

1/35 11:14:53 \*\*\*



# Relational Modeling

# 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

©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.

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 ( <b>tuple</b> ) 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 <b>attribute domain</b> .
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

©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.

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

© 2013 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

© 2013 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 :)

## 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

© 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.7

'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

## 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

©2013 Cengage Learning. All Rights Reserved. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

8

A table 'cannot not' have entity integrity!!



# Nulls; referential integrity

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

© 2013 Cengage Learning. All Rights Reserved. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

9

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

© 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.

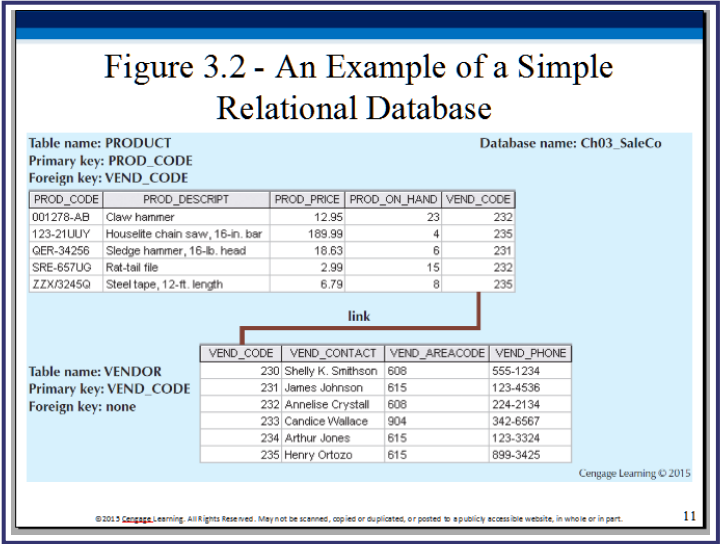
10

# 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



# Nulls - avoid where possible!

## Ways to Handle Nulls

- **Flags:** Special codes used to indicate the absence of some value
- **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

©2013 Cengage Learning. All Rights Reserved. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

15

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]

## 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

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

16

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

## 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

©2013 Cengage Learning. All Rights Reserved. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

17



# SELECT [outputs a subset of rows]

Figure 3.4 - Select

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

**SELECT ALL yields**

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

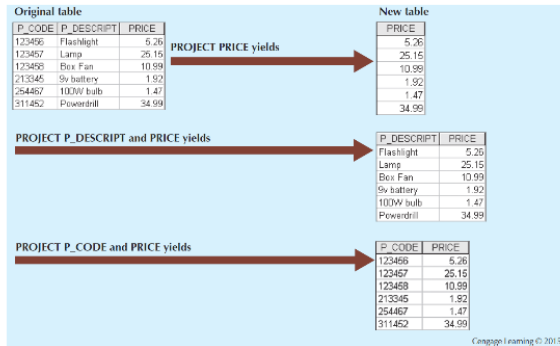
**SELECT only PRICE less than \$2.00 yields**

P_CODE	P_DESCRIPTION	PRICE
311452	Powerdrill	34.99

**SELECT only P\_CODE = 311452 yields**

# PROJECT [outputs a subset of cols]

Figure 3.5 - Project



© 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.

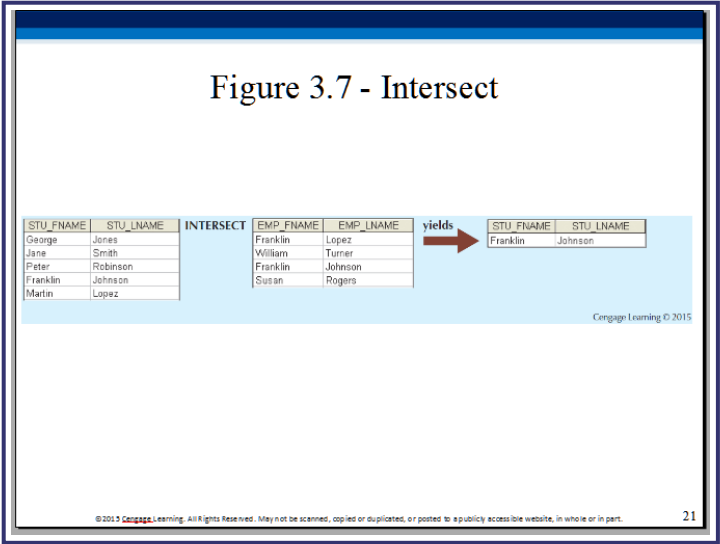
# UNION [eqvt to 'cat a b > c']

Figure 3.6 - Union

P_CODE	P_DESCRIPTION	PRICE	UNION			yields	P_CODE	P_DESCRIPTION	PRICE
123456	Flashlight	5.25	345678	Microwave	150.00	123456	Flashlight	5.25	
123457	Lamp	25.15	345679	Dishwasher	500.00	123457	Lamp	25.15	
123458	Box Fan	10.99	123458	Box Fan	10.99	123458	Box Fan	10.99	
213345	9v battery	1.92				213345	9v battery	1.92	
264467	100W bulb	1.47				264467	100W bulb	1.47	
311452	Powerdrill	34.99				311452	Powerdrill	34.99	
						345678	Microwave	150	
						345679	Dishwasher	500	

Cengage Learning © 2015

# 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

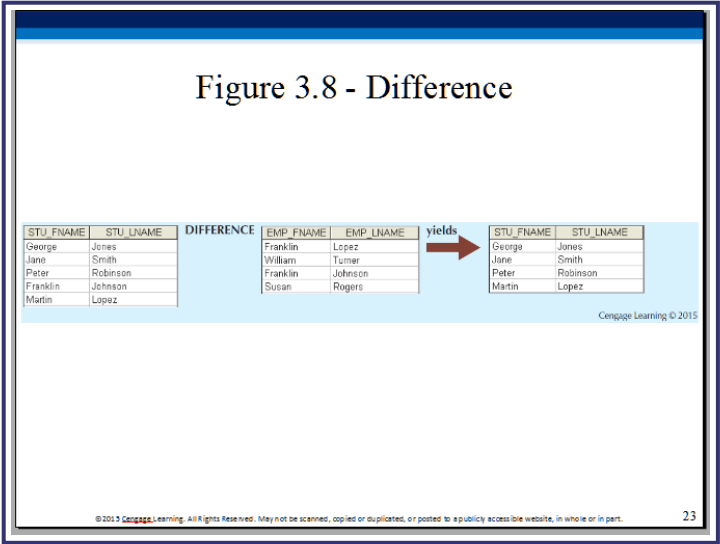
- **Product**

- Yields all possible pairs of rows from two tables

© 2013 Cengage Learning. All Rights Reserved. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

22

# Difference [a - b]



# Product [multiply rows, add columns]

Figure 3.9 - Product

P_CODE	P_DESCRIPTION	PRICE	PRODUCT	STORE	aisle	shelf	yields	P_CODE	P_DESCRIPTION	PRICE	STORE	aisle	shelf
123456	Flashlight	5.26		23	W	5		123456	Flashlight	5.26	23	W	5
123457	Lamp	25.15		24	K	9		123456	Flashlight	5.26	24	K	9
123458	Box Fan	10.99		25	Z	6		123456	Flashlight	5.26	25	Z	6
213345	9v battery	1.92						123457	Lamp	25.15	23	W	5
254467	100W bulb	1.47						123457	Lamp	25.15	24	K	9
311452	Power drill	34.99						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	Power drill	34.99	23	W	5
								311452	Power drill	34.99	24	K	9
								311452	Power drill	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

© 2013 Cengage Learning. All Rights Reserved. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.

25



# 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

© 2013 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

©2013 Cengage Learning. All Rights Reserved. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.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	6152438687
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 (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

Cengage Learning © 2015

# 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

Cengage Learning © 2015

# 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

Cengage Learning © 2015

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
1217752	Adares	32145	125		125		6152439807
1321242	Rodriguez	37134	125		125		6152439807
1312243	Rakowski	34129	167		167		6153426776
1132445	Walker	32145	231		231		6152431124
1657399	Vanloo	32145	231		231		6152431124
1542311	Smithson	37134	421				

Cengage Learning © 2015

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:

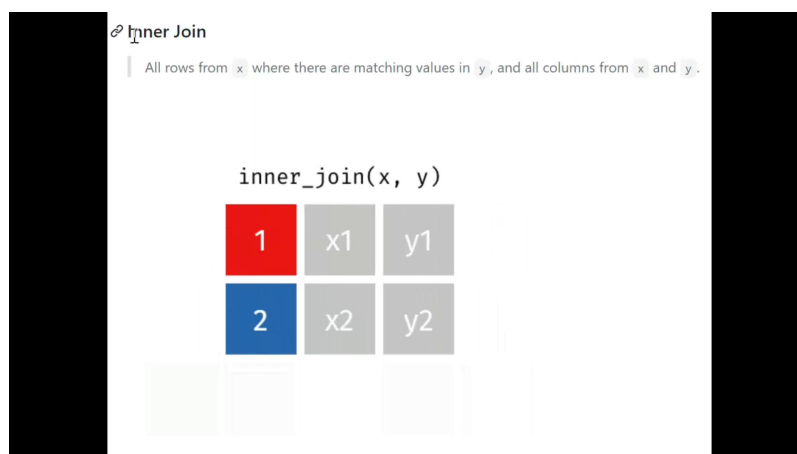
CUS_CODE	CUS_LNAME	CUS_ZIP	CUSTOMER_AGENT_CODE	AGENT_AGENT_CODE	AGENT_PHONE
1217752	Adares	32145	125	125	6152439887
1321242	Rodriguez	37134	125	125	6152439887
1312243	Rakowski	34129	167	167	6152426778
1132445	Walker	32145	231	231	6152431124
1657399	Vanloo	32145	231	231	6152431124
			333		9041234445

Cengage Learning © 2015

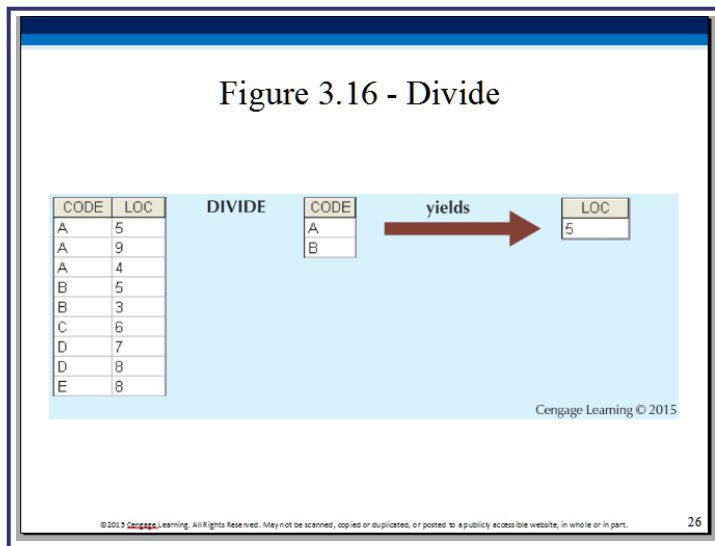
Outer joins are useful in exposing missing information [in our example, customers who don't seem to have an agent, and, agents who don't seem to have customers].

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.

[This clip](#) shows the various types of joins [thanks, Yash Gupta, for sending this]:



# 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]

## 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

© 2013 Cengage Learning. All Rights Reserved. May not be scanned, copied or duplicated, or posted to a publicly accessible website, in whole or in part.30

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