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 database model enables logical representation of the data and its relationships
- Logical simplicity yields simple and effective database design methodologies
- Facilitated by the creation of data relationships based on a logical construct called a relation

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

2. Given the table contents, the STUDENT entity set includes eight distinct entities (rows or students.
5. Most DBMSs support numeric, character, date, logical data types.
6. If STU_GPA values are limited to the range 0-4, inclusive, the domain is [0,4].

STUDENT Table Attribute Values

Table name: STUDENT

STU_NUM	STU_LNAME	STU_FNAME	STU_INIT	STU_DOB	STU_HRS	STU_CLASS	STU_GPA	STU_TRANSFER	DEPT_CODE	STU_PHONE	PROF_NUM
321452	Bowser	William	C	12-Feb-1985	42	So	2.84	No	BIOL	2134	205
324257	Smithson	Anne	K	15-Nov-1991	81	Jr	3.27	Yes	CIS	2256	222
324258	Brewer	Juliette		23-Aug-1979	36	So	2.26	Yes	ACCT	2256	228
324269	Oblonski	Walter	H	16-Sep-1986	66	Jr	3.09	No	CIS	2114	222
324273	Smith	John	D	30-Dec-1968	102	Sr	2.11	Yes	ENGL	2231	199
324274	Katinga	Raphael	P	21-Oct-1989	114	Sr	3.15	No	ACCT	2267	228
324291	Robertson	Gerald	T	08-Apr-1983	120	Sr	3.87	No	EDU	2267	311
324299	Smith	John	B	30-Nov-1996	15	Fr	2.92	No	ACCT	2315	230

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

You can identify the two foreign keys from naming convention.

Cengage Learning © 2015

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

Determination

- State in which knowing the value of one attribute makes it possible to determine the value of another
  - E.g. Given a formula: revenue – cost = profit, you can determine one of the three values when the other two are available.
- Is the basis for establishing the role of a key
- Based on the relationships among the attributes

Dependencies

- **Functional dependence**: Value of one or more attributes determines the value of one or more other attributes (e.g. in STUDENT table, STU_NUM → STU_LNAME)
  - **Determinant**: Attribute whose value determines another (e.g. STU_NUM in the example)
  - **Dependent**: Attribute whose value is determined by the other attribute (e.g. STU_LNAME in the example)
- functional dependence can involve a determinant that comprises more than one attribute and multiple dependent attributes.
  - STU_NUM → (STU_LNAME, STU_FNAME, STU_GPA)
  - (STU_FNAME, STU_LNAME, STU_INIT, STU_PHONE) → (STU_DOB, STU_HRS, STU_GPA)
  - (STU_NUM, STU_LNAME) → STU_GPA (but, STU_LNAME is unnecessary for the relationship)
- **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
  - (STU_LNAME, STU_FNAME, STU_INIT, STU_PHONE) → STU_HRS
- **Key attribute**: Attribute that is a part of a key
  - In the last example, the composite key consists of four key attributes.
- **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

Types of Keys

- **Null**: Absence of any data value that could represent:
  - An unknown attribute value
  - A known, but missing, attribute value
  - A inapplicable condition
- A null is no value at all. It does *not* mean a zero or a space, and it can create problems when functions such as COUNT, AVERAGE, and SUM are used.
- **Referential integrity**: Every reference to an entity instance by another entity instance is valid
  - every foreign key entry must either be null or a valid value in the primary key of the related table.

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

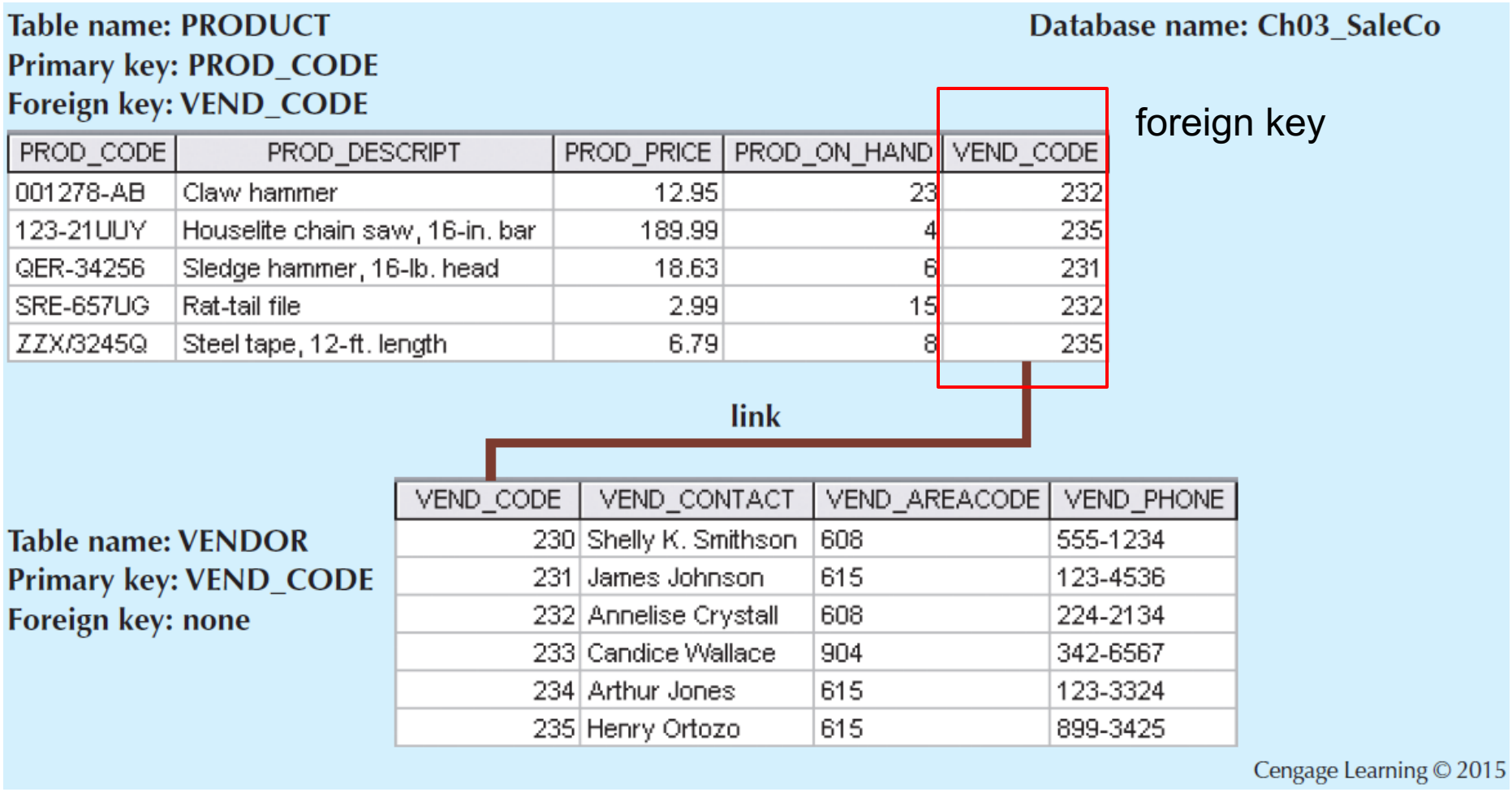
STU_NUM is a superkey, as are the composite keys (STU_NUM, STU_LNAME), (STU_NUM, STU_LNAME, STU_INIT), and (STU_LNAME, STU_FNAME, STU_INIT, STU_PHONE). In fact, because STU_NUM alone is a superkey, any composite key that has STU_NUM as a key attribute will also be a superkey.

A candidate key is based on a full functional dependency. For example, STU_NUM would be a candidate key, as would (STU_LNAME, STU_FNAME, STU_INIT, STU_PHONE). On the other hand, (STU_NUM, STU_LNAME) is a superkey, but it is not a candidate key.

A table can have many different candidate keys.

Note that a secondary key does not necessarily yield a unique outcome.

Figure 3.2 - An Example of a Simple Relational Database



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

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

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

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	It is 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

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

- 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

Relational Algebra

- Theoretical way of manipulating table contents using relational operators
- **Relvar**: Variable that holds a relation
  - A relvar has two parts: the heading and the body
  - 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 Set Operators

Select (Restrict) - denoted as σ

- Unary operator that yields a horizontal subset of a table

Project - denoted as π

- Unary operator that yields a vertical subset of a table

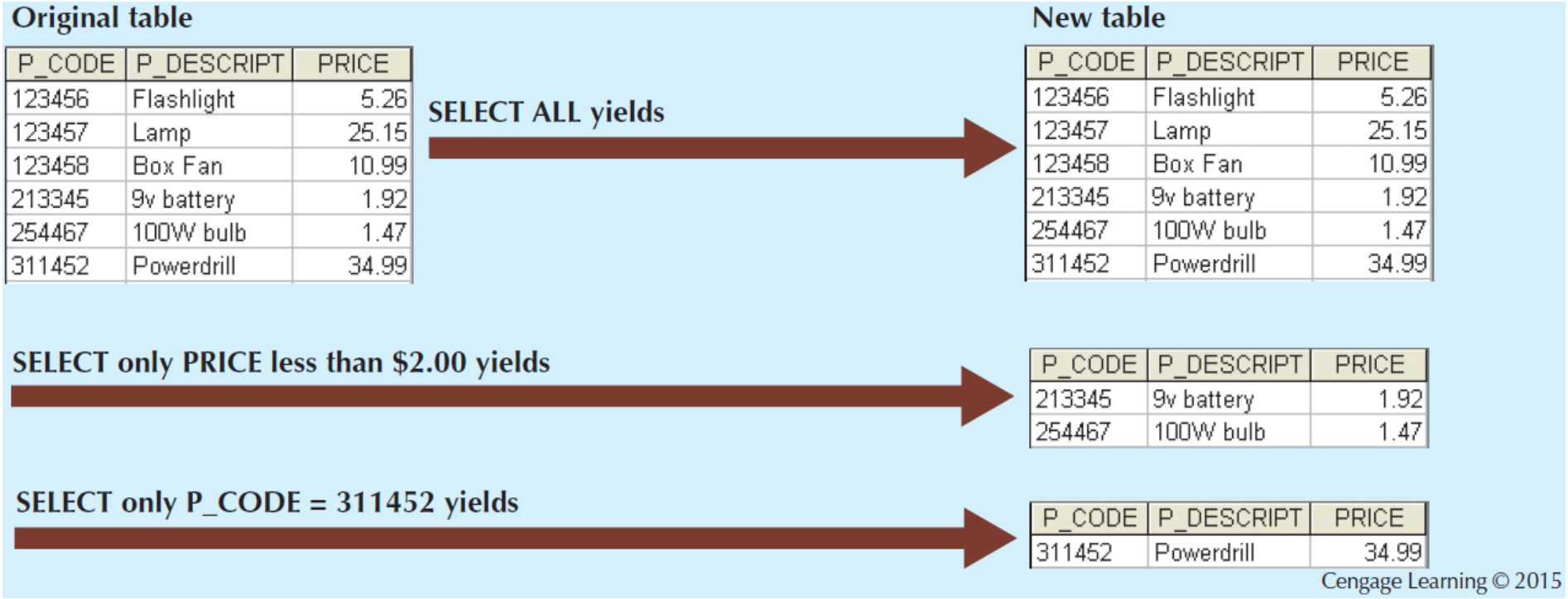
Union – denoted as $\cup$

- 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 – denoted as $\cap$

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

Figure 3.4 - Select

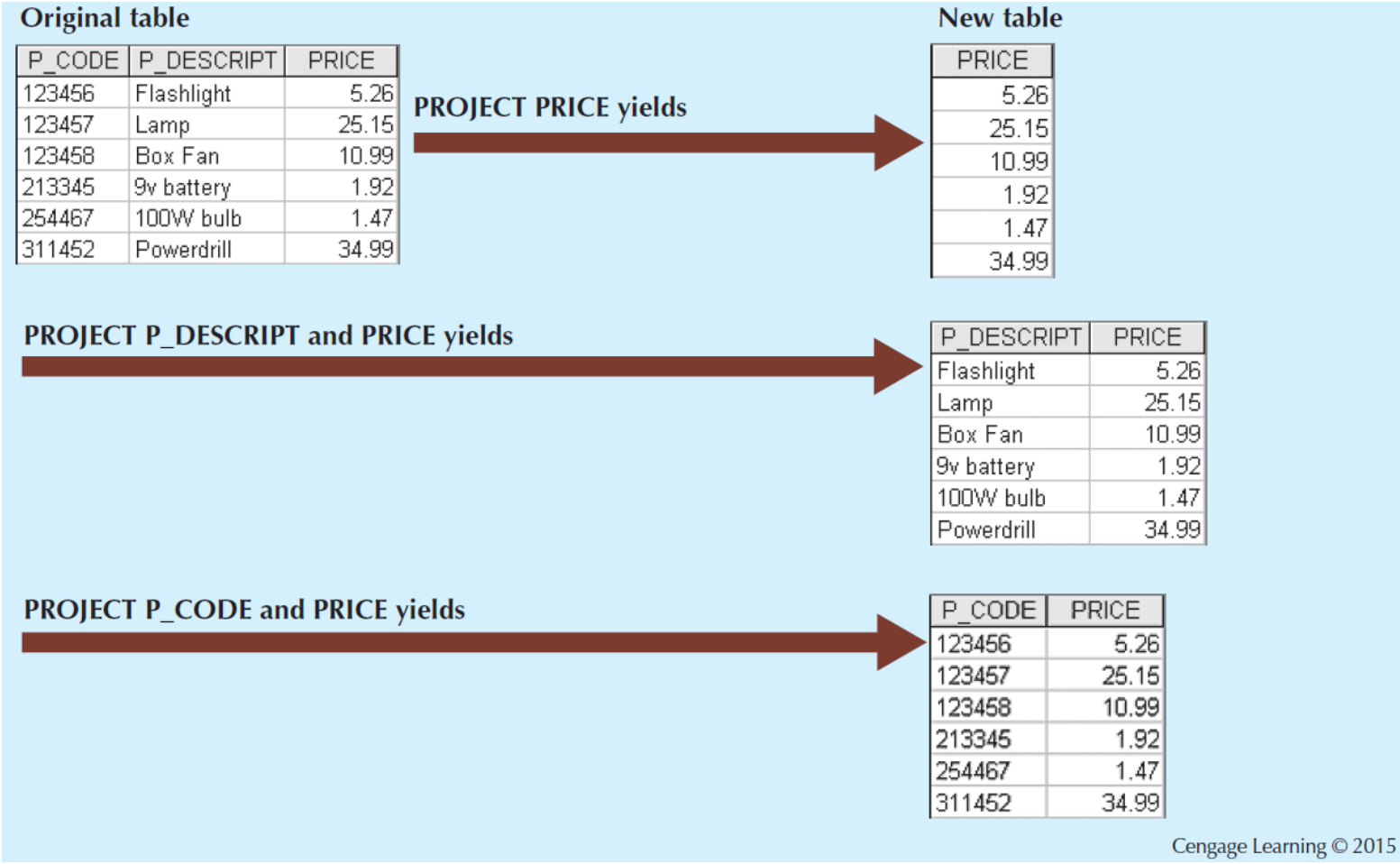


Note

- Sigma is followed by the condition to be evaluated (called a predicate) as a subscript, and then the relation is listed in parentheses. For example, to SELECT all of the rows in the CUSTOMER table that have the value '10010' in the CUS_CODE attribute, you would write the following:

$$\sigma_{cus\_code=10010}(customer)$$

Figure 3.5 - Project



Note

- Pi is followed by the list of attributes to be returned as subscripts, and then the relation listed in parentheses. For example, to PROJECT the CUS_FNAME and CUS_LNAME attributes in the CUSTOMER table, you would write the following:

$$\pi_{cus\_fname,cus\_lname}(customer)$$

Closure Property

- Since relational operators have the property of closure, that is, they accept relations as input and produce relations as output, it is possible to combine operators. For example, you can combine the two previous operators to find the customer first and last name of the customer with customer code 10010:

$$\pi_{cus\_fname,cus\_lname}\left(\sigma_{cus\_code=10010}(customer)\right)$$

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

Note

- It is rather unusual to find two relations that are union-compatible in a database. Typically, PROJECT operators are applied to relations to produce results that are union-compatible. For example, assume the SUPPLIER and VENDOR tables are not union-compatible. If you wish to produce a listing of all vendor and supplier names, then you can PROJECT the names from each table and then perform a UNION with them.

$$\pi_{supplier\_name}(supplier) \cup \pi_{vendor\_name}(vendor)$$

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

Note

- Just as with the UNION operator, it is unusual to find two relations that are union-compatible in a database so PROJECT operators are applied to relations to produce results that can be manipulated with an INTERSECT operator.
- For example, again assume the SUPPLIER and VENDOR tables are not union-compatible. If you wish to produce a listing of any vendor and supplier names that are the same in both tables, then you can PROJECT the names from each table and then perform an INTERSECT with them.

$$\pi_{supplier\_name}(supplier) \cap \pi_{vendor\_name}(vendor)$$

Relational Set Operators

- **Difference – denoted as "-"**
  - Yields all rows in one table that are not found in the other table
  - Tables must be union-compatible to yield valid results
- **Product – denoted as "x"**
  - Yields all possible pairs of rows from two tables

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
- **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

Types of Joins

- **Natural join**: Links tables by selecting only the rows with common values in their common attributes
  - normally just referred to as JOIN in formal treatments.
  - JOIN is denoted by the symbol ⋈
  - **Join columns**: Common columns
  - JOIN is not a fundamental relational algebra operator. It can be derived from other operators
- **Equijoin**: Links tables on the basis of an equality condition that compares specified columns of each table
  - The outcome of the equijoin does not eliminate duplicate columns, and the condition or criterion used to join the tables must be explicitly defined.
- **Theta join**: Extension of natural join, denoted by adding a theta subscript after the JOIN symbol
  - Theta join is denoted by adding a theta subscript after the JOIN symbol: $\bowtie_{\theta}$. Equijoin is then a special type of theta join.

Natural Join

- A natural join links tables by selecting only the rows with common values in their common attribute(s). A natural join is the result of a three-stage process:
- Given CUSTOMER and AGENT tables

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

Step 1

- First, a PRODUCT of the tables is created

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

Step 2

- Second, a SELECT is performed on the output of Step 1 to yield only the rows for which the AGENT_CODE values are equal. The common columns are referred to as the join columns.

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

Step 3

- A PROJECT is performed on the results of Step 2 to yield a single copy of each attribute, thereby eliminating duplicate columns.

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

Equi-join: example

Employees

Employee	Project
Smith	A
Black	A
Black	B

Projects

Code	Name
A	Venus
B	Mars

Employes $\bowtie_{\text{Project=Code}}$ Projects

Employee	Project	Code	Name
Smith	A	A	Venus
Black	A	A	Venus
Black	B	B	Mars

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

Figure 3.16 - Divide

Table 1

CODE	LOC
A	5
A	9
A	4
B	5
B	3
C	6
D	7
D	8
E	8

DIVIDE

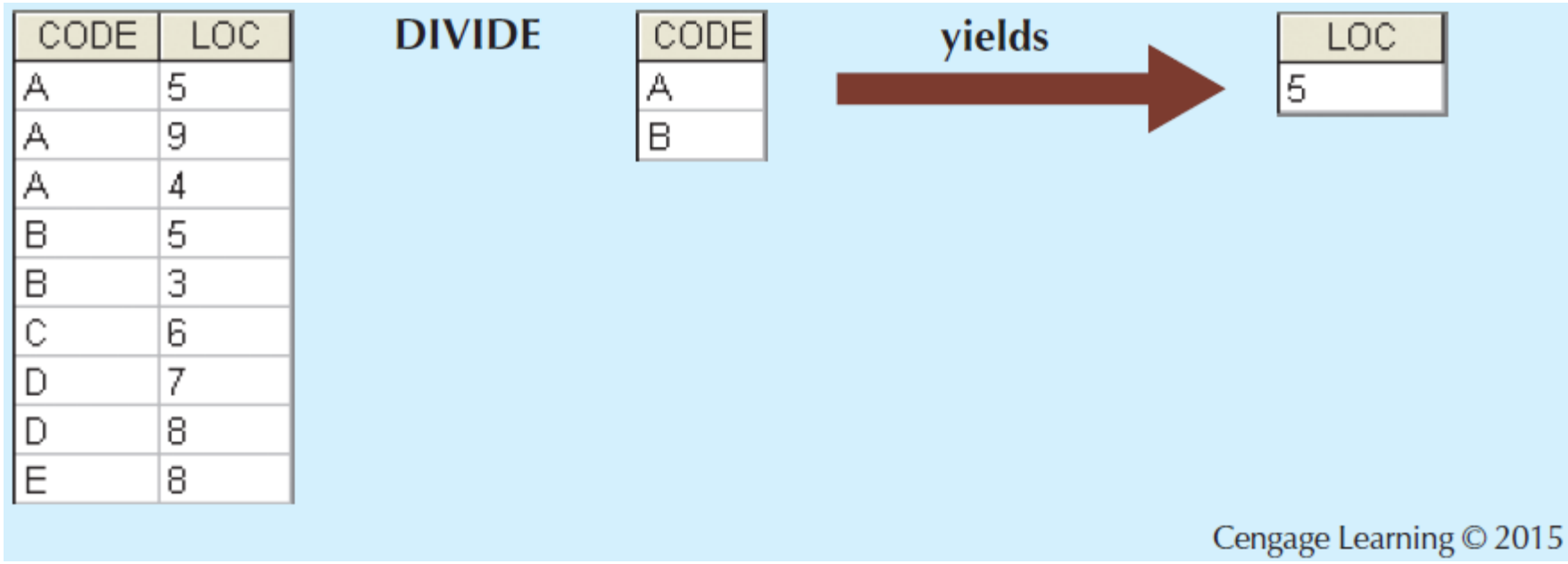
Table 2

CODE
A
B

yields

Table 3

LOC
5



1. Tables 1 and 2 both contain the CODE column but do not share the LOC column.
2. To be included in the resulting Table 3, a value in the unshared column (LOC) must be associated with every value in Table 2.
3. The only value associated with both A and B is 5

Data Dictionary and the System Catalog

- **Data dictionary**: Description of all tables in the database created by the user and designer
  - Sometimes described as "the database designer's database" because it records the design decisions about tables and their structures.
- **System catalog**: System data dictionary that describes all objects within the database
- Homonyms and synonyms must be avoided to lessen confusion
  - **Homonym**: Same name is used to label different attributes (F_NAME is father's name in STUDENT table, but is first name in CUSTOMER table)
  - **Synonym**: Different names are used to describe the same attribute (a synonym is an alias or alternative name for a database object)

A Sample Data Dictionary

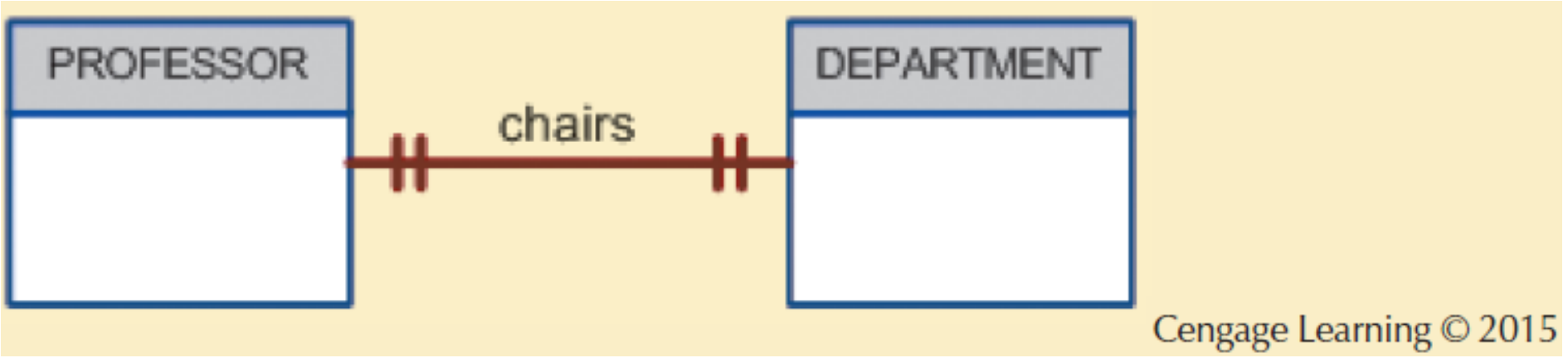
TABLE NAME	ATTRIBUTE NAME	CONTENTS	TYPE	FORMAT	RANGE	REQUIRED	PK or FK	FK REFERENCED TABLE
CUSTOMER	CUS_CODE	Customer account code	CHAR(5)	99999	10000–99999	Y	PK	
	CUS_LNAME	Customer last name	VARCHAR(20)	Xxxxxxx		Y		
	CUS_FNAME	Customer first name	VARCHAR(20)	Xxxxxxx		Y		
	CUS_INITIAL	Customer initial	CHAR(1)	X				
	CUS_RENEW_DATE	Customer insurance renewal date	DATE	dd-mmm-yyyy				
	AGENT_CODE	Agent code	CHAR(3)	999			FK	AGENT
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)	Xxxxxxx		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) is used to specify numbers with up to nine digits, including two digits to the right of the decimal place. Some RDBMS permit the use of a MONEY or CURRENCY data type.

Relationships within the Relational Database

- 1:M relationship - Norm for relational databases
- 1:1 relationship - One entity can be related to only one other entity and vice versa
- Many-to-many (M:N) relationship - Implemented by creating a new entity in 1:M relationships with the original entities
  - **Composite entity** (**Bridge** or **associative entity**): Helps avoid problems inherent to M:N relationships, includes the primary keys of tables to be linked

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



The Implemented 1:1 Relationship

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

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

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

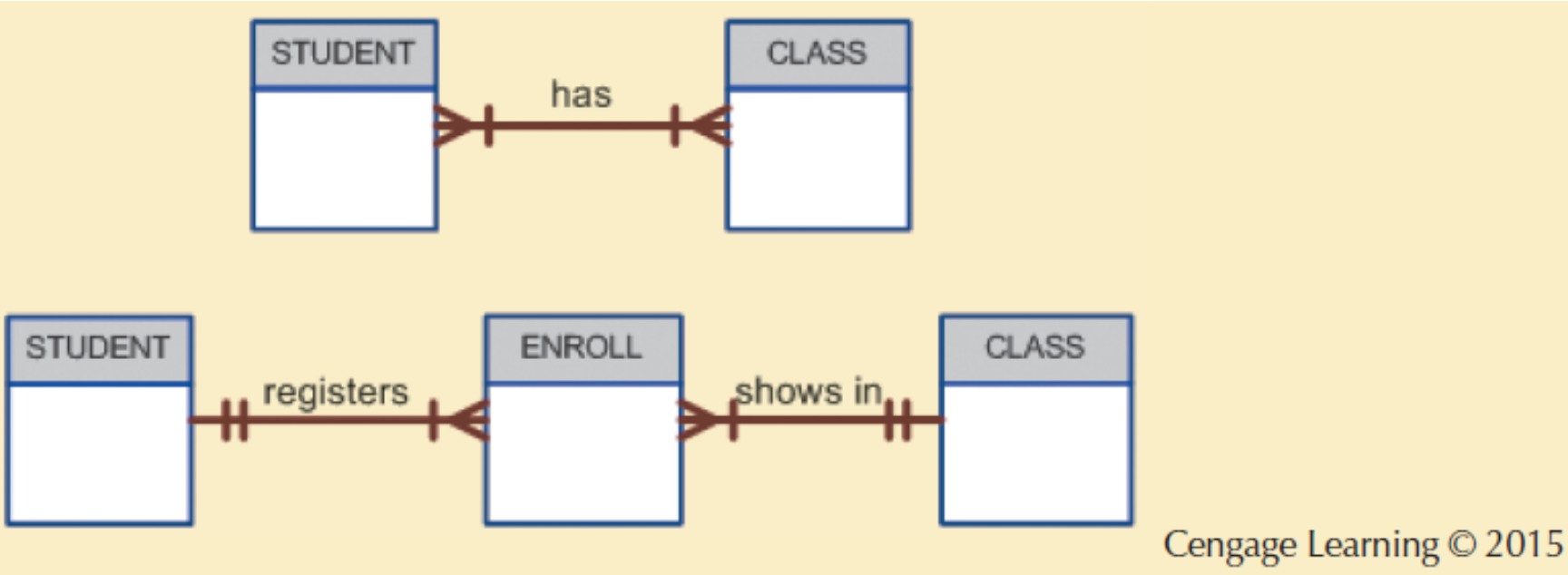
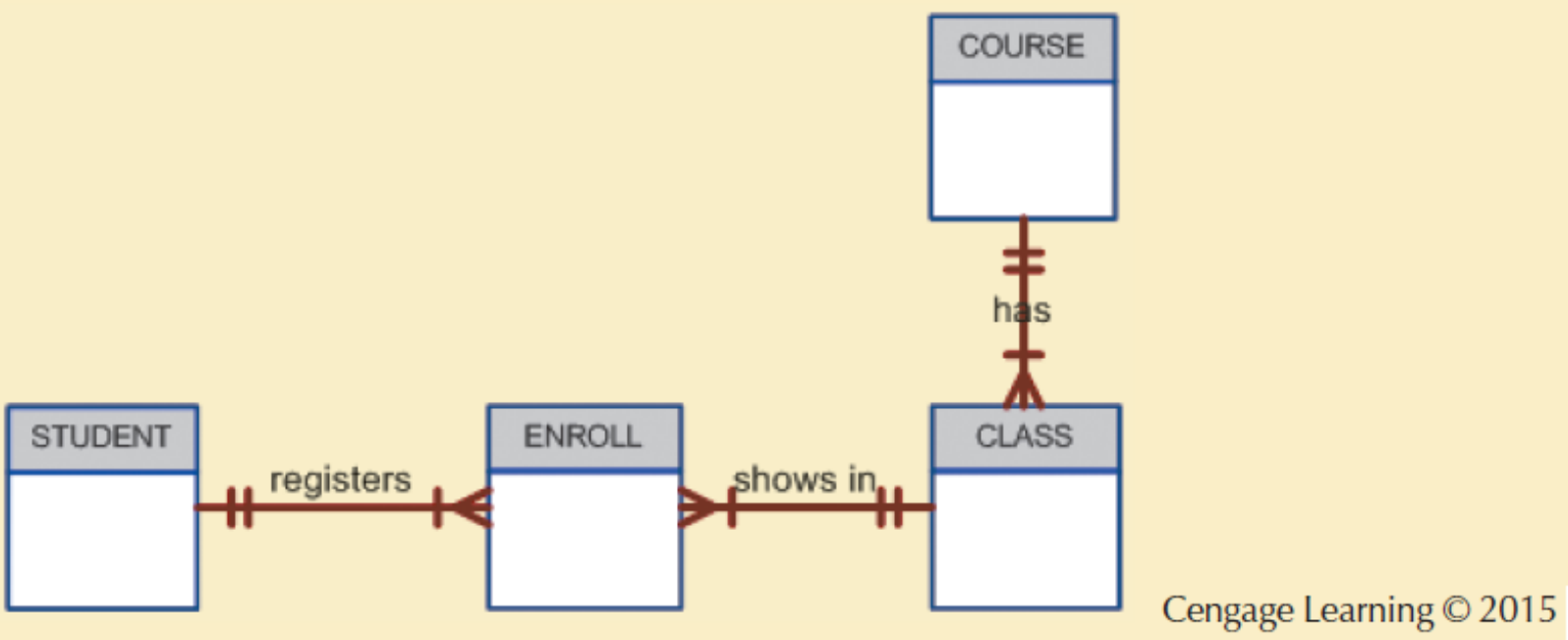


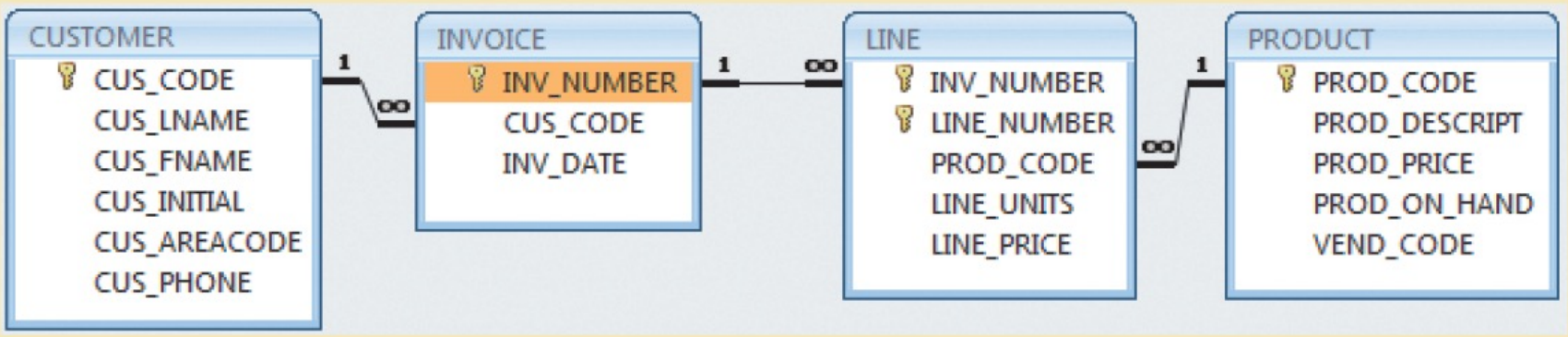
Figure 3.27 - The Expanded ER Model



Data Redundancy

- Relational database facilitates control of data redundancies through use of foreign keys
- To be controlled except the following circumstances
  - 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 - The Relational Diagram for the Invoicing System



Cengage Learning © 2015

Index

- Orderly arrangement to logically access rows in a table
- **Index key**: Index's reference point that leads to data location identified by the key
- **Unique index**: Index key can have only one pointer value associated with it
- Each index is associated with only one table