7

DATA MODELING TO DATABASE DESIGN

CHAPTER OBJECTIVES

- Make the transition from data modeling to database design
- Focus on the relational data model as the logical model of choice
- Study significant fundamentals of the relational model
- Examine the leading transition approaches
- Provide in-depth coverage of the model transformation method
- Walk through transformation of each model component

In an earlier chapter, we reviewed the different information levels that exist in an organization. You need to model the information requirements of the organization at these levels to satisfy different purposes. We looked at data models at the different information levels. We started with an introduction to the conceptual data model. This is at the highest level of abstraction. A conceptual data model serves as a communication tool to converse about and confirm information requirements with domain experts.

Elaborate coverage of the conceptual data model spread over the previous chapters. This data model provides the initial expression and representation of the data content of the eventual database. Therefore, we studied the components of the conceptual data model in great detail. However, we now need to proceed further on the path toward database design and implementation. The next step in the process consists of creating a logical data model. Naturally, the logical data model has to be derived from the conceptual data model. The steps following logical data modeling consummate in designing the physical details and implementing the database.

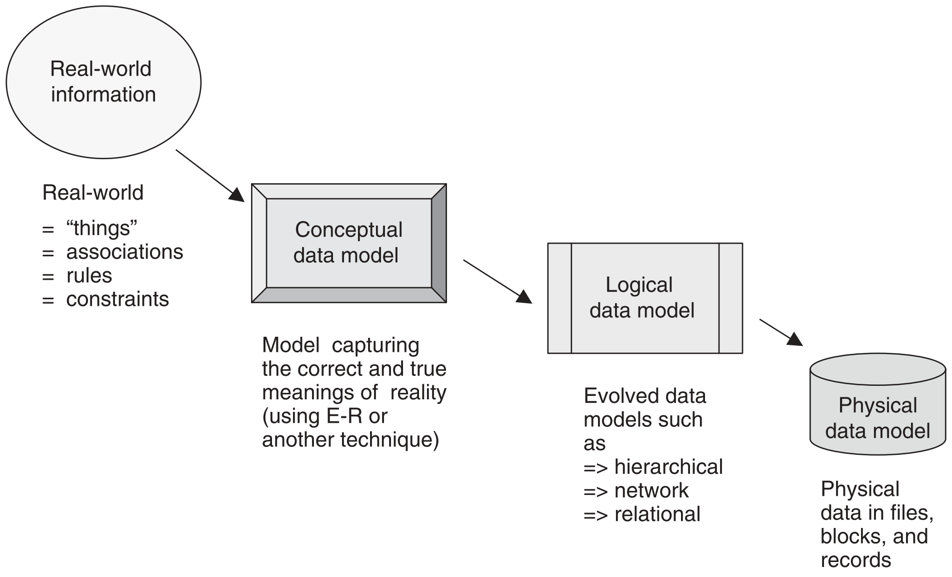


FIGURE 7-1 Data models at different levels.

For us, the relational data model will be the logical model of choice. As you know, the logical data model is to serve as a blueprint for database construction. We have selected the relational model because this has proved to be superior to other models such as hierarchical and network models. Relational databases are the most popular ones, and these work based on the relational data model.

Figure 7-1 shows the data models at different levels. Note how the figure illustrates the transition from one level to the next.

Before we deal with the transition of a conceptual to a relational data model, we need to cover the fundamentals of the relational model in sufficient detail. We have to study the various components of that model so that you may appreciate the transition steps. After the comprehensive coverage of the fundamentals, we will move into the mapping and transformation of model components from conceptual to logical. This method is the model transformation method.

We will also consider another method for creating a relational data model—a more traditional or informal method. That method has elements of intuition, and trial and error. We will do a quick review of the data normalization method. In a way, this is relational data modeling from scratch.

The above figure shows physical data modeling as a final modeling step. Physical data modeling comes very close to implementation. Specific target DBMS and hardware configurations directly influence physical design. Considerations of specific commercial DBMSs or hardware components do not fall within the scope of our discussions in this book. Therefore, we will exclude physical data modeling from our scope. You may consult the several good books available on database design and implementation to learn about that topic.

RELATIONAL MODEL: FUNDAMENTALS

The relational model uses familiar concepts to represent data. In this model, data is perceived as organized in traditional, two-dimensional tables with columns and rows. You find the rigor of mathematics incorporated into the formulation of the model. It has its theoretical basis in mathematical set theory and first-order predicate logic. The concept of a relation comes from mathematics and represents a simple two-dimensional table.

The relational model derives its strength from its simplicity and the mathematical foundation on which it is built. Rows of a relation are treated as elements of a set. Therefore, manipulation of rows may be based on mathematical set operations. Dr. Codd used this principle and provided two generic languages for manipulating data organized as relations or tables.

A relation or two-dimensional table forms the basic structure in the relational model. What are the implications? In requirements gathering, you collect much information about business objects or entities, their attributes, and relationships among them. You create a conceptual data model as a replica of information requirements. All of these various pieces of information can be represented in the form of relations. The entities, their attributes, and even their relationships are all contained in the concept of relations. This provides enormous simplicity and makes the relational model a superior logical data model.

Basic Concepts

We will begin our examination of the relational data model by studying its basic concepts. You need to review the inherent strengths of this model so that you can appreciate why it is so widespread. Having grounding in solid mathematics, data represented as a relational model renders itself for easy storage and manipulation.

Simple modeling concepts constitute the data model. When you need to transform a conceptual data model into a relational data model, the transition becomes easy and straightforward. We will also note how the mathematical concept of a relation serves as the underlying modeling concept.

Strengths of the Relational Model. Before we proceed to explore the relational model in detail, let us begin with a list of its major strengths. This will enable you to appreciate the superiority of the model and help you understand the features in a better light. Here is a summary of the strengths:

Mathematical Relation. The model uses the concept of a mathematical relation or two-dimensional table to organize data; rests on solid mathematical foundation.

Mathematical Set Theory. The model applies the mathematical set theory for manipulation of data in the database. Data storage and manipulation depend on a proven and precise approach.

Disciplined Data Organization. The model rests on a solid mathematical foundation; data organization and manipulation are carried out in a disciplined manner.

Simple and Familiar View. The model provides a common and simple view of data in the form of two-dimensional tables. Users can easily perceive how data is organized; they need not be concerned with how data is actually stored in the database.

Logical Links for Relationships. Other data models such as hierarchical and network use physical links to relate entities. If two entities such as CUSTOMER and ORDER are related, you have to establish the relationship by means of physical addresses embedded within the stored data records. In striking contrast, the relational model uses logical links to establish relationships. This is a major advantage.

Mathematical Foundation. We have mentioned that the relational model rests on a solid mathematical foundation. Specifically, in a relational model, the principles of matrix theory apply. Relational model tables are similar to mathematical matrices arranged as rows and columns. Thus, concepts of matrix manipulations can be applied to the rows and columns in a relational table.

Again, the principles and operations of mathematical set theory may be applied to the relational data model. The rows of a relational table are analogous to members of a mathematical set. If you need to work with data rows in two relational tables, you may consider these as members of two sets and apply set operations.

Figure 7-2 illustrates how mathematical principles can apply to relational tables. First, notice the similarity between elements placed in a mathematical matrix and data in the form of a relational table. Next, look at the two sets and their representations. Individual entity instances may be taken as members of sets.

Single Modeling Concept. As mentioned earlier, a relation or table is the primary data modeling concept in the relational mode. A table is a collection of columns that describe a thing of interest to the organization. For example, if COURSE is a conceptual thing of interest in a university, then a two-dimensional table or relation will represent

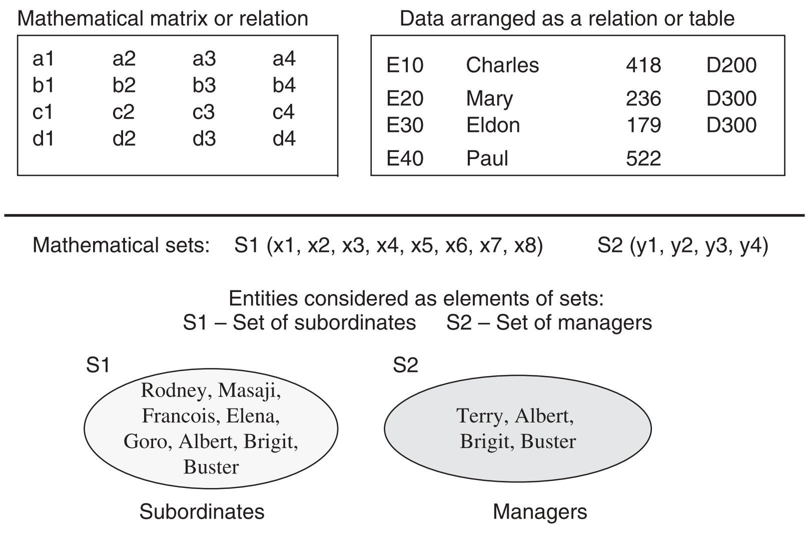

Mathematical matrix or relation
a1 a2 a3 a4
b1 b2 b3 b4
c1 c2 c3 c4
d1 d2 d3 d4
Data arranged as a relation or table
E10 Charles 418 D200
E20 Mary 236 D300
E30 Eldon 179 D300
E40 Paul 522
Mathematical sets: S1 (x1, x2, x3, x4, x5, x6, x7, x8) S2 (y1, y2, y3, y4)
Entities considered as elements of sets:
S1 – Set of subordinates S2 – Set of managers
S1
Rodney, Masaji, Francois, Elena, Goro, Albert, Brigit, Buster
S2
Terry, Albert, Brigit, Buster
Subordinates
Managers

FIGURE 7-2 Mathematical foundation of relational model.

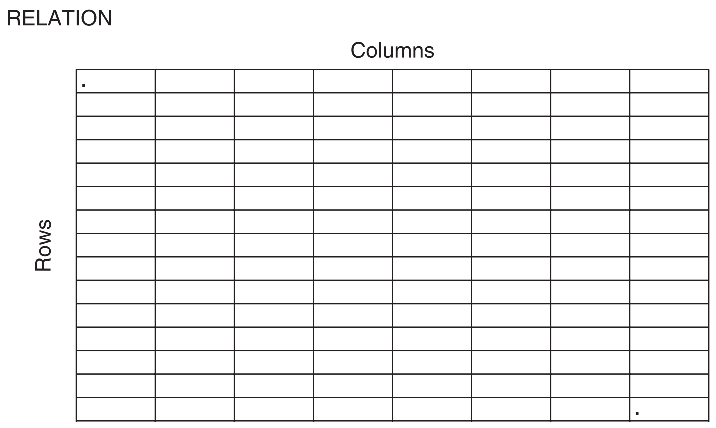


FIGURE 7-3 Relation or two-dimensional table.

COURSE in the relational data model for the university. Figure 7-3 shows a plain two-dimensional table whose format represents an entity or object.

Note the following features about a relation or two-dimensional table:

- Relation is table, representing data about some entity type or object.
- Relation is not any random table, but one that conforms to certain relational rules.
- Relation consists of a specific set of columns that can be named and an arbitrary number of rows.
- Each row contains a set of data values.
- Table names and column names are used to understand the data. The table or relation name indicates the entity type; the column names, its characteristics.

Structure and Components

As we have been repeatedly saying, the relational data model possesses a simple structure. What can you say about a simple two-dimensional table? The table has rows; the table has columns. Somehow, the usual components of a data model—entity types, attributes, relationships, and so on—must be mapped to the simple structure of a relational table.

Let us go over the basic form of a relational table. We will note which part of the table would represent what component of a data model. What are the meanings of the rows, columns, column headings, and the data values in the rows and columns? Which ones are the attributes, attribute values, identifiers, and relationships?

Relation or Table. In the relational data model, a table stands for an entity type. Each entity type in a conceptual data model gets represented by a separate table. If you have 15 entity types in your conceptual data model, then usually the corresponding relational model will contain 15 tables. As we will see later, additional tables may be become necessary. Nevertheless, an entity type in a conceptual model maps into a table or relation.

See Figure 7-4 representing the entity type called EMPLOYEE. The name of the table is the name of the entity type.

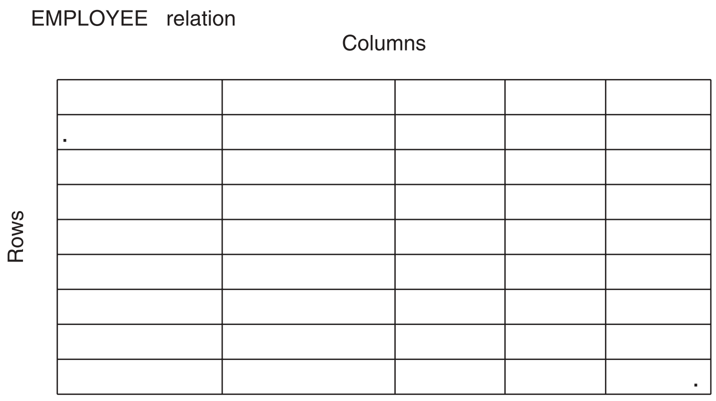


FIGURE 7-4 Employee relation or table.

Columns as Attributes. Figure 7-5 presents a relation representing an entity type EMPLOYEE.

Make note of the following about the columns in a relation as illustrated in the figure.

- Each column indicates a specific attribute of the relation.
- The column heading is the name of the attribute.
- In the relational model, the attributes are referred to by the column names and not by their displacements in a data record. Therefore, no two columns can have the same name.
- For each row, the values of the attributes are shown in the appropriate columns.
- For a row, if the value of an attribute is not known, not available, or not applicable, a null value is placed in the specific column. A null value may be replaced with a correct value at a later time.
- Each attribute takes its values from a set of allowable or legal values called the attribute domain. A domain consists of a set of atomic values of the same data type and format.
- The number of columns in a relation is referred to as the degree of the relation.

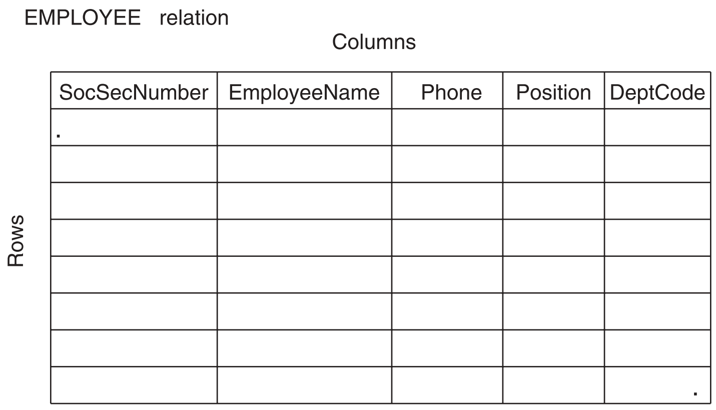


FIGURE 7-5 Employee relation: attributes.

Rows as Instances. Rows, also referred to by the mathematical name of tuples, indicate the occurrences or instances of the entity type represented by a relation. In a relation, each row represents one instance of the entity type. Each column in that row indicates one piece of data about the instance.

Figure 7-6 shows the rows or tuples for the EMPLOYEE relation.

If there are 10,000 employees in the organization, the relation will contain 10,000 rows. The number of tuples in a relation is known as the cardinality of the relation. For an EMPLOYEE relation with 10,000 rows, the cardinality is 10,000.

Now, because a relation is considered as a mathematical set, this EMPLOYEE relation is a set of 10,000 data elements. Manipulation of data in the EMPLOYEE relation, therefore, becomes a set operation. Each row represents a particular employee. Look at the row for the employee Carey Smith. Note the value shown under each column in this row. Each of the values in the columns describes the employee Carey Smith. Each value represents one piece of data about the employee. All data for the employee is contained in the specific row.

Primary Key. As mentioned above, in a relation, each tuple represents one instance of the relation. In an EMPLOYEE relation with 10,000 rows, each row represents a particular employee. But, how can we know which row represents an employee we are looking for? In order to identity a row uniquely, we can use the attribute values. We may say that, if the value of the attribute EmployeeName is "Carey Smith," then that row represents this particular employee. What if there is another Carey Smith in the organization? Thus, you need some attribute whose values will uniquely identify individual tuples. Note that the attribute SocSecNumber can be used to identify a tuple uniquely.

Given below are definitions of keys or identifiers in a relation:

Superkey. A set of columns whose values uniquely identify each tuple in a relation; however, a superkey may contain extra unnecessary columns.

Key. A minimal set of columns whose values uniquely identify each tuple in a relation.

Composite Key. A key consisting of more than one column.

Candidate Key. A set of columns that can be chosen to serve as the key.

EMPLOYEE relation

Columns

Tuples OR rows

SocSecNumber	EmployeeName	Phone	Position	DeptCode
214-56-7835	Robert Moses	516-777-9584	Programmer	501
123-44-5546	Kassia Raj	718-312-4488	Analyst	
101-54-3838	Andrew Rogers	212-313-1267	Manager	408
213-36-7854	Samuel Prabhu	212-126-3428	Controller	201
311-33-4520	Kaitlin Jones	718-567-4321	Assistant	
512-22-8542	Carey Smith	732-346-5533	Senior VP	301
111-22-3344	Amanda Lupo	908-212-5629	Executive VP	101
122-65-5378	Tabitha Williams	215-576-4598	DBA	501

FIGURE 7-6 Employee relation: rows.

Primary Key. One of the candidate keys actually selected as the key for a relation.

Surrogate Key. A key that is automatically generated for a relation by the computer system; for example, CustomerNumber, generated in sequence by the system and assigned to CUSTOMER rows. Surrogate keys ensure that no duplicate values are present in the relation. Surrogate keys are artificial keys.

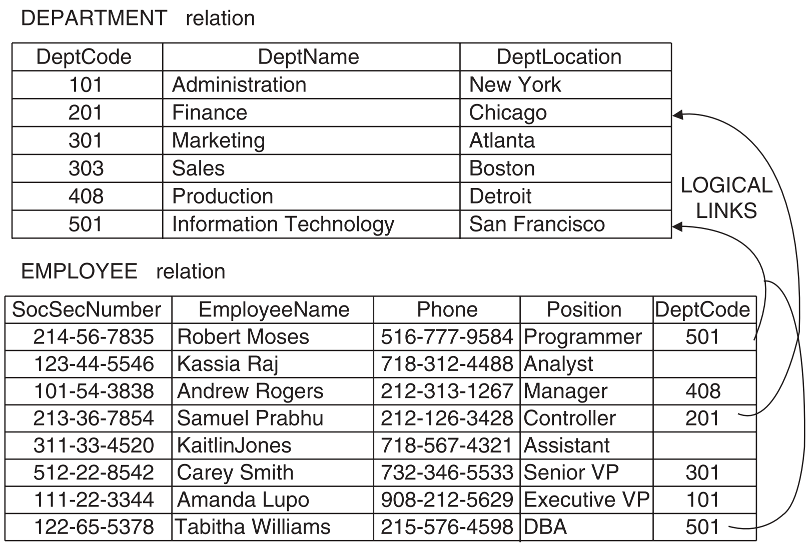
Relationship Through Foreign Keys. You have noted earlier that the relational model is superior to other conventional data models because it does not use physical links to establish relationships. The relational model uses logical links. How is this done? What is a logical link?

Figure 7-7 presents two relations EMPLOYEE and DEPARTMENT. Obviously, these two relations are related to each other because there are associations between employees and departments. One or more employees are assigned to a particular department; an employee is assigned to a specific department.

Observe the links shown between tuples in EMPLOYEE relation to corresponding tuples in DEPARTMENT relation. The DeptCode attribute in EMPLOYEE relation is called a foreign key attribute. Especially, note the value of the foreign key attribute and the value of the primary key attribute of the related row in the other relation.

Let us summarize how relationships are established in the relational data model.

- Relationships in the relational data model are established through foreign keys, not physical pointers.
- The logical link between a tuple in the first relation to a tuple in a second relation is established by placing the primary key value in the tuple of the first relation as the



DEPARTMENT relation

DeptCode	DeptName	DeptLocation
101	Administration	New York
201	Finance	Chicago
301	Marketing	Atlanta
303	Sales	Boston
408	Production	Detroit
501	Information Technology	San Francisco

EMPLOYEE relation

SocSecNumber	EmployeeName	Phone	Position	DeptCode
214-56-7835	Robert Moses	516-777-9584	Programmer	501
123-44-5546	Kassia Raj	718-312-4488	Analyst	
101-54-3838	Andrew Rogers	212-313-1267	Manager	408
213-36-7854	Samuel Prabhu	212-126-3428	Controller	201
311-33-4520	KaitlinJones	718-567-4321	Assistant	
512-22-8542	Carey Smith	732-346-5533	Senior VP	301
111-22-3344	Amanda Lupo	908-212-5629	Executive VP	101
122-65-5378	Tabitha Williams	215-576-4598	DBA	501

FIGURE 7-7 Department and employee relations: relationship.

foreign key value in the corresponding tuple of the second relation. The first relation may be referred to as the parent relation and the second as a child.

- If tuples of a relation are related to some other tuples of the same relation, then the foreign key attribute is included in the same relation. This is a recursive relationship. For example, in an EMPLOYEE relation, some tuples representing employees may be related to other tuples in the same relation representing supervisors.
- Foreign key attributes need not have the same names as the corresponding primary key attributes.
- However, a foreign key attribute must be of the same data type and length of the related primary key attribute.

In the above figure, you notice that some tuples show null values for the foreign key attributes. What is the significance of the null values?

Optional Relationship. Consider a tuple in EMPLOYEE relation with null value in the foreign key column. This shows that the specific tuple is not linked to any tuple in DEPARTMENT relation. This means that this particular employee is not assigned to any department. He or she could be a new employee not yet assigned to a department or an employee on special assignment not tied to a specific department. Null value in the foreign key column indicates the nature of the relationship. Not all employees need be associated with a department. Null values in foreign key indicate an optional relationship between the two relations.

Mandatory Relationship. In EMPLOYEE relation, suppose that null values are not allowed in the foreign key. This requires specific discrete values to be present in all the tuples of this relation. Every tuple in EMPLOYEE relation, therefore, points to a related tuple in DEPARTMENT relation. In other words, every employee must be related to a department. If null values are not allowed in the foreign key, the relationship between the two relations is a mandatory relationship.

Relational Model Notation. Figure 7-8 gives an example of relational tables. Figure 7-9 presents a standard notation used to represent this relational data model.

Note the following description of the notation:

- Notation for each relation begins with the name of the relation. Examples: WORKER, ASSIGNMENT, BUILDING, SKILL.
- For each relation, the column names are enclosed within parentheses. These are the attributes for the relation.
- Primary key attributes are indicated with underscores. Examples: <u>BuildingID</u>, <u>SkillCode</u>.
- Statements immediately following the notation of a relation indicate the foreign key within that relation. Example: Foreign Keys: SkillCode references SKILL.
- The foreign key statement includes the name of the foreign key and the name of the parent relation.
- Note the foreign key SupvID indicating a recursive relationship.

WORKER relation

WorkerNo	Name	HourlyRate	SkillCode	SupvId
1111	Morris	24.50	ELE	
1287	Vitale	20.00	MAS	
3917	Nagel	18.00	ROF	
4467	Hart	20.50	ELE	1111
5179	Grasso	22.50	PLM	

BUILDING relation

BuildingID	Address	Type	Status
H245	135 Green Street	House	S1
H267	212 Tices Road	House	S4
O123	295 Hillside Avenue	Office	S2
O156	15 Camner Terrace	Office	S3
T451	23 Oaks Drive	Townhouse	S5

SKILL relation

SkillCode	SkillType	WklyHrs
MAS	Masonry	35
FRM	Framing	40
ROF	Roofing	35
ELE	Electric	35
PLM	Plumbing	40

ASSIGNMENT relation

WorkerNo	BuildingID	StartDate	DaysWorked
1111	H245	15-Mar	10
1287	O123	15-Feb	8
5179	T451	1-Mar	7
4467	O156	15-Apr	15
1287	H267	1-Apr	9

FIGURE 7-8 Relational tables.

Data Integrity Constraints

It is essential that a database built on any specific data model must ensure validity of data. The data structure must be meaningful and be truly representative of the information requirements. Constraints are rules that make sure proper restrictions are imposed on the data values in a database. The purpose is to regulate and ensure that the data content is valid and consistent. For example, in order to preserve the uniqueness of each tuple in a relation, the constraint or rule is that the primary key has unique values in the relation. Another example is a domain constraint that requires that all values of a specific attribute be taken from the same set or domain of permissible values.

As mentioned earlier, a relational data model consists of tables or relations that conform to relational rules and possess specific properties. We will now discuss the constraints and properties that ensure data correctness and consistency in a relational data model. First, let

WORKER (WorkerNo, Name, HourlyRate, SkillCode, SupvID)
Foreign Keys: SkillCode references SKILL
SupvID references WORKER

ASSIGNMENT (WorkerNo, BuildingID, StartDate, DaysWorked)
Foreign Keys: WorkerNo references WORKER
BuildingID references BUILDING

BUILDING (BuildingID, Address, Type, Status)

SKILL (SkillCode, SkillType, WklyHrs)

FIGURE 7-9 Relational data model: notation.

us establish the reasons for ensuring data integrity. A database is said to possess data integrity if the data values will provide a correct and valid representation of the characteristics of the entities or objects. Data integrity includes consistency of data values. Data values derived from one business process must match up correctly with the same values derived from another process.

Why Data Integrity? Let us summarize the reasons for data integrity and examine how the relational data model must ensure data integrity

- In a mathematical set, no two elements can be the same. Similarly, in the relational model that is based on set theory, no two rows can be exactly the same.
- Each tuple must represent one specific entity. There must be no ambiguity in identification of the tuple for each specific entity.
- The values in all tuples for any single attribute must be of the same data type, format, and length. There must not be variations, confusion, or unpredictability in the values for every attribute.
- The columns must be identified only by names and not by position or physical order in a relation.
- A new row may be added anywhere in the table so that the content does not vary with the order of the rows or tuples in a relation.
- The model should express relationships correctly and without any room for exceptions.
- The data model must consist of well-structured relations with minimum data redundancy.
- Data manipulations in a relational database must not result in any data inconsistencies.

First, we will consider the basic relational properties that support data integrity and data consistency. Next, we will address three special cases that further enhance data integrity.

Basic Relational Properties. Following is a list of the significant relational properties that govern the relations in a relational model.

Row Uniqueness. Each row or tuple is unique—no duplicate rows with the same set of values for the attributes are allowed. No two rows are completely identical.

Unique Identifier. The primary key identifies each tuple uniquely.

Atomic Values. Each value of every attribute is atomic. That is, for a single tuple, the value of each attribute must be single-valued. Multiple values or repeating groups of attributes are not allowed for any attribute in a tuple.

Domain Constraint. The value of each attribute must be an atomic value from a certain domain.

Column Homogeneity. Each column gets values from same domain.

Order of Columns. The sequence of the columns is insignificant. You may change the sequence without changing the meaning or use of the relation. The primary key may be in any column, not necessarily in the first column. Columns may be stored in any sequence and, therefore, must be addressed by column names and not by column positions.

Order of Rows. The sequence of the rows is insignificant. Rows may be reordered or interchanged without any consequence. New rows may be inserted anywhere in the relation. It does not matter whether rows are added at the beginning, or middle, or at the end of a relation.

Entity Integrity. Consider the relation EMPLOYEE. The rows in the relation represent individual employees in an organization. The rows represent real-world entities. Each row represents a specific employee. Similarly, a row in the relation CUSTOMER stands for a particular customer. In other words, each tuple or row in a relation must be uniquely identified because each tuple represents a single and distinct entity. Entity integrity rule in the relational data model establishes this principle for an entity.

But, how is a specific row or tuple in a relation uniquely identified? As you know, the primary key serves this function. The primary key value of each tuple or row uniquely identifies that row. Therefore, entity integrity rule is a rule about the primary key that is meant to identify rows uniquely. The rule applies to single relations.

Entity integrity rule

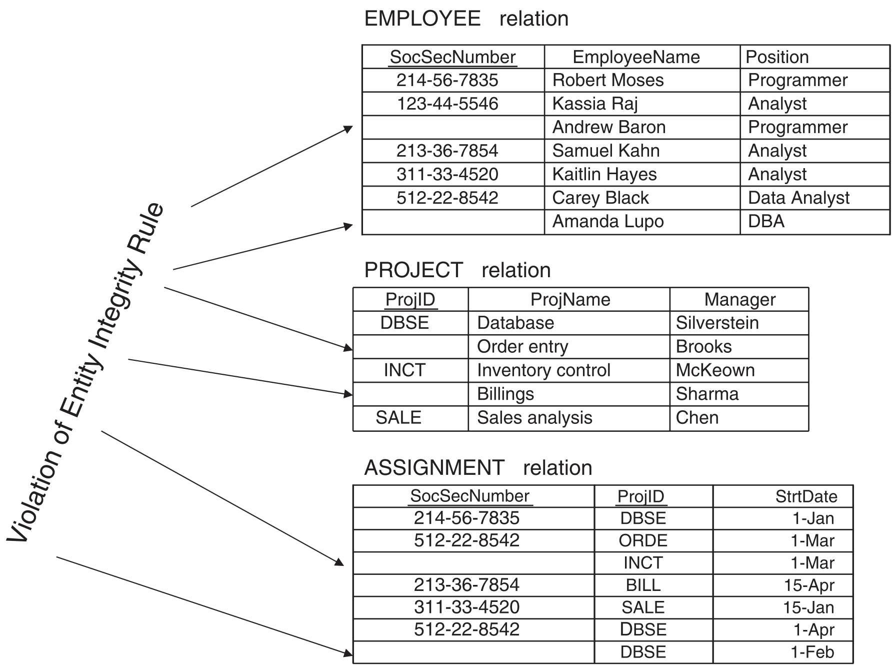
No part of the primary key of any tuple in a relation can have a null value.

Figure 7-10 presents three relations EMPLOYEE, PROJECT, and ASSIGNMENT. Relations EMPLOYEE and PROJECT have primary keys with single attributes; two attributes make up the primary key for the ASSIGNMENT relation. The figure explains how violation of entity integrity rule affects the integrity of the data model.

Note the null values present in a few rows, and because of these rows the entity integrity rule is violated in the relation. If two or more rows have nulls as primary key values, how can you distinguish between these rows? Which row denotes which specific entity? In the case of the relation ASSIGNMENT, even if part of the primary key contains nulls for any rows, the entity integrity rule is violated.

Referential Integrity. You have noted that foreign keys establish relationships between tables or relations. The value placed in the foreign key column of one table for a specific row links to a row with the same value in the primary key column in another table. Figure 7-11 shows two relations DEPARTMENT and EMPLOYEE.

Note how the values in the foreign key column DeptNo in EMPLOYEE relation and in the primary key column DeptNo in DEPARTMENT relation link related rows in the two relations. In the figure, employee Charles is assigned to department Accounting; employee Eldon, to Marketing. What about employee Mary who is supposed to be assigned to department D555? But, the database does not have department D555. Look at the row for employee Paul. This row has a null value in the foreign key column. Is this allowed? What do nulls in foreign key columns indicate? You know that nulls in foreign key columns denote optional relationships. That means employee Paul is not assigned to any department.



EMPLOYEE relation

SocSecNumber	EmployeeName	Position
214-56-7835	Robert Moses	Programmer
123-44-5546	Kassia Raj	Analyst
	Andrew Baron	Programmer
213-36-7854	Samuel Kahn	Analyst
311-33-4520	Kaitlin Hayes	Analyst
512-22-8542	Carey Black	Data Analyst
	Amanda Lupo	DBA

PROJECT relation

ProjID	ProjName	Manager
DBSE	Database	Silverstein
	Order entry	Brooks
INCT	Inventory control	McKeown
	Billings	Sharma
SALE	Sales analysis	Chen

ASSIGNMENT relation

SocSecNumber	ProjID	StrtDate
214-56-7835	DBSE	1-Jan
512-22-8542	ORDE	1-Mar
	INCT	1-Mar
213-36-7854	BILL	15-Apr
311-33-4520	SALE	15-Jan
512-22-8542	DBSE	1-Apr
	DBSE	1-Feb

FIGURE 7-10 Entity integrity rule.

When one relation is related to another through foreign key values, the references of the relationship must be clearly indicated. There should not be any ambiguities. A foreign key value must clearly indicate how that row is related to a row in the other relation. Referential integrity rule addresses the establishment of clear references between related tables. Referential integrity rule, therefore, applies to sets of two relations.

Referential integrity rule

The value of a foreign key in a table must either be null or be one of the values of the primary key in the related table.

EMPLOYEE relation

EmpNo	EmpName	Phone Ext.	DeptNo
E10	Charles	418	D200
E20	Mary	236	D555
E30	Eldon	179	D300
E40	Paul	522	

DEPARTMENT relation

Dept No	DeptName	Location
D100	Engineering	West
D200	Accounting	South
D300	Marketing	East

FIGURE 7-11 Referential integrity rule.

Functional Dependencies. Let us use EMPLOYEE, PROJECT, and ASSIGNMENT relations shown in Figure 7-10 to examine the concept of functional dependency. The notion of functional dependency in a relation arises because the value of one attribute in a tuple determines the value for another attribute. Let us look at some examples.

In the EMPLOYEE relation of Figure 7-10, note the tuple with key value 213-36-7854. This determines that the tuple represents a distinct employee whose name is Samuel Kahn and whose position is Analyst. Now, look at the tuple with key value 311-33-4520. This key value uniquely identifies an employee whose name is Kaitlin Hayes and whose position also happens to be Analyst. Let us inspect the dependencies.

Values of which attribute determine values of other attributes? Does the value of the primary key uniquely and functionally determine the values of other attributes?

- Key value 213-36-7854 uniquely and functionally determines a specific row representing Samuel Kahn with position Analyst.
- Key value 311-33-4520 uniquely and functionally determines a specific row representing Kaitlin Hayes with position Analyst.

Let us ask the questions the other way around. Does the value of the attribute Position uniquely and functionally determine the value of the primary key attribute?

- Attribute value Analyst does not uniquely determine a key value—in this case, it determines two values of the key, namely, 213-36-7854 and 311-33-4520.

What you see clearly is that the value of the primary key uniquely and functionally determines the values of other attributes, and not the other way around.

Let us express this concept using a functional dependency notation,

FD: SocSecNumber → EmployeeName

FD: SocSecNumber → Position

In the ASSIGNMENT relation, two attributes SocSecNumber and ProjectID together make up the primary key. Here, too, the values of the other attribute in a tuple are uniquely determined by the values of the composite primary key.

FD: SocSecNumber, ProjID → StrtDate

The discussion of functional dependencies leads to another important rule or constraint about the primary key of a relation.

Functional dependency rule

Each data item in a tuple of a relation is uniquely and functionally determined by the primary key, by the whole primary key, and only by the primary key.

TRANSITION TO DATABASE DESIGN

From the discussion so far, you have captured the significance of the relational data model. You have understood how it stands on a solid mathematical foundation and is, therefore, a disciplined approach to perceiving data. The view of data in the form of the common two-dimensional tables adds to the elegance and simplicity of the model. At the same time, relational constraints or rules, to which the two-dimensional tables must conform, ensure data integrity and consistency.

Commercial relational database management systems are implementations of the relational data model. So, in order to develop and build a relational database system for your organization, you need to learn how to design, put together a relational data model, and make the transition to database design. Although the model appears to be simple, how do you create a relational data model from the requirements? The previous chapters covered details of creating data models. We went through the methods and steps for creating a conceptual data model using the E-R modeling technique. Now, the task is to create a relational data model, which is not the same as one of designing the conceptual data model. Why do you need to create a relational data model? If you are developing a relational database system, then you require your information requirements represented in a relational data model. Let us explore the methods for creating a relational data model.

Design Approaches

From the previous chapters, you know how to create a conceptual data model from the information requirements. A conceptual data model captures all the meanings and content of information requirements of an organization at a high level of abstraction. Being a generic data model, a conceptual data model is not restricted by the structure and format rules of the conventional data models such as hierarchical, network, or relational data models. Representing information requirements in the form of a conceptual data model is the proper way to start the data modeling process.

Well, what are the steps between creating a conceptual data model and the implementation of a relational database system for your organization? You know that the conceptual data model, if created correctly, will represent every aspect of the information that needs to be found in the proposed database system. The next steps depend on the extent and complexity of your database system. Let us examine the options.

Database practitioners adopt one of two approaches to design and put together a relational data model. The relational data model must, of course, truly represent the information requirements. In the simplest terms, what is a relational data model? It is a collection of two-dimensional tables with rows and columns, and with relationships expressed within the tables themselves through foreign keys. So, in effect, designing and creating a relational data model reduces to creating the proper collection of two-dimensional tables.

Figure 7-12 presents the two design approaches for creating a relational data model.

Note how in one approach, you go through the steps of creating a conceptual data model first and then transform the conceptual model into a relational data model. The other approach appears to be a short-cut method bypassing the conceptual data model. In this approach, you go to the task of creating the relational data model straight from requirements definitions. However, you may still want a conceptual data model to serve as the communication vehicle between data modelers and user groups. Let us examine the basics of the two approaches.

Conceptual to Relational Model

The first method shown in Figure 7-12 takes you through the conceptual data model. In this approach, first you complete the conceptual data model. For creating the conceptual data

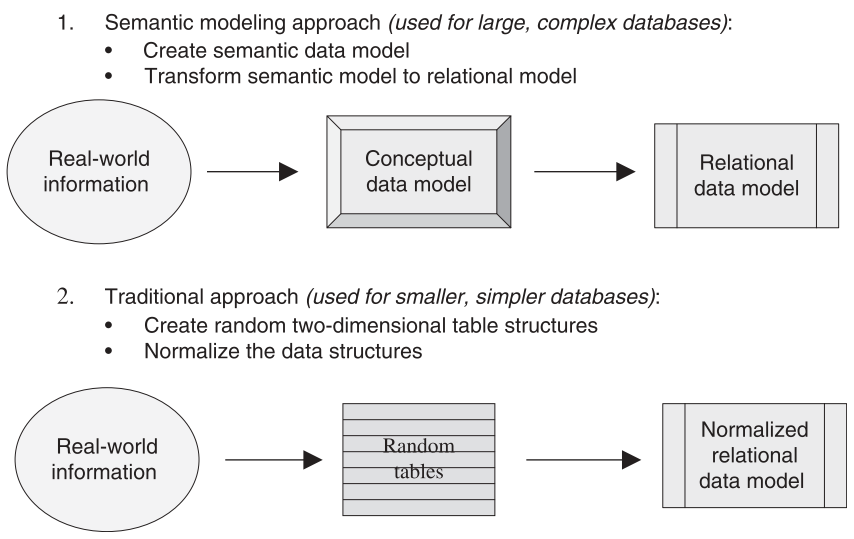

1. Semantic modeling approach (used for large, complex databases):
• Create semantic data model
• Transform semantic model to relational model
Real-world information
Conceptual data model
Relational data model
2. Traditional approach (used for smaller, simpler databases):
• Create random two-dimensional table structures
• Normalize the data structures
Real-world information
Random tables
Normalized relational data model

FIGURE 7-12 Relational data model: design approaches.

model, you may use the E-R data modeling technique. Some other modeling techniques would also produce conceptual data models.

Here are the general steps in this design approach:

- Gather the information requirements of the organization.
- Create a conceptual data model to truly represent the information requirements.
- Review the overall conceptual data model for completeness.
- Take each component of the conceptual data model at a time and examine it.
- Transform each component of the conceptual data model into a corresponding component of the relational data model.
- Pull together all the components of the relational data model resulting from the transformation from the conceptual data model.
- Complete the relational data model.
- Review the relational data model for completeness.

The next major section of this chapter elaborates on this approach to designing the relational data model. We will list the components of the conceptual model and the corresponding components of the relational model. We will determine how each component of the conceptual data model must be mapped to its corresponding component in the relational data model.

Traditional Method

Before the introduction and standardization of data modeling techniques, traditionally database practitioners had adopted a different method. A relational data model is, after all, a set of two-dimensional tables. Why not look at the information requirements and try to come up with the necessary tables to represent the data that would satisfy the information requirements? Why do you need an intermediary step of creating a conceptual data model? Does it not appear to be a practical design approach?

Although this approach is deceptively simple, as you will note in the next chapter, this method is subject to serious problems if the tables are not defined properly. You are likely to end up with a faulty set of tables in your relational data model with a high potential for data corruption and inconsistency.

Dr. Codd suggested an orderly methodology for making this design approach work. After an initial set of tables is put together intuitively, you must go through a step-by-step process of normalization of the initial tables. After completing the normalization steps, your relational data model will result in a set of tables that are free from redundancies and errors.

Here are the steps in this design approach:

- Gather the information requirements of the organization.
- Review the information requirements to determine the types of tables that would be needed.
- Come up with an initial set of tables.
- Ensure that your initial set of tables contains all the information requirements.
- Normalize the tables using a step-by-step methodology.
- Review the resulting set of tables, one by one, and ensure that none of the tables has potential redundancies or errors.
- Complete the relational data model.
- Review the relational data model for completeness.

Chapter 8 covers the data normalization method. In that chapter, we will have a detailed discussion of this approach to designing the relational data model. You will realize the need and motivation for the normalization process. We will list the normalization steps and show how to apply a single normalization principle at each step. You will note how, after each step, the set of tables gets closer to being the correct set of tables and being part of the final relational data model.

Evaluation of Design Methods

Naturally, when there are two ways for arriving at the same place, which path should you take? If both methods produce the same desired result, which method is more appropriate? The answers to these questions depend on the circumstances of the design process.

Note the following points about the two methods while making the choice between the two ways:

Same Result. If you carry out the transformation of the conceptual data model into a relational model or adopt the traditional method using normalization, you will arrive at the same relational data model. However, either method must be used carefully, making sure that every task is executed properly.

One Method Intuitive. In the traditional method, you are supposed to come up with an initial and complete set of tables. But, how do you come up with the initial set? Using what method? There is no standard method for arriving at an initial set of tables. You have to look at the information requirements and arrive at the initial set of tables mostly through intuition. You just start with the best possible set that is complete. Then you go and normalize the tables and complete the relational data model.

Other Method Systematic. The method of creating the conceptual data model first and then transforming it into the required relational data model is a systematic method with well-defined mapping algorithms. Creation of the conceptual data model is through clearly defined data modeling techniques. Then you take the components of the conceptual data model, one by one, and transform these in a disciplined manner.

Choosing Between the Two Methods. When can you adopt the traditional method? Only when you can come up with a good initial set of tables through intuition. If the information requirements are wide and complex, by looking at the information requirements it is not easy to discern the tables for the initial set. If you attempt the process, you are likely to miss portions of information requirements. Therefore, adopt the traditional approach only for smaller and simpler relational database systems. For larger and complex relational database systems, the transformation method is the prudent approach. As data modelers gain experience, they tend to get better at defining the initial set of tables and go with the normalization method.

MODEL TRANSFORMATION METHOD

This method is a straightforward procedure of examining the components of your conceptual data model and then transforming these components into components of the required relational data model. A conceptual model is a generic model. We have chosen to transform it into a relational model.

Let us study the transformation of a conceptual model created using E-R technique into relational data model. The discussions here may also be adapted to a conceptual model created using any other modeling technique. The transformation principles will be similar.

The Approach

Obviously, first you need to firm up your requirements definition before beginning any data modeling. We had discussed requirements gathering methods and contents of requirements definition in great detail. Requirements definition drives the design of the conceptual data model.

Requirements definition captures details of real-world information. After the requirements definition phase, you move to conceptual data modeling to create a replica of information requirements. From conceptual data modeling, you make the transition to a relational data model. This completes the logical design phase. Physical design and implementation follow; however, these are not completely within the purview of our study.

Merits. Why go through the process of creating a full-fledged conceptual 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 this 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 Conceptual Model. You must ensure that your final database system stores and manages all aspects of 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 information requirements.

A general data model captures the true and complete meaning of information requirements at a high level of abstraction understandable by user groups. The model is made up of a complete set of components such as entity types, attributes, relationships, and so is able to represent every aspect of information requirements. If there are variations in entity types or relationship types in the information requirements, a generic data model can correctly reflect such nuances.

Limitations of Implementation Models. Consider the conventional data models such as the hierarchical, network, or relational data models. These are models that are implemented in commercial database systems. You have hierarchical, network, and relational databases 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 on 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 the real-world to this generic set of components, then your concentration will be 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 for creating a relational model comes from the simplicity of the method. Once the conceptual data model gets completed with due diligence, the rest of the process is straightforward. There are no complex, convoluted steps. You have to simply follow an orderly sequence of tasks.

Suppose your organization desires to implement a relational database system. Obviously, 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 conceptual data modeling techniques creates a conceptual 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 information requirements in the form of a conceptual model. The next straightforward step for the data designer is to review the components of the conceptual data model and change each component to a component of the relational data model.

Easy Mapping of Components. A conceptual 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 E-R 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 E-R data model.

What then is the implication here? Your conceptual data model, however large it may be, consists of only a few distinct components. This means 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 do you choose and adopt one method over the other? When do you use the model transformation method and not the normalization method? In a previous section, 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 set 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 conceptual 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 conceptual data model. When a project is large and the data model is expected to contain numerous partial models, the model transformation method is preferable. The partial conceptual data models are integrated and then transformed into a relational data model.

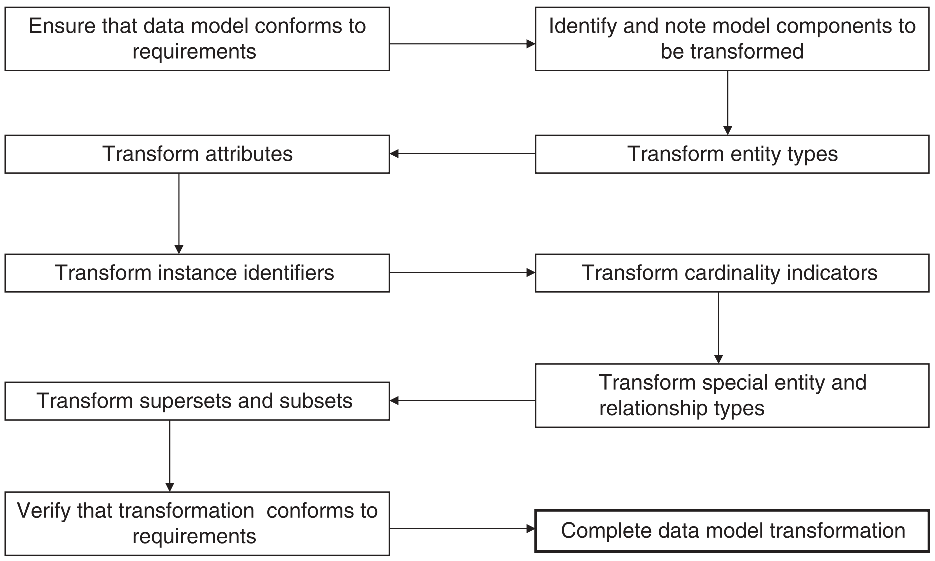

Ensure that data model conforms to requirements
Identify and note model components to be transformed
Transform entity types
Transform attributes
Transform instance identifiers
Transform cardinality indicators
Transform special entity and relationship types
Transform supersets and subsets
Verify that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

FIGURE 7-13 Model transformation: major steps.

Steps and Tasks. Figure 7-13 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.

Mapping of Components

While creating an E-R data model, the data modeler uses the components or building blocks available in that technique to put together the data model. You have studied such components in sufficient detail. Similarly, in order to create a relational model, the building blocks are the ones available in the relational modeling technique. You reviewed these components also. Essentially, transforming an E-R 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 E-R and the relational data models. The list of components makes it easier to begin the study of component mapping and model transformation.

Conceptual Data Model
ENTITY-RELATIONSHIP TECHNIQUE
Entity types
Attributes
Keys
Relationships
Cardinality indicators
Generalization/specialization

Relational Data Model
Relations or tables
Rows
Columns
Primary key
Foreign key
Generalization/specialization

Just by going through the list of components, it is easy to form the basic concepts for mapping and transformation. The conceptual 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 represented, how the characteristics must be symbolized, 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 conceptual data model, one by one, and finding the corresponding component or components in the relational data model.

Entity Types to Relations

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

Figure 7-14 shows the mapping of entity type EMPLOYEE. The mapping shows the transformation of entity type represented in E-R modeling notation to a relation denoted in relational data model notation.

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

- Entity type is transformed into a relation.
- Name of the entity type becomes the name of the relation.
- The entity instances perceived as present inside the entity type box transform into the rows of the relation.
- The complete set of entity 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.

Attributes to Columns

Entities have intrinsic or inherent characteristics. So, naturally the next component to be considered is the set of attributes of an entity type. Figure 7-15 shows the transformation of attributes.

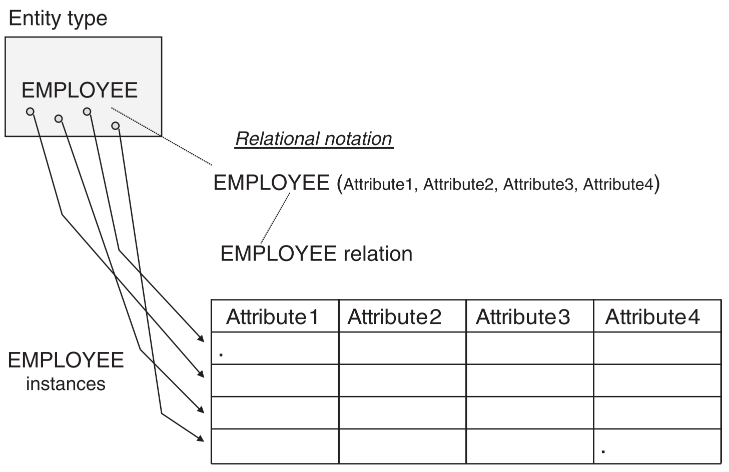
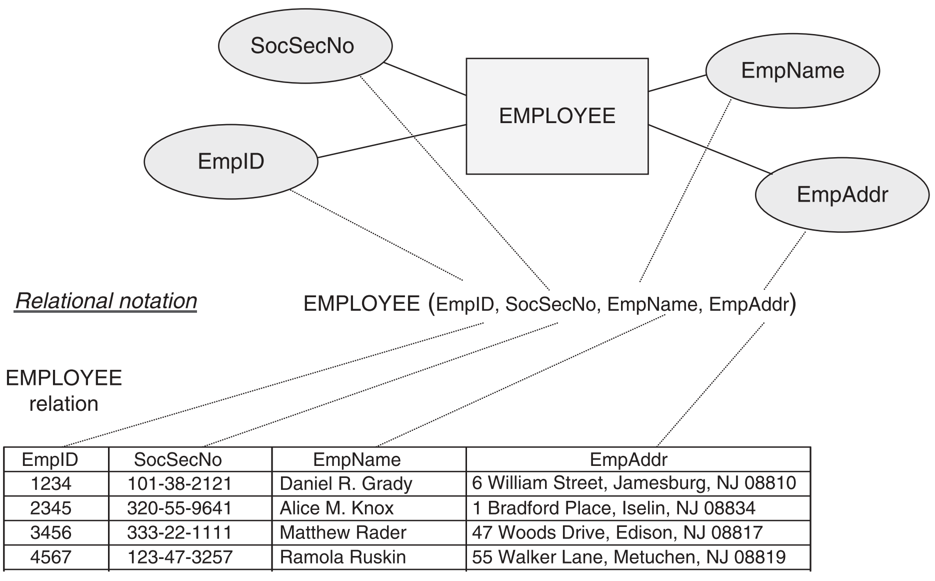


FIGURE 7-14 Mapping of entity type.

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

- Attributes of an entity type are transformed into the columns of the corresponding relation.
- The names of the attributes become the names of the columns.
- Domain of values of each attribute translates into the domain of values for corresponding columns.
- 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.



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 7-15 Mapping of attributes.

- If a multivalued attribute is present, then 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 incorporated as the number of component attributes.

Identifiers to Keys

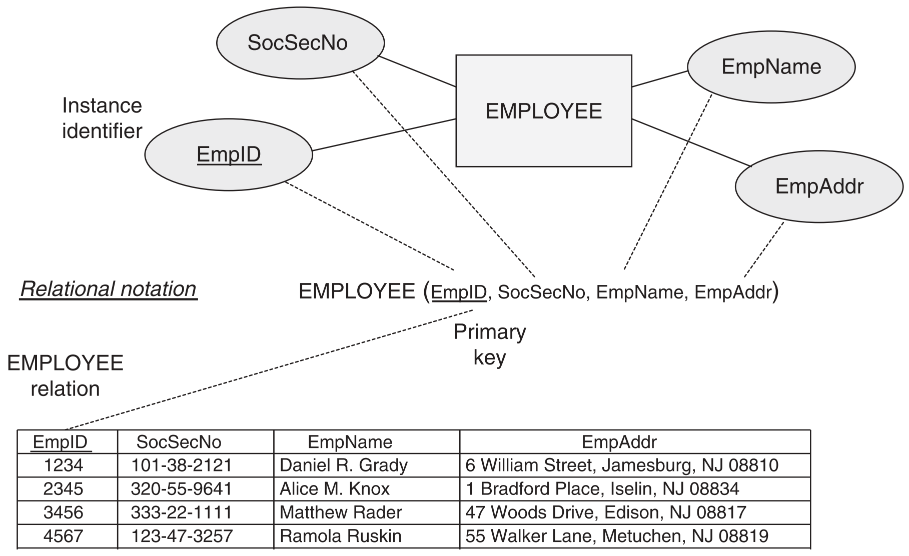
In the E-R data model, each instance of an entity type is uniquely identified by values in one or more attributes. These attributes together form the instance identifier. Figure 7-16 indicates the transformation of instance identifiers.

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 value in these columns for each row is 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

Methods for conceptual data modeling have elegant ways for representing relationships between two entity types. Wherever you perceive direct associations between instances



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 7-16 Mapping of instance identifiers.

of two entity types, the two entity types are connected by lines with a diamond in the middle containing the name of the relationship. How many instances of one entity type are associated with how many instances of the other? The 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 a relational data model establishes relationships between two relations through foreign keys. Therefore, transformation of relationships as represented in the conceptual 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 entity type is associated with a maximum of only one instance of another entity type, we call this relationship a one-to-one relationship. Figure 7-17 shows a one-to-one relationship between the two entity types CLIENT and CONTACT-PERSON.

If a client of an organization has designated a contact person, then the contact person is represented by CONTACT-PERSON entity type. 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 CONTACT-PERSON entity type. Now we can show the relationship by placing the foreign key column in CLIENT relation. Figure 7-18 illustrates this transformation.

Observe how the transformation gets done. How are the rows of CLIENT relation linked to corresponding rows of 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 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 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 CONTACT-PERSON relation, not in CLIENT relation. Figure 7-19 presents this better transformation.

Foreign key 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 7-19, examine how results for the following queries are obtained.

Who Is the Contact Person for Client Number 22222?. Read CONTACT-PERSON table by values in the foreign key column. Find the row having the value 22222 in the foreign key column.

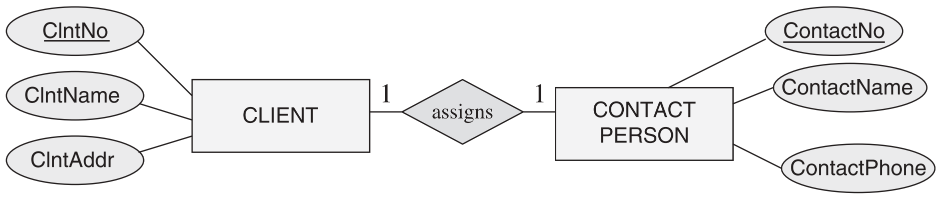
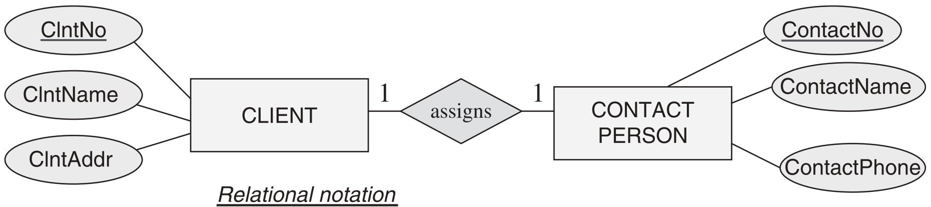


FIGURE 7-17 One-to-one relationship.



Relational notation

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

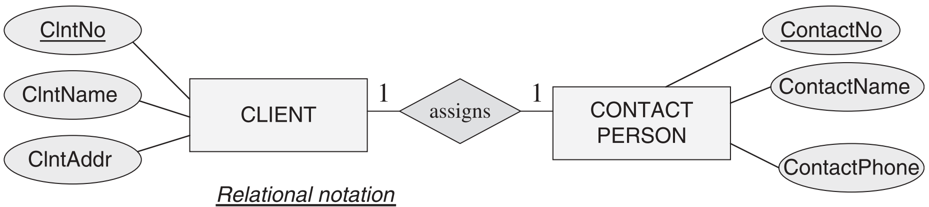
CLIENT relation

<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>Contact No</u>	ContactName	ContactPhone
123	Mary Williams	732-345-8100
234	Winston Poyser	732-555-4000
345	Lisa Moore	732-767-5300

FIGURE 7-18 Transformation of one-to-one relationship.



Relational notation

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

CLIENT relation

<u>ClntNo</u>	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

<u>ContactNo</u>	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 7-19 Better transformation of one-to-one relationship.

Who Is the Client for Contact Person Number 345?. Read CONTACT-PERSON table by values in the primary key column. Find the row having the value 345 in the primary key column. Get the foreign key value of this row, namely, 55555. Read 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 about transformation of one-to-one relationships.

- When two relations are in 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 matches 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.
- 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 the less number of rows.

One-to-Many Relationships. Let us begin our discussion of one-to-many relationship by reviewing Figure 7-20. This figure shows the one-to-many relationship between the two objects CUSTOMER and ORDER.

The figure also indicates how individual instances of these two entity types 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 one-to-one relationship, you 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 CUSTOMER relation. Figure 7-21 shows this transformation of 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

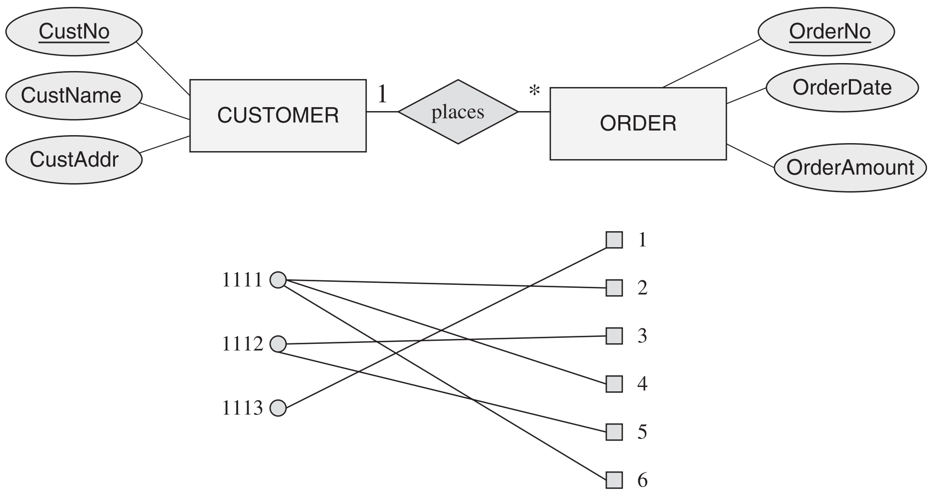
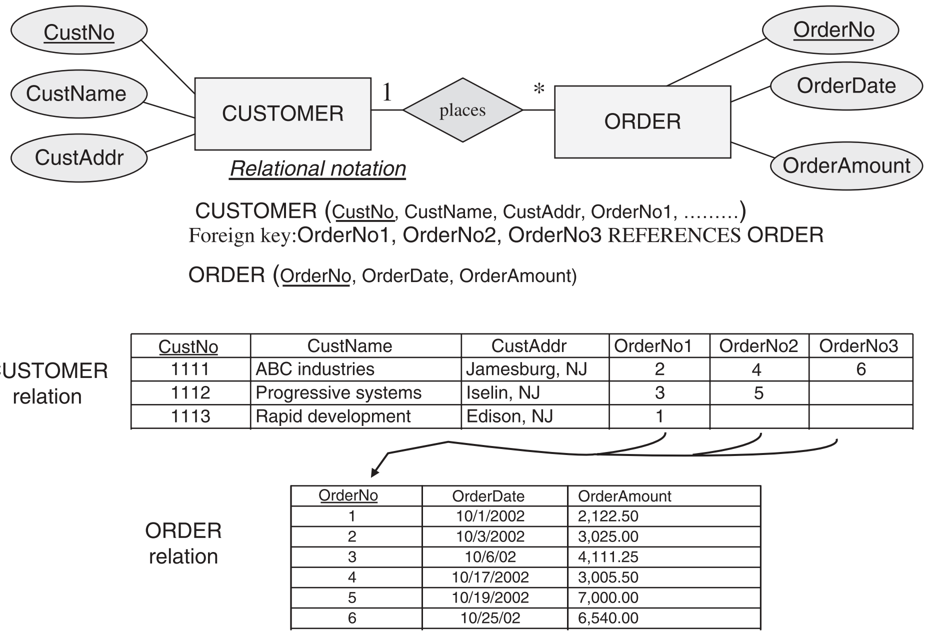


FIGURE 7-20 CUSTOMER and ORDER: one-to-many relationship.



CUSTOMER (<u>CustNo</u>, CustName, CustAddr, OrderNo1, ...)
Foreign key:OrderNo1, OrderNo2, OrderNo3 REFERENCES ORDER

ORDER (<u>OrderNo</u>, OrderDate, OrderAmount)

CUSTOMER relation

<u>CustNo</u>	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

<u>OrderNo</u>	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 7-21 Transformation of one-to-many relationship.

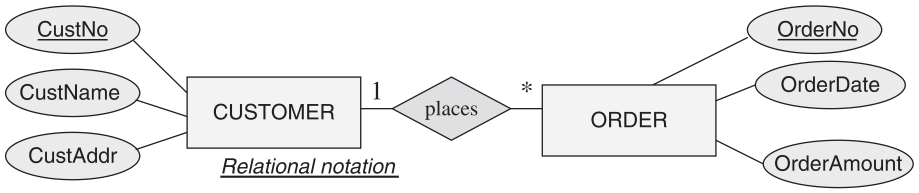
to order 1 in the ORDER relation. But the row for customer 1112 seems to need two foreign key columns, and the row for customer 1111 seems to require three foreign key columns. What if there is a customer with 50 orders? How many foreign key columns are sufficient in the CUSTOMER relation? How will you search for a particular ORDER from the several foreign key columns in the CUSTOMER relation? Obviously, this transformation is not right.

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 7-22 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 ORDER relation have the same value in the foreign key column. This indicates the several orders related to 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 ORDER table by values in the foreign key column. Find the rows having the value 1112 in the foreign key column.

What Is the Name of the Customer for Order Number 5? Read ORDER table by values in the primary key column. Find the row having the value 5 for the primary key attribute. Get foreign key value of this row, namely, 1112. Read CUSTOMER table by values in its primary key column. Find the row having the value 1112 in the primary key column.



Relational notation

CUSTOMER (<u>CustNo</u> , CustName, CustAddr)

ORDER(<u>OrderNo</u>, OrderDate, OrderAmount, CustNo)

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 7-22 Correct transformation of one-to-many relationship.

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

- When two relations are in one-to-many relationship, place the foreign key column in the relation that is on the "many" side of the relationship. Values in 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 entity type is related to one or more instances of a second entity type, and also one instance of the second entity type is related to one or more instances of the first entity type. Figure 7-23 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 E-R data model to a relational data model and establish the many-to-many relationship. For establishing the relationship, you have to create foreign key columns. While transforming a one-to-many relationship, we placed the foreign key column in the relation on the "many" side of the relationship; that is, we placed the foreign key column 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 7-23. Transform the entity types into relations and place the foreign key column in PROJECT relation. Figure 7-24 shows this transformation with the foreign key column placed in PROJECT relation.

Note the foreign keys in the transformed relations? In PROJECT relation, the rows for projects 1 and 4 need three foreign key columns, whereas the rows for projects 2, 3, and 4

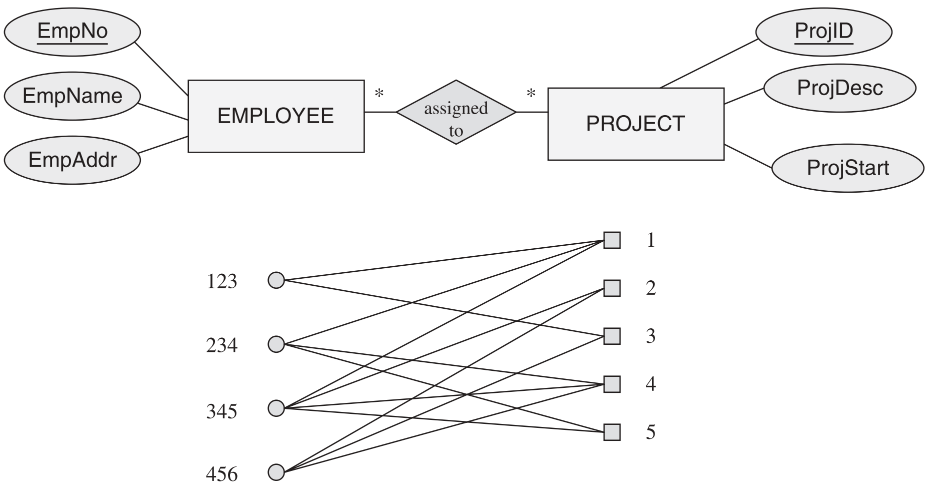
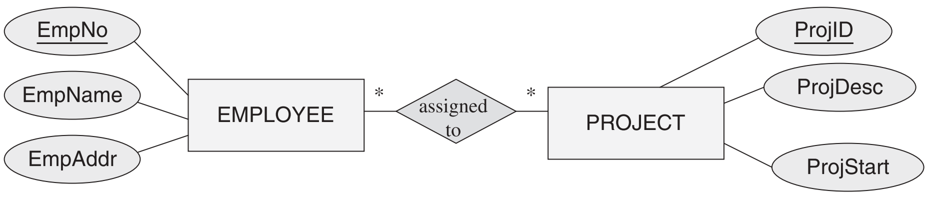


FIGURE 7-23 Example of many-to-many relationship.

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 PROJECT relation have? So, 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 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.



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 7-24 Transformation of many-to-many relationship: first method.

What Are the Names of Employees Assigned to Project 1?. Read PROJECT table by values in the primary key column. Find the row having the value 1 for the primary key attribute. Get foreign key values of this row, namely, 123, 234, and 345. Read EMPLOYEE table by values in the 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.

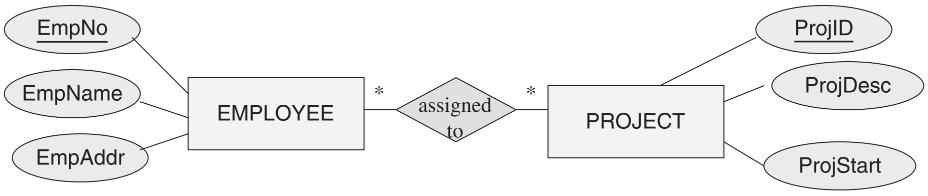
Because the transformation from the first method does not work, let us try another solution by placing the foreign key columns in the EMPLOYEE relation instead of including the foreign key columns in the other related table. Figure 7-25 illustrates this method of transformation.

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 rows for employee 345 needs four foreign key columns. By the reasoning similar to the one 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 EMPLOYEE table by values in the primary key column Find the row having the value 456 for the primary key attribute. Get foreign key values of this row, namely, 2, 3, and 4. Read 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 EMPLOYEE table by values in the foreign key columns But which foreign columns? All of the foreign



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 7-25 Transformation of many-to-many relationship: second method.

columns? Right away, you note that finding the result for this query is going to be very difficult.

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 in Figures 7-24 and 7-25. 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. But, 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—whether 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 technique is the standard method for representing many-to-many relationships in the relational data model.

Figure 7-26 illustrates the correct method of transforming many-to-many relationship. The table containing the pairs of related values of primary keys is known as the intersection table.

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 7-26 Transformation of many-to-many relationship: correct method.

Note the primary key for the intersection table. The primary key consists of two parts: one part, the primary key of PROJECT table and the other part the primary key of EMPLOYEE table. The two parts act separately as the foreign keys 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 relation 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 intersection table by values in one part of the primary key column, namely, EmpNo attribute showing values for employee key numbers. Find the rows having the value 456 for this part of the primary key. Read PROJECT table by values in its primary key column. Find the rows having the values 2, 3, and 4 for primary key attribute. Getting the result for this query seems to be workable.

What Are the Names of Employees Assigned to Project 1? Read intersection table by values in one part of the primary key column, namely, ProjID attribute showing values for project key numbers. Find the rows having the value 1 for this part of the primary key. Read EMPLOYEE table by values in its primary key column. Find the rows having the values 123, 234, and 345 for 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 column for the intersection table. The primary key column of the 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 many-to-many relationship is reduced to creating two one-to-many relationships.

Mandatory and Optional Conditions. The conceptual 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.

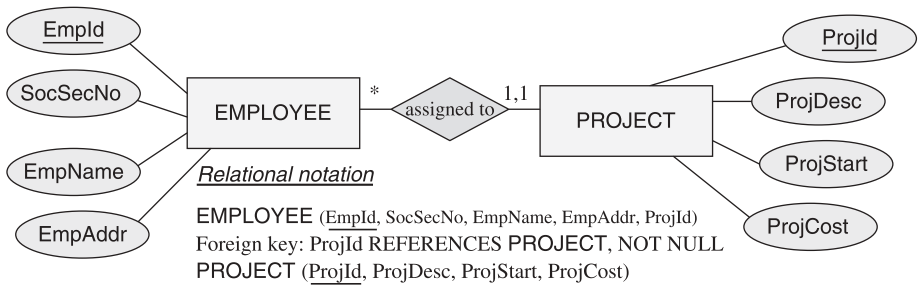
Minimum Cardinality in One-to-Many Relationship. Figure 7-27 shows an example of one-to-many relationship between the two entity types PROJECT and EMPLOYEE.

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

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



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 7-27 One-to-many relationship: mandatory and optional.

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

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.

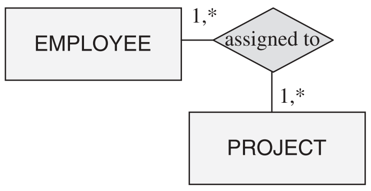
It follows, therefore, that in an optional relationship of this sort, nulls may be allowed in the foreign key attribute. What do the rows with null foreign key attribute in the EMPLOYEE relation represent? These rows represent those employees who are not assigned to a project.

Minimum Cardinality in Many-to-Many Relationship. Figure 7-28 shows an example of many-to-many relationship between the two entity types PROJECT and EMPLOYEE.

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

An employee may be assigned to many projects.

A project may have many employees.



Relational notation

EMPLOYEE (<u>EmpNo</u>, EmpName, EmpAddr)

PROJECT (<u>ProjID</u>, ProjDesc, ProjStart)

ASSIGNMENT (<u>ProjID, EmpNo</u>)

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 7-28 Many-to-many relationship: minimum cardinality.

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 relations 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. Notice 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 PROJECT and EMPLOYEE entity types. Then the indicators will represent the following condition:

An employee may be assigned to many projects.

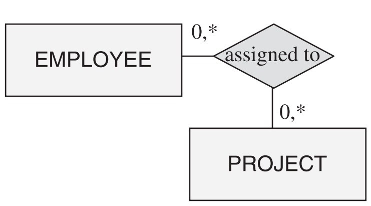
A project may have many employees.

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 needs to have an employee. 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, nulls may be allowed in the foreign key columns. However, in the way the transformation is represented in Figure 7-28, allowing nulls in foreign key columns would present a problem. You have noted the foreign key attributes form the primary key of the intersection relation, and no part of a primary key in a relation can have nulls 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 7-29.

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.



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 7-29 Many-to-many relationship: alternative approach.

Aggregate Objects as Relationships. Recall that in relationships, the participating entity types together form an aggregate entity type by virtue of the relationship itself. Let us discuss how such aggregate entity types are transformed into the components of a relational data model. Figure 7-30 illustrates such a transformation of an aggregate entity type ASSIGNMENT.

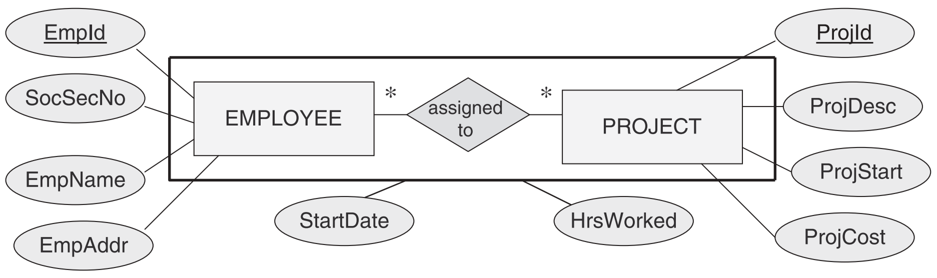
Notice the intersection relation and the attributes shown in this relation. These are the attributes of the aggregate entity type. You will note that the aggregate entity type becomes the intersection relation.

Identifying Relationship. While discussing conceptual data modeling, you 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 7-31 illustrates the transformation of an identifying relationship. Especially note the primary key attributes of the weak entity type.

Supersets and Subsets. While creating conceptual 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 entity types may be generalized in supertype entity types. Each subset of a superset forms a special relationship with its superset.

Figure 7-32 shows the transformation of a superset and its subsets. Notice how the primary key attribute and other attributes migrate from the superset relation to subset relations.



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 WoodsDrive, Edison,NJ 08817
4567	123-47-3257	Ramola Ruskin	55 Walker Lane,Metuchen, NJ 08819

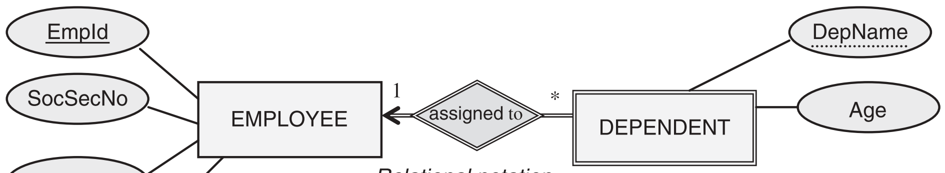
PROJECT relation

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

ASSIGNMENT relation

ProjId	EmpId	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 7-30 Transformation of aggregate entity type.



Relational notation

EMPLOYEE (<u>EmpId</u>, SocSecNo, EmpName, EmpAddr)

DEPENDENT (<u>EmpId</u>, <u>DepName</u>, Age)

Foreign key: EmpId REFERENCES EMPLOYEE

EMPLOYEE relation

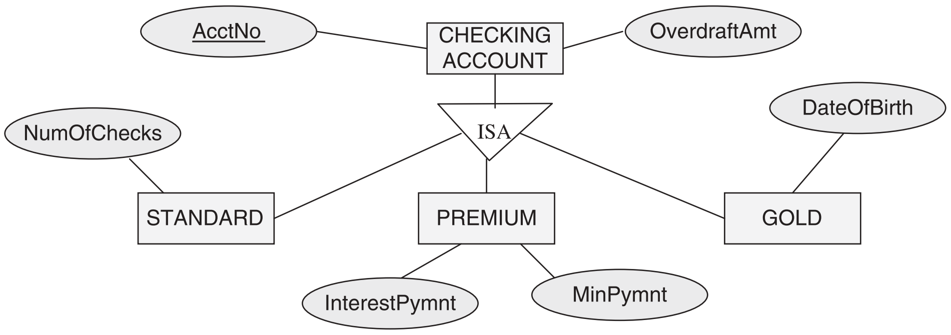
EmpId	SocSecNo	EmpName	Emp Addr
1234	101-38-2121	DanielR.Grady	6 William Street, Jamesburg,NJ 08810
2345	320-55-9641	Alice M. Knox	1 Bradford Place,Iselin, NJ 08834
3456	333-22-1111	MatthewRader	47 Woods Drive, Edison,NJ 08817
4567	123-47-3257	Ramola Ruskin	55 Walker Lane,Metuchen, NJ 08819

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 7-31 Transformation of identifying relationship.



Relational notation

CHECKINGACCOUNT (<u>AcctNo</u>, OverdraftAmt)
STANDARD (<u>AcctNo</u>, OverdraftAmt, NumOfChecks)
Foreign key: AcctNo REFERENCES CHECKINGACCOUNT
PREMIUM (<u>AcctNo</u>, OverdraftAmt, InterestPymnt, MinPymnt)
Foreign key: AcctNo REFERENCES CHECKINGACCOUNT
GOLD (<u>AcctNo</u>, OverdraftAmt, DateOfBirth)
Foreign key: AcctNo REFERENCES CHECKINGACCOUNT

FIGURE 7-32 Transformation of superset and subsets.

Transformation Summary

By now, you have a fairly good grasp of the principles of transformation of a conceptual data model into a relational data model. We took each component of the conceptual data model and reviewed how the component is transformed into a component in the relational model. Let us list the components of the conceptual data model and note how each component gets transformed.

Components of the conceptual data model and how they are transformed into relational data model:

Entity Type

STRONG

Transform into relation.

WEAK

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

Attribute

Transform into column.
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 relationship through a foreign key attribute in either of the two participating relations.

ONE-TO-MANY

Establish 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 in between the participating relations. Establish relationship through foreign key attributes in the intersection relation.

OPTIONAL AND MANDATORY CONDITIONS

Set constraint for the foreign key column. If nulls are not allowed in the foreign key column, it represents a mandatory relationship. Allowing nulls 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.

CHAPTER SUMMARY

- The relational model may be used as a logical data model. The relational model is a popular and widely used model that is superior to the earlier hierarchical and network models.
- The relational model rests on a solid mathematical foundation: it uses the concepts of matrix operations and set theory.
- The relation or two-dimensional table is the single modeling concept in the relational model.
- The columns of a relation or table denote the attributes and the rows represent the instances of an entity type.
- Relationships are established through foreign keys.
- Entity integrity, referential integrity, and functional dependency rules enforce data integrity in a relational model.
- There are two approaches to design from modeling: model transformation method and traditional normalization method.
- Model transformation method from conceptual to logical data model: entity types to relations, attributes to columns, identifiers to keys, relationships through foreign key columns.
- One-to-one and one-to-many relationships are transformed by introducing a foreign key column in the child relation.
- Many-to-many relationships are transformed by the introduction of another intersection relation.
- Optional and mandatory conditions in a relationship are indicated by allowing or disallowing nulls in foreign key columns of relations.

REVIEW QUESTIONS

1. Match the column entries:

1. Relation tuples	A. Primary key
2. Foreign key	B. Conceptual to logical
3. Row uniqueness	C. Relation
4. Entity integrity	D. Column in separate relation
5. Model transformation	E. Entity instances
6. Entity type	F. Primary key not null
7. Identifier	G. Order not important
8. Optional condition	H. Establish logical link
9. Multivalued attribute	I. Nulls in foreign key
10. Relation columns	J. No duplicate rows

2. Show an example to illustrate how mathematical set theory is used for data manipulation in the relational data model.

3. What is a mathematical relation? Explain how it is used in the relational model to represent an entity type.
4. Describe in detail how columns in a relation are used to represent attributes. Give examples.
5. Using an example, illustrate how foreign key columns are used to establish relationships in the relational data model.
6. Discuss the referential integrity rule in the relational model. Provide an example to explain the rule.
7. What are the two design approaches to create a logical data model? What are the circumstances under which you will prefer one to another?
8. Describe the features of the model transformation method.
9. Describe how many-to-many relationships are transformed into the relational model. Provide a comprehensive example.
10. Discuss the transformation of a one-to-one relationship. Indicate with an example where the foreign key column must be placed.