CS34800
Information Systems

Course Overview

Prof. Walid Aref

21 August, 2017

Why this Course?

- Managing Data is one of the primary uses of computers
- This course covers the foundations of organized data management
  - Database Management Systems (DBMS) of various flavors
- Learn how to
  - Model data in ways that protect data integrity
  - Store, retrieve, and manipulate data
  - Perform basic data analysis on large datasets

Course Outline

- Relational Databases
  1. Relational model overview
  2. Formal definitions, relational operations
  3. Query: SQL
- Database Design
  4. Entity-Relationship Model
  5. Relational Design
  6. Database Normalization
  7. Object Databases
  8. XML databases
- Integrity and Consistency
  9. Transactions
  10. Concurrency
  11. Constraints
- Advanced Topics
  12. Big Data: MapReduce, Hadoop, Spark
  13. Data Analysis / Data Mining
  14. Information Retrieval

Course Information

- Contact Information: Professor Aref
  - Office: LWSN 2116J, x4-1997.
  - Office hours: by e-mail appointments
  - aref@cs.purdue.edu

- Teaching Assistants:
  - Hassan, Mohamed S.: msaberab@purdue.edu
  - Ebaid, Amr: aebaid@purdue.edu
  - Maheshwari, Anshu: maheshw3@purdue.edu
  - Mahmood, Ahmed R.: amahmoo@purdue.edu
  - Al Mamun, Abdullal: mamuna@purdue.edu

- Course Web Page: https://www.cs.purdue.edu/homes/aref/Fall2017CS348/

Course Methodology

- Lectures to present the concepts
  - Unless otherwise noted, all the material you are expected to know will be covered in class
  - Interaction (questions/discussion/thinking) encouraged
- Reading will fill in the details. One of:
  - Database System Concepts, Avi Silberschatz, Henry F. Korth, and S. Sudarshan, McGraw-Hill (2010) ISBN 0-07-352332-1
  - Fundamentals of Database Systems, Ramez Elmasri and Shamkant B. Navathe, Pearson (2016) ISBN 9780133970777
- Homework and Projects get you to *understand* what you've read and heard
  - Around 4 written homeworks and 4 programming projects

Communication Tools (see course page)

- Blackboard
  - Turning in assignments, managing grades
- Piazza
- Walid's course home page

Evaluation and Grading

- Points earned as follows:
  - Programming/Project Assignments 30%
  - Homework Assignments 20%
  - Midterm 20%
  - Final 30%
  - Extra Credit Class Contribution 0-5%

- Late work penalized 10% per day

For more details see the course web page

Motivations for DB Technology
Automation of Information Systems

- The concept of “information system” is independent from IT technology: there are organizations, the goal of which is to manage information (like in the case of demographic services), and that have been in place for centuries
- DB Technology: For the automation of information systems

What is a Database?

- Collection of data, used to represent the information of interest to one or more applications in a given organization
  - Usually large
  - Organized for rapid search and retrieval
- Database Management System (DBMS): Tool to ease construction of databases
  - (Vendor) definition of database: Collection of data managed by a DBMS
- Desirable Properties:
  - Persistent Storage

    A File System does this
  - Query Interface

    Information retrieval system
  - Transaction Management

University Database Example

- Application program examples
  - Add new students, instructors, and courses
  - Register students for courses, and generate class rosters
  - Assign grades to students, compute grade point averages (GPA) and generate transcripts
- In the early days, database applications were built directly on top of file systems

DBMS vs file system – an example

- Consider a company that needs to maintain information about its employees and its departments. Suppose that applications, managing data on employees and departments, directly use the file systems for storing and retrieving data
- According to such approach, the data concerning employees and department are stored in records collected in files. There is a file for the employee records and a file for the department records

DBMS vs file system – an example

- Assume that the following application programs are available:
  - A program to modify the salary of a given employee
  - A program to modify the department of a given employee
  - A program to insert and remove employee records
  - A program printing the list of all employee according to the lexicographic order

DBMS vs file system – an example

- The approach based on the use of file systems has the following drawbacks
  - Data redundancy and inconsistency
  - Difficulties in data access
  - Problems with concurrent accesses
  - Problems of fine-grained, content-based access control
  - Problems of data integrity

Drawbacks of using file systems to store data

- Data redundancy and inconsistency
  - Multiple file formats, duplication of information in different files
- Difficulty in accessing data
  - Need to write a new program to carry out each new task
- Data isolation
  - Multiple files and formats
- Integrity problems
  - Integrity constraints (e.g., account balance > 0) become “buried” in program code rather than being stated explicitly
  - Hard to add new constraints or change existing ones

Drawbacks of using file systems to store data (Cont.)

- Atomicity of updates
  - Failures may leave database in an inconsistent state with partial updates carried out
  - Example: Transfer of funds from one account to another should either complete or not happen at all
- Concurrent access by multiple users
  - Concurrent access needed for performance
  - Uncontrolled concurrent accesses can lead to inconsistencies
    - Example: Two people reading a balance (say 100) and updating it by withdrawing money (say 50 each) at the same time
- Security problems
  - Hard to provide user access to some, but not all, data

Database systems offer solutions to all the above problems

History of Database Systems

- 1950s and early 1960s:
  - Data processing using magnetic tapes for storage
    - Tapes provided only sequential access
  - Punched cards for input
- Late 1960s and 1970s:
  - Hard disks allowed direct access to data
  - Network and hierarchical data models in widespread use
  - Ted Codd defines the relational data model
    - Would win the ACM Turing Award for this work
    - IBM Research begins System R prototype
    - UC Berkeley begins Ingres prototype
  - High-performance (for the era) transaction processing

History (cont.)

- 1980s:
  - Research relational prototypes evolve into commercial systems
    - SQL becomes industrial standard
  - Parallel and distributed database systems
  - Object-oriented database systems
- 1990s:
  - Large decision support and data-mining applications
  - Large multi-terabyte data warehouses
  - Emergence of Web commerce
- Early 2000s:
  - XML and XQuery standards
  - Automated database administration
- Later 2000s:
  - Giant data storage systems
    - Google BigTable, Yahoo PNuts, Amazon, ..

Goals of a DBMS

Data Integration	$\longrightarrow$	Enhances the accessibility of data, reduces redundancies and inconsistencies
Data Independency	$\longrightarrow$	Simplifies the development of new applications, and the maintainance of existing applications
Centralized Data Control	$\longrightarrow$	Assures data quality, confidentiality, and integrity

What are we studying?

- Methods to build databases
  - Data modeling
  - Query languages
  - Data integrity, confidentiality, consistency

  While we emphasize the relational model, we will look at other models/approaches as well
- ~~Methods to build DBMSs~~
  - Take CS44800

Motivations for DB Technology *Information and Data*

- Data are elementary facts that need to be interpreted in order to convey information
- Example:
  - Consider the byte 00000011. What does this tell you?
  - 3?
  - <Control-C>?
  - End of Text?
- By contrast, saying that 3 is the number of credits of CS348 provides some information

Motivations for DB Technology
Information and Data

- Computer systems represent information through data
- *Data* represents information. *Information* is the (subjective) interpretation of data
- ***Data* -** Physical phenomena chosen by convention to represent certain aspects of our conceptual and real world. The meaning we assign to data are called information. Data is used to transmit and store information and to derive new information by manipulating the data according to formal rules.

from:

P. Brinch Hansen. *Operating Systems Principles*.
Prentice-Hall, 1973.

Motivations for DB Technology
Information and Data

- One of the fundamental goals of a database management system (DBMS) is to provide <u>an interpretation context to a collection of data</u>, so that users can effectively access information encoded by this collection of data

Goals of a DBMS

- The basic mechanism that makes possible to integrate data is a logical and centralized definition of the data, referred to as database **schema**
- A schema specifies, through a high level formalism referred to as **data model**, the contents of the database

Goals of a DBMS

- Data independency is a second important goal. It guarantees that:
  - Changes to the physical data representation (for example, the use of a storage structure instead of another one) do not require changes to the existing applications – **physical data independency**
  - Changes to the logical representation of data do not require changes to the existing applications - **logical data independency**

Goals of a DBMS

- The third goal is achieved by introducing a centralized control on the data through the DBMS that provide centralized mechanisms for data access and manipulation

Services provided by a DBMS

Service	Description
Data definition	To specify the data to be stored
Data manipulation	To access data, to insert new data, to modify and delete existing data
Semantic integrity	To prevent the storage of incorrect data
Storage structures	To represent in secondary storage the data model constructs, to store efficienty store and retrieve data
Query optimization	To determine the most efficient strategy for data access
Access control	To protect data from unauthorized accesses
Recovery	To prevent data inconsistency in case of errors and failures
Data dictionary	To determine the data stored in a database and access their definitions

Data Models

- A data model is a “conceptual tool”, or *formalism*, that includes three fundamental components:
  - One or more data structures.
  - A notation to specify the data through the data structures of the model.
  - A set of operations for managing data; these operations are defined in terms of the data structures of the model.

Data Models

- A data model allows one to represent real-world entities of interest to a given set of applications
- It is thus useful to identity the basic concepts of such representation; relevant concepts include:
  - *Entity*: an "object" of the application domain
  - *Attribute*: a property of a given entity which meaningful, for the description of the application domain

    Each entity is thus characterized by one or more attributes; an attribute takes one or more values, referred to as *attribute values*, from a set of possible values; such set if referred to as *attribute domain*

Data Models

- A collection of tools for describing
  - Data
  - Data relationships
  - Data semantics
  - Data constraints
- Relational model
- Entity-Relationship data model (mainly for database design)
- Object-based data models (Object-oriented and Object-relational)
- Semistructured data model (XML)
- Other older models:
  - Network model
  - Hierarchical model

Data Models

- *Entity set*: it groups together a set of "objects" characterized by the same features; that is, it groups “similar” entities having the same attributes, even though these attributes do not necessarily have the same values
- A set of attributes the values of which uniquely identify an entity in a given entity set is a **key** for the entity set
- *Relationship*: a correspondence between the elements of two (or more) entity sets

Data Models: Example

- Consider the example of employees and departments:
  - **Entity**: John Smith, the department #30
  - **Entity set**: the set of all employees, the set of all departments
  - **Attribute**: employee name, salary, job, department number, department name
  - **Relationship**: the fact that John Smith works in the department #30

Data Models

- Each data model must answer two fundamental questions:
  - (a) how to represent the entities and their attributes
  - (b) how to represent the relationships

(a) Almost all models use structures such as the record; each component in a record represents an attribute

(b) Data models widely in this respect; relationships can be represented as:

- – specific structures, values, pointers (logical or physical)

Levels of Abstraction

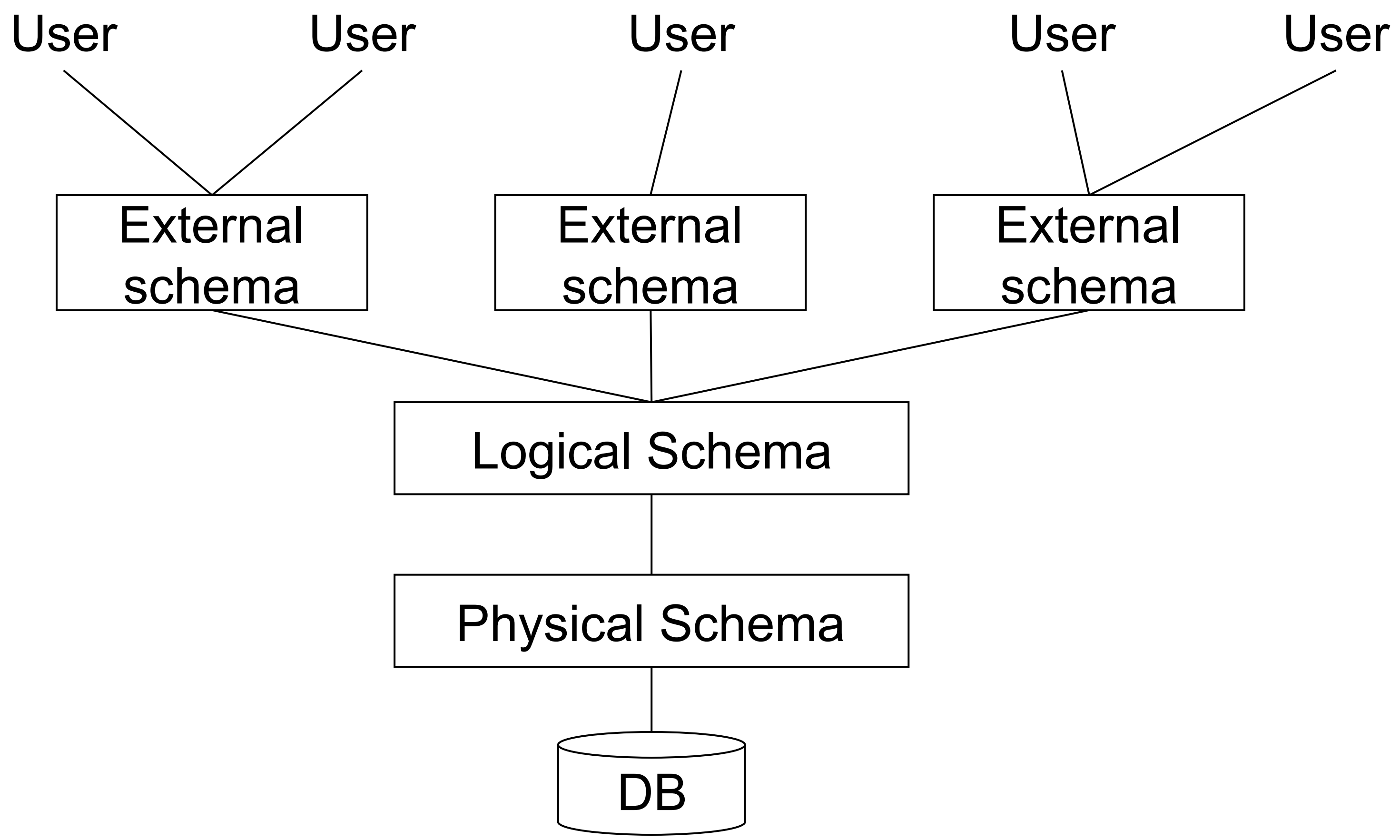
- **Physical level:** describes how a record (e.g., instructor) is stored.
- **Logical level:** describes data stored in database, and the relationships among the data.

  type *instructor* = **record**

  ID : string;
  name : string;
  dept_name : string;
  salary : integer;

  end;
- **View level:** application programs hide details of data types. Views can also hide information (such as an employee's salary) for security purposes.

Architecture ANSI/SPARC Three representation levels

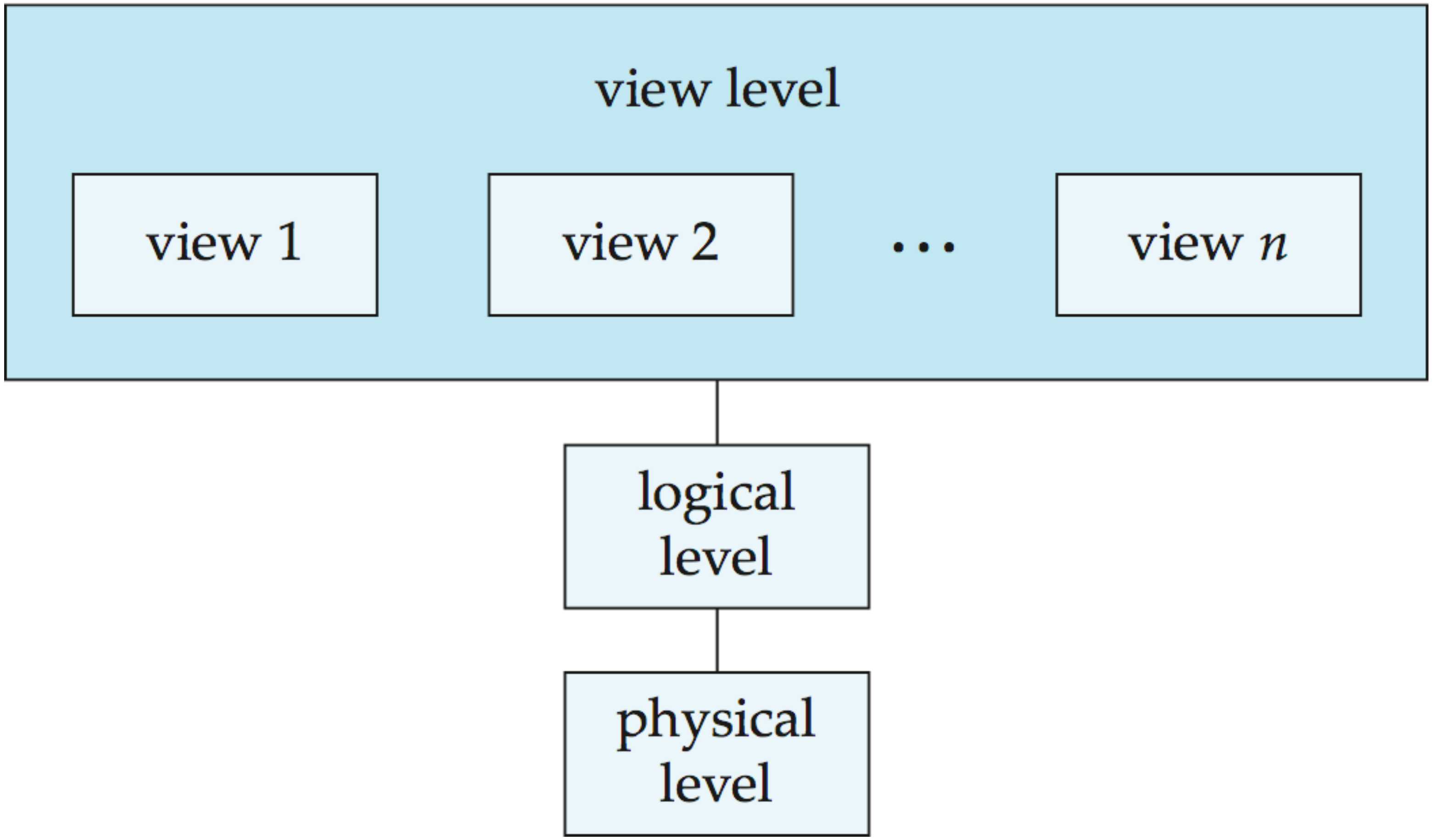


Logical Models: Evolution

- First generation (60):
  - Network data model – Codasyl
  - Hierarchical data model
- Relational data model (70)
- Post-relational data models:
  - Nested relational data models
  - Object-oriented data models
  - Deductive data models (e.g. Datalog and its extensions)
  - Object-relational data models
  - XML

View of Data

An architecture for a database system

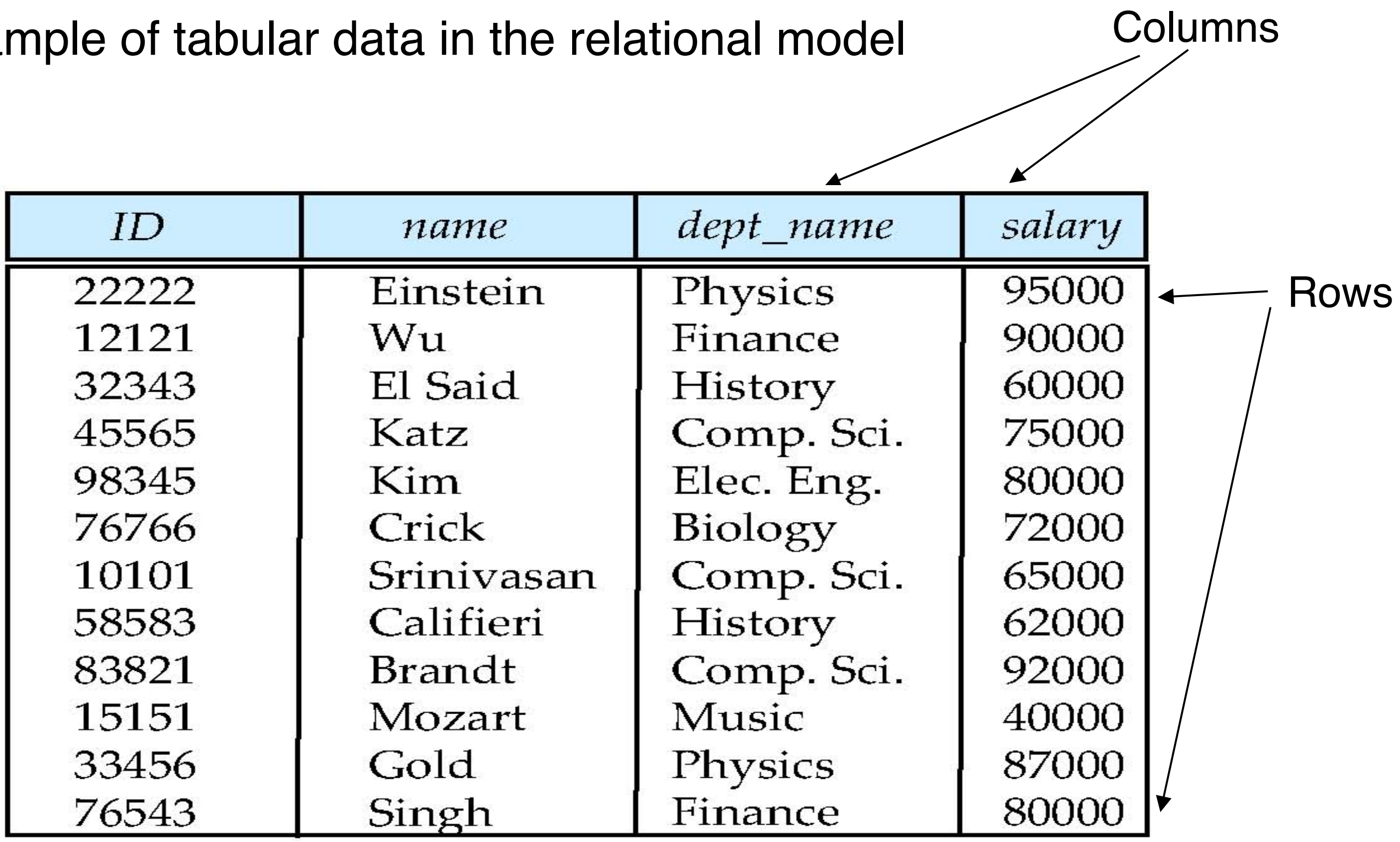


Instances and Schemas

- Similar to types and variables in programming languages
- **Logical Schema** – the overall logical structure of the database
  - Example: The database consists of information about a set of customers and accounts in a bank and the relationship between them
    - Analogous to type information of a variable in a program
- **Physical schema**– the overall physical structure of the database
- **Instance** – the actual content of the database at a particular point in time
  - Analogous to the value of a variable
- **Physical Data Independence** – the ability to modify the physical schema without changing the logical schema
  - Applications depend on the logical schema
  - In general, the interfaces between the various levels and components should be well defined so that changes in some parts do not seriously influence others.

Relational Model

- All the data is stored in various tables.
- Example of tabular data in the relational model


Columns
Rows

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

(a) The *instructor* table

A Sample Relational Database

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

(a) The *instructor* table

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

(b) The *department* table

Database Design

The process of designing the general structure of the database:

- Logical Design – Deciding on the database schema. Database design requires that we find a "good" collection of relation schemas.
  - Business decision – What attributes should we record in the database?
  - Computer Science decision – What relation schemas should we have and how should the attributes be distributed among the various relation schemas?
- Physical Design – Deciding on the physical layout of the database

Database Design (Cont.)

- Is there any problem with this relation?

ID	*name*	*salary*	*dept_name*	*building*	*budget*
22222	Einstein	95000	Physics	Watson	70000
12121	Wu	90000	Finance	Painter	120000
32343	El Said	60000	History	Painter	50000
45565	Katz	75000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000
76766	Crick	72000	Biology	Watson	90000
10101	Srinivasan	65000	Comp. Sci.	Taylor	100000
58583	Califieri	62000	History	Painter	50000
83821	Brandt	92000	Comp. Sci	Taylor	100000
15151	Mozart	40000	Music	Packard	80000
33456	Gold	87000	Physics	Watson	70000
76543	Singh	80000	Finance	Painter	120000

SQL

- The most widely used commercial language
- SQL is NOT a Turing machine equivalent language
- To be able to compute complex functions SQL is usually embedded in some higher-level language
- Application programs generally access databases through one of
  - Language extensions to allow embedded SQL
  - Application program interface (e.g., ODBC/JDBC) which allow SQL queries to be sent to a database

Data Definition Language (DDL)

- Specification notation for defining the database schema

  Example:

  ```
  create table instructor (
          ID          char(5),
          name        varchar(20),
          dept_name   varchar(20),
          salary      numeric(8,2))
  ```

- DDL compiler generates a set of table templates stored in a ***data dictionary***
- Data dictionary contains metadata (i.e., data about data)
  - Database schema
  - Integrity constraints
    - Primary key (ID uniquely identifies instructors)
  - Authorization
    - Who can access what

Data Manipulation Language (DML)

- Language for accessing and manipulating the data organized by the appropriate data model
  - DML also known as query language
- Two classes of languages
  - **Pure** – used for proving properties about computational power and for optimization
    - Relational Algebra
    - Tuple relational calculus
    - Domain relational calculus
  - **Commercial** – used in commercial systems
    - SQL is the most widely used commercial language

SQL, an interactive query language

Courses

Course-Name	Instructor	Room-Name
Database	Aref	DS1
Operating Syst.	Rodriguez	N3
Networks	Fahmy	N3
Security	Spafford	G

Rooms

Room-Name	Building	Floor
DS1	Recitation	1
N3	Recitation	1
G	Univ. Hall	2

```
SELECT Course-Name, Room-Name, Floor
FROM Courses, Rooms WHERE
Courses.Room-Name =
    Rooms.Room-Name
AND  Instructor = ‘Fahmy’;
```

Course-Name	Room-Name	Floor
Networks	N3	1
Security	G	2

Basic Query Structure

- A typical SQL query has the form:

  select $A_1, A_2, ..., A_n$
  from $r_1, r_2, ..., r_m$
  where P

  - A_i represents an attribute
  - R_i represents a relation
  - P is a predicate.
- The result of an SQL query is a relation.

The select Clause

- The **select** clause lists the attributes desired in the result of a query
  - corresponds to the projection operation of the relational algebra
- Example: find the names of all instructors:

  select *name*
  from *instructor*
- NOTE: SQL names are case insensitive (i.e., you may use upper- or lower-case letters.)
  - E.g., *Name* ≡ *NAME* ≡ *name*
  - Some people use upper case wherever we use bold font.

The select Clause (Cont.)

- SQL allows duplicates in relations as well as in query results.
- To force the elimination of duplicates, insert the keyword **distinct** after select**.**
- Find the department names of all instructors, and remove duplicates

  select distinct *dept_name*
  from *instructor*

- The keyword **all** specifies that duplicates should not be removed.

  select all *dept_name*
  from *instructor*

The select Clause (Cont.)

- An asterisk in the select clause denotes "all attributes"

  select *
  from *instructor*

- An attribute can be a literal with no **from** clause

  select '437'

  - Results is a table with one column and a single row with value "437"
  - Can give the column a name using:

    select '437' **as** *FOO*

- An attribute can be a literal with **from** clause

  select 'A'
  from *instructor*

  - Result is a table with one column and *N* rows (number of tuples in the *instructors* table), each row with value "A"

The select Clause (Cont.)

- The **select** clause can contain arithmetic expressions involving the operation, +, –, ∗, and /, and operating on constants or attributes of tuples.
  - The query:

    select *ID, name, salary/12*
    from *instructor*

    would return a relation that is the same as the *instructor* relation, except that the value of the attribute *salary* is divided by 12.
  - Can rename "s*alary/12"* using the **as** clause:

    select *ID, name, salary/12* **as** *monthly_salary*

The where Clause

- The **where** clause specifies conditions that the result must satisfy
  - Corresponds to the selection predicate of the relational algebra.
- To find all instructors in Comp. Sci. dept

  select *name*
  from *instructor*
  where *dept_name* = 'Comp. Sci.'

- Comparison results can be combined using the logical connectives **and, or,** and **not**
  - To find all instructors in Comp. Sci. dept with salary > 80000

  select *name*
  from *instructor*
  where *dept_name* = 'Comp. Sci.' **and** *salary* > 80000

- Comparisons can be applied to results of arithmetic expressions.

String Operations

- SQL includes a string-matching operator for comparisons on character strings. The operator **like** uses patterns that are described using two special characters:
  - percent (%). The % character matches any substring.
  - underscore (_). The _ character matches any character.
- Find the names of all instructors whose name includes the substring "dar".

  select *name*
  from *instructor*
  where *name* **like** '%dar%'

- Match the string "100%"

  like '100 \%' **escape** '\'

  in that above we use backslash (\) as the escape character.

String Operations (Cont.)

- Patterns are case sensitive.
- Pattern matching examples:
  - 'Intro%' matches any string beginning with "Intro".
  - '%Comp%' matches any string containing "Comp" as a substring.
  - '_ _ _' matches any string of exactly three characters.
  - '_ _ _ %' matches any string of at least three characters.

- SQL supports a variety of string operations such as
  - concatenation (using "||")
  - converting from upper to lower case (and vice versa)
  - finding string length, extracting substrings, etc.

Where Clause Predicates

- SQL includes a **between** comparison operator
- Example: Find the names of all instructors with salary between \$90,000 and \$100,000 (that is, ≥ \$90,000 and ≤ \$100,000)
  - **select** *name*
    from *instructor*
    where *salary* **between** 90000 **and** 100000

Ordering the Display of Tuples

- List in alphabetic order the names of all instructors

  select distinct *name*
  from *instructor*
  order by *name*

- We may specify **desc** for descending order or **asc** for ascending order, for each attribute; ascending order is the default.
  - Example: **order by** *name* **desc**
- Can sort on multiple attributes
  - Example: **order by** *dept_name, name*

And now, some examples

- Ideas on a favorite table?
- https://www.cs.purdue.edu/undergraduate/curriculum/bachelor.html