Lecture 2. Database System Concepts and Architectures (Chapter 2)

Alisa.lincke@lnu.se

Linnéuniversitetet
Kalmar Växjö

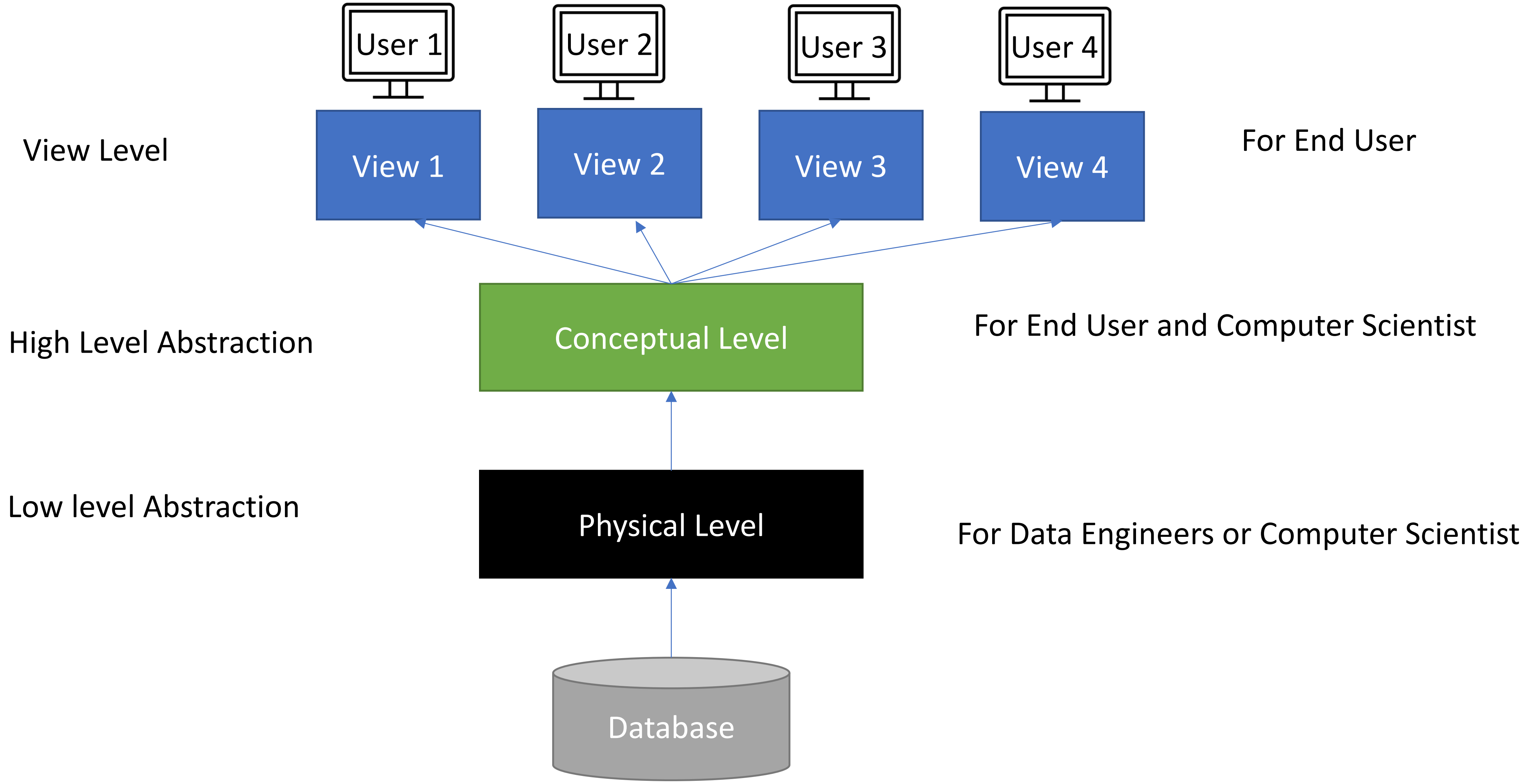
Outline

- Data Abstraction in DBMS
- Understanding Data Independence
- Data Models and Their Categories
- History of Data Models
- DBMS
    - Schemas, Instances, and States
    - DBMS Languages and Interfaces
    - Centralized and Client-Server Architectures
    - Classification of DBMSs

Data Abstraction in DBMS

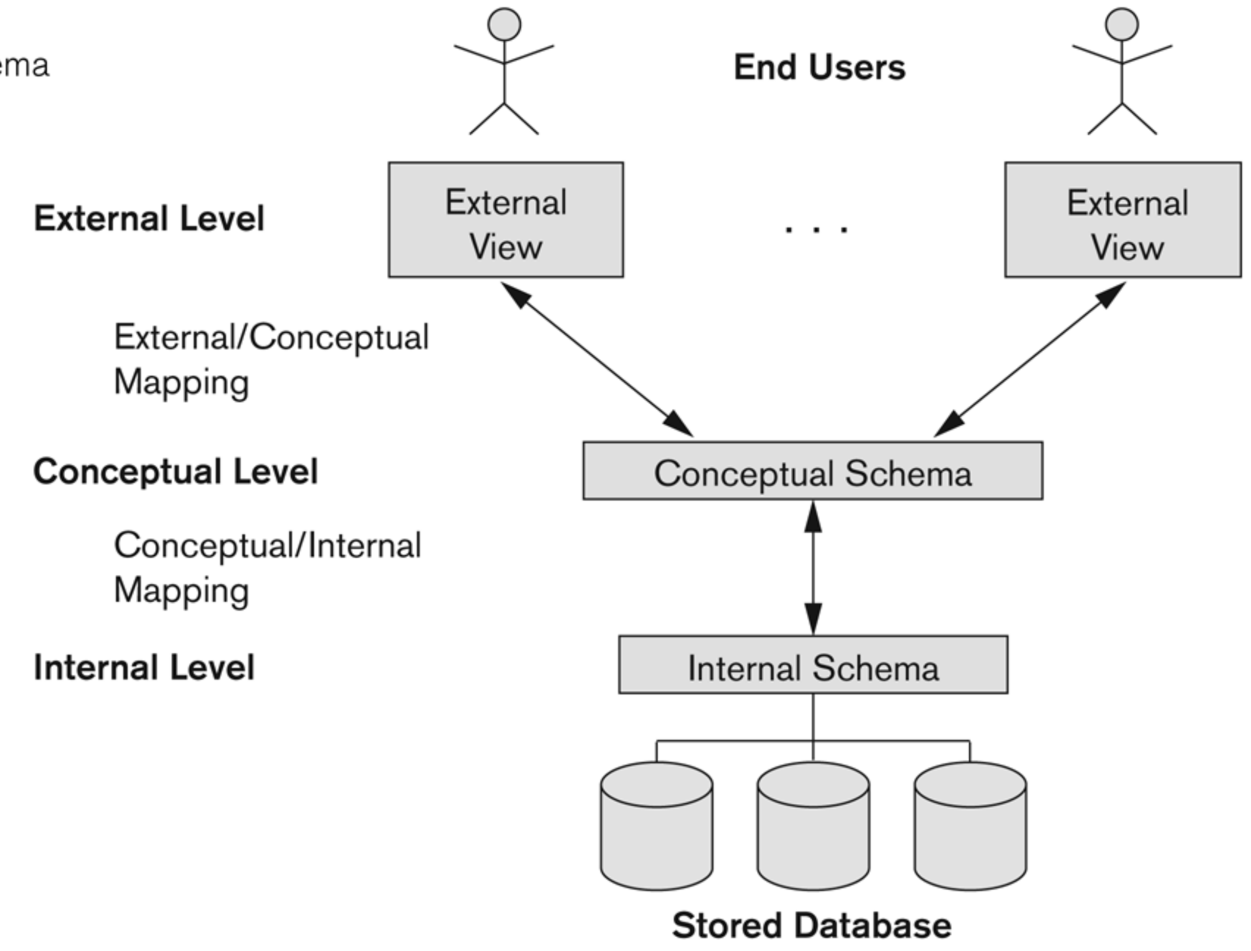
- One fundamental characteristic of the database approach is that it provides some level of **data abstraction**.
- **Data abstraction** refers to suppression of details of data organization and storage (e.g., different users can perceive data at their preferred level of detail) and aims to support **data independence**.
- We use a **data modelling** approach to describe the structure of a database on different levels of abstraction.

Levels of Abstraction



The three-schema architecture

Figure 2.2
The three-schema architecture.



- **Schema Diagram**:
  - An **illustrative** display (most aspects of) a database schema
- **Database Schema**:
  - The ***description*** of a database (e.g., structure, data types, and the constrains on the database)
- **Conceptual Schema:**
  - hides the details of physical structures and concentrates on describing *entities*, *data types, relationships, user operations, and constraints*.
- **Internal Schema:**
  - describes the physical storage structure of the database (e.g., details of data storage and access paths for the database)
- **Mappings:**
  - The process of transforming requests and results between levels. This mapping can be time-consuming.
- **The three-schema architecture**:
  - Not explicitly used in commercial DBMS products, but has been useful in explaining database system organization

Categories of Data Abstraction

- **View level** tells the applications about how the data should be shown to the end user
- **Conceptual level** refers to how the data is stored and structured. We can use different ***data models*** to provide high level of abstraction.
- **Physical level** tells us where the data is stored (actual location). The database administrator decide which data should be kept at which particular disk drive, how the data has to be fragmented, if the data has to be centralized or distributed.
- Most of DBMSs do not separate the three levels completely and explicitly, but they support the three-levels architecture tp some extent.
- Notice that View level and Conceptual level are only ***descriptions*** of data; the actual data is stored at the physical level only.
- Using data abstraction allows us to create different data views and achieve ***data independence***.

Data Independence

- **Data independence** is a abality to make a change in one level of abstraction without having to change the next higher level.
- Two types of data independence:
    - **Logical data independence** (conceptual level) is ability to make changes in conceptual level wihtout making changes to view level (application programs). For example, we can expand the database (add new data), or change constrains, or reduce the data in database.
    - **Physical data independence** (physical level) is ability to change physical logic without having to change the conceptual level and view level. For example, migrate database to another drive disk, or computer to improve the performance.

Conceptual Data Models for Conceptual Level

- **Conceptual Data model:**
  - is a set of concepts to describe the ***structure*** of a database, the ***operations*** for manipulating these structures, and certain ***constraints*** that the database should obey
- Data model structure consist from:
  - Concepts or constructs typically include ***entities***, ***attributes***, and ***relationships***.
  - En ***entity*** represent a descirption of real-world object or concept (e.g., Student, Teacher, Doctor, Patient)
  - An ***attribute*** represents some property of interest that further describes en entity (e.g., student's name, age, teacher's name)
  - A ***relationship*** among two or more entities represents an association among the entities (e.g., teach_on relationship between teacher and a subject)
  - **Constrains** specify some restrictions on valid data and they must be enforced at all times

Conceptual Data Models for Conceptual Level

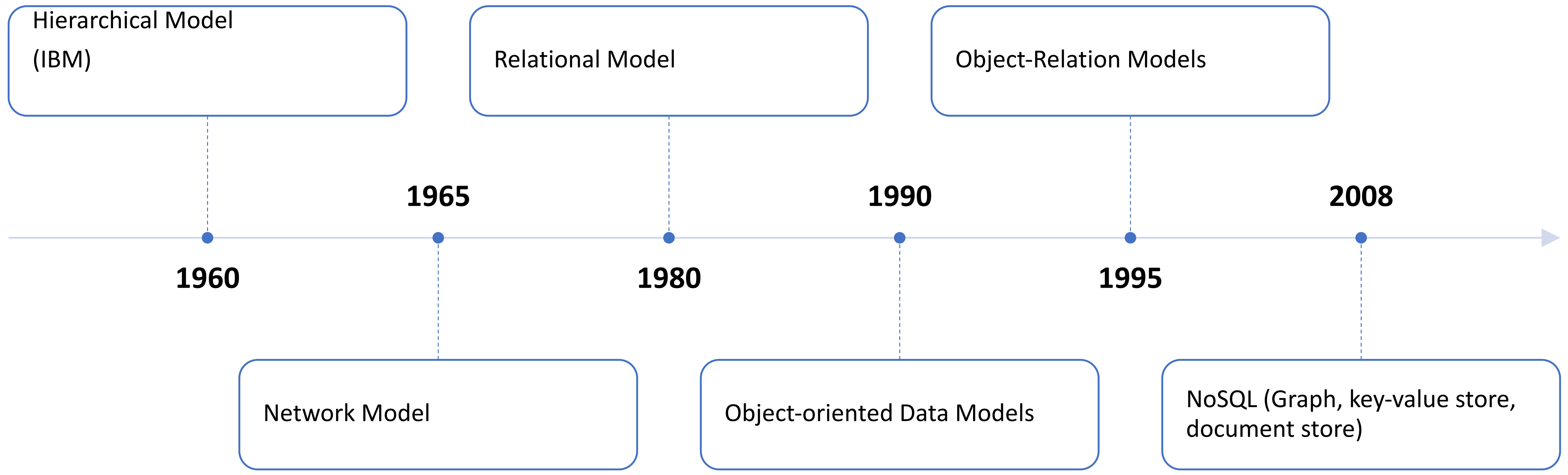
- **Operations** which are used for specifying database *retrievals* and *updates* and may include:
  - ***Basic model operations*** such as CRUD (create, read, update, delete)
  - ***User-defined operations*** (e.g., computer_student_final_grade)

This conceptual model type called the **entity-relationship model (ER)**

Categories of Data Models

- Conceptual (high level) data models:
    - Provide concepts that are close to the way many users perceived data and include:
        - Entity-relationship model (ER) and Enhanced entity-relationship model (EER) (Lecture 3)
        - Object data model
- Physical (low-level) data models:
    - Provide concepts that desribe details of how data is stored in the computer, and include:
        - Relational data models (internal schema database design) (Lecture 4)
        - Self-described data models (such as XML, key-value stores, NoSql) (Lecture 10)

History of Data Models

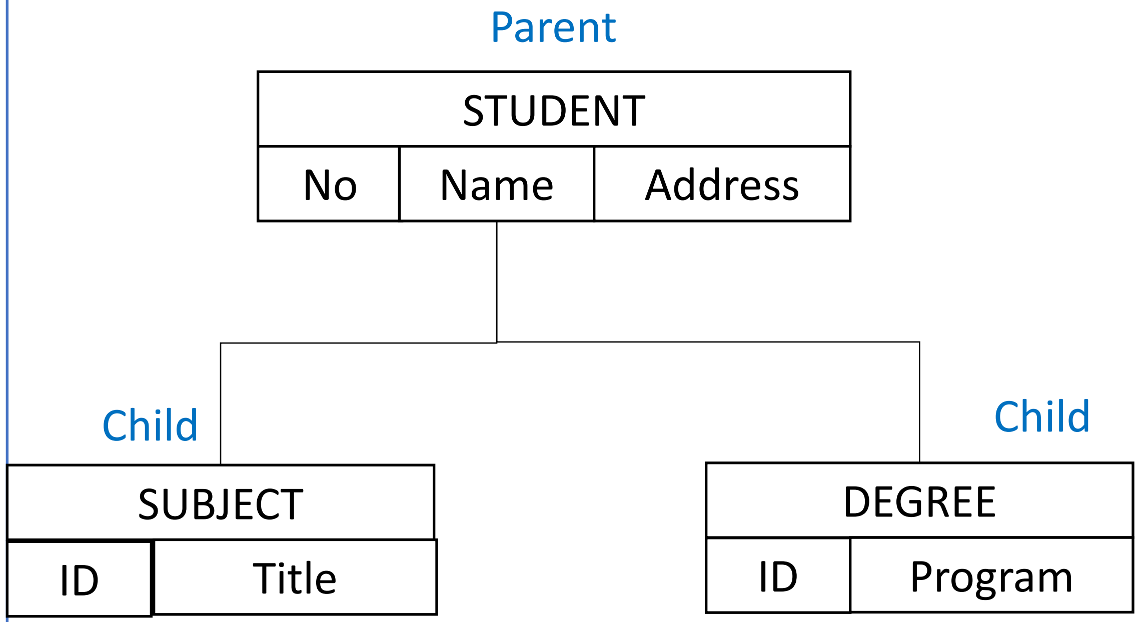


Hierarchical Models

Hierarchical Model

- Initially implemented in a joint effort by IBM and North American Rockwell around 1965.
- Represents data as a ***tree of records*** with *parent/child* relationships. The data are manipulated by using hierarchical and recursive queries.
- **Usability**: Nowdays the concepts of such model exists which supports hierarchical relationship (e.g., Statistical Analysis System(SAS)). Another example is IBM Information Management System (IMS)
- ***Advatnages***: simple to construct and operate, corresponds to a number of natural hierarchically organized organizations, buisness solutions, language is simple: get, get next, get next within parent, get children
- ***Disadvantages***: lack of flexibility, difficult to implement (no specfific standard), navigational and procedural nature of processing, no many-to-many relationship support, no query optimization support.

Example Hierarchical Model



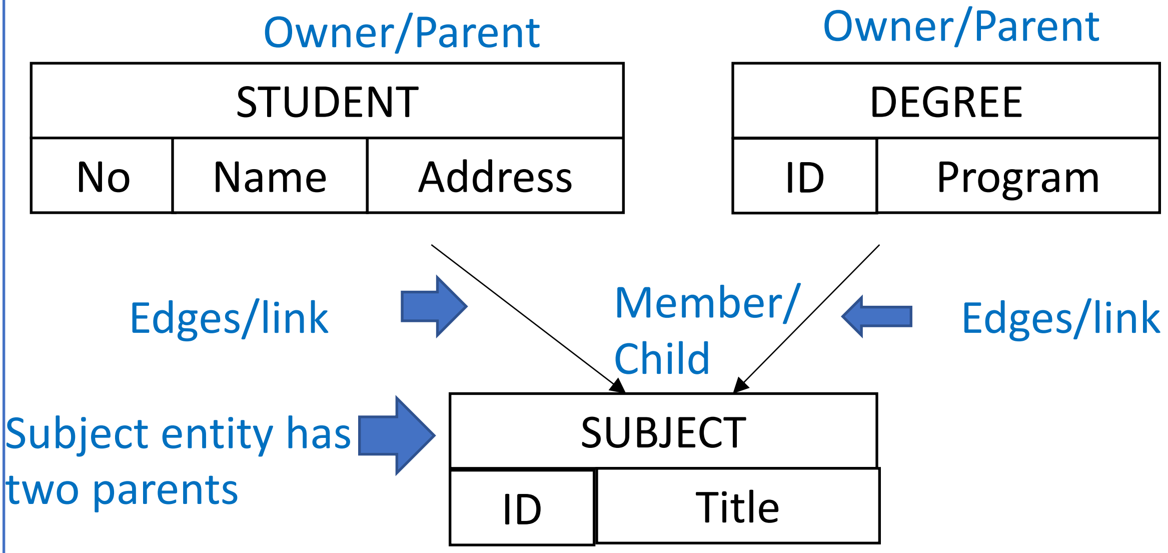
1-to-many relationship only
(one parent, many children)

Network Model

Network Model

- The first network model was implemented by Honeywell in 1964-65
- Represents data as a ***graph of records*** with *links* (as *relationship)* between entities.
- ***Examples***: Raime Database Manager (embedded database management system)
- ***Advantages***: able to model complex relationships (multiple paths to the same record), language is navigational uses constructs like FIND, FIND member, FIND owner, FIND NEXT CHILD/PARENT within set, etc.
- ***Disadvantages***: database conatins a complex array of pointers that thread through a set of records, little scope to automeated “query optimization”, may contain several inter-related entities with each other. In order to add new entity, the DBA needs to understand the whole structure.

Network Model Example

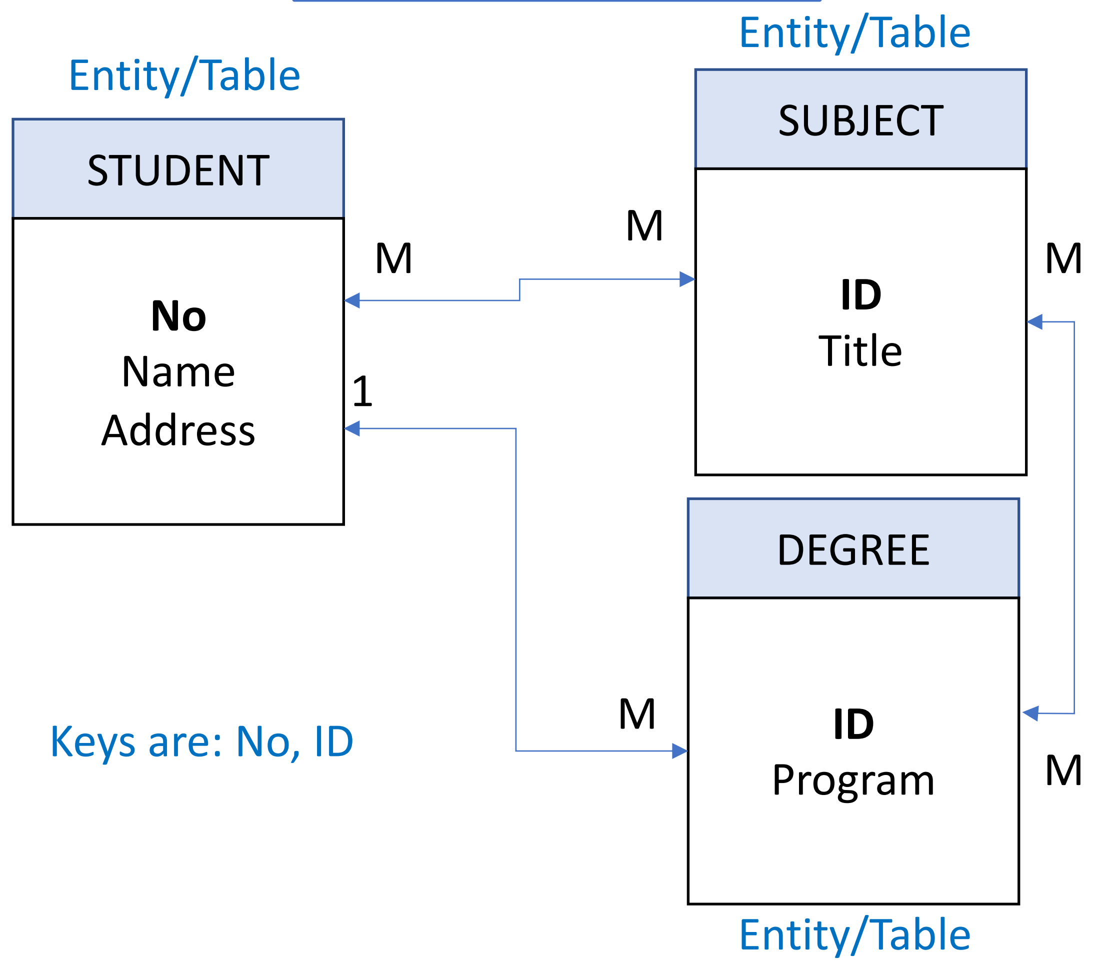


Relational Model

Relational Model

- Proposed in 1970 by E.F. Codd (IBM), first commercial system in 1981
- Represents data as ***tables*** where each table has a ***key*** which uses to represent the relationship between entities. The key is also used to support ***data uniqueness*** (no duplicates are entered in tables)
- Several commerical DBMS products such as DB2, Oracle, SYBASE
- Several free open source DBMS implementations: MySQL, PostgreSQL
- Currently most dominant for developing database applications

Relational Model

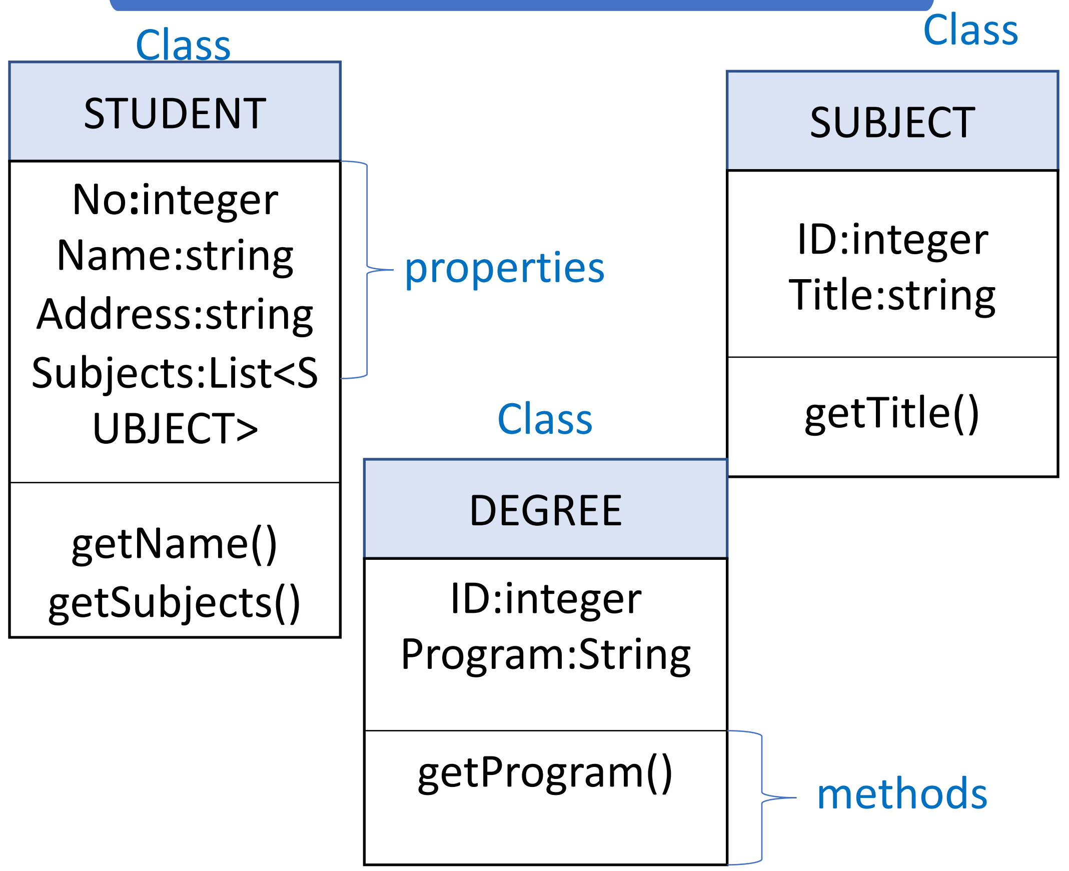


Object-Oriented Model

Object-Oriented Model

- Structure defines in terms of objects, their properties, and operations. Originally, the idea come from object-oriented programming language C++
- Used for engineering design and manufacturing (CAD), biological systems, telecommunications, geographic information, and multimedia systems.
- *Advantages*: the flexibility in defining the *structure/relationships* and *operations* for objects, easy to use in OOP applications.

Example Object-Oriented Model

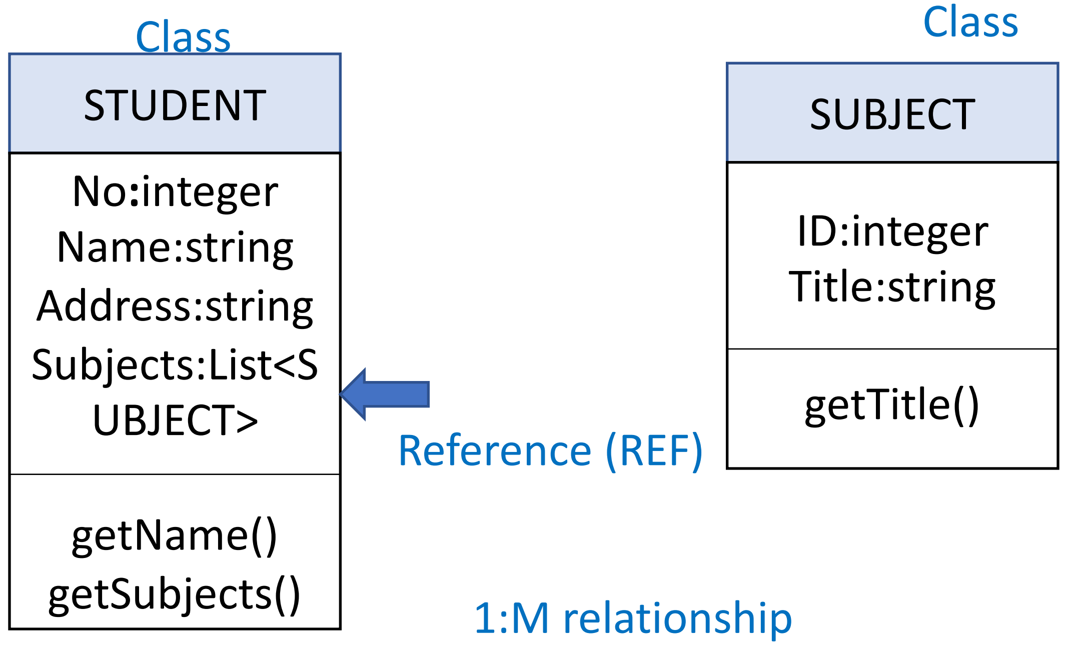


Object-Relational Models

Object-Relational Model

- Extended verison of object-oriented model which incorporates concepts from relational model and object-oriented model.
- Intorduced new SQL standard knows as SQL/Foundation and SQL/Object.

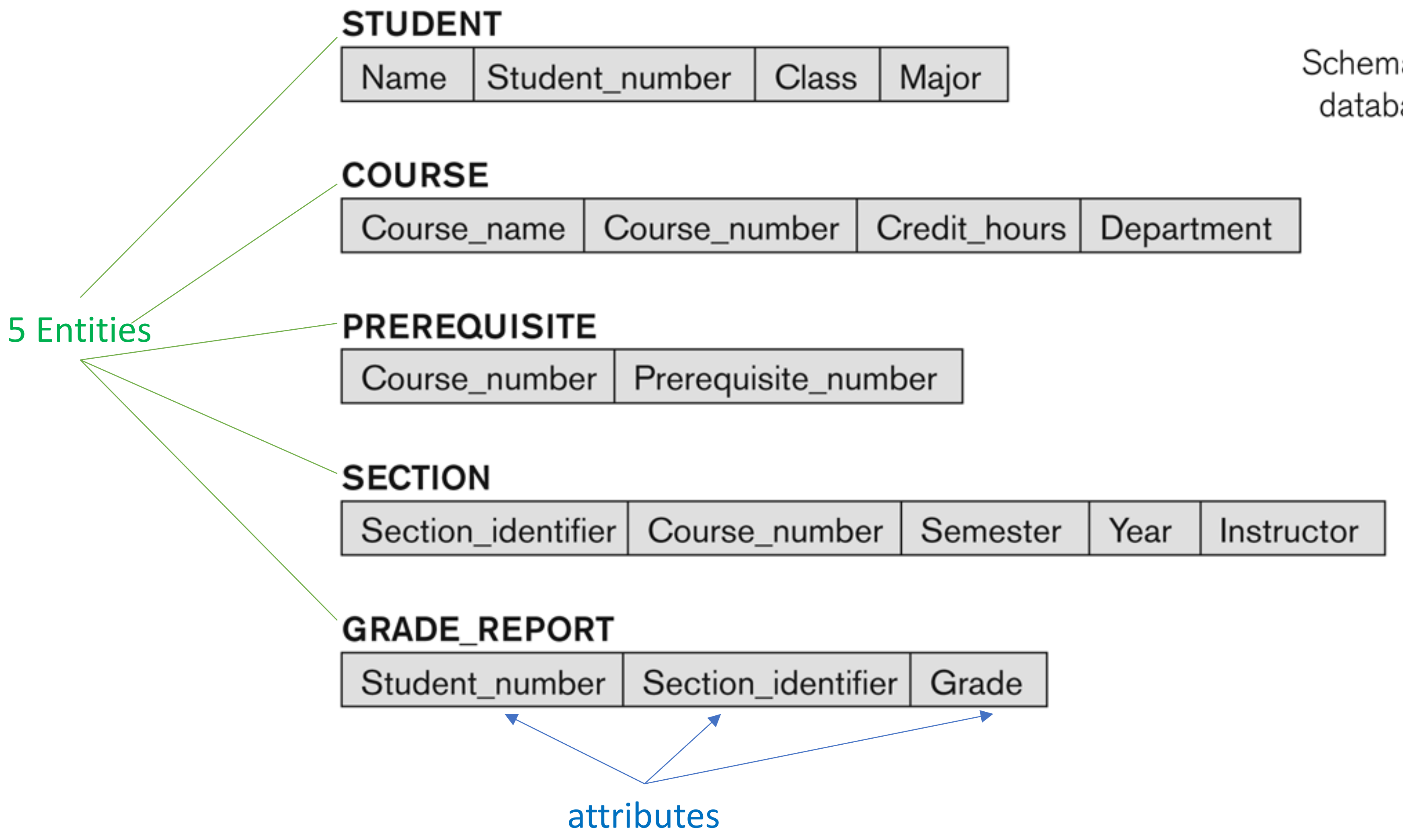
Example Object-Relational Model



Schema

- The ***description*** of a database is called the ***database schema***
- Includes desciptions of the *database structure*, *data types*, and *constrains* of the database
- **Schema diagram** is used to dsiplay most of concepts of a database schema
- Shcema **construct**:
  - a ***component/object*** in the schema such as STUDENT, TEACHER, COURSE

Example of Database Schema



STUDENT

Name	Student_number	Class	Major

COURSE

Course_name	Course_number	Credit_hours	Department

PREREQUISITE

Course_number	Prerequisite_number

SECTION

Section_identifier	Course_number	Semester	Year	Instructor

GRADE_REPORT

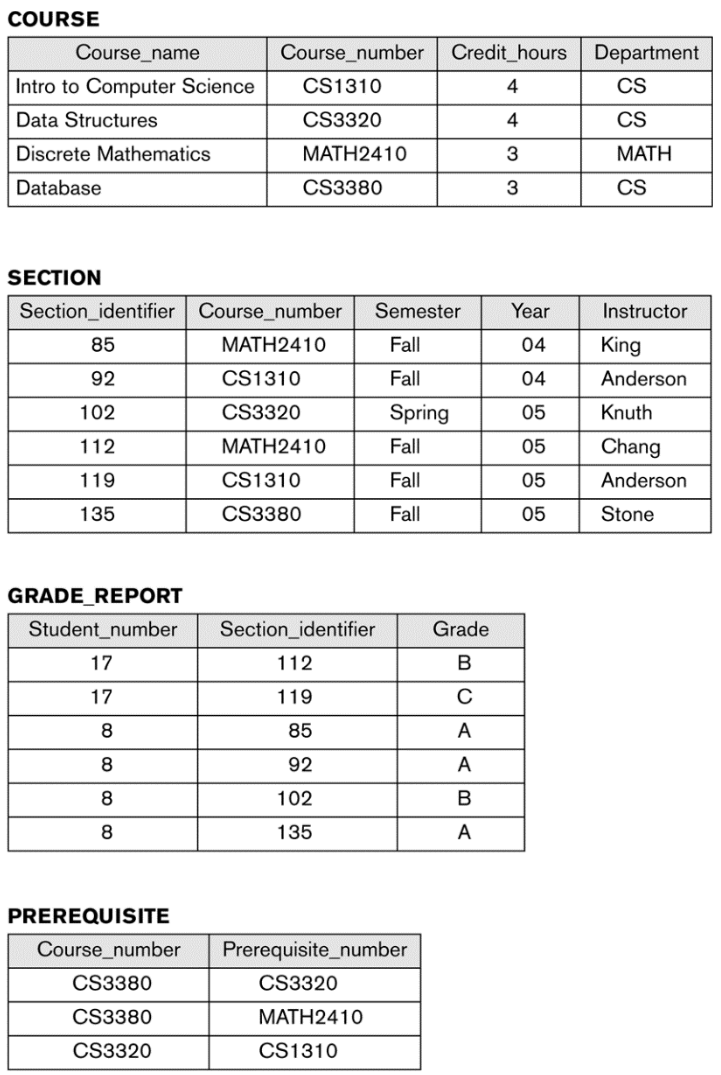
Student_number	Section_identifier	Grade

Figure 2.1
Schema diagram for the database in Figure 1.2.

Database State and Instance

- Database State:
  - Represents the actual data stored in database at a ***particular moment in time***. This includes the collection of all the data in database.
  - Also called database ***instance***
- Distinction:
  - The ***database schema*** changes very infrequently.
  - The ***database state*** changes every time the database is updated.

Examples of database state



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

SECTION

Section_identifier	Course_number	Semester	Year	Instructor
85	MATH2410	Fall	04	King
92	CS1310	Fall	04	Anderson
102	CS3320	Spring	05	Knuth
112	MATH2410	Fall	05	Chang
119	CS1310	Fall	05	Anderson
135	CS3380	Fall	05	Stone

GRADE_REPORT

Student_number	Section_identifier	Grade
17	112	B
17	119	C
8	85	A
8	92	A
8	102	B
8	135	A

PREREQUISITE

Course_number	Prerequisite_number
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

Figure 1.2
A database that stores student and course information.

DMBS Languages

- Once the design of a database is completed and a DBMS is chosen to implement the database, the first step is to:
  - Specify conceptual and internal schemas for the database and any mapping between the two using ***data defintion language*** (DDL)
- Once the database schemas are compiled and the database is populated with data, users must have some means to manipulate the database (read, insert, delete, update data).
  - The DBMS provides the ***data manipulation language*** (DML) for this purpose.
  - The most popular example of the DML is SQL relational language

DBMS Programming Language Interfaces

- **Embedded approach**: e.g., embedded SQL for C, C++, SQLJ for Java
- **Procedure call approach**: e.g., JDBC for Java, ODBC for other programming languages as API's (application programming interfaces)
- **Database progrmaming language approach**: e.g., ORACLE has PL/SQL a progrmaming language based on SQL; the language incorporates SQL and its data types as integral components.
- **Scripting Languages**: PHP and Python (server-side scripting) are used to write database programs

Component modules of a DBMS and their interactions

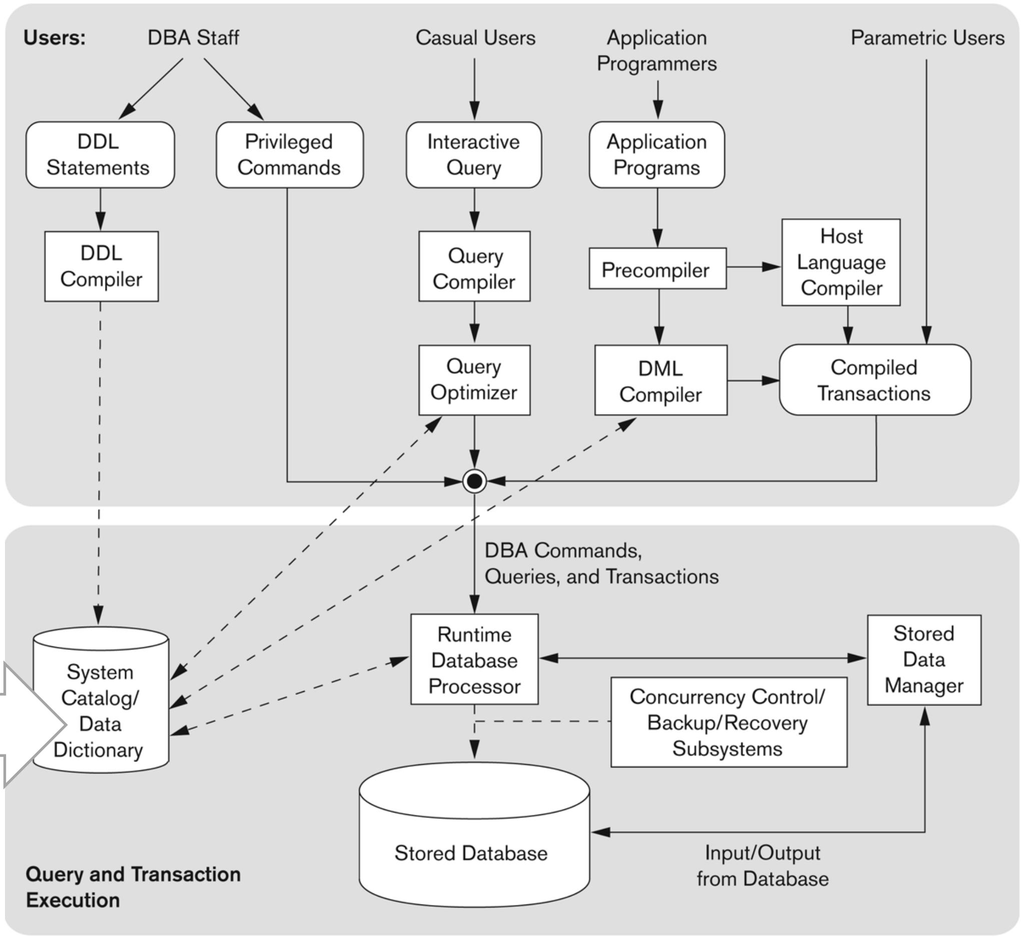


Figure 2.3
Component modules of a DBMS and their interactions.

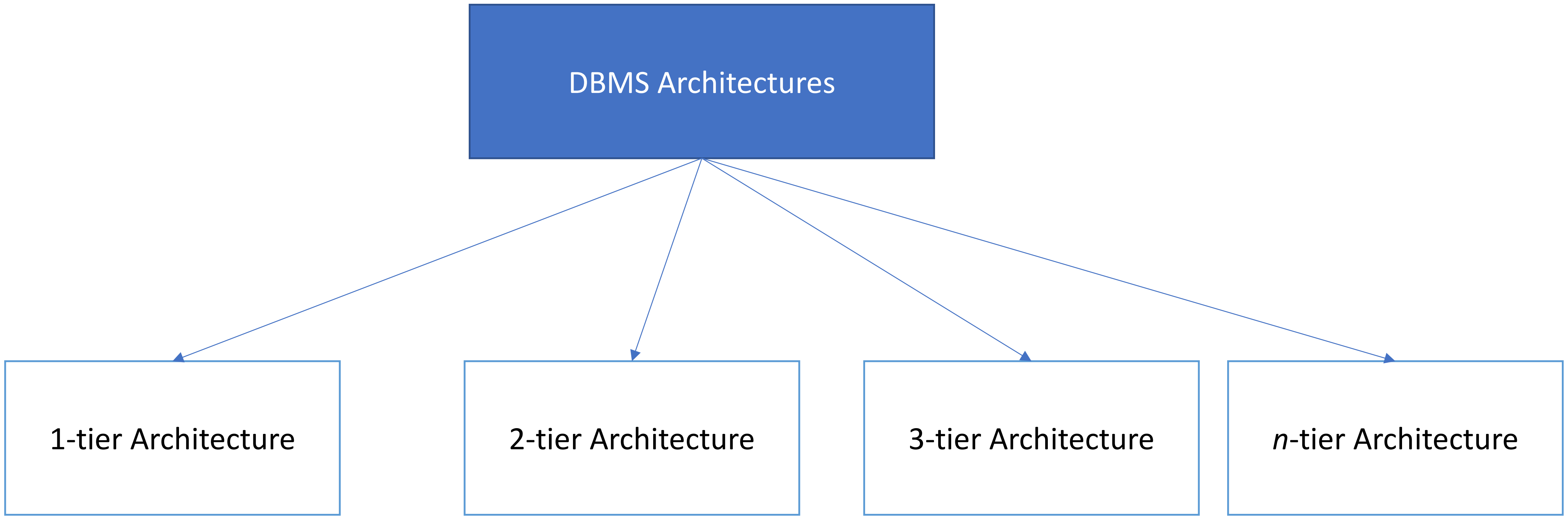
The catalog includes informati on about names and size od files, mapping informati on between schemas

- **Top part of the 2.3 figure**:
  - Contains different examples of interfaces for database administator (DBA) users, casual users, application programmers, and parametric users who don't have any DBMS knowledge but they frequently use the database applications in their daily life to get the desired results.
  - The DBA works on defining the database and tunning it by making changes to its definition using the DDL (e.g., create cluster, create table) and other privileged commands (e.g., creating user, password management, user privileges to read,write database, etc.)
  - DDL compiler process schema definitions and stored description of the schemas (meta-data) in the DBMS catalog.
  - Interactive queries by casual users (e.g., menu-based, form-based). They are parsed and validated for syntax correctness by a ***query compiler***.
  - Query optimizer is concerned with re-arrangment and possible reordering of operations , elimination of redundancy, and use efficient search algorithms.
- **Lower part of the 2.3 figure**:
  - The runtime database processor executes:
    - Privileged commands
    - Optimized queries, and
    - Transactions with parameters
  - Concurrency Control, backup, recovery services integrated into runtime database processor for purpose of **transaction management** (Lecture 9). ***Concurrency control*** feature of a database allowing several users access to the same data record at the same time
  - Stored Data Manager (or Database Control System) is a program that provides an interface between data stored in the database and the queries received.

DBMS Database System Utilities

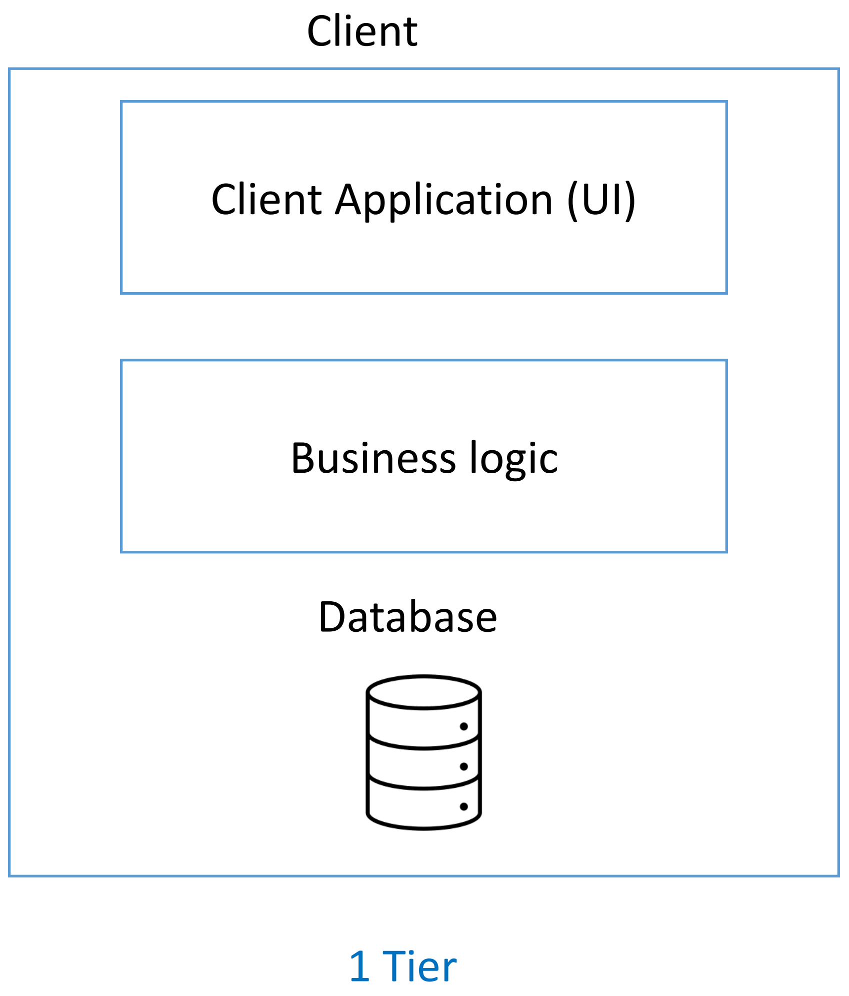
- **Loading** utility is used to load existing data files – such as text files into database. The formats should be compatibe with DBMS. Also used for transfering data from one DBMS to another DBMS
- **Backup** utility creates a backup copy of the database, usually by dumping the entire database onto tape or other mass storage medium. Further the bakcup can be used to restore the database in case of disk failure.
- **Database storage reorganization** utility allows organizing a set of files into different file organizations and create new access paths to imrpove performance
- **Performance monitoring** utility monitors database usage and provides statistics to the database administrator (DA). DA makes decisions such as wether or not reorganize files or wether to add or drop indexes to imrpove performance.
- **Security utility** helps to identify a data breach, granular access control, comprehensive activity monitoring.

DBMS Architectures



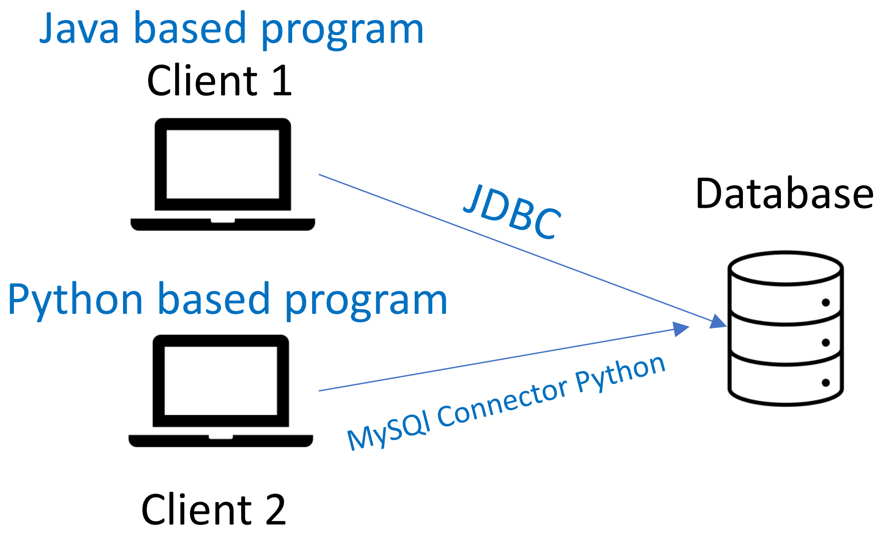
1-Tier Architecture

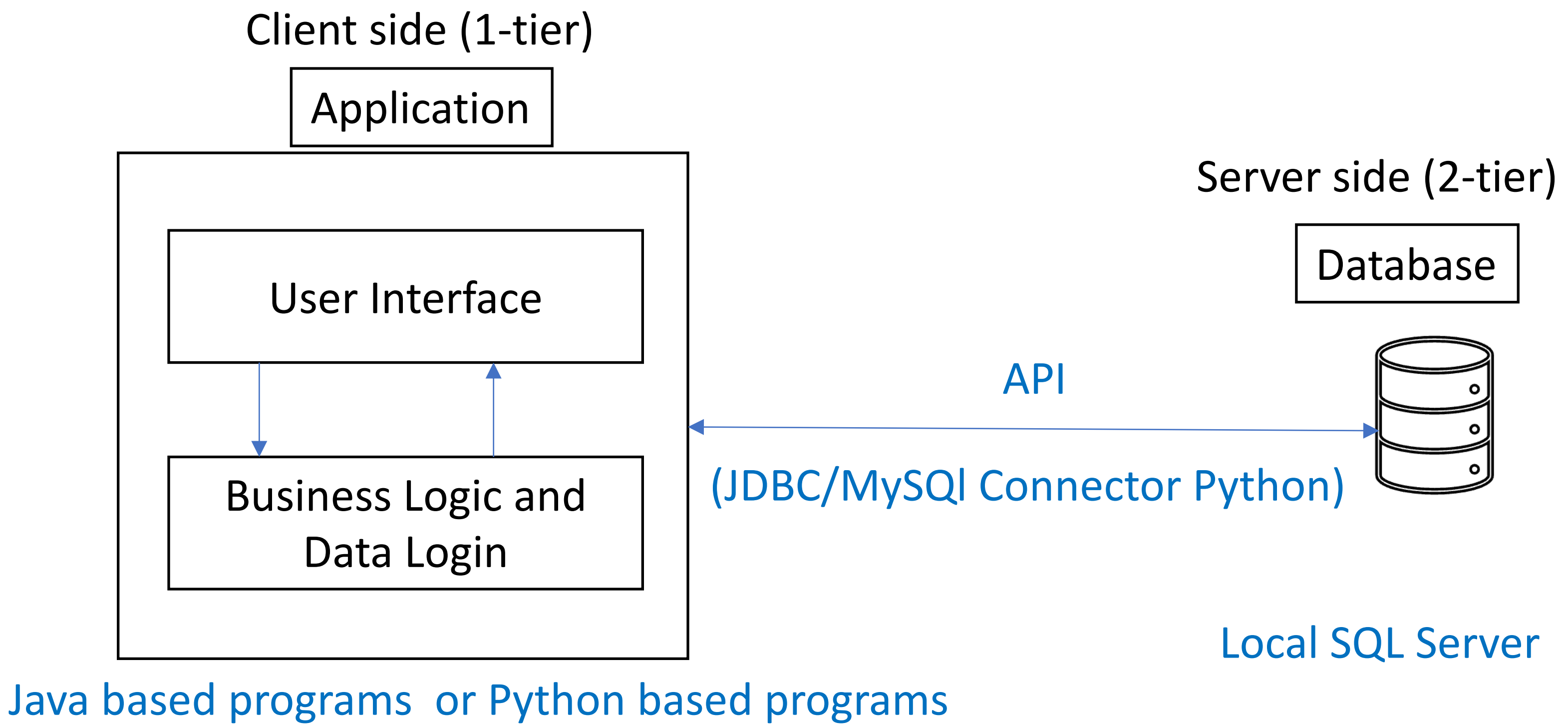
- **1-tier architecture** is where *client* (is UI software provided by the DBMS), *server* (is a client’s computer/machine), database (data itself) stored on one computer/machine
- Examples: MS Access , native mobile application (which does not use Internet/Back end)



2-tier architecture

- **2-tier architecture Client/Server** architecture is centralized architecture consist from client DBMS software and server database software. The application programming interface (APIs) allows to manipulate and communicate with database only if both client and server installed. The application program (user interface and business logic implemented on client side, the server side has connection to database and processing queries to database.

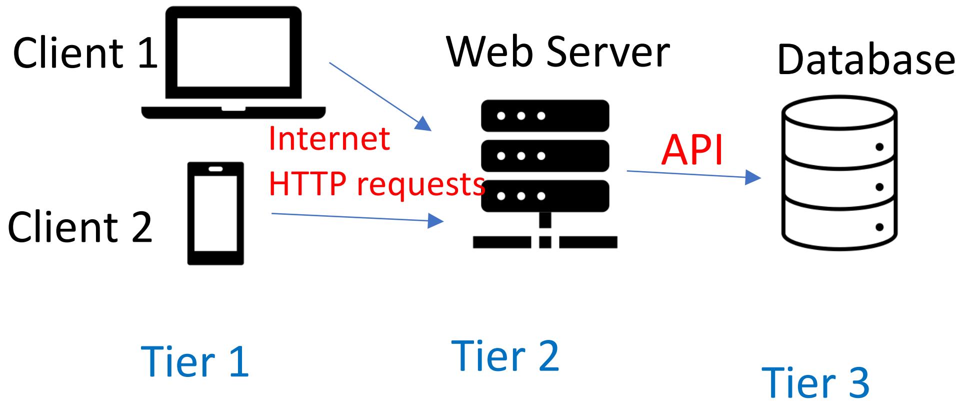


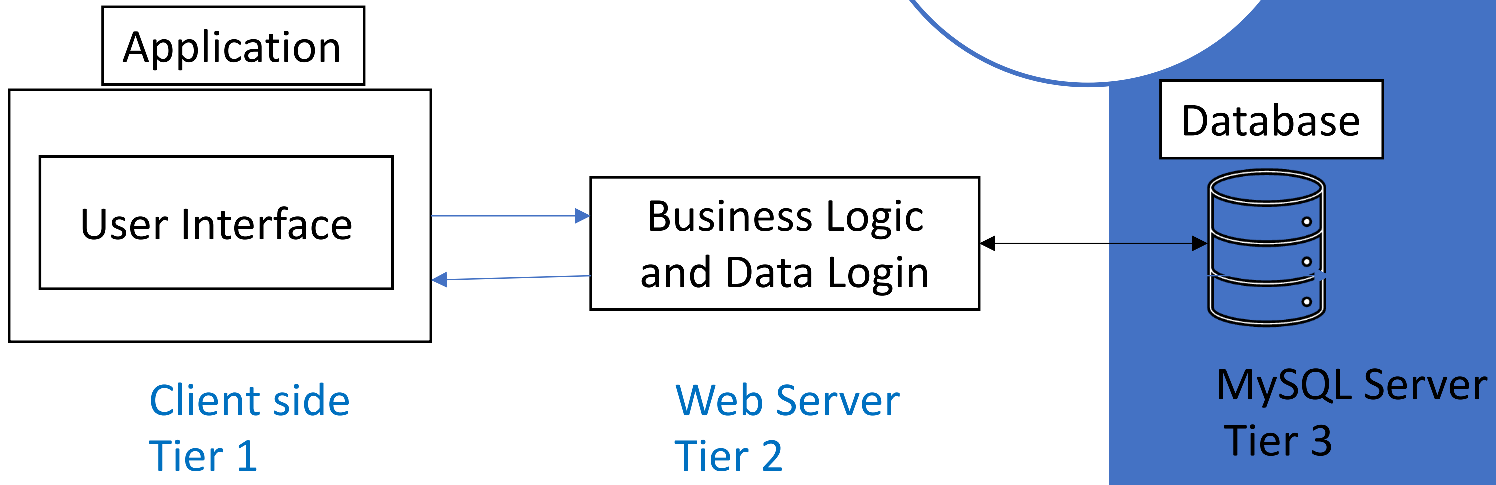


Client side (1-tier)
Application
User Interface
Business Logic and Data Login
Server side (2-tier)
Database
API
(JDBC/MySQl Connector Python)
Local SQL Server
Java based programs or Python based programs

Web Client-Server Architecture (or 3-tier architecture)

- **Server** is a centralized computer that provides services to all connected clients through network connection. For example, web server, file server, etc. each has a specific work to provide services to each client. Has installed server software.
- **Client** is laptop, tablet, mobile phone provided with the appropriate user interfaces to use these services, and provide local processing power to run local applications. Has installed client software.
- Example, any web site on the Internet





Web Client-Server Architecture

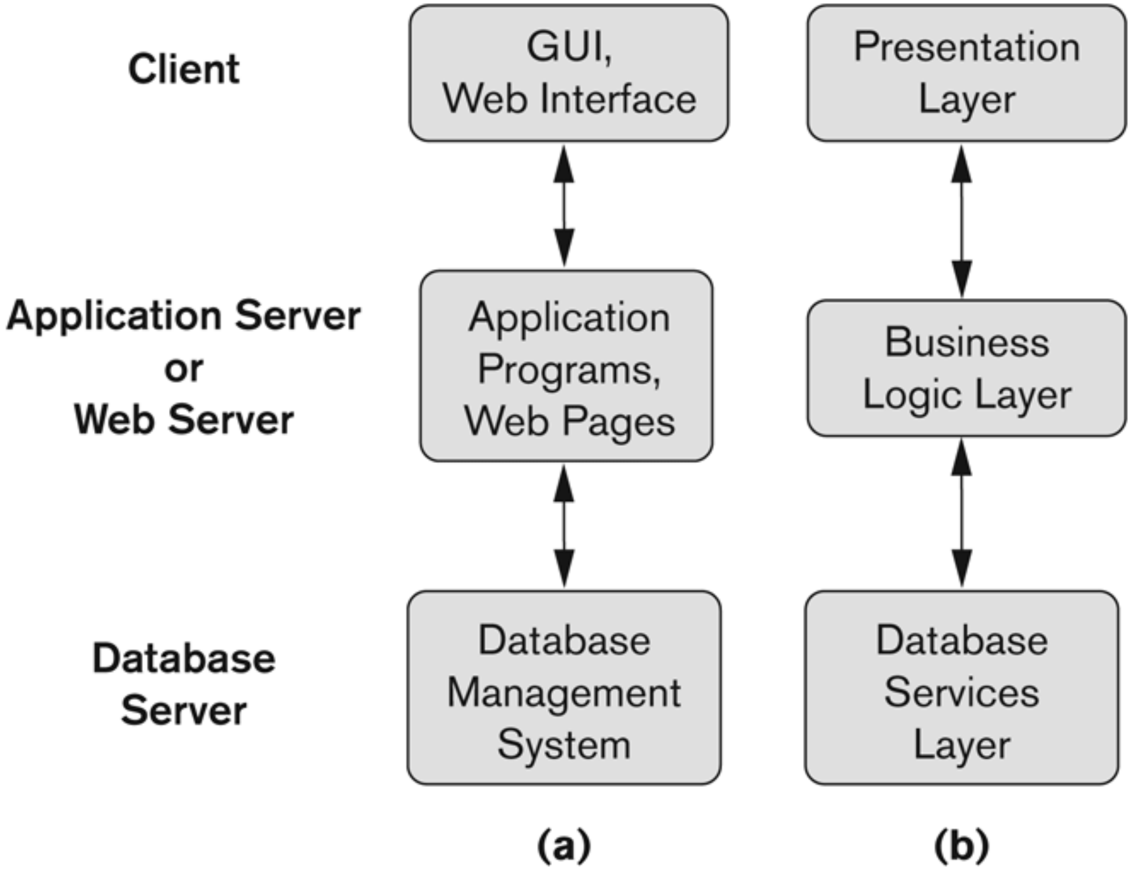
- Advantages:
  - All the data and resources are controlled by server (centralized control, good consistency)
  - Easy to increase number of clients (scalable)
  - Easy to use and not complex
- Disadvantages:
  - Traffic is a big problem in this network
  - When adding large amount of clients the network become overloaded and complicated
  - When the server goes down all the clients are not able to access the database

Two-tier and Three-tier architecture

- **Two-tier architecture** distributes software on two softwares: client and server. A client program should connect to DBMS using DBMS provider. After the connection, the client program uses application programing interface (APIs) to access the data in DBMS.
- **Three-tier architecture** mostly for web applications which adds an intermediate layer between the client and database server called ***Web Server***. A web server is responsible for storing business logic (data manipulation operations, formatting/encoding/decoding objects from/to database). Clients cannot directly access database server (enhanced security).

Example of Three-tier client-server architecture

Figure 2.7
Logical three-tier client/server architecture, with a couple of commonly used nomenclatures.



Centralized DBMSs Architecture (Physical Level)

- **Centralized DBMS** combines everything into single system including: DBMS software, hardware, application programs, and user interface software. All DMBS functionality, application program execution, and user interface were carried out on one machine.
- DMBS system was using the processing power at the user side.

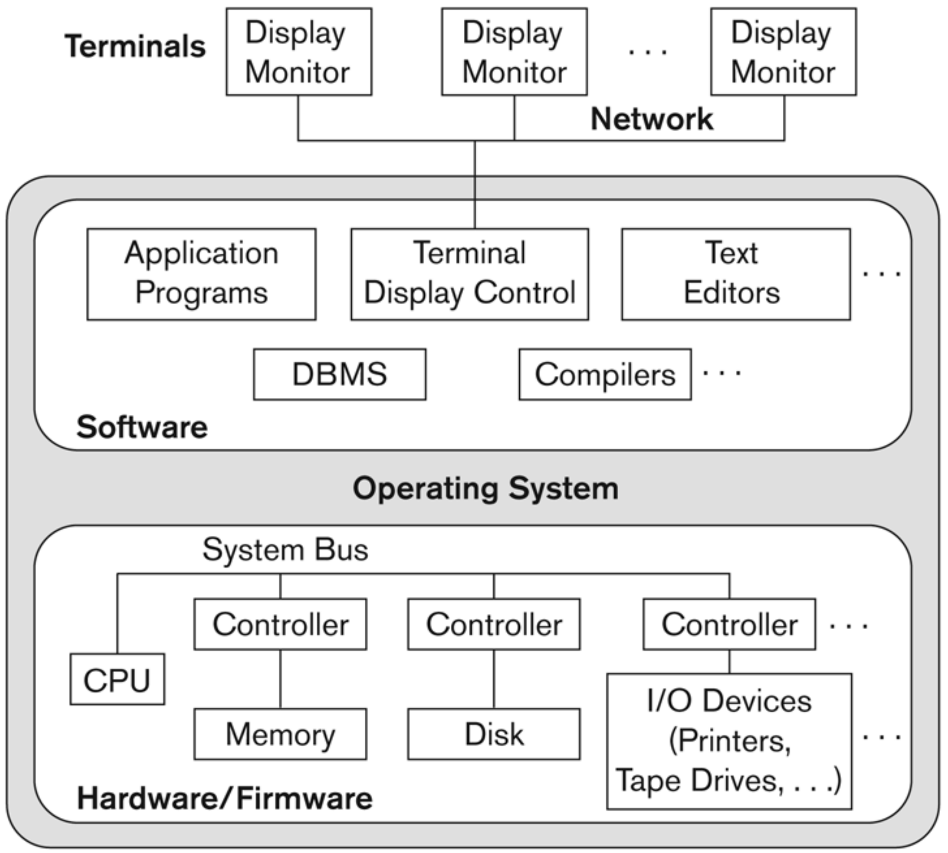
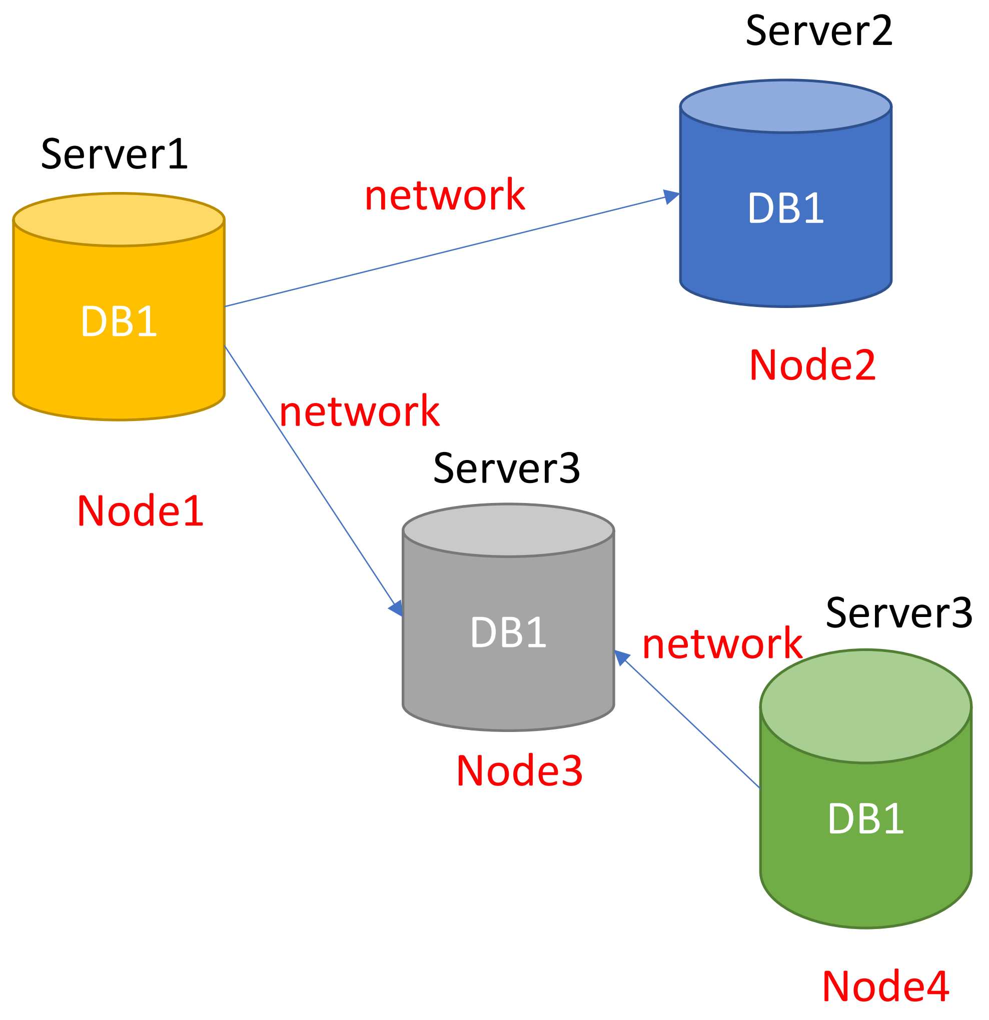


Figure 2.4
A physical centralized architecture.

Decentralized DBMS

- Decentralized databases stores information across a network (multiple servers) and controlled by several nodes or users.
- Decentralized DBMS properties:
  - **data replication**: data duplicated across the network (DB1 is the same on all nodes (servers)). Everything is done by everyone (all nodes) in the network.
  - **data modification**: when modifications are made to one node, it reflects in all nodes in network.
  - **data accessibility**: efficient and fast data retrieval by selecting closest node in a network.
- Decentralized networks usually have no single owner
- Example: Blockchain-based database (BigchainDB)



DBMSs Classification

- **Based on the data model used**:
  - Currently used for traditional real-world organizations/business cases: *relational*, *object-relational* databases. Examples, are MySQL, Oracle, PostgreSQL DBMSs
  - Recent Technologies for Big Data: *key-value storage systems* (e.g., Redis), *NoSQL* or *document-based* (e.g., MongoDB), *graph* (e.g., OrientDB, Neo4j, RedisGraph)
- **Based on number and type of users/clients**:
  - ***Centralized/decentralized*** if the data are stored at a single computer/machine/server /if the data stored across several computers (physical locations) over the network (e.g., Blockchain-based database) and has no single authority (many owner users).
  - ***Distributed*** if the *data* and *data processing* (workload) distributed over many computers/servers and has single owner/authority. (e.g., Apache Ignite, Apache Cassandra, Apache HBase, Couchbase Server, Amazon SimpleDB)

Summary

- Introduced main concepts used in database system: data model, levels of modeling/abstraction (high level, low level, view level), database schema, etc.
- Mostly presented data models on conceptual level
- History of conceptual data models. Nowadays still most used one is ***relational models*** for traditional business applications and organizations
- Physical level discussed in terms of client/server architecture, DBMSs types, languages, interfaces, and utilities

Lecture2 Key Terms

- **data abstraction**: is process to provide different views (details) on the data structure
- **data independence**: the flexibility of the DBMS to make changes on one level of abstraction without making changes on the next level of abstraction (or data representation/view level)
- **data modelling approach**: is a first step in database design and used to describe the data structure at different levels of abstraction (high-level, low-level)
- **conceptual level (high level abstraction)**: is *describes* data structure using a set of concepts (e.g., entities, attributes, tables, keys, schema)
- **physical level (low level abstraction)**: is *describes* the physical storage structure of the database (defines internal database structure (relational data model), file organization, indexing techniques, etc.)
- **difference between conceptual and physical level**: conceptual level describes data structures with concepts while physical level describes the physical location and structure of files where the data are physically stored.
- **conceptual data models**: a collection of concepts (e.g., entities, attributes) to describe the data structure, data types, operations, and constrains
- **physical level data models**: relational data model (internal schema), file structure, DBMSs architecture (1-tier, 2-tier, 3-tier)
- **types of data models**: hierarchical, network, relational, object-oriented model, object-relational model
- **database state/instance**: represents the actual data stored in database at a particular moment in time. This includes the collection of all the data in database.

Lecture2 Key Terms

- **1-tier architecture**: *client*, *server*, and the *database* are located on the same machine. For example, MySQL workbench (client), MySQL Server (software installed on client), database (data stored on client).
- **2-tier architecture**: *client* (desktop application such as Java-based, Python-based) and *server* (MySQL Server installed on another machine), *database* (data are stored on another machine where is MySQL server is installed). Client communicates with server using APIs.
- **3-tier architecture**: *client* (web browser on client’s device), *server* (web application deployed on web server), and *database* (data stored on web server). Clients does not have direct access to the database, clients communicates with server over HTTP protocol, and web server application communicates with database using APIs.
- difference between ***centralized*** and ***decentralized*** databases is that centralized approach has one central server where the database is installed and one database file, while in the distributed database approach, there are many servers installed and same database is replicated in all nodes.
- difference between ***2-tier*** and ***3-tier architectures*** is that 3-tier architecture has extra separation layer (web server application (business logic) located at server side, client side has only user interface application) and two connections (client-web server connection over http, and web server-database server connection via APIs).
- **DBMS types**: relational-based (MySQL, Oracle, PostgreSQL), non-relational (e.g., document-based, key-value, and are MongoDB, Redis), graph-based (e.g., OrientDB, Neo4j)