PART IV

THE RELATIONAL DATA MODEL

CHAPTER 8

RELATIONAL DATA MODEL FUNDAMENTALS

CHAPTER OBJECTIVES

- Understand how the relational model is a superior conventional data model
- Study the components of the relational data model in detail
- Review the significance of data integrity and how this model provides proper constraints
- Learn about data manipulation in the relational model and examine generic languages
- Survey two common methods for relational model design

In Chapter 1, you had a very brief introduction to the relational data model. Before the introduction of this data model, databases were developed based on the hierarchical model first and then based on the network model. We also mentioned that although these two models overcame most of the limitations of earlier file-oriented data systems, a few drawbacks still remained. However, a landmark paper written by Dr. E. F. Codd in 1970 completely changed the database scene. People began to view data and databases in a different way. The era of relational database systems started, and organization after organization began to adopt the new, superior model.

In the previous chapters, you have been studying semantic data modeling. You went through the details of two methods for creating a semantic data model—object-based data modeling and entity-relationship data modeling techniques. A semantic data model represents the true meaning of the information requirements of an organization. As explained in Chapter 5, a conventional model portrays how data in the database are to be perceived. As mentioned before, the hierarchical,

Database Design and Development: An Essential Guide for IT Professionals by Paulraj Ponniah
ISBN 0-471-21877-4

network, and relational data models are conventional data models. These dictate the view and arrangement of data in the database. A conventional data model is based on a set of standards or conventions developed in a certain way and accepted. The hierarchical model views data in a database as arranged in a hierarchical, top-down fashion; the network model views as arranged in a network configuration.

On one side, the semantic data model—a generic data model—represents the information requirements of an organization. On the other side, the conventional data model represents how data stored in the database are perceived. Each of the three conventional data models has its own way of perceiving the arrangement of data. Now, if your organization wants to implement a hierarchical database, you take your semantic data model and transform it into a conventional, hierarchical model and implement the hierarchical database. You take similar routes from the semantic data model to the implementation of your database if your organization wants to have a network database or a relational database. The big advantage of creating a semantic data model first is this—being generic, the semantic data model can be transformed into any type of conventional data model.

STRUCTURE AND COMPONENTS

The relational model uses familiar concepts to represent data. In this model, data are perceived as organized in traditional, two-dimensional tables with columns and rows. The rigor of mathematics is 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 with two generic languages for manipulating data organized as a relational model.

A relation or two-dimensional table forms the basic structure in the relational model. What are the implications? In requirements gathering, you collect so much information about business objects or entities, their attributes, and relationships among them. 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 conventional data model.

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 its features in a better light. Here is a summary of the strengths:

Mathematical relation. Uses the concept of a mathematical relation or two-dimensional table to organize data. Rests on solid mathematical foundation.

Mathematical set theory. Applies the mathematical set theory for manipulation of data in the database. Data manipulation uses a proven approach.

Disciplined data organization. Resting on solid mathematical foundation, data organization and manipulation are carried out in a disciplined manner.

Simple and familiar view. Provides a common and simple view of data in the form of two-dimensional tables. Users can easily perceive how data are organized; they need not be concerned with how data are actually stored in the database.

Logical links for relationships. As you will recall, hierarchical and network data models 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, relational model uses logical links to establish relationships. This is a major advantage.

Relation: The Single Modeling Concept

As mentioned above, a relation or table is the primary data modeling concept in the relational mode. A table is 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 COURSE in the relational data model for the university. Figure 8-1 shows a plain two-dimensional table whose format represents an entity or object.

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

- The relation is a table, representing data about some entity or object.
- The relation is not just any random table, but one that conforms to certain relational rules.
- The 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.

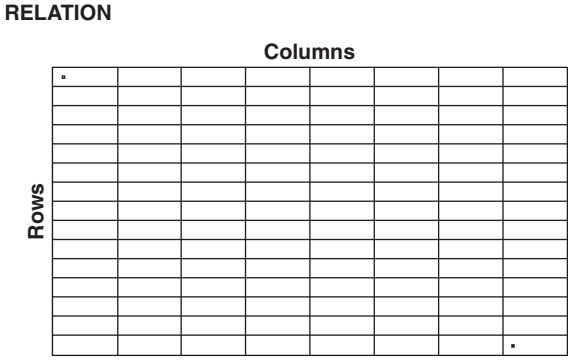


Figure 8-1 Relation or two-dimensional table.

EMPLOYEE Relation

Columns

SocSecNumber	EmployeeName	Phone	Position	DeptCode
.				
				.

Rows

Figure 8-2 Employee relation: attributes.

- Table names and column names are used to understand the data. The table or relation name indicates the entity type; the column names indicate its characteristics.

Columns as Attributes

Figure 8-2 presents a relation representing the 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.

Rows as Instances

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

EMPLOYEE Relation

Columns

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

Tuples OR Rows

Figure 8-3 Employee relation: tuples.

Figure 8-3 shows the rows or tuples for the EMPLOYEE relation.

If there are 5000 employees in the organization, the relation will contain 5000 rows. The number of tuples in a relation is known as the cardinality of the relation. For an EMPLOYEE relation with 5000 rows, the cardinality is 5000.

Now, because a relation is considered as a mathematical set, this EMPLOYEE relation is a set of 5000 data elements. Manipulation of data in the EMPLOYEE relation, therefore, becomes a set operation. Later on, you will study set operations and learn how to apply these in working with data in a relational model.

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 are 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 5000 rows, each row represents a particular employee. But, how can we know which row represents an employee we are looking for? To identify a row uniquely, we can use the attribute values. We may say that if the value of the attribute EmployeeName is "Carey Smith" that row represents this particular employee. What if there is another Carey Smith in the organization? 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 identifiers in a relation:

Superkey. A set of attributes that uniquely identifies each tuple in a relation.

Key. A minimal set of attributes that uniquely identifies each tuple in a relation.

Composite key. A key consisting of more than one attribute.

Candidate key. A set of attributes that can be chosen to serve the key.

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

LOGICAL LINKS

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	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 8-4 Department and employee relations: relationship.

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.

Relationships Through Foreign Keys

You have noted above 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 8-4 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 the EMPLOYEE relation to corresponding tuples in the DEPARTMENT relation. The DeptCode attribute in the EMPLOYEE relation is called a foreign key attribute. Note especially 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 and a tuple in a second relation is established by placing the primary key value in the tuple of the first

relation as the 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 the EMPLOYEE relation with a null value in the foreign key column. This shows that the specific tuple is not linked to any tuple in the 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. A 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 the foreign key column indicate an optional relationship between the two relations.

Mandatory Relationship In the 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 the EMPLOYEE relation, therefore, points to a related tuple in the 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 8-5 gives an example of relational tables. Figure 8-6 presents a standard notation used to represent this relational data model.

Note the following description of the notation:

- The 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 parenthesis. These are the attributes for the relation.
- Primary key attributes are indicated with underscores. Examples—BuildingID, SkillCode.
- Statements immediately following the notation of a relation indicate the foreign keys within that relation. Example—Foreign keys: SkillCode references SKILL.

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 8-5 Relational tables.

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 8-6 Relational data model: notation.

- 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.

DATA INTEGRITY CONSTRAINTS

It is essential that a database built on any specific data model must ensure the validity of the 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, 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 above, 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 us establish the reasons for ensuring data integrity. A database is said to possess data integrity if the data values 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.

- 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

Here 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 the same domain.

Order of columns. The sequence of the columns is insignificant. The sequence may be changed 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, middle, or 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. The 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, the entity integrity rule is a rule about the primary key that is meant to identify rows uniquely. The rule applies to single relations.

> ***Entity identity rule:*** No part of the primary key of any tuple in a relation can have a null value.

Figure 8-7 presents three relations, EMPLOYEE, PROJECT, and ASSIGNMENT. The 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 the entity identify rule affects the integrity of the data model.

Note the null values present in a few rows; because of these rows the entity integrity rule is violated in the relation. If two or more rows have null values as primary key values, how can you distinguish between these rows? Which row

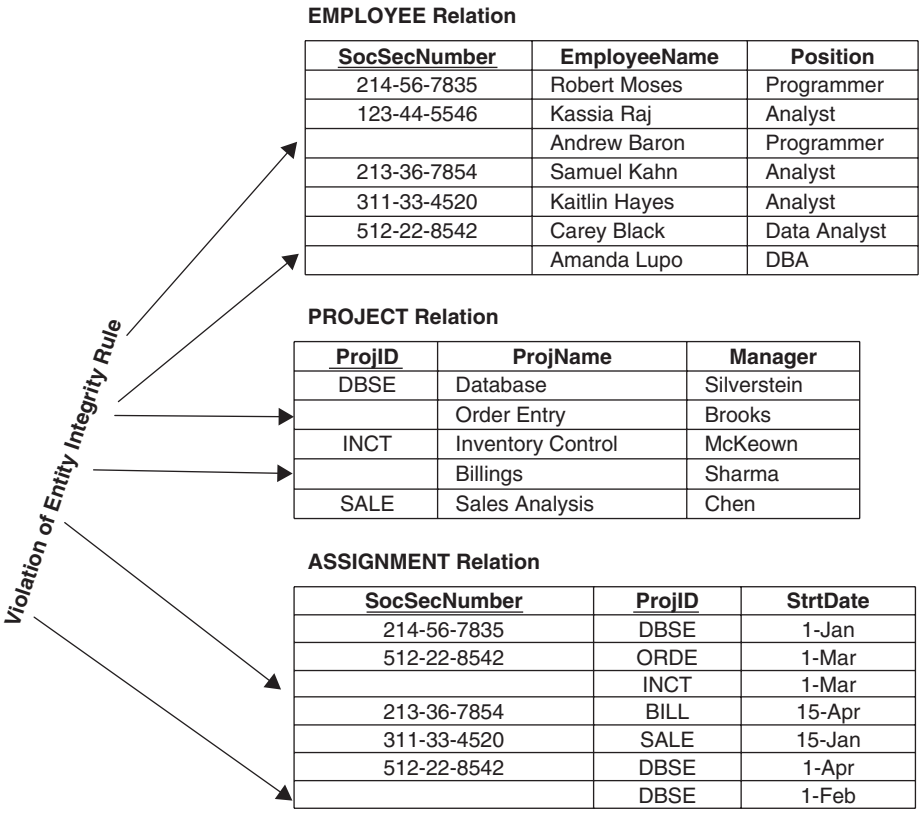


Figure 8-7 Entity integrity rule.

denotes which specific entity? In the case of the relation ASSIGNMENT, even if only part of the primary key contains null values 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 8-8 shows two relations, DEPARTMENT and EMPLOYEE.

Note how the values in the foreign key column DeptNo in the EMPLOYEE relation and in the primary key column DeptNo in the DEPARTMENT relation link related rows in the two relations. In the figure, employee Charles is assigned to department Accounting and employee Eldon is assigned 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 null values in foreign key columns indicate? You know that null values in foreign key columns

EMPLOYEE Relation

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

DEPARTMENT Relation

DeptNo	DeptName	Location
D100	Engineering	West
D200	Accounting	South
D300	Marketing	East

Figure 8-8 Referential integrity rule.

denote optional relationships. That means that employee Paul is not assigned to any department.

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. The referential integrity rule addresses the establishment of clear references between related tables. The 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.

Functional Dependencies

Let us use the EMPLOYEE, PROJECT, and ASSIGNMENT relations shown in Figure 8-7 to examine the concept of functional dependency. 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 8-7, 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.

Which attribute's values 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 with 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 the 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 for 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.

DATA MANIPULATION

What is the motivation for creating a database system for an organization? Why do you examine all the information requirements, create a data model to reflect the information requirements, and build a database system based on the designed data model? Of course, the reason is to provide users with the ability to access the data, share it, and use it to perform the various business processes. The database is meant for usage—read data, change data, add data, and also delete data from the database as needed.

The structure of the relational model in the form of two-dimensional tables rests on a solid mathematical foundation. Can mathematical principles and methods based on the set theory be used to retrieve data from a database built on the relational model? How can you manipulate data? Are there sound languages to interface with the database and work with the data in the database?

Dr. Codd, who formulated the structure and rules of the relational model, also provided methods for data manipulation in a relational database. He suggested languages for doing so. We will now examine these languages and appreciate the robust mathematical foundation on which the languages stand.

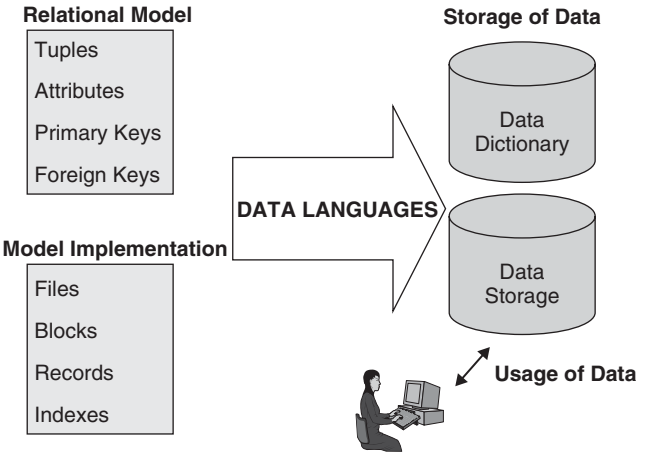


Figure 8-9 Role of data languages.

Role of Data Languages

Figure 8-9 illustrates the role of data languages. You know that data in a relational database are perceived in the form of tables. From the users' or programmers' points of view, the database contains data in the form of two-dimensional tables. Data languages for the relational model, therefore, must relate to data in the form of tables.

To use the data from a relational database, first the structures of the tables must be defined in the data dictionary. Only then will you be able to manipulate the data based on the definition of the structures. For example, if you are looking for all the customers in Alaska from your relational database, to retrieve information about the customers the DBMS must know the structure of the CUSTOMER relation. The structure of the CUSTOMER relation must have been defined first.

Two types of languages are necessary for the relational data model:

Data definition language (DDL). To define the relations, attributes, primary keys, foreign keys, data types, attribute lengths, and so on.

Data manipulation language (DML). To perform the data manipulations such as reading data, updating data, adding data, or deleting data.

Data Manipulation Languages

For data manipulation, Dr. Codd suggested two generic languages, both based on mathematical principles. Because data are perceived as organized in the form of tables, any data manipulation in the relational model can be reduced to operations on rows and columns of tables. The two languages provide well-defined methods to work on rows and columns of tables.

These are the two languages:

Relational Algebra. Consists of mathematical operations on rows and columns of one or more relations. Each operation in a series of operations produces a result-

ing relation. A data manipulation solution comprises a set of these operations performed on relations in a definite sequence.

Relational Calculus. Applies data manipulation conditions to rows and columns of relations. A data manipulation solution does not contain a series of operations but has a single solution statement to produce the necessary result.

Both relational algebra and relational calculus are generic languages. Popular languages included in commercial database management systems implement elements of these generic languages. In these generic languages, Dr. Codd has stipulated the semantics and syntax and has shown how these can be applied to perform data manipulation. Commercial vendors build their language implementations based on the two generic languages. Structured Query Language (SQL) is one such implementation based on the generic languages, mostly based on relational calculus. We will look at SQL in Chapter 13.

Let us now examine the provisions in these two languages. Remember that these are generic languages. That means that no programs are written using the syntax and semantics of the native operations in these languages. No program code is directly written to contain relational algebra and relational calculus commands or operations. However, the commercial data manipulation languages are built based on the operations and commands of the two generic languages. Therefore, to understand how data manipulation is made possible in a relational data model, you need to gain a good understanding of the two generic languages. We will now explore the major operations and constructs of the two languages. Note the underlying principles. Study the various operations and understand how they produce the required results.

Relational Algebra

In relational algebra, you have operations that work on relations. Some operations work on a single relation and others on two relations at a time. A solution for data retrieval consists of a set of operations performed in a certain order. When all the operations are performed in sequence, the final result is obtained. For example, the final result may be the retrieval of all orders for a specific customer.

Note, however, that the result is always a relation. In the case of the retrieval of orders for a specific customer, the result is the set of orders satisfying the given condition. The result is therefore a relation consisting of columns representing the attributes of the ORDER relation and only those rows satisfying the condition of belonging to the specific customer. In any case, the result consists of a relation with rows and columns.

Sometimes, the resulting relation may contain just one column and many rows, may contain many columns and just one row, or may contain just one column and just one row. In every case, the result is still a relation.

Relational algebra consists of the following nine major operations:

SELECT
PROJECT

PATIENT Relation

PatientNo	PatientName	DateAdmit	DateDischarged
1134	Bruce Greenberg	2-Jan	11-Jan
1389	Sheryl Rodericks	10-Jan	26-Jan
2107	Miranda Poko	22-Jan	1-Feb
2298	Pedro Pomales	2-Feb	18-Feb
2576	Bob Totten	11-Feb	25-Feb
3158	Peter Tora	14-Feb	14-Mar
3255	Gary Goran	6-Mar	10-Mar
3380	Pia Dorsey-Adams	15-Mar	22-Apr
4157	Anthony Brett	22-Apr	24-Apr
6199	Milton Kalma	25-Apr	27-Apr

Result of SELECT operation

PatientNo	PatientName	DateAdmit	DateDischarged
1389	Sheryl Rodericks	10-Jan	26-Jan
2298	Pedro Pomales	2-Feb	18-Feb
3158	Peter Tora	14-Feb	14-Mar
3380	Pia Dorsey-Adams	15-Mar	22-Apr

Result of PROJECT operation

PatientName	DateDischarged
Bruce Greenberg	11-Jan
Sheryl Rodericks	26-Jan
Miranda Poko	1-Feb
Pedro Pomales	18-Feb
Bob Totten	25-Feb
Peter Tora	14-Mar
Gary Goran	10-Mar
Pia Dorsey-Adams	22-Apr
Anthony Brett	24-Apr
Milton Kalma	27-Apr

Figure 8-10 Relations for SELECT and PROJECT operations.

ASSIGNMENT
UNION
INTERSECTION
DIFFERENCE
PRODUCT
JOIN
DIVIDE

Let us examine each of these operations with the aid of examples. To study the following three operations, we will use the relations or tables shown in Figure 8-10.

Select

Features and function

- Operates on a single relation.
- Selects specific rows from a relation on the basis of given criteria.
- The result is a relation containing rows and columns as indicated below:
  - Rows—only the selected rows from the original relation
  - Columns—same columns as in the original relation

Example

Data request From PATIENT relation, find all the patients who have stayed in the medical center for more than 15 days.

Relational algebra solution

SELECT (PATIENT : (DateDischarged – DateAdmit) > 15)

Note the syntax for the operation in relational algebra. The keyword SELECT is used to indicate the operation.

Project

Features and function

- Operates on a single relation.
- Selects specific columns from a relation on the basis of given criteria.
- The result is a relation containing rows and columns as indicated below:
  - Rows—same rows as in the original relation
  - Columns—only the selected columns from the original relation

Example

Data request From PATIENT relation, list only the names of patients and their discharge dates.

Relational algebra solution PATIENT [PatientName, DateDischarged]

Note the syntax for the operation in relational algebra. There is no need for a keyword. The selected columns are indicated within the square brackets. In this case, the result is a relation containing the same number of rows as the number of patients in the database and two columns.

Assignment

Features and function

- Simple operation; not really an operation in the sense of other operations.
- Assigns the result of some set of operations to the resulting relation.

Example

Data request From PATIENT relation, find all the patients who have stayed in the medical center for more than 15 days. Assign the result of the operation to a relation called PATIENT-15.

Relational algebra solution

PATIENT-15:= SELECT (PATIENT : (DateDischarged – DateAdmit) > 15)

Note the symbol := used to perform the assignment of the result and name the resulting relation as PATIENT-15. There are no keywords.

Union

Features and function

- Operates on two relations.
- Combines two relations.
- Both relations must be union-compatible; that is, they must have the same columns.
- The result is a single relation containing rows and columns as indicated below:
  - Rows—all the rows from both relations, duly eliminating any duplicate rows
  - Columns—same columns as in either of the original relations

Example

In the database of a medical center, there are two relations representing data about the doctors. One relation, SUPVRDOCTOR, represents data about doctors with supervisory responsibilities who have other doctors reporting to them. The other relation SUBDOCTOR represents data about every doctor who reports to another doctor. Figure 8-11 shows these two tables with data in the rows and columns.

Data request List all the attributes of all the doctors in the medical center. Call the result relation ALLDOCTOR.

SUBDOCTOR Relation

DrID	DrName	Specialty	DrPhone	SuprDrID
10	Kim Maggio	Family Practice	918-756-0831	25
14	Monika Isaac	Neurology	609-238-3377	45
23	Marcus Kulper	Internal Medicine	732-613-9313	36
37	Ira Goldman	Psychiatry	609-246-0331	13
39	Lawrence Stark	Radiology	918-613-1988	45
45	Robert Martin	Surgery General	609-924-9248	25
36	Elaine Hsu	Urology	908-535-5228	25
13	Leroy Richardson	Oncology	908-322-5566	25

SUPVR DOCTOR Relation

DrID	DrName	Specialty	DrPhone	SuprDrID
25	Roger Smarts	Neurology	609-238-5100	
45	Robert Martin	Surgery General	609-924-9248	25
36	Elaine Hsu	Urology	908-535-5228	25
13	Leroy Richardson	Oncology	908-322-5566	25

Figure 8-11 SUBDOCTOR and SUPVRDOCTOR relations.

ALLDOCTOR Relation

DrID	DrName	Specialty	DrPhone	SuprDrID
10	Kim Maggio	Family Practice	918-756-0831	25
14	Monika Isaac	Neurology	609-238-3377	45
23	Marcus Kulper	Internal Medicine	732-613-9313	36
37	Ira Goldman	Psychiatry	609-246-0331	13
39	Lawrence Stark	Radiology	918-613-1988	45
45	Robert Martin	Surgery General	609-924-9248	25
36	Elaine Hsu	Urology	908-535-5228	25
13	Leroy Richardson	Oncology	908-322-5566	25
25	Roger Smarts	Neurology	609-238-5100	

Explanation of UNION operation

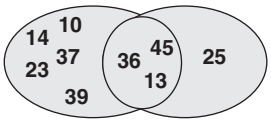


Figure 8-12 UNION operation.

Relational algebra solution
ALLDOCTOR:= SUPVRDOCTOR ∪ SUBDOCTOR

Note the syntax for the operation in relational algebra and the symbol U indicating the union operation. The union operation combines the data from the two relations, eliminating duplication. Note that some doctors, who report to others and also have doctors reporting to themselves, will be in both relations. The union operation in relational algebra is similar to the union function in mathematical set theory. Note the solution shown in Figure 8-12.

Intersection

Features and function

- Operates on two relations.
- Identifies common rows in two relations.
- Both relations must be union-compatible; that is, they must have the same columns.
- The result is a single relation containing rows and columns as indicated below:
  - Rows—only those rows that are common to both relations
  - Columns—same columns as in either of the original relations

Example

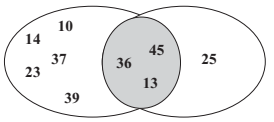
Refer to Figure 8-11 for the discussion of this operation.

Data request List all the attributes of only those doctors who have doctors reporting to them and who themselves report to other doctors. These are the doctors who are both subordinates as well as having supervisory responsibilities. Call the result relation DOCTOR.

DrID	DrName	Specialty	DrPhone	SuprDrID
45	Robert Martin	Surgery General	609-924-9248	25
36	Elaine Hsu	Urology	908-535-5228	25
13	Leroy Richardson	Oncology	908-322-5566	25

DOCTOR Relation

Explanation of INTERSECTION operation



DrID	DrName	Specialty	DrPhone	SuprDrID
10	Kim Maggio	Family Practice	918-756-0831	25
14	Monika Isaac	Neurology	609-238-3377	45
23	Marcus Kulper	Internal Medicine	732-613-9313	36
37	Ira Goldman	Psychiatry	609-246-0331	13
39	Lawrence Stark	Radiology	918-613-1988	45

SUBORDDOCTOR Relation

Explanation of DIFFERENCE operation

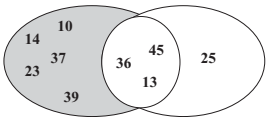


Figure 8-13 INTERSECTION and DIFFERENCE operations.

Relational algebra solution
DOCTOR := SUPVRDOCTOR ∩ SUBDOCTOR

Note the syntax for the operation in relational algebra and the symbol ∩ indicating the intersection operation. The intersection operation finds only those entities that are common to both relations. Intersection operation in relational algebra is similar to intersection function in mathematical set theory. Note the solution shown in Figure 8-13.

Difference

Features and function

- Operates on two relations.
- Identifies the rows in one relation that are not present in the other relation.
- Both relations must be union-compatible; that is, they must have the same columns.
- The result is a single relation containing rows and columns as indicated below:
  - Rows—only those rows that are in the first relation but not in the second
  - Columns—same columns as in either of the original relations

Example

Refer to Figure 8-11 for the discussion of this operation.

Data request List all the attributes of all doctors who report to some other doctors but have no one reporting to them. These are the doctors who are sub-

ordinates without any supervisory responsibilities. Call the result relation SUBORDDOCTOR.

Relational algebra solution
SUBORDDOCTOR : = SUBDOCTOR — SUPVRDOCTOR

Note the syntax for the operation in relational algebra and the symbol — indicating the difference operation. The difference operation finds only those entities in the SUBDOCTOR relation but not present in the SUPVRDOCTOR relation. These are subordinates that do not have any supervisory responsibilities at all. Note the solution shown in Figure 8-13.

Also note that the result will not be same if you reverse the order of the two relations in the difference operation.

(A—B) is not the same as (B—A). (SUPVRDOCTOR — SUBDOCTOR) represents the doctors who only have supervisory responsibilities and do not report to any other doctor at all.

Product

Features and function

- Operates on two relations.
- Produces the scalar product between the two relations.
- By itself, the product operation has no meaning in business functions. But the product operation is an essential interim operation for the more important join operation.
- The result is a single relation containing rows and columns as indicated below:
  - Rows—formed by concatenation each of row of one relation to each row of the other relation
  - Columns—all the columns of both relations put together

Example

Refer to Figure 8-14 for the discussion of this operation. Note the two relations A and B.

Data request Find the product relation C between two relations A and B.
Relational algebra solution C := A ∗ B

Note the syntax for the operation in relational algebra and the symbol ∗ indicating the product. Observe how the rows for relation C are formed from the rows of A and B.

Let us generalize the product operation.

If relation A has m columns and p rows, and relation B has n columns and q rows, then the result relation C will have $(m+n)$ columns and $(p*q)$ rows.

RELATION A

X	Y
12	25
13	28

RELATION B

S	T
34	56
38	99
45	84

RELATION C

X	Y	S	T
12	25	34	56
12	25	38	99
12	25	45	84
13	28	34	56
13	28	38	99
13	28	45	84

Figure 8-14 PRODUCT operation.

Join

Features and function

- Operates on two relations that have common columns.
- Connects data across two relations based on values in common columns.
- Types of join operations: natural join, theta join, outer join.
- The result is a single relation containing rows and columns as indicated below for a natural join of relations A and B with a common column with name c:
  - Take the product of A and B
  - The resulting relation will have two columns for c
  - Eliminate all rows from the product except those rows where the value for column c in one relation is the same the value for column c in the other relation
  - Apply project operation to eliminate one of the two common c columns from the result relation

Example

Refer to Figure 8-15 for the discussion of this operation. Note the two relations PATIENT and BILLING.

Data request List the billing data for all discharged patients. Call the result relation DISCHRGPATIENT.

Relational algebra solution

DISCHRGPATIENT := JOIN (PATIENT, BILLING)

Note the syntax for the operation in relational algebra and the keyword JOIN indicating the join operation. Note the steps and also the rows and columns in the final result. The data about patient discharges is in the PATIENT relation. The BILLING relation contains data about patient billing. We need to

PATIENT Relation

PatientNo	DischargeDate
1234	10-Feb
2345	22-Feb
3456	5-Mar

BILLING Relation

BldPatientNo	BilledCharges
1234	150.00
2345	225.00
1234	315.00
2345	195.00

DISCHRGPATIENT Relation --- three types

Result of equijoin

PatientNo	DischargeDate	BldPatientNo	BilledCharges
1234	10-Feb	1234	150.00
2345	22-Feb	2345	225.00
1234	10-Feb	1234	315.00
2345	22-Feb	2345	195.00

Result of natural join

PatientNo	DischargeDate	BilledCharges
1234	10-Feb	150.00
2345	22-Feb	225.00
1234	10-Feb	315.00
2345	22-Feb	195.00

Result of outer join

PatientNo	DischargeDate	BilledCharges
1234	10-Feb	150.00
2345	22-Feb	225.00
1234	10-Feb	315.00
2345	22-Feb	195.00
3456	5-Mar	

Figure 8-15 JOIN operation.

connect these pieces of data across the two relations based on equal values in PatientNo columns that are common to both relations. Natural join operation accomplishes this task.

Also note the other two types of join operations illustrated in Figure 8-15.

Divide

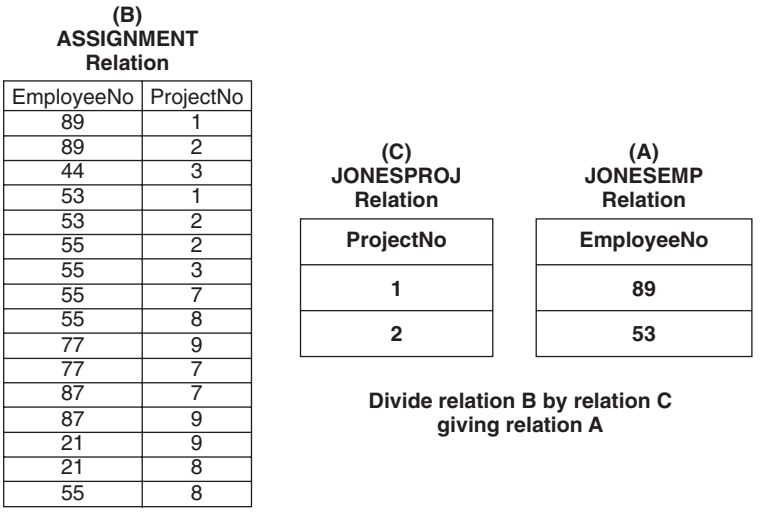
Features and function

- Operates on two relations.
- The divide operation is the reverse of the product operation.
- The result is a single relation containing rows and columns as indicated below:
  - Rows—rows in one relation matching with every row in another relation
  - Columns—all and only columns of the first relation, but not of the second

Example

Refer to Figure 8-16 for the discussion of this operation. Note the three relations A, B, and C.

Divide relation B by relation C, giving A as the result relation. The following conditions apply:



(B) ASSIGNMENT Relation

EmployeeNo	ProjectNo
89	1
89	2
44	3
53	1
53	2
55	2
55	3
55	7
55	8
77	9
77	7
87	7
87	9
21	9
21	8
55	8

(C) JONESPROJ Relation

ProjectNo
1
2

(A) JONESEMP Relation

EmployeeNo
89
53

Divide relation B by relation C giving relation A

Figure 8-16 DIVIDE operation.

Columns of C must be a subset of columns of B.
Columns of A are all and only those columns of B that are not columns of C.
Place a row in A, if and only if it is associated with every row in C.

Data request Find all employees who worked on every project Jones had worked on. Name the result as relation JONESEMP (see Figure 8-16).

Relational algebra solution

JONESEMP := ASSIGNMENT / JONESPROJ

Note the syntax for the operation in relational algebra and the symbol / indicating divide operation. Observe how the rows for the result relation JONESEMP are formed. Note that columns of JONESPROJ are a subset of columns of relation ASSIGNMENT. Columns of the result relation JONESEMP are all and only those columns of ASSIGNMENT that are not columns of JONESPROJ.

Relational Calculus

This is the second generic language proposed for manipulating data in a database conforming to the relational data model. As you have seen, relational algebra consists of distinct operations. To arrive at a required solution relation, you use the operations one after another in a specific sequence. Commercial database vendors provide equivalent commands or statements for the relational algebra operations in their language implementations.

Relational calculus is markedly different. Here, no separate and distinct operations exist similar to the relational algebra operations. Relational calculus expresses a solution relation in a single statement. Database vendors implement the typical relational calculus statement in their database languages.

SALESPERSON Relation

SalespersID	SalesperName	SupvrID	OfficeCity
15	Peterson	42	Boston
21	Espinoza	38	Trenton
23	James	42	Boston
27	Warren	38	Toronto
35	Rolland	38	Montreal
38	Demetrius		Quebec
42	McPherson		Boston

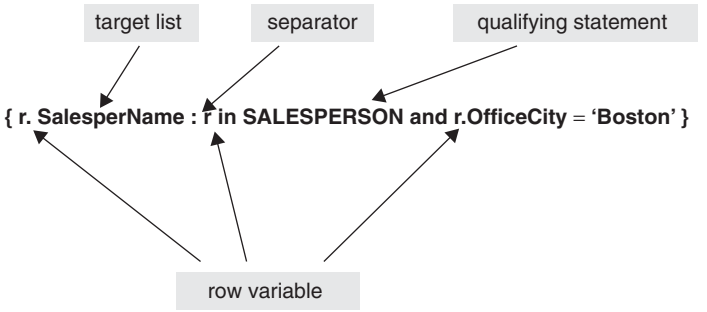


Figure 8-17 Relational calculus statement.

Let us take an example. Refer to Figure 8-17.

Data request What are the names of the salespersons attached to the Boston office?

We mentioned that relational calculus provides single solution statements. First, let us try to write the solution statement in plain English.

Solution to contain

SalesperName
from rows in SALESPERSON relation,
only from certain rows from SALESPERSON relation,
a row in SALESPERSON to qualify to be part of the solution
it must satisfy the condition that
OfficeCity from that row must be Boston.

A relational calculus solution statement is similar to this plain English statement. Instead of expressing the solution in a long-winding way as the above statement, relational calculus statement is more succinct and standardized. See the solution example shown below and also as explained in Figure 8-17.

Relational calculus solution

{ r.SalesperName : r IN SALESPERSON AND r.OfficeCity = 'Boston' }

In the figure, the components of the relational calculus statement are pointed out. Note the descriptions of the components as shown below:

CUSTOMER Relation

CustomerNo	CustomerName	Country
1000	Rosato Electric	U.S.A.
1010	MDZ Electric	U.K.
1050	Electric des Royal	France
1100	Mitshista Electric	Japan
1120	Jean Pierre	France

SUPPLIER Relation

SupplierNo	SupplierName	SupCountry
200	Herman Electricals	U.S.A.
205	Allied Electrics, Ltd.	U.K.
300	Akiro & Amana	Japan

PRODUCT Relation

ProductCode	ProdDesc	SupplierNo
4136	Three-way switch	205
5555	Circuit breaker	200
5678	Junction box	300
6543	5HP Motor	300

SALE Relation

Date	CustomerNo	ProductCode	SalespersID
5-Mar	1000	4136	15
7-Mar	1010	5555	21
9-Mar	1050	5678	23
13-Mar	1100	6543	27
7-Mar	1120	5555	35
15-Mar	1000	4163	38
29-Mar	1100	6543	23

SALESPERSON Relation

SalespersID	SalesperName	SupvrID	OfficeCity
15	Peterson	42	Boston
21	Espinoza	38	Trenton
23	James	42	Boston
27	Warren	38	Toronto
35	Rolland	38	Montreal
38	Demetrius		Quebec
42	McPherson		Boston

Figure 8-18 Sample relations to illustrate solutions.

Target list. List of attributes to be included in the solution relation.

Row variable. A letter to indicate a variable quantity that represents the qualifying rows.

Qualifying statement. The part of the relational calculus solution that contains conditions for qualifying rows.

Separator. The indicator ":" separating the target list of attributes to be present in the solution relation from the rest of the relational calculus statement.

Comparison of Generic Languages

Let us conclude our discussion of the two powerful generic languages proposed by Dr. Codd with a comparison between them.

Comparison of Solutions Let us go over a few examples of relational algebra and relational calculus solutions. Use the relations presented in Figure 8-18.

Data request What are the names of the customers in France? Mark the result relation as CUSTFRANCE

Relational algebra solution

A : = SELECT (CUSTOMER: Country = 'France')

CUSTFRANCE : = A [CustomerName]

Relational calculus solution

CUSTFRANCE

{r.CustomerName : r in CUSTOMER and r.Country = 'France'}

Data request List the names and countries of customers who bought product 5555. Mark the result relation as CUSTPROD5555.

Relational algebra solution

A : = SELECT (SALE : ProductCode = '5555')

B : = JOIN (CUSTOMER, A)

CUSTPROD5555 : = B [CustomerName, Country]

Relational calculus solution

CUSTPROD5555

{(r.CustomerName, r.Country) : r IN CUSTOMER and there exists s in SALE (s.CustomerNo = r.CustomerNo and s.ProductCode = '5555')}

Notice that you need two row variables r and s in the relational calculus solution. Note also the existence condition. A solution is available only if qualifying rows exist in relation SALE. The existence condition in a relational calculus solution is comparable to the JOIN operation in a relational algebra solution.

Procedural and Nonprocedural What is the essential difference you have noticed between the two languages from the above solution examples? The relational algebra solution consists of a few steps of distinct operations, whereas the relational calculus solution is just a single statement. Using relational algebra, you need to determine the operations and their sequence. With relational calculus, you just state the result you require. This significant difference expresses the essential nature of the two generic languages.

Relational Algebra: Procedural Language A procedural language is a language in which the programmer must code the steps in the proper sequence. The language allows execution of the distinct steps in the specified sequence. You have to write the procedure. You have to state how the data manipulation must be done to produce the solution.

Relational Calculus: Nonprocedural Language When using a nonprocedural language, you do not have to specify the steps or operations, nor do you have to state the order in which the operations must be executed. You do not have to code a procedure. You do not have to state how the solution must be arrived at. You just have to state the solution you are looking for. You simply express the solution through a single solution statement.

Same Solution The generic languages are two different methods for language implementations. Let us say you arrive at a certain solution relation for a particular data manipulation request by using an implementation of relational algebra. According to Dr. Codd, the two languages are relationally equivalent. That means that you can also arrive at the same solution relation by using an implementation of relational calculus.

Every data manipulation request that can be completed through the use of one language can be done through the use of the other language. Let us say that you write a relational algebra solution for a complex query involving several source relations. The solution is likely to contain several relational algelra operations such SELECT, PROJECT, JOIN, DIVIDE, and so on. Assume that you are able to arrive at the solution through a commercial language implementing relational algebra. Relational equivalency means that you can arrive at the solution for the same complex query in relational calculus and a proper commercial language implementation of relational calculus will find the exactly same solution.

Relationally Complete According to the author of the relational model, the true test for whether a commercial language implementation is truly relational is straightforward. If a commercial language can produce the same results that are produced by either of the two generic languages for any query, then the commercial language is said to be relationally complete.

The property of being relationally complete is an important qualification for a language that can be used in a relational database. In other words, the commercial language must be able to perform any set of relational algebra operations, however complex the set may be. Similarly, the commercial language must be able to implement and produce the result expressed by any relational calculus statement, however complex the statement may be.

Commercial Language Implementations Structured Query Language (SQL) has gained acceptance as the standard for the relational model. The language incorporates functions of the generic languages, although the standard SQL statement is seen to be more like a relational calculus implementation. However, specific relational algebra operations are also included in the repertoire of SQL.

Commercial database vendors include SQL as part of their database products. SQL interface is a standard component of the relational database management systems on the market. However, each vendor tries to augment SQL standards with its own proprietary enhancements. This results in a kind of incompatibility. SQL code written for one commercial DBMS may not work for another DBMS unless you refrain from using any of the proprietary revisions to SQL standards.

RELATIONAL MODEL 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, to develop and build a relational database system for your organization, you need to learn how to design and put together a relational data model. Although the model appears to be simple, how do you create a relational data model from the requirements? The previous chapters covered the details

of creating data models. We went through the methods and steps for creating the object-based data model and the entity-relationship data model. Now the task is to create a relational data model, which is not the same as one of those two data models. Why do you need to create a relational data model? If you are developing a relational database system, then you require that your information requirements be represented in a relational data model first. Let us explore the methods for creating a relational data model.

Requirements for Data Model

From the previous chapters you know how to create a semantic data model from the information requirements. A semantic data model captures all the meanings and content of the information requirements of an organization. We also discussed the merit of the semantic data model. Being a generic data model, it is not restricted by the structure and format rules of the conventional data models such as hierarchical, network, or relational data models. You are now convinced that representing information requirements in the form of a semantic data model is the proper way.

Well, what are the steps between creating a semantic data model and the implementation of a relational database system for your organization? You know that the semantic 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.

Design Approaches

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 8-19 presents the two design approaches for creating a relational data model.

Note how in one approach, you go through the steps of creating a semantic data model first and then transform the semantic model into a relational data model. The other approach appears to be a short-cut method bypassing the semantic data model. In this approach, you proceed to the task of creating the relational data model straight from the requirements definitions. Let us examine the basics of the two approaches.

Semantic to Relational Model

The first method shown in Figure 8-19 takes you through the semantic data model. In this approach, you complete the semantic data model. To create the semantic data model, you may use either the object-based data modeling technique or the

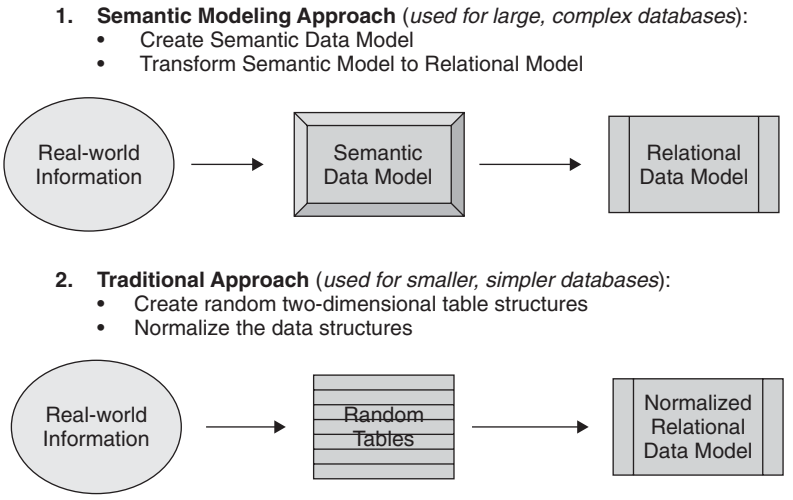


Figure 8-19 Relational data model: design approaches.

entity-relationship data modeling technique. Either of these techniques will produce a semantic data model that can be used to create the relational data model.

Here are the steps in this design approach:

- Gather the information requirements of the organization.
- Create semantic data model to truly represent the information requirements.
- Review the overall semantic data model for completeness.
- Take each component of the semantic data model at a time and examine it.
- Transform each component of the semantic 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 semantic data model.
- Complete the relational data model.
- Review the relational data model for completeness.

Chapter 9 elaborates on this approach to designing the relational data model. We will list the components of the semantic model and the corresponding components of the relational model. We will determine how each component of the semantic 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, database practitioners had traditionally 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 semantic data model? Does this not appear to be a practical design approach?

Although the approach is deceptively simple, as you will note in Chapter 10, 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 ordered methodology for making this design approach work. After an initial set of tables is designed, you must go through a step-by-step process of normalization of the initial tables. After the normalization steps are completed, 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 with 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 10 covers this approach to designing the relational data model in detail. You will recognize the need and motivation for the normalization process. We will list the normalization steps and show how to apply a normalization principle at each step. You will note how, after each step, the set of tables comes closer to being the correct set of tables and becoming part of the final relational data model.

Evaluation of the Two Methods

Naturally, when there are two ways for arriving at the same place, you must ask which path you should 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 semantic 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 proceed to normalize the tables and complete the relational data model.

Other method systematic. The method of creating the semantic data model first and then transforming it into the required relational data model is a systematic method with well-defined steps. The semantic data model is created through clearly defined data modeling techniques. You then take the components of the semantic 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 more 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 choose the normalization method.

CHAPTER SUMMARY

- The relational data model is found to be superior to other earlier conventional data models such as the hierarchical and network data models.
- The relational data model, standing on solid mathematical principles, is a disciplined method for data organization, provides a simple and familiar view of data in the form of tables, and, above all, links related data elements logically and not physically.
- A relation or a two-dimensional table with rows and columns is the single modeling concept for this data model. The columns represent the attributes of the entity; the rows represent single instances of the entities.
- Relationships between tables or relations are established through foreign keys, not physical pointers.
- To provide data integrity and consistency, the set of tables in a relational model must conform to relational constraints or rules. The entity integrity rule governs the integrity of a single relation; the referential integrity rule stipulates conditions for two related tables.
- Each data item in a row or tuple of a relation must depend on the entire primary key alone, not on any other attribute in the row.
- Relational algebra and relational calculus are two generic languages for data manipulation in a relational data model. Relational algebra is a procedural language consisting of distinct operations; relational calculus, being a nonprocedural language, just specifies the solution in the form of single statements.

Commercial language implementations are based on these two generic languages.

- Two methods for creating a relational data model: transformation method, in which you create a semantic model first and then transform it into a relational model; traditional method, in which you come up with a set of initial tables and then normalize the tables to produce a correct relational model.

REVIEW QUESTIONS

1. List and explain the major strengths of the relational data model.
2. "The relational data model is based on a single modeling concept." Do you agree? Explain.
3. What is an attribute domain? Describe with an example.
4. How is the relationship between two related tables established? Explain with an example.
5. What are data integrity constraints? What purposes do they serve in a relational data model?
6. Describe the referential entity rule with an example. What is the consequence of violation of the rule in a relational data model?
7. What are functional dependencies? Explain with examples.
8. Name any six of the relational algebra operations. Give examples for any two of these operations.
9. "Relational calculus is a nonprocedural language." Explain briefly with an example.
10. What are the two methods for creating a relational data model? Under what conditions would you prefer one method to the other?

EXERCISES

1. Indicate whether true or false:
   A. In a relational data model, columns are identified by their names.
   B. Each table or relation can contain only a specified number of rows or tuples.
   C. If the foreign key value of a row in one table is equal to the primary key value of a row in another table, the two rows are related.
   D. A null value in a foreign key attribute signifies a mandatory relationship.
   E. In a relational data model, the order of the rows is insignificant but the columns must be in order, with the primary key as the first column.
   F. The foreign key attribute in one table and the primary key attribute in the related table must have the same column name.
   G. The SELECT and PROJECT operations work on single relations.
   H. The JOIN operation can operate only on union-compatible relations.

I. The existence condition in relational calculus is similar to the JOIN operation in relational algebra.

J. If a commercial language can perform every operation in relational algebra, then the commercial language is said to be relationally complete.

2. Describe the data manipulation method of relational algebra. Compare this with the relational calculus method and explain the difference. Specify a typical data query in a banking environment. Find the result for this query using both generic languages.
3. As a data modeler for a small appliance store, describe the steps you would take to create a relational data model.
4. A large manufacturer of electronic components wants to implement a relational database. Which method would you adopt to create the relational data model? Give a list of the important tables in your data model.
5. The relational data model consists of two-dimensional tables, not random tables but tables that conform to relational rules. What are these rules and properties of a relational model? Describe the rules.