CS 309A- Database Management Systems

Introduction to relational model

Basics of The Relational Model

- Foundation: **relation** (also called **table**)
  - is a matrix composed of intersection rows and columns.
  - Each row in a relation is called a **tuple**.
  - Each column represents an **attribute**.

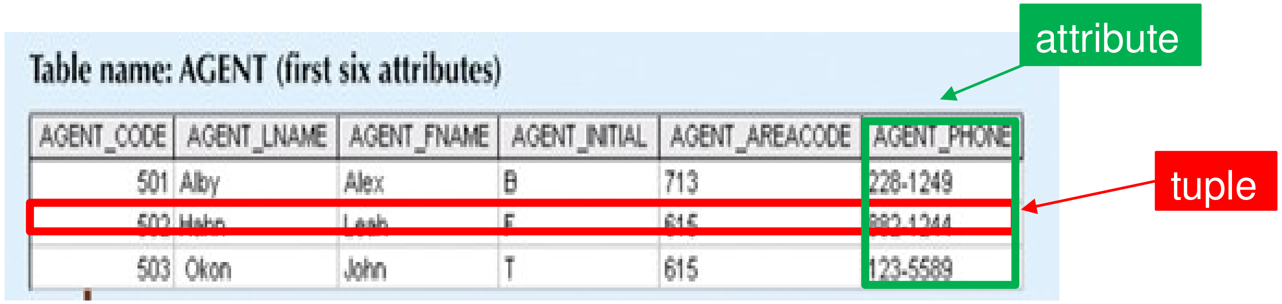
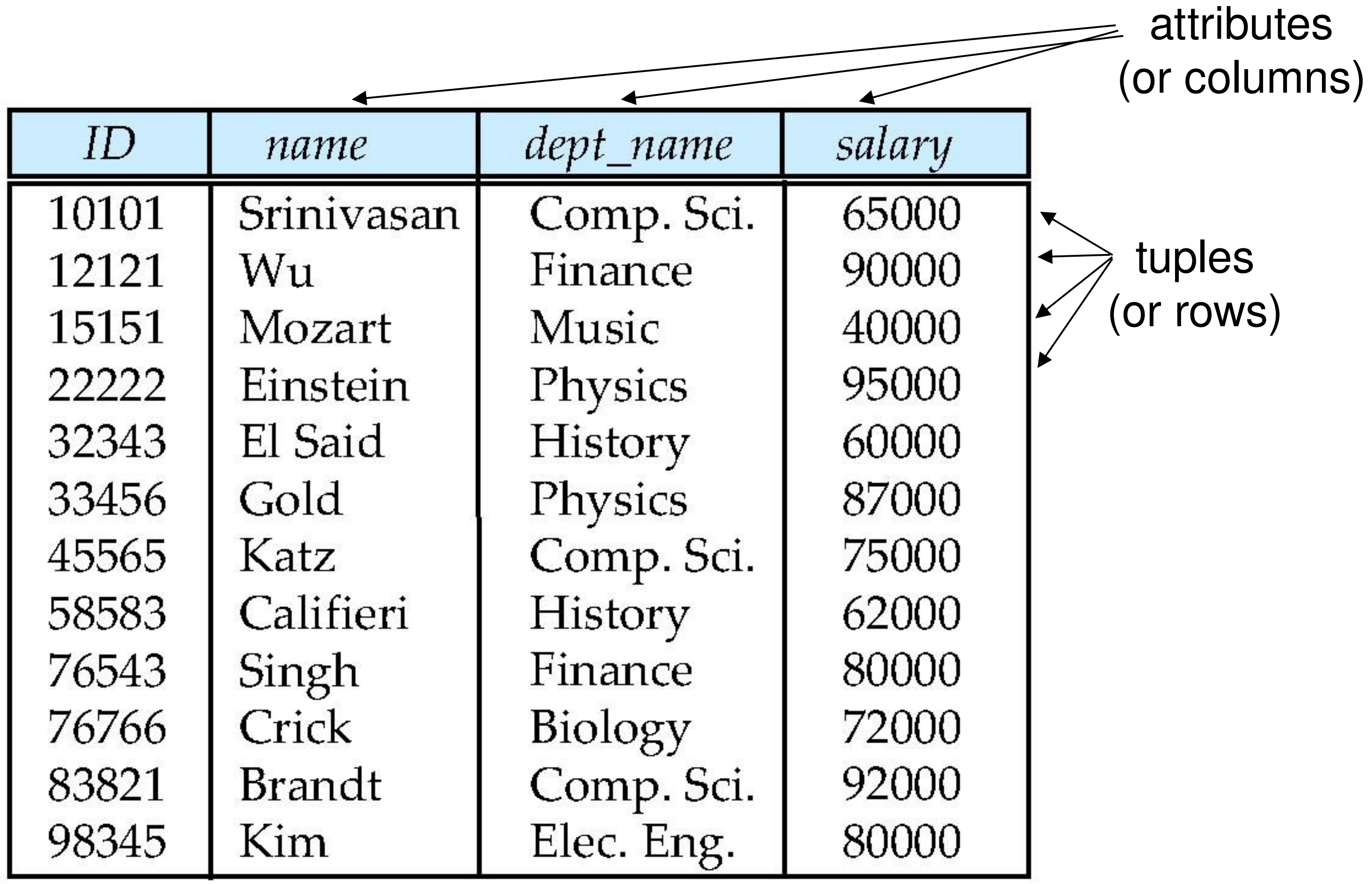

attribute
tuple

Table name: AGENT (first six attributes)

AGENT_CODE	AGENT_LNAME	AGENT_FNAME	AGENT_INITIAL	AGENT_AREACODE	AGENT_PHONE
501	Alby	Alex	B	713	228-1249
502	Hahn	Leah	F	615	882-1244
503	Okon	John	T	615	123-5589

- The relational data model is implemented through **relational database management system (RDBMS)**.
  - The most important advantage of the RDBMS is to *hide the complexities* of the relational model from the user.

Another example of a Relation


attributes
(or columns)
tuples
(or rows)

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

Attribute Types

- The set of allowed values for each attribute is called the **domain** of the attribute
- Attribute values are (normally) required to be **atomic**; that is, indivisible
- The special value ***null*** is a member of every domain. Indicated that the value is “unknown”
- The null value causes complications in the definition of many operations

Relation Schema and Instance

- $A_1, A_2, \ldots, A_n$ are *attributes*
- $R = (A_1, A_2, \ldots, A_n)$ is a *relation schema*

  Example: *instructor* = (*ID*, *name*, *dept_name*, *salary*)
- Formally, given sets $D_1, D_2, \ldots. D_n$, a **relation *r*** is a subset of $D_1 \times D_2 \times \ldots \times D_n$. Thus, a relation is a set of *n*-tuples $(a_1, a_2, \ldots, a_n)$ where each $a_i \in D_i$
- The current values (**relation instance**) of a relation are specified by a table
- An element ***t*** of ***r*** is a *tuple*, represented by a *row* in a table

Relations are Unordered

- Order of tuples is irrelevant (tuples may be stored in an arbitrary order)
- Example: *instructor* relation with unordered tuples

ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

keys

- In the relational model, keys are used to:
  - Ensure that each row in a table is uniquely identifiable
  - Establish relationships among tables
  - Ensure the integrity of the data
- A **key** consists of *one or more* attributes that <u>determines</u> other attributes.

Some concepts

- **Determination**: a state that knowing the value of one attribute makes it possible to know the value of another.

  *total_price = unit_price * count*
- **Functional dependence**: the value of one or more attributes determines the value of one or more other attributes.

  STU_NUM ✉ STU_LNAME

  determinant (**key**) **dependent**

STU_NUM ✉ (STU_LNAME, STU_FNAME, STU_GPA)
(STU_FNAME, STU_LNAME, STU_INIT, STU_PHONE) ✉ (STU_DOB, STU_HRS, STU_GPA)

Some concepts

- **Full functional dependence**: the entire collection of attributes in the determinant is necessary for the relationship.

1) STU_NUM ✉ STU_GPA
2) (STU_NUM, STU_LNAME) ✉ STU_GPA

Which one is *full* functional dependence?

Types of Keys

- **Composite key**: composed of more than one attribute

  STU_NUM -> GPA *(Is not a composite key)*

  (STU_LNAME, STU_FNAME, STU_INIT, STU_PHONE) ✉ STU_HRS *(Is a composite key)*

- **Superkey:** can uniquely identify any row in the table

  a superkey functionally determines *every attribute* in the row

  STU_NUM, (STU_NUM, STU_LNAME), (STU_LNAME, STU_FNAME, STU_INIT)

- **Candidate key**: a minimal superkey; a superkey without any unnecessary attributes

  In the above superkeys, which are candidate keys?

  A table can have many different candidate keys.

Types of Keys: primary key

- The **primary key** is a candidate key chosen to ensure *entity integrity*.

- **Entity integrity**: each row in the table has its own unique identity
  - To ensure entity integrity, the primary key has two requirements:
    1. All of the values must be unique
    2. No key attribute can contain a **null**

The absence of any data value. Nulls should be avoided in the database.

- Another role that the primary key plays is to build relationships between tables.

The relational model uses *common attributes* to link tables.

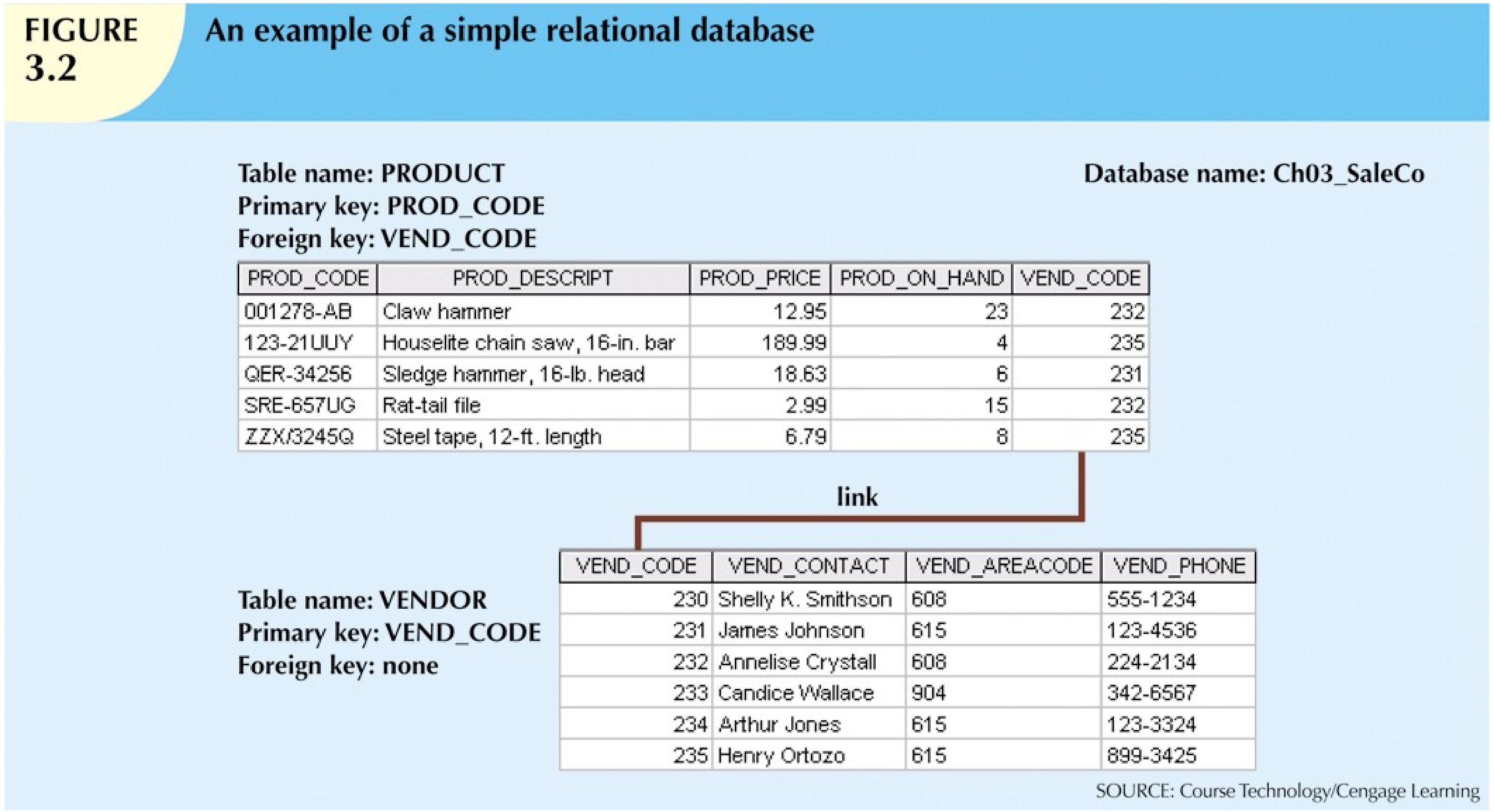


FIGURE 3.2 An example of a simple relational database

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

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

SOURCE: Course Technology/Cengage Learning

Types of Keys: foreign key

- A **Foreign key** is the primary key of one table that has been placed into another table to create a common attribute, which are used to ensure *referential integrity*.

- **Referential integrity**: every reference to an entity occurrence (instance) by another entity occurrence (instance) is valid.

Every foreign key entry must either be *null* or a *valid value* in the primary key of the related table.

Types of Keys

- **Secondary key**: a key used for data retrieval purpose.
  - May not yield a unique identify

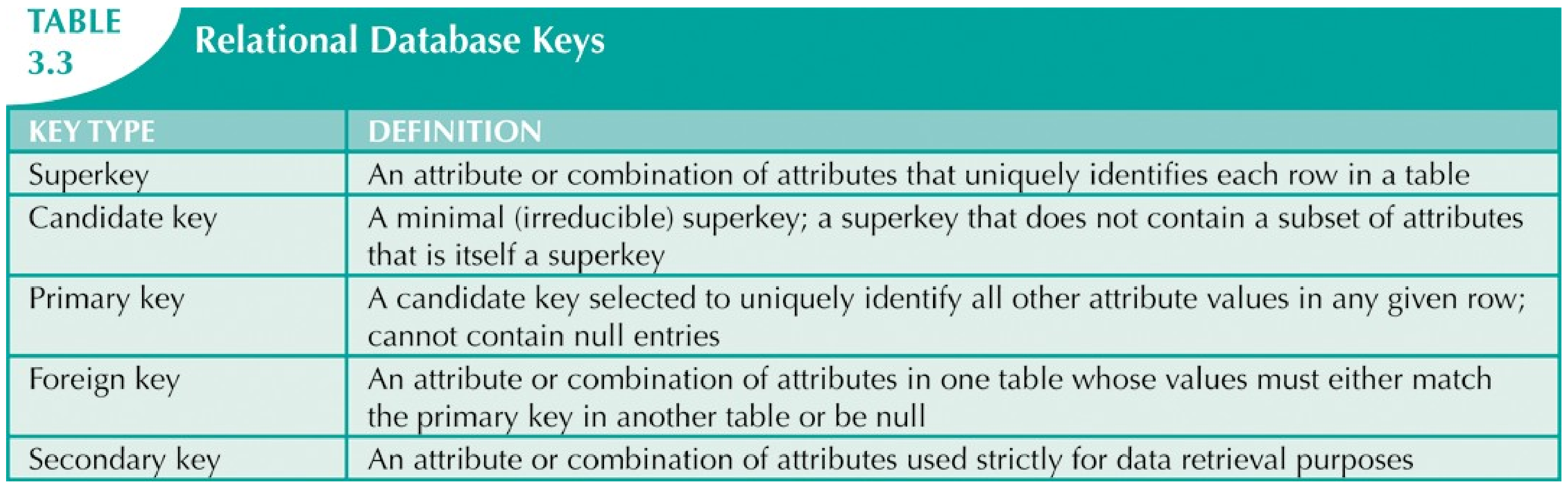


TABLE 3.3 Relational Database Keys

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

- Very important to good database design
- RDBMS enforce integrity rules automatically
- Safer to ensure that application design conforms to the *entity* and *referential rules*

Integrity Rules

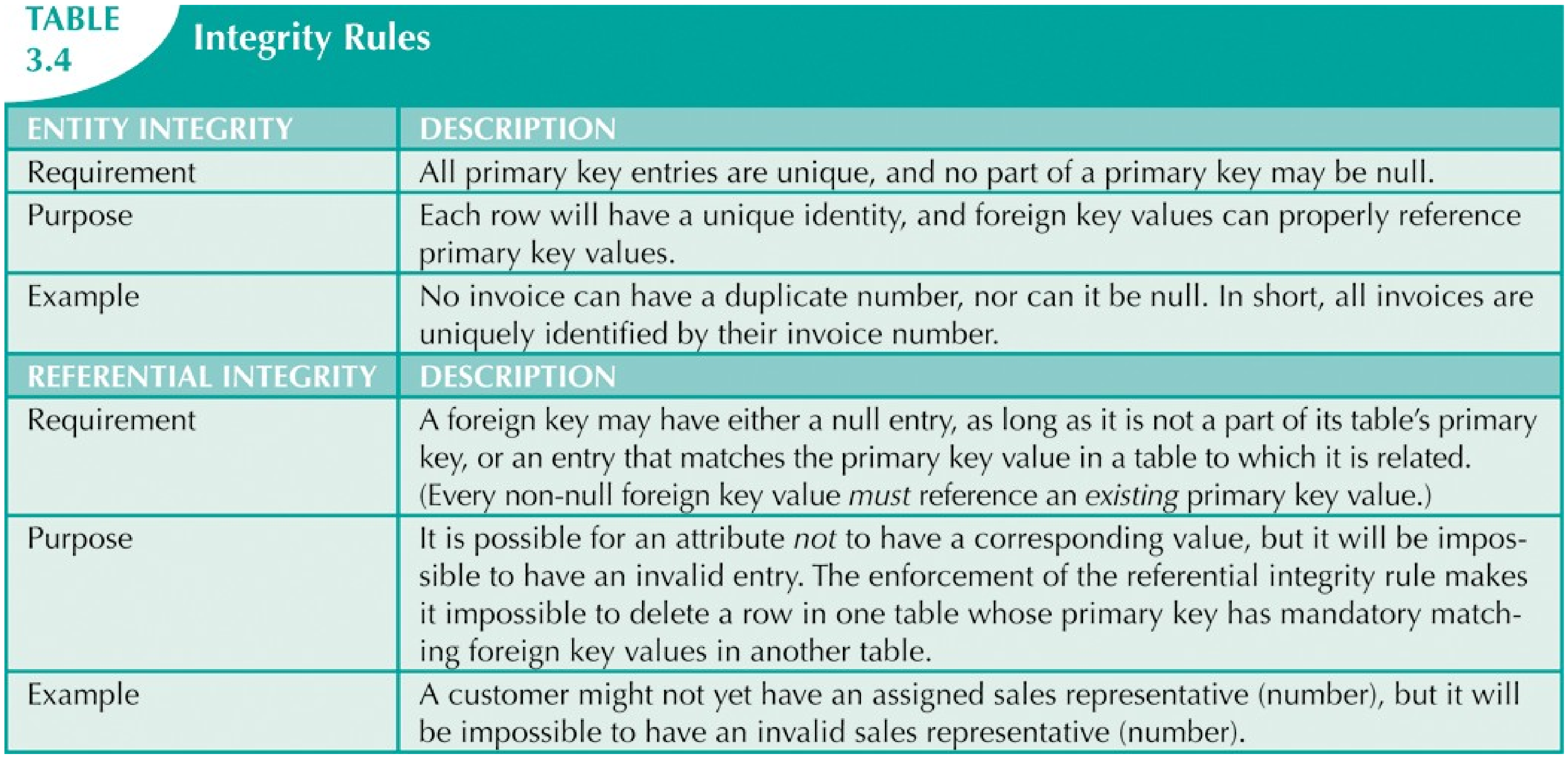
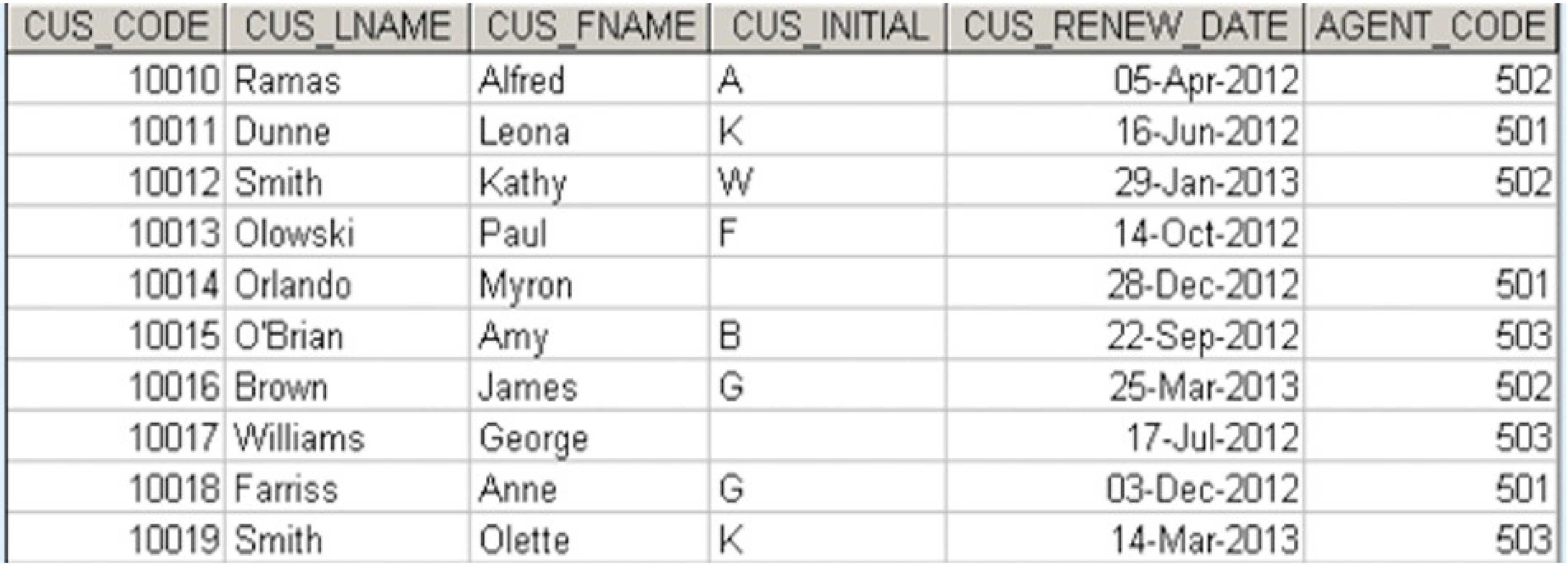


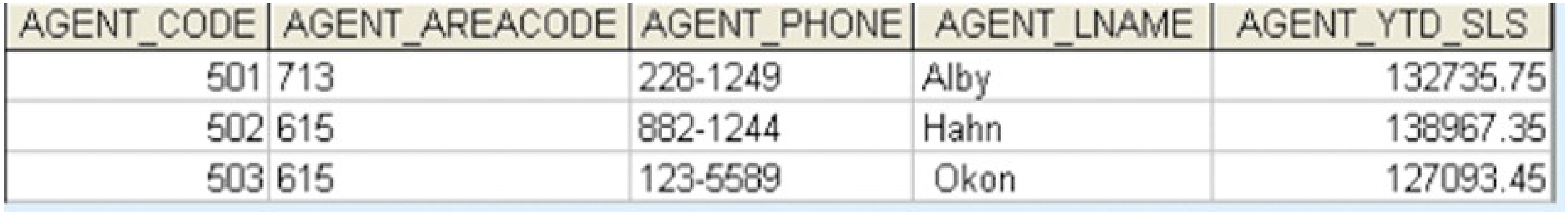
TABLE 3.4 Integrity Rules

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).

Customer Table and Agent table



CUS_CODE	CUS_LNAME	CUS_FNAME	CUS_INITIAL	CUS_RENEW_DATE	AGENT_CODE
10010	Ramas	Alfred	A	05-Apr-2012	502
10011	Dunne	Leona	K	16-Jun-2012	501
10012	Smith	Kathy	W	29-Jan-2013	502
10013	Olowski	Paul	F	14-Oct-2012	
10014	Orlando	Myron		28-Dec-2012	501
10015	O'Brian	Amy	B	22-Sep-2012	503
10016	Brown	James	G	25-Mar-2013	502
10017	Williams	George		17-Jul-2012	503
10018	Farriss	Anne	G	03-Dec-2012	501
10019	Smith	Olette	K	14-Mar-2013	503



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

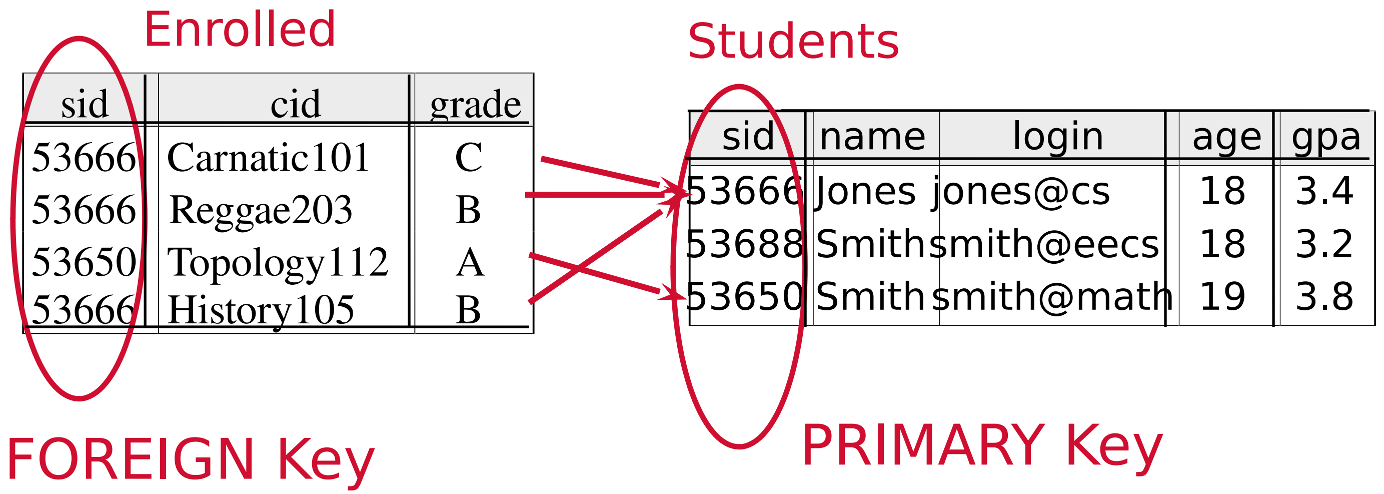
Answer the following questions:

- What is the **primary key** in the CUSTOMER table? How does it ensure the **entity integrity**?

- What is the **foreign key** in the CUSTOMER table? How does it ensure the **referential integrity**?

Another Example

- Consider the following two tables : Enrolled and Students
- What is the **primary key** in the Students table? How does it ensure the **entity integrity**?
- What is the **foreign key** in the Enrolled table? How does it ensure the **referential integrity**?



sid	cid	grade
53666	Carnatic101	C
53666	Reggae203	B
53650	Topology112	A
53666	History105	B

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

Relational Query Languages

- Procedural vs .non-procedural, or declarative
- “Pure” languages:
  - Relational algebra
  - Tuple relational calculus
  - Domain relational calculus
- The above 3 pure languages are equivalent in computing power
- We will concentrate on relational algebra
  - Not turning-machine equivalent

Relational Algebra

- The data in relational tables has limited value.
- However, we can use **relational algebra** to manipulate data to get more useful information.
- The relational operators have the property of **closure**, that is, the use of relational algebra operators on existing relations (tables) produces new relations (tables).

Example Instances

R1

sid	bid	day
22	101	10/10/96
58	103	11/12/96

Boats

bid	bname	color
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

S1

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

S2

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

Selection (σ)

- Selects rows that satisfy *selection condition*.
- Gives a subset of the tuples that match a specified criterion.
- Result is a relation.

~~sid~~	sname	rating	~~age~~
~~28~~	yuppy	9	~~35.0~~
~~31~~	~~lubber~~	~~8~~	~~55.5~~
~~44~~	~~guppy~~	~~5~~	~~35.0~~
~~58~~	rusty	10	~~35.0~~

$$\sigma_{rating>8}(S2)$$

sname	rating
yuppy	9
rusty	10

$$\pi_{sname,rating}(\sigma_{rating>8}(S2))$$

Examples on SELECT: σ

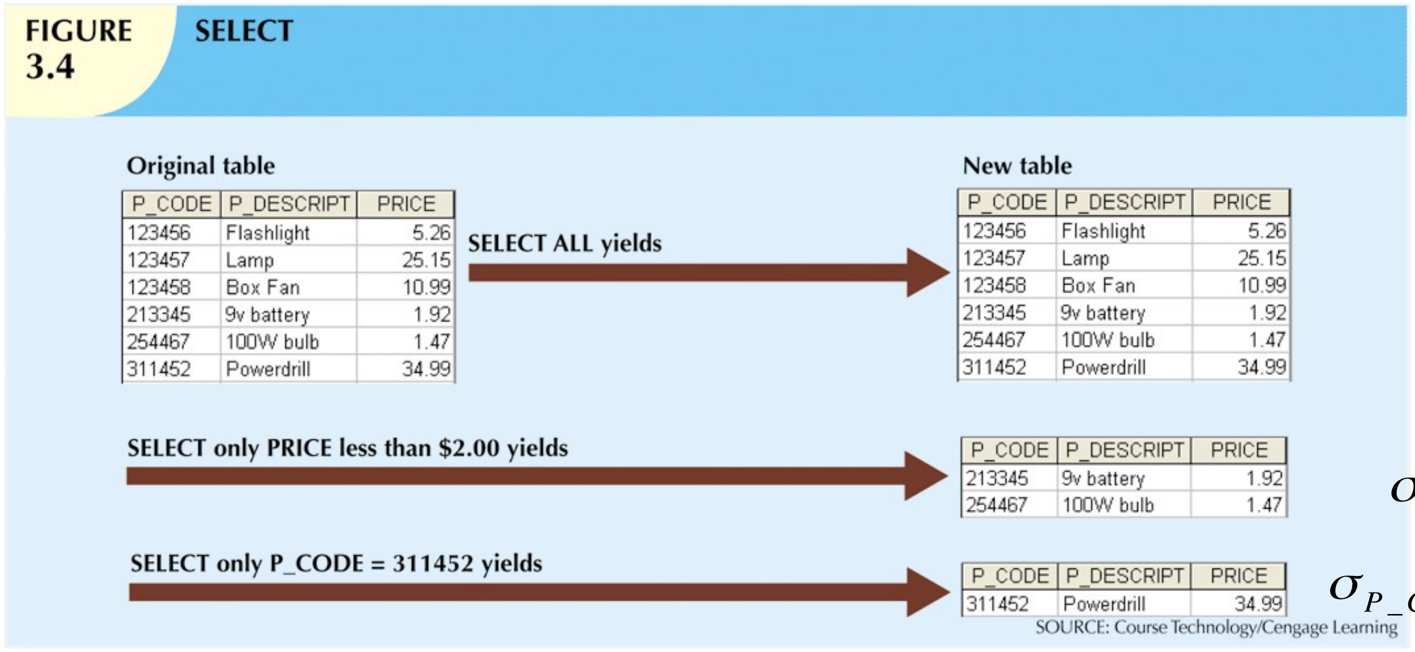


FIGURE 3.4 SELECT

Original table

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

SELECT ALL yields

New table

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

$\sigma(product)$

SELECT only PRICE less than $2.00 yields

P_CODE	P_DESCRIPT	PRICE
213345	9v battery	1.92
254467	100W bulb	1.47

$\sigma_{PRICE<2.00}(product)$

SELECT only P_CODE = 311452 yields

P_CODE	P_DESCRIPT	PRICE
311452	Powerdrill	34.99

$\sigma_{P\_CODE=311452}(product)$

SOURCE: Course Technology/Cengage Learning

Projection (π)

Gives a subset of the attributes that match a specified criterion.

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

S2

sname	rating
yuppy	9
lubber	8
guppy	5
rusty	10

$\pi_{sname,rating}(S2)$

age
35.0
55.5

$\pi_{age}(S2)$

Examples on PROJECT

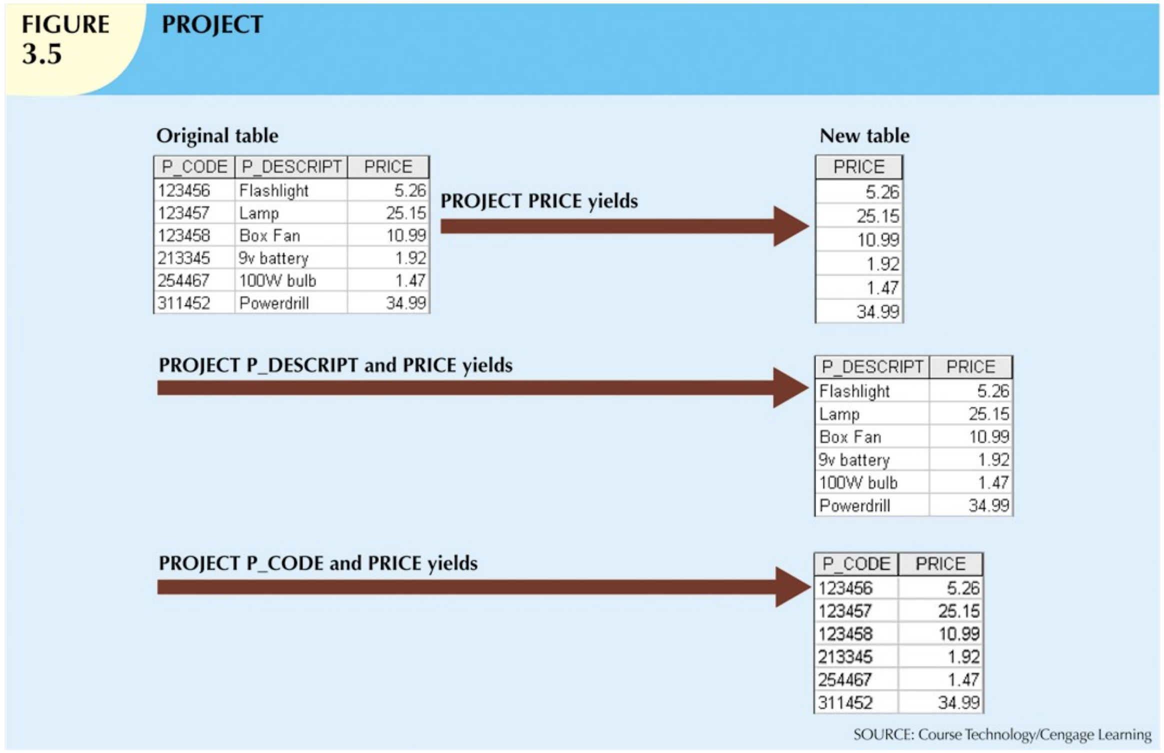


FIGURE 3.5 PROJECT

Original table

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

PROJECT PRICE yields

New table

PRICE
5.26
25.15
10.99
1.92
1.47
34.99

$$\pi_{PRICE}(product)$$

PROJECT P_DESCRIPT and PRICE yields

P_DESCRIPT	PRICE
Flashlight	5.26
Lamp	25.15
Box Fan	10.99
9v battery	1.92
100W bulb	1.47
Powerdrill	34.99

$$\pi_{P\_DESCRIPT,PRICE}(product)$$

PROJECT P_CODE and PRICE yields

P_CODE	PRICE
123456	5.26
123457	25.15
123458	10.99
213345	1.92
254467	1.47
311452	34.99

$$\pi_{P\_CODE,PRICE}(product)$$

SOURCE: Course Technology/Cengage Learning

Union and Set-Difference

- Both of these operations take two input relations that are **union compatible**. Two relations are union compatible if
  - they have the same number of attributes
  - the domain of each attribute in column order is the same in both R and S. `Corresponding' attributes have the same type.

UNION Operation

Consider two relations R and S:

- *UNION of R and S*
  the **union** of two relations is a relation that includes all the tuples that are either in R or in S or in both R and S. Duplicate tuples are eliminated.

Union

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

S1

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
4[illegible]	[illegible]y	5	35.0
[illegible]	[illegible]	10	35.0

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0
44	guppy	5	35.0
28	yuppy	9	35.0

$S1 \cup S2$

SET *DIFFERENCE* Operation

Consider two relations R and S:

- *DIFFERENCE of R and S*
  the **difference** of R and S is the relation that contains all the tuples that are in R but that are not in S.
- Yields all rows in one table that are not in the other table.
- The tables must be *union-compatible*.

Set Difference

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

S1

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

S2

sid	sname	rating	age
22	dustin	7	45.0

*S*1- *S*2

sid	sname	rating	age
28	yuppy	9	35.0
44	guppy	5	35.0

S2 – S1

Cross-Product ((Cartesian Product))

- S1 $\times$ R1: Each row of S1 paired with each row of R1.
- Q: How many rows in the result?
- *Result schema* has one attribute per attribute of S1 and R1, with attribute names `inherited' if possible.
  - *May have a naming conflict*: Both S1 and R1 have a field with the same name.
  - In this case, can use the *renaming operator*:

$$\rho\ (C(1 \rightarrow sid1, 5 \rightarrow sid2),\ S1 \times R1)$$

Cross Product Example

sid	bid	day
22	101	10/10/96
58	103	11/12/96

R1

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

S1

S1 x R1 =

(sid)	sname	rating	age	(sid)	bid	day
22	dustin	7	45.0	22	101	10/10/96
22	dustin	7	45.0	58	103	11/12/96
31	lubber	8	55.5	22	101	10/10/96
31	lubber	8	55.5	58	103	11/12/96
58	rusty	10	35.0	22	101	10/10/96
58	rusty	10	35.0	58	103	11/12/96

Cross Product (Cartesian Product): ×

- Yields all possible pairs of rows from two tables.

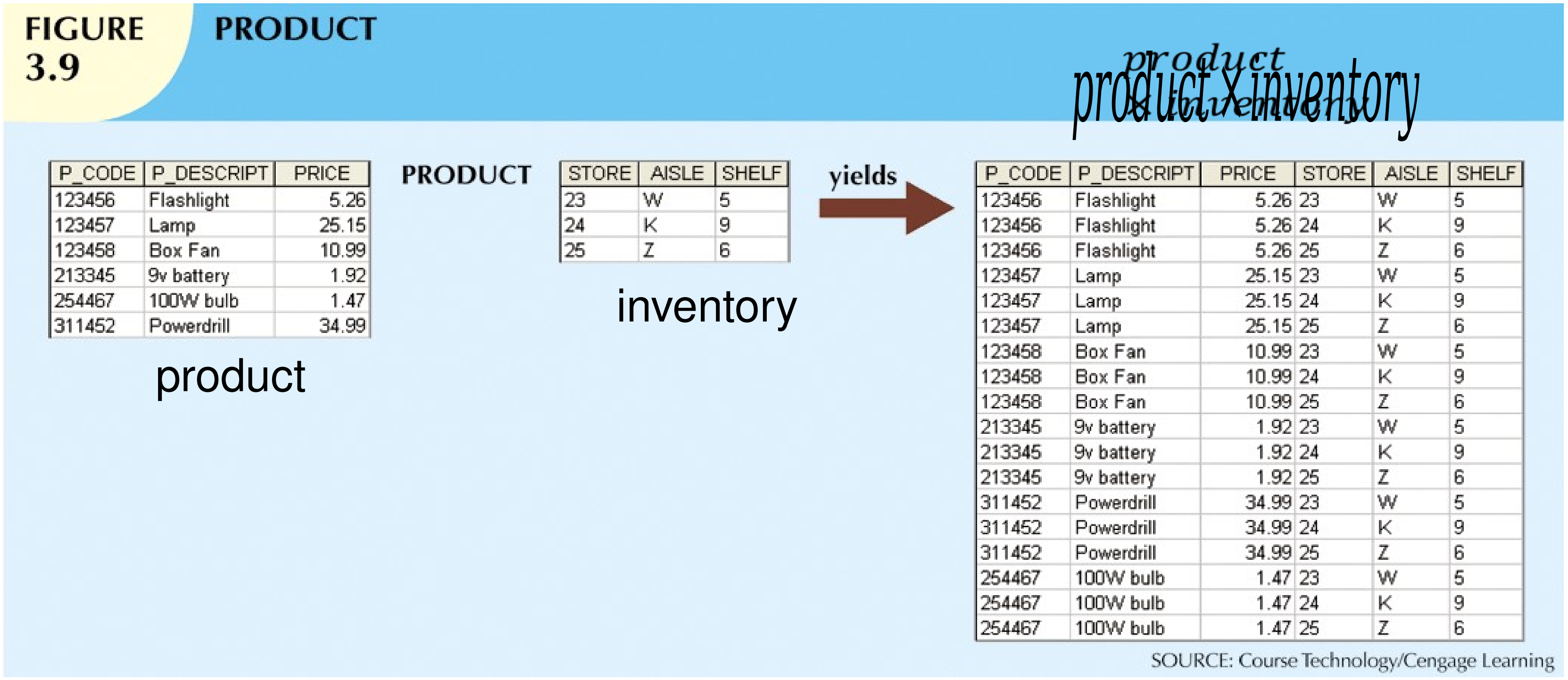


FIGURE 3.9 PRODUCT

product x inventory

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

PRODUCT

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

inventory

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

SOURCE: Course Technology/Cengage Learning

Relational Algebra: 5 Basic Operations

- *Selection* (σ) Selects a subset of *rows* from relation (horizontal).
- *Projection* (π) Retains only wanted *columns* from relation (vertical).
- *Cross-product* ($\times$) Allows us to combine two relations.
- *Set-difference* (—) Tuples in r1, but not in r2.
- *Union* ($\cup$) Tuples in r1 or in r2.

Since each operation returns a relation, operations can be *composed!* (Algebra is “closed”.)

Compound Operator: Intersection

- In addition to the 5 basic operators, there are several additional "Compound Operators"
  - These add no computational power to the language, but are useful shorthands.
  - Can be expressed solely with the basic ops.

- Intersection takes two input relations, which must be *union-compatible*.
- Q: How to express it using basic operators?

$$R \cap S = R - (R - S)$$

Intersection

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

S1

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

S2

sid	sname	rating	age
31	lubber	8	55.5
58	rusty	10	35.0

$S1 \cap S2$

JOIN: ⋈

- Combines information from two or more tables.
- Is the real ***power*** behind the relational database.
- Its simplest form is *cross product* of the two relations.
- As the join becomes more complex, tuples are removed within the cross product to make the result of the join more meaningful.
- JOIN allows you to evaluate **a join condition** between the attributes of the relations on which the join is undertaken.

Compound Operator: Join $\bowtie$

- Joins are compound operators involving cross product, selection, and (sometimes) projection.
- Most common type of join is a "*natural join*" (often just called "join").
- A **natural join** links tables by selecting only the rows with common values in their common attributes.

JOIN: natural join

CUSTOMER ⋈ AGENT

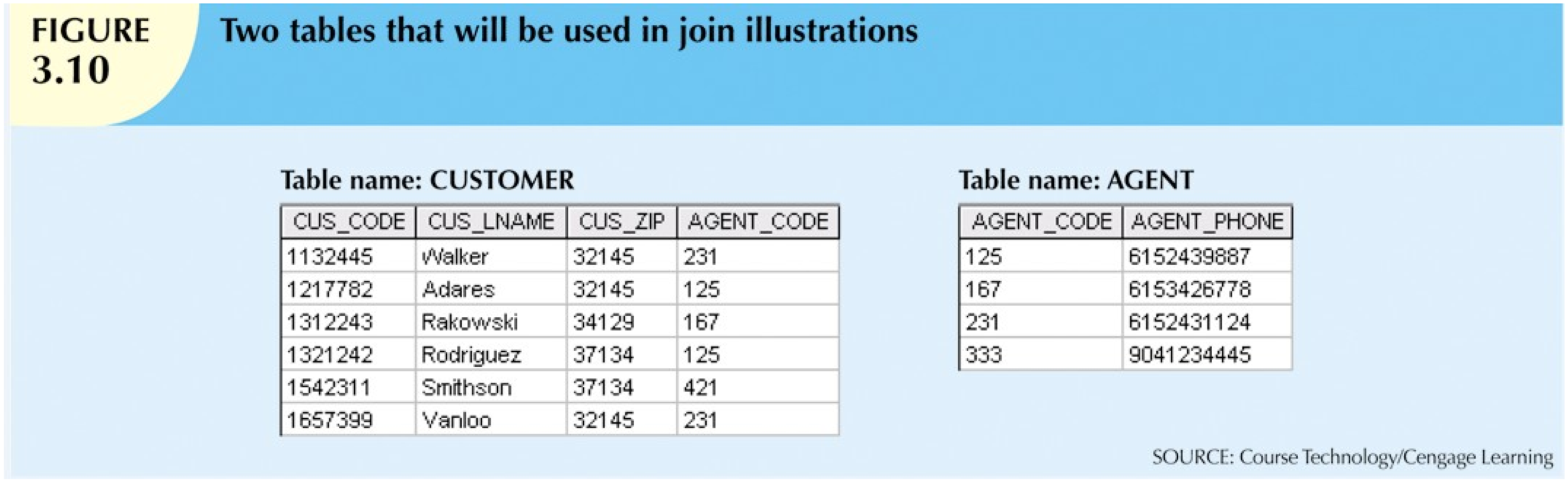


FIGURE 3.10 Two tables that will be used in join illustrations

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

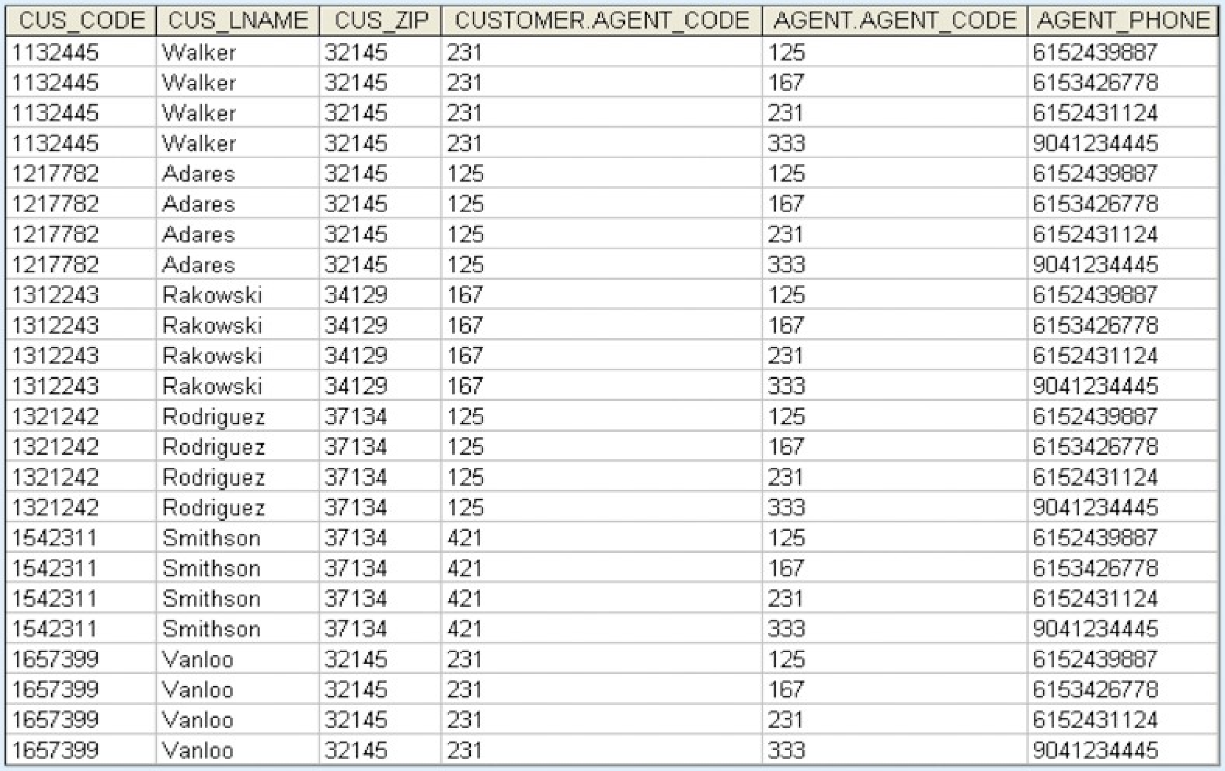
AGENT_CODE	AGENT_PHONE
125	6152439887
167	6153426778
231	6152431124
333	9041234445

SOURCE: Course Technology/Cengage Learning

A natural join is the result of a three-stage process.

1. Create a PRODUCT of the tables

FIGURE 3.11 **Natural join, Step 1: PRODUCT**

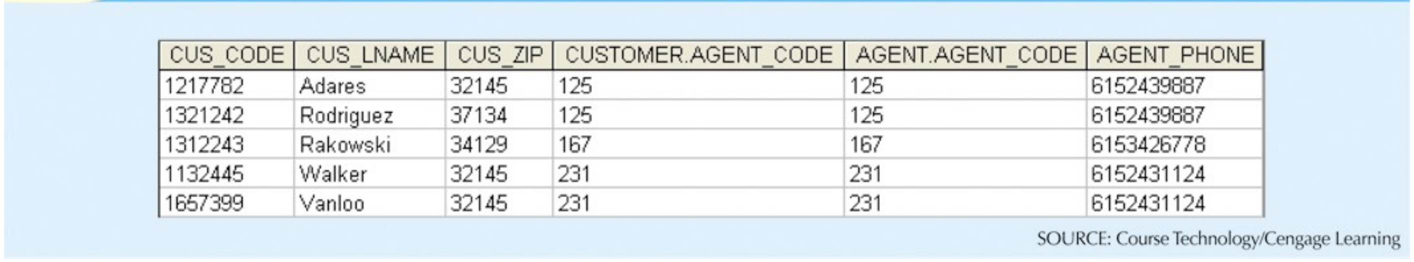


CUS_CODE	CUS_LNAME	CUS_ZIP	CUSTOMER.AGENT_CODE	AGENT.AGENT_CODE	AGENT_PHONE
1132445	Walker	32145	231	125	6152439887
1132445	Walker	32145	231	167	6153426778
1132445	Walker	32145	231	231	6152431124
1132445	Walker	32145	231	333	9041234445
1217782	Adares	32145	125	125	6152439887
1217782	Adares	32145	125	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

SOURCE: Course Technology/Cengage Learning

2. Use a SELECT to yield the rows that the **join columns** (AGENT_CODE) values are equal.

FIGURE 3.12 Natural join, Step 2: SELECT

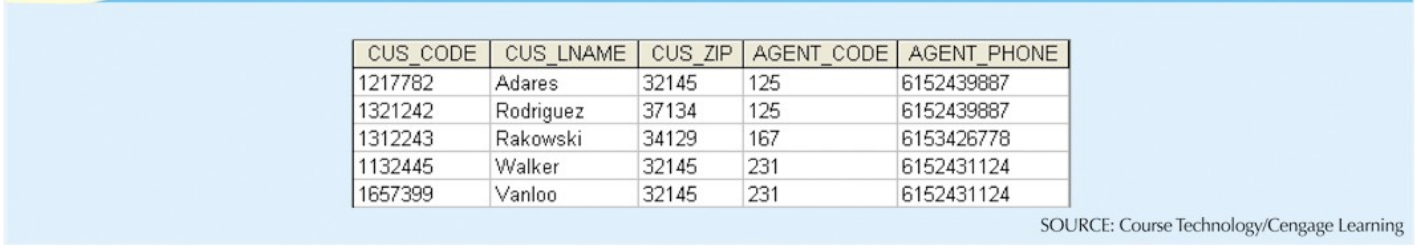


CUS_CODE	CUS_LNAME	CUS_ZIP	CUSTOMER.AGENT_CODE	AGENT.AGENT_CODE	AGENT_PHONE
1217782	Adares	32145	125	125	6152439887
1321242	Rodriguez	37134	125	125	6152439887
1312243	Rakowski	34129	167	167	6153426778
1132445	Walker	32145	231	231	6152431124
1657399	Vanloo	32145	231	231	6152431124

SOURCE: Course Technology/Cengage Learning

3. Perform a PROJECT to yield a single copy of each attribute (remove duplicate columns).

FIGURE 3.13 Natural join, Step 3: PROJECT



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

SOURCE: Course Technology/Cengage Learning

Natural Join Example

sid	bid	day
22	101	10/10/96
58	103	11/12/96

R1

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

S1

S1 $\bowtie$ R1 =

sid	sname	rating	age	bid	day
22	dustin	7	45.0	101	10/10/96
58	rusty	10	35.0	103	11/12/96

JOIN: inner join

- An **inner join** only returns matched records from the tables that are being joined.
  - *Natural join*
  - *Equijoin*: links tables on the basis of *an equality (==) condition* that compares specified columns of each table

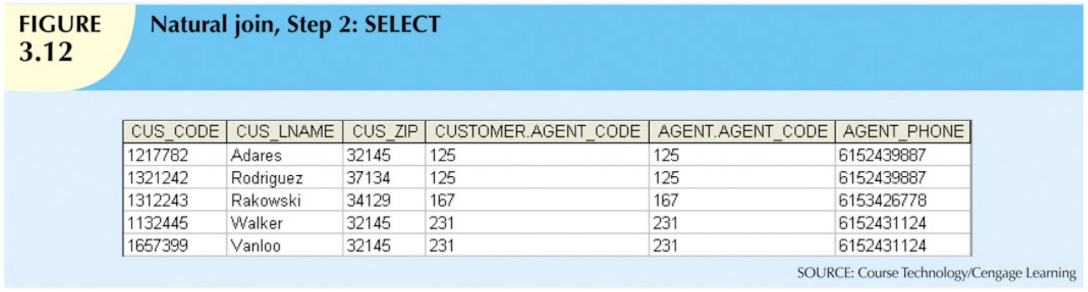


FIGURE 3.12 Natural join, Step 2: SELECT

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

SOURCE: Course Technology/Cengage Learning

  - *Theta join*: links tables on the basis of *any other comparison operator* that compares specified columns of each table

JOIN: outer join

- In an **outer join**, the matched pairs would be retained, and any unmatched values in the other table would be left null.
- Think of an outer join as an “inner join plus”.
- There are three forms of the outer join, depending on which data is to be kept.
  - LEFT OUTER JOIN - keep data from the left-hand table
  - RIGHT OUTER JOIN - keep data from the right-hand table
  - FULL OUTER JOIN - keep data from both tables

JOIN: outer join example 1

R

ColA	*ColB*
A	1
B	2
D	3
F	4
E	5

S

SColA	*SColB*
A	1
C	2
D	3
E	4

R LEFT OUTER JOIN R.ColA = S.SColA S

A	1	A	1
D	3	D	3
E	5	E	4
B	2	-	-
F	4	-	-

R RIGHT OUTER JOIN R.ColA = S.SColA S

A	1	A	1
D	3	D	3
E	5	E	4
-	-	C	2

JOIN: outer join example 2

R

ColA	*ColB*
A	1
B	2
D	3
F	4
E	5

S

SColA	*SColB*
A	1
C	2
D	3
E	4

R FULL OUTER JOIN R.ColA = S.SColA S

A	1	A	1
D	3	D	3
E	5	E	4
B	2	-	-
F	4	-	-
-	-	C	2

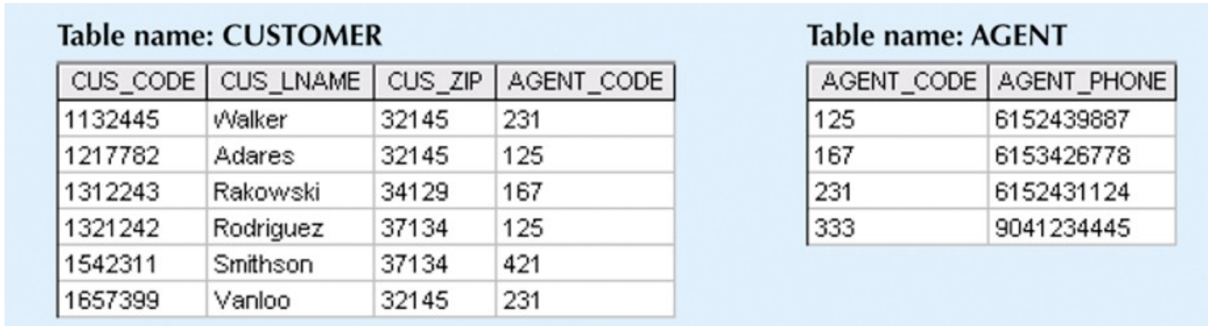


Table name: CUSTOMER

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

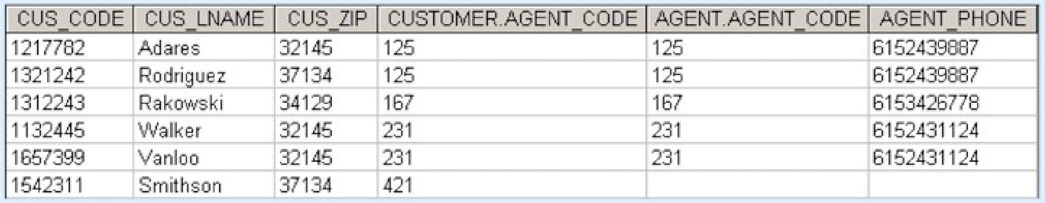
Table name: AGENT

AGENT_CODE	AGENT_PHONE
125	6152439887
167	6153426778
231	6152431124
333	9041234445

Left outer join: yields all of the rows in the CUSTOMER table

CUSTOMER ⟕ AGENT

FIGURE 3.14 Left outer join



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
1542311	Smithson	37134	421		

SOURCE: Course Technology/Cengage Learning

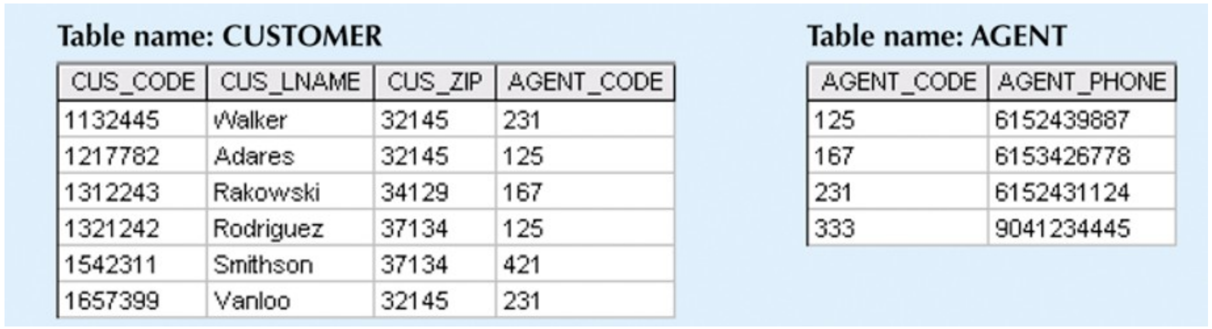


Table name: CUSTOMER

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

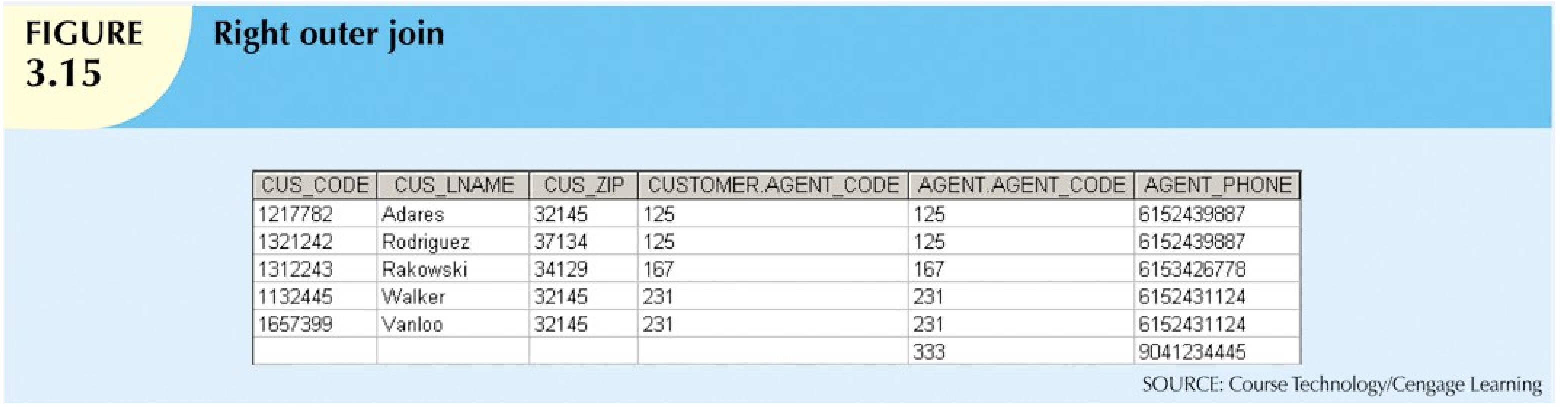
Table name: AGENT

AGENT_CODE	AGENT_PHONE
125	6152439887
167	6153426778
231	6152431124
333	9041234445

Right outer join: yields all of the rows in the AGENT table

CUSTOMER ⟖ AGENT

FIGURE 3.15 Right outer join



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
				333	9041234445

SOURCE: Course Technology/Cengage Learning

DIVIDE: ÷

- Uses one 2-column table as the *dividend* and one single-column table as the *divisor*.
- **The tables must have a common column.**
- Outputs a single column that contains *all values* from the second column of the dividend that are associated with every row in the divisor

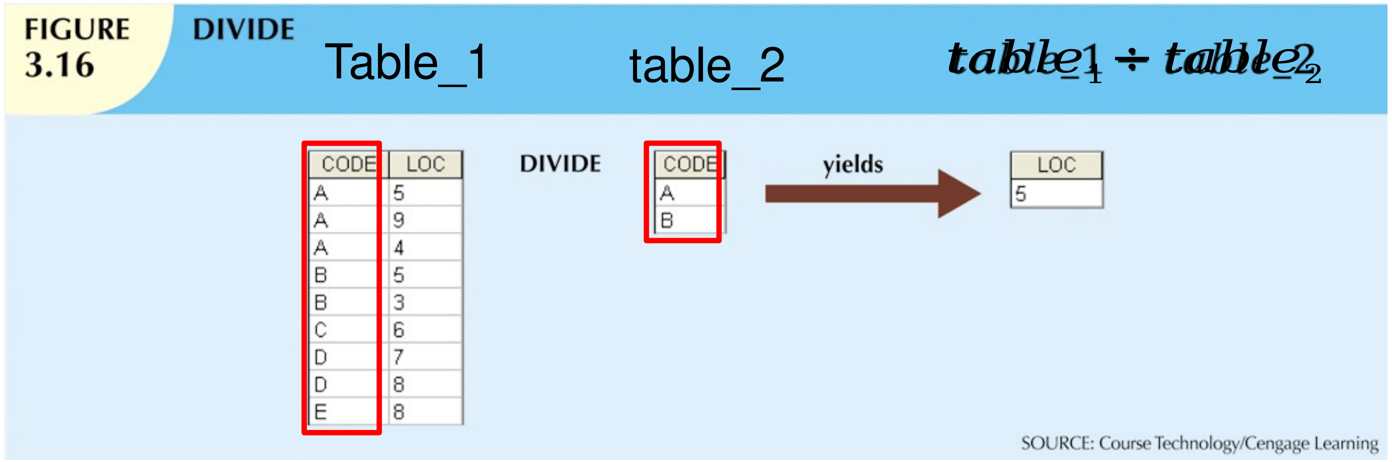


FIGURE 3.16 **DIVIDE**

Table_1 table_2 $table\_1 \div table\_2$

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

DIVIDE

CODE
A
B

yields

LOC
5

SOURCE: Course Technology/Cengage Learning

Thank you & Questions