Data Modeling and Databases II

Database System Concepts and Architecture

Lecture 3 - Alexey Kanatov

Content

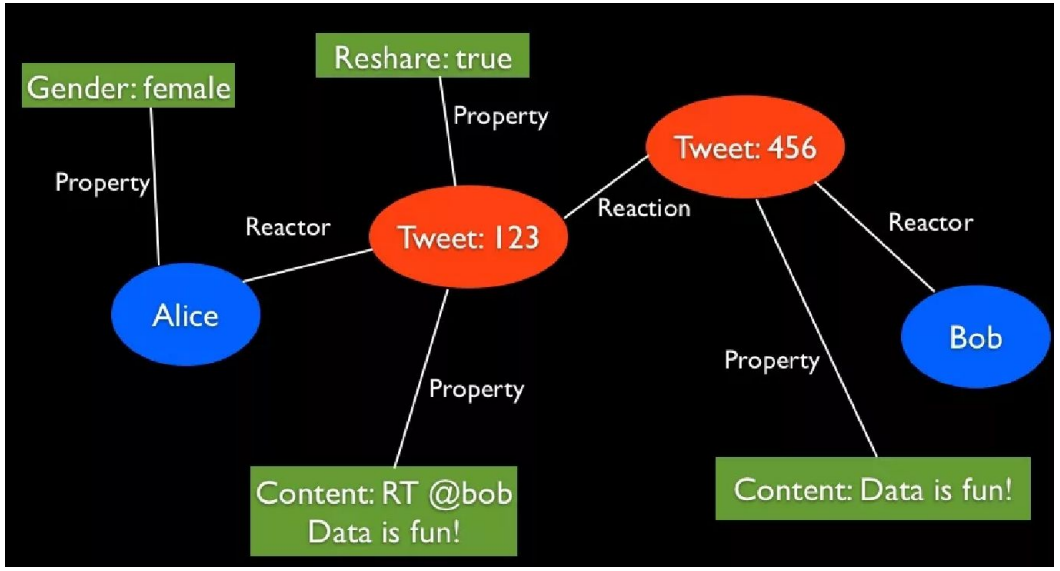
- Data models, schemas and instances
- Three-schema architecture and data independence
- Database language and interfaces
- The database system environment
- Centralised and Client/Server architectures for DBMSs
- Classification of DB management systems

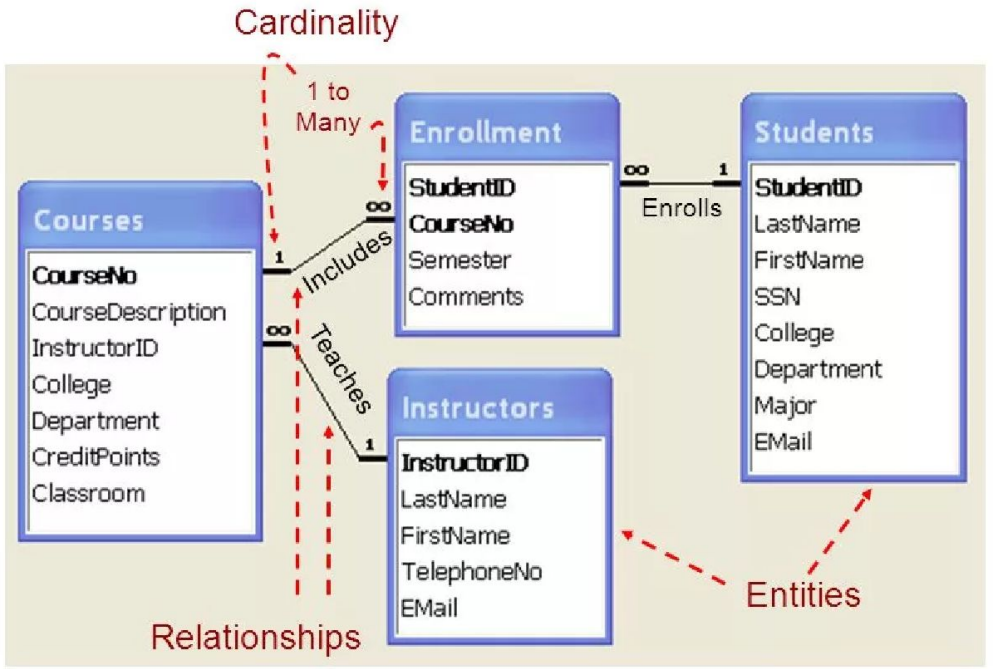
What is a data model?

Data Model

- An abstract model
- It organizes elements of data
- It standardizes how they relate to one another
- It standardizes how they relate to properties of the real world entities

Data model structures the world, that is how human brains operate.

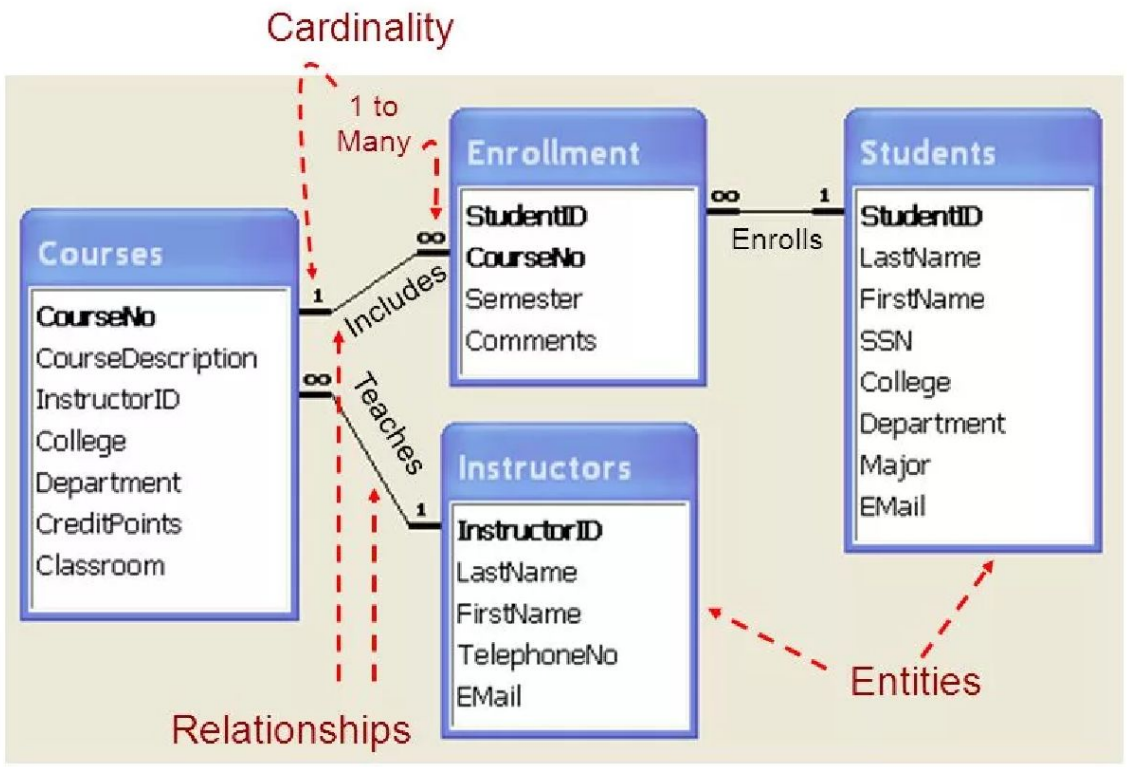




What is a database model?

Database Model

- Collection of concepts that can be used to describe the structure of a database.
- By *structure of a database* we mean the data types, relationships, and constraints that apply to the data. Most data models also include a set of **basic operations** for specifying retrievals and updates on the database.



Data models classification

- High-level: conceptual data model (user, biz. analyst) (entities, attributes, relations)
- Low-level: physical data model (technical) (files, indices, sorting)
- Representational (implementation) data model (user and technical folks may use together) :
  - relational (record-based) data model
  - object data model (Object Data Management Group, C++, Java and Smalltalk binding)
  - self-describing data models (XML, NOSQL, key-value stores)
  - hierarchical and network data models (used in the past)

Schemas, Instances, and Database State

2 items to be distinguished:

- Database description - database schema
- Database itself – raw data (cells, attribute (columns), tuples (rows)), data state is called a snapshot, set of instances
- Consistency: snapshot matches the schema!

STUDENT

Name	Student_number	Class	Major

COURSE

Course_name	Course_number	Credit_hours	Department

Fixed

STUDENT

Name	Student_number	Class	Major
Smith	17	1	CS
Brown	8	2	CS

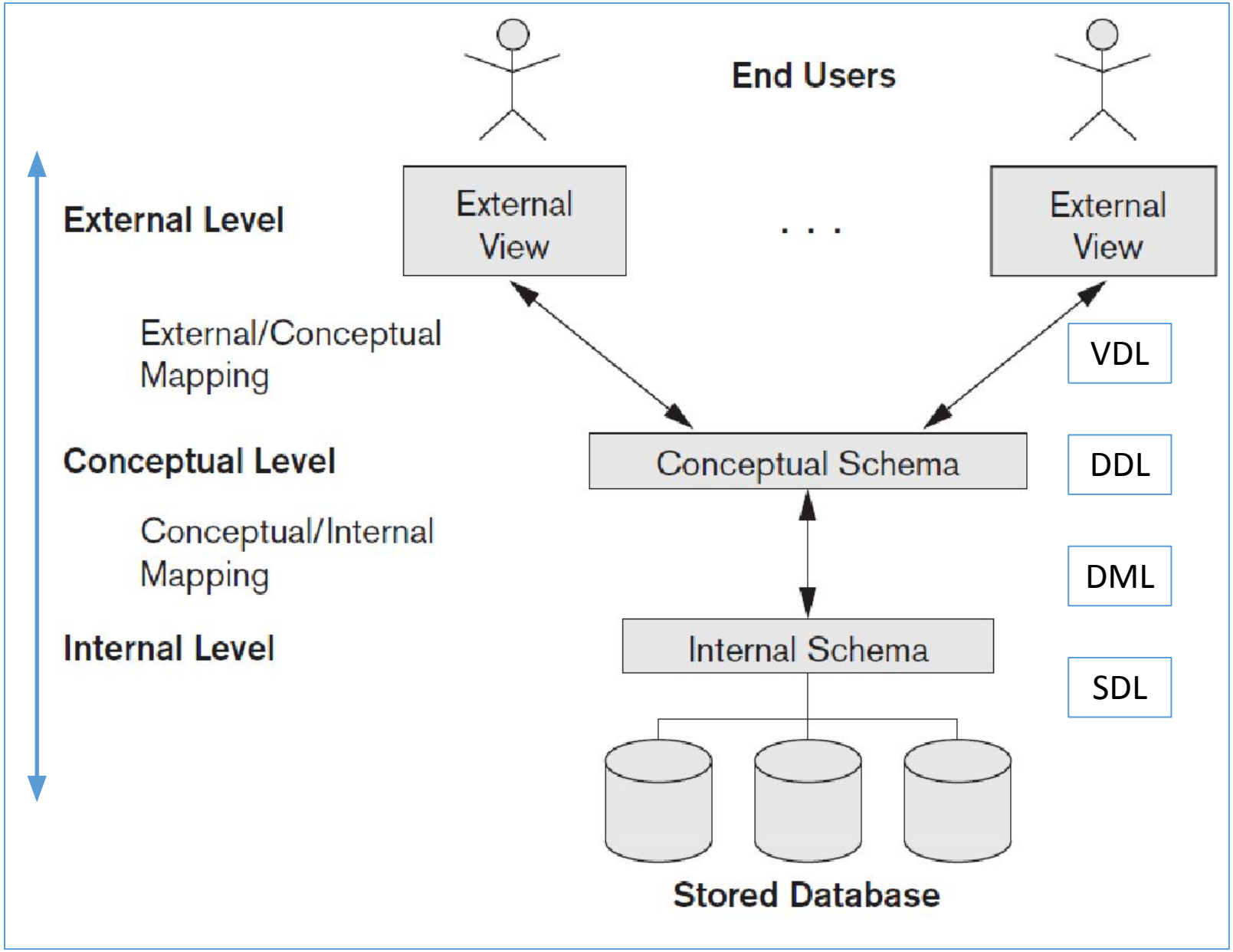
COURSE

Course_name	Course_number	Credit_hours	Department
Intro to Computer Science	CS1310	4	CS
Data Structures	CS3320	4	CS
Discrete Mathematics	MATH2410	3	MATH
Database	CS3380	3	CS

Regularly updated

Three-Schema Architecture and Data Independence

- **Internal schema** describes the physical storage structure. It uses a physical data model and describes the complete details of data storage and access paths for the database.
- **Conceptual schema** describes the structure of the whole database for a community of users. It hides the details of physical storage structures and focuses on describing entities, data types, relationships, user operations, and constraints. Usually, a representational data model is used to describe the conceptual schema when a database system is implemented.
- There are **external schemas (user views)**. Each external schema describes the part of the database that a particular user group is interested in. Each external schema is typically implemented using a representational data model.
- Mappings and data isolation



Three-Schema Architecture and Data Independence

Note: three schemas are only *descriptions* of data. The actual data is stored at the physical level only!

Data Independence:

- **Logical data independence** is the capacity to change the conceptual schema without having to change external schemas or application programs. (new column added)
- **Physical data independence** is the capacity to change the internal schema without having to change the conceptual schema. (new index created)

Database Languages

- **Data definition/description language (DDL) has** a syntax like a computer programming language for defining data structures, especially database schemas. SQL example =>

```
CREATE TABLE employees (
    id              INTEGER         PRIMARY KEY,
    first_name      VARCHAR(50)     not null,
    last_name       VARCHAR(75)     not null,
    fname           VARCHAR(50)     not null,
    dateofbirth     DATE            not null
);
```

- **Storage definition language** (**SDL**): In most relational DBMSs today, there is no specific language that performs the role of SDL. Instead, the internal schema is specified by a combination of functions, parameters, and specifications related to storage of files.
- **View definition language** (**VDL**) is to specify user views and their mappings to the conceptual schema
- **Data manipulation language** (**DML**) is to provide typical data manipulations - retrieval, insertion, deletion, and modification of the data. SQL example =>
  - `SELECT ... FROM ... WHERE ...`
    - `SELECT ... INTO ...`
  - `INSERT INTO ... VALUES ...`
  - `UPDATE ... SET ... WHERE ...`
  - `DELETE FROM ... WHERE ...`

```
INSERT INTO employees (first_name, last_name, fname) VALUES ('John', 'Capita', 'xcapit00');
```

- In current DBMSs these languages are *not considered distinct languages*; instead one comprehensive integrated language is used for conceptual schema definition, view definition, and data manipulation. SQL is a combination of DDL, VDL, and DML, as well as statements for constraint specification, schema evolution, and other features.

DBMS Interfaces

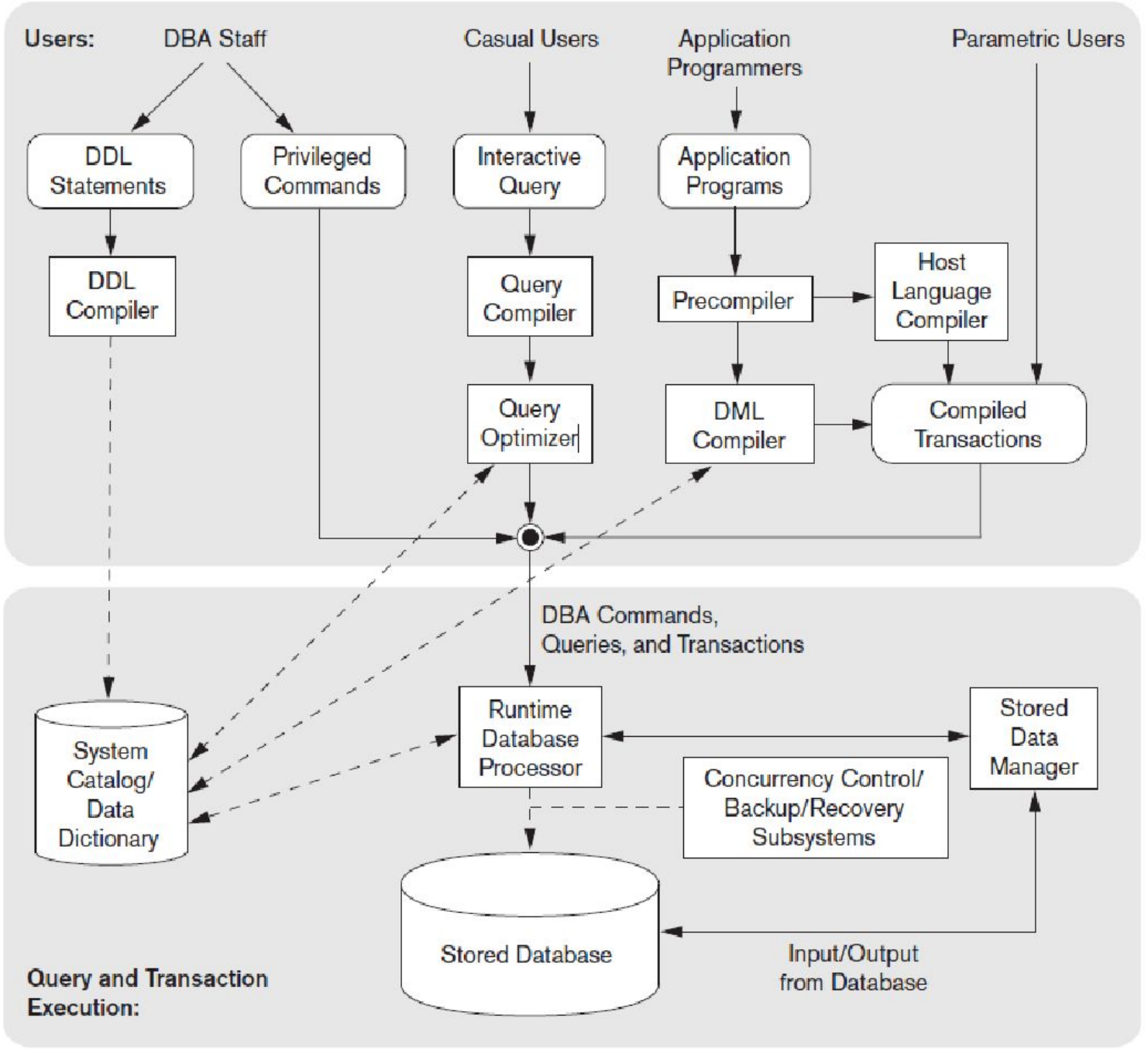
- Menu-based Interfaces for Web Clients or Browsing
  - Selection-based: query is composed step-by step by picking options from a menu that is displayed by the system. No need to memorize the specific commands and syntax of a query language
- Apps for Mobile Devices
  - Tap! ☺️
  - Tickets, banks, weather, etc.
- Forms-based Interfaces
  - Enter data into forms
  - Forms are developed for specific users and specific tasks
- Graphical User Interfaces
  - Draw!
  - GUI typically displays a schema to the user in diagrammatic form
- Natural Language Interfaces
  - Free text request
  - If request interpretation is successful then high-level query corresponding to the natural language request generated and submitted to the DBMS for processing

DBMS Interfaces

- Keyword-based Database Search
  - Search engines – Google, Yandex, Yahoo, …
  - Enter a search phrase – get the list of matching documents
- Speech Input and Output
  - Speak and listen!
- Interfaces for Parametric Users
  - Special interface for each known class of naive users is designed and implemented. Usually a small set of abbreviated commands is included, with the goal of minimizing the number of keystrokes required for each request.
- Interfaces for the DBA (database administrators)
  - commands for creating accounts, setting system parameters, granting account authorization, changing a schema, and reorganizing the storage structures of a database.

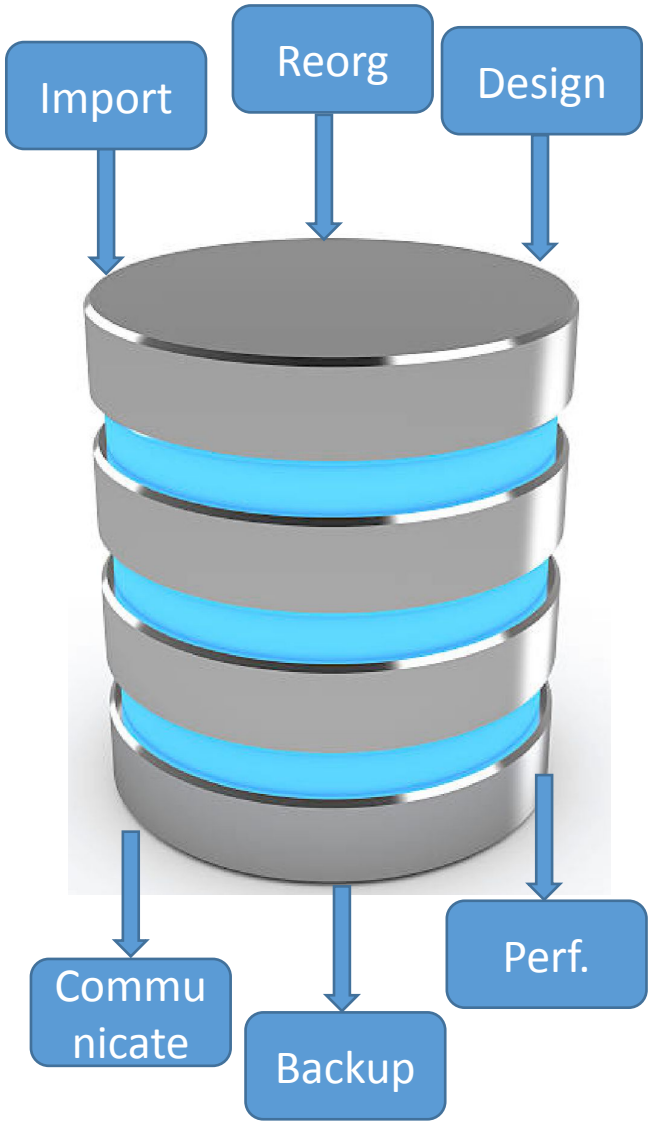
The Database System Environment (typical)

- Upper part:
  - 4 kinds of usage modes
  - Database languages (SQL) and programming languages
- Lower part:
  - Core DBMS functionality
  - Runtime database processor has binary API which deals with actual data



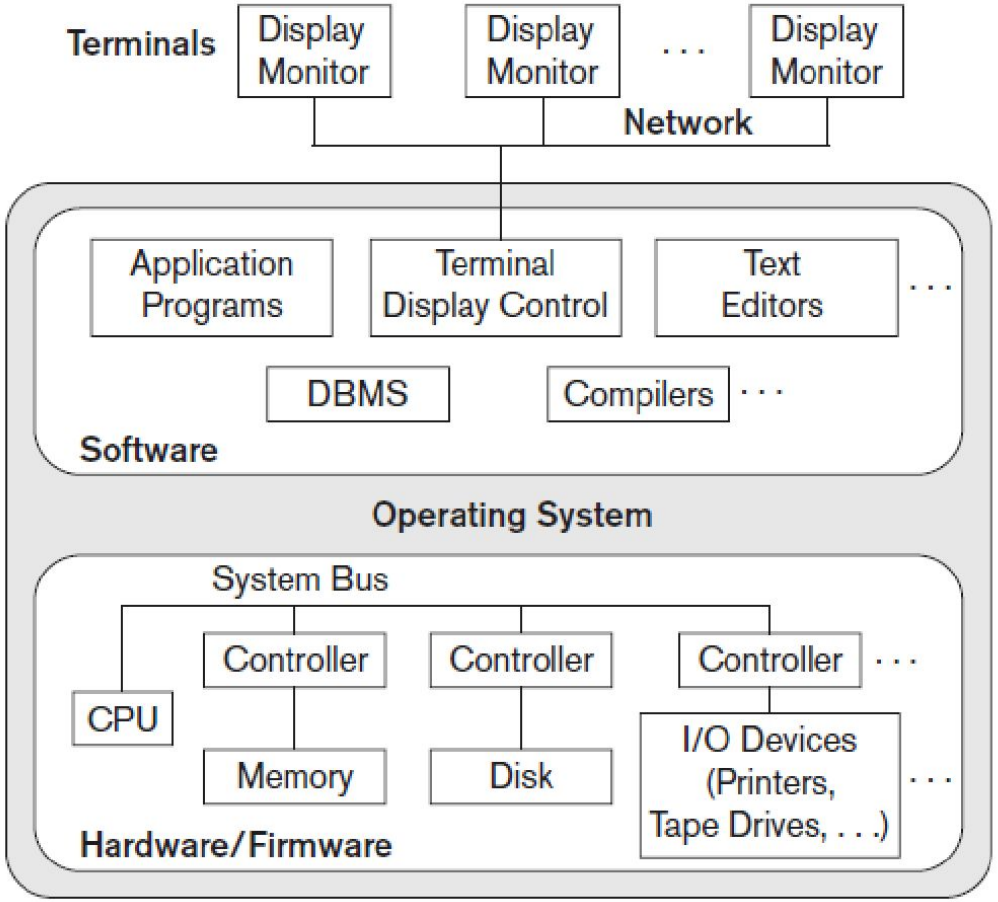
The Database System Environment

- **Loading. (Data import)** A loading utility is used to load existing data files—such as text files or sequential files—into the database. Usually, the current (source) format of the data file and the desired (target) database file structure are specified to the utility, which then automatically reformats the data and stores it in the database.
    - **Conversion tools** generate the appropriate loading programs, given the existing source and target database storage descriptions (internal schemas).
- **Backup.** A backup utility creates a backup copy of the database, usually by dumping the entire database onto tape or other mass storage medium. The backup copy can be used to restore the database in case of catastrophic disk failure. Incremental backups are also often used, where only changes since the previous backup are recorded. Incremental backup is more complex but saves storage space.
- **Database storage reorganization.** This utility can be used to reorganize a set of database files into different file organizations and create new access paths to improve performance.
- **Performance monitoring.** Such a utility monitors database usage and provides statistics to the DBA. The DBA uses the statistics in making decisions such as whether to reorganize files or whether to add or drop indexes to improve performance.
- **Application development environments**, such as PowerBuilder (Sybase) or JBuilder (Borland), have been quite popular. These systems provide an environment for developing database applications and include facilities that help in many facets of database systems, including database design, GUI development, querying and updating, and application program development.
- Database design (CASE)
- The DBMS also needs to interface with **communications software**, whose function is to allow users at locations remote from the database system site to access the database through computer terminals, workstations, or personal computers.



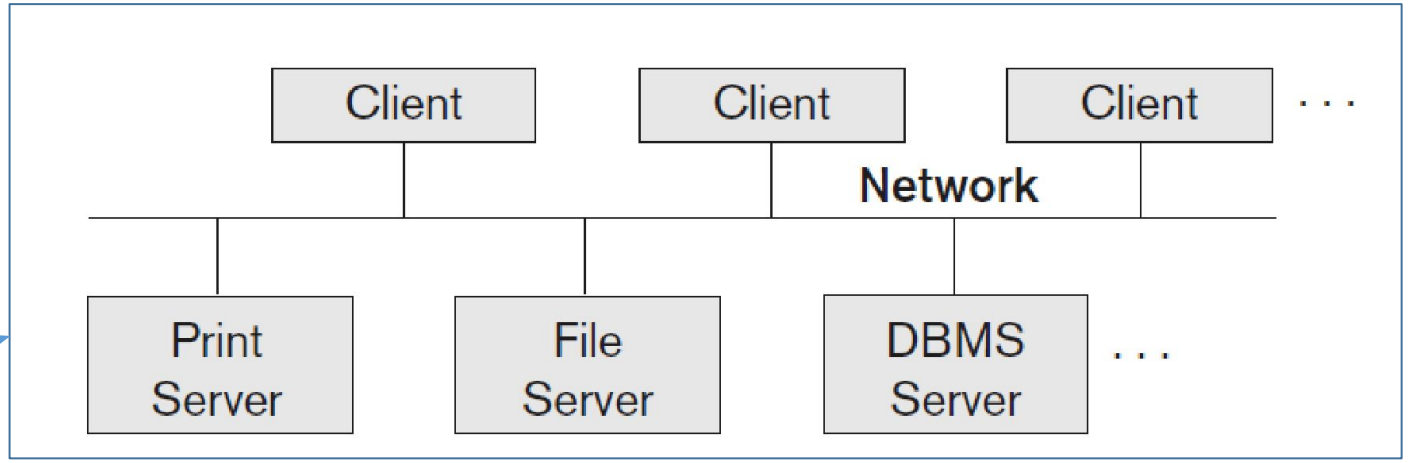
Centralized DBMSs Architecture

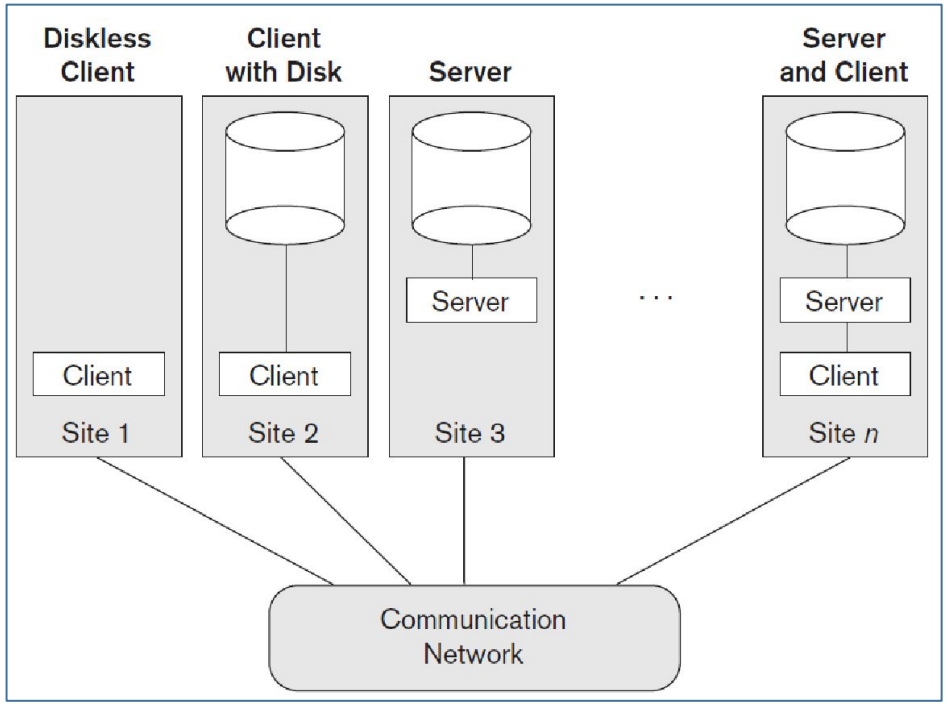
- One central hub serves many clients.
- Terminals are cheap and have nearly no processing power.



Basic Client/Server Architectures

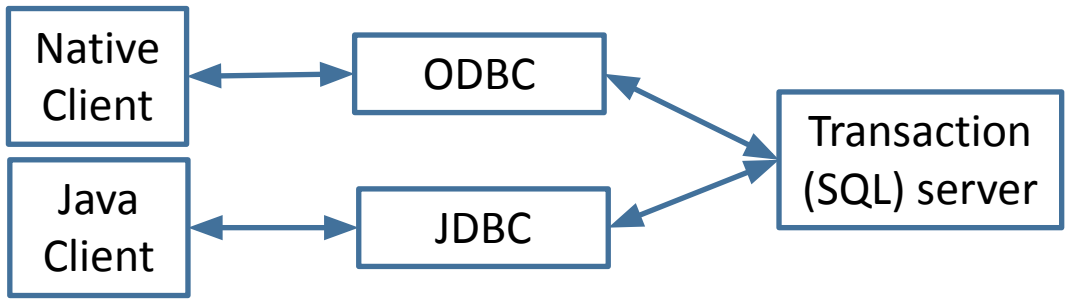
- The idea is to define specialized servers with specific functionalities: print server, file server, Web server, …
- 2 tier architecture: logical and physical view





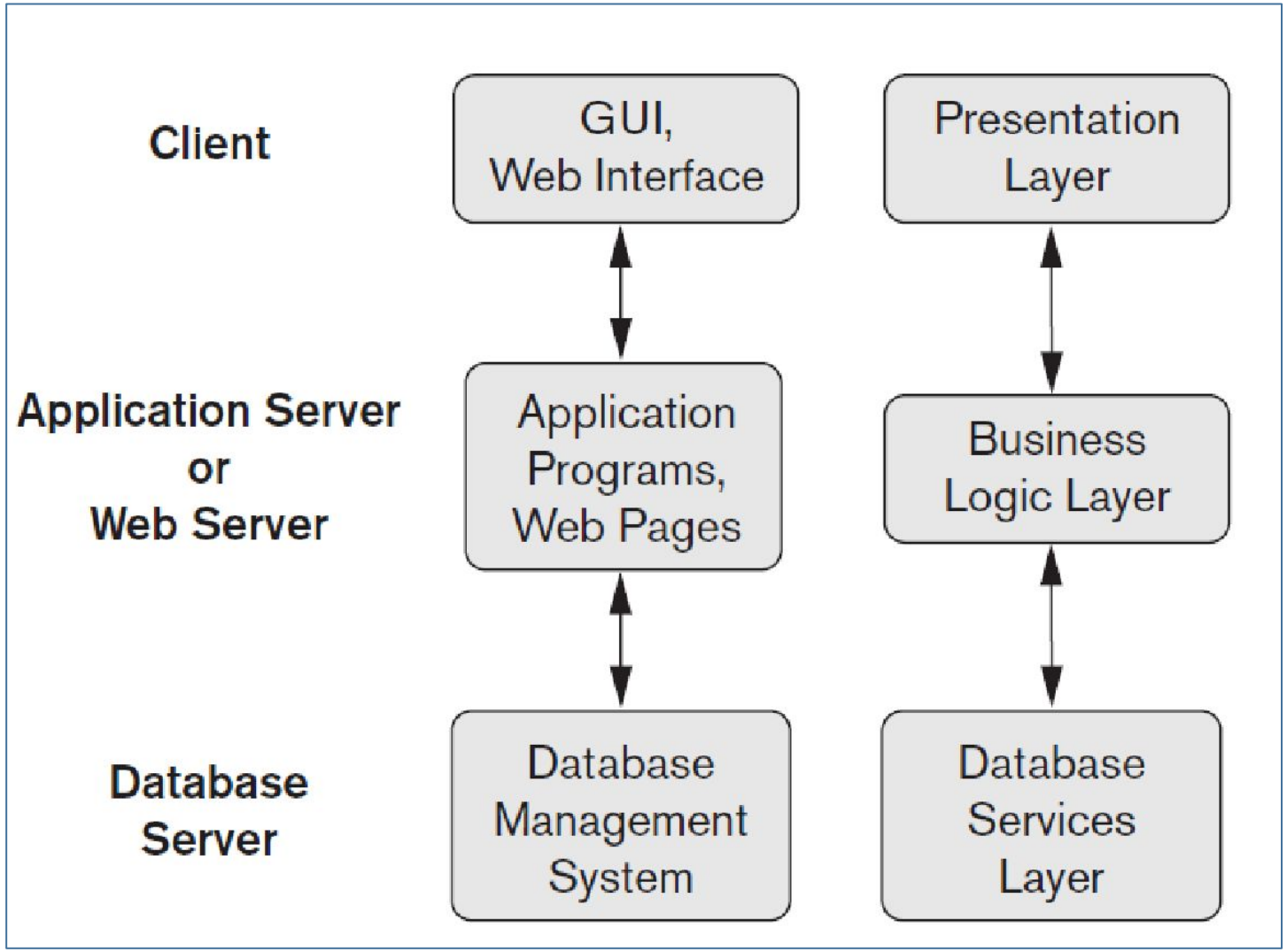
Client/Server Architectures - programming

- A standard called **Open Database Connectivity** (**ODBC**) provides an **application programming interface** (**API**), which allows client-side programs to call the DBMS, as long as both client and server machines have the necessary software installed. Most DBMS vendors provide ODBC drivers for their systems. A client program can actually connect to several RDBMSs and send query and transaction requests using the ODBC API, which are then processed at the server sites. Any query results are sent back to the client program, which can process and display the results as needed.
- A related standard for the Java programming language, called **JDBC**, has also been defined. This allows Java client programs to access one or more DBMSs through a standard interface.



Three-Tier and n-Tier Architectures for Web Applications

- The intermediate layer or **middle tier** is called the **application server** or the **Web server**, depending on the application. This server plays an intermediary role by running application programs and storing business rules that are used to access data from the database server.
- More layers can be introduced for finer granularity (*n*-tier architectures) Typically, the business logic layer is divided into multiple layers. Besides distributing programming and data throughout a network, *n*-tier applications afford the advantage that any one tier can run on an appropriate processor or operating system platform and can be handled independently.



Classification of Database Management Systems

The main classification - data model

- Relational data model (SQL systems)
- Object data model
- Document-based (JSON), graph-based, column-based, and key-value data models
- Legacy applications still work databases based on the hierarchical and network (COBOL) data models
- Experimental DBMSs are based on a tree-structured data model (XML)

Number of users:

- Single-user at a time
- Multiuser

Number of sites:

- Centralized
- Distributed (big data): homogeneous vs. heterogeneous (the same/different DBMS)

Cost:

- Free (MySQL, PostgreSQL)
- Commercial

Purpose – types of access path:

- General purpose
- Special purpose (online transaction processing (OLTP) systems)

Summary

- Data model defined:
  - High-level or conceptual data models (based on entities and relationships)
  - Low-level or physical data models
  - Representational or implementation data models (record-based, object-oriented)
- Schema, or description of a database is different from the database itself.
- Three-schema DBMS architecture presented:
  - An internal schema describes the physical storage structure of the database.
  - A conceptual schema is a high-level description of the whole database.
  - External schemas describe the views of different user groups.
  - Mappings allows to transform requests and query results from one level to the next. That is the basis for the data logical and physical independence.
- Main types of DBMS languages:
  - A data definition language (DDL) is used to define the database conceptual schema.
  - A data manipulation language (DML) is used for specifying database retrievals and updates.
  - A view definition language (VDL) is to specify user views and their mappings to the conceptual schema
  - Storage definition language (SDL)
- There are different types of interfaces provided by DBMSs and different types of DBMS users
- Typical database system environment and DBMS utilities presented

"When you innovate, you've got to be prepared for everyone telling you you're nuts."

– Larry Ellison

Co-Founder, Oracle