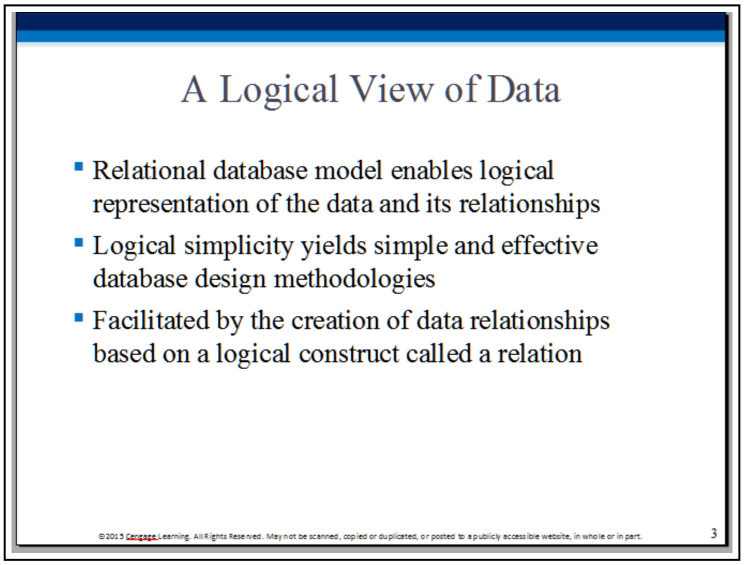
Relational Modeling

Logical view: 'relation'



Relational tables

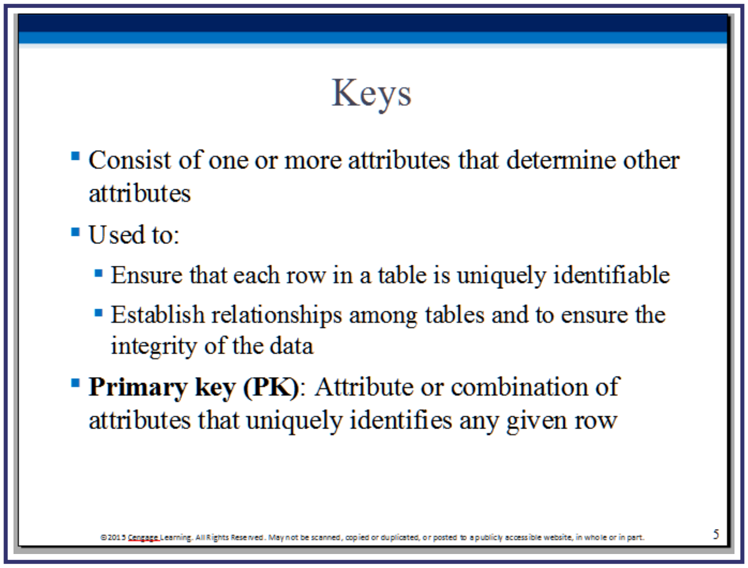
Table 3.1 - Characteristics of a Relational Table

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.

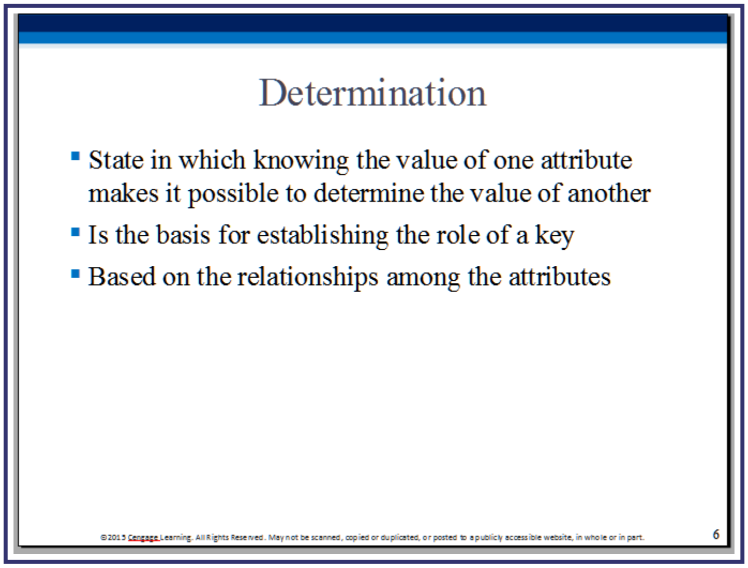
Cengage Learning © 2015

 4

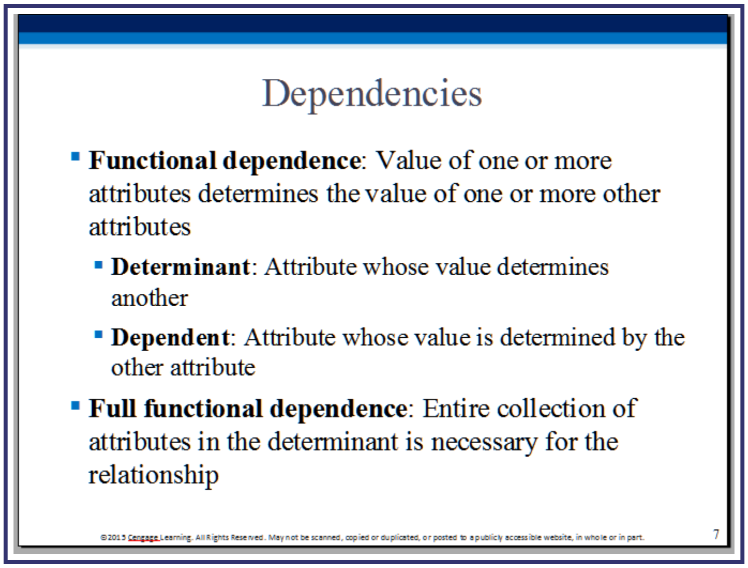
Keys


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
©2015 Cengage Learning. All Rights Reserved. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.
5

"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
©2015 Cengage Learning. All Rights Reserved. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.
6

Determinants determine dependents [via] dependencies :)



'Full' functional dependence is a "good" thing.

Functional dependency

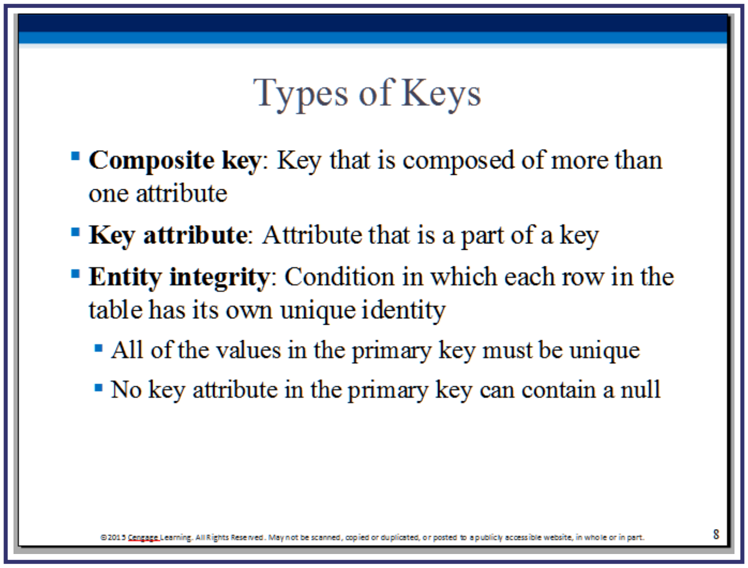
STU_ID[determinant] ->[functionally determines] STU_LNAME[dependent]

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)

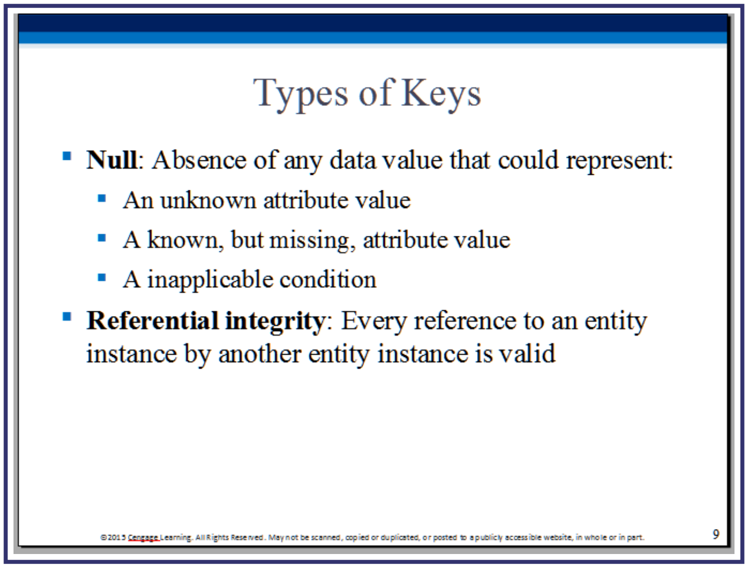
Solemnly swear: "The key, the whole key, and nothing but the key, so help me Codd." :) :)

Composite key; entity integrity



A table 'cannot not' have entity integrity!!

Nulls; referential integrity



Whereas entity integrity has to do with a single table, referential integrity relates to two tables (loosely, 'don't allow invalid pointers').

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

10

Keys: many types

* primary (foreign) keys are a subset of candidate keys are a subset of superkeys (note - superkeys could be 'wasteful', ie. contain superfluous, non-needed attrs)

* 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

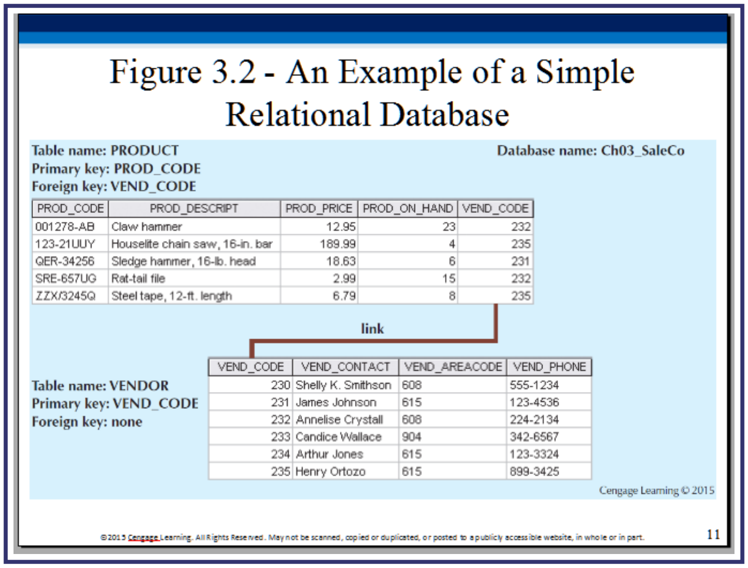


Figure 3.2 - An Example of a Simple Relational Database

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

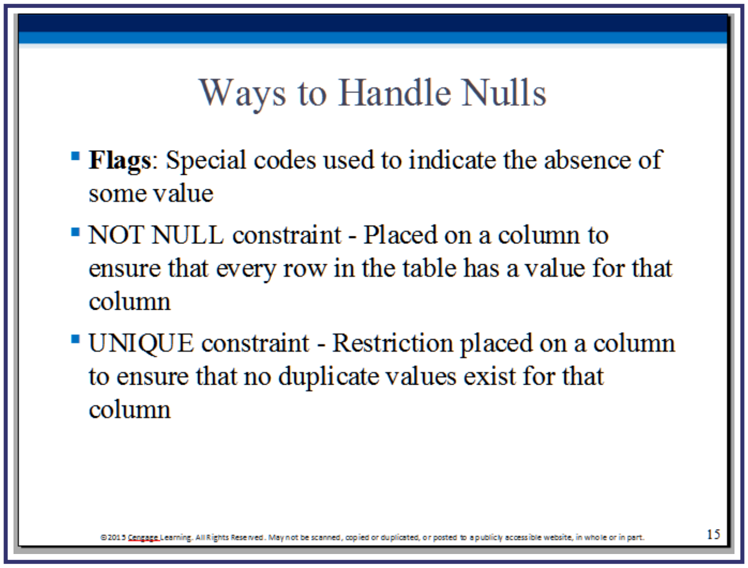
Table name: VENDOR
Primary key: VEND_CODE
Foreign key: none

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

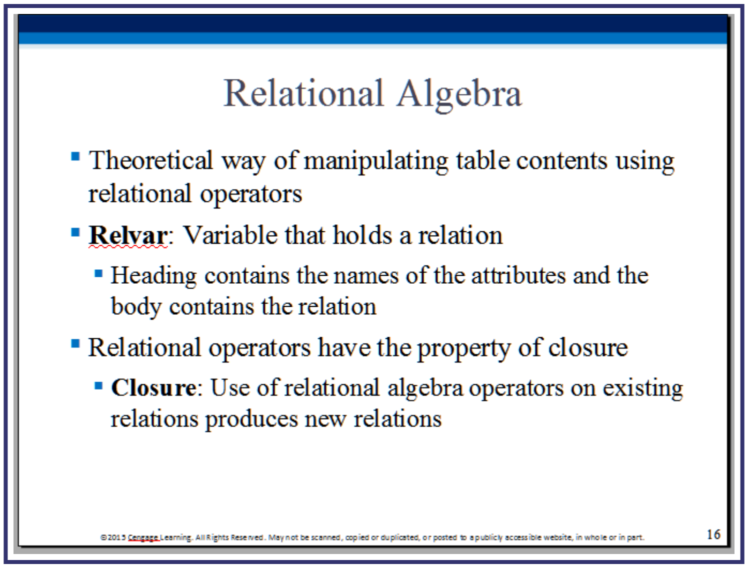
11

Nulls - avoid where possible!



In RL, NULLs can't be entirely avoided (look here, for 'interpreted as any of the following').

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



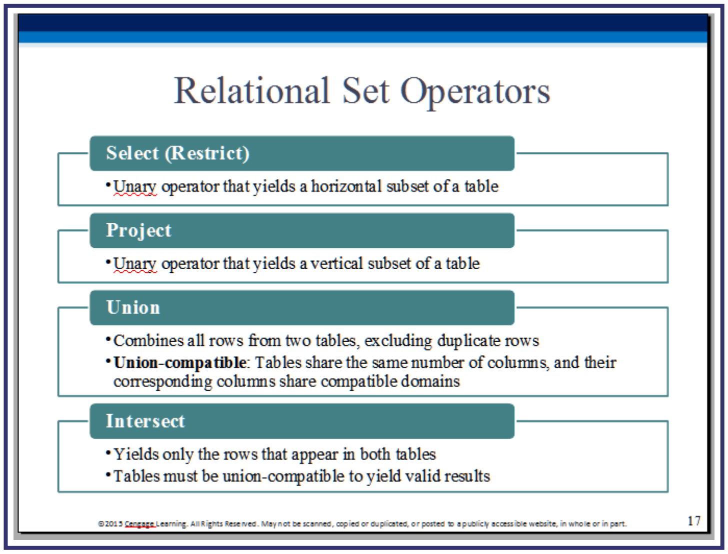
What if a table were a datatype (similar to an int, Vec3D, ComplexNumber, etc)?! Specifically, what operations could be perform on them (eg. similar to addition, square root on doubles)?!

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

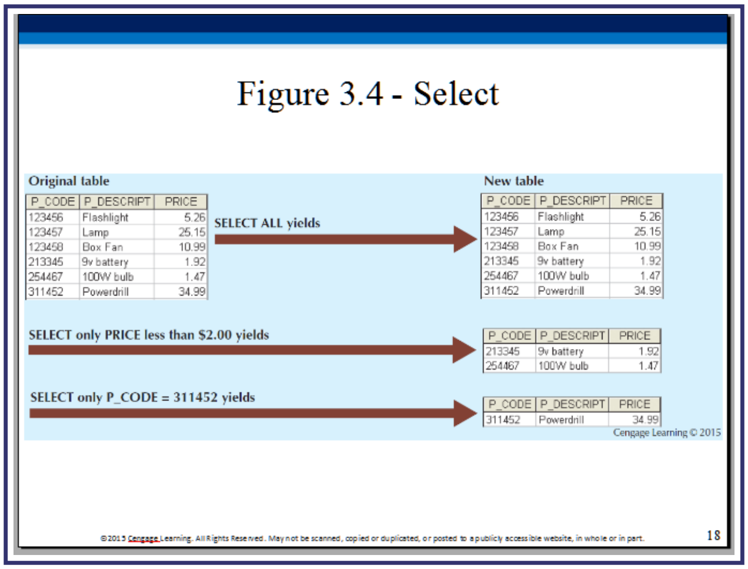
There are (only) EIGHT 'relational set operators' (defined by Ed Codd, at IBM, in 1970), which are all used to operate ("perform relational algebra") on tables: Select, Project, Union, Intersect, Difference, Product, Join, Divide.

This is no exaggeration: these operators are the basis for SQL and the entire relational DB industry!

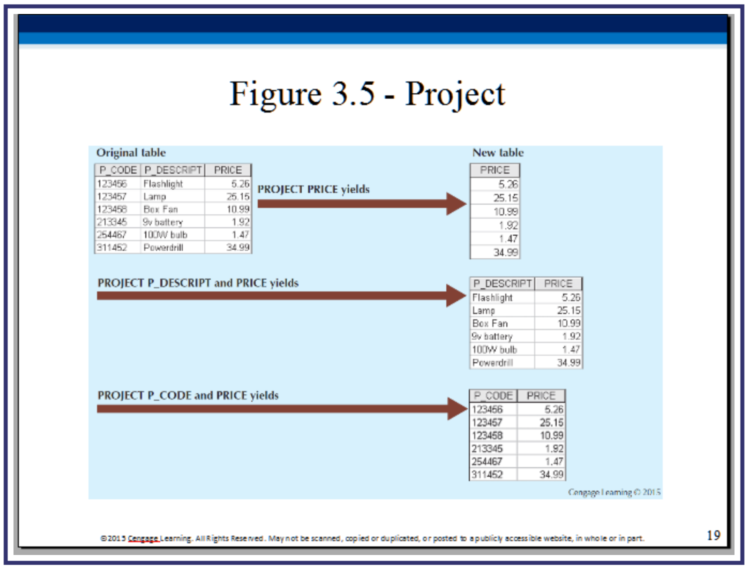
SELECT; PROJECT; UNION; INTERSECT



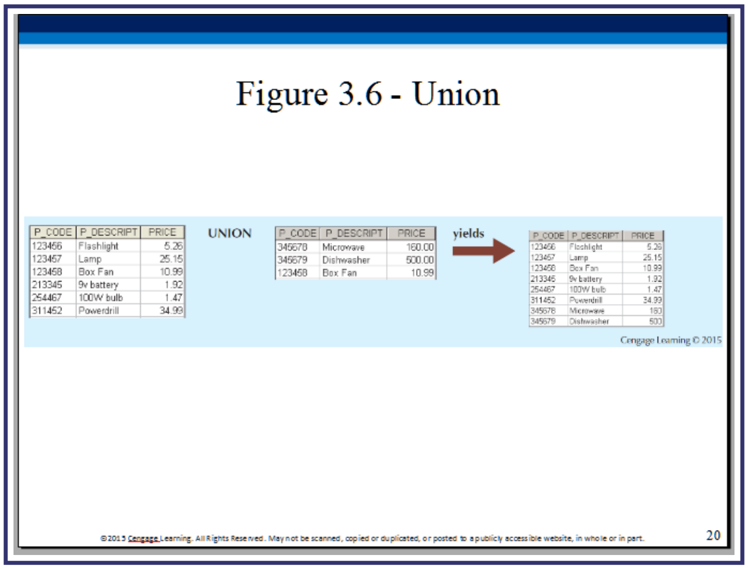
SELECT [outputs a subset of rows]



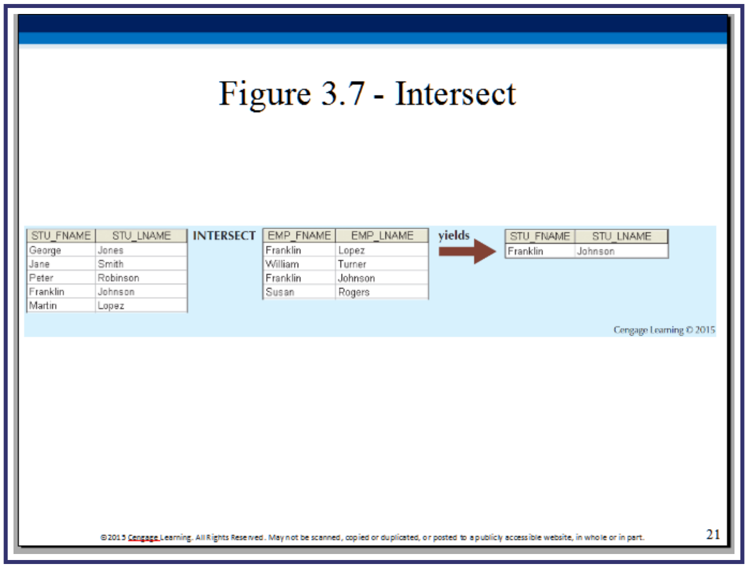
PROJECT [outputs a subset of cols]



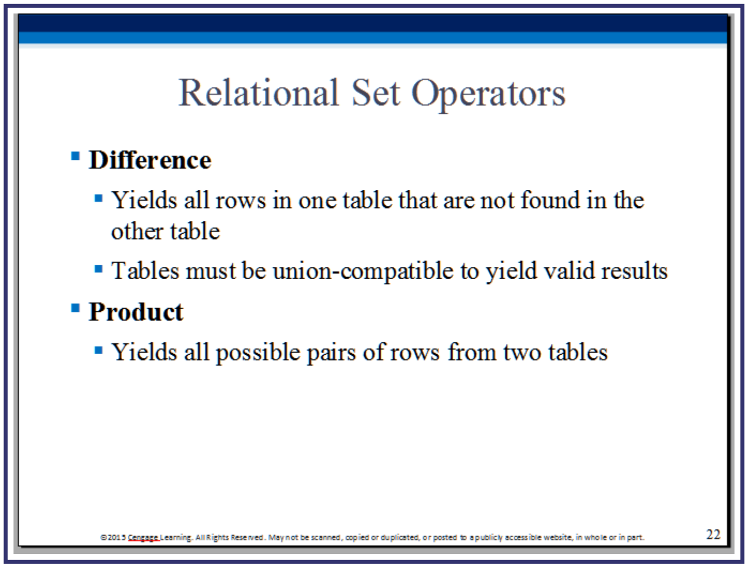
UNION [eqvt to 'cat a b > c']



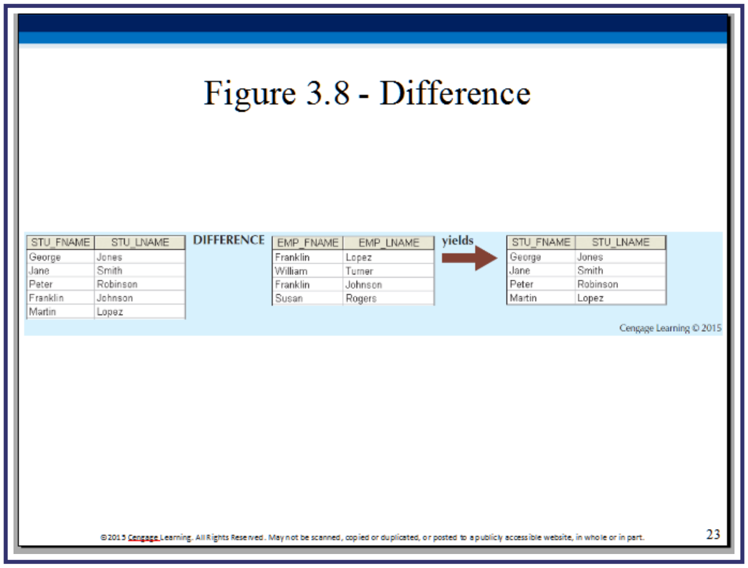
INTERSECT [rows common to a and b]



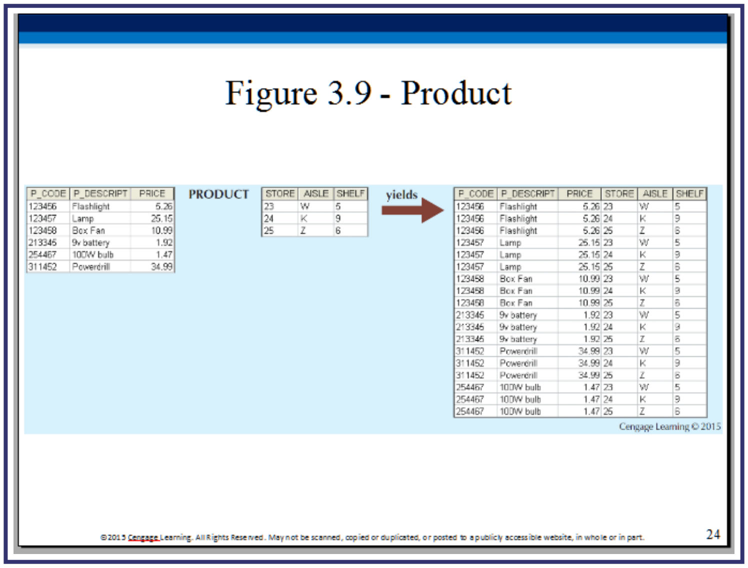
Difference; Product



Difference [a - b]



Product [multiply rows, add columns]

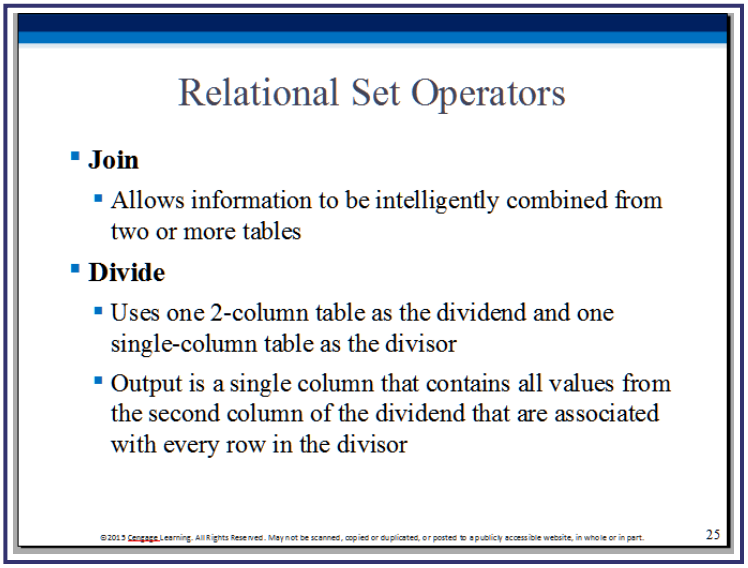


P_CODE	P_DESCRIPT	PRICE
123456	Flashlight	5.26
123457	Lamp	25.15
123458	Box Fan	10.99
213345	9v battery	1.92
254467	100W bulb	1.47
311452	Powerdrill	34.99

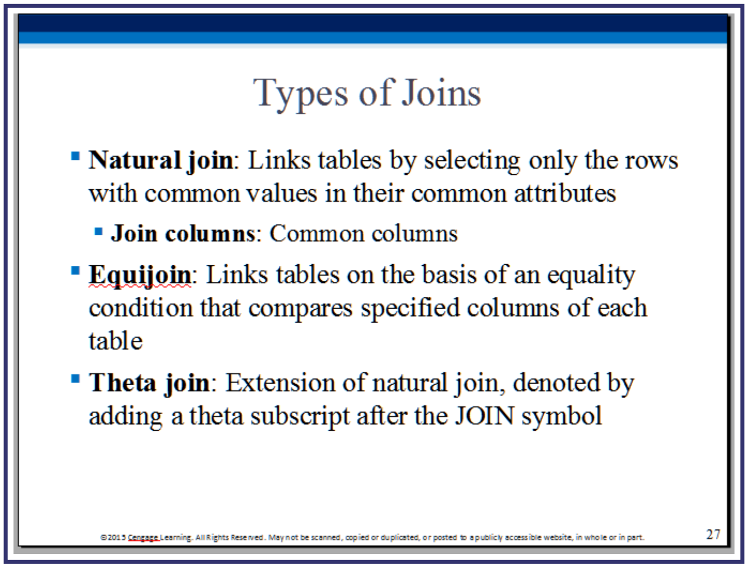
STORE	AISLE	SHELF
23	W	5
24	K	9
25	Z	6

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

JOIN (several kinds); DIVIDE (?!)



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
©2015 Cengage Learning. All Rights Reserved. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.
27

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

 28

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

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 (product of rows, juxtaposition of columns):

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

Natural join: select

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

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.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		

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

Right outer join, full outer join

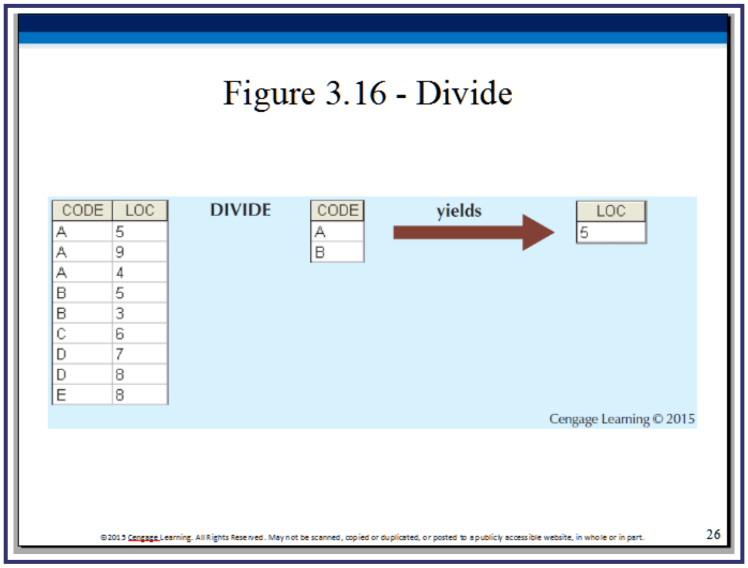
Output all rows of the right (AGENT) table, including ones for which there are no matching values in the join column in the other (CUSTOMER) table:

CUS_CODE	CUS_LNAME	CUS_ZIP	CUSTOMER.AGENT_CODE	AGENT.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

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

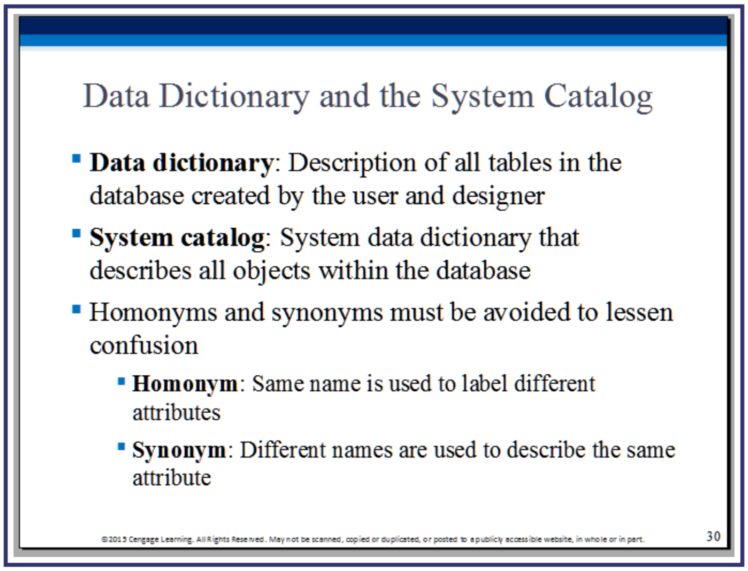
A 'full outer join' is a union of left outer join and right outer join - output all the rows from both tables, including ones for which there are no matches in the other table - this could result in nulls on the left side of some rows, as well as nulls on the right side of others.

DIVIDE



We're dividing by A and B in the divisor (bottom) table. There's (A,5) and (B,5) in the dividend (top) table, so we output 5 as the result; if the dividend were to contain (A,9) and (B,9) also, then we'd output 5 9 as the result.

Dictionaries [hold metadata]



A data dictionary is metadata about tables (only); a system catalog, that includes (is a supeset of, although confusingly, the two are conflated in RL) the data dictionary, and more.