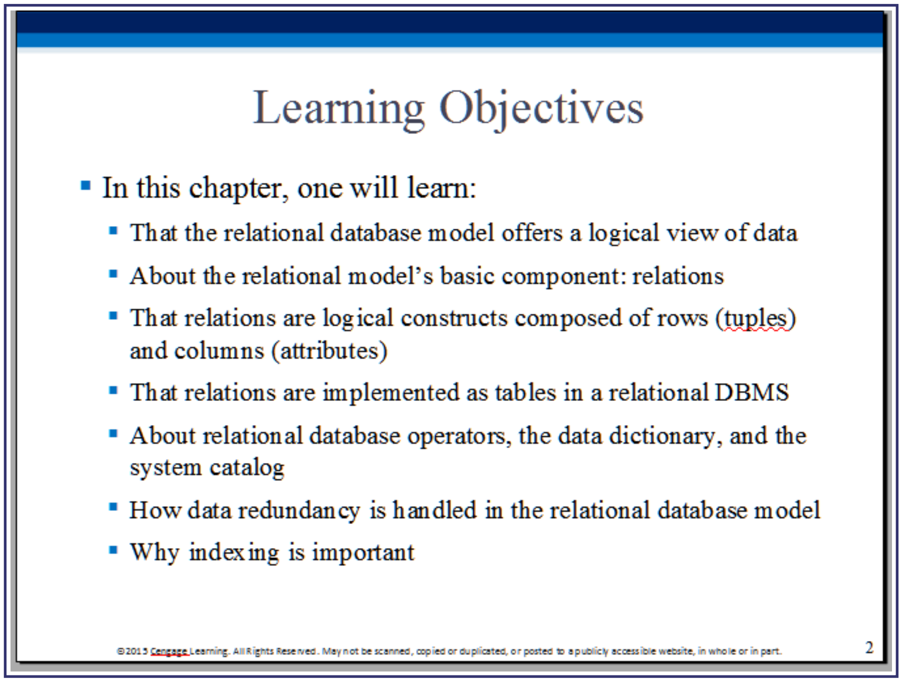
Relational Modeling

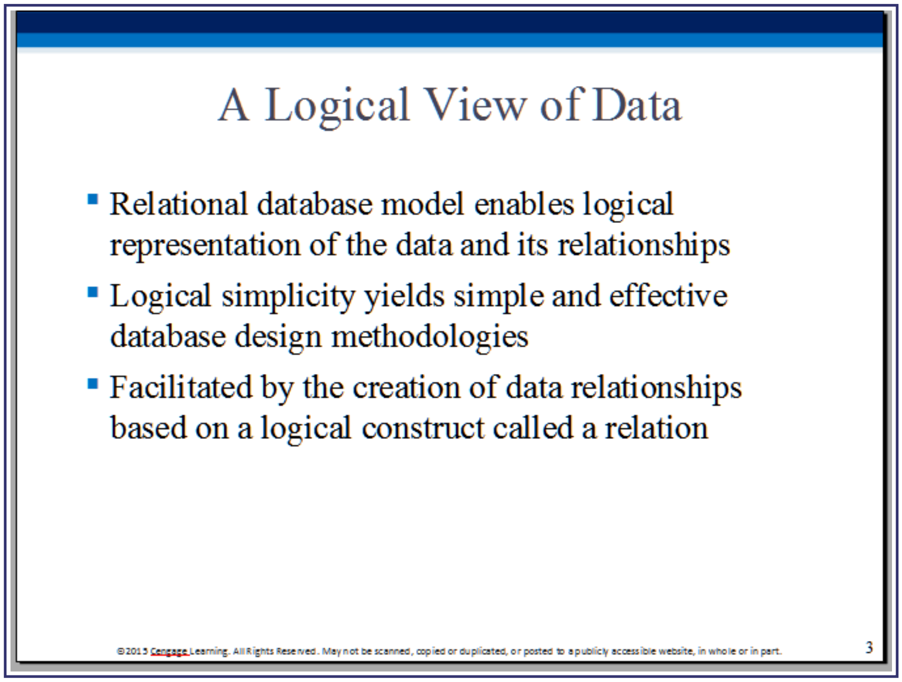
Objectives



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
  - 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

Logical view: 'relation'



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

3

Relational tables

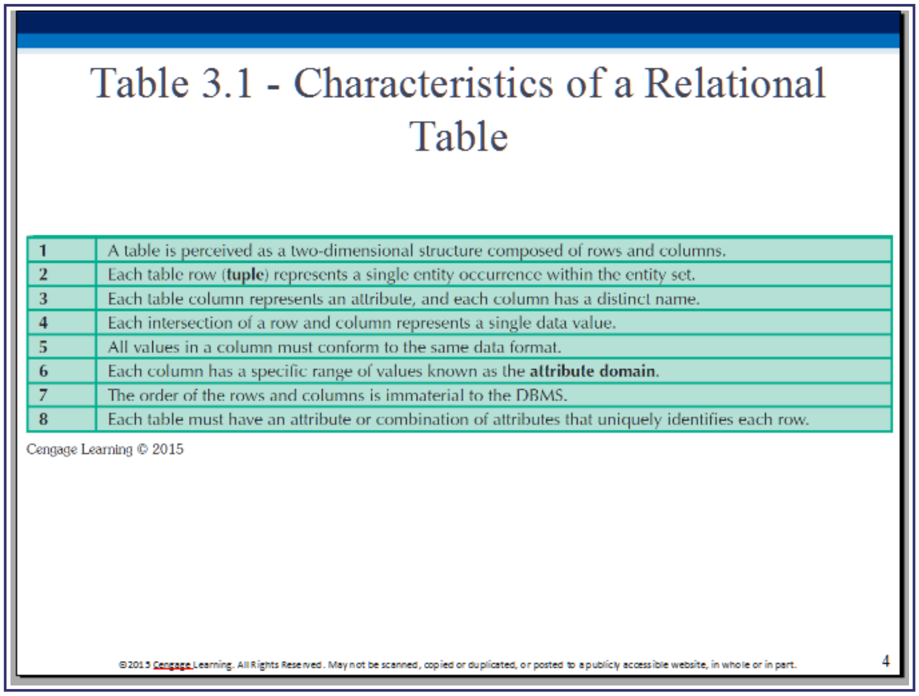
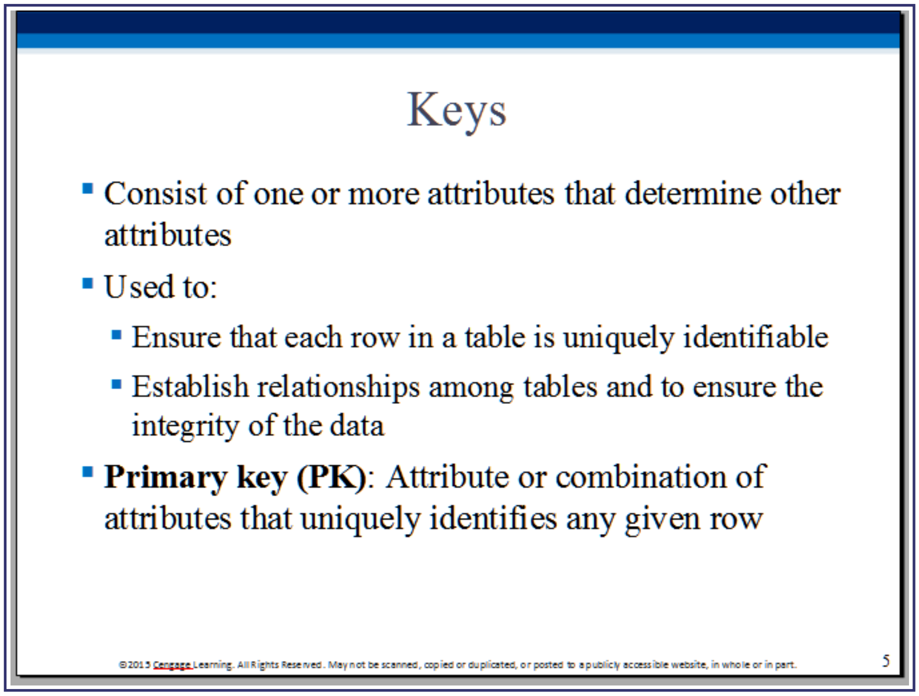


Table 3.1 - Characteristics of a Relational Table

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

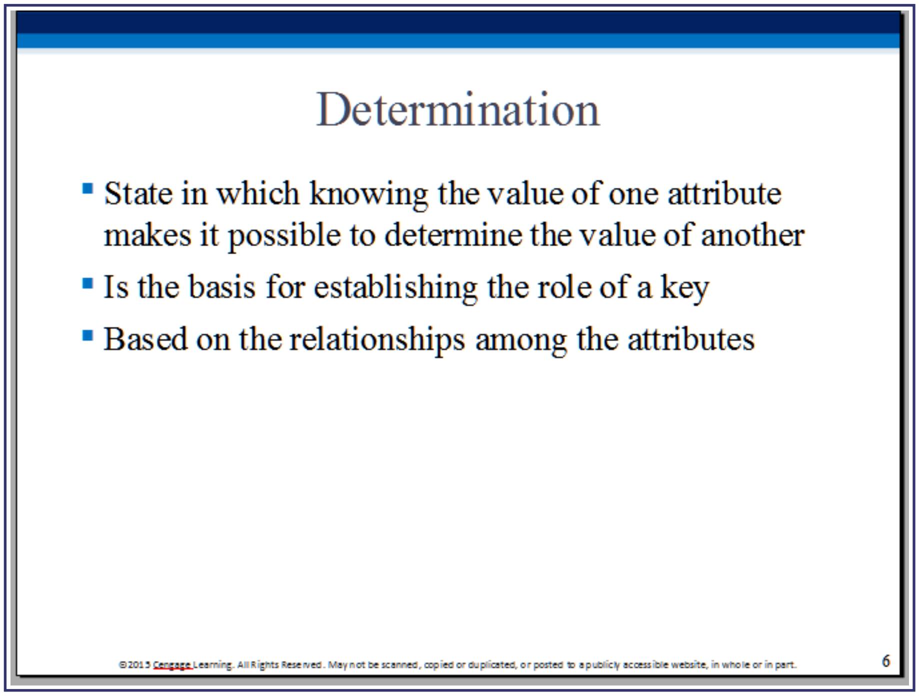
Keys



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

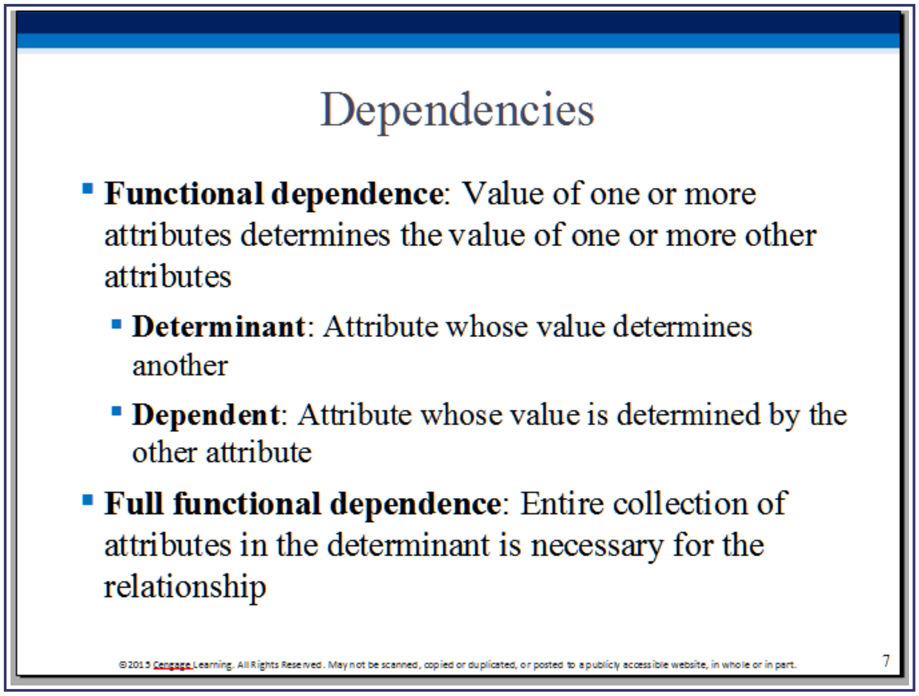
"determines"



Determination

- State in which knowing the value of one attribute makes it possible to determine the value of another
- Is the basis for establishing the role of a key
- Based on the relationships among the attributes

Determinants determine dependents [via] dependencies :)



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

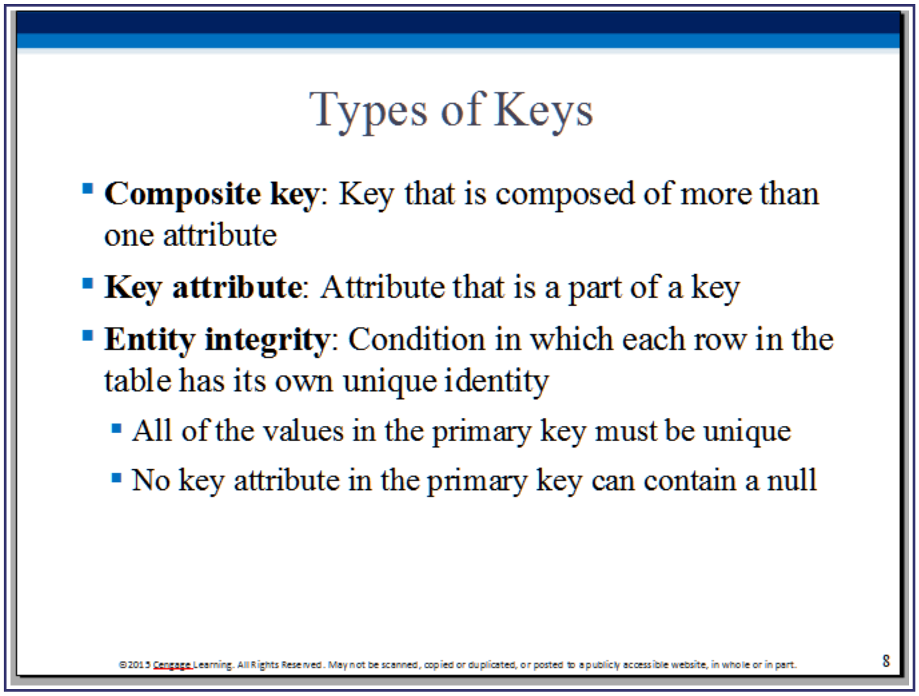
Functional dependency

STU_ID[determinant] ->[functionally determines] STU_LNAME[dependent]

STU_ID,STU_LNAME -> GPA is NOT a 'full functional dependency' because the determinant contains an extra (unwanted) attr (STU_LNAME)

STU_LNAME,STU_FNAME -> GPA is a 'full functional dependency' (assuming lastname,firstname is unique)

Composite key; entity integrity



Types of Keys

- **Composite key**: Key that is composed of more than one attribute
- **Key attribute**: Attribute that is a part of a key
- **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

Nulls; referential integrity

Types (categories) of keys

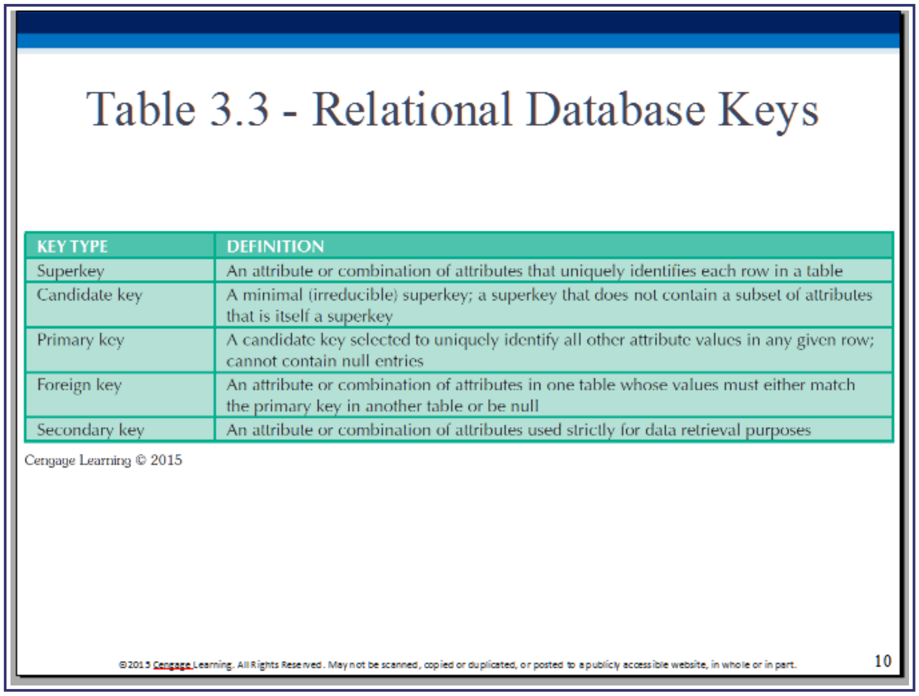


Table 3.3 - Relational Database Keys

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

Keys: many types

* primary (foreign) keys are a subset of candidate keys are a subset of superkeys

* simple keys vs compound keys vs composite keys

* natural keys - keys that are created from real-world entities (eg. for a US resident, their SSN could be a natural key)

* surrogate keys (just make up brand new unique keys)

* secondary, or 'alternate' keys

You can read a bit more keys here.

Example relation

Nulls - avoid where possible!

Relational 'algebra' [fun with one, two or more tables]

Operations on tables [table(s) in, table out, ie. "closure"]

There are (only) EIGHT 'relational set operators' (defined by Ed Codd, at IBM, in 1970), which are all used to operate ("perform relational algebra") on tables: Select, Project, Union, Intersect, Difference, Product, Join, Divide. This is no exaggeration: these operators are the basis for SQL and the entire relational DB industry!

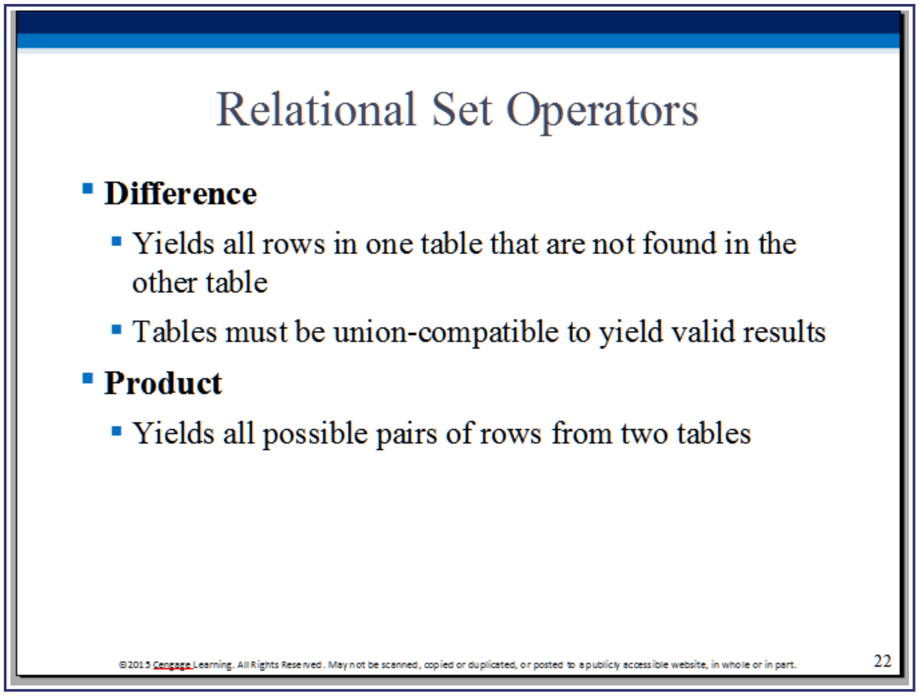
SELECT [outputs a subset of rows]

PROJECT [outputs a subset of cols]

UNION [eqvt to 'cat a b > c']

INTERSECT [rows common to a and b]

Difference; Product



Relational Set Operators

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

Difference [a - b]

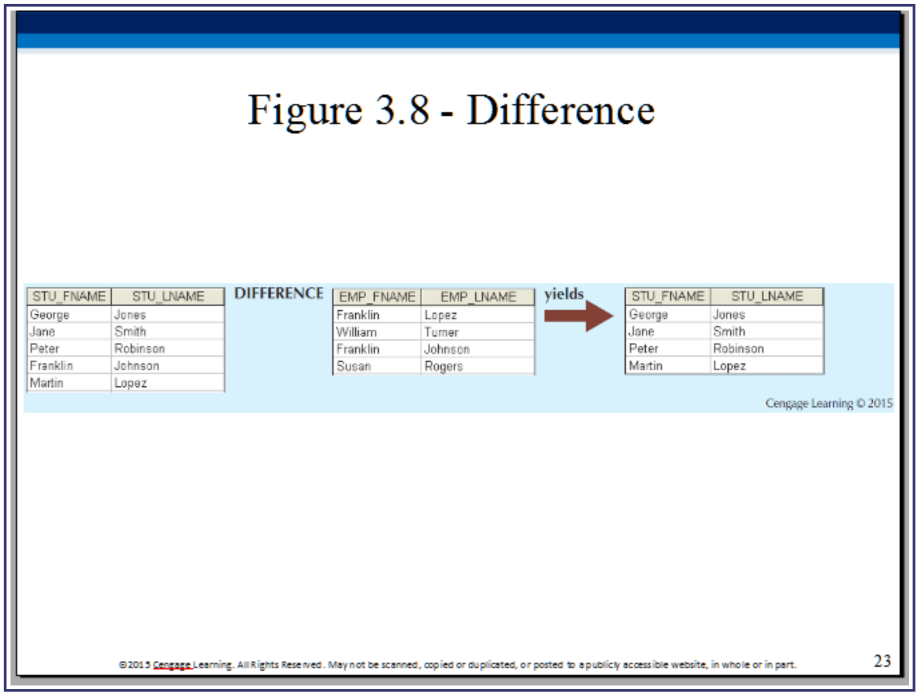


Figure 3.8 - Difference

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

Product [multiply rows, add columns]

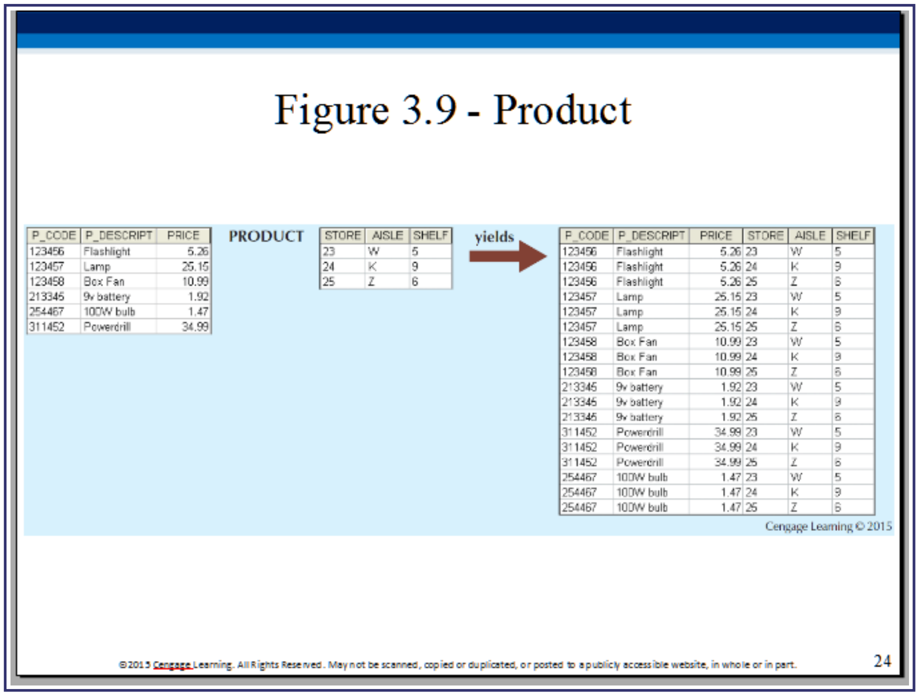


Figure 3.9 - Product

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

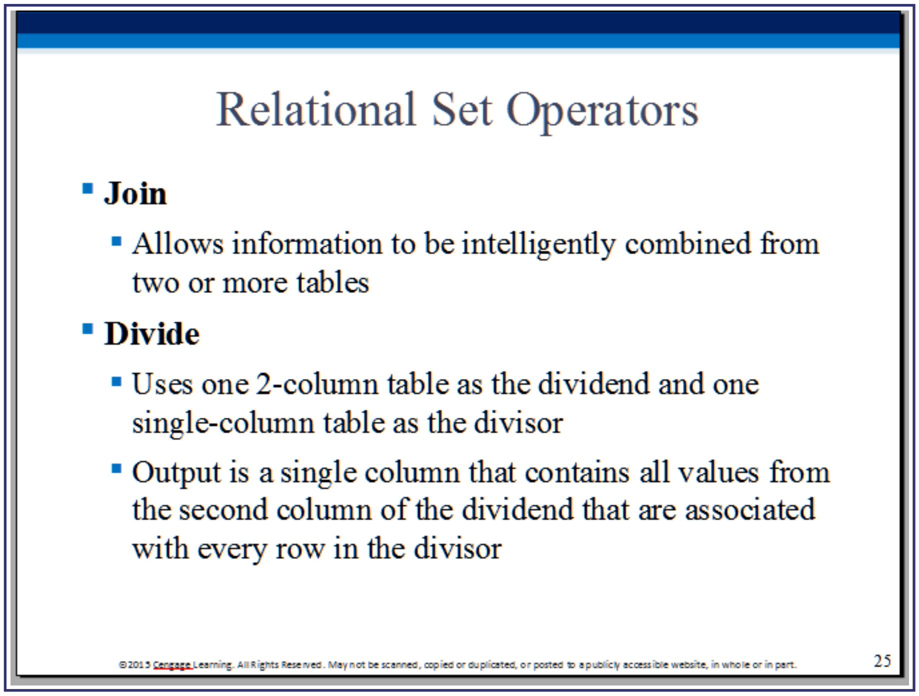
STORE	AISLE	SHELF
23	W	5
24	K	9
25	Z	6

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

JOIN (several kinds); DIVIDE (?!)



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

DIVIDE

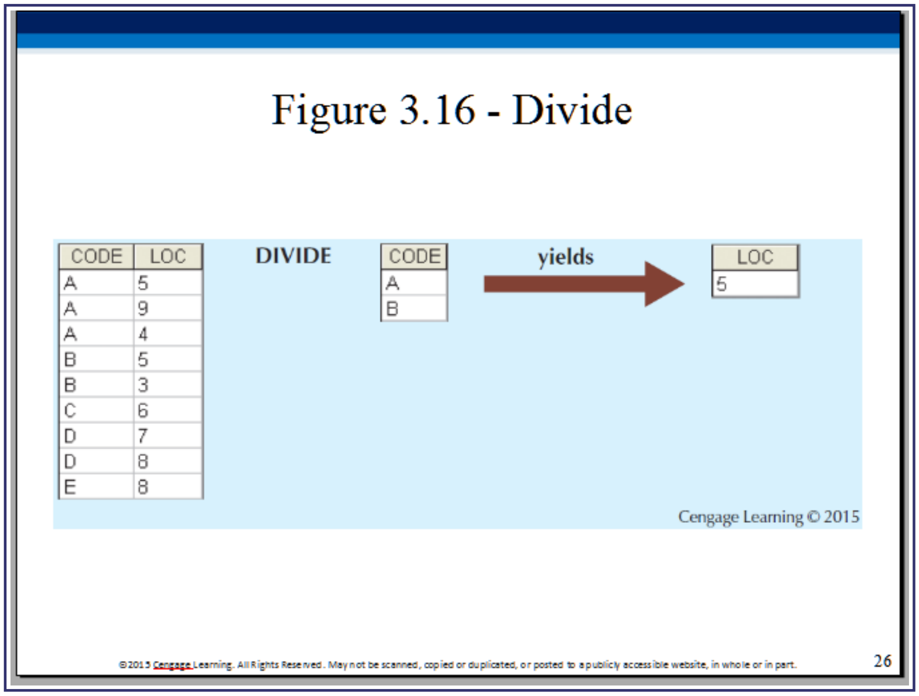


Figure 3.16 - Divide

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

DIVIDE

CODE
A
B

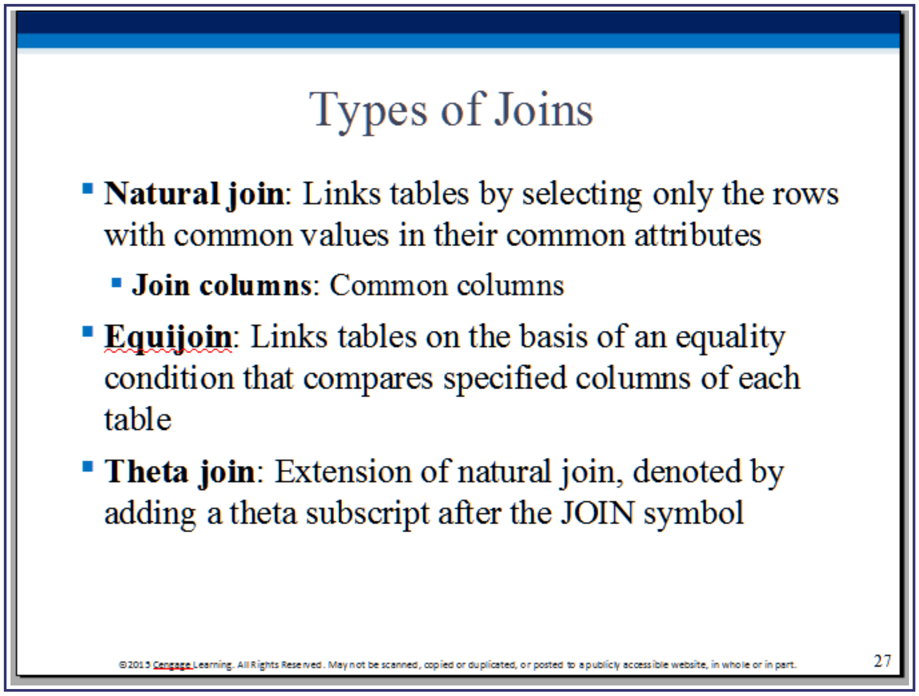
yields

LOC
5

Cengage Learning © 2015

26

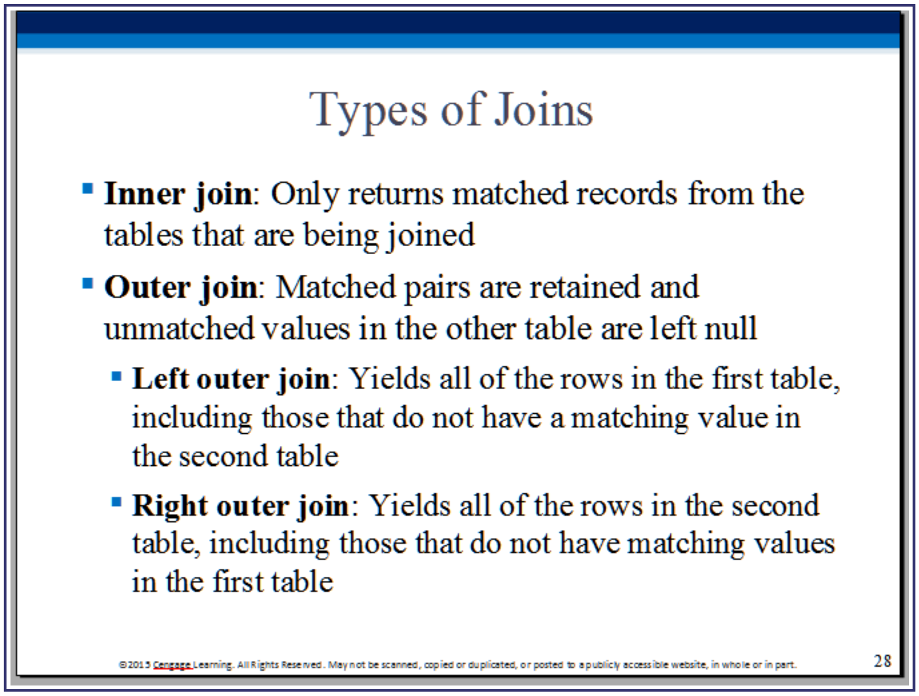
JOIN



Types of Joins

- **Natural join**: Links tables by selecting only the rows with common values in their common attributes
  - **Join columns**: Common columns
- **Equijoin**: Links tables on the basis of an equality condition that compares specified columns of each table
- **Theta join**: Extension of natural join, denoted by adding a theta subscript after the JOIN symbol

JOIN [cont'd]



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

Tables to illustrate JOIN operations

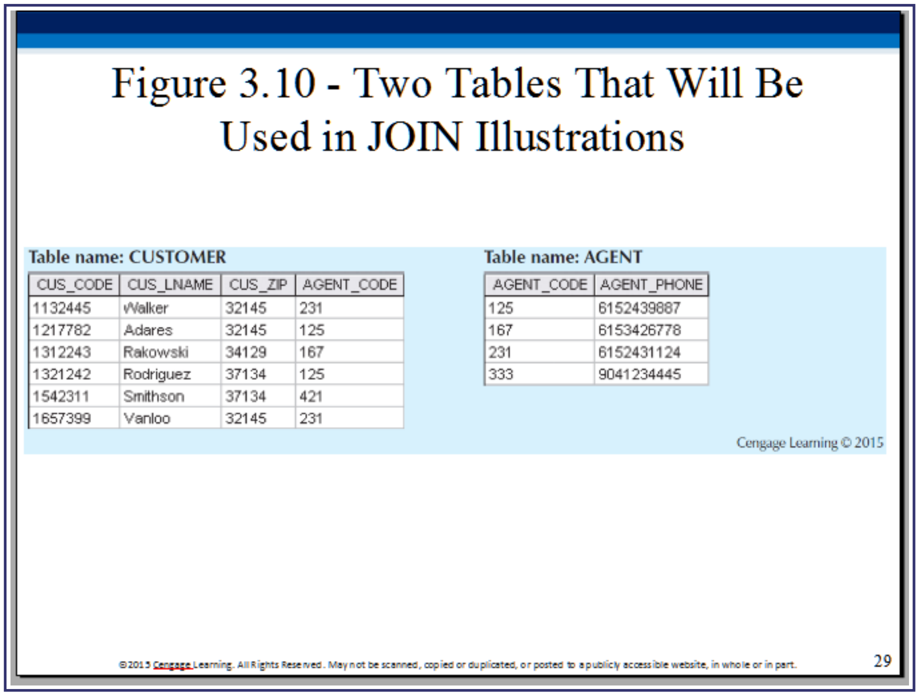


Figure 3.10 - Two Tables That Will Be Used in JOIN Illustrations

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

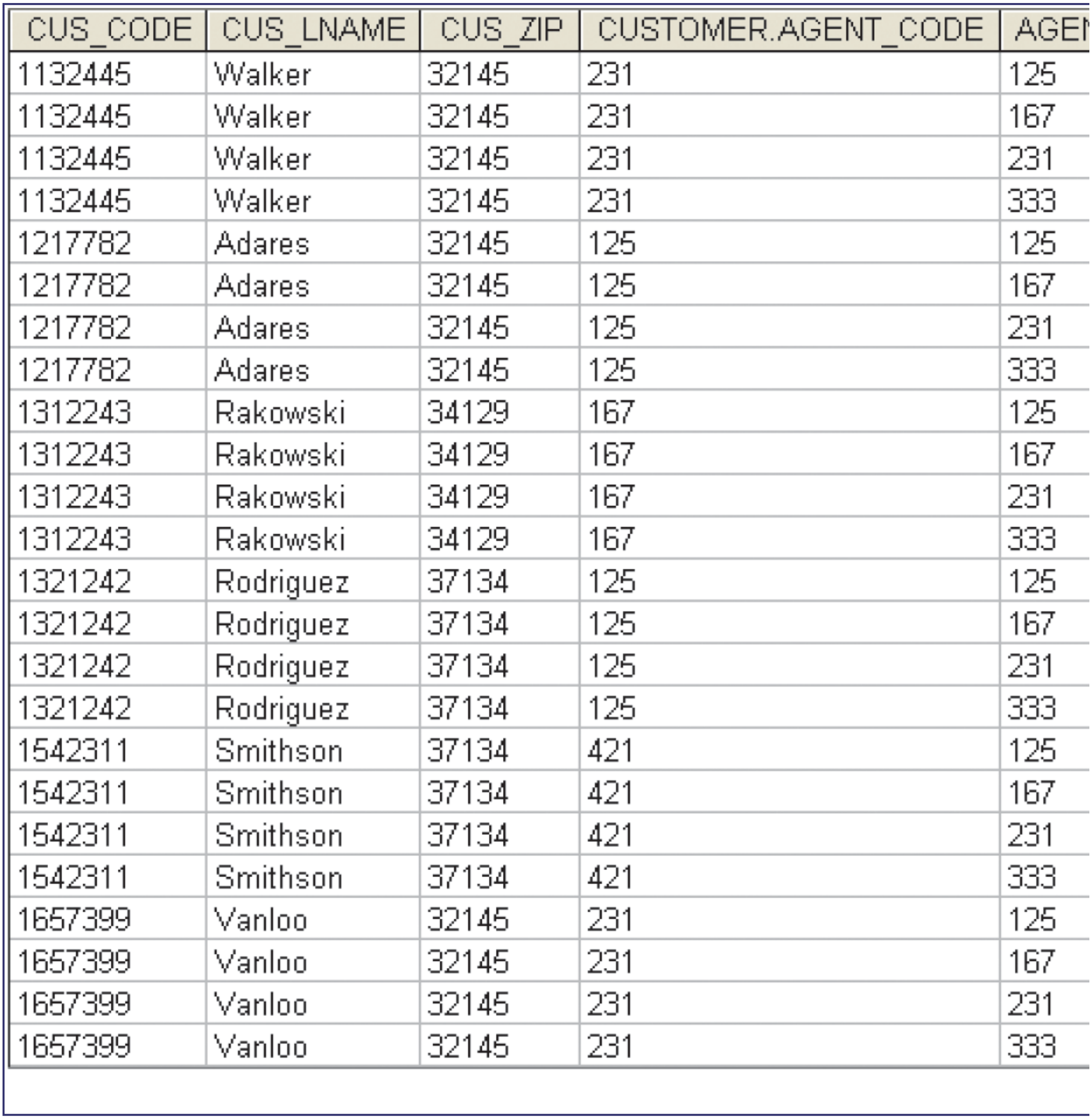
Natural join

A natural join links tables by selecting from two tables, only those rows that have common (identical) values for common attributes.

These three steps result in a natural join: create product, select, project.

Natural join: product

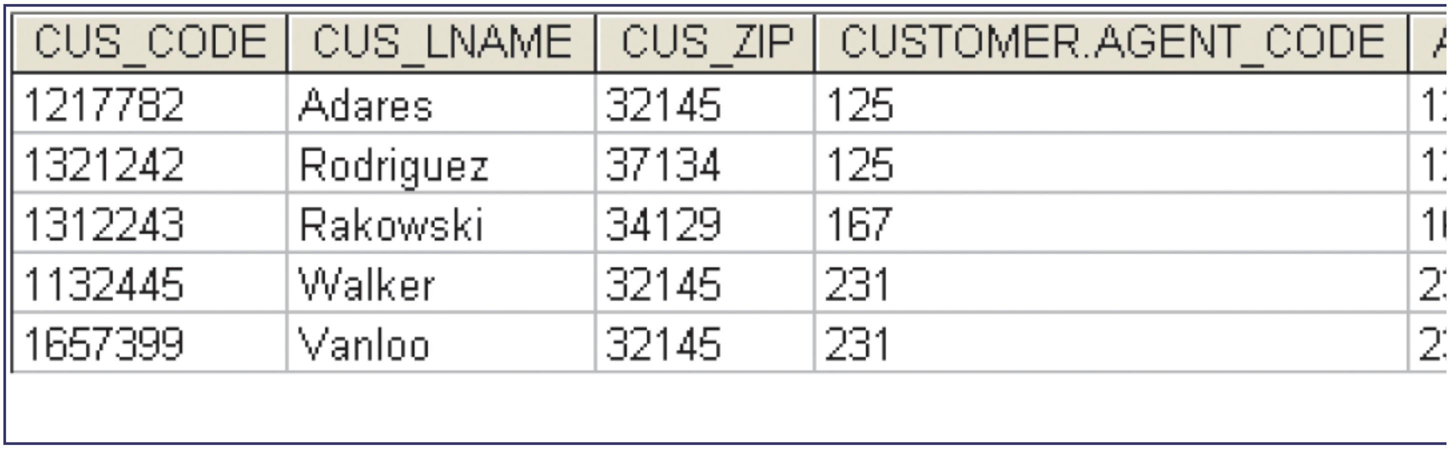
Cartesian product of the two tables:



CUS_CODE	CUS_LNAME	CUS_ZIP	CUSTOMER.AGENT_CODE	AGEN
1132445	Walker	32145	231	125
1132445	Walker	32145	231	167
1132445	Walker	32145	231	231
1132445	Walker	32145	231	333
1217782	Adares	32145	125	125
1217782	Adares	32145	125	167
1217782	Adares	32145	125	231
1217782	Adares	32145	125	333
1312243	Rakowski	34129	167	125
1312243	Rakowski	34129	167	167
1312243	Rakowski	34129	167	231
1312243	Rakowski	34129	167	333
1321242	Rodriguez	37134	125	125
1321242	Rodriguez	37134	125	167
1321242	Rodriguez	37134	125	231
1321242	Rodriguez	37134	125	333
1542311	Smithson	37134	421	125
1542311	Smithson	37134	421	167
1542311	Smithson	37134	421	231
1542311	Smithson	37134	421	333
1657399	Vanloo	32145	231	125
1657399	Vanloo	32145	231	167
1657399	Vanloo	32145	231	231
1657399	Vanloo	32145	231	333

Natural join: select

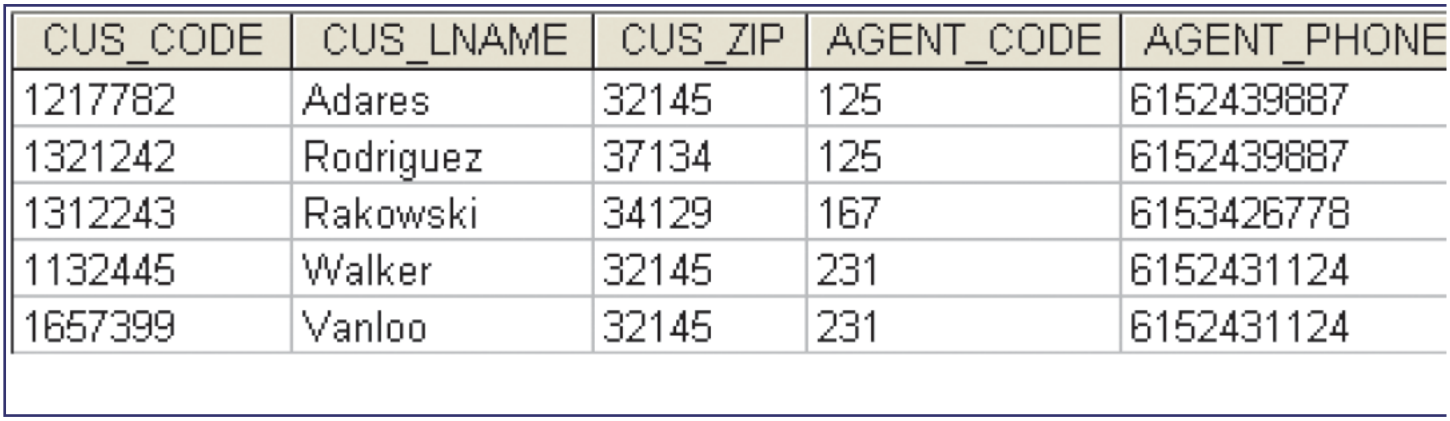
Select only rows with identical values in the common (joining) columns:



CUS_CODE	CUS_LNAME	CUS_ZIP	CUSTOMER.AGENT_CODE	A
1217782	Adares	32145	125	1
1321242	Rodriguez	37134	125	1
1312243	Rakowski	34129	167	1
1132445	Walker	32145	231	2
1657399	Vanloo	32145	231	2

Natural join: project

Project away, ie. remove one of the two duplicate columns:

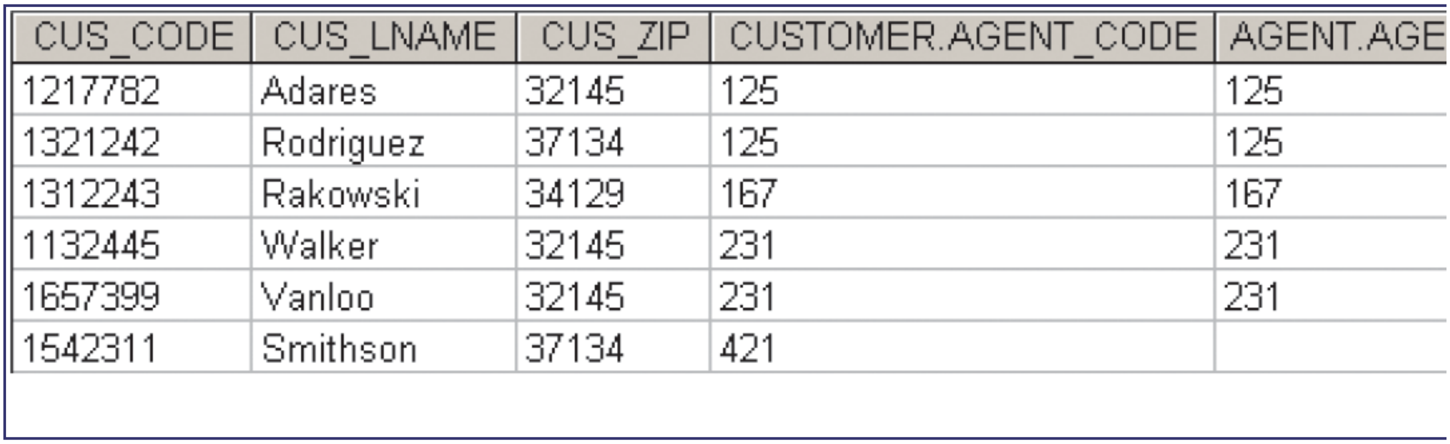


CUS_CODE	CUS_LNAME	CUS_ZIP	AGENT_CODE	AGENT_PHONE
1217782	Adares	32145	125	6152439887
1321242	Rodriguez	37134	125	6152439887
1312243	Rakowski	34129	167	6153426778
1132445	Walker	32145	231	6152431124
1657399	Vanloo	32145	231	6152431124

Result (the table above): natural join.

Left outer join

Output all rows of the left (CUSTOMER) table, including ones for which there are no matching values in the join column in the other (AGENT) table:

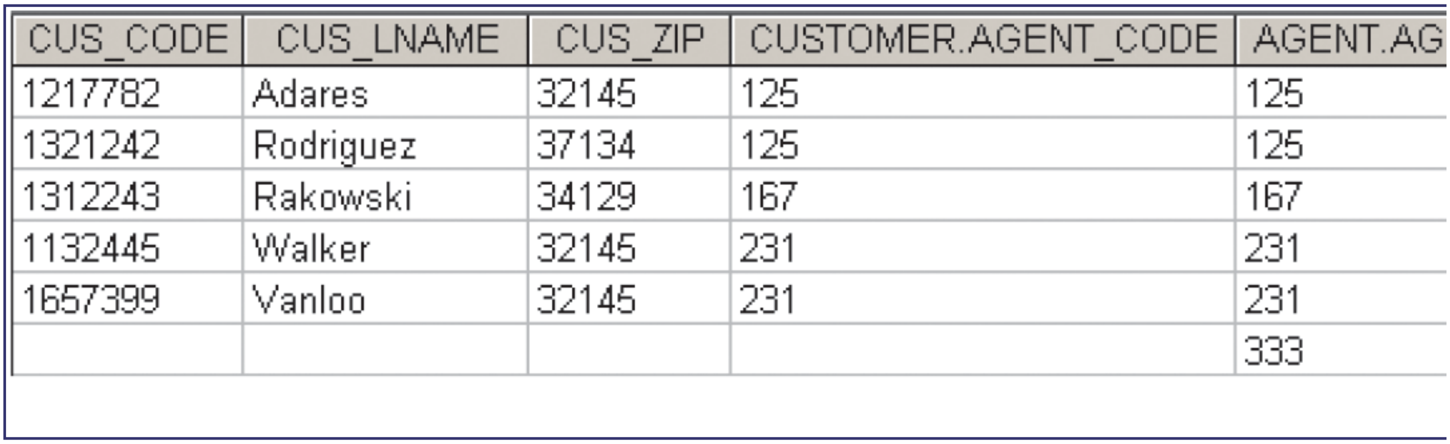


CUS_CODE	CUS_LNAME	CUS_ZIP	CUSTOMER.AGENT_CODE	AGENT.AGE
1217782	Adares	32145	125	125
1321242	Rodriguez	37134	125	125
1312243	Rakowski	34129	167	167
1132445	Walker	32145	231	231
1657399	Vanloo	32145	231	231
1542311	Smithson	37134	421	

Note that an outer join is an "inner join plus" [NOT an opposite of inner join].

Right outer join

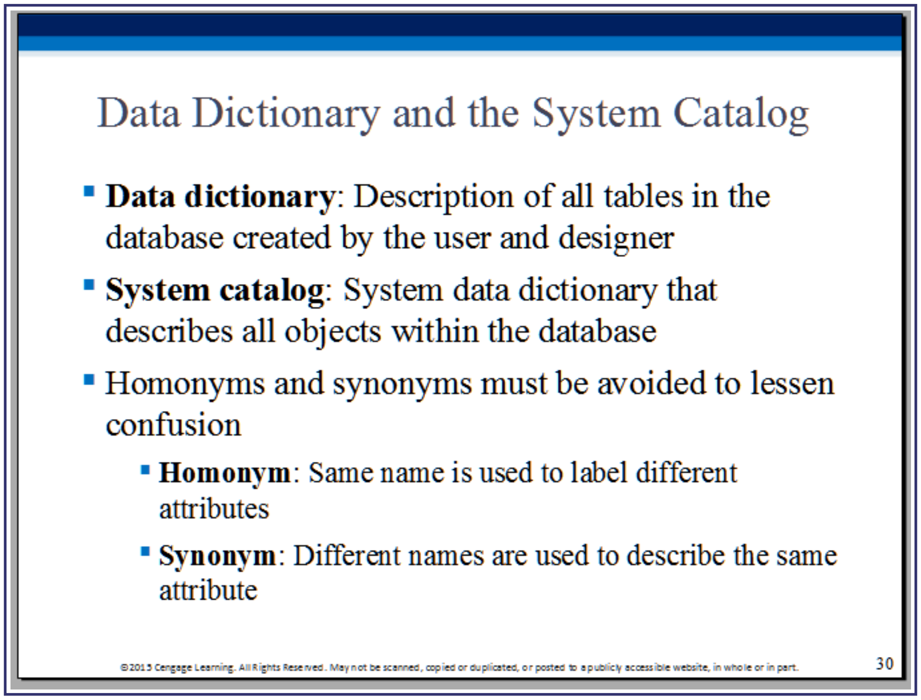
Output all rows of the right (AGENT) table, including ones for which there are no matching values in the join column in the other (CUSTOMER) table:



CUS_CODE	CUS_LNAME	CUS_ZIP	CUSTOMER.AGENT_CODE	AGENT.AG
1217782	Adares	32145	125	125
1321242	Rodriguez	37134	125	125
1312243	Rakowski	34129	167	167
1132445	Walker	32145	231	231
1657399	Vanloo	32145	231	231
				333

Outer joins are useful in exposing missing information [in our example, customers who don't seem to have an agent, agents who don't seem to have customers].

Dictionaries [hold metadata]



Data Dictionary and the System Catalog

- **Data dictionary**: Description of all tables in the database created by the user and designer
- **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
  - **Synonym**: Different names are used to describe the same attribute