Database Systems: Design, Implementation, and Management

Tenth Edition

Chapter 3
The Relational Database Model

Objectives

In this chapter, students will learn:

- That the relational database model offers a logical view of data
- About the relational model's basic component: relations
- That relations are logical constructs composed of rows (tuples) and columns (attributes)
- That relations are implemented as tables in a relational DBMS

Objectives (cont'd.)

- About relational database operators, the data dictionary, and the system catalog
- How data redundancy is handled in the relational database model
- Why indexing is important

A Logical View of Data

- Relational model
  - View data logically rather than physically
- Table
  - Structural and data independence
  - Resembles a file conceptually
- Relational database model is easier to understand than hierarchical and network models

Tables and Their Characteristics

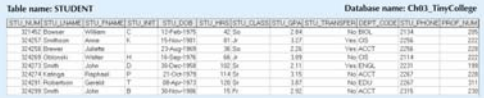
- Logical view of relational database is based on relation
  - Relation thought of as a table
- Table: two-dimensional structure composed of rows and columns
  - Persistent representation of logical relation
- Contains group of related entities (entity set)

TABLE 3.1 Characteristics of a Relational Table

1	A table is perceived as a two-dimensional structure composed of rows and columns.
2	Each table row (**tuple**) represents a single entity occurrence within the entity set.
3	Each table column represents an attribute, and each column has a distinct name.
4	Each intersection of a row and column represents a single data value.
5	All values in a column must conform to the same data format.
6	Each column has a specific range of values known as the **attribute domain**.
7	The order of the rows and columns is immaterial to the DBMS.
8	Each table must have an attribute or combination of attributes that uniquely identifies each row.

FIGURE 3.1 STUDENT table attribute values

Table name: STUDENT — Database name: Ch03_TinyCollege



STU_NUM	= Student number
STU_LNAME	= Student last name
STU_FNAME	= Student first name
STU_INIT	= Student middle initial
STU_DOB	= Student date of birth
STU_HRS	= Credit hours earned
STU_CLASS	= Student classification
STU_GPA	= Grade point average
STU_TRANSFER	= Student transferred from another institution
DEPT_CODE	= Department code
STU_PHONE	= 4-digit campus phone extension
PROF_NUM	= Number of the professor who is the student's advisor

SOURCE: Course Technology/Cengage Learning

Keys

- Each row in a table must be uniquely identifiable
- Key: one or more attributes that determine other attributes
  - Key's role is based on determination
    - If you know the value of attribute A, you can determine the value of attribute B
  - Functional dependence
    - Attribute B is functionally dependent on A if all rows in table that agree in value for A also agree in value for B

TABLE 3.2 Student Classification

HOURS COMPLETED	CLASSIFICATION
Less than 30	Fr
30–59	So
60–89	Jr
90 or more	Sr

Types of Keys

- Composite key
  - Composed of more than one attribute
- Key attribute
  - Any attribute that is part of a key
- Superkey
  - Any key that uniquely identifies each row
- Candidate key
  - A superkey without unnecessary attributes

Types of Keys (cont'd.)

- Entity integrity
  - Each row (entity instance) in the table has its own unique identity
- Nulls
  - No data entry
  - Not permitted in primary key
  - Should be avoided in other attributes

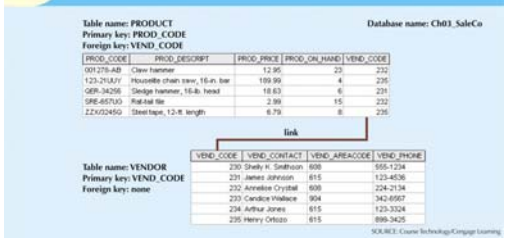
Types of Keys (cont'd.)

  - Can represent:
    - An unknown attribute value
    - A known, but missing, attribute value
    - A "not applicable" condition
  - Can create problems when functions such as COUNT, AVERAGE, and SUM are used
  - Can create logical problems when relational tables are linked

Types of Keys (cont'd.)

- Controlled redundancy
  - Makes the relational database work
  - Tables within the database share common attributes
    - Enables tables to be linked together
  - Multiple occurrences of values not redundant when required to make the relationship work
  - Redundancy exists only when there is unnecessary duplication of attribute values

FIGURE 3.2 An example of a simple relational database



Types of Keys (cont'd.)

- Foreign key (FK)
  - An attribute whose values match primary key values in the related table
- Referential integrity
  - FK contains a value that refers to an existing valid tuple (row) in another relation
- Secondary key
  - Key used strictly for data retrieval purposes

TABLE 3.3 Relational Database Keys

KEY TYPE	DEFINITION
Superkey	An attribute or combination of attributes that uniquely identifies each row in a table
Candidate key	A minimal (irreducible) superkey; a superkey that does not contain a subset of attributes that is itself a superkey
Primary key	A candidate key selected to uniquely identify all other attribute values in any given row; cannot contain null entries
Foreign key	An attribute or combination of attributes in one table whose values must either match the primary key in another table or be null
Secondary key	An attribute or combination of attributes used strictly for data retrieval purposes

Integrity Rules

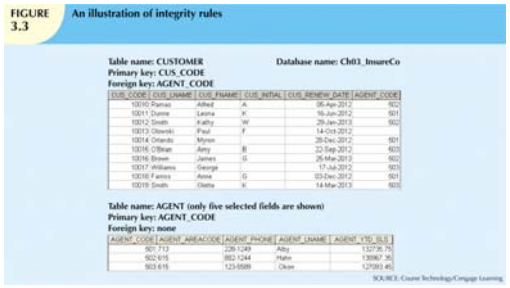
- Many RDBMs enforce integrity rules automatically
- Safer to ensure that application design conforms to entity and referential integrity rules
- Designers use flags to avoid nulls
  - Flags indicate absence of some value

TABLE 3.4 Integrity Rules

ENTITY INTEGRITY	DESCRIPTION
Requirement	All primary key entries are unique, and no part of a primary key may be null.
Purpose	Each row will have a unique identity, and foreign key values can properly reference primary key values.
Example	No invoice can have a duplicate number, nor can it be null. In short, all invoices are uniquely identified by their invoice number.
REFERENTIAL INTEGRITY	**DESCRIPTION**
Requirement	A foreign key may have either a null entry, as long as it is not a part of its table's primary key, or an entry that matches the primary key value in a table to which it is related. (Every non-null foreign key value *must* reference an *existing* primary key value.)
Purpose	It is possible for an attribute *not* to have a corresponding value, but it will be impossible to have an invalid entry. The enforcement of the referential integrity rule makes it impossible to delete a row in one table whose primary key has mandatory matching foreign key values in another table.
Example	A customer might not yet have an assigned sales representative (number), but it will be impossible to have an invalid sales representative (number).

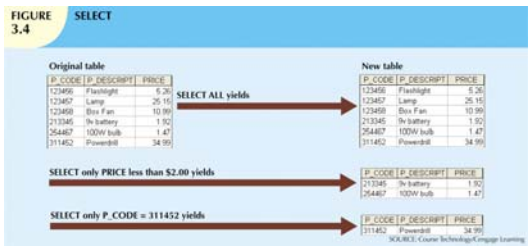
TABLE 3.5 A Dummy Variable Value Used as a Flag

AGENT_CODE	AGENT_AREACODE	AGENT_PHONE	AGENT_LNAME	AGENT_YTD_SLS
-99	000	000-0000	Nooe	$0.00



Relational Set Operators

- Relational algebra
  - Defines theoretical way of manipulating table contents using relational operators
  - Use of relational algebra operators on existing relations produces new relations:
    - SELECT
    - PROJECT
    - JOIN
    - INTERSECT
    - UNION
    - DIFFERENCE
    - PRODUCT
    - DIVIDE



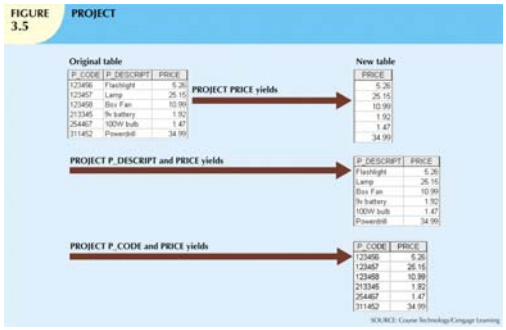


FIGURE 3.5
PROJECT
Original table
New table
PROJECT PRICE yields
PROJECT P_DESCRIPT and PRICE yields
PROJECT P_CODE and PRICE yields

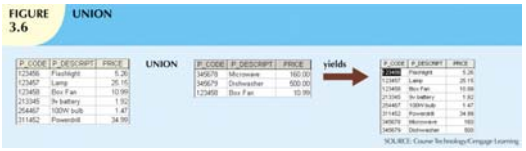


FIGURE 3.6
UNION
yields

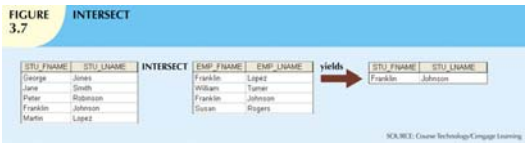


FIGURE 3.7
INTERSECT
yields

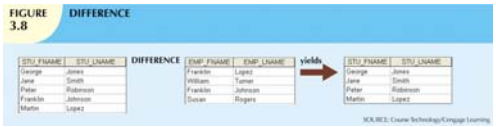


FIGURE 3.8
DIFFERENCE
yields

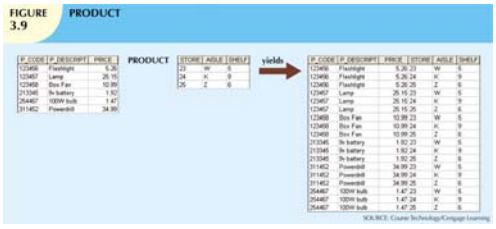


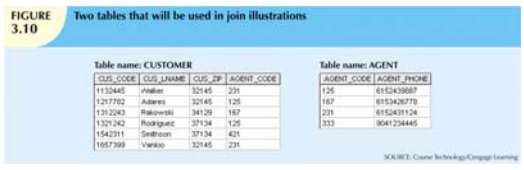
FIGURE 3.9
PRODUCT
yields

Relational Set Operators (cont'd.)

- Natural join
  - Links tables by selecting rows with common values in common attributes (join columns)
- Equijoin
  - Links tables on the basis of an equality condition that compares specified columns
- Theta join
  - Any other comparison operator is used

Relational Set Operators (cont'd.)

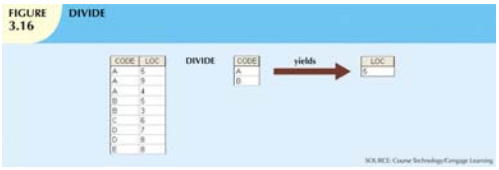
- Inner join
  - Only returns matched records from the tables that are being joined
- Outer join
  - Matched pairs are retained, and any unmatched values in other table are left null



Relational Set Operators (cont'd.)

- Left outer join
  - Yields all of the rows in the CUSTOMER table
  - Including those that do not have a matching value in the AGENT table
- Right outer join
  - Yields all of the rows in the AGENT table
  - Including those that do not have matching values in the CUSTOMER table

FIGURE 3.16 DIVIDE



The Data Dictionary and System Catalog

- Data dictionary
  - Provides detailed accounting of all tables found within the user/designer-created database
  - Contains (at least) all the attribute names and characteristics for each table in the system
  - Contains metadata: data about data
- System catalog
  - Contains metadata
  - Detailed system data dictionary that describes all objects within the database

TABLE 3.6 A Sample Data Dictionary

TABLE NAME	ATTRIBUTE NAME	CONTENTS	TYPE	FORMAT	RANGE	REQUIRED	PK or FK	FK REFERENCED TABLE
CUSTOMER	CUS_CODE	Customer account code	CHAR(5)	99999	10000–99999	Y	PK	
	CUS_LNAME	Customer last name	VARCHAR(20)	Xxxxxxxx		Y		
	CUS_FNAME	Customer first name	VARCHAR(20)	Xxxxxxxx		Y		
	CUS_INITIAL	Customer initial	CHAR(1)	X				
	CUS_RENEW_DATE	Customer insurance renewal date	DATE	dd-mmm-yyyy				
	AGENT_CODE	Agent code	CHAR(3)	999			FK	AGENT_CODE
AGENT	AGENT_CODE	Agent code	CHAR(3)	999		Y	PK	
	AGENT_AREACODE	Agent area code	CHAR(3)	999		Y		
	AGENT_PHONE	Agent telephone number	CHAR(8)	999-9999		Y		
	AGENT_LNAME	Agent last name	VARCHAR(20)	Xxxxxxxx		Y		
	AGENT_YTD_SLS	Agent year-to-date sales	NUMBER(9,2)	9,999,999.99				

FK =Foreign key
PK =Primary key
CHAR =Fixed character length data (1–255 characters)
VARCHAR =Variable character length data (1–2,000 characters)
NUMBER =Numeric data (NUMBER(9,2) are used to specify numbers with two decimal places and up to nine digits, including the decimal places. Some RDBMSs permit the use of a MONEY or CURRENCY data type.

The Data Dictionary and System Catalog (cont'd.)

- Homonym
  - Indicates the use of the same name to label different attributes
- Synonym
  - Opposite of a homonym
  - Indicates the use of different names to describe the same attribute

Relationships within the Relational Database

- 1:M relationship
  - Relational modeling ideal
  - Should be the norm in any relational database design
- 1:1 relationship
  - Should be rare in any relational database design

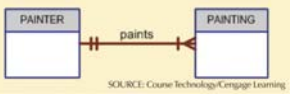
Relationships within the Relational Database (cont'd.)

- M:N relationships
  - Cannot be implemented as such in the relational model
  - M:N relationships can be changed into 1:M relationships

The 1:M Relationship

- Relational database norm
- Found in any database environment

FIGURE 3.17 The 1:M relationship between PAINTER and PAINTING



SOURCE: Course Technology/Cengage Learning

FIGURE 3.18 The implemented 1:M relationship between PAINTER and PAINTING

Table name: PAINTER
Primary key: PAINTER_NUM
Foreign key: none
Database name: Ch03_Museum

PAINTER_NUM	PAINTER_LNAME	PAINTER_FNAME	PAINTER_INITIAL
123	Ross	Georgette	P
126	Itero	Julio	G

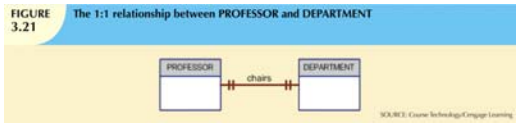
Table name: PAINTING
Primary key: PAINTING_NUM
Foreign key: PAINTER_NUM

PAINTING_NUM	PAINTING_TITLE	PAINTER_NUM
1338	Dawn Thunder	123
1339	Vanilla Roses To Nowhere	123
1340	Tired Flounders	126
1341	Hasty Exit	123
1342	Plastic Paradise	126

SOURCE: Course Technology/Cengage Learning

The 1:1 Relationship

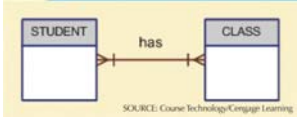
- One entity related to only one other entity, and vice versa
- Sometimes means that entity components were not defined properly
- Could indicate that two entities actually belong in the same table
- Certain conditions absolutely require their use



The M:N Relationship

- Implemented by breaking it up to produce a set of 1:M relationships
- Avoid problems inherent to M:N relationship by creating a composite entity
  - Includes as foreign keys the primary keys of tables to be linked

FIGURE 3.23 The ERM's M:N relationship between STUDENT and CLASS



SOURCE: Course Technology/Cengage Learning

FIGURE 3.25 Converting the M:N relationship into two 1:M relationships

Table name: STUDENT
Primary key: STU_NUM
Foreign key: none

Database name: Ch03_CollegeTry2

STU_NUM	STU_LNAME
321452	Bowser
324257	Smithson

Table name: ENROLL
Primary key: CLASS_CODE + STU_NUM
Foreign key: CLASS_CODE, STU_NUM

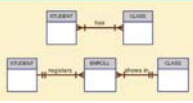
CLASS_CODE	STU_NUM	ENROLL_GRADE
10014	321452	C
10014	324257	B
10018	321452	A
10018	324257	B
10021	321452	C
10021	324257	C

Table name: CLASS
Primary key: CLASS_CODE
Foreign key: CRS_CODE

CLASS_CODE	CRS_CODE	CLASS_SECTION	CLASS_TIME	CLASS_ROOM	PROF_NUM
10014	ACCT-211	3	TTh 2:30-3:45 p.m.	BUS252	342
10018	CIS-220	2	MWF 9:00-9:50 a.m.	KLR211	114
10021	QM-261	1	MWF 8:00-8:50 a.m.	KLR200	114

SOURCE: Course Technology/Cengage Learning

FIGURE 3.26 Changing the M:N relationship to two 1:M relationships



SOURCE: Course Technology/Cengage Learning

FIGURE 3.27 The expanded entity relationship model

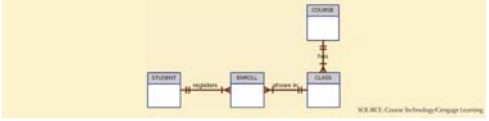
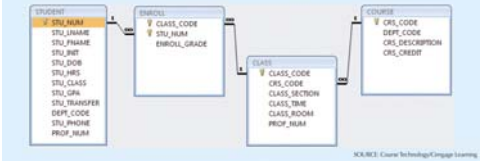


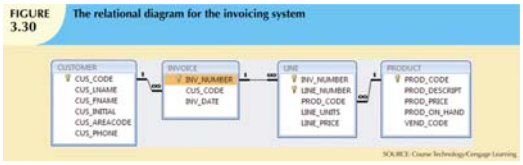
FIGURE 3.28 The relational diagram for the Ch03_TinyCollege database



Data Redundancy Revisited

- Data redundancy leads to data anomalies
  - Can destroy the effectiveness of the database
- Foreign keys
  - Control data redundancies by using common attributes shared by tables
  - Crucial to exercising data redundancy control
- Sometimes, data redundancy is necessary

FIGURE 3.30 The relational diagram for the invoicing system

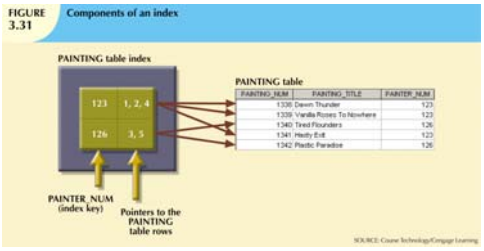


SOURCE: Course Technology/Cengage Learning

Indexes

- Orderly arrangement to logically access rows in a table
- Index key
  - Index's reference point
  - Points to data location identified by the key
- Unique index
  - Index in which the index key can have only one pointer value (row) associated with it
- Each index is associated with only one table

FIGURE 3.31 Components of an index



SOURCE: Course Technology/Cengage Learning

Codd's Relational Database Rules

- In 1985, Codd published a list of 12 rules to define a relational database system
  - Products marketed as "relational" that did not meet minimum relational standards
- Even dominant database vendors do not fully support all 12 rules

Summary

- Tables are basic building blocks of a relational database
- Keys are central to the use of relational tables
- Keys define functional dependencies
  - Superkey
  - Candidate key
  - Primary key
  - Secondary key
  - Foreign key

Summary (cont'd.)

- Each table row must have a primary key that uniquely identifies all attributes
- Tables are linked by common attributes
- The relational model supports relational algebra functions
  - SELECT, PROJECT, JOIN, INTERSECT UNION, DIFFERENCE, PRODUCT, DIVIDE
- Good design begins by identifying entities, attributes, and relationships
  - 1:1, 1:M, M:N