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 Lab Sections.
  - First Gradiance Assignment will be posted soon.
- Lab1 assignment has been posted on Piazza under Resources→Lab1. General Information about Labs has also been posted. See Piazza announcement about Lab1.
  - Lab1 will be discussed at Lab Sections.
  - Due **Sunday, April 14**, by 11:59pm.
    - Late Lab Assignments will not be accepted.
    - Be sure that you post the correct file!
  - Your solution should be submitted via Canvas as a zip file.
  - Canvas will be used for both Lab submission and grading.

First, Let's Answer These Questions
Practice Homework 0

Not a homework, but for CMPS180 students, these should be very 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

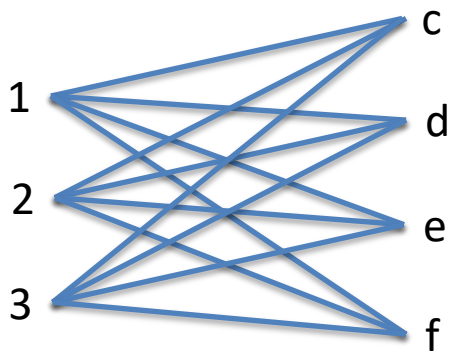
- What is the Cartesian product of {a,b,c,d} and {1,2,3}?

  { (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

- Internet Stocks:
  - http://www.marketwatch.com/tools/industry/stocklist.asp?bcind_ind=9535
- Presidential info (page down)
  - http://politicsandprosperity.com/facts-about-presidents/
- NFL Stadium info:
  - https://en.wikipedia.org/wiki/List_of_current_National_Football_League_stadiums - List_of_current_stadiums
- 2017-2018 National Basketball Association Standings
  - http://media.nba.com/Stats/Standings.aspx?leagueid=00&seasonid=22017

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 $D_1 \times \ldots \times D_k$?

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:text, major:text, gender:text, avgGPA:double)
  - … or simply, Student(studentID, name, major, gender, avgGPA)

    when types are implicit

Note: *text* is not a standard datatype, but many 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.
  - 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: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) }

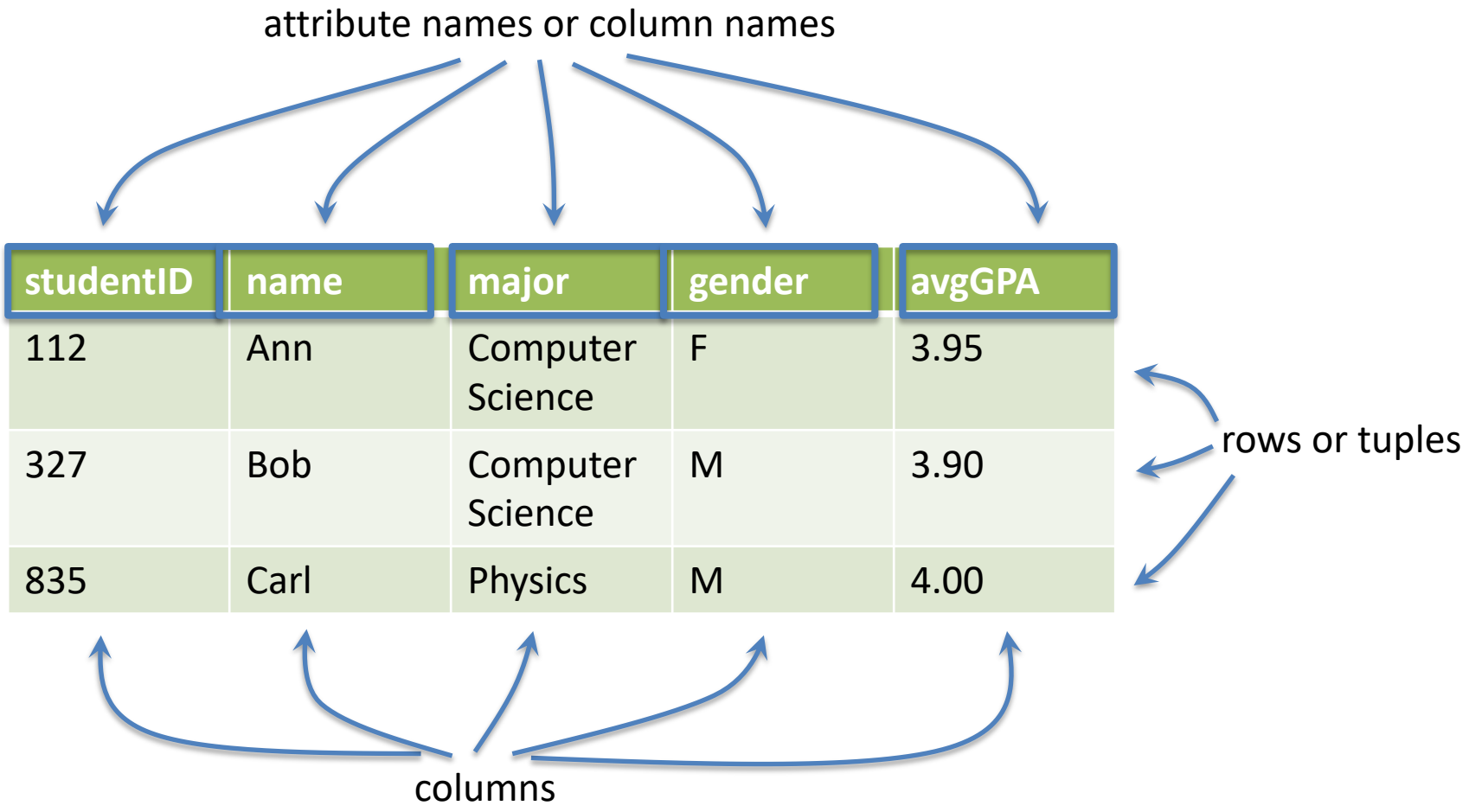
{ (‘112’, ‘Ann’, ‘CS’, ‘F’, 3.95),
(‘327’, ‘Bob’, 'CS’, ‘M’, 3.90),
(‘835’, ‘Carl’, ‘Physics’, ‘M’, 4.00) }

X

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, …

DB-Engines Popularity Rating for RDBMS

For 2018, look here

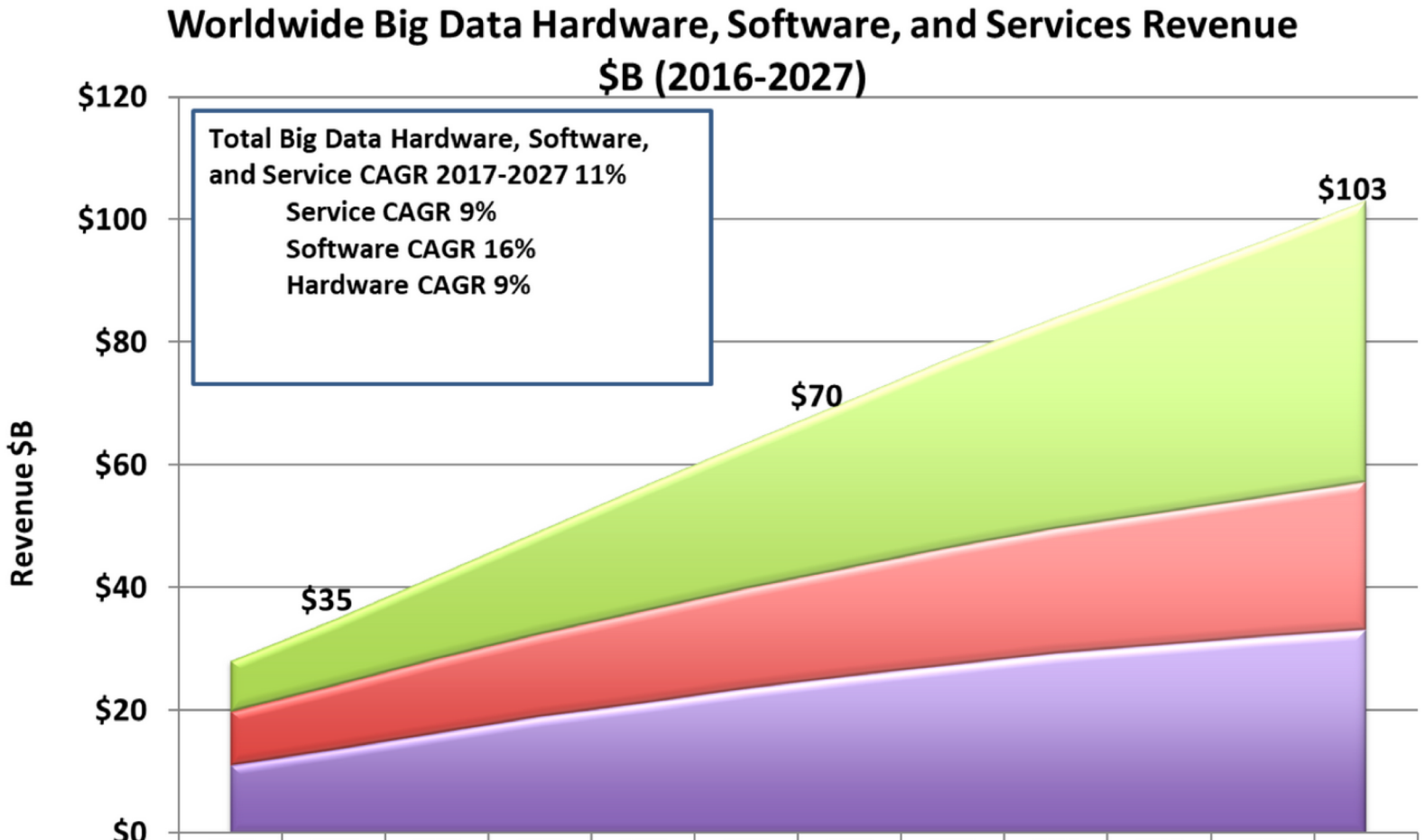
Rank Sep 2017	Rank Aug 2017	Rank Sep 2016	DBMS	Database Model	Score Sep 2017	Score Aug 2017	Score Sep 2016
1.	1.	1.	Oracle	Relational DBMS	1359.09	-8.78	-66.47
2.	2.	2.	MySQL	Relational DBMS	1312.61	-27.69	-41.41
3.	3.	3.	Microsoft SQL Server	Relational DBMS	1212.54	-12.93	+0.99
4.	4.	4.	PostgreSQL	Relational DBMS	372.36	+2.60	+56.01
5.	5.	5.	DB2	Relational DBMS	198.34	+0.87	+17.15
6.	6.	6.	Microsoft Access	Relational DBMS	128.81	+1.78	+5.50
7.	7.	7.	SQLite	Relational DBMS	112.04	+1.19	+3.41
8.	8.	8.	Teradata	Relational DBMS	80.91	+1.67	+7.84
9.	9.	9.	SAP Adaptive Server	Relational DBMS	66.75	-0.16	-2.41
10.	10.	10.	FileMaker	Relational DBMS	61.00	+1.35	+5.64
11.	11.	↑ 13.	MariaDB	Relational DBMS	55.47	+0.78	+16.94
12.	↑ 13.	↓ 11.	Hive	Relational DBMS	48.62	+1.31	-0.21
13.	↓ 12.	↓ 12.	SAP HANA	Relational DBMS	48.33	+0.36	+4.91
14.	14.	14.	Informix	Relational DBMS	27.84	+0.41	-0.35
15.	↑ 16.	15.	Vertica	Relational DBMS	22.01	+0.20	+0.95
16.	↓ 15.	↑ 17.	Microsoft Azure SQL Database	Relational DBMS	21.60	-0.31	+2.18
17.	17.	↓ 16.	Netezza	Relational DBMS	19.40	-0.17	-0.41
18.	18.	18.	Firebird	Relational DBMS	17.86	-0.20	+2.28
19.	19.	↑ 22.	Impala	Relational DBMS	13.56	+0.51	+4.34
20.	20.	↓ 19.	Amazon Redshift	Relational DBMS	13.04	+0.21	+2.21
21.	21.	↑ 24.	Google BigQuery	Relational DBMS	11.87	+0.06	+5.29
22.	22.	↓ 21.	Greenplum	Relational DBMS	11.42	+0.01	+2.01
23.	23.		Oracle Essbase	Relational DBMS	11.35	+0.01	
24.	24.	↑ 26.	Spark SQL	Relational DBMS	10.92	+0.15	+5.07
25.	25.	↓ 20.	dBASE	Relational DBMS	10.44	+0.26	+0.65

DB-Engines Popularity Rating for DB-Engines

For 2018, look here

Rank Sep 2017	Rank Aug 2017	Rank Sep 2016	DBMS	Database Model	Score Sep 2017	Score Aug 2017	Score Sep 2016
1.	1.	1.	Oracle	Relational DBMS	1359.09	-8.78	-66.47
2.	2.	2.	MySQL	Relational DBMS	1312.61	-27.69	-41.41
3.	3.	3.	Microsoft SQL Server	Relational DBMS	1212.54	-12.93	+0.99
4.	4.	4.	PostgreSQL	Relational DBMS	372.36	+2.60	+56.01
5.	5.	5.	MongoDB	Document store	332.73	+2.24	+16.74
6.	6.	6.	DB2	Relational DBMS	198.34	+0.87	+17.15
7.	7.	↑ 8.	Microsoft Access	Relational DBMS	128.81	+1.78	+5.50
8.	8.	↓ 7.	Cassandra	Wide column store	126.20	-0.52	-4.29
9.	9.	↑ 10.	Redis	Key-value store	120.41	-1.49	+12.61
10.	10.	↑ 11.	Elasticsearch	Search engine	120.00	+2.35	+23.52
11.	11.	↓ 9.	SQLite	Relational DBMS	112.04	+1.19	+3.41
12.	12.	12.	Teradata	Relational DBMS	80.91	+1.67	+7.84
13.	13.	↑ 14.	Solr	Search engine	69.91	+2.95	+2.95
14.	14.	↓ 13.	SAP Adaptive Server	Relational DBMS	66.75	-0.16	-2.41
15.	15.	15.	HBase	Wide column store	64.34	+0.82	+6.53
16.	16.	↑ 17.	Splunk	Search engine	62.57	+1.11	+11.28
17.	17.	↓ 16.	FileMaker	Relational DBMS	61.00	+1.35	+5.64
18.	18.	↑ 20.	MariaDB	Relational DBMS	55.47	+0.78	+16.94
19.	↑ 20.	↓ 18.	Hive	Relational DBMS	48.62	+1.31	-0.21
20.	↓ 19.	↓ 19.	SAP HANA	Relational DBMS	48.33	+0.36	+4.91
21.	21.	21.	Neo4j	Graph DBMS	38.42	+0.42	+2.06
22.	22.	↑ 25.	Amazon DynamoDB	Document store	37.82	+0.20	+10.40
23.	23.	↓ 22.	Couchbase	Document store	33.11	+0.14	+4.57
24.	24.	↓ 23.	Memcached	Key-value store	28.94	-1.02	+0.51

Wikibon Big Data Compound Annual Growth Rate Prediction

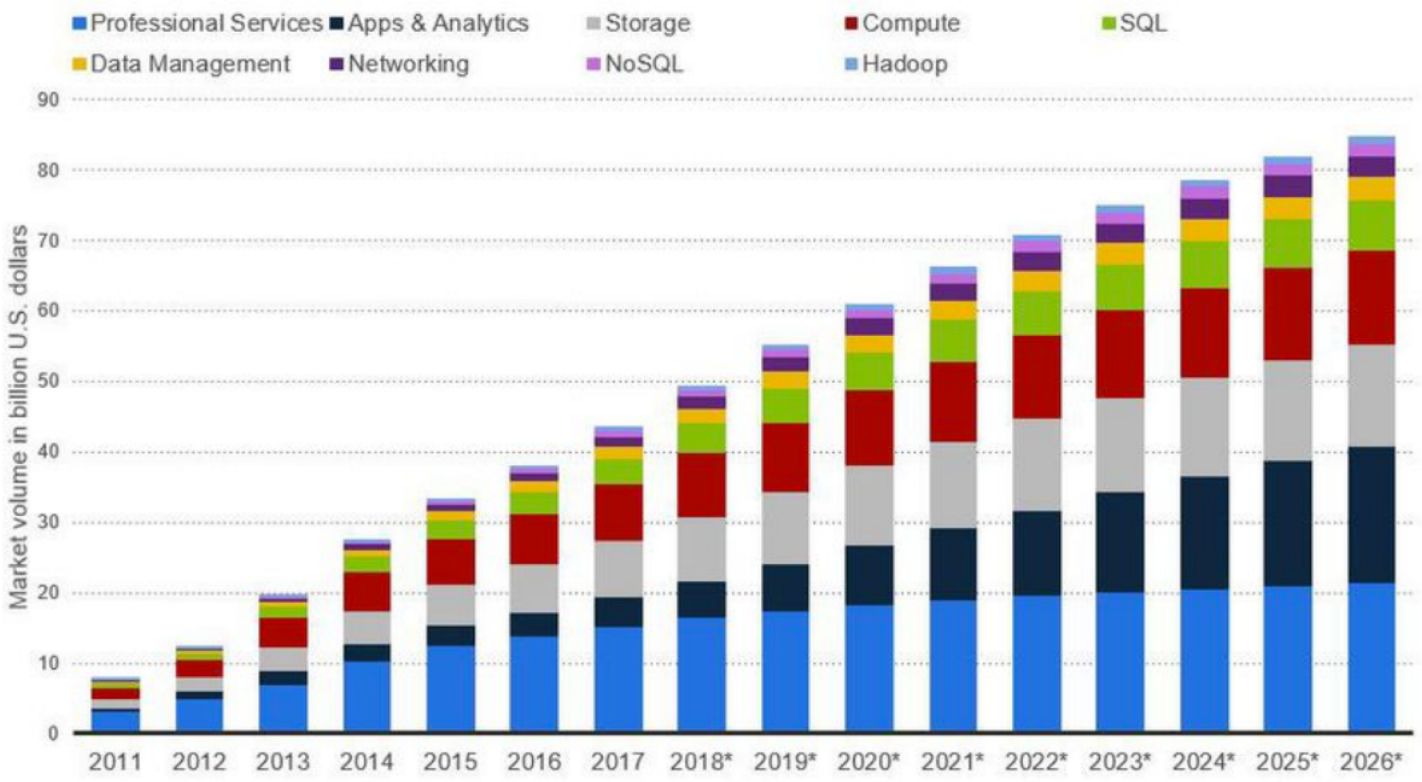


	2016	2017	2018	2019	2020	2021	2022	2023	2024	2025	2026	2027
Software	$8	$11	$14	$17	$20	$24	$27	$31	$34	$38	$42	$46
Hardware	$9	$10	$12	$14	$15	$16	$18	$19	$20	$22	$23	$24
Service	$11	$14	$16	$19	$21	$24	$26	$27	$29	$31	$32	$33
Total Big Data	$28	$35	$42	$49	$56	$64	$70	$77	$84	$90	$96	$103

Figure 1: Worldwide Big Data Hardware, Software, and Services Revenue $B 2016-2027

Big Data Market Worldwide Segment Revenue Forecast 2011-2026

Big Data Market Forecast Worldwide from 2011 to 2026, by segment (in billion U.S. dollars)



statista

SOURCE: WIKIBON AND REPORTED BY STATISTA.

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.

- 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 key is a constraint on the allowable instances of relation R.

Key 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 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
3. Temporary results from computations, including answers to queries

The relational model is closed under composition.

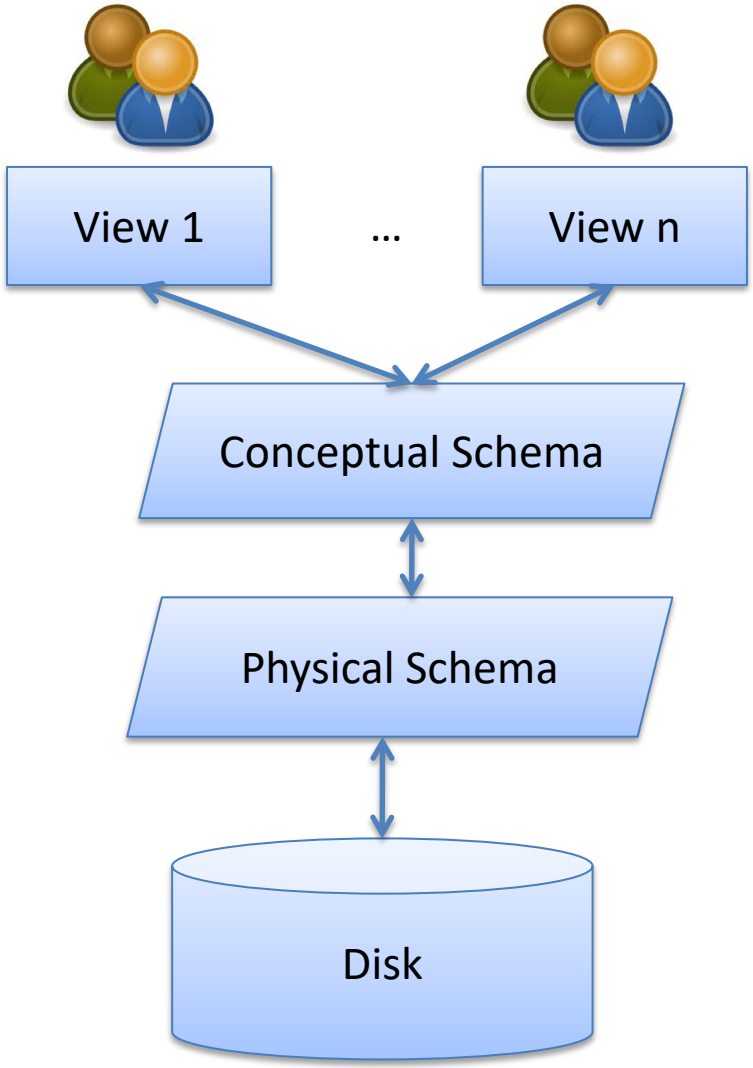
- It's also set-oriented
- … and functional (no side-effects)
- … and non-procedural (declarative)
- … and enables data-independence, at two different levels

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?
- ***Physical*** *data independence*: Protects conceptual schema from changes in **physical** structure of data.
  - How are tables stored?

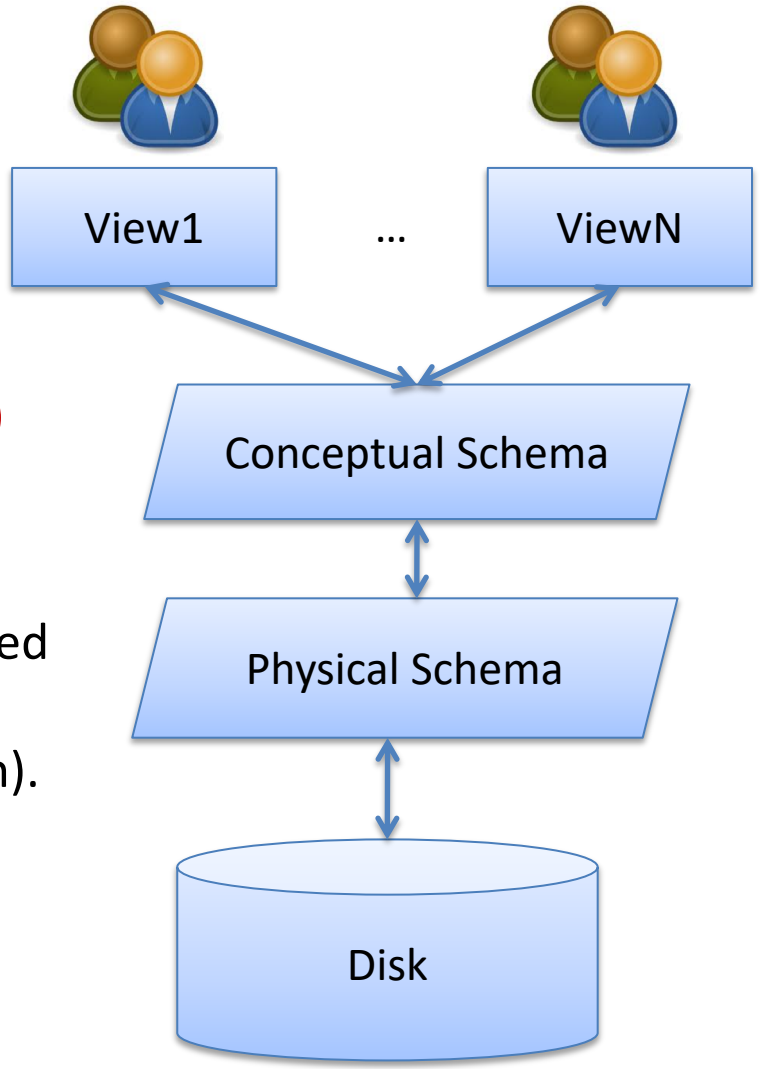


1: Store Professor One Way

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

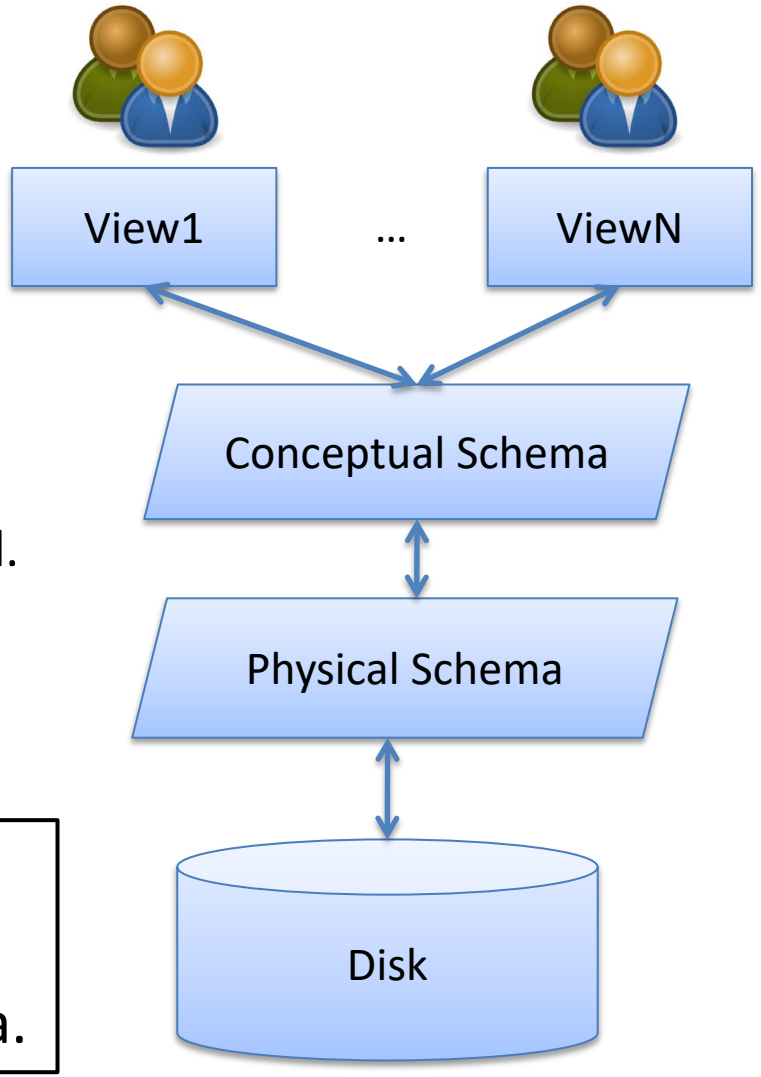
View1 defines Faculty(name, department) based on the Professor relation schema.

Professor(profid, name, department, salary)

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

(That's one alternative possible physical representation; there are many others.)

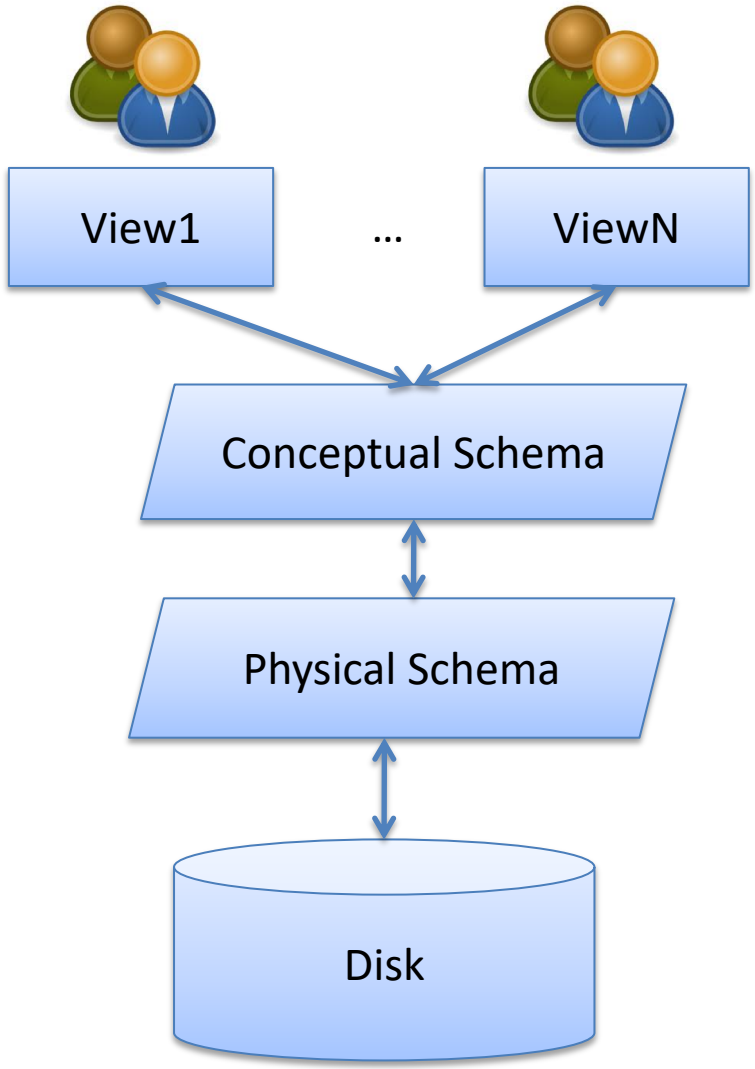
Physical data independence:
Protects conceptual schema from changes in **physical** structure of data.



Benefits of Data Independence

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

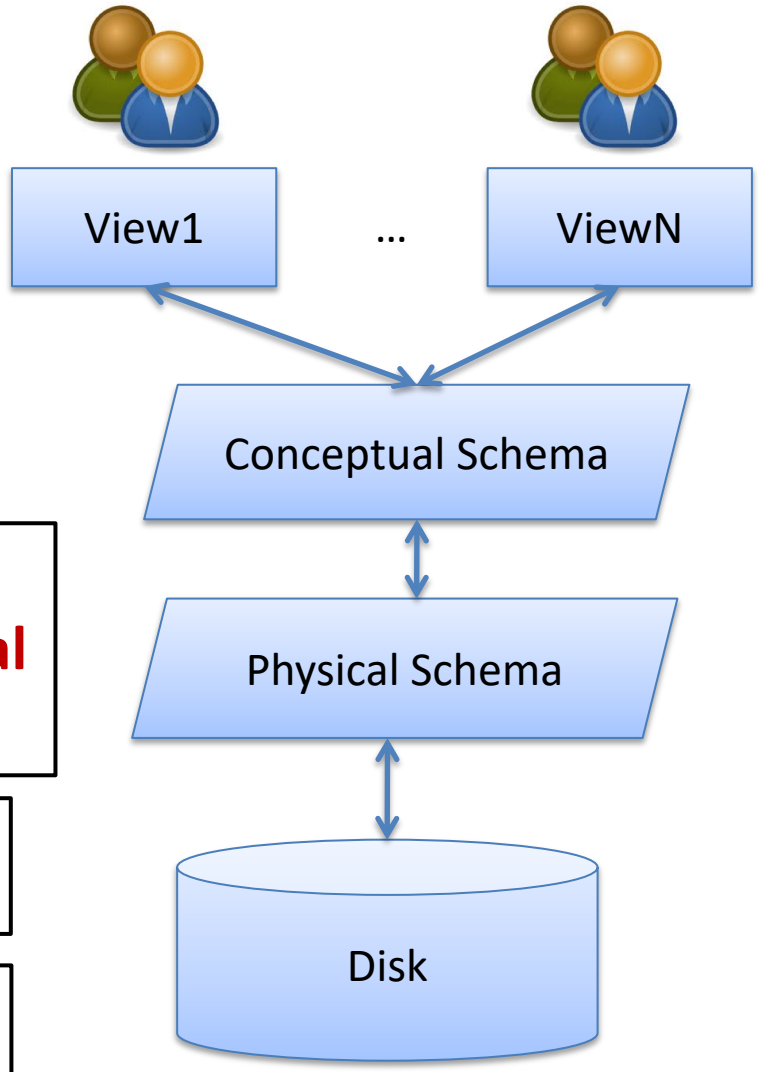
- ***Physical data independence***:
  Protects conceptual schema from changes in **physical** structure of data.
  - Okay to change physical storage of tables.
  - But performance may change depending on how tables are stored!
    - There's usually an index (B-tree or hash) on the primary key.



3-Change Conceptual Schema

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 Professors 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
  - Arity
- Relation instance
- Relational database schema
- A database (an instance of a database schema)
- Logical and Physical data independence