Module - I

Lecture-03

- Database-System Applications
- Purpose of Database Systems
- View of Data
- Database Languages

Database Applications Examples

- Enterprise Information
  - ✓ Sales: customers, products, purchases
  - ✓ Accounting: payments, receipts, assets
  - ✓ Human Resources: Information about employees, salaries, payroll taxes.
- Manufacturing: management of production, inventory, orders, supply chain.

Database Applications Examples

- Banking and finance
  - ✓ customer information, accounts, loans, and banking transactions.
  - ✓ Credit card transactions
  - ✓ Finance: sales and purchases of financial instruments (e.g., stocks and bonds; storing real-time market data
- Universities: registration, grades

Database Applications Examples

- Airlines: reservations, schedules
- Telecommunication: records of calls, texts, and data usage, generating monthly bills, maintaining balances on prepaid calling cards
- Web-based services:
  - Online retailers: order tracking, customized recommendations
  - Online advertisements
- Document databases
- Navigation systems: For maintaining the locations of varies places of interest along with the exact routes of roads, train systems, buses, etc.

Purpose of Database Systems

In the early days, database applications were built directly on top of file systems, which leads to:

- Data redundancy and inconsistency: data is stored in multiple file formats resulting induplication 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

Purpose of Database Systems

- 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
- 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

Purpose of Database Systems

- Concurrent access by multiple users
  - ✓ Concurrent access needed for performance
  - ✓ Uncontrolled concurrent accesses can lead to inconsistencies
    - Ex: 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

University Database Example

- Data consists of information about:
  - ✓ Students
  - ✓ Instructors
  - ✓ Classes
- 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

View of Data

- A database system is a collection of interrelated data and a set of programs that allow users to access and modify these data.
- A major purpose of a database system is to provide users with an abstract view of the data.
  - Data models
    - ✓ A collection of conceptual tools for describing data, data relationships, data semantics, and consistency constraints.
  - Data abstraction
    - ✓ Hide the complexity of data structures to represent data in the database from users through several levels of data abstraction.

Data Models

- Data Model is a collection of tools for describing
  - ✓ Data
  - ✓ Data relationships
  - ✓ Data semantics
  - ✓ Data constraints

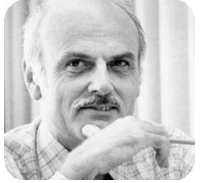
Data Models

Types of Data Models

- Relational model
- Entity-Relationship data model (mainly for database design)
- Object-based data models (Object-oriented and Object-relational)
- Semi-structured data model (XML)
- Other older models:
  - ✓ Network model
  - ✓ Hierarchical model

Relational Model

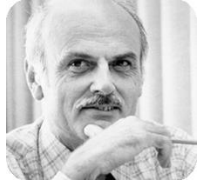
- The relational data model was introduced by E. F. Codd in 1970. Currently, it is the most widely used data model.
- All the data is stored in various tables.



Edgar Frank "Ted" Codd
Turing Award 1981

Relational Model

- Example of tabular data in the relational model



Edgar Frank "Ted" Codd
Turing Award
1981

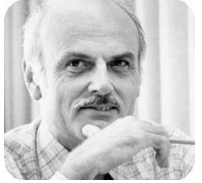
Columns

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

Rows

(a) The *instructor* table

A Sample Relational Database



Edgar Frank "Ted" Codd
Turing Award
1981

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

Levels of Abstraction

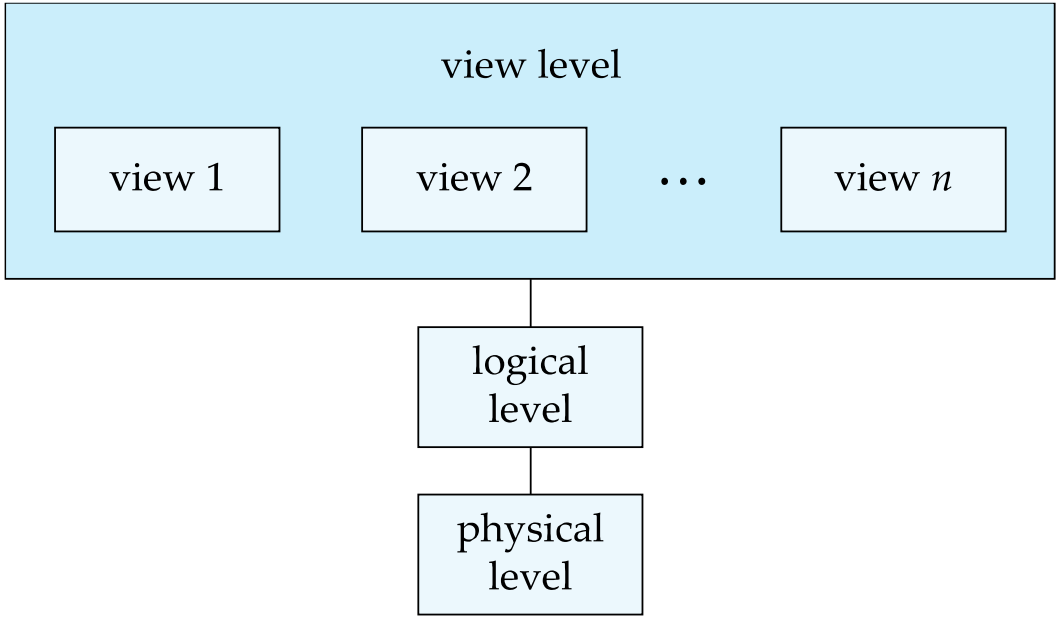
- Physical level: describes how a record (e.g., instructor) is stored.
- Logical level: describes data stored in database, and the relationships among the data.

```
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.

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
    - ✓ Similar 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
  - Similar to the value of a variable

Physical Data Independence

- 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.

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 Definition Language (DDL)

- 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 updating the data organized by the appropriate data model
  - DML also known as query language
- There are basically two types of data-manipulation language
  - Procedural DML -- require a user to specify what data are needed and how to get those data.
  - Declarative DML -- require a user to specify what data are needed without specifying how to get those data.

Data Manipulation Language (DML)

- Declarative DMLs are usually easier to learn and use than are procedural DMLs.
- Declarative DMLs are also referred to as non-procedural DMLs
- The portion of a DML that involves information retrieval is called a query language.

SQL (Structured Query Language)

- SQL query language is nonprocedural.
- A query takes as input several tables (possibly only one) and always returns a single table.
- Example to find all instructors in Comp. Sci. dept

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

SQL (Structured Query 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

Database Access from Application Program

- Non-procedural query languages such as SQL are not as powerful as a universal Turing machine.
- SQL does not support actions such as input from users, output to displays, or communication over the network.
- Such computations and actions must be written in a host language, such as C/C++, Java or Python, with embedded SQL queries that access the data in the database.
- Application programs -- are programs that are used to interact with the database in this fashion.

Copyright Note

Database System Concepts by *Avi Silberschatz, Henry F. Korth* and *S. Sudarshan.* 7th Edition

Slides Link:

https://www.db-book.com/slides-dir/index.html

Solutions to Practice Exercises

https://www.db-book.com/Practice-Exercises/index-solu.html

Online SQL interpreter

https://www.db-book.com/university-lab-dir/sqljs.html

Sample lab exercises and term projects

https://www.db-book.com/university-lab-dir/lab-exercises-projects.html