Database and Information Systems

Course Roadmap

Chapter 1	Introduction to Databases
Chapter 2	Integrity Constraints and ER Model
Chapter 3	Relational Databases and Schema Refinement
Chapter 4	Query Language
Chapter 5	Transaction and Concurrency Control
Chapter 6	Indexing

Data and Information

- Data
  - Raw and isolated facts about any subject or entity (recorded)
    - Data is used to provide useful information
    - Text, audio, video, images, etc.
- Information
  - Processed, meaningful, and usable data
    - A train information from IRCTC data
    - A student record in University data
    - A customer account information in a Bank
- Difference between Data or Information?
  - Information is derived from data
  - Data or Information depends on the observer
    - Database lecture can be **Information** for CSE students but can be **Data** for high school students
    - An assignment can be **Information** for a student that was created by the student from book, web **Data**

Introduction to Databases

- Database System
  - Database
  - Database Management System
- Database
  - Collection of **related** data
    - IRCTC has trains, passengers information
  - Used to solve a particular problem
- Types of Database
  - Structured Database
    - Relational Databases (IRCTC, University Database)
  - Unstructured Database
    - Web pages
  - Example: A video in YouTube is a Data, YouTube is a Database.

Introduction to Databases

- Database Management System
  - Perform operations in Database such as insert, delete, update
  - Manage Database in efficient way
  - An environment that is both ***convenient*** and ***efficient*** to use
- Relational Database Management System
  - DBMS that designed for Relational Databases (or Structured Databases)
  - A relational database is a database that stores data in ***relations (tables)***
  - RDBMS are most widely accepted DBMS
- Example of RDBMS
  - SQL server 2005, 2008, 2012, 2014, 2016, 2017, 2019
  - Oracle 9i, 10g, 11g, 12c, 18c, 19c
  - MySQL
  - DB2

Introduction to Databases

- Databases touch all aspects of our lives
- Database Applications:
  - Banking: transactions
  - Airlines: reservations, schedules
  - Universities: registration, grades
  - Sales and online retailers: customers, products, purchases, order tracking, customized recommendations
  - Manufacturing: production, inventory, orders, supply chain
  - Human resources: employee records, salaries, tax deductions

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
  - Now, Databases are usually large
  - And we use client-server architecture
    - Gmail, Online Banking, Online Shopping, Web-based applications, etc.

Database System vs. File System

- File System
  - Manages and organizes the files
- Database System
  - Manages and organizes the databases

- What is the need of a Database System?

Drawbacks of Using File System to Store Data

- Ineffective utilization of memory and high input-output cost
  - **Large file transfer**. Inefficient memory and time utilization
  - In DBMS, only a train record will be retrieved using DBMS query. E.g., find a train record in IRCTC
- Difficulty in accessing data
  - **Need metadata**. E.g., need actual file location and name
  - In DBMS, simple query or API can be used to access the data without knowing location or other attributes. E.g., search train information
- Data redundancy
  - **Duplication of information in a file**, multiple same file with different formats, duplication of same information in different files
  - In DBMS, constraints such as primary key, foreign key are present
- Data inconsistency
  - Inconsistency can arise when we change just one part of redundant information present in file(s)

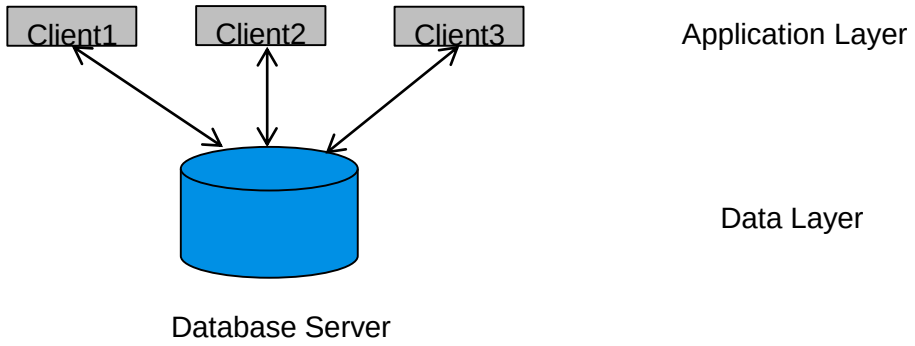
Drawbacks of Using File System to Store Data

- Concurrent access by multiple users
  - Concurrent access needed for performance. E.g., in IRCTC lakhs of transactions are done in a day
  - Uncontrolled concurrent accesses can lead to **inconsistencies**
  - In DBMS, protocols exist to ensure concurrency
- Security problems
  - Unavailability of **role-based data access**
  - DBMS provide role-based security
    - Different role for different users such as student, faculty, dean role in university database

Database systems offer solutions to all the above problems

Database Architecture

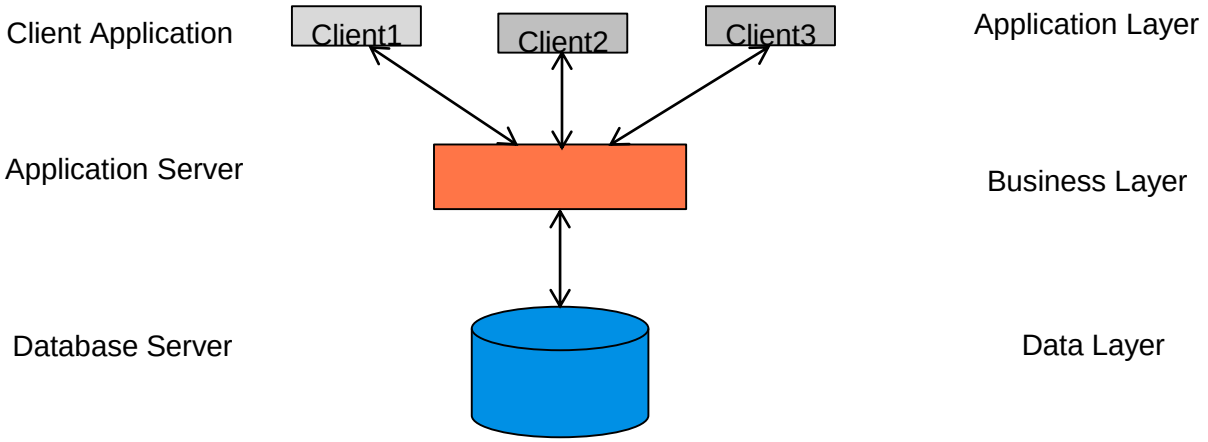
- Two-tier Architecture



- Advantages
  - Simple
  - Easy to maintain
- Disadvantages
  - If lots of users access the data, scalability problem can arise
    - Server can be overloaded
  - Security issues as clients are directly interacting with Database

Database Architecture

- Three-tier Architecture
  - It contains three layers
  - It has an additional intermediate Business Layer
  - Business layer process the query and check the conditions
    - Load is decreased at Database side
  - Scalability and security are few advantages
  - Maintenance is little costlier
  - Web-based applications are usually based on this architecture



Abstraction

- Hide internal irrelevant details of users
  - To ease the user interaction with database
  - To achieve security
- DBMS made-up of complex data structures
  - Developers hide internal irrelevant details from users
  - Example
    - Users are unware
      - Where data is stored, what is format of data, what are files, what schema is used
    - Real-world examples are IRCTC, Gmail

The process of hiding irrelevant details from user is called Data Abstraction

Levels of Abstraction

- **Physical level:** Describes how a record (e.g., instructor) is stored
  - Location, name, size of Database in memory, indexing
- **Logical level:** Describes data stored in database, and the relationships among the data
  - It is a logical blueprint of Database

type *instructor* = **record**

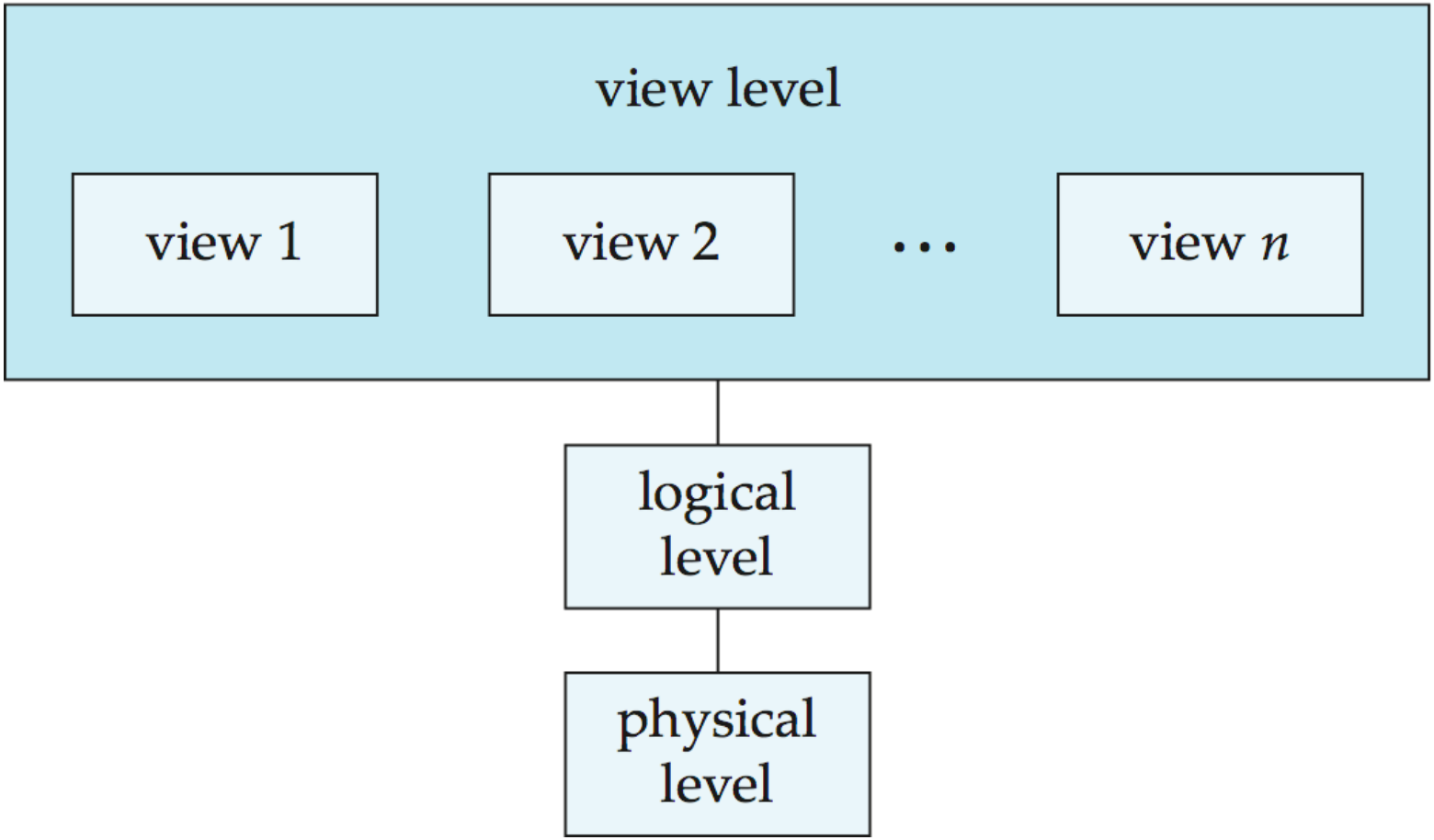
ID : string;
name : string;
dept_name : string;
salary : integer;

end;

- **View level:** Application programs hide details of data types. Views can also hide information (such as an employee's salary) for privacy/security purposes
  - Only a part of the actual database is viewed by the users

View of Data

An architecture for a database system



Schema

- Schema
  - Framework to describe the structure of a specific Database System
- To achieve Data Abstraction, Schema is used
  - Abstracts the database at three levels
- Three Schema Architecture
  - Introduced in 1975
  - Breaks the Database System into three different categories
  - Used to separate the user applications and physical database

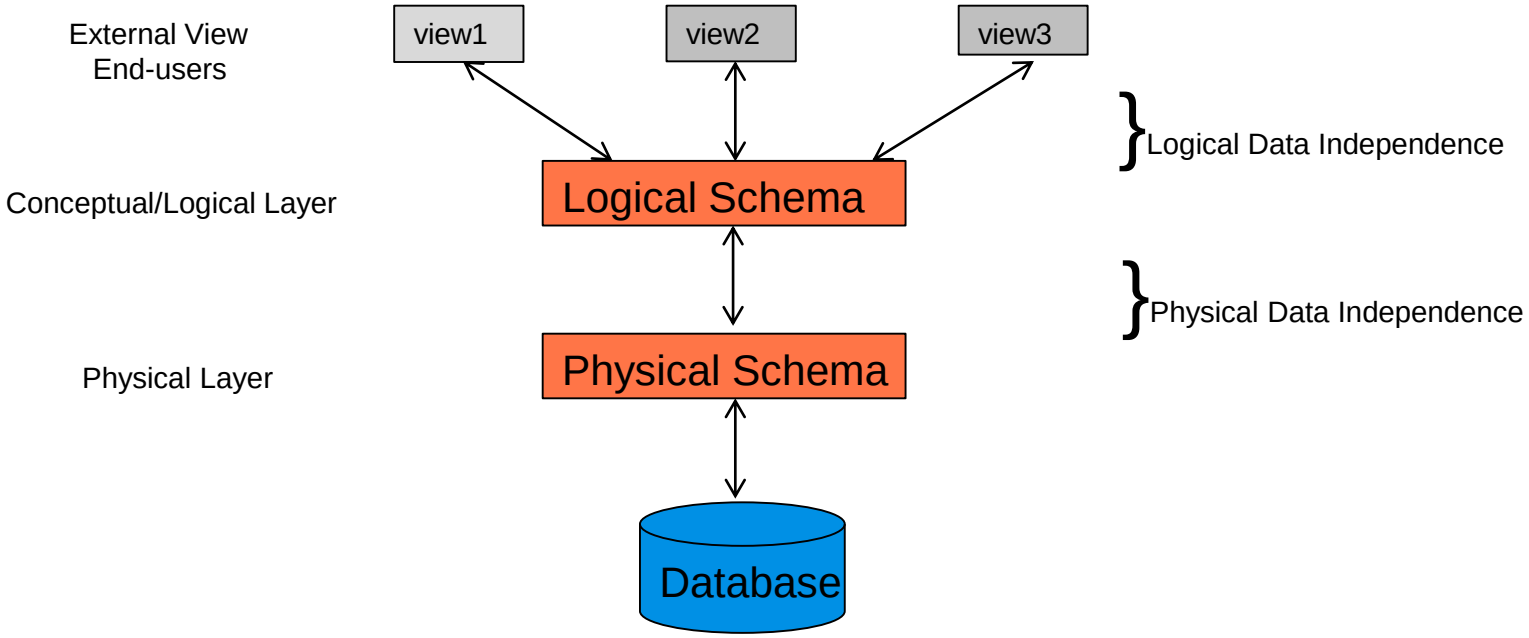
Schemas and Instances

- **Three Schema Architecture**
  - Hide how and where data is stored
- **External Schema** – the view of data to users
  - For example, interface of university Database, different view for different users
- **Logical Schema** – the overall logical structure of the database
  - Example: University Database consists of information about a set of students, courses and the relationship between them
  - It acts as blueprint to build a Database
  - RDBMS stores data in a table format

Roll No	Name	Address	Courses	Contact

- **Physical Schema**– the overall physical structure of the database
- **Instance** – the actual content of the database at a particular point in time

Three Schema Architecture



Data Independence

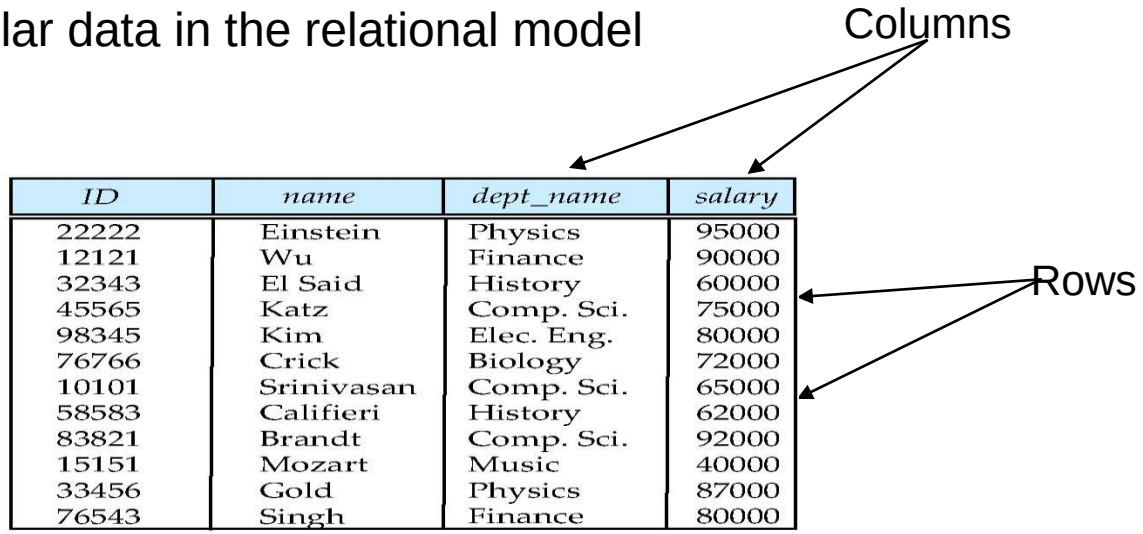
- **Data Independence** – Change the Database schema at one level of a database system without changing the schema at the next higher level
  - **Logical Data Independence** – Ability to modify to change the logical schema without changing external views, external API, or programs
  - **Physical Data Independence** – Ability to modify the physical schema without changing 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 Models

- A collection of tools for describing
  - Data
  - Data relationships
  - Data semantics
  - Data constraints
  - Conceptual or logical schema
- Hierarchical model
- Network model
- **Relational model**
- **Entity-Relationship data model** (mainly for database design)
- XML data model

Relational Model

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



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

- Relational models are managed using Data Definition Language and Data Manipulation Language

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),
)

- DDL compiler generates a set of table templates
- 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 is also known as query language
  - SQL is the most widely used commercial language

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

Database Design (Cont.)

- Any suggestion to improve the below 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

Important Note: Many times, we refer logical schema as schema

Design Approaches

- Need to come up with a methodology to ensure that each of the relations in the database is “good”
- Two ways of doing so:
  - Entity Relationship Model
    - Models an enterprise as a collection of *entities* and *relationships*
    - Represented diagrammatically by an *entity-relationship diagram*
  - Normalization Theory
    - Formalize what designs are bad, and test for them

Online Processing System

- Online Processing Systems
  - Operate data of business environment
  - Captures, stores, and processes data from transactions in real time
  - Analyze aggregated historical data
- Types of Online Processing Systems
  - OLAP- works on historical data (~95% data)
  - OLTP- works on real-time or current data (~5% data)
- Why two different types of Processing Systems
  - Low performance in day to day operations due to huge size of data. e.g., bank
  - Access Time usually depends on the size of data

OLAP vs. OLTP

OLAP	OLTP
Online Analytical Processing	Online Transaction Processing
Works on historical data (~95% data)	Works on current data (~5% data)
Subject oriented e.g., research on bad loans prediction	Application oriented e.g., transactions
Used for decision making such as **prediction**, **recommendation.** If a team will win football/cricket match, if build a warehouse at a location, share market prediction to invest money	Used for day to day operations
Works on huge data (TB, PB)	Works on relatively less data (GB)
Deals by higher management (CEO, MD, GM)	Deals by clerks and managers
Requires read operations	Requires read as well as write operations

History of Database Systems

- 1950s and early 1960s:
  - Data processing using magnetic tapes for storage (First in 1928)
    - Tapes provided only sequential access
- Late 1960s and 1970s:
  - Hard disks allowed direct access to data (First in 1955 by IBM)
  - Network and hierarchical data models in widespread use
  - E.F. Ted Codd defines the relational data model
    - ACM Turing Award (1981)
  - 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 standards
  - Automated database administration
    - Automatically back up new records and delete old records
- Later 2000s:
  - Giant data storage organizations

References

- Silberschatz, Abraham, Henry F. Korth, and Shashank Sudarshan. *Database system concepts*. Vol. 6. New York: McGraw-Hill, 1997.
- Ramez Elmasri, Shamkant B. Navathe. *Fundamentals of Database Systems.* Edition 6. Pearson, 2010.

End of Chapter 1