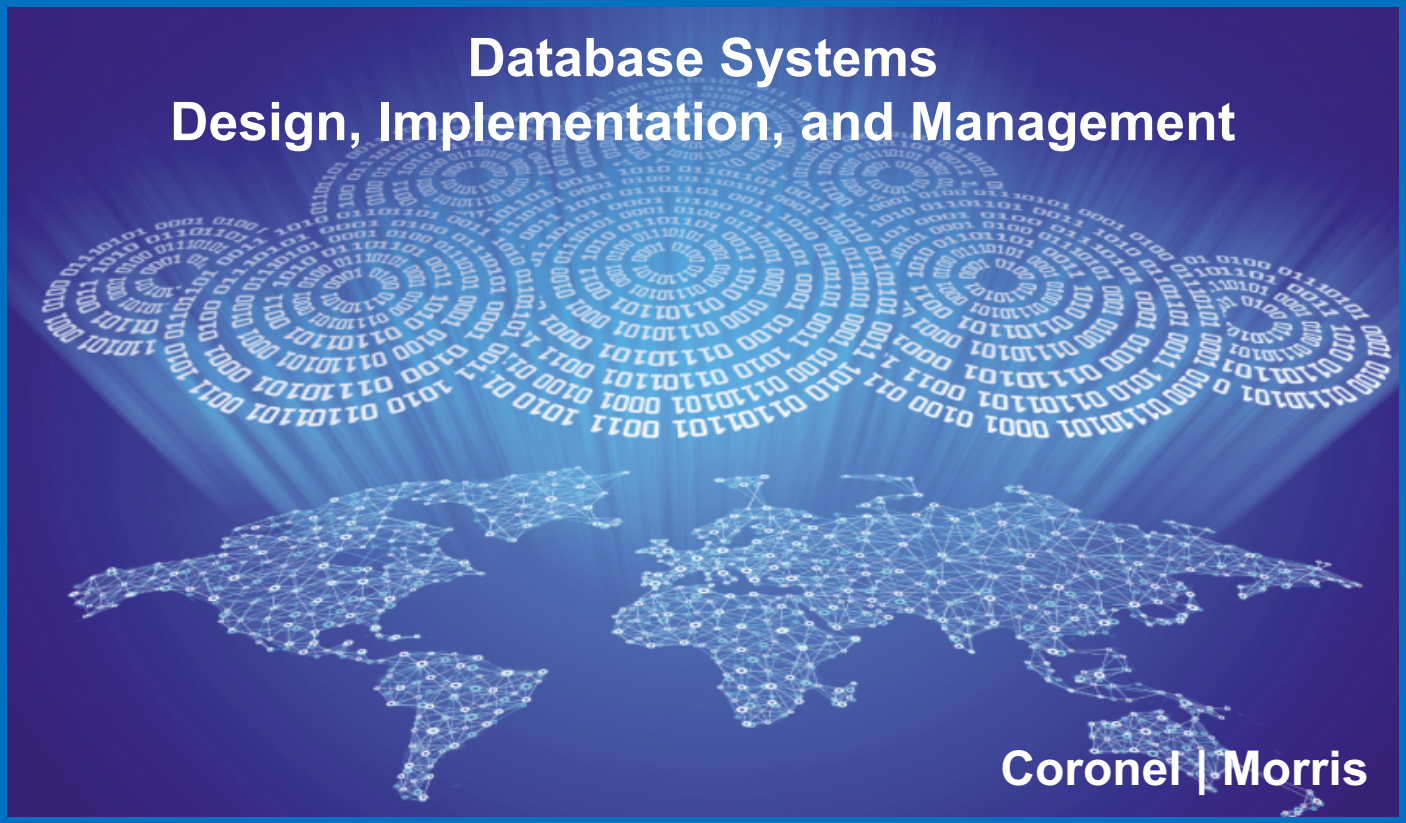
11e



Chapter 3

The Relational Database Model

Learning Objectives

- In this chapter, one 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

Learning Objectives

- In this chapter, one will learn:
  - 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
  - Enables programmer to view data logically rather than physically
  - Logical simplicity yields simple and effective database design methodologies
  - Facilitated by the creation of data relationships based on a logical construct called a relation

Tables and Their Characteristics

- Table: two-dimensional structure composed of rows and columns
  - Has advantages of structural and data independence
  - Resembles a file from conceptual point of view
  - Easier to understand than its hierarchical and network database predecessors
- Contains group of related entities = an entity set
  - Terms entity set and table are often used interchangeably

Tables and Their Characteristics (continued)

- Table also called a relation because the relational model's creator, Codd, used the term relation as a synonym for table
- Think of a table as a persistent relation:
  - A relation whose contents can be permanently saved for future use

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.

Cengage Learning © 2015

Tables and Their Characteristics (continued)

FIGURE 3.1 **STUDENT table attribute values**

Database name: Ch03_TinyCollege
Table name: STUDENT

	STU_NUM	STU_LNAME	STU_FNAME	STU_INIT	STU_DOB	STU_HRS	STU_CLASS
▶	321452	Bowser	William	C	12-Feb-1975	42	So
	324257	Smithson	Anne	K	15-Nov-1981	81	Jr
	324258	Brewer	Juliette		23-Aug-1969	36	So
	324269	Oblonski	Walter	H	16-Sep-1976	66	Jr
	324273	Smith	John	D	30-Dec-1958	102	Sr
	324274	Katinga	Raphael	P	21-Oct-1979	114	Sr
	324291	Robertson	Gerald	T	08-Apr-1973	120	Sr
	324299	Smith	John	B	30-Nov-1986	15	Fr

STUDENT table, continued

	STU_GPA	STU_TRANSFER	DEPT_CODE	STU_PHONE	PROF_NUM
▶	2.84	No	BIOL	2134	205
	3.27	Yes	CIS	2256	222
	2.26	Yes	ACCT	2256	228
	3.09	No	CIS	2114	222
	2.11	Yes	ENGL	2231	199
	3.15	No	ACCT	2267	228
	3.87	No	EDU	2267	311
	2.92	No	ACCT	2315	230

STU_HRS = Credit hours earned
STU_CLASS = Student classification
STU_DOB = Student date of birth
STU_GPA = Grade point average
STU_PHONE = 4-digit campus phone extension
PROF_NUM = Number of the professor who is the student's advisor

Keys

- Consist of one or more attributes that determine other attributes
- Used to:
  - Ensure that each row in a table is uniquely identifiable
  - Establish relationships among tables and to ensure the integrity of the data
- **Primary key (PK)**: Attribute or combination of attributes that uniquely identifies any given row
- Key's role is based on determination
  - If you know the value of attribute A, you can look up (determine) the value of attribute B

Dependencies

- **Functional dependence**: Value of one or more attributes determines the value of one or more other attributes
  - **Determinant**: Attribute whose value determines another
  - **Dependent**: Attribute whose value is determined by the other attribute
- **Full functional dependence**: Entire collection of attributes in the determinant is necessary for the relationship

Types of Keys

- **Composite key**: Key that is composed of more than one attribute
- **Key attribute**: Attribute that is a part of a key
- **Superkey**: Any key that uniquely identifies each row
- **Candidate key**: A superkey without redundancies

- **Identify superkeys, candidate keys**
  - EMPLOYEE(SSN, EmployeeID, Last Name, First Name, Salary, Date of Birth, Address, Supervisor)
  - CLASS (Course#, Prof, Sched, Room)

Keys (continued)

- **Entity integrity**: Condition in which each row in the table has its own unique identity
  - All of the values in the primary key must be unique
  - No key attribute in the primary key can contain a null

- **Referential integrity**: Every reference to an entity instance by another entity instance is valid

Keys (continued)

- Nulls:
  - No data entry
  - Not permitted in primary key
  - Should be avoided in other attributes
  - 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

Keys (continued)

- Controlled redundancy:
  - Makes the relational database work
  - Tables within the database share common attributes that enable the tables to be linked together
  - Multiple occurrences of values in a table are not redundant when they are 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

Table name: PRODUCT
Primary key: PROD_CODE
Foreign key: VEND_CODE

Database name: Ch03_SaleCo

PROD_CODE	PROD_DESCRIPT	PROD_PRICE	PROD_ON_HAND	VEND_CODE
001278-AB	Claw hammer	12.95	23	232
123-21UUY	Houselite chain saw, 16-in. bar	189.99	4	235
QER-34256	Sledge hammer, 16-lb. head	18.63	6	231
SRE-657UG	Rat-tail file	2.99	15	232
ZZX/3245Q	Steel tape, 12-ft. length	6.79	8	235

link

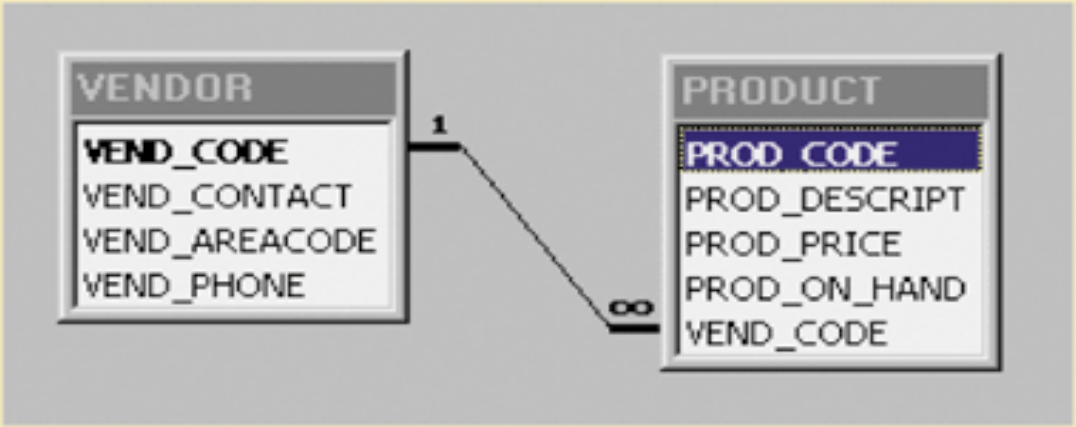
Table name: VENDOR
Primary key: VEND_CODE
Foreign key: none

VEND_CODE	VEND_CONTACT	VEND_AREACODE	VEND_PHONE
230	Shelly K. Smithson	608	555-1234
231	James Johnson	615	123-4536
232	Annelise Crystall	608	224-2134
233	Candice Wallace	904	342-6567
234	Arthur Jones	615	123-3324
235	Henry Ortozo	615	899-3425

Cengage Learning © 2015

Keys (continued)

FIGURE 3.3 The relational diagram for the Ch03_SaleCo database



Keys (continued)

- 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

Secondary Keys

- Suppose that customer data are stored in a CUSTOMER table in which the customer number is the primary key. Do you think that most customers will remember their number?
- Data retrieval for a customer is easier when the customer's last name and phone number are used.
- The primary key is the customer number; the secondary key is the combination of the customer's last name and phone number.
- A secondary key does not necessarily yield a unique outcome.
- A customer's last name and home telephone number could easily yield several matches in which one family lives together and shares a phone line.

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

Cengage Learning © 2015

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 it can be null. All invoices are uniquely identified by their invoice number.

Integrity Rules

Referential Integrity	Description
Requirement	A foreign key may have either a null entry or a entry that matches a primary key value in a table to which it is related
Purpose	It is possible for an attribute not to have a corresponding value but it is impossible to have an invalid entry It is impossible to delete row in a table whose primary keys 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 invalid sales representative number

Figure 3.3 - An Illustration of Integrity Rules

Table name: CUSTOMER **Database name: Ch03_InsureCo**
Primary key: CUS_CODE
Foreign key: AGENT_CODE

CUS_CODE	CUS_LNAME	CUS_FNAME	CUS_INITIAL	CUS_RENEW_DATE	AGENT_CODE
10010	Ramas	Alfred	A	05-Apr-2014	502
10011	Dunne	Leona	K	16-Jun-2014	501
10012	Smith	Kathy	W	29-Jan-2015	502
10013	Olowski	Paul	F	14-Oct-2014	502
10014	Orlando	Myron		28-Dec-2014	501
10015	O'Brian	Amy	B	22-Sep-2014	503
10016	Brown	James	G	25-Mar-2015	502
10017	Williams	George		17-Jul-2014	503
10018	Farriss	Anne	G	03-Dec-2014	501
10019	Smith	Olette	K	14-Mar-2015	503

Table name: AGENT (only five selected fields are shown)
Primary key: AGENT_CODE
Foreign key: none

AGENT_CODE	AGENT_AREACODE	AGENT_PHONE	AGENT_LNAME	AGENT_YTD_SLS
501	713	228-1249	Alby	132735.75
502	615	882-1244	Hahn	138967.35
503	615	123-5589	Okon	127093.45

Cengage Learning © 2015

Ways to Handle Nulls

- **Flags**: Special codes used to indicate the absence of some value
- Other integrity rules in the relational model:
  - NOT NULL constraint - Placed on a column to ensure that every row in the table has a value for that column
  - UNIQUE constraint - Restriction placed on a column to ensure that no duplicate values exist for that column

Integrity Rules (continued)

TABLE 3.5 A Dummy Variable Value Used as a Flag

AGENT_CODE	AGENT_AREACODE	AGENT_PHONE	AGENT_LNAME	AGENT_YTD_SALES
-99	000	000-0000	None	$0.00

Relational Database Operators / Relational Algebra

- Theoretical way of manipulating table contents using relational operators
- **Relvar**: Variable that holds a relation
  - Heading contains the names of the attributes and the body contains the relation
- Relational operators have the property of closure
  - **Closure**: Use of relational algebra operators on existing relations produces new relations

Relational Database Operators / Relational Algebra

- Relation is he data that we see in our tables.
- Relvar is a variable that holds a relation (a container).
- Ex: You are writing a program and created a variable name *qty* for holding integer data. The variable *qty* is not an integer itself; it is a container for holding integers.
- When you create a table, the table structure holds the table data . The structure is a *relvar*, and the data in the structure would be a relation.

Relational Algebra Operators (continued)

- SELECT
- PROJECT
- UNION
- INTERSECT
- DIFFERENCE
- PRODUCT
- JOIN
- DIVIDE

Relational Set Operators

Select (Restrict)

- Unary operator that yields a horizontal subset of a table

Project

- Unary operator that yields a vertical subset of a table

Union

- Combines all rows from two tables, excluding duplicate rows
- **Union-compatible**: Tables share the same number of columns, and their corresponding columns share compatible domains

Intersect

- Yields only the rows that appear in both tables
- Tables must be union-compatible to yield valid results

Figure 3.4 - Select

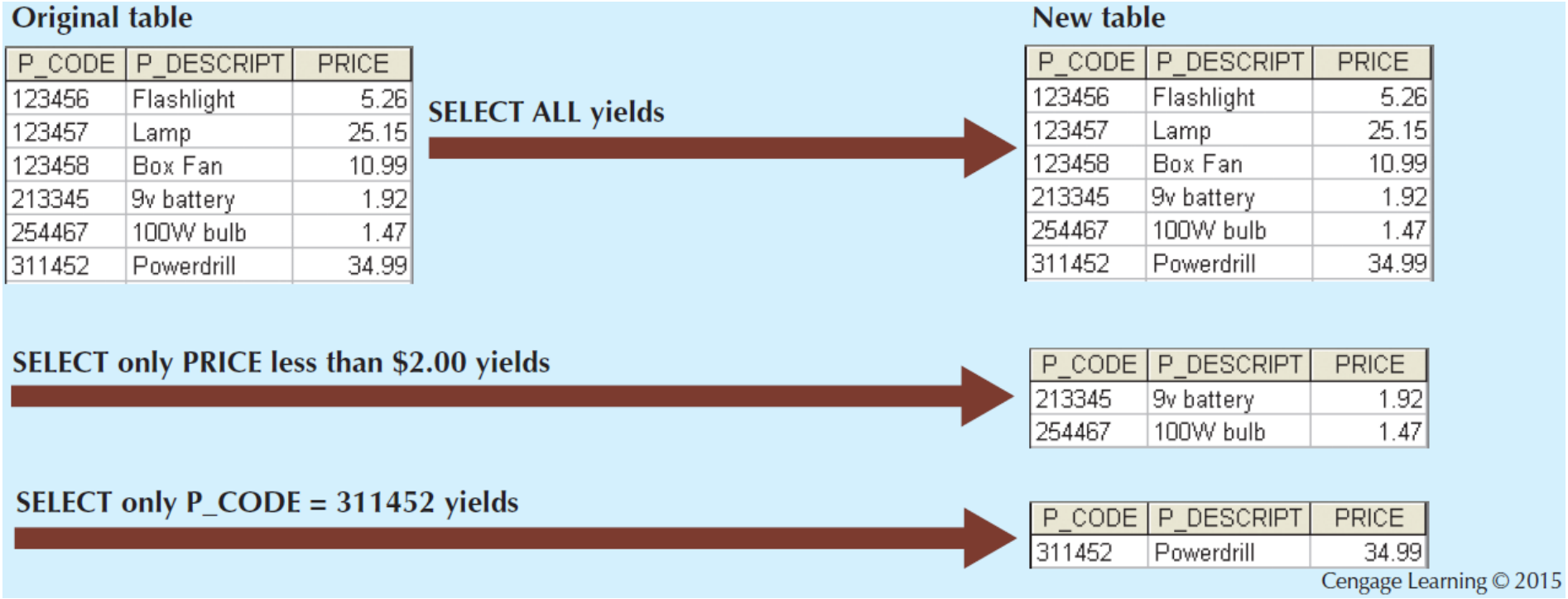


Figure 3.5 - Project

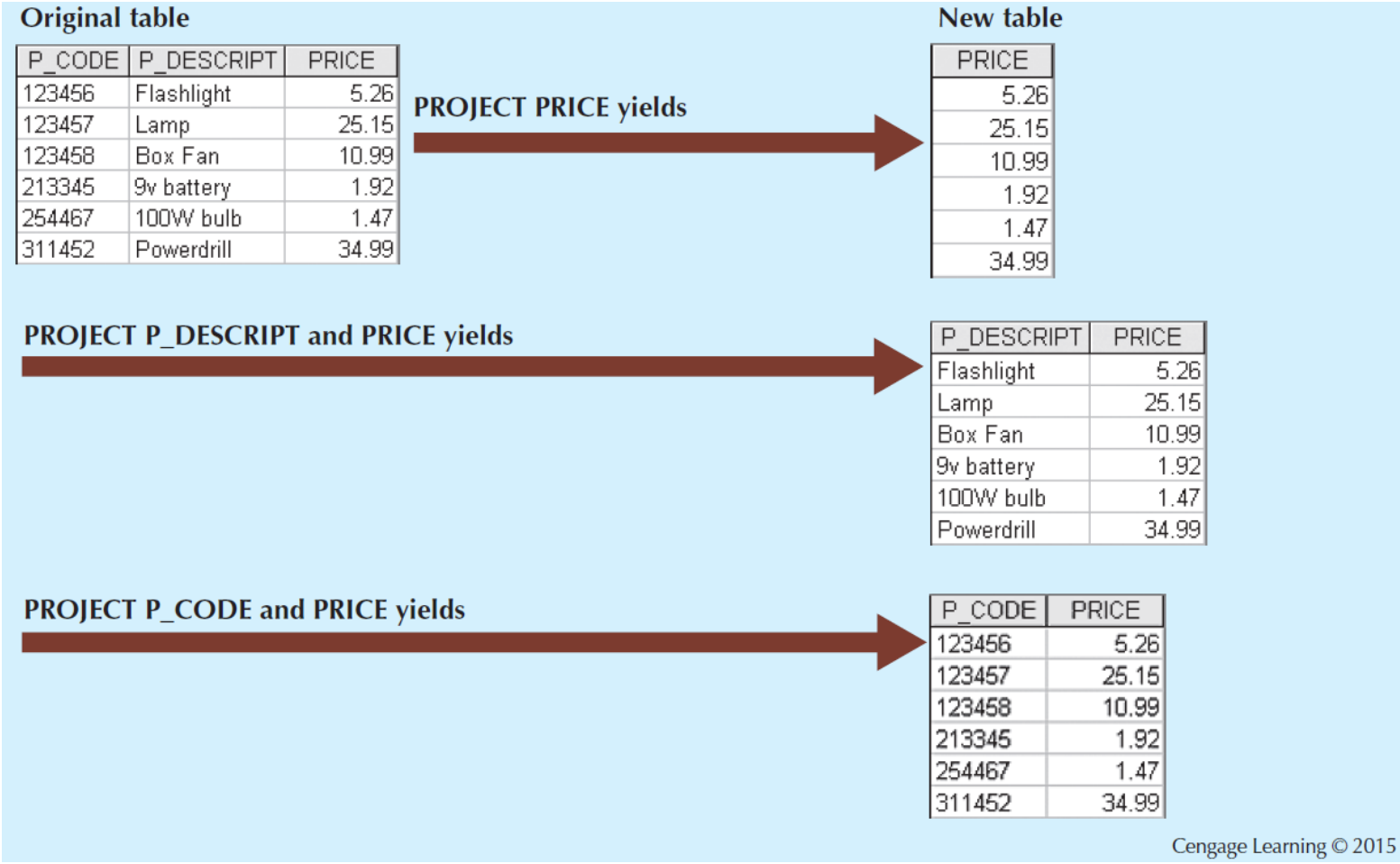


Figure 3.6 - Union

P_CODE	P_DESCRIPT	PRICE
123456	Flashlight	5.26
123457	Lamp	25.15
123458	Box Fan	10.99
213345	9v battery	1.92
254467	100W bulb	1.47
311452	Powerdrill	34.99

UNION

P_CODE	P_DESCRIPT	PRICE
345678	Microwave	160.00
345679	Dishwasher	500.00
123458	Box Fan	10.99

yields

P_CODE	P_DESCRIPT	PRICE
123456	Flashlight	5.26
123457	Lamp	25.15
123458	Box Fan	10.99
213345	9v battery	1.92
254467	100W bulb	1.47
311452	Powerdrill	34.99
345678	Microwave	160
345679	Dishwasher	500

Cengage Learning © 2015

Figure 3.7 - Intersect

STU_FNAME	STU_LNAME
George	Jones
Jane	Smith
Peter	Robinson
Franklin	Johnson
Martin	Lopez

INTERSECT

EMP_FNAME	EMP_LNAME
Franklin	Lopez
William	Turner
Franklin	Johnson
Susan	Rogers

yields

STU_FNAME	STU_LNAME
Franklin	Johnson

Cengage Learning © 2015

Relational Set Operators

- **Difference**
  - Yields all rows in one table that are not found in the other table – that is, it subtracts one table from the other
  - Tables must be union-compatible to yield valid results
- **Product**
  - Yields all possible pairs of rows from two tables
    - Also known as the Cartesian product

Figure 3.8 - Difference

STU_FNAME	STU_LNAME
George	Jones
Jane	Smith
Peter	Robinson
Franklin	Johnson
Martin	Lopez

DIFFERENCE

EMP_FNAME	EMP_LNAME
Franklin	Lopez
William	Turner
Franklin	Johnson
Susan	Rogers

yields

STU_FNAME	STU_LNAME
George	Jones
Jane	Smith
Peter	Robinson
Martin	Lopez

Cengage Learning © 2015

Figure 3.9 - Product

P_CODE	P_DESCRIPT	PRICE
123456	Flashlight	5.26
123457	Lamp	25.15
123458	Box Fan	10.99
213345	9v battery	1.92
254467	100W bulb	1.47
311452	Powerdrill	34.99

PRODUCT

STORE	AISLE	SHELF
23	W	5
24	K	9
25	Z	6

yields

P_CODE	P_DESCRIPT	PRICE	STORE	AISLE	SHELF
123456	Flashlight	5.26	23	W	5
123456	Flashlight	5.26	24	K	9
123456	Flashlight	5.26	25	Z	6
123457	Lamp	25.15	23	W	5
123457	Lamp	25.15	24	K	9
123457	Lamp	25.15	25	Z	6
123458	Box Fan	10.99	23	W	5
123458	Box Fan	10.99	24	K	9
123458	Box Fan	10.99	25	Z	6
213345	9v battery	1.92	23	W	5
213345	9v battery	1.92	24	K	9
213345	9v battery	1.92	25	Z	6
311452	Powerdrill	34.99	23	W	5
311452	Powerdrill	34.99	24	K	9
311452	Powerdrill	34.99	25	Z	6
254467	100W bulb	1.47	23	W	5
254467	100W bulb	1.47	24	K	9
254467	100W bulb	1.47	25	Z	6

Cengage Learning © 2015

Relational Set Operators

- **Join**
  - Allows information to be intelligently combined from two or more tables
  - Real power behind the relational database, allowing the use of independent tables linked by common attributes

Figure 3.10 - Two Tables That Will Be Used in JOIN Illustrations

Table name: CUSTOMER

CUS_CODE	CUS_LNAME	CUS_ZIP	AGENT_CODE
1132445	Walker	32145	231
1217782	Adares	32145	125
1312243	Rakowski	34129	167
1321242	Rodriguez	37134	125
1542311	Smithson	37134	421
1657399	Vanloo	32145	231

Table name: AGENT

AGENT_CODE	AGENT_PHONE
125	6152439887
167	6153426778
231	6152431124
333	9041234445

Cengage Learning © 2015

Types of Joins

- **Natural join**
  - Links tables by selecting only the rows with common values in their common attributes
    - **Join columns**: Common columns
  - Result of a three-stage process:
    - PRODUCT of the tables is created
    - SELECT is performed on Step 1 output to yield only the rows for which the AGENT_CODE values are equal
      - Common column(s) are called join column(s)
    - PROJECT is performed on Step 2 results to yield a single copy of each attribute, thereby eliminating duplicate columns

Relational Algebra Operators (continued)

FIGURE 3.12 Natural join, Step 1: PRODUCT

	CUS_CODE	CUS_LNAME	CUS_ZIP	CUSTOMER.AGENT_CODE	AGENT.AGENT_CODE	AGENT_PHONE
▶	1132445	Walker	32145	231	125	6152439887
	1132445	Walker	32145	231	167	6153426778
	1132445	Walker	32145	231	231	6152431124
	1132445	Walker	32145	231	333	9041234445
	1217782	Adares	32145	125	125	6152439887
	1217782	Adares	32145	125	167	6153426778
	1217782	Adares	32145	125	231	6152431124
	1217782	Adares	32145	125	333	9041234445
	1312243	Rakowski	34129	167	125	6152439887
	1312243	Rakowski	34129	167	167	6153426778
	1312243	Rakowski	34129	167	231	6152431124
	1312243	Rakowski	34129	167	333	9041234445
	1321242	Rodriguez	37134	125	125	6152439887
	1321242	Rodriguez	37134	125	167	6153426778
	1321242	Rodriguez	37134	125	231	6152431124
	1321242	Rodriguez	37134	125	333	9041234445
	1542311	Smithson	37134	421	125	6152439887
	1542311	Smithson	37134	421	167	6153426778
	1542311	Smithson	37134	421	231	6152431124
	1542311	Smithson	37134	421	333	9041234445
	1657399	Vanloo	32145	231	125	6152439887
	1657399	Vanloo	32145	231	167	6153426778
	1657399	Vanloo	32145	231	231	6152431124
	1657399	Vanloo	32145	231	333	9041234445

Relational Algebra Operators (continued)

FIGURE 3.13 Natural join, Step 2: SELECT

	CUS_CODE	CUS_LNAME	CUS_ZIP	CUSTOMER.AGENT_CODE	AGENT.AGENT_CODE	AGENT_PHONE
▶	1217782	Adares	32145	125	125	6152439887
	1321242	Rodriguez	37134	125	125	6152439887
	1312243	Rakowski	34129	167	167	6153426778
	1132445	Walker	32145	231	231	6152431124
	1657399	Vanloo	32145	231	231	6152431124

Relational Algebra Operators (continued)

FIGURE 3.14 Natural join, Step 3: PROJECT

CUS_CODE	CUS_LNAME	CUS_ZIP	AGENT_CODE	AGENT_PHONE
217782	Adares	32145	125	6152439887
1321242	Rodriguez	37134	125	6152439887
1312243	Rakowski	34129	167	6153426778
1132445	Walker	32145	231	6152431124
1657399	Vanloo	32145	231	6152431124

Relational Algebra Operators (continued)

- Natural Join:
  - Final outcome yields table that
    - Does not include unmatched pairs
    - Provides only copies of matches
  - If no match is made between the table rows
    - the new table does not include the unmatched row

Relational Algebra Operators (continued)

- Natural Join (continued):
  - The column on which the join was made - that is, AGENT_CODE - occurs only once in the new table
  - If the same AGENT_CODE were to occur several times in the AGENT table,
    - a customer would be listed for each match

Types of Joins

- **Equijoin**
  - Links tables on the basis of an equality condition that compares specified columns of each table
  - Outcome does not eliminate duplicate columns
  - Condition or criterion to join tables must be explicitly defined
  - Takes its name from the equality comparison operator (=) used in the condition
- **Theta join**:
  - If any other comparison operator is used

Types of Joins

- **Inner join**: Only returns matched records from the tables that are being joined
- **Outer join**: Matched pairs are retained and unmatched values in the other table are left null
  - **Left outer join**: Yields all of the rows in the first table, including those that do not have a matching value in the second table
  - **Right outer join**: Yields all of the rows in the second table, including those that do not have matching values in the first table

Relational Algebra Operators (continued)

- Outer join:
  - Matched pairs are retained and any unmatched values in other table are left null
  - In outer join for tables CUSTOMER and AGENT, two scenarios are possible:
    - Left outer join
      - Yields all rows in CUSTOMER table, including those that do not have a matching value in the AGENT table
    - Right outer join
      - Yields all rows in AGENT table, including those that do not have matching values in the CUSTOMER table

Relational Algebra Operators

FIGURE 3.15 Left outer join

	CUS_CODE	CUS_LNAME	CUS_ZIP	AGENT_CODE	AGENT_PHONE
▶	1217782	Adares	32145	125	6152439887
	1321242	Rodriguez	37134	125	6152439887
	1312243	Rakowski	34129	167	6153426778
	1132445	Walker	32145	231	6152431124
	1657399	Vanloo	32145	231	6152431124
	1542311	Smithson	37134	421	

Relational Algebra Operators (continued)

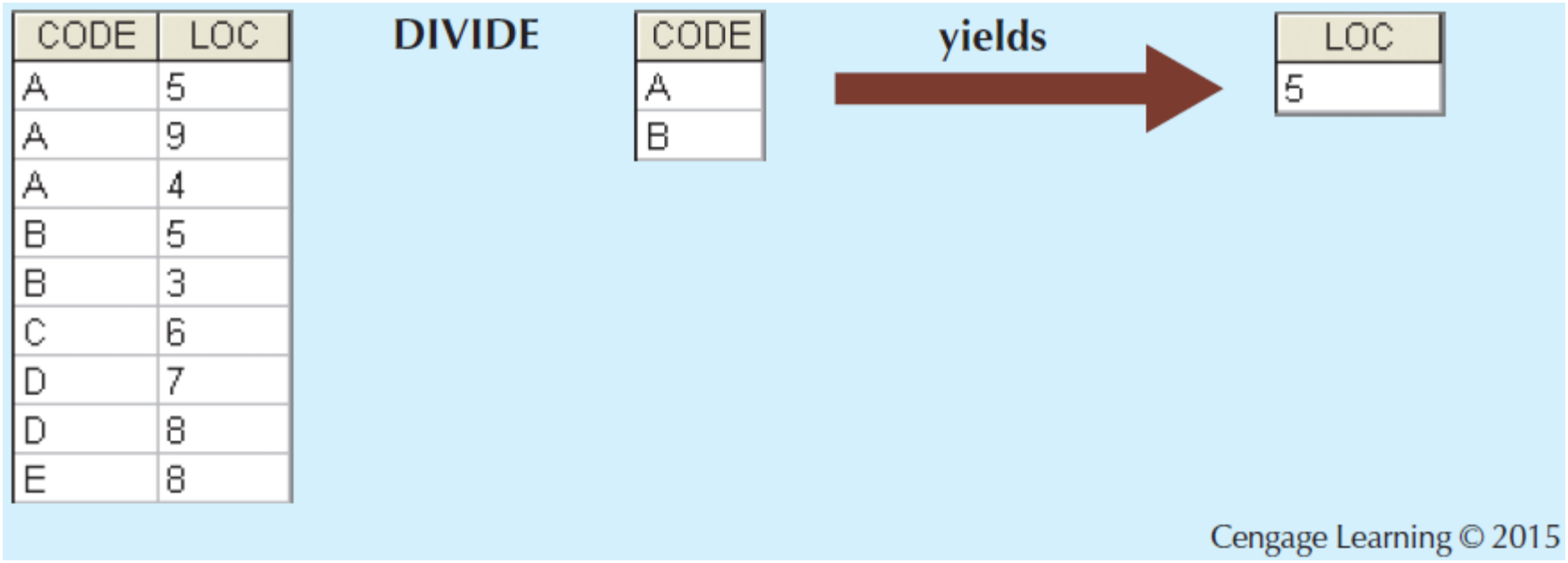
FIGURE 3.16 Right outer join

	CUS_CODE	CUS_LNAME	CUS_ZIP	AGENT_CODE	AGENT_PHONE
▶	1217782	Adares	32145	125	6152439887
	1321242	Rodriguez	37134	125	6152439887
	1312243	Rakowski	34129	167	6153426778
	1132445	Walker	32145	231	6152431124
	1657399	Vanloo	32145	231	6152431124
				333	9041234445

Relational Set Operators

- **Divide**
  - Uses one 2-column table as the dividend and one single-column table as the divisor
  - Output is a single column that contains all values from the second column of the dividend that are associated with every row in the divisor

Figure 3.16 - Divide



USABLE

Flight	Equip
83	727
83	747
84	727
84	747
109	707

CERTIFIED

Pilot	Equip
Smith	727
Smith	747
Jones	727
Jones	747
Hill	727
Hill	747
Smith	707

Find those pilots who are certified to fly any type of equipment in any flight

Relational Algebra Queries

- The relational algebra is a procedural language
- queries in relational algebra specify *how* to produce a result
- *How* to produce a result should be the responsibility of the system
- User queries should be declarative specifying *what* is to be retrieved

Data Dictionary and the System Catalog

- Data dictionary
  - Provides detailed description 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
  - In short, data dictionary contains metadata—data about data
  - Sometimes described as “the database designer’s database” because it records the design decisions about tables and their structures

A Sample Data Dictionary

TABLE 3.6 A Sample Data Dictionary

TABLE NAME	ATTRIBUTE NAME	CONTENTS	TYPE	FORMAT	RANGE	REQUIRED	PK OR FK	FK REFERENC TABLE
CUSTOMER	CUS_CODE	Customer account code	CHAR(5)	99999	10000–99999	Y	PK	
	CUS_LNAME	Customer last name	VCCHAR(20)	Xxxxxxxx		Y		
	CUS_FNAME	Customer first name	VCHAR(20	Xxxxxxxx		Y		
	CUS_INITIAL	Customer initial	CHAR(1)	X	100–999		FK	AGENT
	CUS_RENEW_DATE	Customer insurance renewal date	DATE	dd-mmm-yyyy				
	AGENT_CODE	Agent code	CHAR(3)	999				
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	0.00–9,999,999.99	Y		
	AGENT_LNAME	Agent last name	VCHAR(20)	Xxxxxxxx		Y		
	AGENT_YTD_SLS	Agent year-to-date sales	NUMBER(9,2)	9,999,999.99		Y		

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) is used to specify numbers with two decimal places and up to nine digits, including the decimal places. Some RDB permit the use of a MONEY or CURRENCY data type.)

Note: Telephone area codes are always composed of digits 0–9. Because area codes are not used arithmetically, they are most efficiently stored as character data. A the area codes are always composed of three digits. Therefore, the area code data type is defined as CHAR(3). On the other hand, names do not conform to some stand length. Therefore, the customer first names are defined as VARCHAR(20), thus indicating that up to 20 characters may be used to store the names. Character data shown as left-justified.

Data Dictionary and the System Catalog

- System catalog
  - Contains metadata
  - Detailed system data dictionary that describes all objects within the database
  - Terms “system catalog” and “data dictionary” are often used interchangeably
  - Can be queried just like any user/designer-created table
- Homonyms and synonyms must be avoided to lessen confusion
  - **Homonym**: Same name is used to label different attributes
  - **Synonym**: Different names are used 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
- M:N relationships
  - Cannot be implemented as such in the relational model
  - M:N relationships can be changed into two 1:M relationships
  - **Composite entity** (**Bridge** or **associative entity**): Helps avoid problems inherent to M:N relationships, includes the primary keys of tables to be linked

The 1:M Relationship

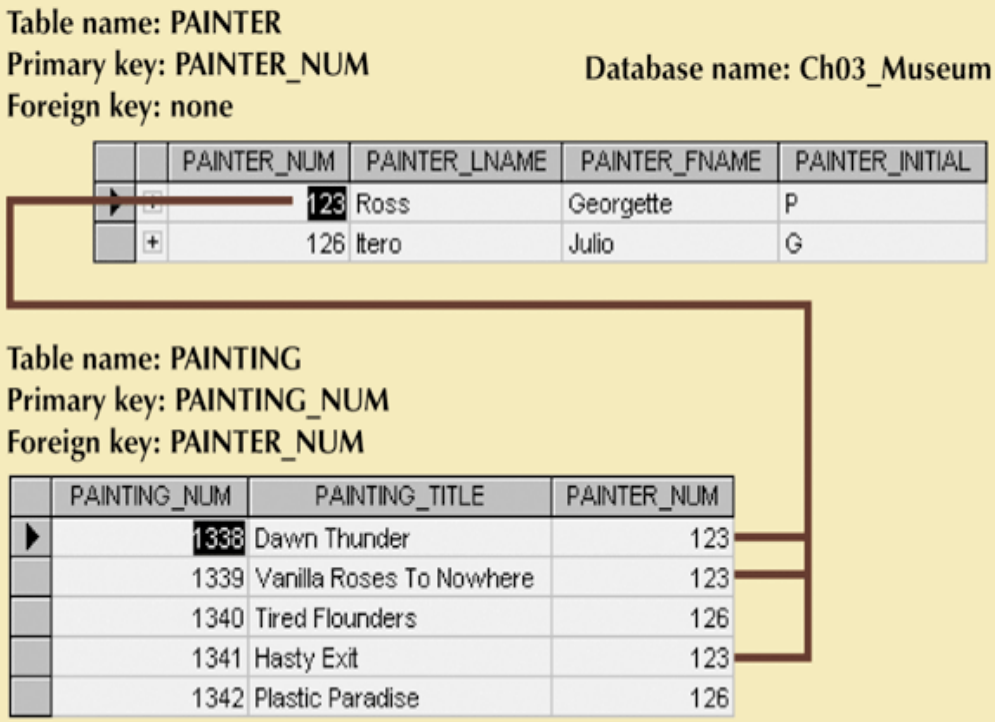
- Relational database norm
- Found in any database environment

The 1:M Relationship (continued)

FIGURE 3.18 The 1:M relationship between PAINTER and PAINTING

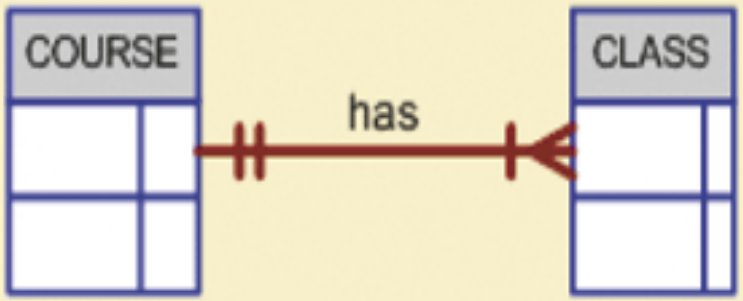
The 1:M Relationship (continued)

FIGURE 3.19 The implemented 1:M relationship between PAINTER and PAINTING



The 1:M Relationship (continued)

FIGURE 3.20 The 1:M relationship between COURSE and CLASS



The 1:M Relationship (continued)

FIGURE 3.21 The implemented 1:M relationship between COURSE and CLASS

Table name: COURSE
Primary key: CRS_CODE
Foreign key: none
Database name: Ch03_TinyCollege

CRS_CODE	DEPT_CODE	CRS_DESCRIPTION	CRS_CREDIT
ACCT-211	ACCT	Accounting I	3
ACCT-212	ACCT	Accounting II	3
CIS-220	CIS	Intro. to Microcomputing	3
CIS-420	CIS	Database Design and Implementation	4
QM-261	CIS	Intro. to Statistics	3
QM-362	CIS	Statistical Applications	4

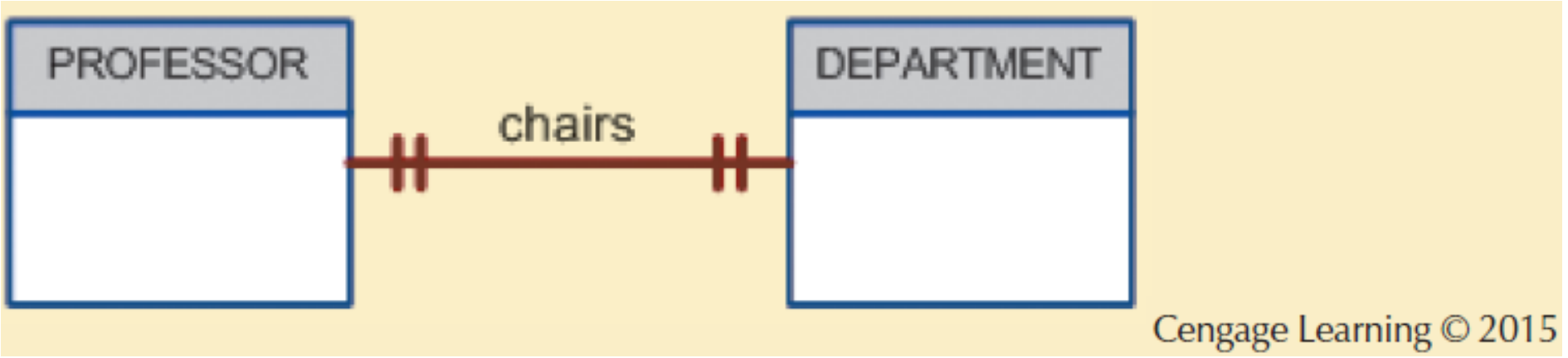
Table name: CLASS
Primary key: CLASS_CODE
Foreign key: CRS_CODE

CLASS_CODE	CRS_CODE	CLASS_SECTION	CLASS_TIME	CLASS_ROOM	PROF_NUM
10012	ACCT-211	1	MWF 8:00-8:50 a.m.	BUS311	105
10013	ACCT-211	2	MWF 9:00-9:50 a.m.	BUS200	105
10014	ACCT-211	3	TTh 2:30-3:45 p.m.	BUS252	342
10015	ACCT-212	1	MWF 10:00-10:50 a.m.	BUS311	301
10016	ACCT-212	2	Th 6:00-8:40 p.m.	BUS252	301
10017	CIS-220	1	MWF 9:00-9:50 a.m.	KLR209	228
10018	CIS-220	2	MWF 9:00-9:50 a.m.	KLR211	114
10019	CIS-220	3	MWF 10:00-10:50 a.m.	KLR209	228
10020	CIS-420	1	W 6:00-8:40 p.m.	KLR209	162
10021	QM-261	1	MWF 8:00-8:50 a.m.	KLR200	114
10022	QM-261	2	TTh 1:00-2:15 p.m.	KLR200	114
10023	QM-362	1	MWF 11:00-11:50 a.m.	KLR200	162
10024	QM-362	2	TTh 2:30-3:45 p.m.	KLR200	162

The 1:1 Relationship

- One entity can be 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
- As rare as 1:1 relationships should be, certain conditions absolutely require their use

Figure 3.21 - The 1:1 Relationship between PROFESSOR and DEPARTMENT



Th ed)

FIGURE 3.23 The implemented 1:1 relationship between PROFESSOR and DEPARTMENT

Table name: PROFESSOR
Primary key: EMP_NUM
Foreign key: DEPT_CODE

Database name: Ch03_TinyCollege

EMP_NUM	DEPT_CODE	PROF_OFFICE	PROF_EXTENSION	PROF_HIGH_DEGREE
103	HIST	DRE 156	6783	Ph.D.
104	ENG	DRE 102	5561	MA
105	ACCT	KLR 229D	8665	Ph.D.
106	MKT/MGT	KLR 126	3899	Ph.D.
110	BIOL	AAK 160	3412	Ph.D.
114	ACCT	KLR 211	4436	Ph.D.
155	MATH	AAK 201	4440	Ph.D.
160	ENG	DRE 102	2248	Ph.D.
162	CIS	KLR 203E	2359	Ph.D.
191	MKT/MGT	KLR 409B	4016	DBA
195	PSYCH	AAK 297	3550	Ph.D.
209	CIS	KLR 333	3421	Ph.D.
228	CIS	KLR 300	3000	Ph.D.
297	MATH	AAK 194	1145	Ph.D.
299	ECON/FIN	KLR 284	2851	Ph.D.
301	ACCT	KLR 244	4683	Ph.D.
335	ENG	DRE 208	2000	Ph.D.
342	SOC	BBG 208	5514	Ph.D.
387	BIOL	AAK 230	8665	Ph.D.
401	HIST	DRE 156	6783	MA
425	ECON/FIN	KLR 284	2851	MBA
435	ART	BBG 185	2278	Ph.D.

The 1:M DEPARTMENT employs PROFESSOR relationship is implemented through the placement of the DEPT_CODE foreign key in the PROFESSOR table.

The 1:1 PROFESSOR chairs DEPARTMENT relationship is implemented through the placement of the EMP_NUM foreign key in the DEPARTMENT table.

Table name: DEPARTMENT
Primary key: DEPT_CODE
Foreign key: EMP_NUM

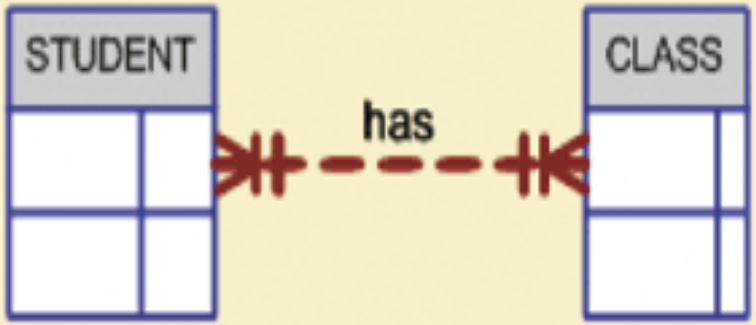
DEPT_CODE	DEPT_NAME	SCHOOL_CODE	EMP_NUM	DEPT_ADDRESS	DEPT_EXTENSION
ACCT	Accounting	BUS	114	KLR 211, Box 52	3119
ART	Fine Arts	A&SCI	435	BBG 185, Box 128	2278
BIOL	Biology	A&SCI	387	AAK 230, Box 415	4117
CIS	Computer Info. Systems	BUS	209	KLR 333, Box 56	3245
ECON/FIN	Economics/Finance	BUS	299	KLR 284, Box 63	3126
ENG	English	A&SCI	160	DRE 102, Box 223	1004
HIST	History	A&SCI	103	DRE 156, Box 284	1867
MATH	Mathematics	A&SCI	297	AAK 194, Box 422	4234
MKT/MGT	Marketing/Management	BUS	106	KLR 126, Box 55	3342
PSYCH	Psychology	A&SCI	195	AAK 297, Box 438	4110
SOC	Sociology	A&SCI	342	BBG 208, Box 132	2008

The M:N Relationship

- Can be implemented by breaking it up to produce a set of 1:M relationships
- Can avoid problems inherent to M:N relationship by creating a composite entity or bridge entity

The M:N Relationship (continued)

FIGURE 3.24 The ERD's M:N relationship between STUDENT and CLASS



The M:N Relationship (continued)

TABLE 3.7 Sample Student Enrollment Data

STUDENT'S LAST NAME	SELECTED CLASSES
Bowser	Accounting 1, ACCT-211, code 10014 Intro to Microcomputing, CIS-220, code 10018 Intro to Statistics, QM-261, code 10021
Smithson	Accounting 1, ACCT-211, code 10014 Intro to Microcomputing, CIS-220, code 10018 Intro to Statistics, QM-261, code 10021

The M:N Relationship (continued)

FIGURE 3.25 The M:N relationship between STUDENT and CLASS

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

Database name: Ch03_CollegeTry

	STU_NUM	STU_LNAME	CLASS_CODE
▶	321452	Bowser	10014
	321452	Bowser	10018
	321452	Bowser	10021
	324257	Smithson	10014
	324257	Smithson	10018
	324257	Smithson	10021

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

	CLASS_CODE	STU_NUM	CRS_CODE	CLASS_SECTION	CLASS_TIME	CLASS_ROOM	PROF_NUM
▶	10014	321452	ACCT-211	3	TTh 2:30-3:45 p.m.	BUS252	342
	10014	324257	ACCT-211	3	TTh 2:30-3:45 p.m.	BUS252	342
	10018	321452	CIS-220	2	MWF 9:00-9:50 a.m.	KLR211	114
	10018	324257	CIS-220	2	MWF 9:00-9:50 a.m.	KLR211	114
	10021	321452	QM-261	1	MWF 8:00-8:50 a.m.	KLR200	114
	10021	324257	QM-261	1	MWF 8:00-8:50 a.m.	KLR200	114

The M:N Relationship (continued)

- Implementation of a composite entity
- Yields required M:N to 1:M conversion
- Composite entity table must contain at least the primary keys of original tables
- Linking table contains multiple occurrences of the foreign key values
- Additional attributes may be assigned as needed

Figure 3.26 - Changing the M:N Relationship to Two 1:M Relationships

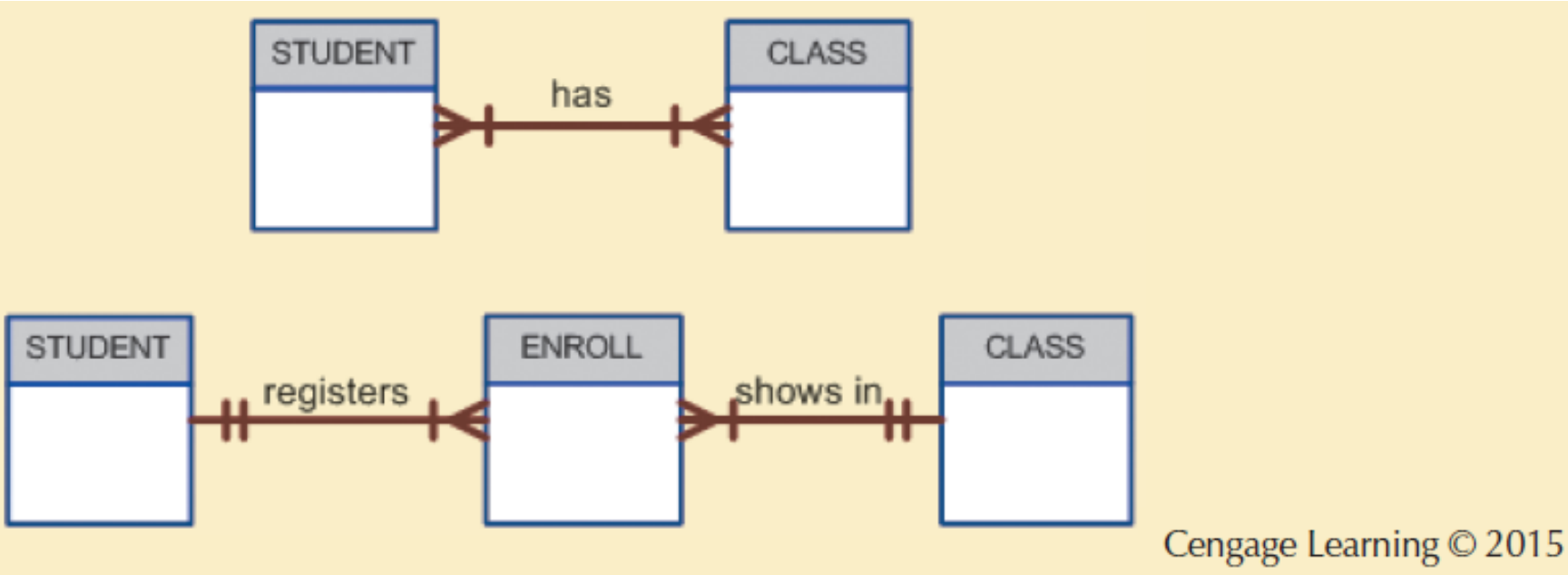
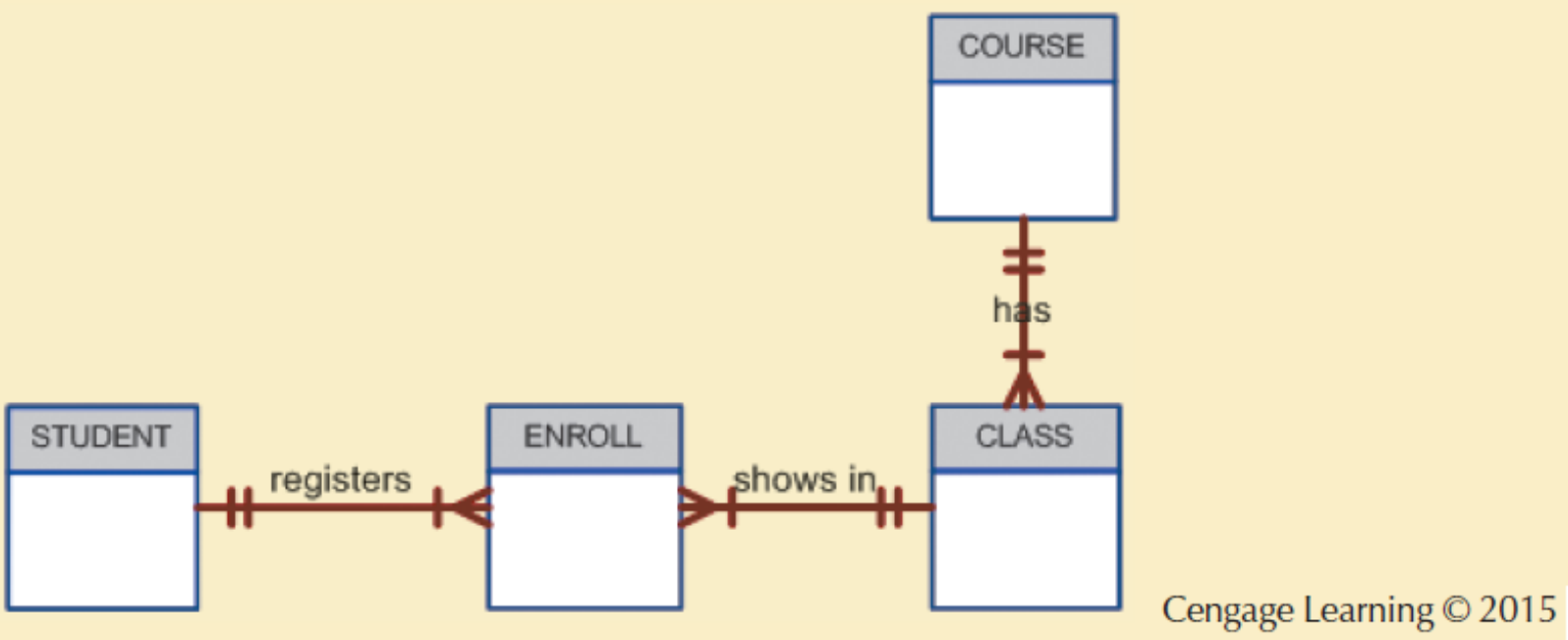


Figure 3.27 - The Expanded ER Model



The M:N Relationship (continued)

FIGURE 3.29 The relational diagram for the Ch03_TinyCollege database

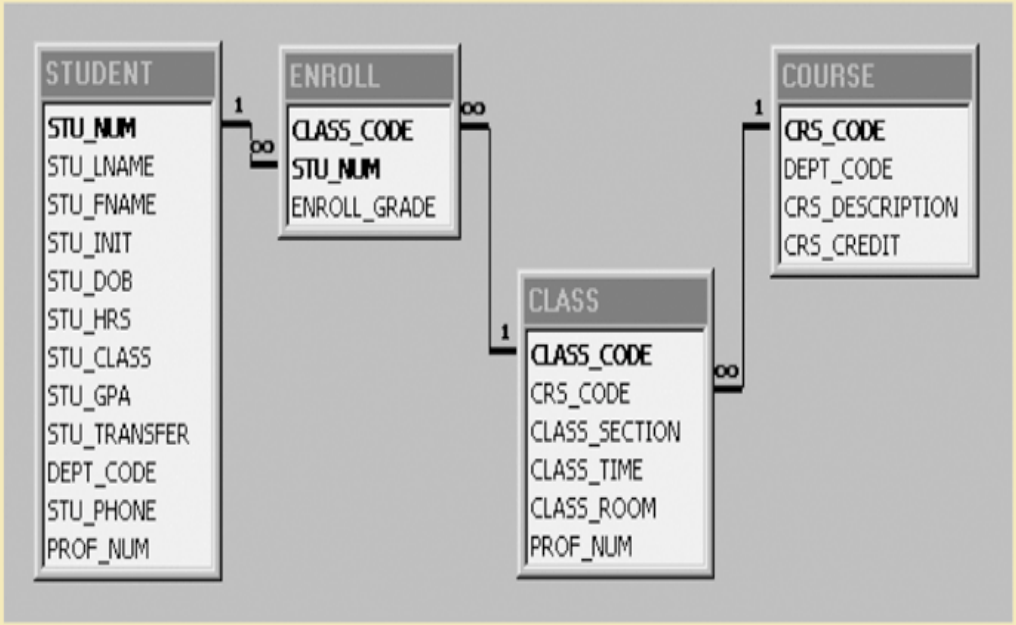


FIGURE 3.26 Converting the M:N relationship into two 1:M relationships

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

Data Redundancy Revisited

- Data redundancy leads to data anomalies
  - Such 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
  - Data redundancy must be increased to make the database serve crucial information purposes
  - Exists to preserve the historical accuracy of the data

FIGURE 3.30 A small invoicing system

Table name: CUSTOMER
Primary key: CUS_CODE **Database name: Ch03_SaleCo**
Foreign key: none

CUS_CODE	CUS_LNAME	CUS_FNAME	CUS_INITIAL	CUS_AREACODE	CUS_PHONE
10010	Ramas	Alfred	A	615	844-2573
10011	Dunne	Leona	K	713	894-1238
10012	Smith	Kathy	W	615	894-2285
10013	Olowski	Paul	F	615	894-2180
10014	Orlando	Myron		615	222-1672
10015	O'Brian	Amy	B	713	442-3381
10016	Brown	James	G	615	297-1228
10017	Williams	George		615	290-2556
10018	Farriss	Anne	G	713	382-7185
10019	Smith	Olette	K	615	297-3809

Table name: INVOICE
Primary key: INV_NUMBER
Foreign key: CUS_CODE

INV_NUMBER	CUS_CODE	INV_DATE
1001	10014	08-Mar-06
1002	10011	08-Mar-06
1003	10012	08-Mar-06
1004	10011	09-Mar-06

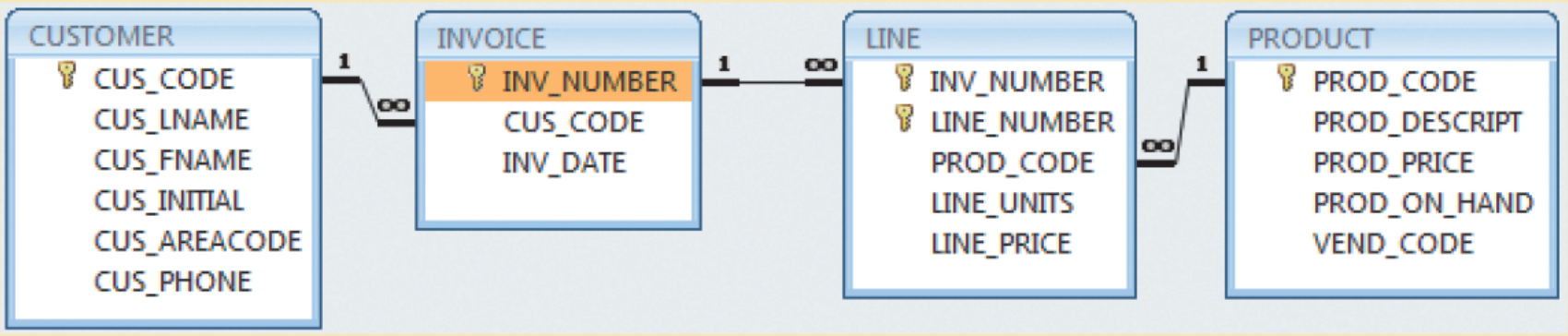
Table name: LINE
Primary key: INV_NUMBER + LINE_NUMBER
Foreign keys: INV_NUMBER, PROD_CODE

INV_NUMBER	LINE_NUMBER	PROD_CODE	LINE_UNITS	LINE_PRICE
1001	1	123-21UUY	1	$189.99
1001	2	SRE-657UG	3	$2.99
1002	1	QER-34256	2	$18.63
1003	1	ZZX/3245Q	1	$6.79
1003	2	SRE-657UG	1	$2.99
1003	3	001278-AB	1	$12.95
1004	1	001278-AB	1	$12.95
1004	2	SRE-657UG	2	$2.99

Table name: PRODUCT
Primary key: PROD_CODE
Foreign key: none

PROD_CODE	PROD_DESCRIPT	PROD_PRICE	PROD_ON_HAND	VEND_CODE
001278-AB	Claw hammer	$12.95	23	232
123-21UUY	Houselite chain saw, 16-in. bar	$189.99	4	235
QER-34256	Sledge hammer, 16-lb. head	$18.63	6	231
SRE-657UG	Rat-tail file	$2.99	15	232
ZZX/3245Q	Steel tape, 12-ft. length	$6.79	8	235

Figure 3.30 - The Relational Diagram for the Invoicing System

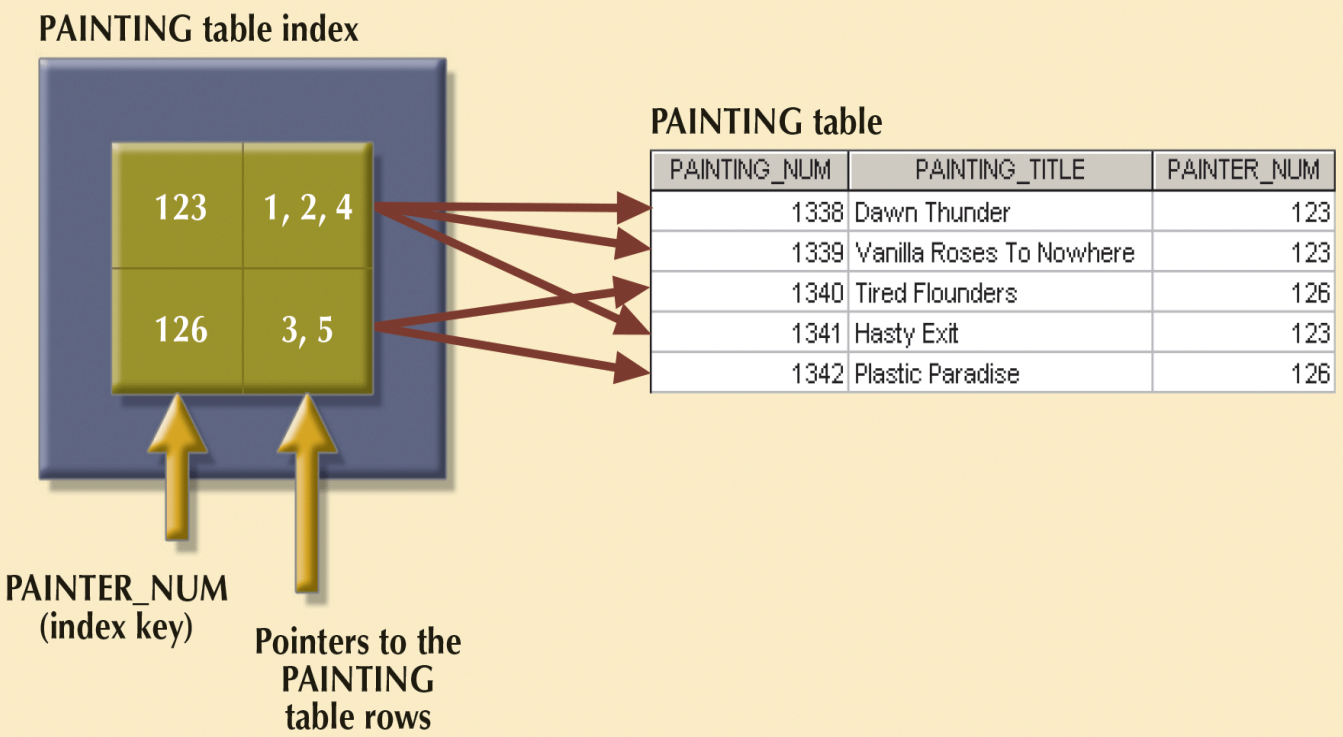


Cengage Learning © 2015

Index

- Arrangement used 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

Indexes (continued)



Codd's Relational Database Rules

- In 1985, Codd published a list of 12 rules to define a relational database system
- The reason was the concern that many vendors were marketing products as "relational" even though those products did not meet minimum relational standards

Codd's Relational Database Rules

TABLE 3.8 Dr. Codd's 12 Relational Database Rules

RULE	RULE NAME	DESCRIPTION
1	**Information**	All information in a relational database must be logically represented as column values in rows within tables.
2	**Guaranteed Access**	Every value in a table is guaranteed to be accessible through a combination of table name, primary key value, and column name.
3	**Systematic Treatment of Nulls**	Nulls must be represented and treated in a systematic way, independent of data type.
4	**Dynamic On-Line Catalog Based on the Relational Model**	The metadata must be stored and managed as ordinary data, that is, in tables within the database. Such data must be available to authorized users using the standard database relational language.
5	**Comprehensive Data Sublanguage**	The relational database may support many languages. However it must support one well- defined declarative language with support for data definition, view definition, data manipulation (interactive and by program), integrity constraints, authorization, and transaction management (begin, commit, and rollback).
6	**View Updating**	Any view that is theoretically updatable must be updatable through the system.

Codd's Relational Database Rules (Continued)

7	**High-Level Insert, Update and Delete**	The database must support set-level inserts, updates, and deletes.
8	**Physical Data Independence**	Application programs and ad hoc facilities are logically unaffected when physical access methods or storage structures are changed.
9	**Logical Data Independence**	Application programs and ad hoc facilities are logically unaffected when changes are made to the table structures that preserve the original table values (changing order of column or inserting columns).
10	**Integrity Independence**	All relational integrity constraints must be definable in the relational language and stored in the system catalog, not at the application level.
11	**Distribution Independence**	The end users and application programs are unaware and unaffected by the data location (distributed vs. local databases).
12	**Nonsubversion**	If the system supports low-level access to the data, there must not be a way to bypass the integrity rules of the database.
	Rule Zero	All preceding rules are based on the notion that in order for a database to be considered relational, it must use its relational facilities exclusively to manage the database.

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 (continued)

- Each table row must have a primary key which uniquely identifies all attributes
- Tables can be linked by common attributes. Thus, the primary key of one table can appear as the foreign key in another table to which it is linked
- The relational model supports relational algebra functions: SELECT, PROJECT, JOIN, INTERSECT, UNION, DIFFERENCE, PRODUCT, and DIVIDE.
- Good design begins by identifying appropriate entities and attributes and the relationships among the entities. Those relationships (1:1, 1:M, and M:N) can be represented using ERDs.