雨课堂
Rain Classroom

本次课程是

# 线上+线下

# 融合式教学

请**现场**的同学们：

1. 打开雨课堂，点击页面右下角喇叭按钮调至静音状态

请**远程上课**的同学们：

1. 打开雨课堂，点击页面右下角喇叭按钮调至静音状态
2. 打开“瞩目”（会议室：182 943 865；密码：见学堂公告），进入会议室，并关闭麦克风

Database Concepts (II)

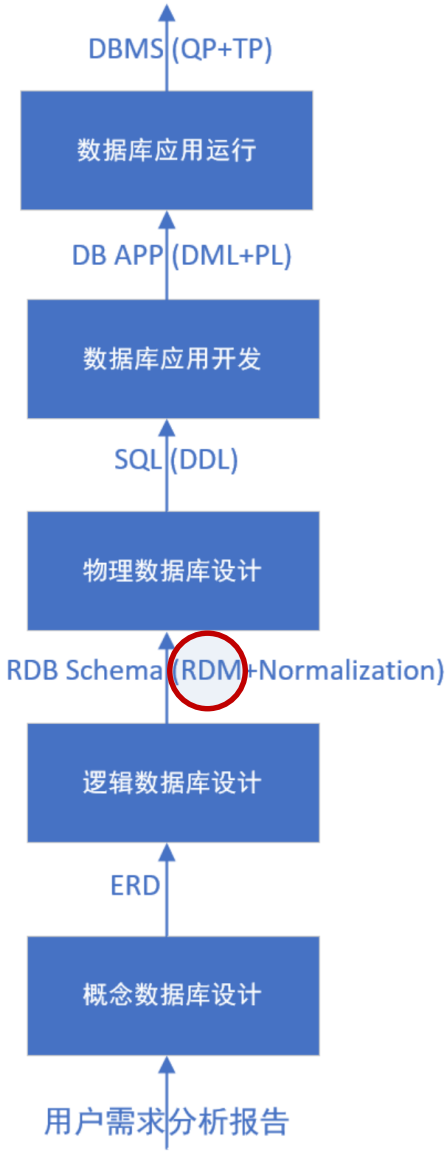
# Data Models


**Chaokun Wang**

School of Software, Tsinghua University

chaokun@tsinghua.edu.cn


March 7, 2022




# Outline

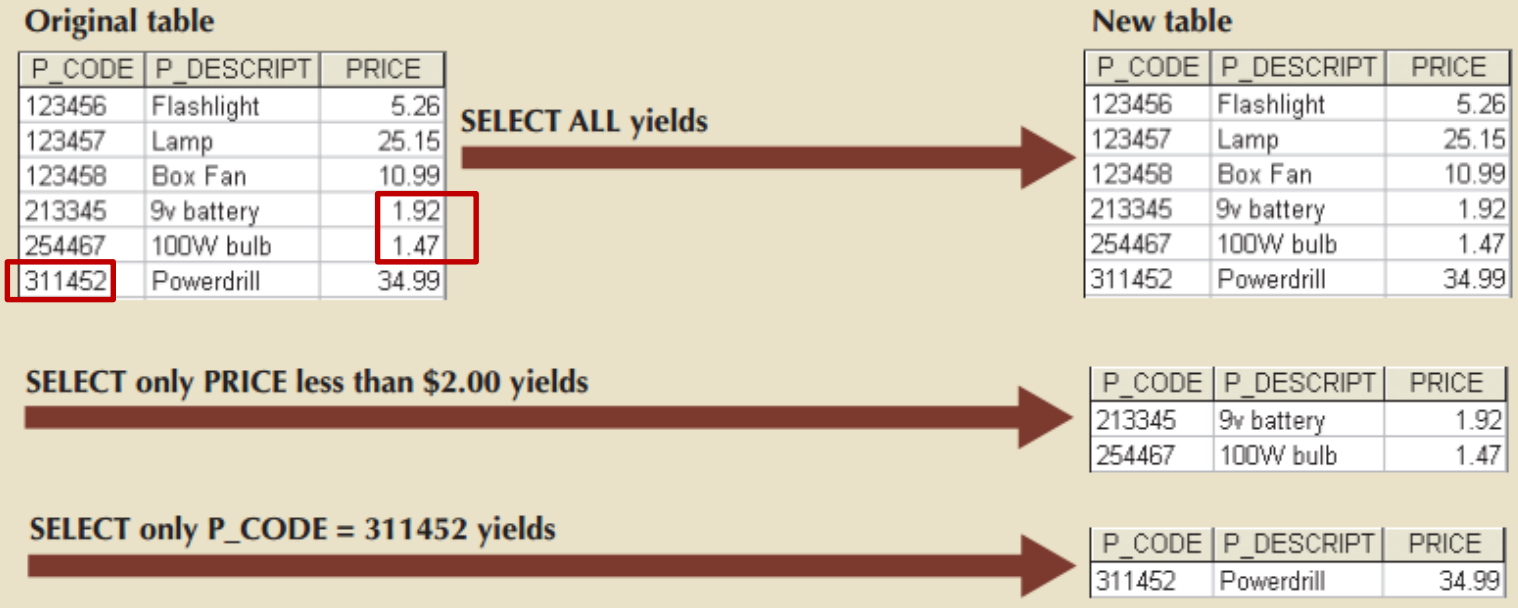
- Introduction to data models
- Relational data model*
- No-SQL data models

# Relational Set Operators (1 of 13)

- Selection (restrict)
  - Unary operator that yields a horizontal subset of a table
- Projection
  - 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
- Intersection
  - Yields only the rows that appear in both tables
  - Tables must be union-compatible to yield valid results

# Relational Set Operators (2 of 13)

FIGURE 3.4 SELECT

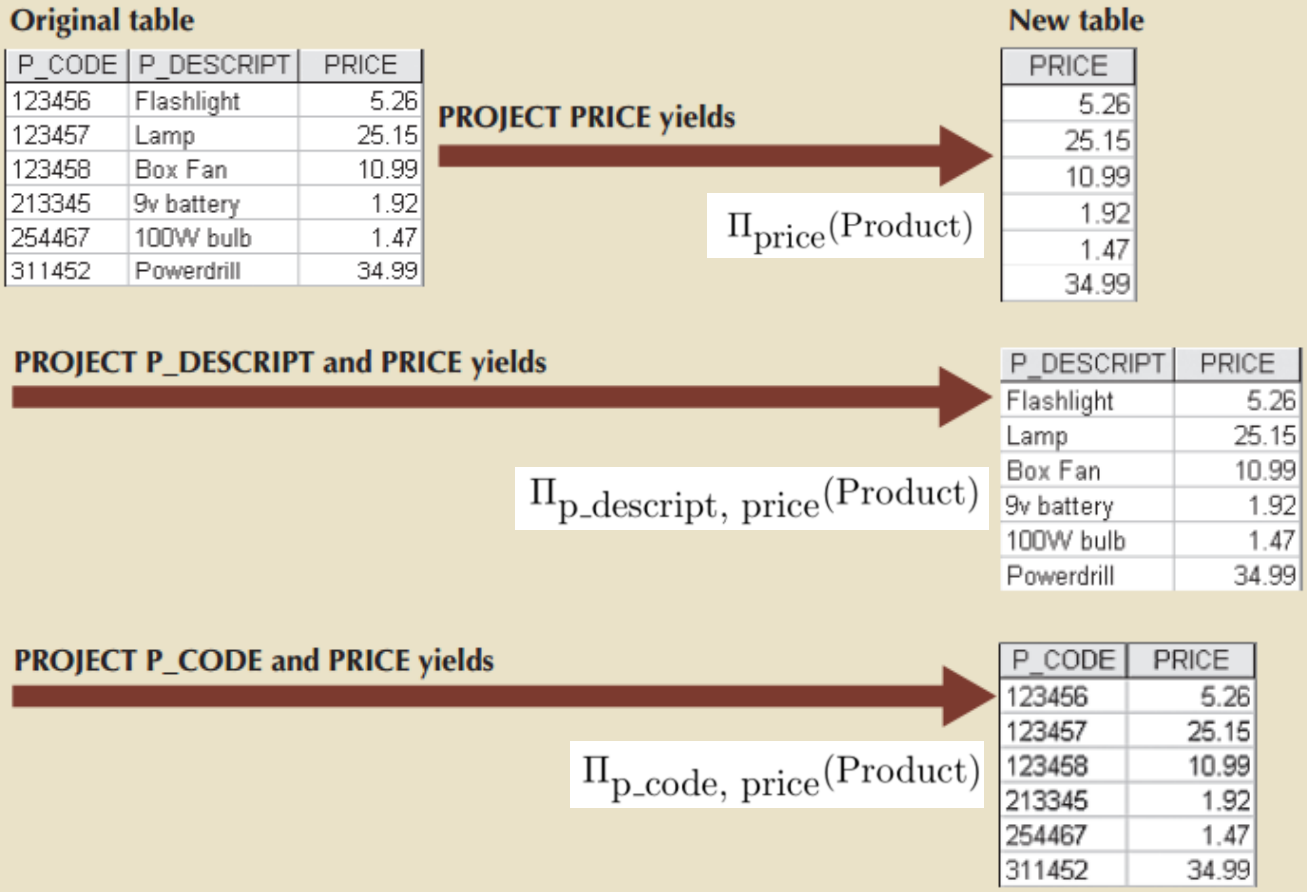



$\sigma_{\text{price}<2}(\text{Product})$ $\sigma_{\text{p\_code}=311452}(\text{Product})$

**$\sigma_{\text{predicate}}(\mathbf{R})$** The Selection operation works on a single relation R and defines a relation that contains only those tuples of R that satisfy the specified condition (*predicate*).

# Relational Set Operators (3 of 13)

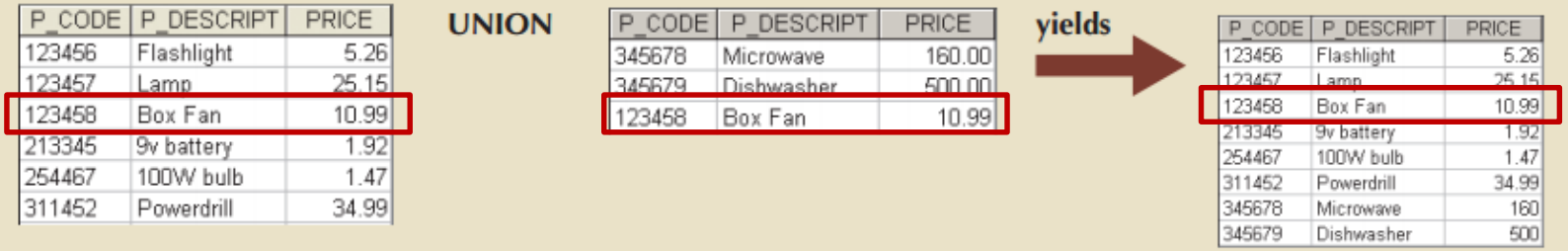
## FIGURE 3.5 PROJECT




**$\Pi_{a_1, \ldots, a_n}(R)$** The Projection operation works on a single relation R and defines a relation that contains a vertical subset of R, extracting the values of specified attributes and eliminating duplicates.

# Relational Set Operators (4 of 13)

## 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 |

$$\Pi_{p\_code,p\_descript,price}(Product1) \cup \Pi_{p\_code,p\_descript,price}(Product2)$$

**R ∪ S** The union of two relations R and S defines a relation that contains all the tuples of R, or S, or both R and S, duplicate tuples being eliminated. R and S must be union-compatible.

## 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 |

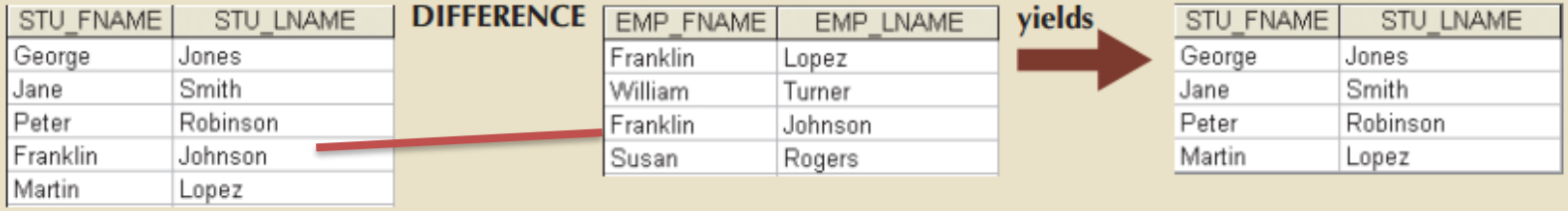
$$\Pi_{stu\_fname,stu\_lname}(STU) \cap \Pi_{emp\_fname,emp\_lnname}(EMP)$$

**R ∩ S** The Intersection operation defines a relation consisting of the set of all tuples that are in both R and S. R and S must be union-compatible.

# Relational Set Operators (5 of 13)

- Set difference
  - Yields all rows in one table that are not found in the other table
  - Tables must be union-compatible to yield valid results

FIGURE 3.8 DIFFERENCE




| STU_FNAME | STU_LNAME |
|---|---|
| George | Jones |
| Jane | Smith |
| Peter | Robinson |
| Franklin | Johnson |
| Martin | Lopez |

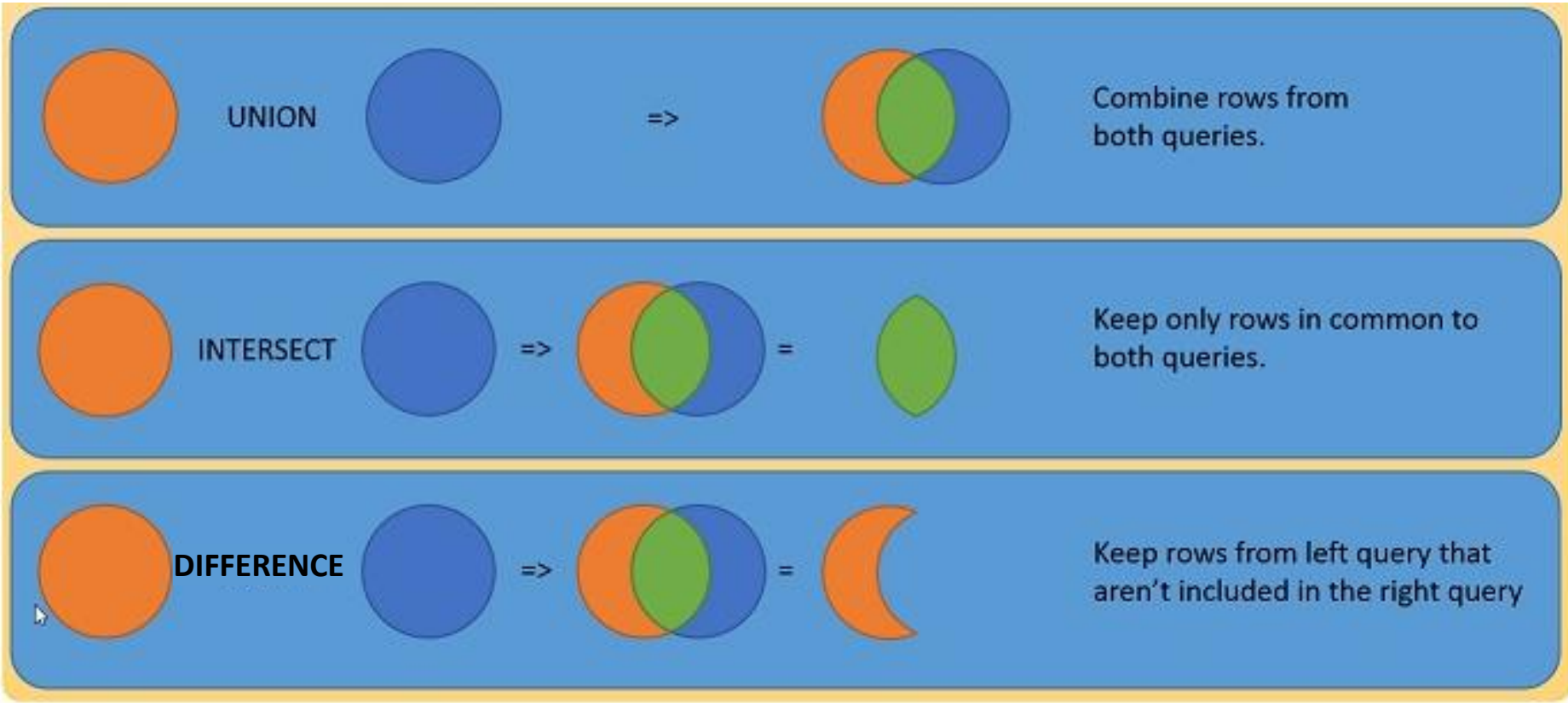
| EMP_FNAME | EMP_LNAME |
|---|---|
| Franklin | Lopez |
| William | Turner |
| Franklin | Johnson |
| Susan | Rogers |

| STU_FNAME | STU_LNAME |
|---|---|
| George | Jones |
| Jane | Smith |
| Peter | Robinson |
| Martin | Lopez |

$$\Pi_{\text{stu\_fname,stu\_lname}}(\text{STU}) - \Pi_{\text{emp\_fname,emp\_lnname}}(\text{EMP})$$

**R – S** The Set difference operation defines a relation consisting of the tuples that are in relation R, but not in S. R and S must be union-compatible.

# $R \cap S = R - (R - S)$ by Venn diagram



# Relational Set Operators (6 of 13)

- Cartesian Product
  - Yields all possible pairs of rows from two tables

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 |

$$\Pi_{\text{p\_code, p\_descript, price}}(\text{Product}) \times \Pi_{\text{store, aisle, shelf}}(\text{Stores})$$

**R × S** The Cartesian product operation defines a relation that is the concatenation of every tuple of relation R with every tuple of relation S.

# Relational Set Operators (7 of 13)

- Joins allow information to be intelligently combined from two or more tables
  - Natural join: links tables by selecting only the rows with common values in their common attribute
  - Equijoin: links tables on the basis of an equality condition that compares specified columns of each table
  - Theta join: links tables using an inequality comparison operator
  - 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

# Relational Set Operators (8 of 13)

## 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 |

| 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 |

Customer ⋈ Agent

**R ⋈ S** The Natural join is an Equijoin of the two relations R and S over all common attributes *x*. One occurrence of each common attribute is eliminated from the result.

# Relational Set Operators (9 of 13)

## 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 |

# Relational Set Operators (10 of 13)

## 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 |

## 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 |

$$\Pi_{\text{cus\_code, cus\_lname, cus\_zip, customer.agent\_code, agent\_code}} (\text{Customer} \bowtie_{\text{customer.agent\_code=agent.agent\_code}} \text{Agent})$$

**$R \bowtie_F S$** The Theta join operation defines a relation that contains tuples satisfying the predicate *F* from the Cartesian product of R and S. The predicate *F* is of the form $R.a_i\ \theta\ S.b_i$ where θ may be one of the comparison operators ($<, \leq, >, \geq, =, \neq$).

# Relational Set Operators (11 of 13)

## 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 | | |

**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 |

Customer ⟕ Agent

**R ⟕ S** The (left) Outer join is a join in which tuples from R that do not have matching values in the common attributes of S are also included in the result relation. Missing values in the second relation are set to null.

## 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 |

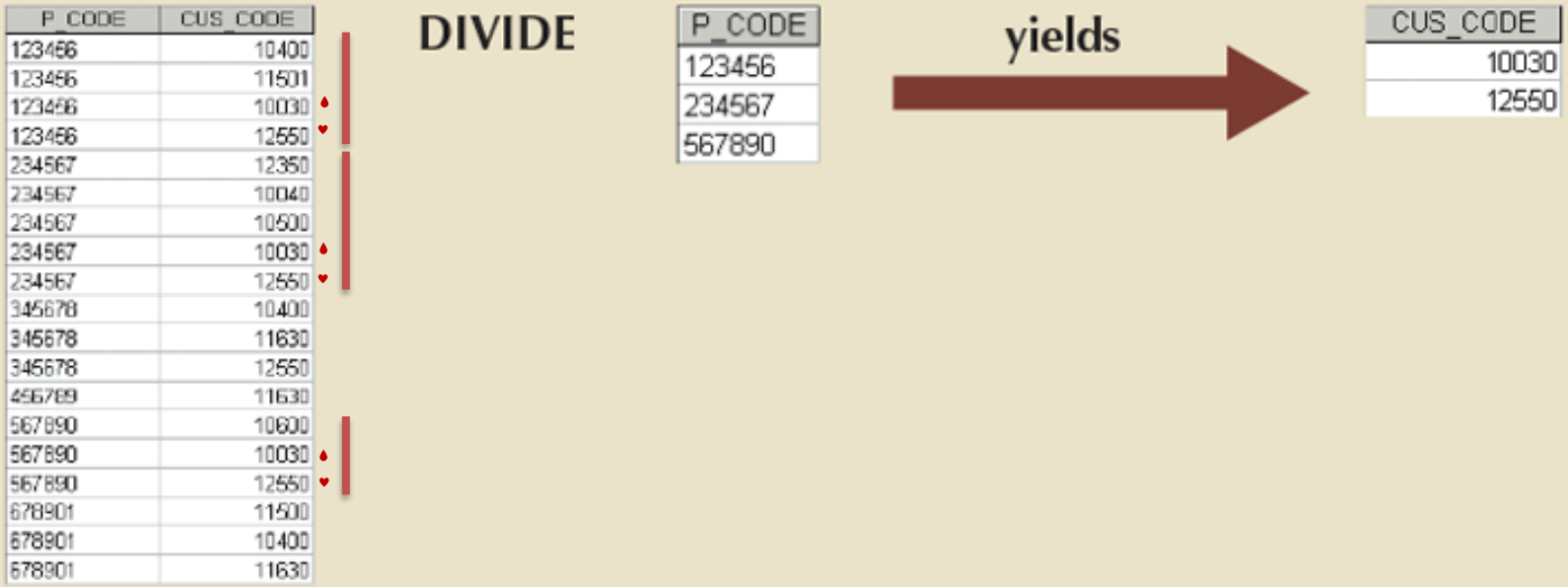
# Relational Set Operators (12 of 13)

- Division
  - Uses one double-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

| | |
|---|---|
| **R ÷ S** | The Division operation defines a relation over the attributes C that consists of the set of tuples from R that match the combination of **every** tuple in S. |

# Relational Set Operators (13 of 13)

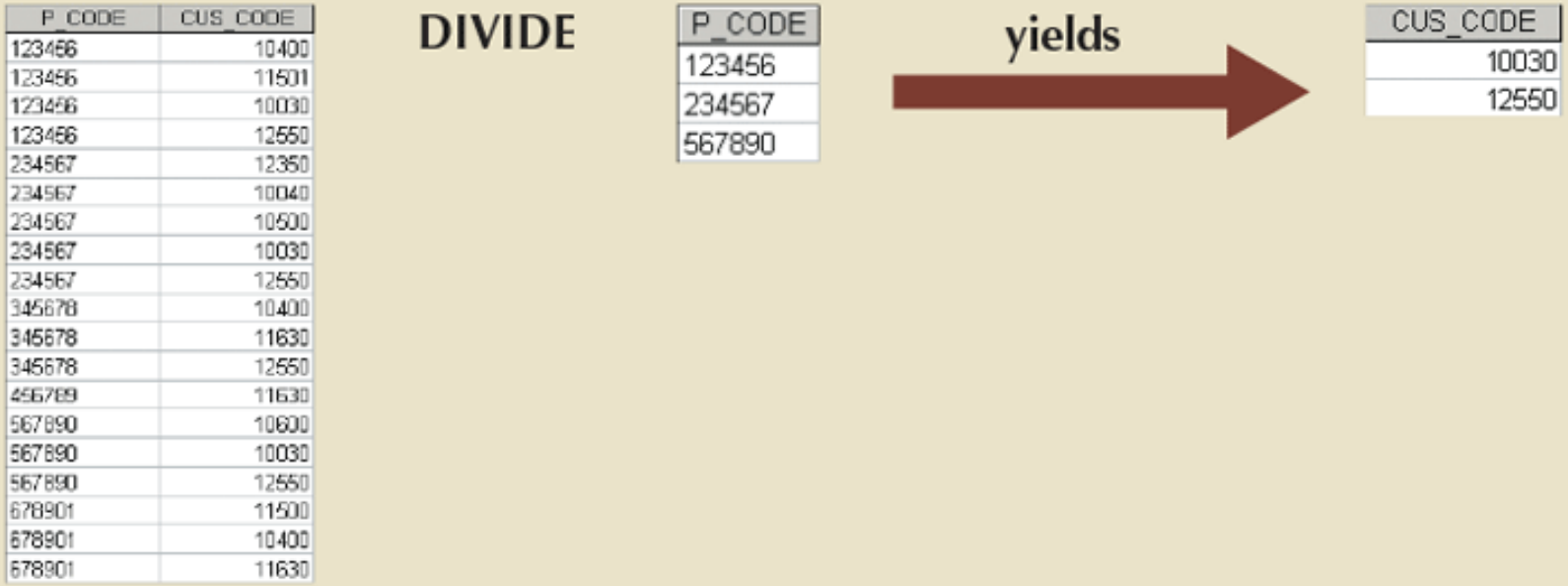
## FIGURE 3.16 DIVIDE




$$(\Pi_{\text{p\_code, cus\_code}}\ (\text{Purchase})) \div (\Pi_{\text{p\_code}}\ (\text{Product}))$$

# How to implement the division operation

## FIGURE 3.16 DIVIDE




$$(\Pi_{\text{p\_code, cus\_code}}\ (\text{Purchase})) \div (\Pi_{\text{p\_code}}\ (\text{Product}))$$

$$T_1 \leftarrow \Pi_{\text{cus\_code}}\ (\text{Purchase})$$

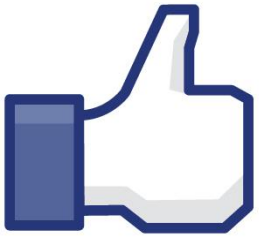
$$T_2 \leftarrow \Pi_{\text{cus\_code}}\ (\ (T_1 \times \Pi_{\text{p\_code}}\ (\text{Product})\ ) - (\Pi_{\text{cus\_code,p\_code}}\ (\text{Purchase})\ )\ )$$

$$T \leftarrow T_1 - T_2$$

# Data Dictionary and the System Catalog

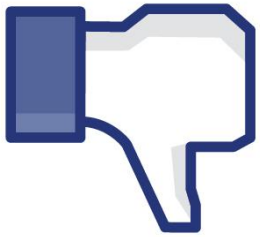
- Data dictionary
  - Description of all tables in the database created by the user and designer
- System catalog
  - System data dictionary that describes all objects within the database
- Homonyms and synonyms must be avoided to lessen confusion
  - Homonym: same name is used to label different attributes
  - Synonym: different names are used to describe the same attribute

# RDBMS: Pros & Cons



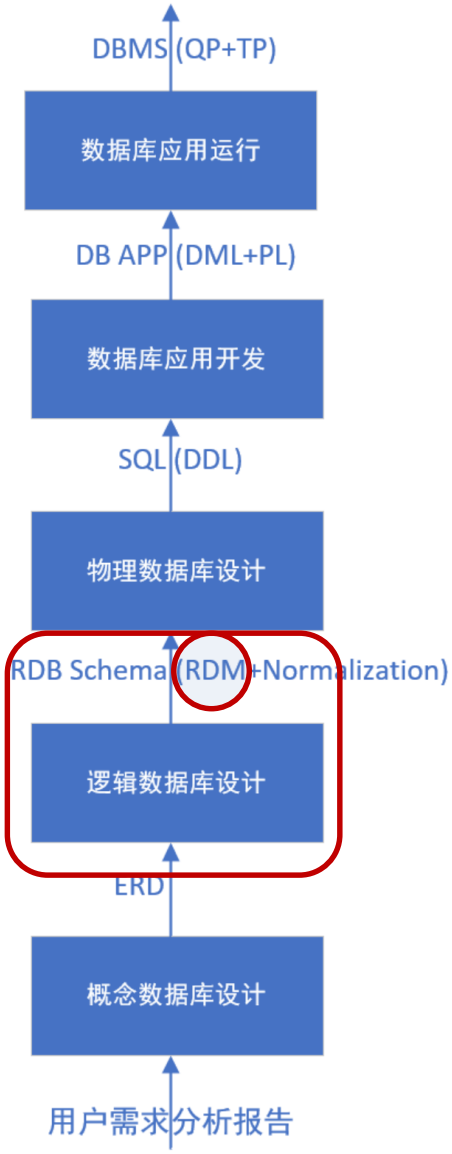
## Pros

- Strong theoretical foundation
- Declarative syntax
- Standard data access language through SQL
- …



## Cons

- Scalability
- Data has to fit into tables
- …




# Outline

- Introduction to data models
- Relational data model*
- No-SQL data models

# Big Data

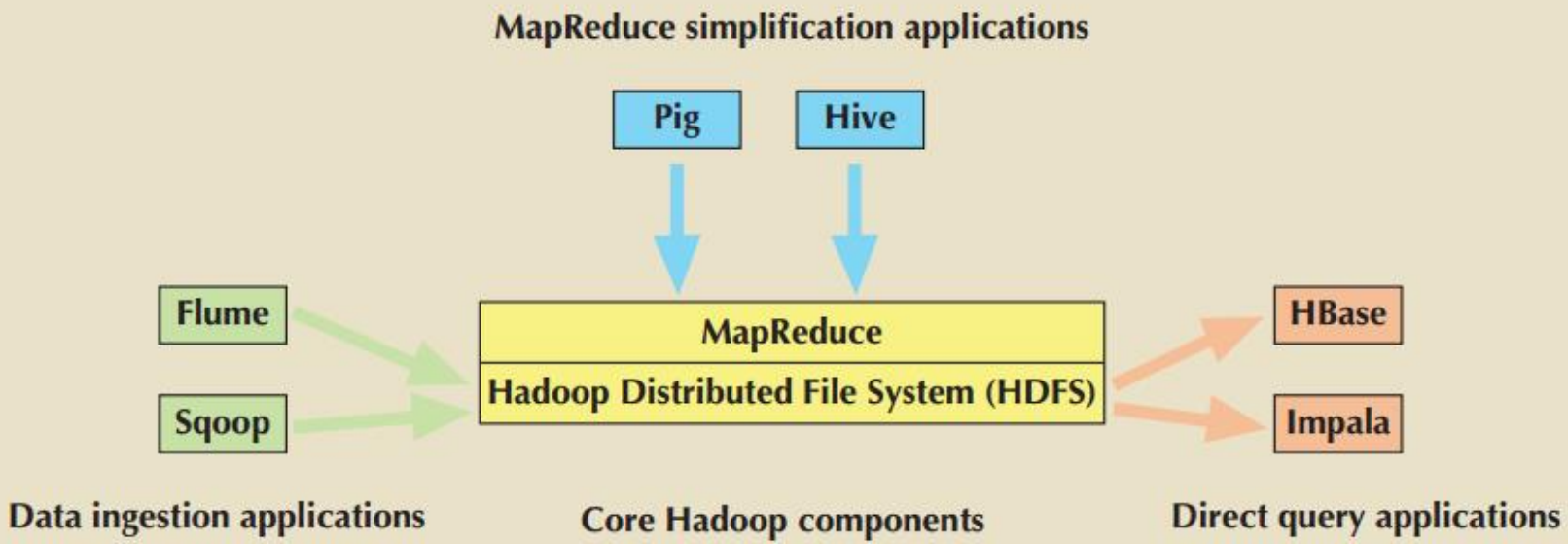
- Other characteristics beyond 3Vs
  - Value: degree data can be analyzed for meaningful insight
  - Veracity: trustworthiness of data
  - Variability: changes in meaning of data based on context
  - Sentimental analysis: attempts to determine if a statement conveys a positive, negative, or neutral attitude about a topic
  - Visualization: ability to graphically resent data to make it understandable

# Hadoop

- De facto standard for most Big Data storage and processing
  - Java-based framework for distributing and processing very large data sets across clusters of computers
- Important components
  - Hadoop Distributed File System (HDFS): low-level distributed file processing system that can be used directly for data storage
  - MapReduce: programming model that supports processing large data sets

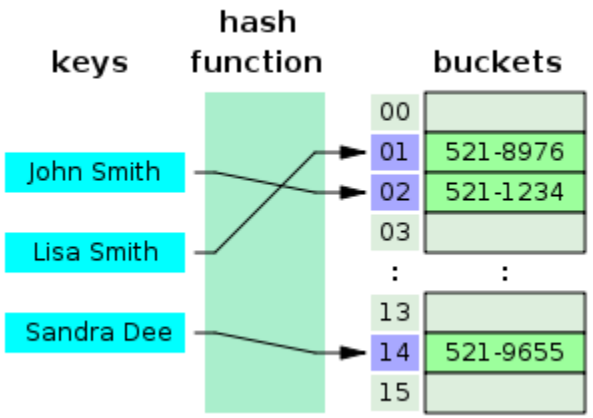
# Hadoop Ecosystem

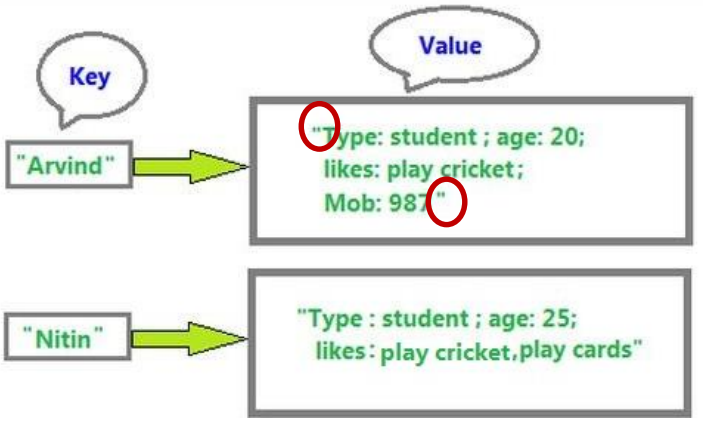
FIGURE 14.6 A SAMPLE OF THE HADOOP ECOSYSTEM

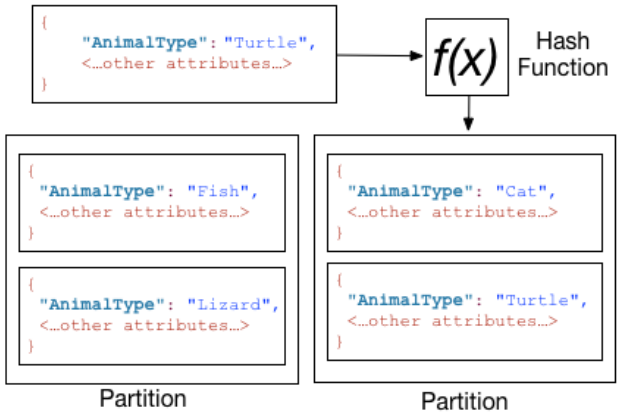


# No SQL—Not Only SQL

- Key-value DBMS
  - Schema free
  - High Scalability




redis

amazon DynamoDB

# No SQL—Not Only SQL

- Document-oriented DBMS
  - Self-describing
  - Hierarchical tree data structure
  - Documents have differences in their attributes
    - But belong to same collection

```
<Books>
    <Book ISBN="0553212419">
        <title>Sherlock Holmes: Complete Novels...
        <author>Sir Arthur Conan Doyle</author>
    </Book>
    <Book ISBN="0743273567">
        <title>The Great Gatsby</title>
        <author>F. Scott Fitzgerald</author>
    </Book>
    <Book ISBN="0684826976">
        <title>Undaunted Courage</title>
        <author>Stephen E. Ambrose</author>
    </Book>
    <Book ISBN="0743203178">
        <title>Nothing Like It In the World</title>
        <author>Stephen E. Ambrose</author>
    </Book>
</Books>
```

Customer Document

```
"customer" =
{
   "id": "Customer:1",
   "firstName": "John",
   "lastName": "Wick",
   "age: 25,
   "address": {
        "country": "US",
        "city": "New York",
        "state": "NY"
        "street": "21 2nd Street",
      },
   "hobbies": [ Football, Hiking ],
   "phoneNumbers": [
      {
        "type": "Home",
        "number": "212 555-1234"
      },
      {
        "type": "Office",
        "number": "616 565-6789"
      }
    ]
}
```

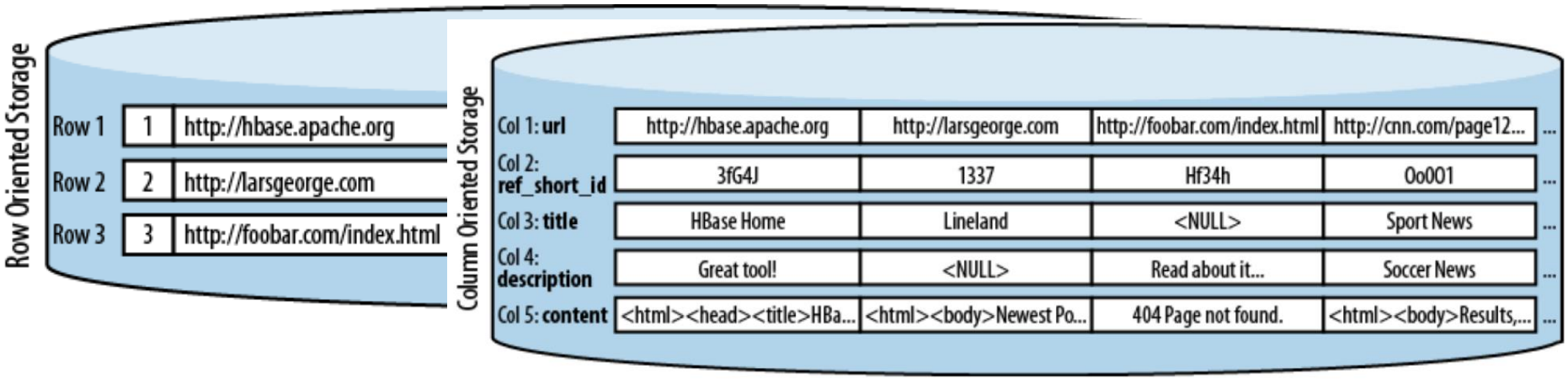
mongoDB

# No SQL—Not Only SQL

- Column-oriented DBMS

SQL Schema

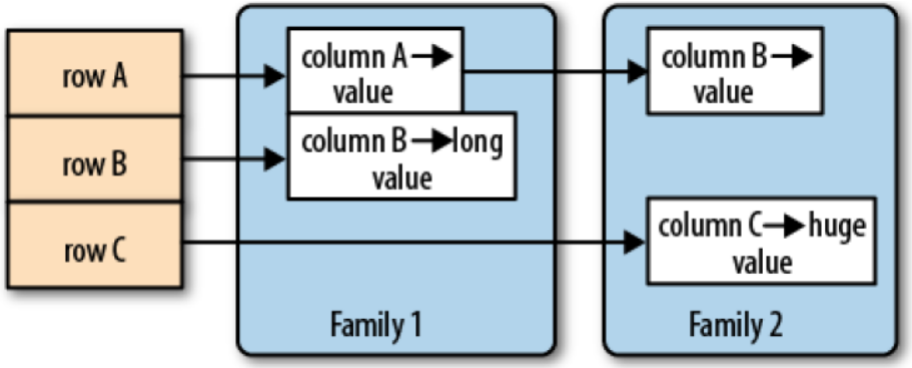
| URLS | | | | | |
|---|---|---|---|---|---|
| **url_id** INTEGER PK | **url** VARCHAR(4096) | **ref_short_id** CHAR(8) | **title** VARCHAR(200) | **description** VARCHAR(400) | **content** TEXT |
| 1 | http://hbase.apache.org | 3fG4J | HBase Home | Great tool! | <html><head><title>HBase Home</ti... |
| 2 | http://larsgeorge.com | 1337 | Lineland | <NULL> | <html><body>Newest Posts... |
| 3 | http://foobar.com/index.html | Hf34h | <NULL> | Read about it... | 404 Page not found. |
| 4 | http://cnn.com/page123.html | 0o001 | Sport News | Soccer News | <html><body>Results, Reviews, ... |

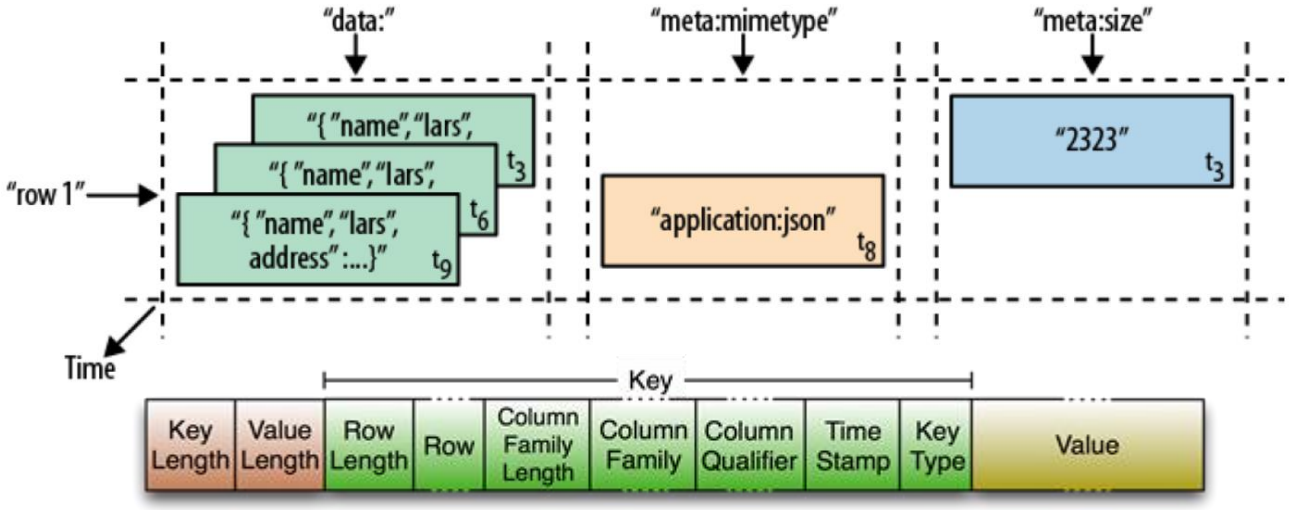


# No SQL—Not Only SQL

- Column-oriented DBMS

| | column A (int) | column B (varchar) | column C (boolean) | column D (date) |
|---|---|---|---|---|
| row A | | | | |
| row B | | | | |
| row C | | | NULL? | |
| row D | | | | |



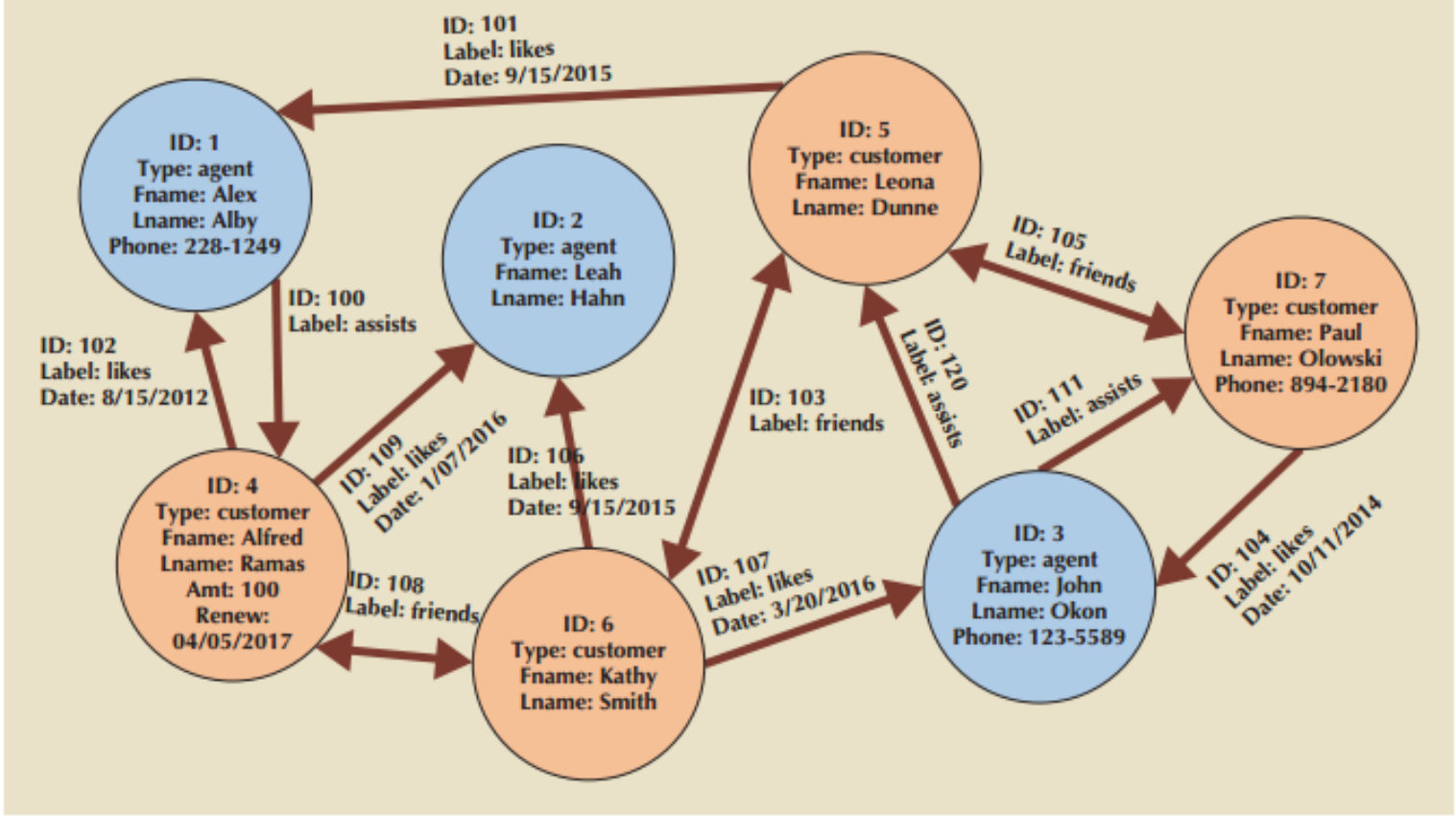

cassandra

APACHE HBASE

# No SQL—Not Only SQL

- Graph DBMS

FIGURE 14.11 GRAPH DATABASE REPRESENTATION

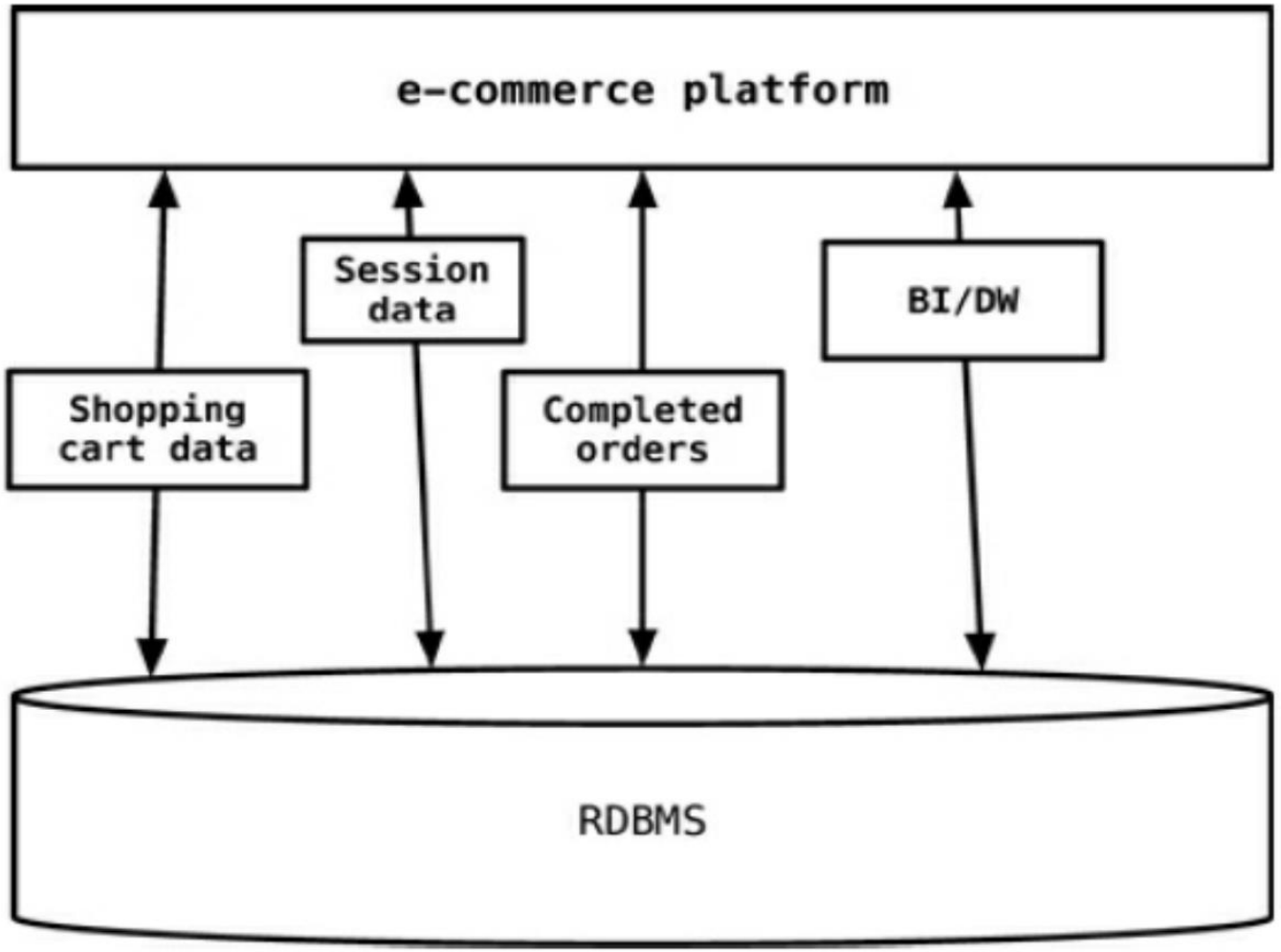


# NewSQL Databases

- NewSQL databases support:
  - SQL as the primary interface
  - ACID-compliant transactions
- Similar to NoSQL, NewSQL databases also support:
  - Highly distributed clusters
  - Key-value or column-oriented data stores
- Latest technologies to address Big Data problems
  - Have been adopted by relatively few organizations

# Discuss: Use of RDBMS for every aspect of storage?

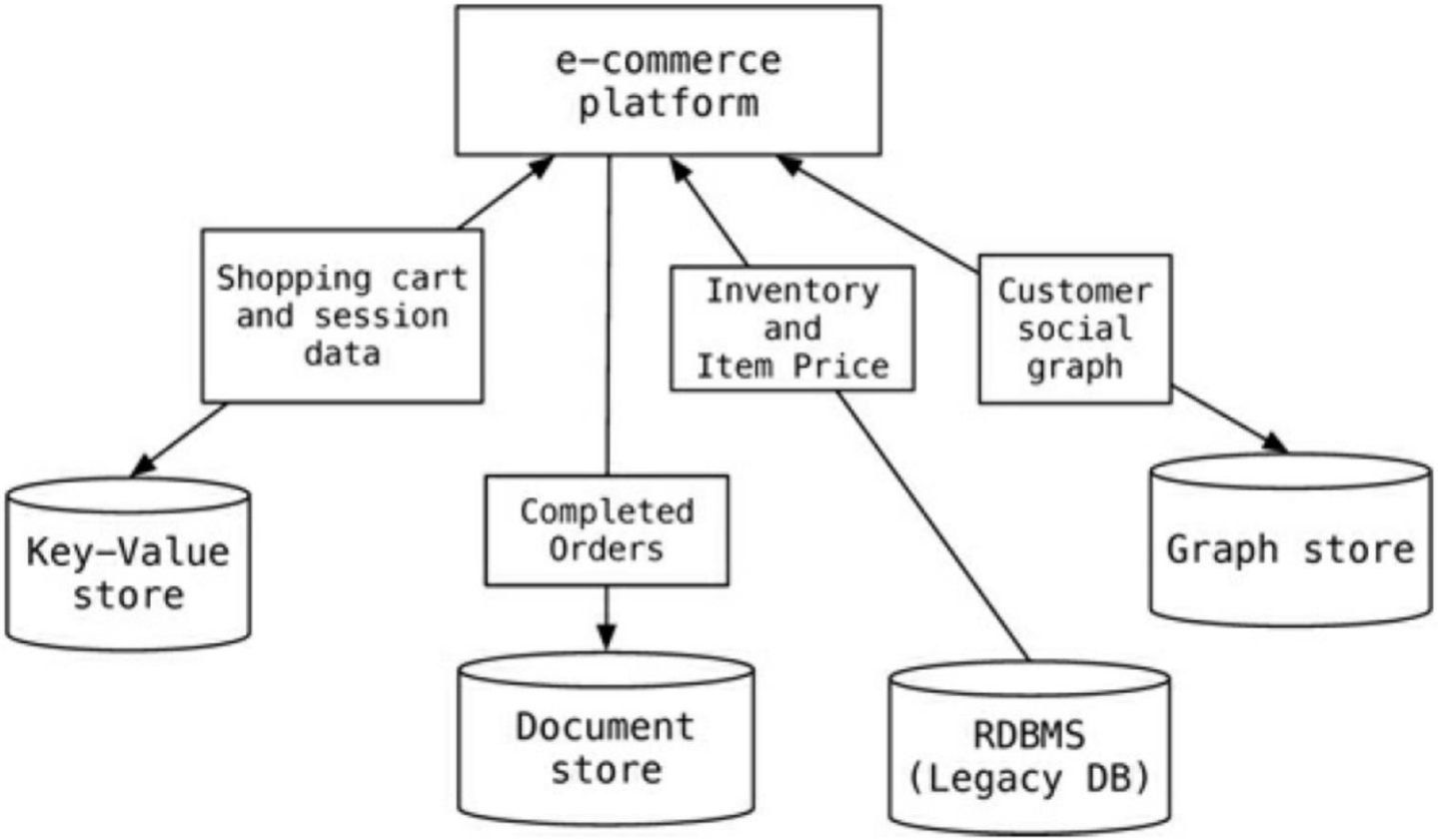
Use of RDBMS for every aspect of storage for the application



# Discuss: Polyglot persistence

Example implementation of Polyglot persistence

coexistence of a variety of data storage and management technologies within an organization's infrastructure



# Conclusions

- Relational data model
  - Data Structures
  - Constraints
  - Operations (Relational Algebra)
- Hadoop
  - HDFS, MapReduce
- No-SQL data models
  - Key-Value
  - Column-Oriented
  - Document
  - Graph

# Homework

- Read Chapter 3 (3.4--3.5) of DS1.
- Assignment
  - Later in Yuketang
- Further Reading
  - Chapter 14 of DS1.
  - Chapter 5.1 of DS2.
  - Chapter 13 (Polyglot Persistence) in "NoSQL Distilled".

# Thank you!