chaper 2

Database Design

chaper 3 Objectives

- To understand basic relational terminology.
- To understand the characteristics of relations.
- To understand alternative terminology used in describing the relational model
- To be able to identify functional dependencies, determinants, and dependent attributes
- To identify primary, candidate, and composite keys
- To be able to identify possible insertion, deletion, and update anomalies in a relation
- To be able to place a relation into BCNF normal form
- To understand the special importance of domain/key normal form
- To be able to identify multivalued dependencies
- To be able to place a relation in fourth normal form

Contents

As we discussed in Chapter 1, databases arise from three sources: from existing data, from the development of new information systems, and from the redesign of existing databases. In this chapter and the next, we consider the design of databases from existing data, such as data from spreadsheets or extracts of existing databases.

The premise of Chapters 3 and 4 is that you have received one or more tables of data from some source that are to be stored in a new database. The question is: Should this data be stored as is, or should it be transformed in some way before it is stored? For example, consider the two tables in the top part of Figure 3-1. These are the SKU_DATA and ORDER_ITEM tables extracted from the Cape Codd Outdoor Sports database as used in the database in Chapter 2.

Figure3-1

How Many Tables?

ORDER_ITEM

	OrderNumber	SKU	Quantity	Price	ExtendedPrice
1	1000	201000	1	300.00	300.00
2	1000	202000	1	130.00	130.00
3	2000	101100	4	50.00	200.00
4	2000	101200	2	50.00	100.00
5	3000	100200	1	300.00	300.00
6	3000	101100	2	50.00	100.00
7	3000	101200	1	50.00	50.00

SKU_DATA

	SKU	SKU_Description	Department	Buyer
1	100100	Std. Scuba Tank, Yellow	Water Sports	Pete Hansen
2	100200	Std. Scuba Tank, Magenta	Water Sports	Pete Hansen
3	101100	Dive Mask, Small Clear	Water Sports	Nancy Meyers
4	101200	Dive Mask, Med Clear	Water Sports	Nancy Meyers
5	201000	Half-dome Tent	Camping	Cindy Lo
6	202000	Half-dome Tent Vestibule	Camping	Cindy Lo
7	301000	Light Fly Climbing Harness	Climbing	Jerry Martin
8	302000	Locking carabiner, Oval	Climbing	Jerry Martin

SKU_ITEM

	OrderNumber	SKU	Quantity	Price	SKU_Description	Department	Buyer
1	1000	201000	1	300.00	Half-dome Tent	Camping	Cindy Lo
2	1000	202000	1	130.00	Half-dome Tent Vestibule	Camping	Cindy Lo
3	2000	101100	4	50.00	Dive Mask, Small Clear	Water Sports	Nancy Meyers
4	2000	101200	2	50.00	Dive Mask, Med Clear	Water Sports	Nancy Meyers
5	3000	100200	1	300.00	Std. Scuba Tank, Magenta	Water Sports	Pete Hansen
6	3000	101100	2	50.00	Dive Mask, Small Clear	Water Sports	Nancy Meyers
7	3000	101200	1	50.00	Dive Mask, Med Clear	Water Sports	Nancy Meyers

You can design the new database to store this data as two separate tables, or you can join the tables together and design the database with just one table. Each alternative has advantages and disadvantages. When you make the decision to use one design, you obtain certain advantages at the expense of certain costs. The purpose of this chapter is to help you understand those advantages and costs.

Such questions do not seem difficult, and you may be wondering why we need two chapters to answer them. In truth, even a single table can have surprising complexity. Consider, for example, the table in Figure 3-2, which shows sample data extracted from a corporate database. This simple table has three columns: the buyer's name, the SKU of the products that the buyer purchases, and the names of the buyer's college major(s). Buyers manage more than one SKU, and they can have multiple college majors.

To understand why this is an odd table, suppose that Nancy Meyers is assigned a new SKU, say 101300. What addition should we make to this table? Clearly, we need to add a row for the new SKU, but if we add just one row, say the row ('Nancy Meyers', 101300, 'Art'), it will appear that she manages product 101300 as an Art major, but not as an Info Systems major. To avoid such an illogical state, we need to add two rows: ('Nancy Meyers', 101300, 'Art') and ('Nancy Meyers', 101300, 'Info Systems').

Figure3-2

PRODUCT_BUYER—A Very Strange Table

PRODUCT BUYER

	BuyerName	SKU_Managed	CollegeMajor
1	Pete Hansen	100100	Business Administration
2	Pete Hansen	100200	Business Administration
3	Nancy Meyers	101100	Art
4	Nancy Meyers	101100	Info Systems
5	Nancy Meyers	101200	Art
6	Nancy Meyers	101200	Info Systems
7	Cindy Lo	201000	History
8	Cindy Lo	202000	History
9	Jenny Martin	301000	Business Administration
10	Jenny Martin	301000	English Literature
11	Jenny Martin	302000	Business Administration
12	Jenny Martin	302000	English Literature

This is a strange requirement. Why should we have to add two rows of data simply to record the fact that a new SKU has been assigned to a buyer? Further, if we assign the product to Pete Hansen instead, we would only have to add one row, but if we assigned the product to a buyer who had four majors, we would have to add four new rows.

The more one thinks about the table in Figure 3-2, the more strange it becomes. What changes should we make if SKU 101100 is assigned to Pete Hansen? What changes should we make if SKU 100100 is assigned to Nancy Meyers? What should we do if all the SKU values in Figure 3-2 are deleted? Later in this chapter, you will learn that these problems arise because this table has a problem called a multivalued dependency. Even better, you will learn how to remove that problem.

Tables can have many different patterns; some patterns are susceptible to serious problems and other patterns are not. Before we can address this question, however, you need to learn some basic terms.

1、Relational Model Terminology

Figure 3-3 lists the most important terms used by the relational model. By the time you finish Chapters 3 and 4, you should be able to define each of these terms and explain how each pertains to the design of relational databases. Use this list of terms as a check on your comprehension.

Figure3-3

Important Relational Model Terms

- Relation
- Functional dependency
- Determinant
- Candidate key
- Composite key
- Primary key
- Surrogate key
- Foreign key
- Referential integrity constraint
- Normal form
- Multivalued dependency

1、Relational Model Terminology

➢**Relations**

So far, we have used the terms table and relation interchangeably. In fact, a relation is a special case of a table. This means that all relations are tables, but not all tables are relations. Codd defined the characteristics of a relation in his 1970 paper that laid the foundation for the relational model.[1] Those characteristics are summarized in Figure 3-4.

Figure3-4 Characteristics of Relations

Characteristics of Relations
Rows contain data about an entity.
Columns contain data about attributes of the entities.
All entries in a column are of the same kind.
Each column has a unique name.
Cells of the table hold a single value.
The order of the columns is unimportant.
The order of the rows is unimportant.
No two rows may be identical.

1、Relational Model Terminology

➢**Relations**

By the way In Figure 3-4 and in this discussion, we use the term entity to mean Some identifiable thing. A customer, a salesperson, an order, a part, and a lease are all examples of what we mean by an entity. When we introduce the entity-relationship model in Chapter 5, we will make the definition of entity more precise. For now, just think of an entity as some identifiable thing that users want to track.

1、Relational Model Terminology

➢ **Characteristics of Relations**

Figure 3-5 Sample EMPLOYEE Relation

EmployeeNumber	FirstName	LastName	Department	Email	Phone
100	Jerry	Johnson	Accounting	JJ@somewhere.com	834-1101
200	Mary	Abernathy	Finance	MA@somewhere.com	834-2101
300	Liz	Smathers	Finance	LS@somewhere.com	834-2102
400	Tom	Caruthers	Accounting	TC@somewhere.com	834-1102
500	Tom	Jackson	Production	TJ@somewhere.com	834-4101
600	Eleanore	Caldera	Legal	EC@somewhere.com	834-3101
700	Richard	Bandalone	Legal	RB@somewhere.com	834-3102

1、Relational Model Terminology

➢**Characteristics of Relations**

A relation has a specific definition, as shown in Figure 3-4, and for a table to be a relation the criteria of this definition must be met. First, the rows of the table must store data about an entity and the columns of the table must store data about the characteristics of those entities. Further, in a relation all of the values in a column are of the same kind. If, for example, the second column of the first row of a relation has FirstName, then the second column of every row in the relation has FirstName. Also, the names of the columns are unique; no two columns in the same relation may have the same name. The EMPLOYEE table shown in Figure 3-5 meets these criteria and is a relation.

1、Relational Model Terminology

➢Characteristics of Relations

Each cell of a relation has only a single value or item; multiple entries are not allowed. The table in Figure 3-6 is not a relation, because the Phone values of employees Caruthers and Bandalone store multiple phone numbers.

In a relation, the order of the rows and the order of the columns are immaterial. No information can be carried by the ordering of rows or columns. The table in Figure 3-7 is not a relation, because the entries for employees Caruthers and Caldera require a particular row arrangement. If the rows in this table were rearranged, we would not know which employee has the indicated Fax and Home numbers.

Finally, according to the last characteristic in Figure 3-4, for a table to be a relation no two rows can be identical. As you learned in Chapter 2, some SQL statements do produce tables with duplicate rows. In such cases, you can use the DISTINCT keyword to force uniqueness. Such row duplication only occurs as a result of SQL manipulation. Tables that you design to be stored in the database should never contain duplicate rows.

1、Relational Model Terminology

- **Characteristics of Relations**

Figure 3-6 Nonrelational Table—Multiple Entries per Cell

EmployeeNumber	FirstName	LastName	Department	Email	Phone
100	Jerry	Johnson	Accounting	JJ@somewhere.com	834-1101
200	Mary	Abernathy	Finance	MA@somewhere.com	834-2101
300	Liz	Smathers	Finance	LS@somewhere.com	834-2102
400	Tom	Caruthers	Accounting	TC@somewhere.com	834-1102, 834-1191, 834-1192
500	Tom	Jackson	Production	TJ@somewhere.com	834-4101
600	Eleanore	Caldera	Legal	EC@somewhere.com	834-3101
700	Richard	Bandalone	Legal	RB@somewhere.com	834-3102, 834-3191

1、Relational Model Terminology

➢**Characteristics of Relations**

Figure 3-7

Nonrelational Table—Order of Rows Matters and Kind of Column Entries Differs in Email

EmployeeNumber	FirstName	LastName	Department	Email	Phone
100	Jerry	Johnson	Accounting	JJ@somewhere.com	834-1101
200	Mary	Abernathy	Finance	MA@somewhere.com	834-2101
300	Liz	Smathers	Finance	LS@somewhere.com	834-2102
400	Tom	Caruthers	Accounting	TC@somewhere.com	834-1102
				Fax:	834-9911
				Home:	723-8795
500	Tom	Jackson	Production	TJ@somewhere.com	834-4101
600	Eleanore	Caldera	Legal	EC@somewhere.com	834-3101
				Fax:	834-9912
				Home:	723-7654
700	Richard	Bandalone	Legal	RB@somewhere.com	834-3102

1、Relational Model Terminology

- **Characteristics of Relations**

By the way Do not fall into a common trap. Even though every cell of a relation Must have a single value, this does not mean that all values must have the same length. The table in Figure 3-8 is a relation even though the length of the Comment column varies from row to row. It is a relation because, even though the comments have different lengths, there is only one comment per cell.

1、Relational Model Terminology

➢Characteristics of Relations

Figure 3-8

Relation with Variable-Length Column Values

EmployeeNumber	FirstName	LastName	Department	Email	Phone	Comment
100	Jerry	Johnson	Accounting	JJ@somewhere.com	834-1101	Joined the Accounting Department in March after completing his MBA. Will take the CPA exam this fall.
200	Mary	Abernathy	Finance	MA@somewhere.com	834-2101	
300	Liz	Smathers	Finance	LS@somewhere.com	834-2102	
400	Tom	Caruthers	Accounting	TC@somewhere.com	834-1102	
500	Tom	Jackson	Production	TJ@somewhere.com	834-4101	
600	Eleanore	Caldera	Legal	EC@somewhere.com	834-3101	
700	Richard	Bandalone	Legal	RB@somewhere.com	834-3102	Is a full-time consultant to Legal on a retainer basis.

1、Relational Model Terminology

➢Alternative Terminology

As defined by Codd, the columns of a relation are called attributes, and the rows of a relation are called tuples (rhymes with "couples"). Most practitioners, however, do not use these academic-sounding terms and instead use the terms column and row. Also, even though a table is not necessarily a relation, most practitioners mean relation when they say table. Thus, in most conversations the terms relation and table are synonymous. In fact, for the rest of this book table and relation will be used synonymously.

Additionally, a third set of terminology also is used. Some practitioners use the terms file, field, and record for the terms table, column, and row, respectively. These terms arose from traditional data processing and are common in connection with legacy systems. Sometimes, people mix and match these terms. You might hear someone say, for example, that a relation has a certain column and contains 47 records. These three sets of terms are summarized in Figure 3-9.

1、Relational Model Terminology

➢ **Alternative Terminology**

Figure 3-9

Three Sets of Equivalent Terms

Table	Column	Row
Relation	Attribute	Tuple
File	Field	Record

1、Relational Model Terminology

➢Functional Dependencies

Functional dependencies are the heart of the database design process, and it is vital for you to understand them. We first explain the concept in general terms and then examine two examples. We begin with a short excursion into the world of algebra. Suppose you are buying boxes of cookies and someone tells you that each box costs $5.00. With this fact, you can compute the cost of several boxes with the formula:

$$\text{CookieCost} = \text{NumberOfBoxes} \times \$5$$

A more general way to express the relationship between CookieCost and NumberOfBoxes is to say that CookieCost depends on NumberOfBoxes. Such a statement tells us the character of the relationship between CookieCost and NumberOfBoxes, even though it doesn't give us the formula. More formally, we can say that CookieCost is functionally dependent on NumberOfBoxes. Such a statement can be written as:

$$\text{NumberOfBoxes} \rightarrow \text{CookieCost}$$

This expression can be read as "NumberOfBoxes determines CookieCost." The variable on the left, here NumberOfBoxes, is called the determinant.

1、Relational Model Terminology

➢Functional Dependencies

Using another formula, we can compute the extended price of a part order by multiplying the quantity of the item times its unit price, or:

ExtendedPrice = Quantity × UnitPrice

In this case, we say that ExtendedPrice is functionally dependent on Quantity and UnitPrice, or:

(Quantity, UnitPrice) → ExtendedPrice

Here, the determinant is the composite (Quantity, UnitPrice).

1、Relational Model Terminology

➢Functional Dependencies

Functional Dependencies That Are Not Equations

In general, a functional dependency exists when the value of one or more attributes determines the value of another attribute. Many functional dependencies exist that do not involve equations.

Consider an example. Suppose you know that a sack contains either red, blue, or yellow objects. Further, suppose you know that the red objects weigh 5 pounds, the blue objects weigh 5 pounds, and the yellow objects weigh 7 pounds. If a friend looks into the sack, sees an object, and tells you the color of the object, you can tell her the weight of the object. We can formalize this as:

ObjectColor → Weight

Thus, we can say that Weight is functionally dependent on ObjectColor and that ObjectColor determines Weight. The relationship here does not involve an equation, but the functional dependency holds. Given a value for ObjectColor, you can determine the object's weight.

1、Relational Model Terminology

➢Functional Dependencies

Functional Dependencies That Are Not Equations

If we also know that the red objects are balls, the blue objects are cubes, and the yellow objects are cubes, we can also say:

ObjectColor →Shape

Thus, ObjectColor determines Shape. We can put these two together to state:

ObjectColor → (Weight, Shape)

Thus, ObjectColor determines Weight and Shape.

Another way to represent these facts is to put them into a table:

Object Color	Weight	Shape
Red	5	Ball
Blue	5	Cube
Yellow	7	Cube

1、Relational Model Terminology

➢Functional Dependencies

Functional Dependencies That Are Not Equations

This table meets all of the conditions listed in Figure 3-4, and therefore it is a relation. You may be thinking that we performed a trick or sleight of hand to arrive at this relation, but, in truth, the only reason for having relations is to store instances of functional dependencies. If there were a formula by which we could take ObjectColor and somehow compute Weight and Shape, then we would not need the table. We would just make the computation. Similarly, if there were a formula by which we could take EmployeeNumber and compute EmployeeName and HireDate, then we would not need an EMPLOYEE relation. However, because there is no such formula, we must store the combinations of EmployeeNumber, EmployeeName, and HireDate in the rows of a relation.

1、Relational Model Terminology

➢Functional Dependencies

Composite Functional Dependencies

The determinant of a functional dependency can consist of more than one attribute. For example, a grade in a class is determined by both the student and the class, or:

(StudentNumber, ClassNumber) →Grade

In this case, the determinant is called a composite determinant.

Notice that both the student and the class are needed to determine the grade. In general, if (A, B) →C, then neither A nor B will determine C by itself. However, if A → (B, C), then it is true that A →B and A →C. Work through examples of your own for both of these cases so that you understand why this is true.

1、Relational Model Terminology

➢Finding Functional Dependencies

To fix the idea of functional dependency in your mind, consider what functional dependencies exist in the SKU_DATA and ORDER_ITEM tables in Figure 3-1.

Functional Dependencies in the SKU_DATA Table

To find functional dependencies in a table, we must ask “Does any column determine the value of another column?” For example, consider the values of the SKU_DATA table in Figure 3-1:

	SKU	SKU_Description	Department	Buyer
1	100100	Std. Scuba Tank, Yellow	Water Sports	Pete Hansen
2	100200	Std. Scuba Tank, Magenta	Water Sports	Pete Hansen
3	101100	Dive Mask, Small Clear	Water Sports	Nancy Meyers
4	101200	Dive Mask, Med Clear	Water Sports	Nancy Meyers
5	201000	Half-dome Tent	Camping	Cindy Lo
6	202000	Half-dome Tent Vestibule	Camping	Cindy Lo
7	301000	Light Fly Climbing Harness	Climbing	Jerry Martin
8	302000	Locking carabiner, Oval	Climbing	Jerry Martin

1、Relational Model Terminology

➢Finding Functional Dependencies

Consider the last two columns. If we know the value of Department, can we determine a unique value of Buyer? No, we cannot, because a Department may have more than one Buyer. In this sample data, ‘Water Sports’ is associated with Pete Hansen and Nancy Meyers. Therefore, Department does not functionally determine Buyer.

What about the reverse? Does Buyer determine Department? In every row, for a given value of Buyer, do we find the same value of Department? Every time Jerry Martin appears, for example, is he paired with the same department? The answer is yes. Further, every time Cindy Lo appears, she is paired with the same department. The same is true for the other buyers. Therefore, assuming that these data are representative, Buyer does determine Department, and we can write:

Buyer → Department

Does Buyer determine any other column? If we know the value of Buyer, do we know the value of SKU? No, we do not, because a given buyer has many SKUs assigned to him or her. Does Buyer determine SKU_Description? No, because a given value of Buyer occurs with many values of SKU_Description.

1、Relational Model Terminology

➢Finding Functional Dependencies

By the way As stated, for the Buyer : Department functional dependency a Buyer is paired with one and only one value of Department. Notice that a buyer can appear more than once in the table, but, if so, that buyer is always paired with the same department. This is true for all functional dependencies. If A : B, then each value of A will be paired with one and only one value of B. A particular value of A may appear more than once in the relation, but, if so, it is always paired with the same value of B. Note, too, that the reverse is not necessarily true. If A → B, then a value of B may be paired with many values of A.

1、Relational Model Terminology

➢Finding Functional Dependencies

What about the other columns? It turns out that if we know the value of SKU, we also know the values of all of the other columns. In other words:

SKU→SKU_Description

because a given value of SKU will have just one value of SKU_Description. Next,

SKU →Department

because a given value of SKU will have just one value of Department. And, finally,

SKU→Buyer

because a given value of SKU will have just one value of Buyer.

1、Relational Model Terminology

➢Finding Functional Dependencies

We can combine these three statements as:

SKU→ (SKU_Description, Department, Buyer)

For the same reasons, SKU_Description determines all of the other columns, and we can write:

SKU_Description → (SKU, Department, Buyer)

In summary, the functional dependencies in the SKU_DATA table are:

SKU → (SKU_Description, Department, Buyer)
SKU_Description → (SKU, Department, Buyer)
Buyer →Department

1、Relational Model Terminology

➢**Finding Functional Dependencies**

By the way You cannot always determine functional dependencies from sample data. You may not have any sample data, or you may have just a few rows that are not representative of all of the data conditions. In such cases, you must ask the users who are experts in the application that creates the data. For the SKU_DATA table, you would ask questions such as, “Is a Buyer always associated with the same Department?” and “Can a Department have more than one Buyer?” In most cases, answers to such questions are more reliable than sample data. When in doubt, trust the users.

1、Relational Model Terminology

➢Finding Functional Dependencies

Functional Dependencies in the ORDER_ITEM Table

Now consider the ORDER_ITEM table in Figure 3-1. For convenience, here is a copy of the data in that table:

	OrderNumber	SKU	Quantity	Price	ExtendedPrice
1	1000	201000	1	300.00	300.00
2	1000	202000	1	130.00	130.00
3	2000	101100	4	50.00	200.00
4	2000	101200	2	50.00	100.00
5	3000	100200	1	300.00	300.00
6	3000	101100	2	50.00	100.00
7	3000	101200	1	50.00	50.00

What are the functional dependencies in this table? Start on the left. Does OrderNumber determine another column? It does not determine SKU, because several SKUs are associated with a given order. For the same reasons, it does not determine Quantity, Price, or ExtendedPrice.

1、Relational Model Terminology

➢Finding Functional Dependencies

Functional Dependencies in the ORDER_ITEM Table

What about SKU? SKU does not determine OrderNumber because several OrderNumbers are associated with a given SKU. It does not determine Quantity or ExtendedPrice for the same reason.

What about SKU and Price? From this data, it does appear that

$$SKU \rightarrow Price$$

but that might not be true in general. In fact, we know that prices can change after an order has been processed. Further, an order might have special pricing due to a sale or promotion. To keep an accurate record of what the customer actually paid, we need to associate a particular SKU price with a particular order. Thus:

$$(OrderNumber, SKU) \rightarrow Price$$

Considering the other columns, Quantity, Price, and ExtendedPrice do not determine anything else. You can decide this by looking at the sample data. You can reinforce this conclusion by thinking about the nature of sales. Would a Quantity of 2 ever determine an OrderNumber or a SKU? This makes no sense. At the grocery store, if I tell you I bought two of something, you have no reason to conclude that my OrderNumber was 1010022203466 or that I bought carrots. Quantity does not determine OrderNumber or SKU.

1、Relational Model Terminology

➢Finding Functional Dependencies

Functional Dependencies in the ORDER_ITEM Table

Similarly, if I tell you that the price of an item was $3.99, there is no logical way to conclude what my OrderNumber was or that I bought a jar of green olives. Thus, Price does not determine OrderNumber or SKU. Similar comments pertain to ExtendedPrice. It turns out that no single column is a determinant in the ORDER_ITEM table.

What about pairs of columns? We already know that

(OrderNumber, SKU) →Price

Examining the data, (OrderNumber, SKU) determines the other two columns as well. Thus:

(OrderNumber, SKU) →(Quantity, Price, ExtendedPrice)

This functional dependency makes sense. It means that given a particular order and a particular item on that order, there is only one quantity, one price, and one extended price.

1、Relational Model Terminology

➢Finding Functional Dependencies

Functional Dependencies in the ORDER_ITEM Table

Notice, too, that because ExtendedPrice is computed from the formula ExtendedPrice = (Quantity * Price) we have:

(Quantity, Price) →ExtendedPrice

In summary, the functional dependencies in ORDER_ITEM are:

(OrderNumber, SKU) → (Quantity, Price, ExtendedPrice)

(Quantity, Price) → ExtendedPrice

No single skill is more important for designing databases than the ability to identify functional dependencies. Make sure you understand the material in this section. Work problems 3.58 and 3.59 and the Marcia's Dry Cleaning and Morgan Importing projects at the end of the chapter. Ask your instructor for help if necessary. You must understand functional dependencies and be able to work with them.

1、Relational Model Terminology

➢Finding Functional Dependencies

When Are Determinant Values Unique?

In the previous section, you may have noticed an irregularity. Sometimes the determinants of a functional dependency are unique in a relation, and sometimes they are not. Consider the SKU_DATA relation, with determinants SKU, SKU_Description, and Buyer. In SKU_DATA, the values of both SKU and SKU_Description are unique in the table. For example, the SKU value 100100 appears just once. Similarly, the SKU_Description value 'Half-dome Tent' occurs just once. From this, it is tempting to conclude that values of determinants are always unique in a relation. However, this is not true.

For example, Buyer is a determinant, but it is not unique in SKU_DATA. The buyer 'Cindy Lo' appears in two different rows. In fact, for this sample data all of the buyers occur in two different rows.

In truth, a determinant is unique in a relation only if it determines every other column in the relation. For the SKU_DATA relation, SKU determines all of the other columns. Similarly, SKU_Description determines all of the other columns. Hence, they both are unique. Buyer, however, only determines the Department column. It does not determine SKU or SKU_Description.

1、Relational Model Terminology

➢Finding Functional Dependencies

When Are Determinant Values Unique?

The determinants in ORDER_ITEM are (OrderNumber, SKU) and (Quantity, Price). Because (OrderNumber, SKU) determines all of the other columns, it will be unique in the relation. The composite (Quantity and Price) only determines ExtendedPrice. Therefore, it will not be unique in the relation.

This fact means that you cannot find the determinants of all functional dependencies simply by looking for unique values. Some of the determinants will be unique, but some will not be. Instead, to determine if column A determines column B, look at the data and ask, “Every time that a value of column A appears is it matched with the same value of Column B?” If so, it can be a determinant of B. Again, however, sample data can be incomplete, so the best strategies are to think about the nature of the business activity from which the data arise and to ask the users.

1、Relational Model Terminology

- **Keys**

The relational model has more keys than a locksmith. There are candidate keys, composite keys, primary keys, surrogate keys, and foreign keys. In this section, we will define each of these types of keys. Because key definitions rely on the concept of functional dependency, make sure you understand that concept before reading on.

In general, a key is a combination of one or more columns that is used to identify particular rows in a relation. Keys that have two columns or more are called composite keys.

Candidate Keys

A candidate key is a determinant that determines all of the other columns in a relation. The SKU_DATA relation has two candidate keys: SKU and SKU_Description. Buyer is a determinant, but it is not a candidate key because it only determines Department.

The ORDER_ITEM table has just one candidate key: (OrderNumber, SKU). The other determinant in this table, (Quantity, Price), is not a candidate key because it determines only ExtendedPrice.

Candidate keys identify a unique row in a relation. Given the value of a candidate key, we can find one and only one row in the relation that has that value. For example, given the SKU value of 100100, we can find one and only one row in SKU_DATA. Similarly, given the OrderNumber and SKU values (2000, 101100), we can find one and only one row in ORDER_ITEM.

1、Relational Model Terminology

➢Keys

Primary Keys

When designing a database, one of the candidate keys is selected to be the primary key. This term is used because this key will be defined to the DBMS, and the DBMS will use it as its primary means for finding rows in a table. A table has only one primary key. The primary key can have one column or it can be a composite.

In this text, to clarify discussions we will sometimes indicate table structure by showing the name of a table followed by the names of the table's columns enclosed in parentheses. When we do this, we will underline the column(s) that comprise the primary key. For example, we can show the structure of SKU_DATA and ORDER_ITEM as follows:

```
SKU_DATA (SKU, SKU_Description, Department, Buyer)
ORDER_ITEM (OrderNumber, SKU, Quantity, Price, ExtendedPrice)
```

This notation indicates that SKU is the primary key of SKU_DATA and that (OrderNumber, SKU) is the primary key of ORDER_ITEM.

1、Relational Model Terminology

- **Keys**

By the way What do you do if a table has no candidate keys? In that case, define the primary key as the collection of all of the columns in the table. Because there are no duplicate rows in a stored relation, the combination of all of the columns of the table will always be unique. Again, although tables generated by SQL manipulation may have duplicate rows, the tables that you design to store data should never be constructed to have data duplication. Thus, the combination of all columns is always a candidate key.

1、Relational Model Terminology

➢Keys

Surrogate Keys

A surrogate key is an artificial column that is added to a table to serve as the primary key. The DBMS assigns a unique value to a surrogate key when the row is created. The assigned value never changes. Surrogate keys are used when the primary key is large and unwieldy. For example, consider the relation RENTAL_PROPERTY:

```
RENTAL_PROPERTY (Street, City, State/Province, Zip/PostalCode, Country, Rental_Rate)
```

The primary key of this table is (Street, City, State/Province, Zip/PostalCode, Country). As you will learn in Chapter 6, for good performance a primary key should be short and, if possible, numeric. The primary key of RENTAL_PROPERTY is neither.

In this case, the designers of the database would likely create a surrogate key. The structure of the table would then be:

```
RENTAL_PROPERTY (PropertyID, Street, City, State/Province, Zip/PostalCode, Country, Rental_Rate)
```

The DBMS will assign a numeric value to PropertyID when a row is created. Using that key will result in better performance than using the original key. Note that surrogate key values are artificial and have no meaning to the users. In fact, surrogate key values are normally hidden in forms and reports.

1、Relational Model Terminology

- **Keys**

Foreign Keys

A foreign key is a column or composite of columns that is the primary key of a table other than the one in which it appears. The term arises because it is a key of a table foreign to the one in which it appears. In the following two tables, DEPARTMENT.DepartmentName is the primary key of DEPARTMENT, and EMPLOYEE.DepartmentName is a foreign key. In this text, we will show foreign keys in italics:

```
DEPARTMENT (DepartmentName, BudgetCode, ManagerName)
EMPLOYEE (EmployeeNumber, EmployeeLastName, EmployeeFirstName,
DepartmentName)
```

Foreign keys express relationships between rows of tables. In this example, the foreign key EMPLOYEE.DepartmentName stores the relationship between an employee and his or her department.

Consider the SKU_DATA and ORDER_ITEM tables. SKU_DATA.SKU is the primary key of SKU_DATA, and ORDER_ITEM.SKU is a foreign key.

```
SKU_DATA (SKU, SKU_Description, Department, Buyer)
ORDER_ITEM (OrderNumber, SKU, Quantity, Price, ExtendedPrice)
```

1、Relational Model Terminology

➢Keys

Notice that ORDER_ITEM.SKU is both a foreign key and also part of the primary key of ORDER_ITEM. This condition sometimes occurs, but it is not required. In the example above, EMPLOYEE.DepartmentName is a foreign key, but it is not part of the EMPLOYEE primary key. You will see some uses for foreign keys later in this chapter and the next, and you will study them at length in Chapter 6.

In most cases, we need to ensure that the values of a foreign key match a valid value of a primary key. For the SKU_DATA and ORDER_ITEM tables, we need to ensure that all of the values of ORDER_ITEM.SKU match a value of SKU_DATA.SKU. To accomplish this, we create a referential integrity constraint, which is a statement that limits the values of the foreign key. In this case, we create the constraint:

```
SKU in ORDER_ITEM must exist in SKU in SKU_DATA
```

This constraint stipulates that every value of SKU in ORDER_ITEM must match a value of SKU in SKU_DATA.

LOGO

Thank You!