The Relational Data Model

Instructor: Shel Finkelstein

Reference:
A First Course in Database Systems,
3rd edition, Chapter 2.1, 2.2

Important Notices

- You should have Gradiance access this week, via Labs with Vassilis and Ashutosh.
  - First Gradiance Assignment will be posted soon.

- Lab1 assignment will be posted by Friday, Jan 13 on Piazza under Resources. General Information about Labs has already been posted. See Piazza announcement.
  - Lab1 will be discussed at next week's Labs.
  - Due **Sunday, January 22**, by 11:59pm.
  - Your solution should be submitted via Canvas as zip file.
  - Canvas will be used for both Lab submission and grading.

Accessing Canvas

LOGIN INFORMATION FOR CANVAS FALL COURSES

1. Go to http://login.uconline.edu/
2. Click on UC Campus Student and Faculty > UC Santa Cruz.
3. Log in with your username and Gold password.
4. You will see your Dashboard. Click on the course you want.

Please ensure that you have access, although there's nothing there for CMPS 180 yet.

First, Let's Answer These Questions
Practice Homework 0

Not a homework, but for CMPS180 students, these should be easy.

0-If set S is {1,3,5,7} and set T is {2,3,5,7}, what are S UNION T and S INTERSECT T?

1-If set A is {1,2,3} and set B is {u,v,w,x,y}, how many ways can you pick pairs of items, with the first from A and the second from B?

2-If you have a set of employees (with names and salaries) where John makes 10K, George makes 20K, Ringo makes 30K and Paul makes 40K, what are the names of the employee(s) who make less than the average salary?

3-Can there ever be an employee who makes more than every employee? If so, give an example. If not, explain why not

4-Write the truth-table for p AND q, where p can be TRUE or FALSE and q can be TRUE or FALSE.

What is a Data Model?

- A *data model* is a mathematical formalism that consists of:
  - Structure of the data
  - Operations on the data
  - Constraints on the data

- We have mentioned:
  - Network data model, Hierarchical data model, Relational data model, XML data model, JSON data model.

What is the Relational Data Model?

- The relational data model (Edgar F. Codd, 1970)
  - Data is described and represented by the mathematical concept of a *relation*

- What is a relation?
  - A structure with rows (tuples) and columns (attributes, fields)
  - Textbook uses “attribute” to mean the name of a column

- Codd defined relations as sets, with no duplicates
  - Stored relations (tables) typically don’t have duplicates (unique keys)
  - … but SQL allows duplicates during processing and in results
    - Why? We’ll see later. (Any ideas?)

What is the Relational Data Model? (cont'd)

- The relational data model (Edgar F. Codd, 1970)
  - Data is described and represented by the mathematical concept of a *relation*.

- What is a relation?
  - A structure with rows and columns
  - A subset of a Cartesian product of sets
  - What is the Cartesian product of {a,b,c,d} and {1,2,3}?

Cartesian Product

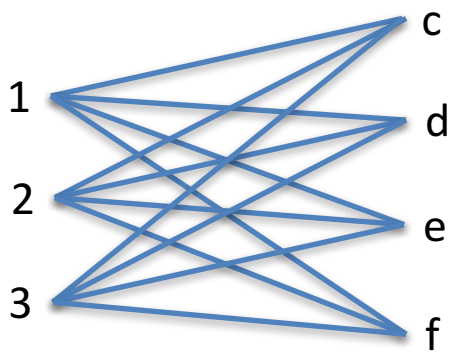
- What is the Cartesian product of {a,b,c,d} and {1,2,3}?

  { (a,1), (a,2), (a,3),
  (b,1), (b,2), (b,3),
  (c,1), (c,2), (c,3),
  (d,1), (d,2), (d,3) }

- What are some examples of relations from that Cartesian product?

Another Cartesian Product Example

- A: {1,2,3}
- B: {d,e,f,g}
- A × B = { (1,d), (1,e), (1,f), (1,g),
  (2,d), (2,e), (2,f), (2,g),
  (3,d), (3,e), (3,f), (3,g) }



- Suppose that C = {x,y}. What would A x B x C be?

Tuples and Relations

- *Tuple*:
  - A *k-tuple* is an ordered sequence of k values (not necessarily different)
    - (1,2) is a binary tuple or 2-tuple
    - (a,b,b) is a ternary tuple or 3-tuple
    - (112, “Ann”, “CS”, “F”, 3.95) is a 5-tuple

- If D_1, D_2, …, D_k are sets of elements, then the *Cartesian product* $D_1 \times D_2 \times \ldots \times D_k$ is the set of all k-tuples $(d_1, d_2, \ldots, d_k)$ such that $d_i \in D_i$, for all i with $1 \leq i \leq k$.

- *Relation*:
  - A *k-ary relation* is a subset of $D_1 \times D_2 \times \ldots \times D_k$, where each D_i is a set of elements
  - D_i is the *domain (or datatype)* of the *i*th column of the relation
  - Domains may be enumerated {“AMS”, “CMPS”, “TM”} or may be of standard types (integer, float, date)

A Few Examples of Relations

- New York Stock Exchange Listings:
  - https://www.nyse.com/listings_directory/stock
- Presidential info:
  - http://politicsandprosperity.com/facts-about-presidents/
- NFL Stadium info:
  - http://en.wikipedia.org/wiki/List_of_current_National_Football_League_stadiums
- National Basketball Association Standings
  - http://www.nba.com/standings/team_record_comparison/conferenceNew_Std_Div.html

Another Practice Homework

(Not collected, will be discussed during next lecture)

- If D_1 has n_1 elements and D_2 has n_2 elements, then how many elements are there in $D_1 \times D_2$?
  - That is, if $|D_1| = n_1$, $|D_2| = n_2$, what is $|D_1 \times D_2|$?

- If D_i has n_i elements, then how many elements are there in the Cartesian product $D_1 \times \ldots \times D_k$?

- If D_i has n_i elements, then how many relations can one construct from $D_1 \times \ldots \times D_k$?

Attributes and Relation Schema

- An *attribute* is the name of a column in a relation.
  - E.g., studentID, name, major, gender, avgGPA
- A *relation schema* R is a set $\{A_1, \ldots, A_k\}$ of attributes, often written as $R(A_1, \ldots, A_k)$, where A_i is the name of the *i*th column of the relation.
  - The datatype/domain of each attribute is of some elementary type, such as integer, string or an enumerated type, not a structure, array, list, sequence or other compound structure: "First normal form"
  - E.g., Student(studentID:int, name:str, major:str, gender:gender, avgGPA:double)
  - … or simply, Student(studentID, name, major, gender, avgGPA)

    when types are implicit

First Normal Form (1NF)

- Domains in relational database were atomic. That atomicity means that relational databases satisfy what Ted Codd called "First Normal Form" 1NF.
  - Major has to be a single value, not a list of values
  - Address has to be a single value, not a structure
- Relational database is structured, which makes many things simple
  - Semi-structured and unstructured data (e.g., with XML, JSON or Protocol Buffers) doesn't satisfy 1NF.
  - Relational databases now also can include semi-structured data, not just structured data.
    - But we won't get to that until late in the term.

Relation Schema and Instance of a Relation Schema

- An *instance of a relation schema* is a relation that conforms to the relation schema.
  - E.g., Student(studentID:int, name:str, major:str, gender:gender, avgGPA:double)
    - Every tuple in the relation must be a 5-tuple.
    - The i^{th} component of each tuple in the relation must have the corresponding type (correct domain)

{ (112, “Ann”, “CS”, “F”, 3.95),
(327, “Bob”, “CS”, “M”, 3.90),
(835, “Carl”, “Physics”, “M”, 4.00) }

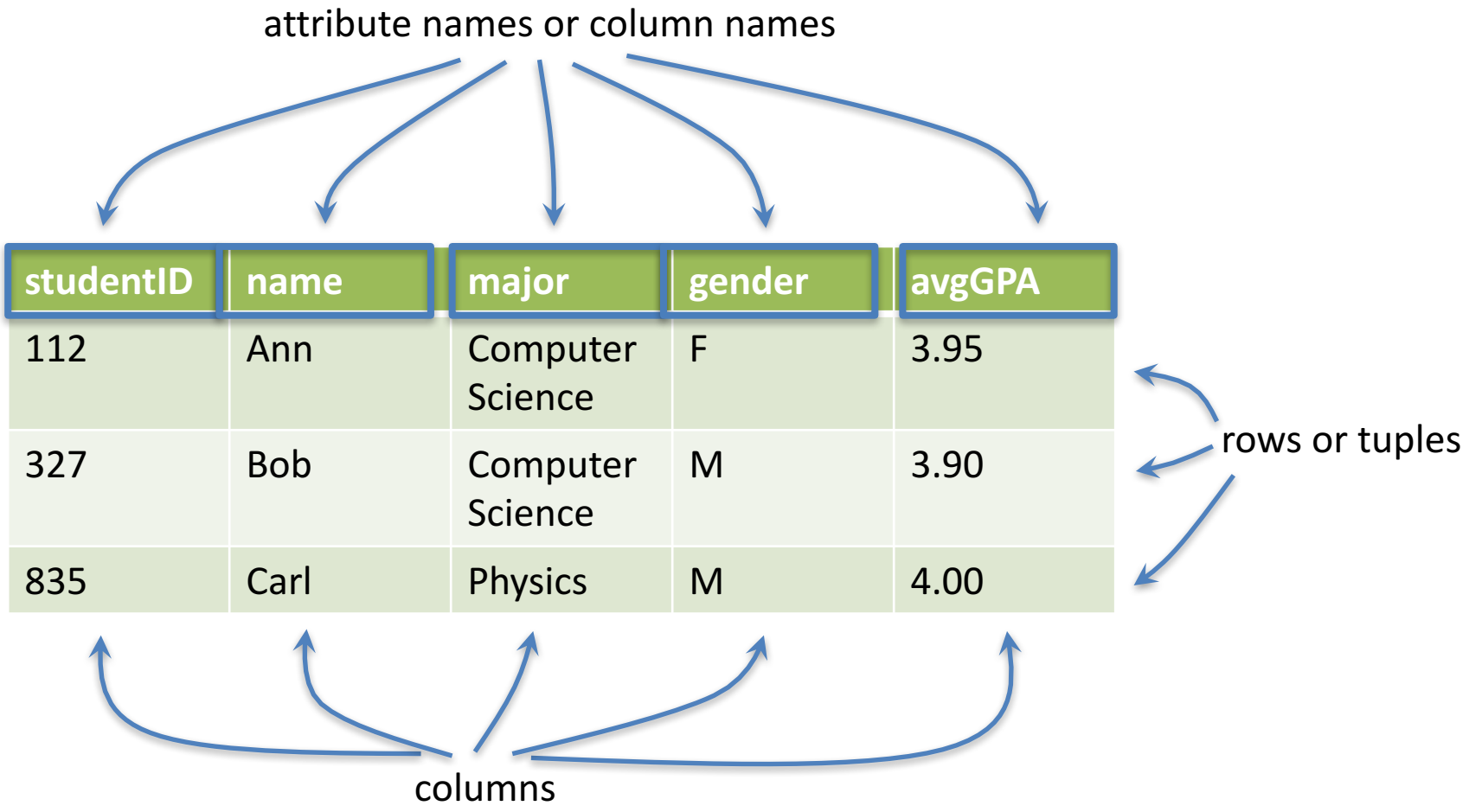
✔

{ (“112”, “Ann”, “CS”, “F”, 3.95),
(“327”, “Bob”, “CS”, “M”, 3.90),
(“835”, “Carl”, “Physics”, “M”, 4.00) }

✗

An Instance of the Student Relation

studentID	name	major	gender	avgGPA
112	Ann	Computer Science	F	3.95
327	Bob	Computer Science	M	3.90
835	Carl	Physics	M	4.00



3 rows, 5 columns. The relation has *arity* 5.

A Relation Database Schema

- A *relation database schema* or, simply, a *database schema* is a set of relation schemas with disjoint relation names.
  - Informally, it's a bunch of different relations.

- A university database schema:
  - Student(studentID, name, major, gender, avgGPA)
  - Course(courseID, description, department)
  - Teach(profID, courseID, quarter, year)
  - Enroll(studentID, courseID, grade)
  - Professor(profID, name, department, level)

Database Schema versus Relation Schema

- In a Relational DBMS like PostgreSQL, the term “Schema” refers to a collections of Relations/Tables that are in a named Database Schema.
  - You’ll be using Schemas (with this meaning) in Labs.
  - Databases may have more than one Database Schema, each with its own name.
  - You’re typically working with some current schema, but you can always qualify a table it with a schema name ahead of it to refer to a table in another schema, e.g., differentschema.employees
- In our textbook, schema usually refers to a Relation Schema, the schema of a named Relation with attributes that have specific data types.
  - In lectures, we’ll almost always use “Schema” with this meaning.
- It should usually be clear which meaning of Schema is being used.
  - If you’re not sure, please ask!

Instance of a Database Schema

- An *instance of a database schema* $\{R_1, \ldots R_k\}$ (or a *database instance* in short) is a set $\{r_1, \ldots, r_k\}$ of relations such that r_i is an instance of R_i, for $1 \leq i \leq k$.

Student

studentID	name	major	gender	avgGPA
112	Ann	Computer Science	F	3.95
327	Bob	Computer Science	M	3.90
835	Carl	Physics	M	4.00

Course

courseID	description	department
CMPS101	Algorithms	CS
BINF223	Intro. To Bio	Biology

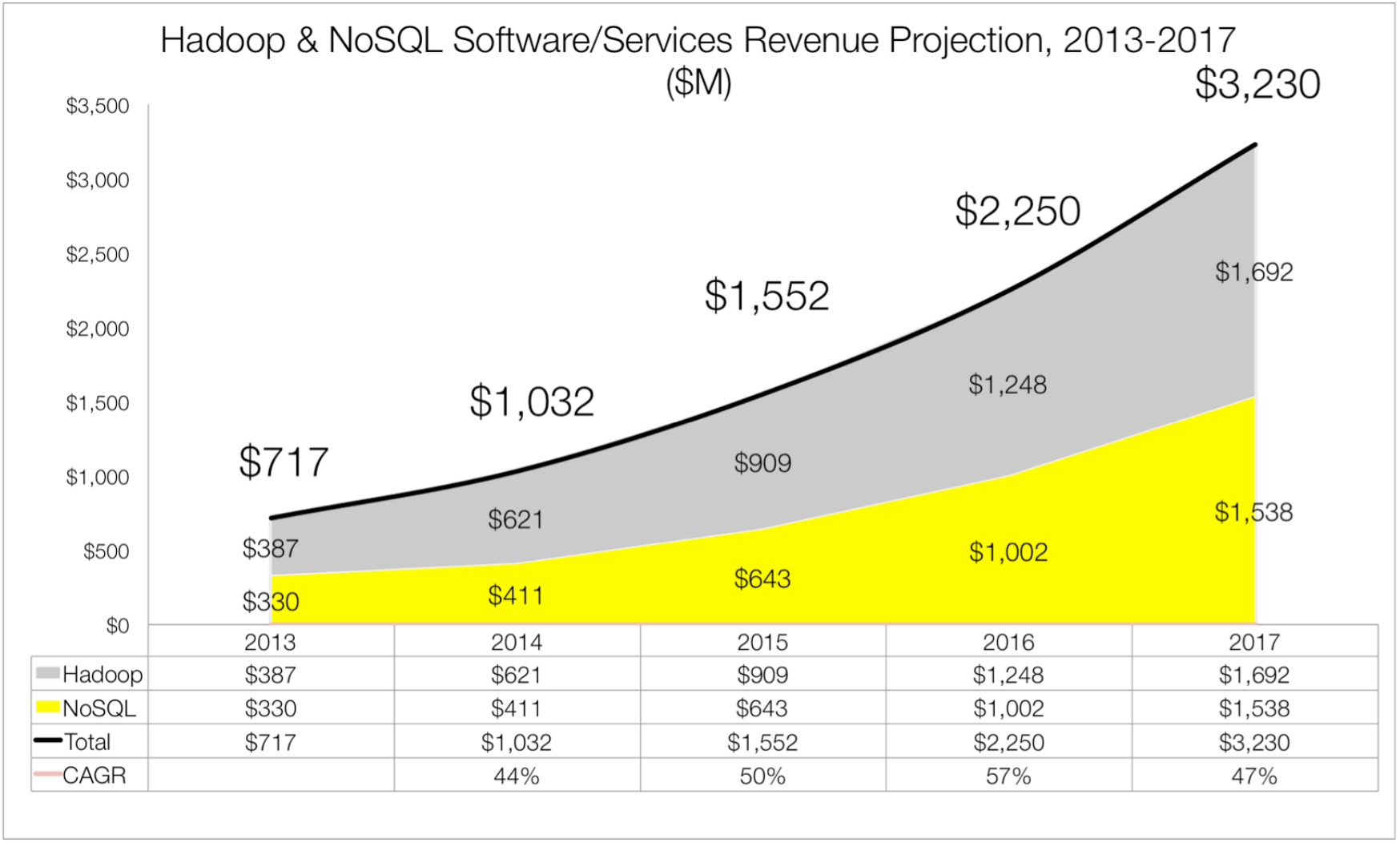
Also: Teach, Enroll, Professor, …

Major Database Vendors (2012)

(See Quadrant for Operational Database Management Systems" from Gartner for info on current vendors)

Relational Database Management Systems (RDBMS) Vendors					
Total Software Revenue, Worldwide, 2010-2011 (Millions of U.S. Dollars)					
Vendor	2010	2011	Share of 2010	Share of 2011	Growth 2011
Oracle	9,990.5	11,787.0	48.2%	48.8%	18.0%
IBM	4,300.4	4,870.4	20.7%	20.2%	13.3%
Microsoft	3,641.2	4,098.9	17.6%	17.0%	12.6%
SAP/Sybase	744.4	1,101.1	3.6%	4.6%	47.9%
Teradata	754.7	882.3	3.6%	3.7%	16.9%
Other Vendors	1,315.3	1,389.7	6.3%	5.8%	5.7%
Grand Total	20,746.6	24,129.5	100.0%	100.0%	16.3%
Source: Gartner (March 2012)					

Wikibon 2014 Prediction



	2013	2014	2015	2016	2017
Hadoop	$387	$621	$909	$1,248	$1,692
NoSQL	$330	$411	$643	$1,002	$1,538
Total	$717	$1,032	$1,552	$2,250	$3,230
CAGR		44%	50%	57%	47%

Keys

- A *key constraint* (or a *key* in short) of a relation schema R is a subset K of the attributes of R such that the following both hold:
  1. For every instance r of R, every two distinct tuples of r must differ in their values from K.
     - Contrapositive: There can't be two different tuples that have the same value for key K.
  2. Minimal: No proper subset of K has the above property.

- A *superkey* is a set of attributes of R that includes a key of R.
  - That is, for any superkey, a key is a subset of the attributes.
  - All keys are superkeys but some superkeys are not keys.

Examples

- Student(studentID, name, major, gender, avgGPA).
  - {studentID} is a key because two different students can't have the same studentId. It is also a superkey.
  - {studentID, name} is a superkey but not a key.
  - {studentID, name, major, gender, avgGPA} is a superkey but not a key.


- There can be multiple keys in general.
  - One key is chosen and define as the *primary key*, while the rest are *candidate keys*.
- Student(studentID, name, dob, major, gender, avgGPA)
  - {studentID}, {name, dob} are keys and also superkeys.
    - {studentID} is the primary key.
    - {name, dob} is the candidate key.
      - » [Note: This is a questionable toy example.]
    - {name, dob, avgGPA} is a superkey.
  - Can you think of a realistic multi-attribute key for a <u>different</u> table?

True or False?

By looking at a bunch of instances of a relation schema, we can determine whether or not a set of attributes is a key.

Answer: ??

Careful. You can determine that a set of attributes is not a key by looking at instances,

… but you can't determine that a set of attributes is a key.

What is a Data Model?

- A *data model* is a mathematical formalism that consists of three parts:
  1. A notation for describing data and mathematical objects for representing data
  2. A set of operations for manipulating data
     - Retrieving and storing/modifying (insert, update, delete)
  3. Constraints on the data

Three Types of Relations in an RDBMS

1. Stored relations, called tables
2. Views
3. Temporary results from computations, including answers to queries

The relational model is closed under composition.

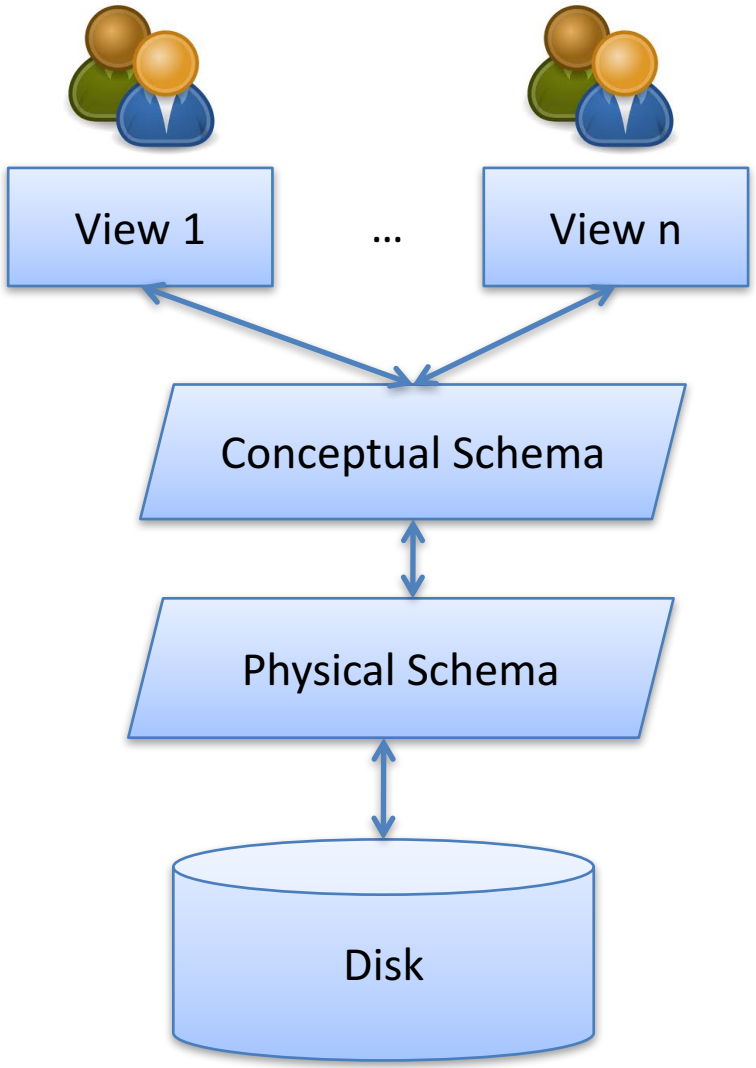
- It's also set-oriented
- … and functional (no side-effects)
- … and non-procedural (declarative)
- … and enables data-independence (at two levels)

Data Independence

- The relational data model provides a logical view of the data, and hides the physical representation of data.
  - Data is represented, conceptually, as tables, and that may not correspond to how it is stored on disk.
  - Operators manipulate data as tables. Users focus formulate queries based on *what* is needed, without knowing *how* the data is stored.
    - Optimizer generates a plan on *how* to compute the answers.
- Advantages:
  - Applications are shielded from low-level details.
  - Physical database design and optimizer can evolve, without affecting applications.

Two levels of Data Independence

- *Logical data independence*: Protects views from changes in logical (conceptual) structure of data.
  - Which tables do you have?
- *Physical data independence*: Protects conceptual schema from changes in physical structure of data.
  - How are tables stored?

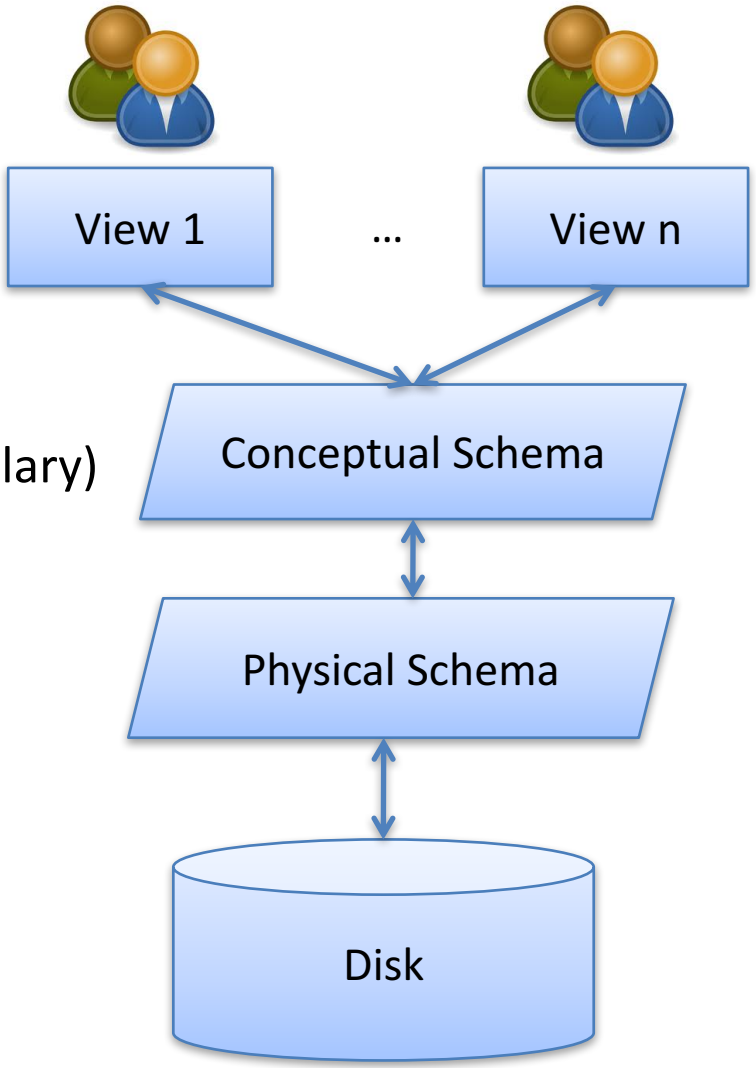


1: Store Professor One Way

View 1 defines
Faculty(name, department)
based on the Professor
relation schema.

Professor(profid, name, department, salary)

Professor relation may be
stored as a sorted file, ordered
by profid (one possible
physical representation).



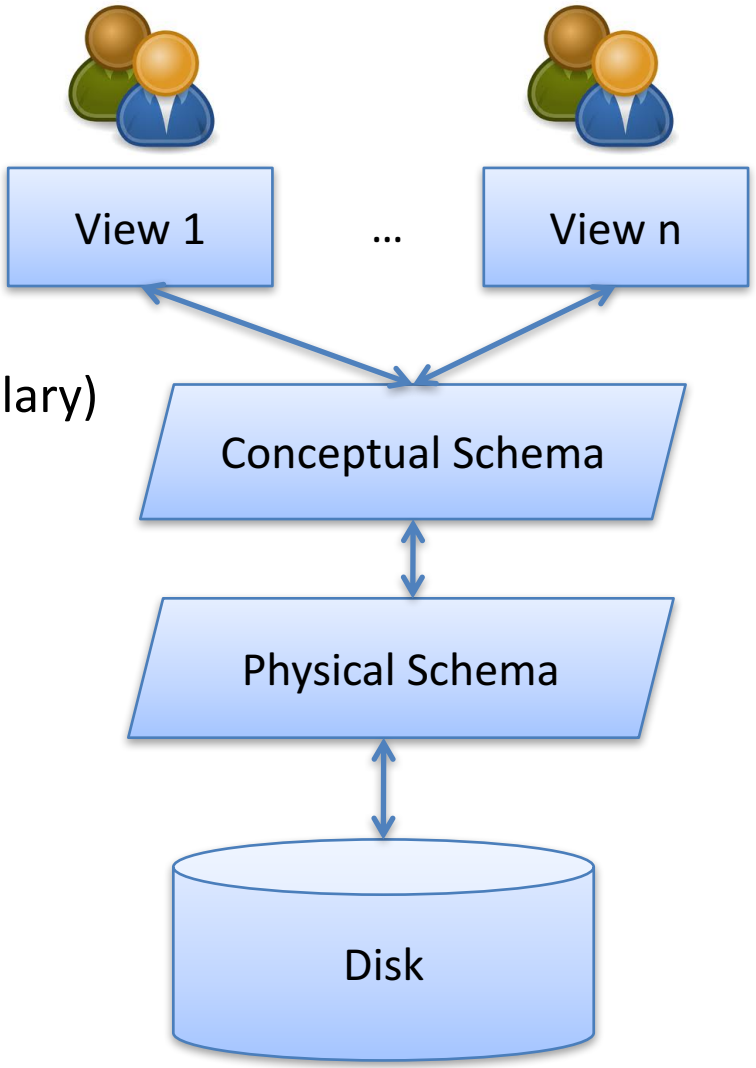
2: Store Professor a Different Way

View 1 defines
Faculty(name, department)
based on the Professor
relation schema.

Professor(profid, name, department, salary)

Professor relation may be stored as an unsorted file, with a B+ tree index on profid (another possible physical representation).

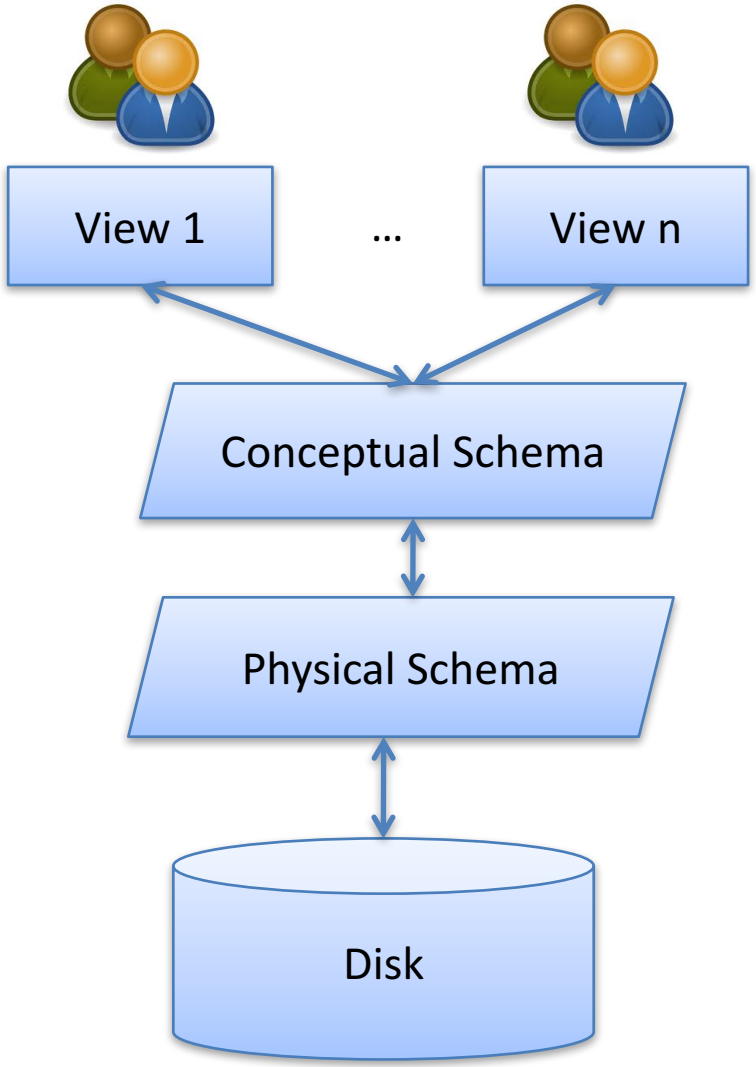
Physical data independence: Protects conceptual schema from changes in physical structure of data.



Issues for Data Independence

- *Logical data independence*: Protects views from changes in logical (conceptual) structure of data.
  - Okay to change which tables you have, as long as views can be defined correctly to support retrieval and modification operations.

- *Physical data independence*: Protects conceptual schema from changes in physical structure of data.
  - Okay to change physical storage of tables.
  - But performance may change depending on how tables are stored!
    - There's usually an index (B-tree or hash) on the primary key.



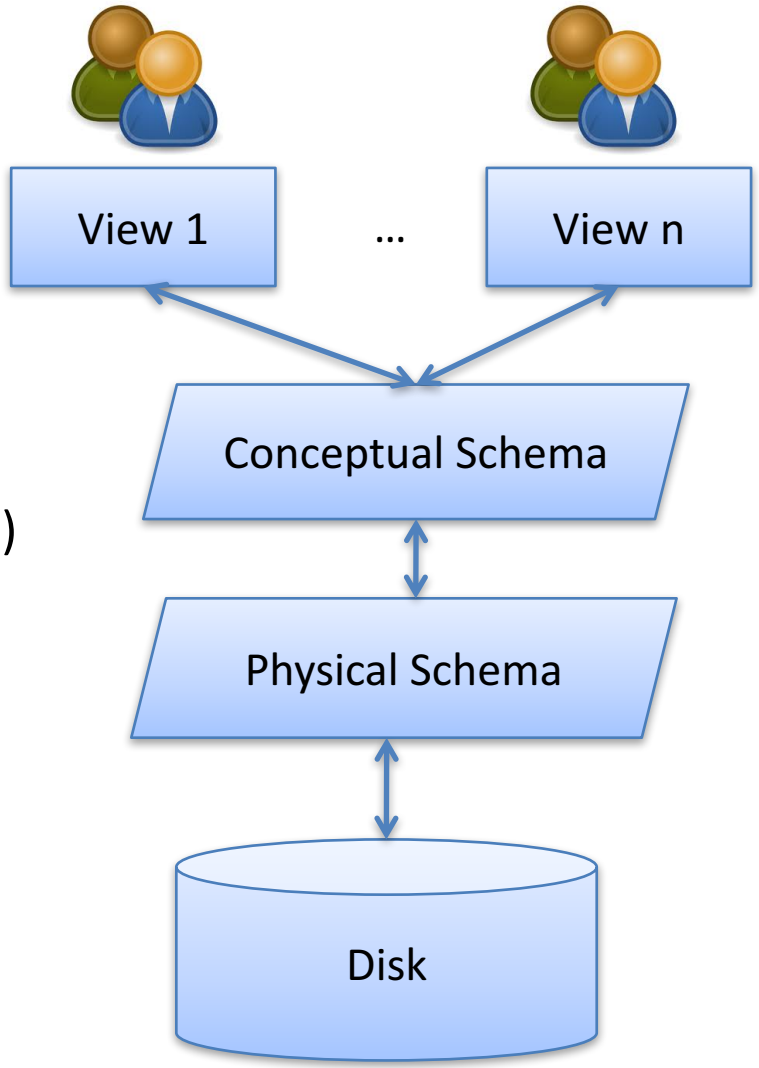
3-Change Conceptual Schema

View 1 defines
Faculty(name, department)
based on the ProfessorPublic relation schema. Users are unaffected.

ProfessorPrivate(pid, salary)
ProfessorPublic(pid, name, department)

Logical data independence: Protects views from changes in logical (conceptual) structure of data.

Do relations ProfessorPrivate and ProfessorPublic have to be stored the same way?



Summary

- Data model
- Relation schema
  - Attributes or column names
  - Tuple or row
  - Columns
  - Arity
- Relation instance
- Relational database schema
- A database (an instance of a database schema)
- Logical and Physical data independence