ER model.

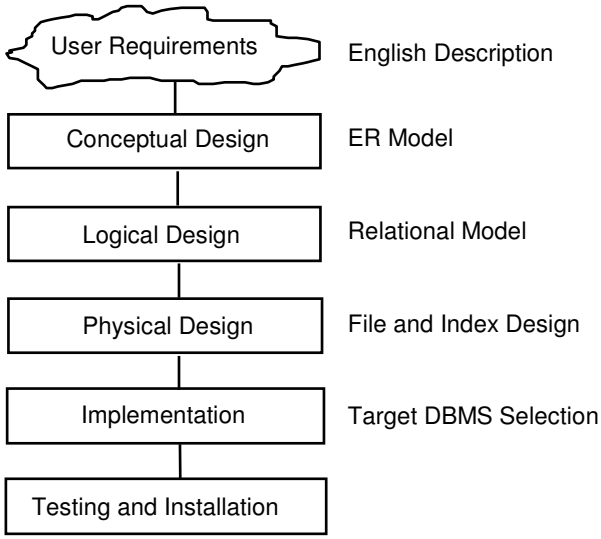

Figure 1. Database Systems Design and Implementation

In this article we first explain the mapping algorithm between the ER model and the relational model. Then, a cyclical reference problem of foreign keys is discussed. Finally, we provide a set of rules of ER modeling for avoiding cyclical references of foreign keys.

MAPPING THE ER MODEL TO THE RELATIONAL MODEL

Basically, the ER model consists of a set of entity types and relationship types. Entity types are a thing of relevance and relationships represent an association between entity types. There are three types of cardinality ratios specified for a relationship type: one to one (1:1), one-to-many (1:M) and many-to-many (M:N).

The relational model is composed of relations [2]. A relation in the relational model is interchangeably used as a table in SQL. Relations (or tables) are

connected using a foreign key. A foreign key is an attribute or composite of attributes referencing a primary key of some other relation.

Each relationship type in the ER model is mapped to a foreign key in the relational model. A foreign key is defined in the relational model depending on the cardinality ratio of a relationship type like the following:

Relationship with One-to-One (1:1) Cardinality Ratio

As shown in Figure 2, entity types *E1* and *E2* are mapped to tables *T1* and *T2* in the relational model. A relationship with 1:1 cardinality ratio can be mapped as a foreign key defined on either table *T1* or *T2*. A relationship type *R_1_1* is mapped to a foreign key *FK_T1* in the relational model. Consequently, SQL can be generated as shown in Figure 2.c.

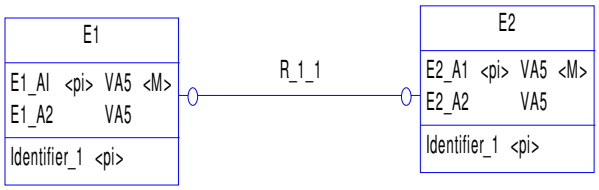

Figure 2a. ER Model with 1:1 Relationship

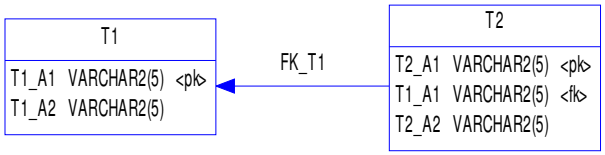

Figure 2b. Relational Model

In many CASE (Computer Aided Software Engineering) tools, designers can choose where to define a foreign key. In case of Power Designer from Sybase, for example, designers can specify which reference between E1← E2 (E2 references E1) and E2← E1 (E1 references E2) is more "dominant." If a "dominant role" is specified for E2←E1, then a foreign key is defined on E1.

```
create table T1 (
 T1_A1 VARCHAR2(5) not null,
 T1_A2 VARCHAR2(5),
 constraint    PK_T1    primary    key
(T1_A1)
)

create table T2 (
 T2_A1 VARCHAR2(5) not null,
 T1_A1 VARCHAR2(5),
 T2_A2 VARCHAR2(5),
 constraint    PK_T2    primary    key
(T2_A1),
 constraint    FK_T1    foreign    key
(T1_A1)
 references T1 (T1_A1)
)
```

Figure 2c. SQL

Relationship with One-to-Many (1:M) Cardinality Ratio

As shown in Figure 3, entity types *E1* and *E2* are mapped to tables *T1* and *T2* in the relational model. A relationship with 1:M cardinality ratio can be mapped as a foreign key defined on a table on the *many* side, *T2*. A relationship type *R_1_1* is mapped to a foreign key *FK_T1* on *T2*. Consequently, SQL can be generated as shown in Figure 3.c.

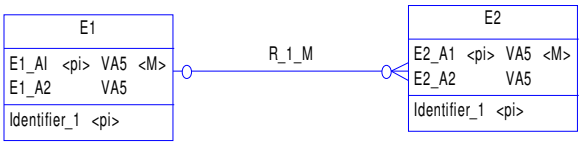

Figure 3a. ER Model with One to Many Relationship

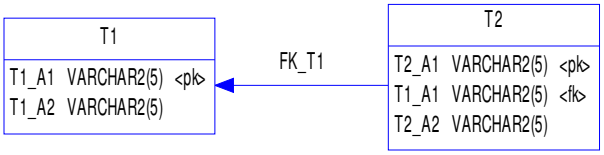

Figure 3b. Relational Model

```
create table T1 (
 T1_A1 VARCHAR2(5) not null,
 T1_A2 VARCHAR2(5),
 constraint    PK_T1    primary    key
(T1_A1)
)

create table T2 (
```

```
 T2_A1 VARCHAR2(5) not null,
 T1_A1 VARCHAR2(5),
 T2_A2 VARCHAR2(5),
 constraint    PK_T2    primary    key
(T2_A1),
 constraint    FK_T1    foreign    key
(T1_A1)
 references T1 (T1_A1)
)
```

Figure 3c. SQL

Relationship with Many-to-Many (M:N) Cardinality Ratio

As shown in Figure 4, entity types *E1* and *E2*, are mapped to tables *T1* and *T2* in the relational model. To map a relationship with M:N cardinality ratio, a new table needs to be created and foreign keys are defined on a new table. To map a relationship type *R_M_N*, a new table, *R_M_N,* is created. Foreign keys *FK_T1* and *FK_T2* are defined there. Consequently, SQL can be generated as shown in Figure 4.c.

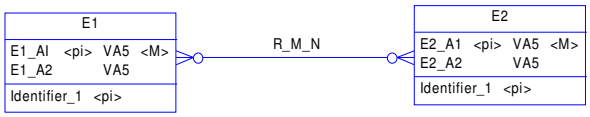

Figure 4a. ER Model with Many-to-Many Relationship

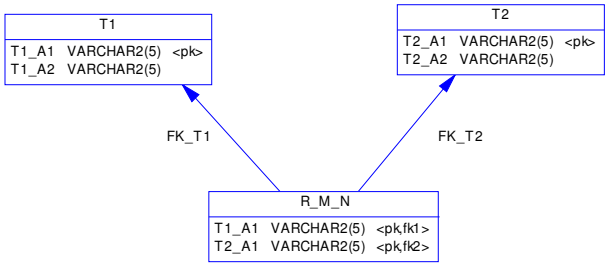

Figure 4b. Relational Model

```
create table T1 (
 T1_A1 VARCHAR2(5) not null,
 T1_A2 VARCHAR2(5),
 constraint    PK_T1    primary    key
(T1_A1)
)

create table T2 (
 T2_A1 VARCHAR2(5) not null,
 T2_A2 VARCHAR2(5),
 constraint    PK_T2    primary    key
(T2_A1)
)

create table R_M_N (
 T1_A1 VARCHAR2(5) not null,
 T2_A1 VARCHAR2(5) not null,
 constraint   PK_R_M_N   primary   key
(T1_A1, T2_A1),
 constraint    FK_T1    foreign    key
(T1_A1)
 references T1 (T1_A1),
 constraint    FK_T2    foreign    key
(T2_A1)
 references T2 (T2_A1)
)
```

Figure 4c. SQL

ENFORCEMENT OF REFERENTIAL INTEGRITY

When a foreign key is defined, its referential integrity constraint can be enforced procedurally (using triggers) or declaratively. Otherwise, a foreign key value in a referencing table may not have a matching primary key value in a referenced table. A SQL standard, SQL-1999, defines the following enforcement of referential integrity on updates or deletions on the referenced table as explained by Türker et al. [4]:

- *Cascade:* the update/deletion of a referenced row leads to an update/deletion of all rows referencing this row.
- *Set Null:* each column f_i of the foreign key of each row that references the updated/deleted row is set to null.
- *Set Default:* the update/deletion of a referenced row requires that the foreign keys of all rows that reference this row are set to their default values.
- *Restrict:* this rule disallows the update/deletion if the row is referenced by a row in any table.
- *No Action:* this rule is similar to the previous one except a few differences. In this case, intermediate referential integrity constraints violation is not enforced unless it remains a violation until the end.

Cyclical Reference of Foreign Keys

Cyclical reference of foreign keys occurs when a foreign key (fk1) defined on one table (T1) uses "cascade" option for its enforcement and a foreign key (fk2) defined on the other table (T2) references a primary key (pk1) of T1. For example, in Figure 5, if fk1 on DEPT is using a cascade option for delete and a row with E_ID = 101 in EMP is to be deleted, a row with D_ID = 100 has to be cascade-deleted. Since pk1 of DEPT is also referenced by fk2 of EMP, however, a row with D_ID=100 cannot be deleted. Thus, a deadlock is caused by cyclical reference of fk1 and fk2.

Unless this deadlock by the cyclical reference is corrected at the time of the design phase of tables, it will cause an unexpected freeze of the database systems during the run time. Redefinition of tables would cause major damages for enterprises due to the stoppage of corporate applications running on the database systems. Thus, it would be wise to prevent the cyclical reference at the design phase of database systems. Next, rules for avoiding cyclical references of foreign keys are introduced while designing an entity-relationship model.

DEPT

pk1 fk1:EMP.E_ID

D_ID	D_Name	Manager_id
100	Accounting	101
200	Finance	102
300	Inventory	103

fk1 references E_ID (pk2) of EMP. fk2 references D_ID (pk1) of DEPT.

EMP

pk2 fk2:DEPT.D_ID

E_ID	E_LName	Salary	D_No
101	Smith	100000	100
102	Johnson	95000	100
103	Dickson	110000	200
104	Brown	99000	300
105	King	120000	

Figure 5. Example of Cyclical Reference

DESIGN RULES FOR AVOIDING CYCLICAL REFERENCES

The ER model does not provide any theoretical constraints to avoid cyclical references of foreign keys. In this section, design rules for avoiding a cyclical reference of foreign keys are explained when developing an ER model. By following these rules, we can avoid the cyclical reference problem at the design stage of building database systems before encountering unexpected errors when implementing it in SQL.

1:1 and 1:1

As shown in Figure 2, a foreign key can be defined on either table when the cardinality constraint of the relationship between entity types is 1:1. Thus, when two 1:1 relationships are specified on the two entity types (E1 and E2), we have three options for how to define foreign keys on the tables mapped from E1 and E2. Let's say Tables T1 and T2 are mapped from E1 and E2, respectively.

Then, we can map two foreign keys on either T1 or T2. Also, we can map one foreign key on each table. For example, as shown in Figure 6, relationship R1 is mapped to a foreign key on Table T2 and relationship R2 to a foreign key on Table T1. This causes a cyclical reference as discussed previously. Hence, it is important to specify the option where to define a foreign key when designing an ER model. If we choose to define two foreign keys on the table (either T1 or T2) on the ER model, there will be no cyclical reference problems.

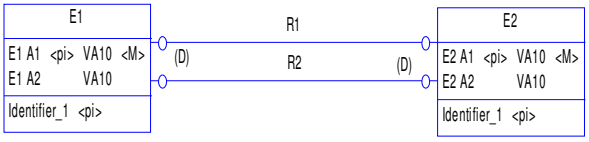

Figure 6a. 1:1 and 1:1 via ER Model

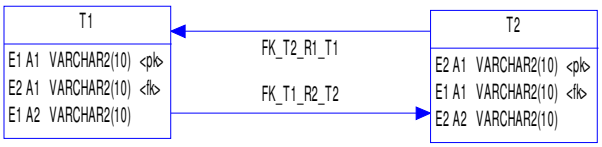

Figure 6b. 1:1 and 1:1 via Relational Model

1:1 and 1: M

As discussed previously, in case of a relationship with a cardinality ratio of 1:M, the ER model can be mapped to the relational model by defining a foreign key on the table on the many side as shown in Figure 3. A relationship with cardinality ratio of 1: 1 can be

mapped by defining a foreign key on either table as shown in Figure 2.

If two foreign keys for the relationships with cardinality ratios of 1:1 and 1:M are defined on the same table, there will be no cyclical reference problem. As can be seen in Figure 7, however, if foreign keys are defined on different tables, there will be a cyclical reference. Thus, as discussed in the previous section, a location of foreign key definition should be specified accordingly.

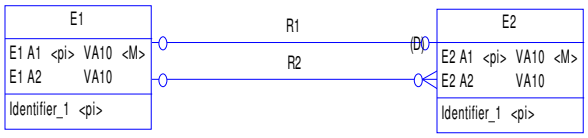

Figure 7a. 1:1 and 1:M via ER Model

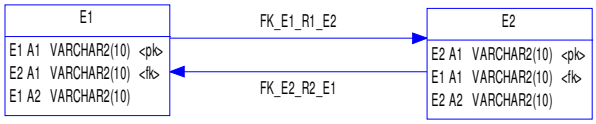

Figure 7b. 1:1 and 1:M via Relational Model

1:M and 1:M

As shown in Figure 3, in case of 1:M relationships, a foreign key is defined on the table mapped from the entity type on the many-side. If two relationships with the cardinality ratio of 1:M are specified with each many-side defined on two entity types as shown in Figure 8.a, there will be a cyclical reference as can be seen in Figure 8.b. Thus, to avoid the cyclical reference, this type of design needs to be avoided.

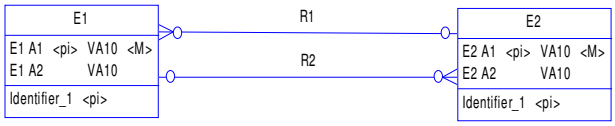

Figure 8a. 1:M and 1:M via ER Model

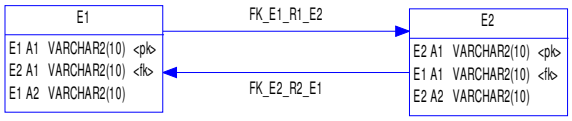

Figure 8b. 1:M and 1:M via Relational Model

1:1 and M:N

When the ER model with the cardinality ratios of 1:1 and 1:M is mapped to the relational model, there is no cyclical reference of foreign keys as can be seen in Figure 9.

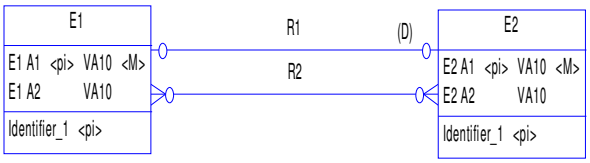

Figure 9a. 1:1 and M:N via ER Model

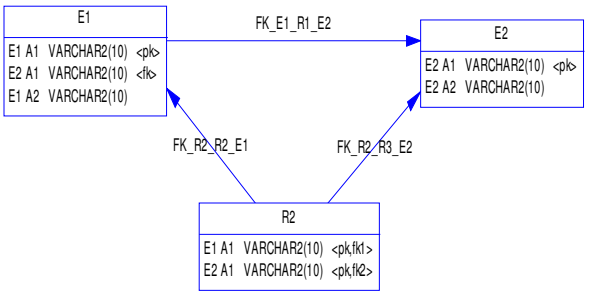

Figure 9b. 1:1 and M:N via Relational Model

CONCLUDING REMARKS

As described by Elmasri and Navathe [3], the design of database systems typically involves requirements analysis, conceptual modeling (using ER model), logical modeling (using the relational model), physical design, and implementation/testing (see Figure 1). As a computationally formal model independent of the logical models (e.g., relational model, hierarchical model or network model), an ER model can be algorithmically mapped to the relational model.

Unless the rules described here for avoiding cyclical references of foreign keys are followed, however, the ER model cannot be mapped to the relational model properly due to the cyclical reference problem of foreign keys. A relational model with cyclical references will generate run-time errors when implemented in SQL. In this article, the rules to avoid cyclical references of foreign keys are discussed when designing an ER model. If CASE tools for ER modeling include these rules as a constraint to check the validity of an ER model, many unnecessary design errors by cyclical references can be avoided.

REFERENCES

1. Chen, P. (1976). The entity relationship model—Towards a unified view of data, *ACM Transactions on Database Systems*, 1(1), 9-36.
2. Codd, E. (1970). A relational model for large shared data banks, *Communications of ACM*, *13*(6), 377-387.
3. Elmasri, R. & Navathe, S. (2004) *Fundamentals of Database Systems*, 4/E, Boston, MA: Addison Wesley.
4. Türker, C. & Gertz, M. (2001). Semantic integrity support in SQL:1999 and commercial (object) relational database management systems, *The VLDB Journal — The International Journal on Very Large Data Bases*, *10*(4), 241-269.