CSCB20

Introduction to Database and Web Application Programming

Anna Bretscher
Winter 2017

Welcome to CSCB20

Course Description:

A practical introduction to databases and Web app development.

Databases:

- terminology and applications;
- creating, querying and updating databases;
- the entity-relationship model for database design.

Welcome to CSCB20

Course Description:

A practical introduction to databases and Web app development.

Web documents and applications:

- static and interactive documents;
- Web servers and dynamic server-generated content;
- Web application development and interface with databases.

Course Layout

Database Design

5-6 weeks

Web Application Design

6-7 weeks

Lectures

2 hours per week

Tutorials

1 hour per week – start in week 2

Course Work

Term Work

- 3 Assignments 15% each

Exams

- Midterm 15%
- Final Exam 40%

How Do I Get Help?

Lectures

Course Website:
www.utsc.utoronto.ca/bretscher/b20

Course Discussion Board

Tutorials

TA and Instructor Office Hours

Textbook
– no official book
– good online resources linked to on website.

How Do I Stay Informed?

Come to class!

Join Piazza and check often.

Check the calendar for due dates of term work.

Check your utoronto email – this is where I will send out emails to the class.

Today

Databases:

- What?
- Where?
- Why?
- When?
- How?

Terminology

What…

Is a Database?

- A collection of *interrelated data.*
- The data is relevant to an *enterprise*.

Is a Database Management System (DBMS)?

- A database *and*
- A set of programs to *access* the database
- Provides a way to *store* and *retrieve* database information.
- Must be *convenient* and *efficient*.

Where?

Enterprise Information:

- Sales
- Accounting
- Human Resources
- Manufacturing
- Online Retailers

Banking and Finance:

- Banking
- Credit Card Transactions
- Finance

Other Applications?

- Universities
- Airlines
- Telecommunications
- …

When?

In the 1960s data storage changed from *tape* to *direct access*.

This allowed *shared interactive* data use.

Early databases were *navigational* which was very inefficient for searching.

Edgar Codd created a new system in the 1970s based on the *relational model*.

Late 1970s and early 1980s SQL was developed based on the *relational model* which is the foundation of *current databases* and what we will study.

In the 2000s, with increasingly *large datasets*, new *XML databases* and *NoSQL* databases are becoming more prevalent.

Why use databases?

- Commercialized management of large amounts of data
- Ability to update and maintain data
- Keep track of relationships between subsets of the data
- Efficient access and searching capabilities
- Multiple users can access and share data
- Ability to limit access to a portion of the data according to user type and enables security of data
- Minimizes redundancy of multiple data sets
- Enables consistency constraints
- Allows users an abstract view of the data which hides the details of how the data are stored and maintained.

Data Abstraction - How?

Physical Level

- Lowest level, *how* the data are actually stored.
- Usually in complex low-level data structures.

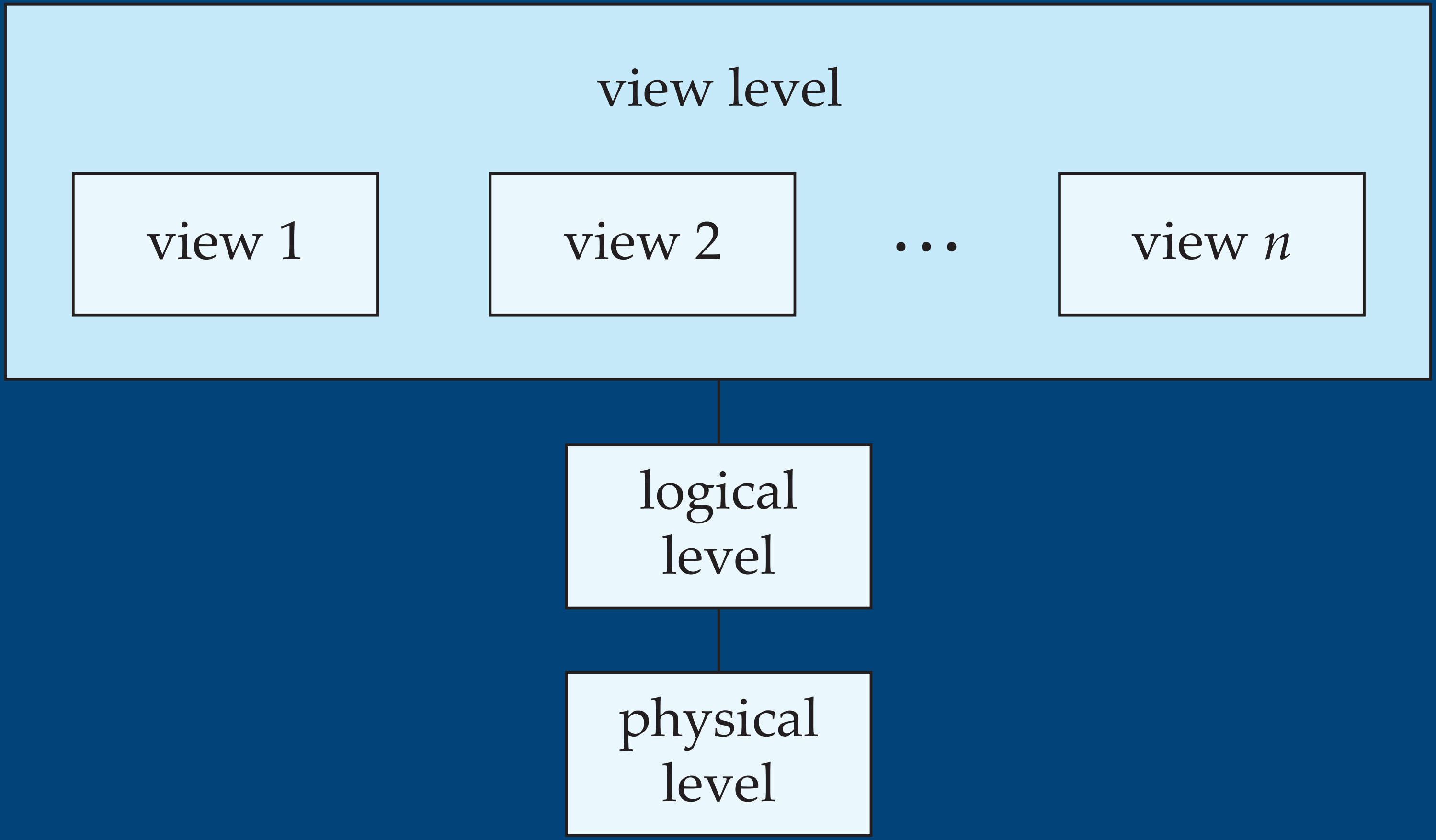
Logical Level

- *What* data are stored in the database and what *relationships* exist between the data.
- Implementing the *simple structure* of the logical level may require complex *physical low level structures.*
- Users of the logical level don't need to know about this.
- We refer to this as the *physical data independence*.

View Level:

- Highest level of abstraction - describes only a *small* portion of the database
- Allows user to simplify their interaction with the database system.
- Can have many *views. WHY is this good?*

Data Abstraction



*taken from: Database System Concepts 6th Ed.,Korth,Silberschatz, Sudharshan

Relational Model

Database is a collection of *tables* each having a unique name.

Each table also known as a *relation*.

Rows are referred to as *tuples*.

Columns are referred to as *attributes*.

An *instance* of a database is the information stored at a particular moment in time.

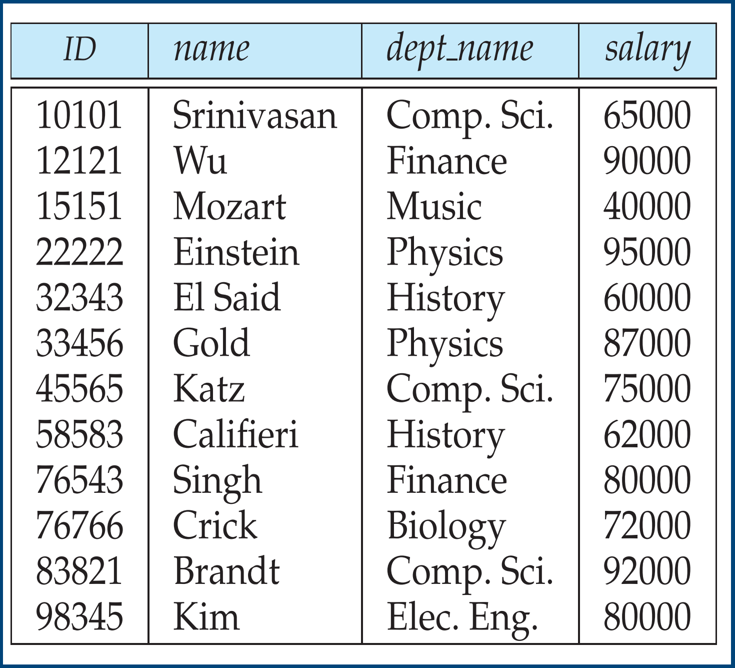
A database *schema* is the overall design of the database.

Which changes frequently? The *instance* or *schema* of a database?

- *taken from: Database System Concepts 6th Ed.,Korth,Silberschatz, Sudharshan

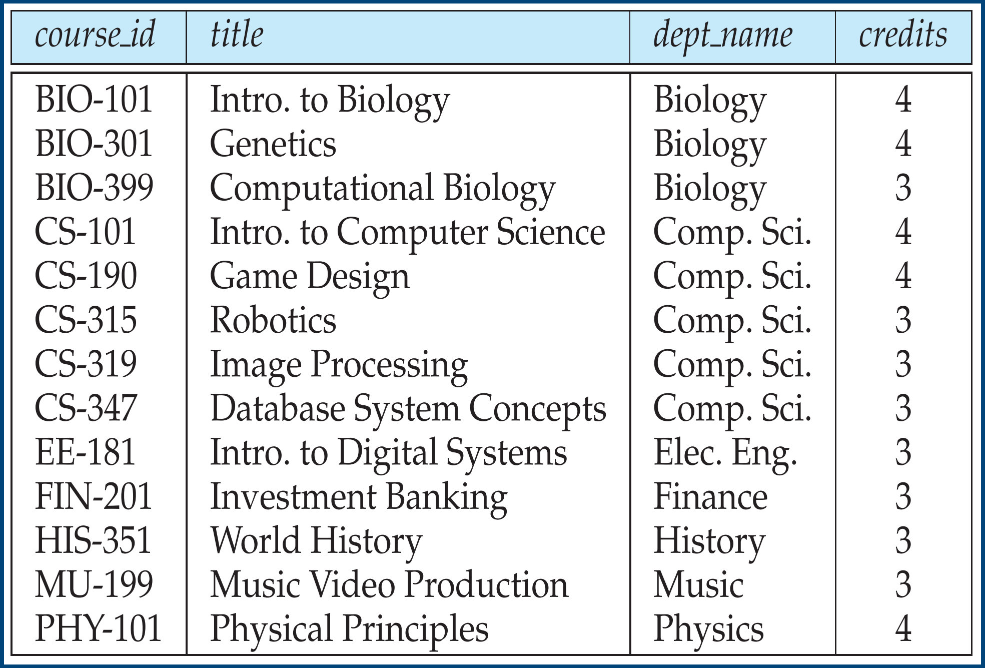
University Example*

Instructor Relation



ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

Course Relation



course_id	*title*	*dept_name*	*credits*
BIO-101	Intro. to Biology	Biology	4
BIO-301	Genetics	Biology	4
BIO-399	Computational Biology	Biology	3
CS-101	Intro. to Computer Science	Comp. Sci.	4
CS-190	Game Design	Comp. Sci.	4
CS-315	Robotics	Comp. Sci.	3
CS-319	Image Processing	Comp. Sci.	3
CS-347	Database System Concepts	Comp. Sci.	3
EE-181	Intro. to Digital Systems	Elec. Eng.	3
FIN-201	Investment Banking	Finance	3
HIS-351	World History	History	3
MU-199	Music Video Production	Music	3
PHY-101	Physical Principles	Physics	4

Give an example of an attribute, tuple.

What is the *domain* of the column *salary?*

*taken from: Database System Concepts 6th Ed.,Korth,Silberschatz, Sudharshan

Terminology

Database Schema: The logical design of the database.

Database Instance: A snapshot of the data in the database.

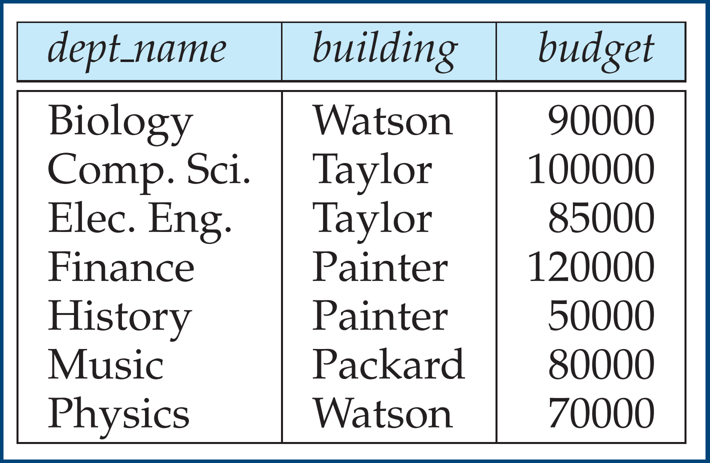
Relation Schema: A list of attributes and their corresponding domains.

The *department relation* has the schema:

department(dept_name, building, budget)

The *instructor relation* has the schema:

instructor(ID, name, dept_name, salary)



dept_name	*building*	*budget*
Biology	Watson	90000
Comp. Sci.	Taylor	100000
Elec. Eng.	Taylor	85000
Finance	Painter	120000
History	Painter	50000
Music	Packard	80000
Physics	Watson	70000

Why is it useful to have *dept_name* in both schemas?

University Example: Relations

So far we have the following schemas:

department(dept_name, building, budget)

instructor(ID, name, dept_name, salary)

course(course_id, title, dept_name, credits)

What other schemas might we want?

teaches(ID, course_id,sec_id, semester, year)

section(course_id, sec_id, semester,year, , building, room_number, time_slot_id)

student(ID, name, dept_name, tot_cred)

takes(ID, course_id, sec_id, semester, year, grade)

time_slot(time_slot_id, day, start_time, end_time)

…

How do we uniquely refer to a tuple or row in a schema?

Keys

Superkey: a set of one or more attributes that taken together *uniquely identify* a tuple in the relation.

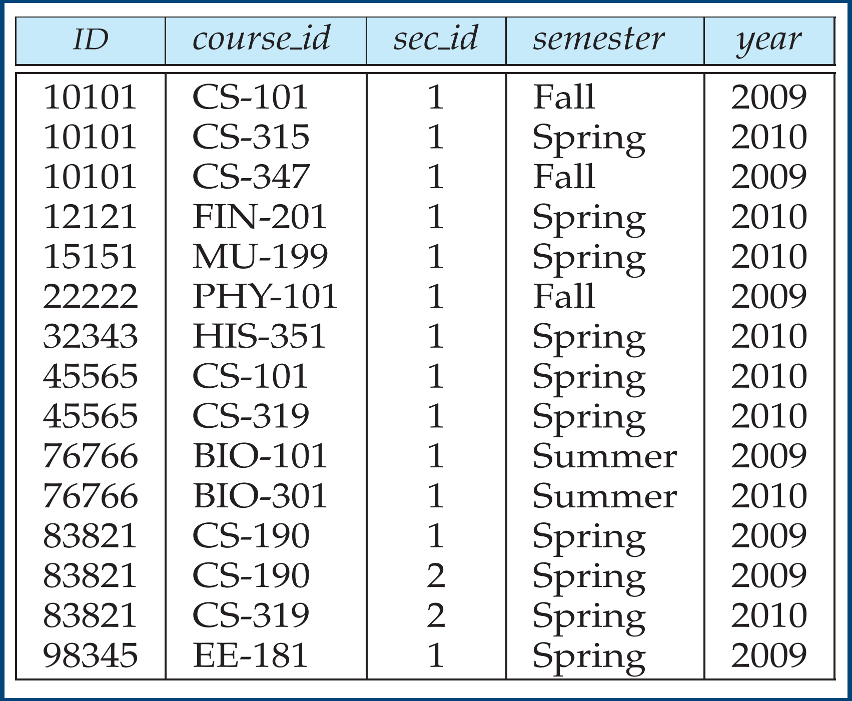
What are possible *superkeys* for the instructor relation?

instructor(ID, name, dept_name, salary)

Keys

Superkey: a set of one or more attributes that taken together *uniquely identify* a tuple in the relation.

What about the *teaches relation*?



ID	*course_id*	*sec_id*	*semester*	*year*
10101	CS-101	1	Fall	2009
10101	CS-315	1	Spring	2010
10101	CS-347	1	Fall	2009
12121	FIN-201	1	Spring	2010
15151	MU-199	1	Spring	2010
22222	PHY-101	1	Fall	2009
32343	HIS-351	1	Spring	2010
45565	CS-101	1	Spring	2010
45565	CS-319	1	Spring	2010
76766	BIO-101	1	Summer	2009
76766	BIO-301	1	Summer	2010
83821	CS-190	1	Spring	2009
83821	CS-190	2	Spring	2009
83821	CS-319	2	Spring	2010
98345	EE-181	1	Spring	2009

We are interested in superkey sets that are *minimal*.

Candidate Key

instructor(ID, name, dept_name, salary)

Superkeys for relation instructor:
{ID}, {name, dept_name}, {ID, name}

Candidate Key: A *minimal superkey*.

Q. Which of the above superkeys are *candidate keys*?

A. {ID}, {name, dept_name}

Primary Key: A *candidate key* chosen by the database designer to *distinguish* between tuples.

Next Week

Tutorials begin

Relational Model Continued

- Relational diagrams
- Relational operations
- Relational algebra

Intro to SQL and MySQL (tentative)