CHAPTER 9

SEMANTIC DATA MODEL TO RELATIONAL DATA MODEL

CHAPTER OBJECTIVES

- Examine the data model transformation method in detail
- Understand the circumstances under which this method must be used
- Study the mapping of components from the semantic to the relational model
- Specifically, learn the transformation of relationships
- Finally, compare the two models and check out the results of transformation

Toward the end of Chapter 8, we indicated that either of two methods might be used for creating a relational data model. Model transformation method is one of these two methods. This method is a straightforward way of examining the components of your semantic data model and then transforming these components into components of the required relational data model.

Two techniques were presented for creating a semantic model. A semantic model is a generic model. The object-based data modeling technique produces a semantic model by starting out with business objects that are of interest to an organization. The more popular technique, the entity-relationship data modeling technique, also produces a semantic model by adopting a natural way of looking at information requirements. For a given set of information requirements, both techniques produce similar semantic models with comparable components. You know how the components of an object-based data model maps to the components of the entity-relationship data model.

In this chapter, you will study the transformation of the object-based data model into the relational data model. It will not be necessary to repeat the discussion to

Database Design and Development: An Essential Guide for IT Professionals by Paulraj Ponniah
ISBN 0-471-21877-4 Copyright © 2003 by John Wiley and Sons, Inc.

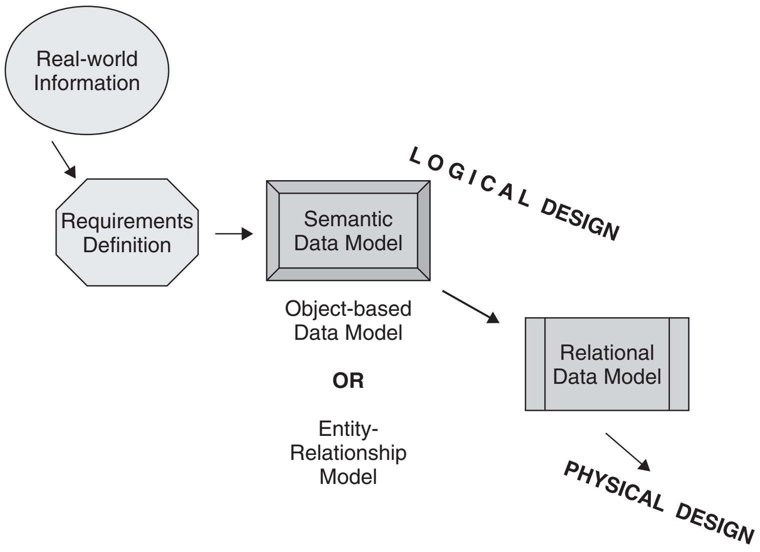


Figure 9-1 Data model transformation in the flow.

cover the transformation of the entity-relationship data model to the relational data model; the transformation principles are the same. You can easily derive them yourselves. So our concentration is on the transformation of the object-based data model. Nevertheless, while providing examples or listing data model components, we will present notations and components from both object-based and entity-relationship data models.

MODEL TRANSFORMATION APPROACH

Obviously, first you need to firm up your requirements definition before beginning any data modeling. We discussed requirements-gathering methods and contents of requirements definitions in great detail. Requirements definition drives the design of the semantic data model. Figure 9-1 shows the transition from the requirements definition phase. Note the flow from real-world information to the eventual phase of physical design. Note the model transformation activity.

Merits

Why go through the process of creating a semantic model first and then transforming it into a relational data model? Does it not sound like a longer route to logical design? What are the merits and advantages of the approach? Although we have addressed these questions earlier, in bits and pieces, let us summarize the merits and rationale for the model transformation approach.

Need for Semantic Model You must ensure that your final database system stores and manages all aspects of the information requirements. Nothing must be missing from the database system. Everything should be correct. The proposed database system must be able to support all the relevant business processes and provide users with proper information. Therefore, any data model as a prelude to the proposed database system must be a true replica of the information requirements.

A semantic data model captures the true and complete meaning of information requirements. The model is made up of a complete set of components such as objects, attributes, and relationships and so is able to represent every aspect of the information requirements. If there are variations in object types or relationship types in the information requirements, a semantic data model can correctly reflect such nuances.

Limitations of Implementation Models Consider the conventional data models such as the hierarchical, network, and relational data models. These are models that are implemented in commercial database systems. Hierarchical, network, and relational databases are offered by vendors. The conventional or implementation models are the ones that stipulate how data is perceived, stored, and managed in a database system. For example, the relational data model lays down the structure and constraints for how data can be perceived as two-dimensional tables and how relationships may be established through logical links. As such, the implementation data models address data modeling from the point of view of storing and managing data in the database system.

However, the objectives of database development are to ensure that any data model used must truly replicate all aspects of information requirements. The conventional data models do not directly perceive data from the point of view of information requirements; they seem to come from the other side. Therefore, a conventional data model is not usually created directly from information requirements. Such an attempt may not produce a complete and correct data model.

Need for Generic Model Imagine a process of creating a conventional data model from information requirements. First of all, what is the conventional data model that is being created? If it is a hierarchical data model, then you, as a data modeler, must know the components of the hierarchical data model thoroughly and also know how to relate real-world information to these model components. On the other hand, if your organization opts for a relational data model, again you, as a data modeler, must know the components of the relational data model and also know how to relate real-world information to the relational model components.

However, data modeling must concentrate on correctly representing real-world information irrespective of whether the implementation is going to be hierarchical, network, or relational. As a data modeler, if you learn one set of components and gain expertise in mapping real-world information to this generic set of components, then you will be concentrating on capturing the true meaning of real-world information and not on variations in modeling components.

Simple and Straightforward The attraction for the model transformation method of creating a relational model comes from the simplicity of the method. Once the semantic data model gets completed with due diligence, the rest of the process is straightforward. There are no complex or convoluted steps. You simply have to follow an orderly sequence of tasks.

Suppose your organization desires to implement a relational database system. Obviously, the information requirements must be defined properly no matter which type of database system is being implemented. Information requirements define the set of real-world information that must be modeled. A data modeler who specializes in semantic data modeling techniques creates a semantic data model based on information requirements. At this stage, the data modeler need not have any knowledge of the relational data model. All the data modeler does is to represent the information requirements in the form of semantic model components. The next straightforward step for the data designer is to review the components of the semantic data model and change each component to a component of the relational data model.

Easy Mapping of Components An object-based data model is composed of a small, distinct set of components. It does not matter how large and expansive the entire data model is; the whole data model is still constructed with a few distinct components. You may be creating an object-based data model for a large multinational corporation or a small medical group practice. Yet, in both cases, you will be using a small set of components to put together the object-based data model. This is also true of an entity-relationship data model.

What then is the implication here? Your semantic data model, however large it may be, consists of only a few distinct components. This means that you just need to know how to transform a few distinct components. From the other side, a relational data model also consists of a few distinct components. So mapping and transforming the components becomes easy and very manageable.

When to Use This Method

When there is more than one method for creating a relational data model, a natural question arises as to how you choose to adopt one method over the other. When do you use the model transformation method and not the normalization method? In Chapter 8, we had a few hints. The model transformation method applies when the normalization method is not feasible. Let us now list the conditions that would warrant the use of the model transformation method:

Large database system. When a proposed database system is large and the data model is expected to contain numerous component pieces, the model transformation method is preferable.

Complex information requirements. Some sets of information requirements may require modeling complex variations and many types of generalization and specialization. There may be several variations in the relationships, and the attributes themselves may be of different types. Under such conditions, modeling complex information requirements directly in the relational model, bypassing the semantic data model, proves to be very difficult.

Large project. A large project requires many data modelers to work in parallel to complete the data modeling activity within a reasonable time. Each data modeler will work on a portion of information requirements and produce a partial seman-

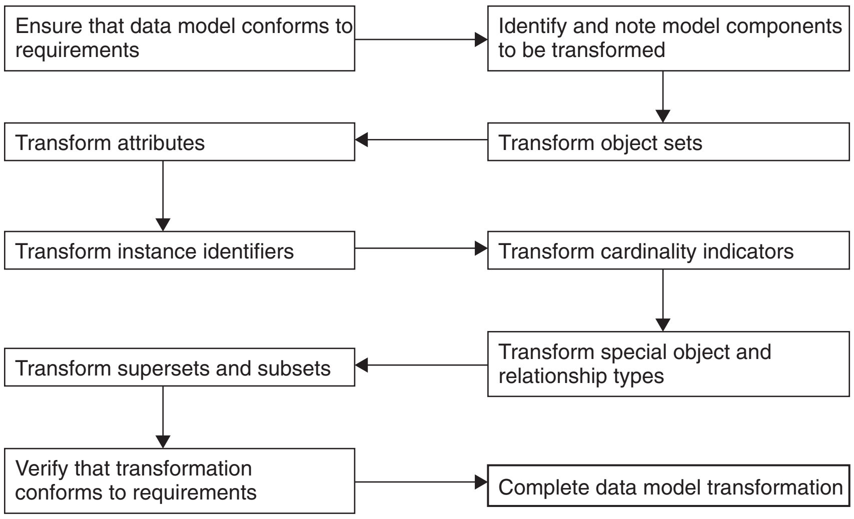

Ensure that data model conforms to requirements
Identify and note model components to be transformed
Transform object sets
Transform attributes
Transform instance identifiers
Transform cardinality indicators
Transform special object and relationship types
Transform supersets and subsets
Verify that transformation conforms to requirements
Complete data model transformation

Figure 9-2 Model transformation: major steps.

tic data model. When a project is large and the data model is expected to contain numerous component pieces, the model transformation method is preferable. The partial semantic data models are integrated and then transformed into a relational data model.

Steps and Tasks

Figure 9-2 presents the major steps in the model transformation method. Study these major steps and note how each major step enables you to proceed toward the final transformation of the data model.

Critical Issues

Although the model transformation method for creating a relational data model is straightforward and generally simple, you have to be concerned about a few critical issues. In any method, there are aspects that need special attention. Let us go over the significant issues.

Need for thoroughness. It is important to be well-disciplined and thorough in performing all the steps and tasks with utmost care. Otherwise, your final relational data model is likely to miss significant portions of information requirements.

Long and tedious. In the development of a large database system, the model transformation method could be long and tedious. It could take a long time to complete all portions of the information requirements. The project team needs to stay with it and ensure that each step and each task is performed correctly.

Integration of partial models. Again, in a large project, integration of all the partial semantic data models put together by several data modelers could present a challenge. You have to make sure that all modelers adopt the same set of modeling conventions. It is a good practice to integrate all the partial semantic models first and then transform the whole semantic data model into a relational data model.

Verification with requirements. At each step, you need to check back with the information requirements and verify that the evolving semantic model is a true representation. This verification becomes especially important when the database system tends to be large.

Review for completeness. Do not neglect to make sure that all portions of the information requirements are reflected correctly and completely in the final relational data model.

MAPPING OF COMPONENTS

While creating an object-based data model, the data modeler uses the components or building blocks available in that technique to put together the data model. You studied these components in sufficient detail in Chapter 6. Similarly, to create a relational model, the building blocks are the ones available in the relational modeling technique. You reviewed these components in Chapter 8. Essentially, transforming an object-based data model involves finding matching components in the relational data model and transferring the representation of information requirements from one model to the other. Model transformation primarily consists of mapping of corresponding components from one data model to the other.

Let us recapitulate the components or building blocks for each of the two models—the semantic and the relational data models. The list of components makes it easier to begin the study of component mapping and model transformation.

Semantic data models

Object-based data model

- Object sets
- Attributes
- Identifiers
- Relationships
- Cardinalities
- Supersets/Subsets

Entity-relationship data model

- Entity Types
- Attributes
- Keys
- Relationships
- Cardinality Indicators
- Generalization/Specialization

Relational data model
- Relations or Tables
- Rows
- Columns
- Primary Key
- Foreign Key
- Generalization/Specialization

Mapping and Transformation

Just by going through the list of components, it is easy to form the basic concepts for mapping and transformation. The semantic data model deals with the things that are of interest to the organization, the characteristics of these things, and the relationships among these things. On the other hand, the relational model stipulates how data about the things of interest must be perceived and stored, how the characteristics must be represented, and how the links between related things must be established.

First, let us consider the mapping of things and their characteristics. Then we will move on to the discussion of relationships. As you know, a major strength of the relational model is the way it represents relationships through logical links. We will describe the mapping of relationships in detail and also take up special conditions. Mapping involves taking the components of the semantic data model one by one and finding the corresponding component or components in the relational data model.

Object Sets to Relations

Let us begin with the most obvious component—the object set in the object-based data model or the entity type in the entity-relationship data model. What is an object set? If *employee* is a “thing” the organization is interested in storing information about, then *employee* is an object represented in the semantic data model. The set of all employees in the organization about whom data must be captured in the proposed relational database system is the object set EMPLOYEE.

Figure 9-3 shows the mapping of the object set EMPLOYEE. The mapping shows the transformation of the object set represented in object-based data modeling notation to the relation denoted in relational data model notation.

From the figure, note the following points about the transformation from object-based data model to relational data model:

- The object set is transformed into a relation.
- The name of the object set becomes the name of the relation.
- The object instances viewed as present inside the object set box transform into the rows of the relation.
- The complete set of object instances becomes the total set of rows of the relation or table.
- In the transformation, nothing is expressed about the order of the rows in the transformed relation.

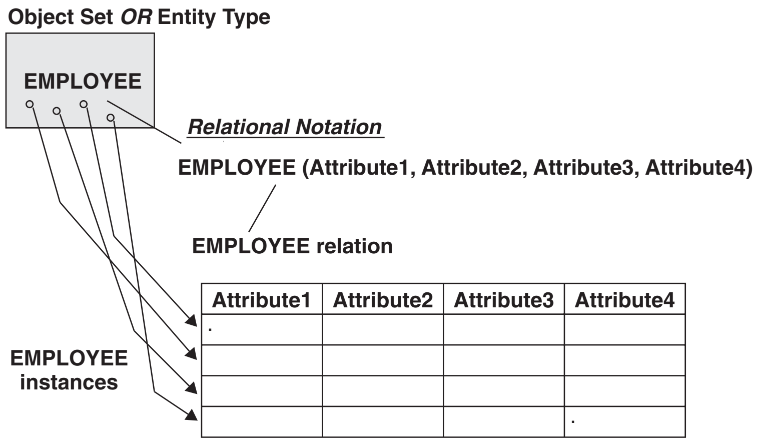
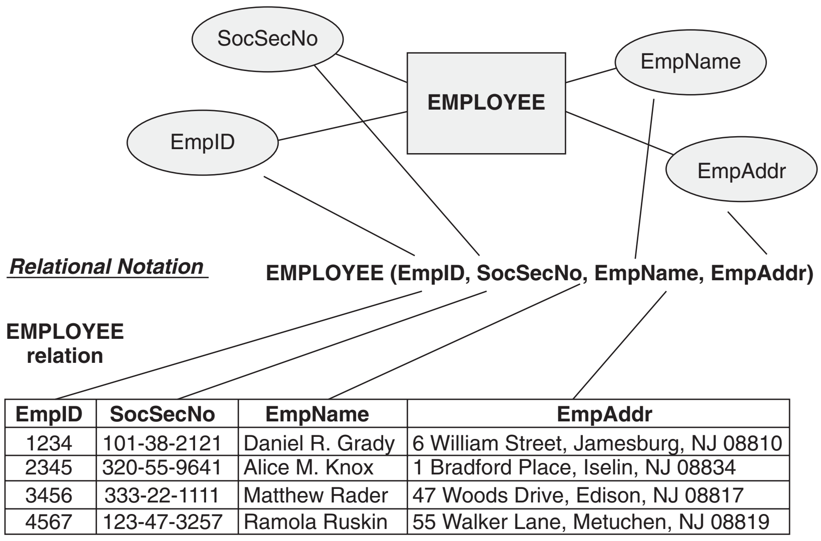


Figure 9-3 Mapping of object set or entity type.



EmpID	SocSecNo	EmpName	EmpAddr
1234	101-38-2121	Daniel R. Grady	6 William Street, Jamesburg, NJ 08810
2345	320-55-9641	Alice M. Knox	1 Bradford Place, Iselin, NJ 08834
3456	333-22-1111	Matthew Rader	47 Woods Drive, Edison, NJ 08817
4567	123-47-3257	Ramola Ruskin	55 Walker Lane, Metuchen, NJ 08819

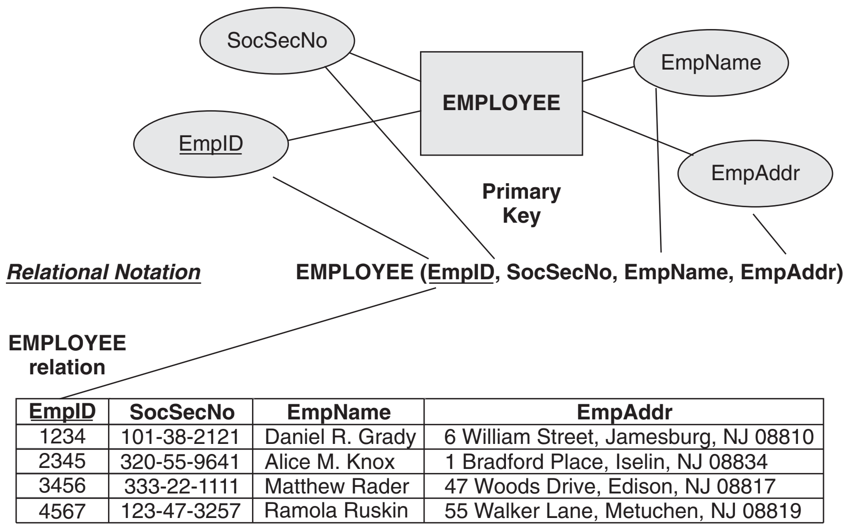
Figure 9-4 Mapping of attributes.

Attributes

Objects have intrinsic or inherent characteristics, so naturally, the next component to be considered is the set of attributes of an object. Figure 9-4 shows the transformation of attributes.

Make note of the following points with regard to the transformation of attributes:

- The attributes of an object are transformed into the columns of the corresponding relation.
- The names of the attributes become the names of the columns.
- The domain of values of each attribute translates into the domain of values for the corresponding column.



<u>EmpID</u>	SocSecNo	EmpName	EmpAddr
1234	101-38-2121	Daniel R. Grady	6 William Street, Jamesburg, NJ 08810
2345	320-55-9641	Alice M. Knox	1 Bradford Place, Iselin, NJ 08834
3456	333-22-1111	Matthew Rader	47 Woods Drive, Edison, NJ 08817
4567	123-47-3257	Ramola Ruskin	55 Walker Lane, Metuchen, NJ 08819

Figure 9-5 Mapping of instance identifiers.

- In the transformation, nothing is expressed about the order of the columns in the transformed relation.
- A single-valued or a derived attribute becomes one column in the resulting relation.
- If a multivalued attribute is present, this is handled by forming a separate relation with this attribute as a column in the separate relation.
- For a composite attribute, as many columns are used as the number of component attributes.

Instance Identifiers

In the semantic data model, each instance is uniquely identified by values in one or more attributes. These attributes together form the instance identifier. Figure 9-5 indicates the transformation of instance identifiers.

Note the following points on this transformation:

- The set of attributes forming the instance identifier becomes the primary key of the relation.
- If there is more than one attribute, all the corresponding columns are indicated as primary key columns.
- Because the primary key columns represent instance identifiers, the combined values in these columns for each row are unique.
- No two rows in the relation can have the same values in the primary key columns.
- Because instance identifiers cannot have null values, no part of the primary key columns can have null values.

TRANSFORMATION OF RELATIONSHIPS

The semantic data modeling techniques have elegant ways for representing relationships between two object sets. Wherever you perceive direct associations between instances of two object sets, the two object sets are connected by lines with a diamond in the middle containing the name of the relationship. How many of instances of one object set are associated with how many instances of the other? An indication about the numbers is given by cardinality indicators, especially the maximum cardinality indicator. The minimum cardinality indicator denotes whether a relationship is optional or mandatory.

You know that the relational data model establishes relationships between two relations through foreign keys. Therefore, transformation of relationships as represented in the semantic model involves mapping of the connections and cardinality indicators into foreign keys. We will discuss how this is done for one-to-one, one-to-many, and many-to-many relationships. We will also go over the transformation of optional and mandatory conditions for relationships. While considering transformation of relationships, we need to review relationships between a superset and its subsets.

One-to-One Relationships

When one instance of an object set is associated with a maximum of only one instance of another object set, we call this relationship a one-to-one relationship. Figure 9-6 shows a one-to-one relationship between the two object sets CLIENT and CONTACT PERSON.

If a client of the organization has designated a contact person, then the contact person is represented by the CONTACT PERSON object set. Only one contact person exists for a client. But some clients may not have contact persons, in which case there is no corresponding instance in the CONTACT PERSON object set. Now we can show the relationship by placing the foreign key column in the CLIENT relation. Figure 9-7 illustrates this transformation.

Observe how the transformation is done. How are the rows of the CLIENT relation linked to corresponding rows of the CONTACT PERSON relation? The values in the foreign key columns and primary key columns provide the linkage. Do you note some foreign key columns in the CLIENT relation with null values? What are these? For these clients, client contact persons do not exist. If the majority of clients do not have assigned contact persons, then many of the rows in the CLIENT relation will contain null values in the foreign key column. This is not a good transformation. A better transformation would be to place the foreign key column in the

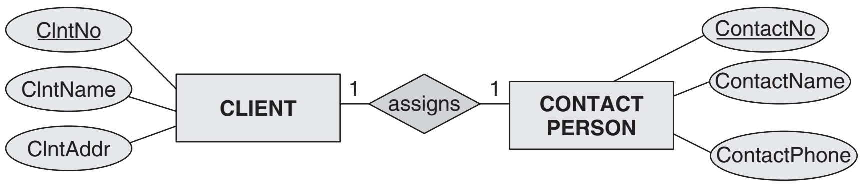
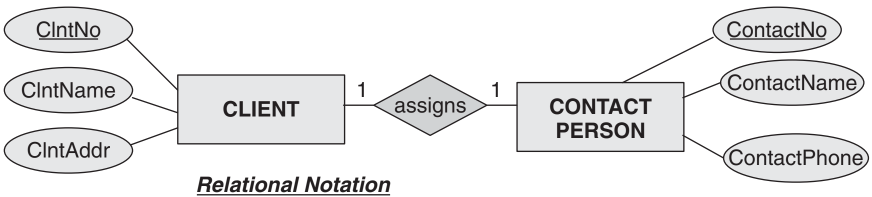


Figure 9-6 One-to-one relationship.



Relational Notation

CLIENT (<u>ClntNo</u>, ClntName, ClntAddr, ContactNo)
Foreign Key: **ContactNo** REFERENCES **CONTACT PERSON**
CONTACT PERSON (<u>ContactNo</u>,ContactName, ContactPhone)

CLIENT relation

<u>ClntNo</u>	ClntName	ClntAddr	ContactNo
11111	ABC Industries	6 William Street, Jamesburg, NJ 08810	234
22222	Progressive Systems	1 Bradford Place, Iselin, NJ 08834	123
33333	Rapid Development	47 Woods Drive, Edison, NJ 08817	
44444	Richard Associates	55 Walker Lane, Metuchen, NJ 08819	
55555	Quality Consulting	35 Rues Ave., E. Brunswick, NJ 08821	345

CONTACT PERSON relation

<u>ContactNo</u>	ContactName	ContactPhone
123	Mary Williams	732-345-8100
234	Winston Poyser	732-555-4000
345	Lisa Moore	732-767-5300

Figure 9-7 Transformation of one-to-one relationship.

CONTACT PERSON relation, not in the CLIENT relation. Figure 9-8 presents this better transformation.

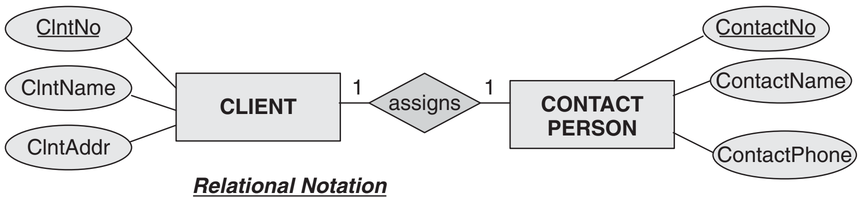
Foreign keys links two relations. If so, you must be able to get answers to queries involving data from two related tables by using the values in foreign key columns. From Figure 9-8, examine how results for the following queries are obtained.

Who is the contact person for client number 22222? Read the CONTACT PERSON table by values in the foreign key column. Find the row having the value 22222 for the foreign key attribute.

Who is the client for contact person number 345? Read the CONTACT PERSON table by values in the primary key column. Find the row having the value 345 for the primary key attribute. Get the foreign key value of this row, namely, 55555. Read the CLIENT table by values in the primary key column. Find the row having the value 5555 for the primary key attribute.

Let us summarize the points on transformation of one-to-one relationships.

- When two relations are in a one-to-one relationship, place a foreign key column in either one of the two relations. Values in the foreign key column for rows in this table match with primary key values in corresponding rows of the related table.
- The foreign key attribute has the same data type, length, and domain values as the corresponding primary key attribute in the other table.



Relational Notation

CLIENT (ClntNo, ClntName, ClntAddr)
CONTACT PERSON (ContactNo,ContactName, ContactPhone, ClntNo)
Foreign Key: **ClntNo** REFERENCES **CLIENT**

CLIENT relation

ClntNo	ClntName	ClntAddr
11111	ABC Industries	6 William Street, Jamesburg, NJ 08810
22222	Progressive Systems	1 Bradford Place, Iselin, NJ 08834
33333	Rapid Development	47 Woods Drive, Edison, NJ 08817
44444	Richard Associates	55 Walker Lane, Metuchen, NJ 08819
55555	Quality Consulting	35 Rues Ave., E. Brunswick, NJ 08821

CONTACT PERSON relation

ContactNo	ContactName	ContactPhone	ClntNo
123	Mary Williams	732-345-8100	22222
234	Winston Poyser	732-555-4000	11111
345	Lisa Moore	732-767-5300	55555

Figure 9-8 Better transformation of one-to-one relationship.

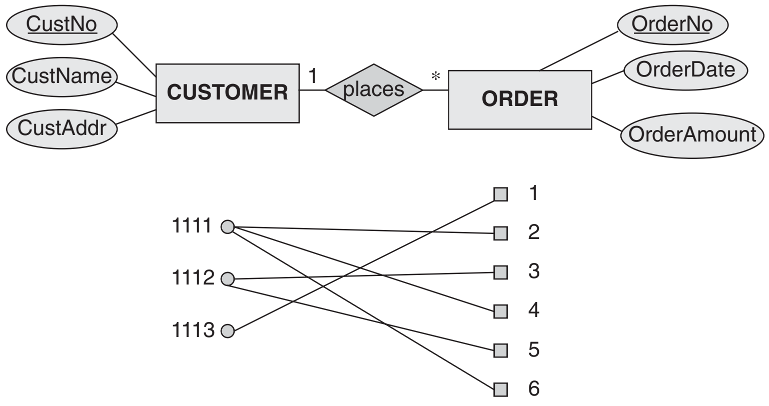
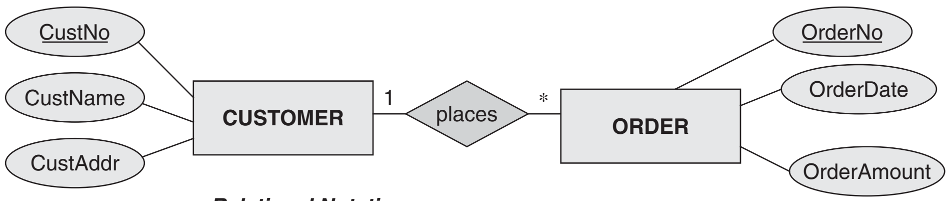


Figure 9-9 CUSTOMER and ORDER: one-to-many relationship.

- It does not really matter whether you place the foreign key column in one table or the other. However, to avoid wasted space, it is better to place the foreign key column in the table that is likely to have fewer rows.

One-to-Many Relationships

Let us begin our discussion of the one-to-many relationship by reviewing Figure 9-9. This figure shows the one-to-many relationship between the two objects CUSTOMER and ORDER.



Relational Notation

CUSTOMER (CustNo, CustName, CustAddr, OrderNo1, ...)
Foreign Key: **OrderNo1, OrderNo2, OrderNo3** REFERENCES **ORDER**

ORDER (OrderNo, OrderDate, OrderAmount)

CUSTOMER relation

CustNo	CustName	CustAddr	OrderNo1	OrderNo2	OrderNo3
1111	ABC Industries	Jamesburg, NJ	2	4	6
1112	Progressive Systems	Iselin, NJ	3	5	
1113	Rapid Development	Edison, NJ	1		

ORDER relation

OrderNo	OrderDate	OrderAmount
1	10/1/2002	2,122.50
2	10/3/2002	3,025.00
3	10/6/02	4,111.25
4	10/17/2002	3,005.50
5	10/19/2002	7,000.00
6	10/25/02	6,540.00

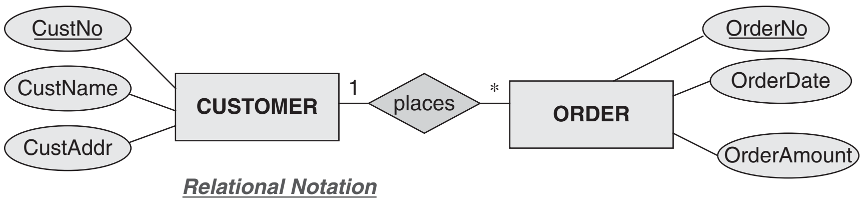
Figure 9-10 Transformation of one-to-many relationship.

The figure also indicates how individual instances of these two object sets are associated with one another. You see a clear one-to-many relationship—one customer can have one or more orders. So how should you transform this relationship? As you know, the associations are established through the use of a foreign key column. But in which table do you place the foreign key column? For transforming the one-to-one relationship, we noted that you might place the foreign key column in either relation. In the same way, let us try to place the foreign key in the CUSTOMER relation. Figure 9-10 shows this transformation of a one-to-many relationship.

What do you observe about the foreign keys in the transformed relations? In the CUSTOMER relation, the row for customer 1113 needs just one foreign key column to connect to order 1 in the ORDER relation. But the row for customer 1112 seems to need two foreign key columns, and the row for customer 1111 seems to require three foreign key columns. What if there is a customer with 50 orders? How many foreign key columns are sufficient in the CUSTOMER relation? How will you search for a particular order from the several foreign keys in the CUSTOMER relation? Obviously, this transformation is not correct.

We can try another solution by placing the foreign key column in the ORDER relation instead of including the foreign key column in the other related table. Figure 9-11 illustrates the correct solution.

Examine this figure. First, you notice that there is no need for multiple foreign keys to represent one relationship. Multiple rows in the ORDER relation have the same value in the foreign key column. This indicates the several orders related to



Relational Notation

CUSTOMER (<u>CustNo</u>, CustName, CustAddr)
ORDER (<u>OrderNo, OrderDate, OrderAmount,CustNo</u>)
Foreign Key: **CustNo** REFERENCES **CUSTOMER**

CUSTOMER relation

<u>CustNo</u>	CustName	CustAddr
1111	ABC Industries	Jamesburg, NJ
1112	Progressive Systems	Iselin, NJ
1113	Rapid Development	Edison, NJ

ORDER relation

<u>OrderNo</u>	OrderDate	OrderAmount	CustNo
1	10/1/2002	2,122.50	1113
2	10/3/2002	3,025.00	1111
3	10/6/02	4,111.25	1112
4	10/17/2002	3,005.50	1111
5	10/19/2002	7,000.00	1112
6	10/25/02	6,540.00	1111

Figure 9-11 Correct transformation of one-to-many relationship.

the same customer. The values in the foreign key column link the associated rows. From the figure, let us examine how queries involving data from two related tables work.

Which are the orders related to CUSTOMER number 1112? Read the ORDER table by values in the foreign key column. Find the rows having the value 1112 for the foreign key attribute.

What is the name of the customer for order number 4? Read the ORDER table by values in the primary key column. Find the row having the value 4 for the primary key attribute. Get the foreign key value of this row, namely, 1111. Read the CUSTOMER table by values in its primary key column. Find the row having the value 1111 for the primary key attribute.

Let us summarize the points on transformation of one-to-many relationships.

- When two relations are in a one-to-many relationship, place a foreign key column in the relation that is on the "many" side of the relationship. Values in the foreign key column for rows in this table match with primary key values in corresponding rows of the related table.
- The foreign key attribute has the same data type, length, and domain values as the corresponding primary key attribute in the other table.

Many-to-Many Relationships

As you know, in a many-to-many relationship, one instance of an object set is related to one or more instances of a second object set and also one instance of the second

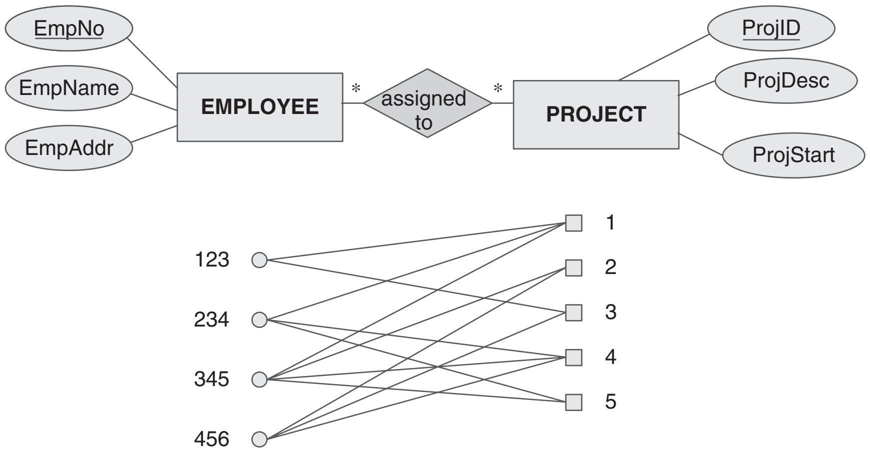


Figure 9-12 Example of many-to-many relationship.

object set is related to one or more instances of the first object set. Figure 9-12 presents an example of a many-to-many relationship.

One employee is assigned to one or more projects simultaneously or over time. Again, one project is related to one or more employees. Let us try to transform the object-based data model to a relational data model and establish the many-to-many relationship. For establishing the relationship, you have to place foreign keys. While transforming a one-to-many relationship, we placed the foreign key attribute in the relation on the "many" side of the relationship; that is, we placed the foreign key attribute in the child relation.

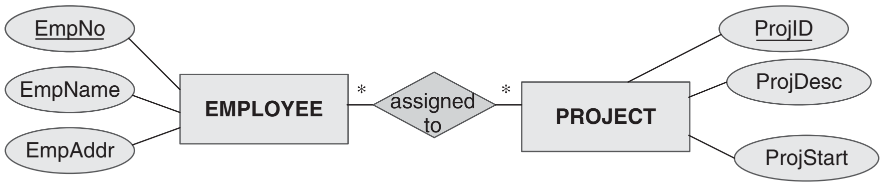
In a many-to-many relationship, which of the two relations is the child relation? It is not clear. Both relations participate in the relationship in the same way. Look at the associations shown in Figure 9-12. Transform the object sets into relations and place the foreign key in the PROJECT relation. Figure 9-13 shows this transformation with foreign key attributes placed in the PROJECT relation.

Note the foreign keys in the transformed relations. In the PROJECT relation, the rows for projects 1 and 4 need three foreign key columns whereas the rows for projects 2, 3, and 4 need two foreign key columns each. You get the picture. If some projects are related to many employees, as many as 50 or so, how many foreign key columns must the PROJECT relation have? Therefore, it appears that this method of transformation is not correct.

Let us determine how queries involving data from two related tables work.

Which are the projects related to employee 456? Read the PROJECT table by values in the foreign key columns. But which foreign key columns? All of the foreign columns? Right away, you note that finding the result for this query is going to be extremely difficult.

What are the names of employees assigned to project 1? Read the PROJECT table by values in the primary key column. Find the row having the value 1 for the primary key attribute. Get the foreign key values of this row, namely, 123, 234, and 345. Read the EMPLOYEE table by values in the



EMPLOYEE relation

EmpNo	EmpName	EmpAddr
123	Daniel R. Grady	Jamesburg, NJ 08810
234	Alice M. Knox	Iselin, NJ 08834
345	Matthew Rader	Edison, NJ 08817
456	Ramola Ruskin	Metuchen, NJ 08819

PROJECT relation

ProjID	ProjDesc	ProjStart	EmpNo1	EmpNo1	EmpNo1
1	Requirements	2/15/2002	123	234	345
2	DB Design	4/22/2002	345	456	
3	DB Implementation	8/31/2002	123	456	
4	User Training	8/1/2002	234	345	456
5	DB Fine Tuning	9/15/2002	234	345	

Related project-employee instance pairs: (1, 123), (1, 234), (1, 345), (2, 345), (2, 456), (3, 123),(3, 456), (4, 234), (4, 345), (4, 456), (5, 234), (5, 345)

Figure 9-13 Transformation of many-to-many relationship: first method.

primary key column. Find the rows having the values 123, 234, and 345 for its primary key attribute. Getting the result for this query seems to be workable.

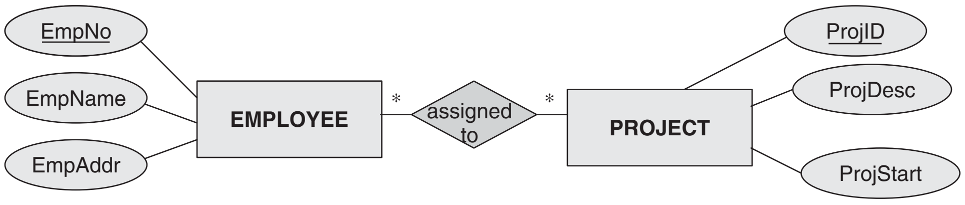
Because the transformation from the first method does not work, let us try another solution by placing the foreign key columns in the EMPLOYEE relation instead of including the foreign key column in the other related table. Figure 9-14 illustrates this method of transformation.

Where are the foreign keys in the transformed relations? In the EMPLOYEE relation, the row for employee 123 needs two foreign key columns whereas the rows for employees 234 and 456 need three foreign key columns each and the row for employee 345 needs four foreign key columns. By reasoning similar to that for the first method, if an employee is related to 25 projects over time, then you need to have that many foreign key columns in the EMPLOYEE relation.

Let us examine how queries involving data from two related tables work.

Which are the projects related to employee 456? Read the EMPLOYEE table by values in the primary key column. Find the row having the value 456 for the primary key attribute. Get the foreign key values of this row, namely, 2, 3, and 4. Read the PROJECT table by values in the primary key column. Find the rows having the values 2, 3, and 4 for its primary key attribute. Getting the result for this query seems to be workable.

What are the names of employees assigned to project 1? Read the EMPLOYEE table by values in the foreign key columns. But which foreign columns? All of them? Right away, you note that finding the result for this query is going to be very difficult.



EMPLOYEE relation

EmpNo	EmpName	EmpAddr	ProjID1	ProjID2	ProjID3	ProjID4
123	Daniel R. Grady	Jamesburg, NJ 08810	1	3		
234	Alice M. Knox	Iselin, NJ 08834	1	4	5	
345	Matthew Rader	Edison, NJ 08817	4	2	1	5
456	Ramola Ruskin	Metuchen, NJ 08819	2	3	4	

PROJECT relation

ProjID	ProjDesc	ProjStart
1	Requirements	2/15/2002
2	DB Design	4/22/2002
3	DB Implementation	8/31/2002
4	User Training	8/1/2002
5	DB Fine Tuning	9/15/2002

Related employee-project instance pairs: (123, 1), (123, 3), (234, 1), (234, 4), (234, 5), (345, 4), (345, 2), (345, 1), (345, 5), (456, 2), (456, 3), (456, 4)

Figure 9-14 Transformation of many-to-many relationship: second method.

It is clear that the second method of transformation also does not work. We seem to be in a quandary. Where should you place the foreign key column—in which of the two related tables? Placing foreign key columns in either table does not seem to work. So this second method of transformation is also not correct.

Note the pairs of related primary key values shown the above figures. Each pair represents a set of a project and a corresponding employee. Look at the pairs (1,123) and (1,234). Each pair indicates a set of related rows from the two tables. For example, the pair (1,123) indicates that the row for project 1 is related to employee 123, the pair (1,234) indicates that the row for project 1 is related to employee 234, and so on. In fact, you note that the complete set of pairs represents all the associations between rows in the two tables. In other words, the set of pairs establishes the many-to-many relationship. However, the values in the pairs are not present as foreign keys in either of the two tables. In our above two attempts at transformation, the real problem is that we do not know where to place the foreign keys—in the PROJECT relation or in the EMPLOYEE relation. What if you make a separate table out of these pairs of related values and use the values in the pairs as foreign key values? Then this new table can establish the many-to-many relationship. This elegant method is the standard method for representing many-to-many relationships in the relational data model.

Figure 9-15 illustrates the correct method of transforming a many-to-many relationship. The table containing the pairs of related values of primary keys is known as the intersection table.

Note the primary key for the intersection table. The primary key consists of two parts—one part is the primary key of the PROJECT table and the other part is the primary key of the EMPLOYEE table. The two parts act as the foreign keys

Relational Notation

EMPLOYEE (EmpNo, EmpName, EmpAddr)

PROJECT (ProjID, ProjDesc, ProjStart)

ASSIGNMENT (ProjID, EmpNo)
Foreign Keys: **ProjId** REFERENCES **PROJECT**
EmpNo REFERENCES **EMPLOYEE**

EMPLOYEE relation

EmpNo	EmpName	EmpAddr
123	Daniel R. Grady	Jamesburg, NJ 08810
234	Alice M. Knox	Iselin, NJ 08834
345	Matthew Rader	Edison, NJ 08817
456	Ramola Ruskin	Metuchen, NJ 08819

PROJECT relation

ProjID	ProjDesc	ProjStart
1	Requirements	2/15/2002
2	DB Design	4/22/2002
3	DB Implementation	8/31/2002
4	User Training	8/1/2002
5	DB Fine Tuning	9/15/2002

ASSIGNMENT relation

ProjID	EmpNo
1	123
1	234
1	345
2	345
2	456
3	123
3	456
4	234
4	345
4	456
5	234
5	345

Figure 9-15 Transformation of many-to-many relationship: correct method.

to establish both sides of the many-to-many relationship. Also, observe that each of the two relations PROJECT and EMPLOYEE is in a one-to-many relationship with the intersection relation ASSIGNMENT.

Now, let us review how queries involving data from the two related tables work.

Which are the projects related to employee 456? Read the intersection table by values in part of the primary key columns, namely, EmpNo attribute showing values for employee key numbers. Find the rows having the value 456 for this part of the primary key. Read the PROJECT table by values in its primary key column. Find the rows having the values 2, 3, and 4 for the primary key attribute. Getting the result for this query seems to be workable.

What are the names of employees assigned to project 1? Read the intersection table by values in part of the primary key columns, namely, ProjID attribute showing values for project key numbers. Find the rows having the value 1 for this part of the primary key. Read the EMPLOYEE table by values in its primary key column. Find the rows having the values 123, 234, and 345 for the primary key attribute. Getting the result for this query is straightforward and easy.

To end our discussion of transformation of many-to-many relationships, let us summarize the main points.

- Create a separate relation, called the intersection table. Use both primary keys of the participating relations as the concatenated primary key for the intersection table. The primary intersection table contains two attributes—one attribute establishing the relationship to one of the two relations and the other attribute linking the other relation.
- Each part of the primary key of the intersection table serves as a foreign key.
- Each foreign key attribute has the same data type, length, and domain values as the corresponding primary key attribute in the related table.
- The relationship of the first relation to the intersection relation is one-to-many; the relationship of the second relation to the intersection relation is also one-to-many. In effect, transformation of the many-to-many relationship is reduced to creating two one-to-many relationships.

Mandatory and Optional Conditions

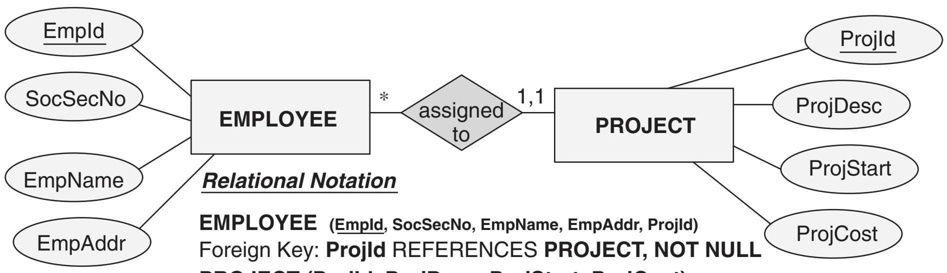
The semantic model is able to represent whether a relationship is optional or mandatory. As you know, the minimum cardinality indicator denotes mandatory and optional conditions. Let us explore the implications of mandatory and optional conditions for relationships in a relational model. In our discussions so far, we have examined the relationships in terms of maximum cardinalities. If the maximum cardinalities are 1 and 1, then the relationship is implemented by placing the foreign key attribute in either of the participating relations. If the maximum cardinalities are 1 and *, then the relationship is established by placing the foreign key attribute in the relation on the "many" side of the relationship. Finally, if the maximum cardinalities are * and *, then the relationship is broken down into two one-to-many relationships by introducing an intersection relation. Let us consider a few examples with minimum cardinalities and determine the effect on the transformation.

Minimum Cardinality in One-to-Many Relationship Figure 9-16 shows and example of a one-to-many relationship between the two object sets PROJECT and EMPLOYEE.

Note the cardinality indicators (1,1) shown next to the PROJECT object set. Intentionally, the figure does not show the minimum cardinality indicator next to EMPLOYEE. We will discuss the reason for this very shortly. What is the meaning of the cardinality indicators next to the PROJECT object set? The indicators represent the following conditions:

An employee can be assigned to a maximum of only one project. Every employee must be assigned to a project. That is, an employee instance must be associated with a minimum of 1 project instance. In other words, every employee instance must participate in the relationship. The relationship as far as the employee instances are concerned is mandatory.

Now look at the foreign key column in the EMPLOYEE table. If every employee is assigned to a project, then every EMPLOYEE row must have a value in the foreign key column. You know that this value must be the value of the primary key of the related row in the PROJECT table. What does this tell you about the foreign



Relational Notation

EMPLOYEE **(EmpId, SocSecNo, EmpName, EmpAddr, ProjId)**
Foreign Key: **ProjId** REFERENCES **PROJECT, NOT NULL**
PROJECT (ProjId, ProjDesc, ProjStart, ProjCost)

EMPLOYEE relation

EmpId	SocSecNo	EmpName	EmpAddr	ProjId
1234	101-38-2121	Daniel R. Grady	6 William Street, Jamesburg, NJ 08810	DESN
2345	320-55-9641	Alice M. Knox	1 Bradford Place, Iselin, NJ 08834	IMPL
3456	333-22-1111	Matthew Rader	47 Woods Drive, Edison, NJ 08817	DESN
4567	123-47-3257	Ramola Ruskin	55 Walker Lane, Metuchen, NJ 08819	DESN

PROJECT relation

ProjId	ProjDesc	ProjStart	ProjCost
DESN	DB Design	6/1/2000	200,000
IMPL	DB Implementation	12/15/2000	100,000
USER	User Training	10/22/2000	75,000
TUNE	DB Fine Tuning	3/1/2001	50,000

Figure 9-16 One-to-many relationship: mandatory and optional.

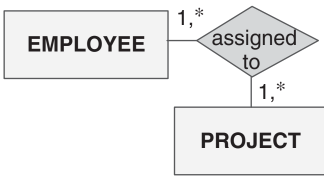
key column? In a mandatory relationship, the foreign key column cannot contain null values. Observe the Foreign Key statement under relational notation in the figure. It stipulates the constraints with the words "NOT NULL" expressing that null values are not allowed in the foreign key attribute.

Next, consider the optional condition. Suppose the cardinality indicators (0,1) are shown next to the PROJECT object set. Then the indicators will represent the following conditions:

An employee can be assigned to a maximum of only one project. Not every employee need be assigned to a project. That is, some employee instances may not be associated with any project instance at all. At a minimum, an employee instance may be associated with no project instance or with zero project instances. In other words, not every employee instance needs to participate in the relationship. The relationship as far as the employee instances are concerned is optional.

It follows, therefore, that in an optional relationship of this sort, null values may be allowed in the foreign key attribute. What do the rows with null foreign key attribute in the EMPLOYEE relation represent? These rows represent those employees who are not assigned to a project.

Minimum Cardinality in Many-to-Many Relationship Figure 9-17 shows an example of many-to-many relationship between the two object sets PROJECT and EMPLOYEE.



Relational Notation

EMPLOYEE (EmpNo, EmpName, EmpAddr)

PROJECT (ProjID, ProjDesc, ProjStart)

ASSIGNMENT (ProjID, EmpNo)

Foreign Keys: **ProjId** REFERENCES **PROJECT NOT NULL**
EmpNo REFERENCES **EMPLOYEE NOT NULL**

EMPLOYEE relation

EmpNo	EmpName	EmpAddr
123	Daniel R. Grady	Jamesburg, NJ 08810
234	Alice M. Knox	Iselin, NJ 08834
345	Matthew Rader	Edison, NJ 08817
456	Ramola Ruskin	Metuchen, NJ 08819

PROJECT relation

ProjID	ProjDesc	ProjStart
1	Requirements	2/15/2002
2	DB Design	4/22/2002
3	DB Implementation	8/31/2002
4	User Training	8/1/2002
5	DB Fine Tuning	9/15/2002

ASSIGNMENT relation

ProjID	EmpNo
1	123
1	234
1	345
2	345
2	456
3	123
3	456
4	234
4	345
4	456
5	234
5	345

Figure 9-17 Many-to-many relationship: minimum cardinality.

Note the cardinality indicators (1,*) shown next to the PROJECT object set and (1,*) shown next to the EMPLOYEE object set. What do these cardinality indicators represent? The indicators represent the following conditions:

An employee may be assigned to many projects.

A project may have many employees.

Every employee must be assigned to at least one project. That is, an employee instance must be associated with a minimum of 1 project instance. In other words, every employee instance must participate in the relationship. The relationship as far as the employee instances are concerned is mandatory.

Every project must have at least one employee. That is, a project instance must be associated with a minimum of 1 employee instance. In other words, every project instance must participate in the relationship. The relationship as far as the project instances are concerned is mandatory.

Carefully observe the transformed relationship described in the figure. Look at the intersection relation and the concatenated primary key of this relation. As you know, each part of the primary key forms the foreign key. Note the two one-to-many relationships and the corresponding tables showing attribute values. As discussed in the previous subsection on one-to-many relationship the foreign keys in the intersection table, that is, either of the two parts of the primary key table, cannot be nulls. You may stipulate the constraints with the words "NOT NULL" in the Foreign Key statement for the intersection table. However, the two foreign keys are part of the primary key, and because the primary key attribute cannot have nulls, the explicit stipulation of "NOT NULL" may be omitted.

Next, let us take up optional conditions on both sides. Suppose the cardinality indicators (0,*) are shown next to the PROJECT and EMPLOYEE object sets. Then the indicators will represent the following conditions:

An employee may be assigned to many projects.

A project may have many employees.

Not every employee need be assigned to a project. That is, some employee instances may not be associated with any project instance at all. At a minimum, an employee instance may be associated with no project instance or with zero project instances. In other words, not every employee instance needs to participate in the relationship. The relationship as far as the employee instances are concerned is optional.

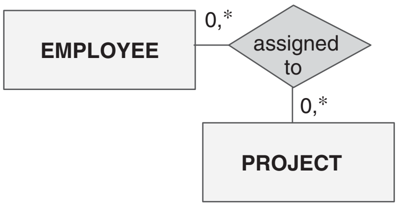
Not every project need have an employees. That is, some project instances may not be associated with any employee instance at all. At a minimum, a project instance may be associated with no employee instance or with zero employee instances. In other words, not every project instance needs to participate in the relationship. The relationship as far as the project instances are concerned is optional.

It follows, therefore, that in an optional relationship of this sort, null values may be allowed in the foreign key attributes. However, in the way the transformation is represented in Figure 9-17, allowing null values in foreign key attributes would present a problem. You have noted that the foreign key attributes form the primary key of the intersection relation, and no part of a primary key in a relation can have null values according to the integrity rule for the relational model. Therefore, in such cases, you may adopt an alternate transformation approach by assigning a separate primary key as shown in Figure 9-18.

What do the rows with null foreign key attributes in the ASSIGNMENT relation represent? These rows represent those employees who are not assigned to a project or those projects that have no employees. In practice, you may want to include such rows in the relations to indicate employees already eligible for assignment but not officially assigned and to denote projects that usually have employees assigned but not yet ready for assignment.

Minimum Cardinality on the "Many" Side In the above figures, we considered minimum cardinalities shown next to object sets on the "one" side of one-to-many relationships. For example, Figure 9-16 shows minimum cardinality indicator next to the PROJECT object set that is on the "one" side of the relationship. Now, let us consider the minimum cardinality indicator on the "many" side. Assume cardinality indicators of (1,*) on the "many" side next to the EMPLOYEE object set. This would indicate the following conditions:

A project may have many employees. Every project must have at least one employee. That is, a project instance must be associated with a minimum of 1 employee instance. In other words, every project instance must participate in the relationship. The relationship as far as the project instances are concerned is mandatory.



Relational Notation

EMPLOYEE (EmpNo, EmpName, EmpAddr)

PROJECT (ProjID, ProjDesc, ProjStart)

ASSIGNMENT (AsmntNo, ProjID, EmpNo)
Foreign Keys: **ProjId** REFERENCES **PROJECT**
EmpNo REFERENCES **EMPLOYEE**

EMPLOYEE relation

EmpNo	EmpName	EmpAddr
123	Daniel R. Grady	Jamesburg, NJ 08810
234	Alice M. Knox	Iselin, NJ 08834
345	Matthew Rader	Edison, NJ 08817
456	Ramola Ruskin	Metuchen, NJ 08819

PROJECT relation

ProjID	ProjDesc	ProjStart
1	Requirements	2/15/2002
2	DB Design	4/22/2002
3	DB Implementation	8/31/2002
4	User Training	8/1/2002
5	DB Fine Tuning	9/15/2002

ASSIGNMENT relation

AsmntNo	ProjiD	EmpNo
10	1	123
11	1	234
12	1	345
13	2	345
14	3	123
15	4	234
16	4	345
17		456
18	5	

Figure 9-18 Many-to-many relationship: alternative approach.

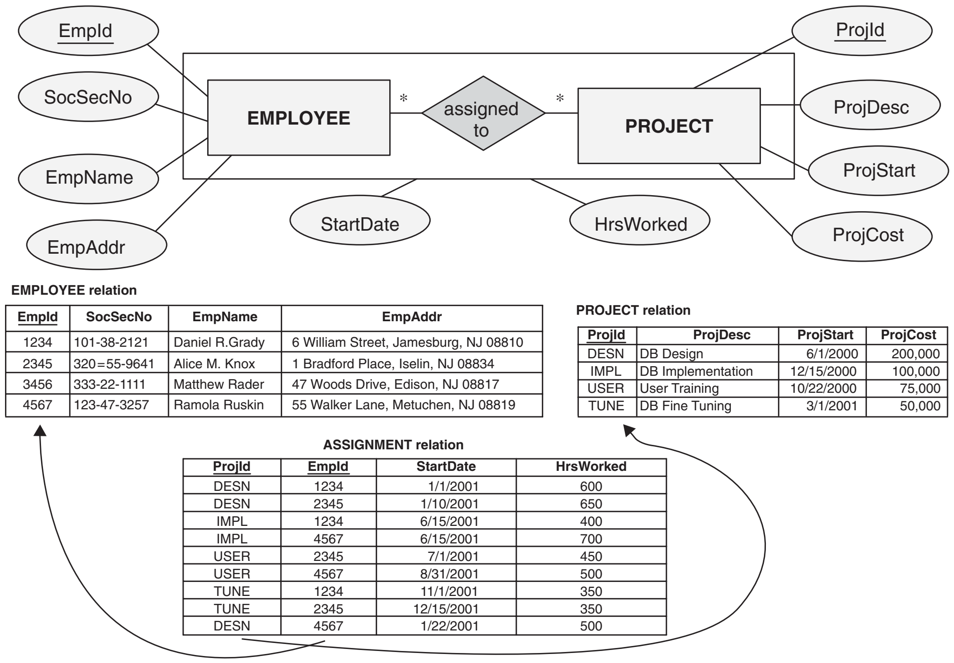
Let us interpret the mandatory nature of this relationship. The mandatory condition says that, after transformation, for every primary key value in the PROJECT relation, there must be at least one row in the EMPLOYEE relation with the same value for the foreign key attribute. Such a condition is not needed in a relational model to satisfy any integrity constraints. Therefore, it is not generally shown in the relational model. Any enforcement of this mandatory condition must be implemented by other means in the database system.

Before we end this discussion, let us also consider the corresponding optional condition. Assume cardinality indicators of (0,*) next to the EMPLOYEE object set. What are the implications in the transformation to a relational data model?

What is the meaning of the cardinality indicators (0,*) next to the EMPLOYEE object set? The indicators represent the following conditions:

A project may have many employees. Not every project need have employees. That is, some project instances may not be associated with any employee instance at all. At a minimum, a project instance may be associated with no employee instance or with zero employee instances. In other words, not every project instance needs to participate in the relationship. The relationship as far as the project instances are concerned is optional.

What is the interpretation of the optional nature of this relationship? The optional condition says that, after transformation, there may be rows in the



EMPLOYEE relation

<u>EmpId</u>	SocSecNo	EmpName	EmpAddr
1234	101-38-2121	Daniel R.Grady	6 William Street, Jamesburg, NJ 08810
2345	320=55-9641	Alice M. Knox	1 Bradford Place, Iselin, NJ 08834
3456	333-22-1111	Matthew Rader	47 Woods Drive, Edison, NJ 08817
4567	123-47-3257	Ramola Ruskin	55 Walker Lane, Metuchen, NJ 08819

PROJECT relation

<u>ProjId</u>	ProjDesc	ProjStart	ProjCost
DESN	DB Design	6/1/2000	200,000
IMPL	DB Implementation	12/15/2000	100,000
USER	User Training	10/22/2000	75,000
TUNE	DB Fine Tuning	3/1/2001	50,000

ASSIGNMENT relation

<u>ProjId</u>	<u>EmpId</u>	StartDate	HrsWorked
DESN	1234	1/1/2001	600
DESN	2345	1/10/2001	650
IMPL	1234	6/15/2001	400
IMPL	4567	6/15/2001	700
USER	2345	7/1/2001	450
USER	4567	8/31/2001	500
TUNE	1234	11/1/2001	350
TUNE	2345	12/15/2001	350
DESN	4567	1/22/2001	500

Figure 9-19 Transformation of aggregate object set.

PROJECT relation with no related rows in the EMPLOYEE relation. Again, such a condition is not needed in a relational model to satisfy any integrity constraints. Therefore, it is not generally shown in the relational model. In the relational database system, you may create PROJECT rows without creating corresponding EMPLOYEE rows without violating any integrity constraints.

Aggregate Objects as Relationships

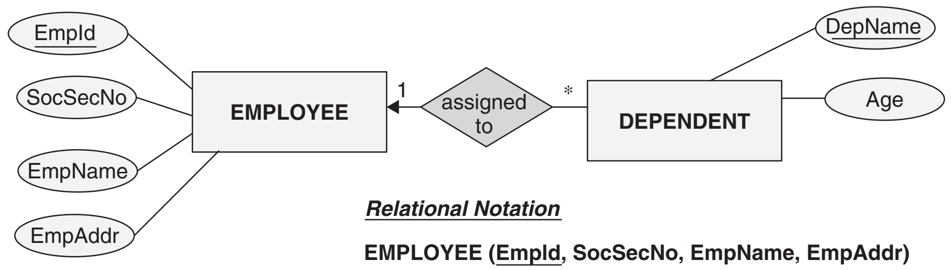
Recall that in relationships, the participating object sets together form an aggregate object set by virtue of the relationship itself. Let us discuss how such aggregate object sets are transformed into the components of a relational data model. Figure 9-19 illustrates such a transformation of an aggregate object set ASSIGNMENT.

Note the intersection relation and the attributes shown in this relation. These are the attributes of the aggregate object set. You will note that the aggregate object set becomes the intersection relation.

Identifying Relationship

In the discussion of semantic data modeling, we studied identifying relationships. A weak entity type is one that depends on another entity type for its existence. A weak entity type is, in fact, identified by the other entity type. The relationship is therefore called an identifying relationship.

Figure 9-20 illustrates the transformation of an identifying relationship. Note especially the primary key attributes of the weak entity type.



Relational Notation

EMPLOYEE (EmpId, SocSecNo, EmpName, EmpAddr)

DEPENDENT (EmpId, DepName, Age)

Foreign Key: **EmpId** REFERENCES **EMPLOYEE**

EMPLOYEE relation

EmpId	SocSecNo	EmpName	EmpAddr
1234	101-38-2121	Daniel R.Grady	6 William Street, Jamesburg, NJ 08810
2345	320=55-9641	Alice M. Knox	1 Bradford Place, Iselin, NJ 08834
3456	333-22-1111	Matthew Rader	47 Woods Drive, Edison, NJ 08817
4567	123-47-3257	Ramola Ruskin	55 Walker Lane, Metuchen, NJ 08819

EmpId	DepName	Age
1234	Mary Grady	14
1234	John Grady	11
2345	David Knox	15
2345	Jeremy Knox	13
2345	Amanda Knox	12
3456	Luke Nader	16
4567	Anne Ruskin	10
4567	Drew Ruskin	5

DEPENDENT relation

(Weak Entity Type)

Figure 9-20 Transformation of identifying relationship.

Supersets and Subsets

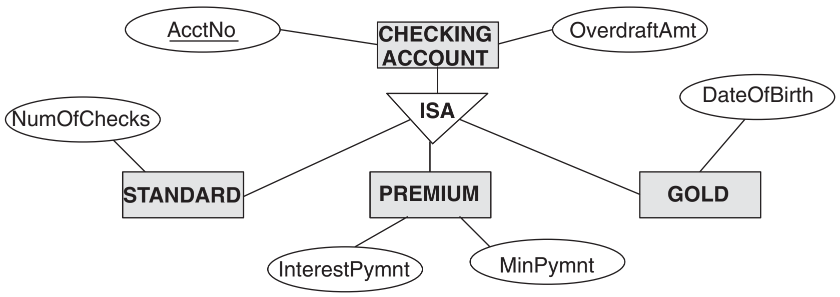
While creating semantic data models, you discover objects in the real world that are subsets of other objects. Some objects are specializations of other objects. On the other hand, you realize that individual object sets may be generalized in supertype object sets. Each subset of a superset forms a special relationship with its superset.

Figure 9-21 shows the transformation of a superset and its subsets. Note how the primary key attribute and other attributes migrate from the superset relation to the subset relations.

OUTCOME OF MODEL TRANSFORMATION

You know the phases that lead up to a relational data model. Initially, you define the information requirements. Then, you create a semantic data model based on the real-world information requirements. You may apply either of the two data modeling techniques—object-based or entity-relationship modeling—to create the semantic data model. The next step is the important transformation step. You have already studied the components and concepts of a relational data model. Now the model transformation accomplishes the creation of the relational data model.

In this section, let us review the model transformation process. We will summarize what has been covered so far in this chapter.



Relational Notation

CHECKINGACCOUNT (AcctNo, OverdraftAmt)
STANDARD (AcctNo, OverdraftAmt, NumOfChecks)
Foreign Key: **AcctNo** REFERENCES **CHECKINGACCOUNT**
PREMIUM (AcctNo, OverdraftAmt, InterestPymnt, MinPymnt)
Foreign Key: **AcctNo** REFERENCES **CHECKINGACCOUNT**
GOLD (AcctNo, OverdraftAmt, DateOfBirth)
Foreign Key: **AcctNo** REFERENCES **CHECKINGACCOUNT**

Figure 9-21 Transformation of superset and subsets.

Comparison of Models

By now, you have a fairly good grasp of the principles of transformation of a semantic data model into a relational data model. We took each component of the semantic data model and reviewed how the component is transformed into a component in the relational model. Let us list the components of the semantic data model and note how each component gets transformed. Although most of our discussion so far in this chapter has used the semantic model with the object-based modeling technique, we list the components of the entity-relationship data model here. This will further confirm that whether your semantic data model is arrived at through object-based data modeling or entity-relationship data modeling, the transformation process is the same.

The components of the semantic data model and how they are transformed into relational data model follow:

Entity Type

Strong Transform into relation.

Weak Transform into relation. Include the primary key of the identifying relation in the primary key of the relation representing the weak entity type.

Attribute

Transform attribute name into column name.

Translate attribute domains into domains for corresponding columns.

Simple, single-valued Transform into a column of the corresponding relation.

Composite Transform into columns of the corresponding relation with as many columns as the number of component attributes.

Multivalued Transform into a column of a separate relation.

Derived Transform into a column of the corresponding relation.

Primary Key

Single attribute Transform into a single-column primary key.

Composite Transform into a multicolumn primary key.

Relationship

One-to-One Establish the relationship through a foreign key attribute in either of the two participating relations.

One-to-Many Establish the relationship through a foreign key attribute in the participating relation on the "many" side of the relationship.

Many-to-Many Transform by forming two one-to-many relationships with a new intersection relation between the participating relations. Establish the relationship through foreign key attributes in the intersection relation.

Optional and mandatory conditions Set constraint for the foreign key column. If null values are not allowed in the foreign key column, this represents a mandatory relationship. Allowing null values denotes an optional relationship. Mandatory and optional conditions apply only to the participation of the relation on the "many" side of a one-to-many relationship, that is, to the participation of rows in the relation that contains the foreign key column.

Summary of Transformation

Recall the example of the semantic data model for the teaching aspects of a university given in Chapter 7. Figure 7-22 shows an entity-relationship diagram (ERD). Let us summarize our discussion on transformation of a semantic data model by transforming the model represented by that ERD into a relational data model. Figure 9-22 shows the relational data model, the outcome of the transformation.

Go through each component of the semantic data model and check out the corresponding component in the relational data model. Note the columns for each relation. Observe the establishment of relationships through foreign key columns.

CHAPTER SUMMARY

- Transformation of the semantic data model into the relational data model is an approach with many merits for creating the relational model.
- The model transformation method applies to large database systems with complex information requirements.
- However, the model transformation method could be long and tedious and must be carried out with thoroughness and care.
- Semantic data models created by either the object-based modeling technique or the entity-relationship modeling technique may be transformed into

Relational Notation

STUDENT (SocSecNo, StudntName, Major)

TEXTBOOK (ISBN, Title, Price)

AUTHOR (ISBN, Author)
Foreign Key: **ISBN** REFERENCES **TEXTBOOK**

COURSE (CourseNo, CourseDesc, Credits, ISBN)
Foreign Key: **ISBN** REFERENCES **TEXTBOOK**

CLASS (CourseNo, ClassNo, RoomNo, FacultyID)
Foreign Key: **CourseNo** REFERENCES **COURSE**
FacultyID REFERENCES **FACULTY**

FACULTY (FacultyID, FacultyName, Phone, Department, Speczn)

EXAMTYPE (TypeID, ExamName, MinScore)

GRADE (CourseNo, ClassNo, TypeID, SocSecNo, Score)
Foreign Key: **CourseNo, ClassNo** REFERENCES **CLASS**
TypeID REFERENCES **EXAMTYPE**
SocSecNo REFERENCES **STUDENT**

Figure 9-22 University ERD: transformation into relational data model.

relational data models in the same way by adopting similar transformation principles.

- Each component of the semantic data model transforms into one or more components of the relational data model.
- Object sets transform into relations of the relational data model; attributes transform into columns of relations.
- Relationships transform into logical links through foreign key attributes.
- In a one-to-one relationship, place the foreign key attribute in either of the participating relations; in a one-to-many relationship, place the foreign key attribute in the relation on the "many" side of the relationship; in a many-to-many relationship, break the relationship down into two one-to-many relationships with an intersection relation between them and place foreign key attributes in the intersection relation.

REVIEW QUESTIONS

1. List the major merits of the model transformation method of creating a relational data model.

2. Differentiate between a semantic data model and a conventional data model. Define the terms.
3. Explain some of the critical issues of the model transformation method.
4. List any three components of an object-based data model and describe how these are transformed into a relational data model.
5. List any two types of attributes. How are these transformed into the relational data model?
6. How is a composite instance identifier transformed into the relational data model? Provide two examples.
7. Describe the transformation of a one-to-many relationship with an example.
8. What is an intersection relation? When is it necessary? Give an example.
9. What is an identifying relationship? How is this transformed into the relational data model? Explain with an example.
10. What are supersets and subsets? Show the transformation of superset and subsets into the relational data model with a very simple example.

EXERCISES

1. Match the columns:

1. intersection relation	A. transformed into conventional data model
2. model transformation method	B. same primary key in transformed relations
3. weak entity type	C. relation
4. semantic data model	D. relation columns
5. aggregate object set	E. foreign key in either participating relation
6. entity type	F. two foreign keys
7. superset and subset	G. mandatory relationship
8. one-to-one relationship	H. good for large complex databases
9. foreign key not null	I. relation with concatenated primary key
10. composite attribute	J. additional relation

2. As the database designer of the relational database system for an automobile manufacturer, you have chosen the model transformation method to create the relational data model. Write a note to your project manager, describing the process and highlighting aspects of the method that would need special attention.
3. Describe how mandatory and optional conditions in relationships are transformed into the relational data model. Give specific examples.
4. Draw an ERD for a small farm market. Transform it into a relational data model.

5. Three data modelers worked on the semantic data model for a local bank. As a senior analyst on the project, it is your responsibility to integrate the partial data models created by the data modelers and transform the integrated data model into a relational model. Create a simple example with partial data models, and explain the process with this example.