Lecture 2:
The Relational Data Model

Instructor: Shel Finkelstein

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

Important Notices

- Lecture slides and other class information will be posted on Piazza (not Canvas).
  - Slides are posted under Resources→Lectures
  - Lecture Capture recordings are available to all students under Yuja.
  - There's no Lecture Capture for Lab Sections.
- All Lab Sections (and Office Hours) meet this week, In-Person.
  - Please be sure to attend to learn about Gradiance, PostgreSQL, etc.!
  - Some slides for Lab Sections will be posted on Piazza under Resources→Lab Section Notes
- You should have Gradiance access this week, via Lab Sections.
  - For Gradiance, the token is **EC7184D3**
  - The First Gradiance Assignment will be posted next week.
- Office Hours for TAs and me are posted in Syllabus and Lecture1, as well as in Piazza notice @7.

Important Notices

- Monday, January 15 is a holiday, Martin Luther King Day.
  - No Lecture, no Lab Section, no Office Hours, no Tutoring on that day.
- On Wednesday, January 17, I'll be at the Conference on Innovative Database Research (CIDR), so I'll record that Lecture in advance, and it for directly under Yuja on Canvas.
  - I will not be in our classroom on Wednesday, January 17.
  - But I will post a reminder notice on Piazza, pointing you to that recording.
  - If you have questions about that Lecture recording, please post them on Piazza, and I'll try to answer them quickly.
    - You may make Piazza posts anonymously.
    - You may make Piazza posts which only instructors can see. But anonymous is better, so that we only have to answer questions once.
  - And I will be in my office, E2-249B, for my usual Wednesday Office Hour, which is from 2:00 - 3:00pm.
  - Normal Lab Section and Office Hours will be held by Teaching Assistants on that day.

Group Tutor-1

Alex Asch
alasch@ucsc.edu

Tutoring Times:

- Tuesday: 1:00pm – 2:00pm
- Friday: 9:20am – 10:25am
- Friday: 1:30pm – 2:30pm

Tutoring will be in Jack's Lounge, on the first floor of Baskin Engineering.

Group Tutor-2

Yash Raj Singh
ysingh5@ucsc.edu

Tutoring Times:

- Monday: 100pm – 2:00pm
- Tuesday: 2:00pm – 3:00pm
- Wednesday: 1pm – 2:00pm

Tutoring will be in Jack's Lounge, on the first floor of Baskin Engineering.

Can You Answer These Questions?

Not a homework; CSE180 students should find these questions 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

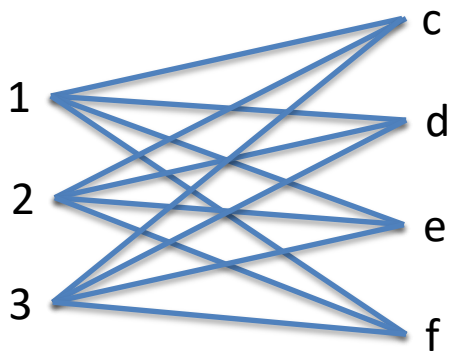
- What is the Cartesian product of {a,b,c,d} and {1,2,3}?

  { (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', 'TIM'}, or may be of standard types (INTEGER, FLOAT, DATE, …)

A Few Examples of Relations

- New York Stock Exchange Quotes
- U.S. Presidents
  - Scroll down to the pictures
  - Not in “First Normal Form”
- Current NFL Stadiums
- National Basketball Association Standings

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 $D_1 \times D_2$?
- 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_1 has n_1 elements and D_2 has n_2 elements, then how many relation instances can one construct from $D_1 \times D_2$?
- If D_i has n_i elements, then how many relation instances 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 sequence of attributes, (A_1, …, A_k), often written as R(A_1, …, 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:text, major:text, gender:text, avgGPA:double)
  - … or simply, Student(studentID, name, major, gender, avgGPA)

    when types are implicit

Note: *text* is <u>not</u> a standard datatype, but some systems, including PostgreSQL, support it. We'll discuss CHAR and VARCHAR soon.

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.
  - Many relational databases now support semi-structured data, as well as 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:text, major:text, gender:text, 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)

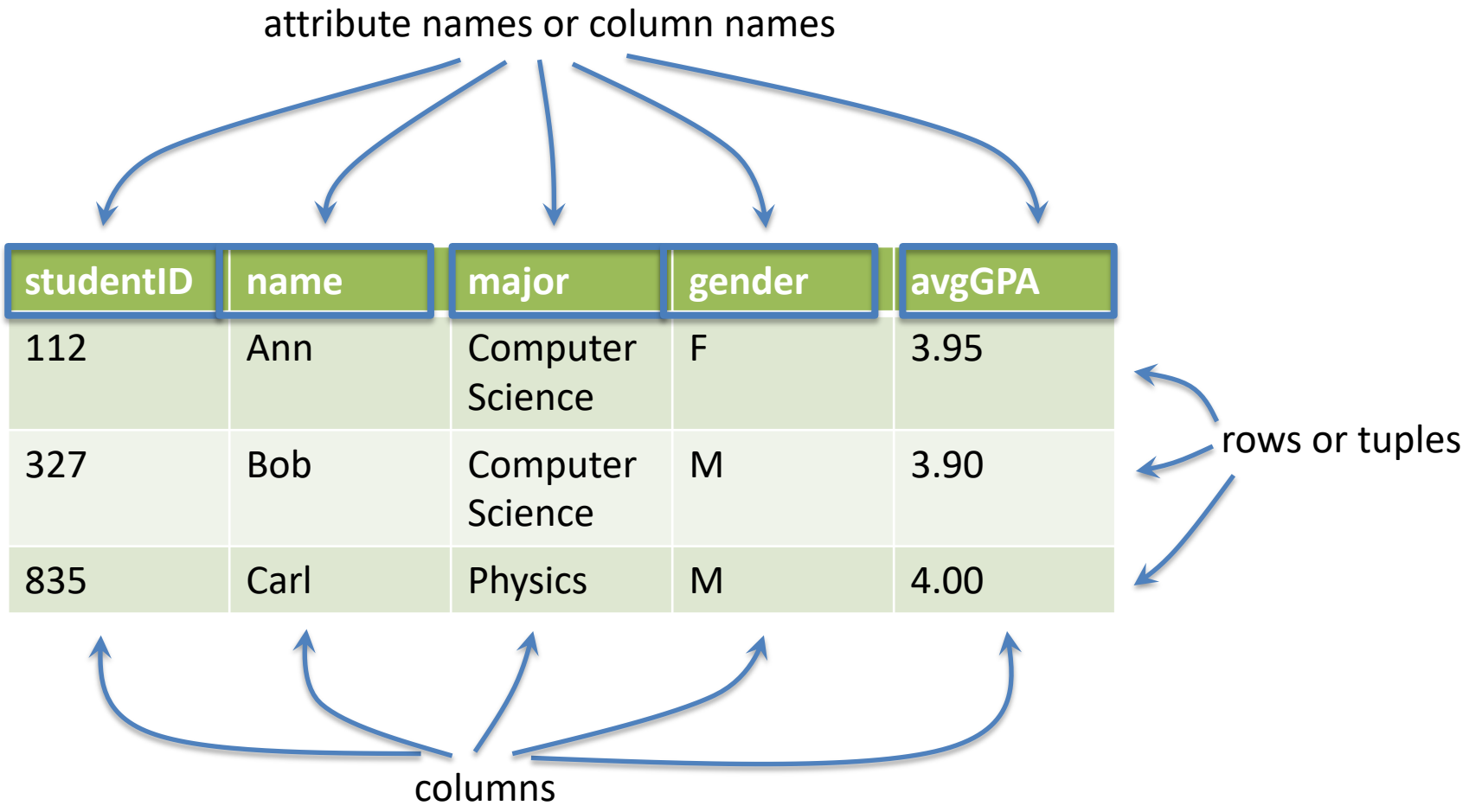
{ (112, ‘Ann’, ‘CS’, ‘F’, 3.95),
(327, ‘Bob’, ‘CS’, ‘M’, 3.90),
(835, ‘Carl’, ‘Physics’, ‘M’, 4.00) }

{ (‘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

Quotes not shown around strings/character types



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

A Relational Database Schema

- A *relational 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)

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, …

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 Lab Assignments.
  - 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!

Superkeys and Keys

A **superkey S** for a relation schema R is a subset of the attributes of R such that:

1. There can't be two different tuples in an instance of R that have the same value for all the attributes in S.

We'll assume that relations cannot have duplicates. Then for any relation R, there always is a superkey of R, namely attrib(R), all the attributes of R.

A **key K** for a relation schema R is a subset of the attributes of R such that:

1. There can't be two different tuples in an instance of R that have the same value for all the attributes in K.
2. Minimal: No proper subset of K has the above property.

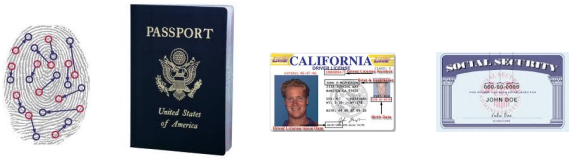
All keys are superkeys … but some superkeys are not keys.

A superkey is a **constraint** on the allowable instances of relation R.

– Since every key is a superkey, a key is also a constraint on the allowable instances of relation R.

Key Examples

- Student(studentID, name, major, gender, avgGPA).
  - {studentID} would be a key, because two different students can't have the same studentId. It is also a superkey.
  - {studentID, name} is a superkey, but it's not a key.
  - {studentID, name, major, gender, avgGPA} is a superkey but it's 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 data
     - Modifying data (insert, update, delete)
  3. Constraints on the data

Three Types of Relations in an RDBMS

1. Stored relations, called tables (or relation instances)
2. Views
   - Views are the results of queries on tables.
   - Views are like tables, except that they are not stored in the database.
3. Temporary results from computations, including answers to queries

The relational model is closed under composition.

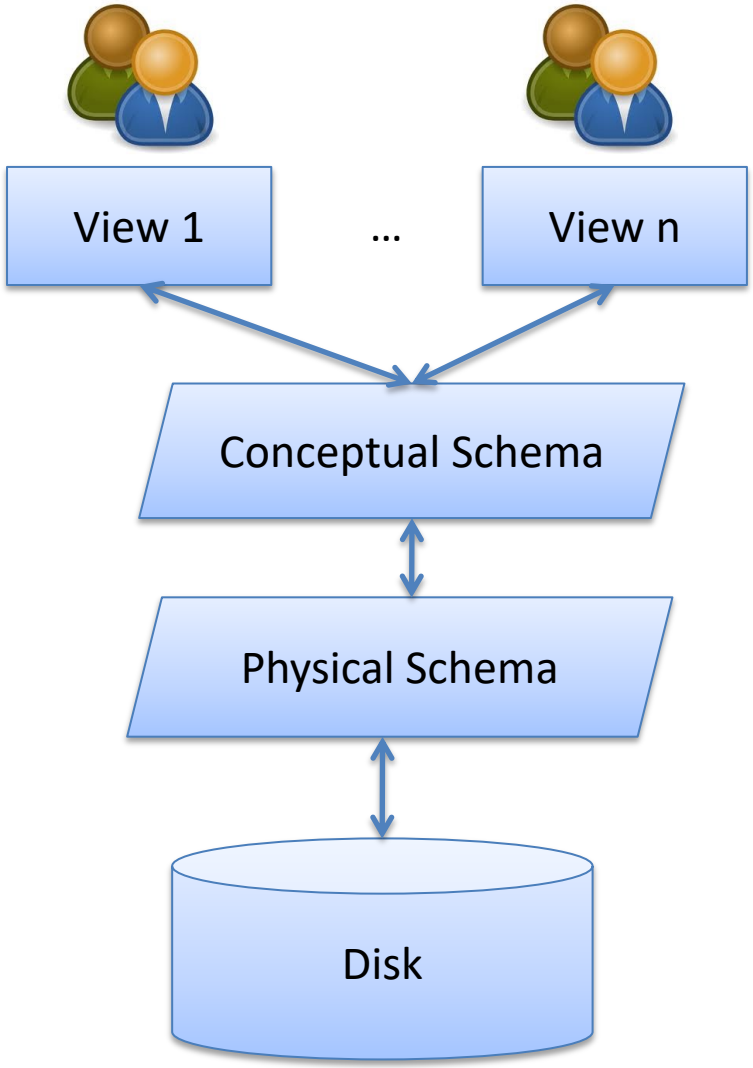
- It's also set-oriented
- … and functional (no side-effects)
- … and non-procedural (declarative)
- … and enables data-independence, at two different levels

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, which might not correspond to how data is stored.
  - 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?
  - A view is like a table, except that it's computed from tables, not stored in the database.
- ***Physical*** *data independence*: Protects conceptual schema from changes in **physical** structure of data.
  - How are tables stored?

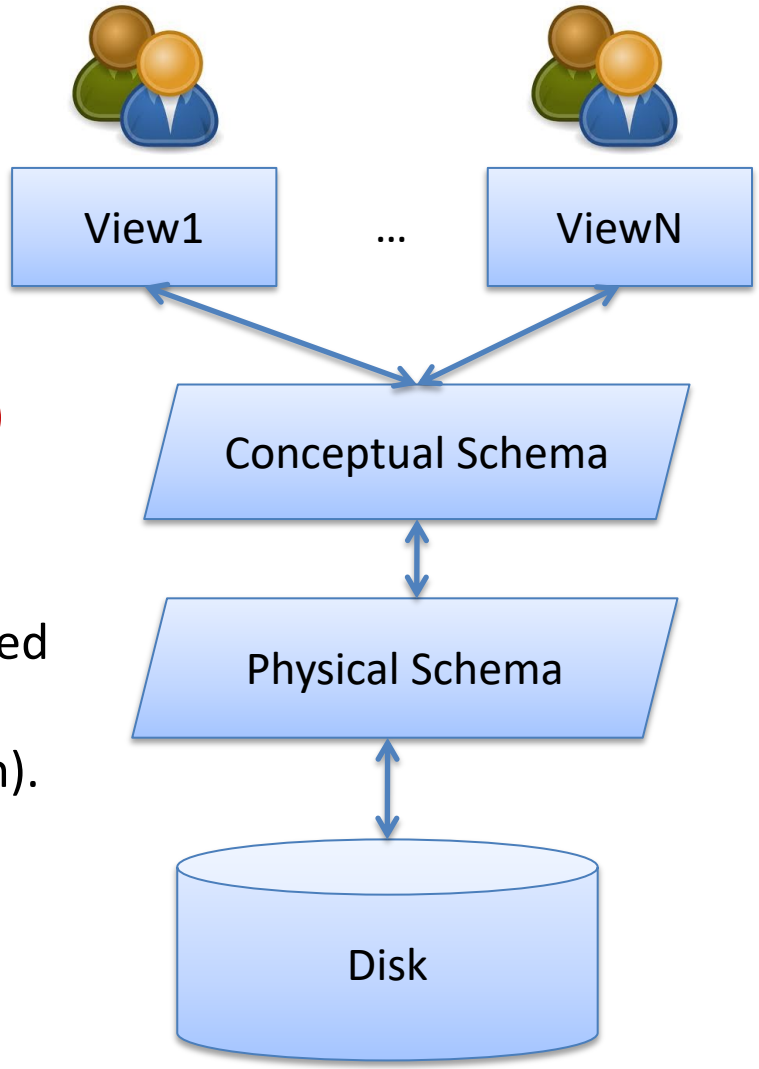


1: Store Professor One Way

View1 defines Faculty(name, department) based on the Professor relation schema.

Professor(profid, name, department, salary)

Professor relation might be stored as a sorted file, ordered by profid.
(That's one possible physical representation).



2: Store Professor a Different Way

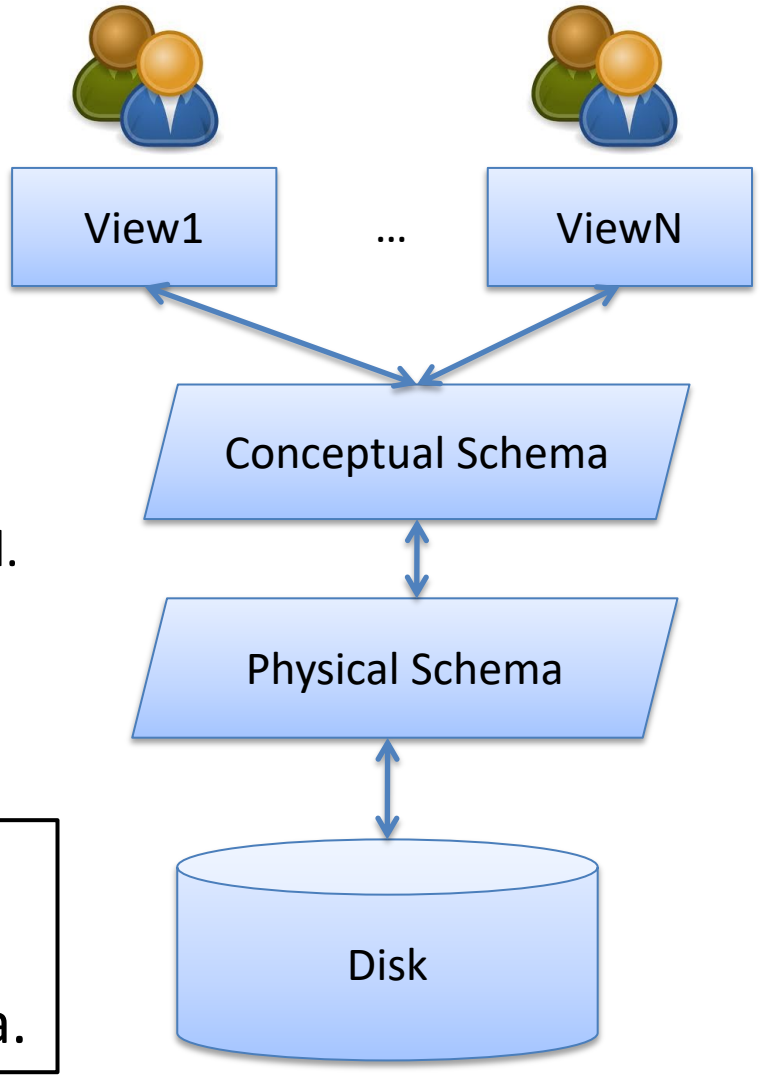
View1 defines Faculty(name, department) based on the Professor relation schema.

Professor(profid, name, department, salary)

Professor relation might be stored as an unsorted file, with a B+ tree index on profid.

(That's a different possible physical representation; there are many others.)

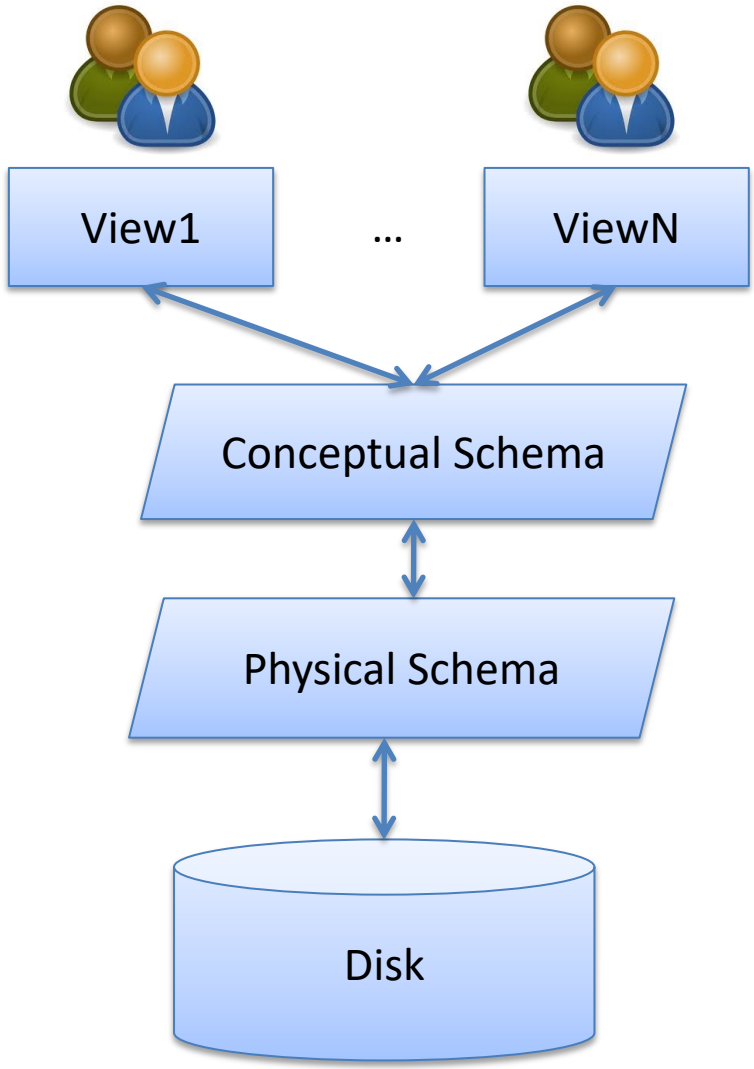
Physical data independence:
Protects conceptual schema from changes in **physical** structure of data.



Benefits of Data Independence

- ***Logical data independence***:
  Protects views from changes in **logical** (conceptual) structure of data.
  - Okay to change which tables you have, if all views can be defined to support retrieval and modification operations correctly.
  - Something might be a table today, and instead be a view tomorrow!

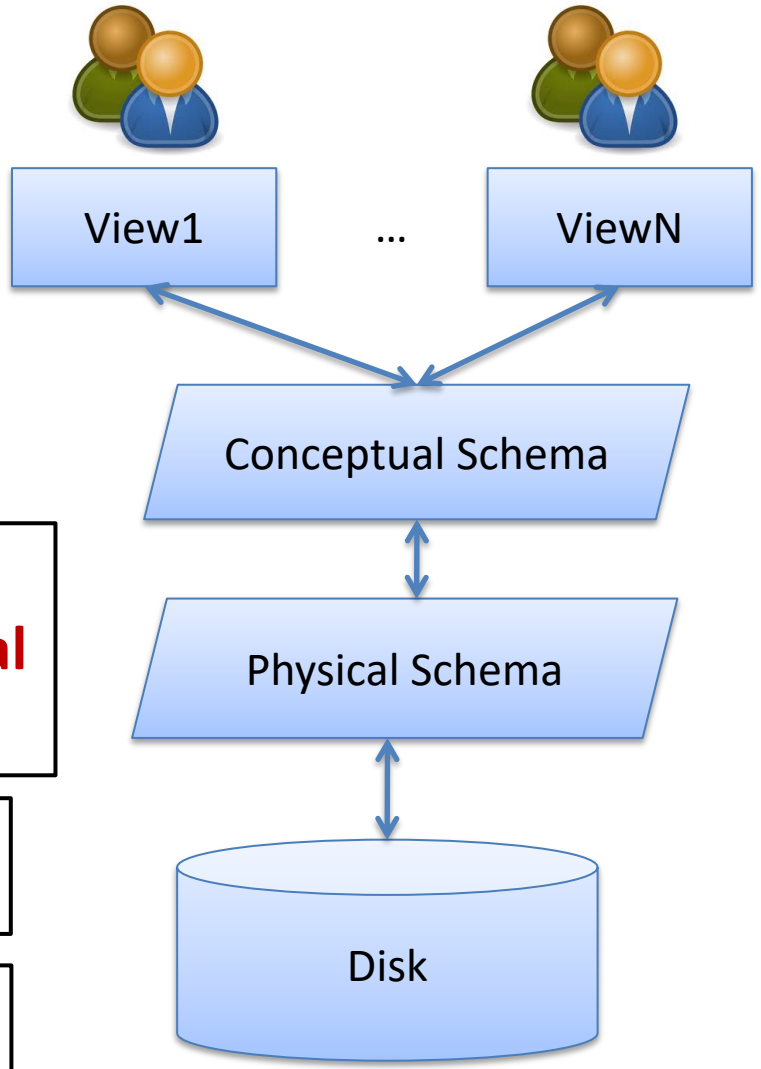
- ***Physical data independence***:
  Protects conceptual schema from changes in **physical** structure of data.
  - Okay to change physical storage of tables.
  - But performance may change depending on how tables are stored!
    - There's usually an index (B-tree or Hash) on the Primary Key.



3-Change Logical (Conceptual) Schema

Suppose that instead of having relation:
Professor(profid, name, department, salary),
we have relations: ProfPrivate(profid, salary)
and ProfPublic(profid, name, department)

And View1 defines Faculty(name, department)
based on the ProfPublic relation,
instead of based on the Professor relation.
Users of View1 are unaffected.



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

Could we reconstruct the Professor relation as a View from ProfPublic and ProfPrivate?

Do the relations ProfPrivate and ProfPublic have to be stored the same way?

Summary

- Data model
- Relation Schema
  - Attributes or column names
  - Tuple or row
  - Columns
- Relation Instance
- Relational Database Schema
- A Relational Database Instance (“a database”)
  - An Instance of a Relational Database Schema
- Logical and Physical Data Independence