

# **Chapter 1: Introduction**

# Outline

- The Need for Databases
- Data Models
- Relational Databases
- Database Design
- Storage Manager
- Query Processing
- Transaction Manager

# Conventional File Processing System

The information system can be either a file processing system or database system.



## Disadvantages

1. Data Redundancy
2. Data Inconsistency
3. Difficulty in Accessing
4. Data Isolation
5. Integrity Problems
6. Concurrent Access
7. Security Problems

# Drawbacks of using file systems to store data

- **Data redundancy and inconsistency**

- Multiple file formats, duplication of information in different files. Data redundancy means, the same information is repeated in several files.

- **Difficulty in accessing data**

- Need to write a new program to carry out each new task

- **Data isolation**

- Multiple files and formats. The data is scattered in various files with different 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

- **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**

# Database Management System (DBMS)

- DBMS contains information about a particular enterprise
  - Collection of interrelated data
  - Set of programs to access the data
  - An environment that is both *convenient* and *efficient* to use
- Database Applications:
  - Banking: transactions
  - Airlines: reservations, schedules
  - Universities: registration, grades
  - Sales: customers, products, purchases
  - Online retailers: order tracking, customized recommendations
  - Manufacturing: production, inventory, orders, supply chain
  - Human resources: employee records, salaries, tax deductions
- Databases can be very large.
- Databases touch all aspects of our lives

# Functions of DBMS:

- DBMS free the programmers from the need to worry about the organization and location of the data i.e. it shields the users from complex hardware level details.
- DBMS can organize process and present data elements from the database. This capability enables decision makers to search and query database contents in order to extract answers that are not available in regular Reports.
- Programming is speeded up because programmer can concentrate on logic of the application.
- It includes special user friendly query languages which are easy to understand by non programming users of the system.

# 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**

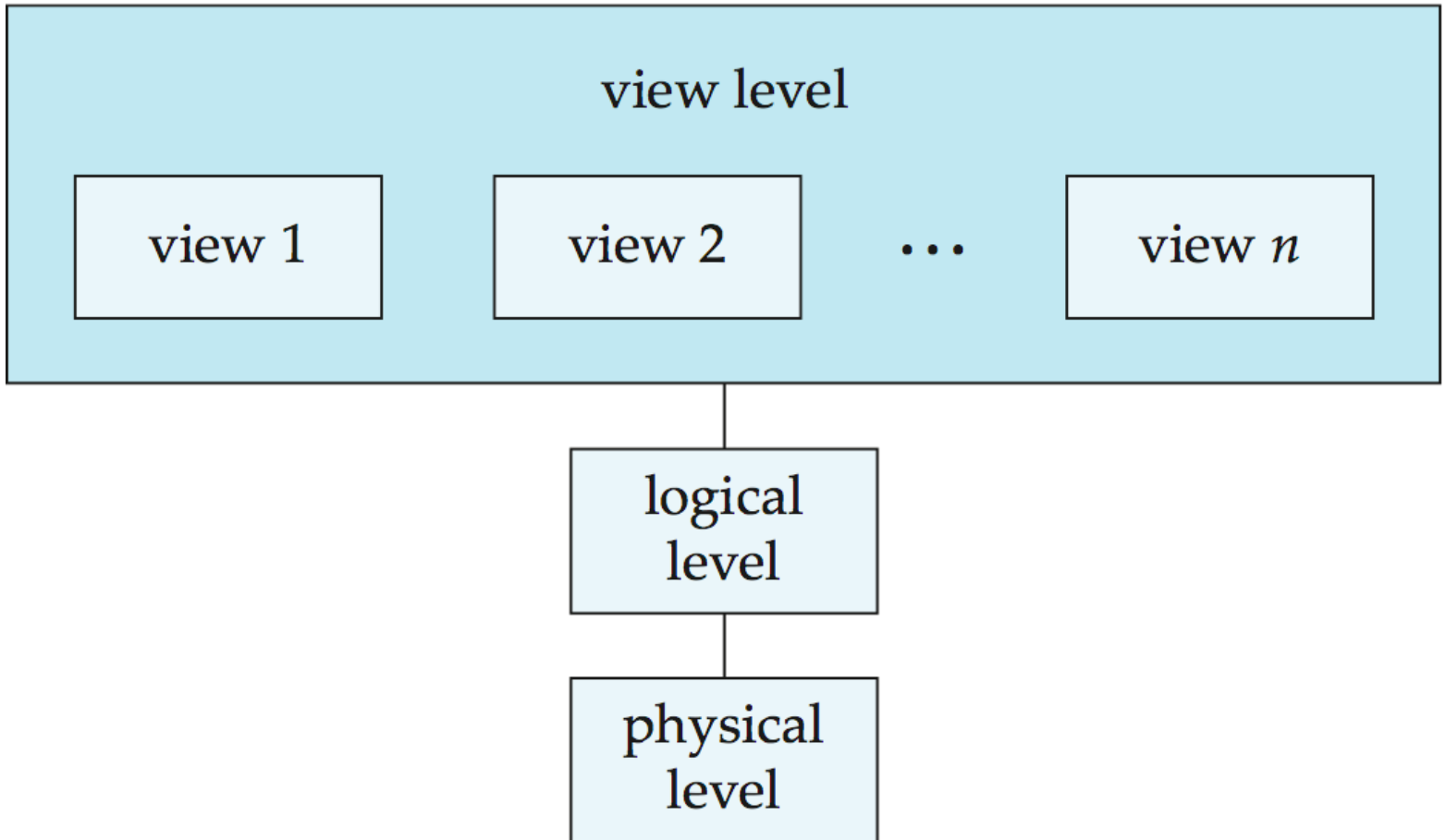
```
ID : string;  
name : string;  
dept_name : string;  
salary : integer;  
end;
```

- **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
    - ▶ 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.

# Database Administrator

The person to have central control of both the data and the programs that access those data is called a **database administrator (DBA)**. The functions of a DBA include:

- **Schema definition.** The DBA creates the original database schema by executing a set of data definition statements in the DDL.
- **Storage structure and access-method definition.**
- **Schema and physical-organization modification**
- **Granting of authorization for data access.**

It can regulate which part of the database various users can access

- **Routine maintenance:** database backup, ensuring free disk space, monitoring of jobs running on the database.

# Functions and responsibilities of DBAs

**DBA:** person in the organization who controls the design and the use of the database refers as DBA.

## **1. Schema Definition:**

- The DBA defines the logical Schema of the database. A Schema refers to the overall logical structure of the database.
- According to this schema, database will be developed to store required data for an organization.

## **2. Storage Structure and Access Method Definition:**

- The DBA decides how the data is to be represented in the stored database.

## **3. Assisting Application Programmers:**

- The DBA provides assistance to application programmers to develop application programs.

## **4. Physical Organization Modification:**

- The DBA modifies the physical organization of the database to reflect the changing needs of the organization or to improve performance.

## **5. Approving Data Access:**

- The DBA determines which user needs access to which part of the database.
- According to this, various types of authorizations are granted to different users.

## **6. Monitoring Performance:**

- The DBA monitors performance of the system. The DBA ensures that better performance is maintained by making changes in physical or logical schema if required.

## **7. Backup and Recovery:**

- Database should not be lost or damaged.
- The DBA ensures this periodically backing up the database on magnetic tapes or remote servers.
- In case of failure, such as virus attack database is recovered from this backup.

# Database Users and User Interfaces

❑ **Application Programmers** computer professionals who write application Programs, who interact with the system through DML calls which are embedded in a program written in a host language. Rapid application development (RAD) tools enable an application programmer to construct forms and reports with minimal programming effort.

Example: In a banking system include programs that generate payroll checks, that debit accounts, that credit accounts or that transfer funds between accounts.

❑ **Naive users** : unsophisticated users who interact with the system by invoking one of the application programs that have been written previously.

User interface is a forms interface, *reports generated from the database*

Example: a) consider a student, who during class registration period, wishes to register for a class by using a Web interface.

b) Bank teller who needs to transfer rs 50 from account A to B  
invokes a program called transfer.(3 things important)

# Database Users and User Interfaces

❑ **Sophisticated users:** Interact with the system without writing programs. They form their requests either using a database query language or by using tools such as data analysis software. Analysts who submit queries to explore data in the database fall in this category. Sophisticated users can be engineers, scientists, business analyst, who are familiar with the database. They can develop their own database applications according to their requirement. They don't write the program code but they interact the database by writing SQL queries directly through the query processor.

❑ **Specialized users:**

Specialized users who write specialized database applications such as computer-aided design systems, knowledgebase and expert systems, systems that store data with complex data types (i.e. graphics data and audio data)

# Data Models

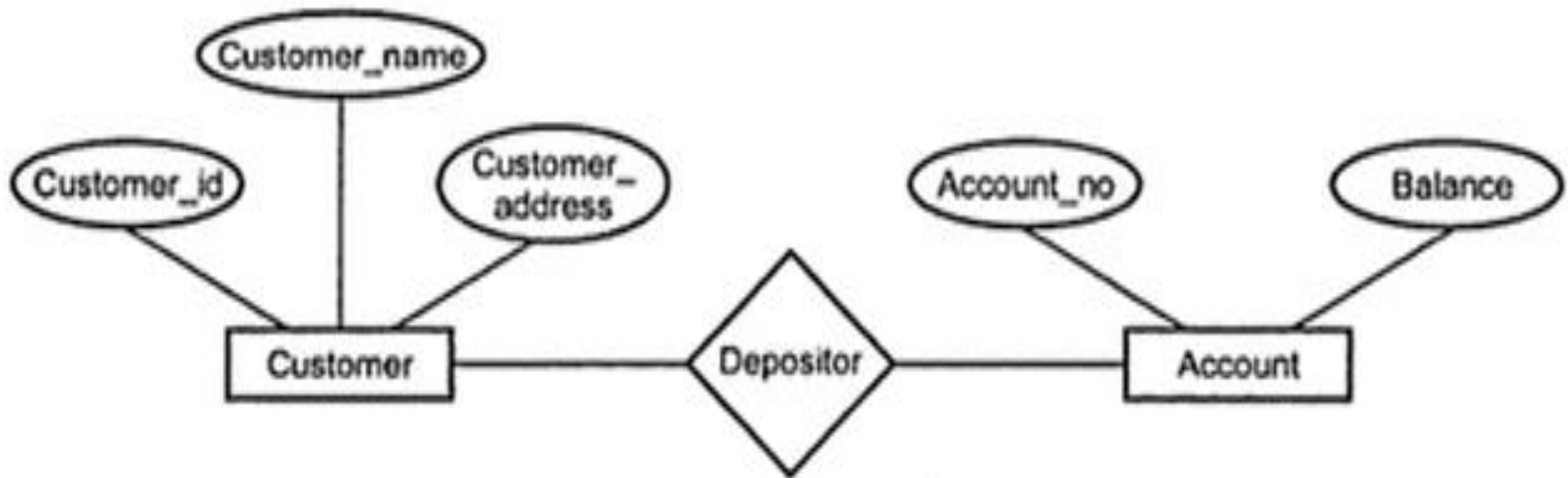
- A collection of tools for describing
  - Data
  - Data relationships
  - Data semantics
  - Data constraints
- Entity-Relationship data model (mainly for database design)
- Relational model
- Other older models:
  - Network model
  - Hierarchical model
- Semistructured data model (XML)



# E-R Model

- The entity-relationship (E-R) data model uses a collection of basic objects, called *entities*, and *relationships among these objects*
- Entity : is a 'thing' or 'object' in the real world which is distinguishable from other objects. Examples: person, student, doctor, etc.
- Relationship: is an association among several entities. Example: A depositor relationship associates a customer with each account that he has.

# A Sample E-R Model



An E-R Diagram for banking system

# E-R Model

## Advantages:

- Easy to develop relational model using E-R model
- Specifies Mapping Cardinalities
- Specifies keys like primary keys
- Generalization and Specialization is possible

## Disadvantages:

- Used only for database design not for implementation

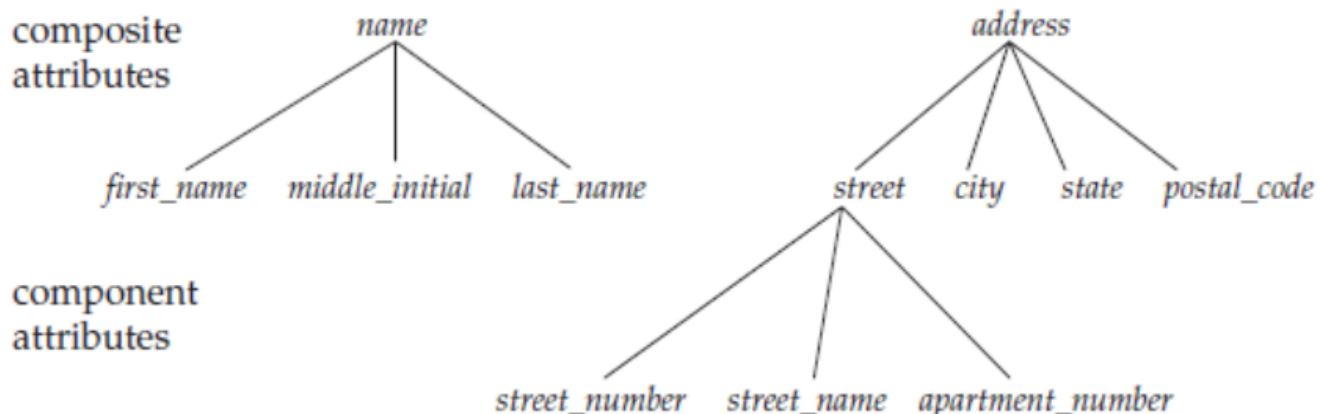
# E-R Model

## Attributes:

Each entity has attributes—the particular properties that describe it. For example, an EMPLOYEE entity may be described by the employee's name, age, address, salary, and job. For each attribute, there is a set of permitted values, called the domain, or value set, of that attribute. The domain of attribute `course_id` might be the set of all text strings of a certain length. Similarly, the domain of attribute `salary` of an employee might be a numeric value.

## Composite versus Simple (Atomic) Attributes:

Composite attributes can be divided into smaller subparts, which represent more basic attributes with independent meanings. Attributes that are not divisible are called **simple** or **atomic attributes**



# E-R Model

## **Single-Valued versus Multivalued Attributes.**

Most attributes have a single value for a particular entity; such attributes are called single-valued. For example, Age is a single-valued attribute of a person. In some cases an attribute can have a set of values for the same entity—for instance, a Colors attribute for a car, or a College\_degrees attribute for a person. Cars with one color have a single value, whereas two-tone cars have two color values. Similarly, one person may not have a college degree, another person may have one, and a third person may have two or more degrees; therefore, different people can have different numbers of values for the College\_degrees attribute. Such attributes are called multivalued.

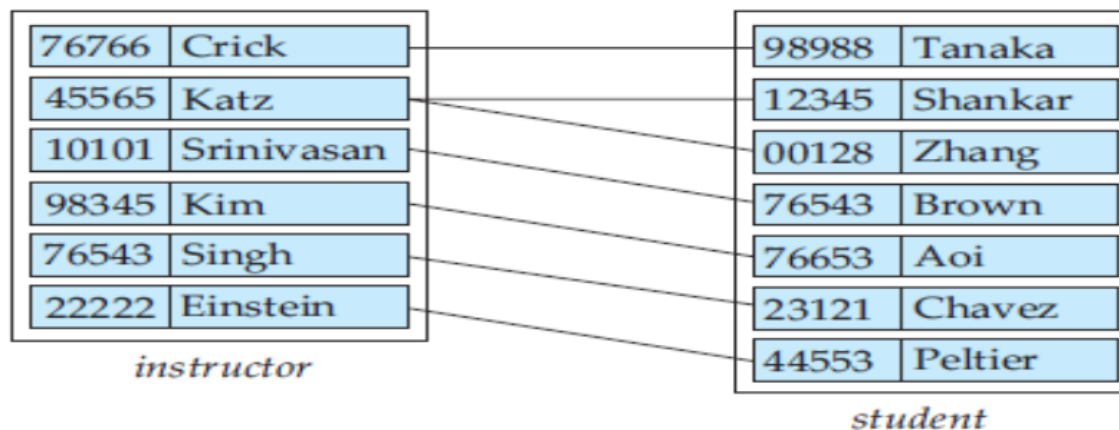
## **Stored versus Derived Attributes.**

In some cases, two (or more) attribute values are related—for example, the Age and Birth\_date attributes of a person. For a particular person entity, the value of Age can be determined from the current (today's) date and the value of that person's Birth\_date. The Age attribute is hence called a derived attribute and is said to be derivable from the Birth\_date attribute, which is called a stored attribute.

# E-R Model

A relationship is an association among several entities. For example, we can define a relationship advisor that associates instructor Katz with student Shankar. This relationship specifies that Katz is an advisor to student Shankar as shown in figure 2.

A relationship set is a set of relationships of the same type. Formally, it is a mathematical relation on  $n \geq 2$  entity sets. If  $E_1, E_2, \dots, E_n$  are entity sets, then a relationship set  $R$  is a subset of  $\{(e_1, e_2, \dots, e_n) \mid e_1 \in E_1, e_2 \in E_2, \dots, e_n \in E_n\}$ , where  $(e_1, e_2, \dots, e_n)$  is a relationship.



# E-R Model

## Mapping Cardinalities:

Mapping cardinalities, or cardinality ratios, express the number of entities to which another entity can be associated via a relationship set. Mapping cardinalities are most useful in describing binary relationship sets, although they can contribute to the description of relationship sets that involve more than two entity sets.

For a binary relationship set  $R$  between entity sets  $A$  and  $B$ , the mapping cardinality must be one of the following:

- One-to-One. An entity in  $A$  is associated with at most one entity in  $B$ , and an entity in  $B$  is associated with at most one entity in  $A$ . (See Figure 3(a))

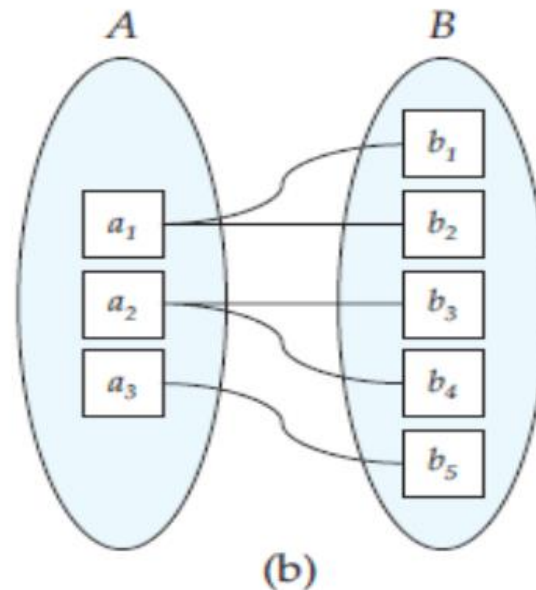
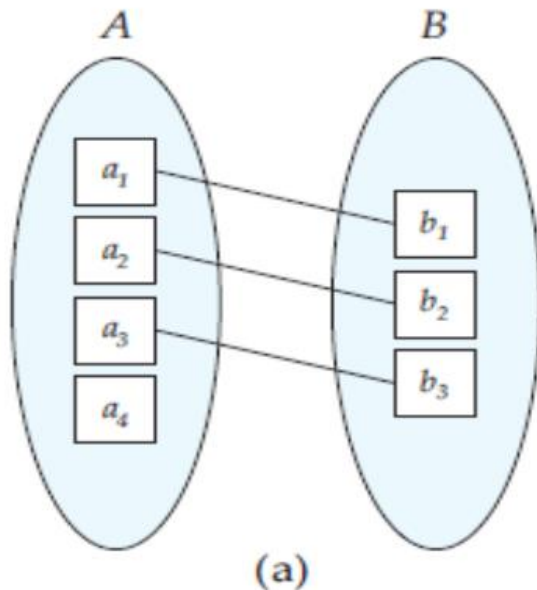
Examples: 1. Manager – Department (one manager can manage at most one department and one department can be managed by at most one manager only)

2. Employee – Medical Policy / Company Car

# E-R Model

- One-to-Many. An entity in A is associated with any number (zero or more) of entities in B. An entity in B, however, can be associated with at most one entity in A. Examples:

1. Department – Employee (one department can have many employees but one employee can have at most one department)
2. Train – Passenger
3. Supervisor – Employee
4. Advisor – Student
5. Customer – Account



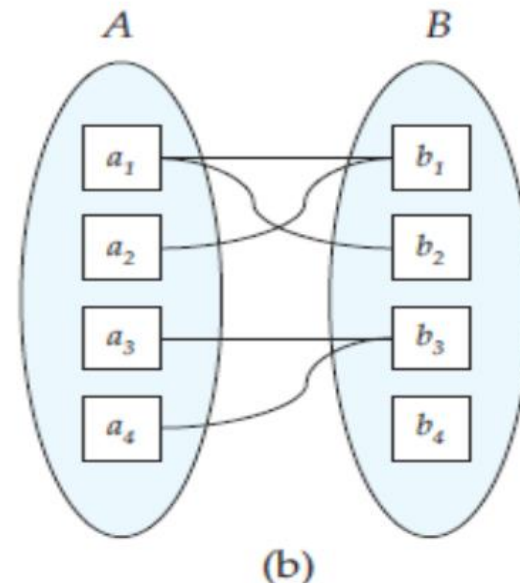
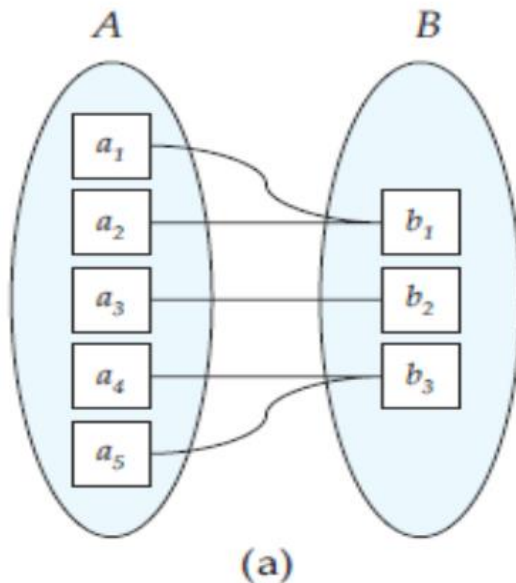


# E-R Model

- Many-to-One. An entity in A is associated with at most one entity in B. An entity in B, however, can be associated with any number (zero or more) of entities in A. (See Figure 4(a))

Examples:

1. Employee – Department (One employee is associated with at most one department whereas one department can have many employees)
2. Passenger – Train
3. Employee – Supervisor
4. Student – Advisor

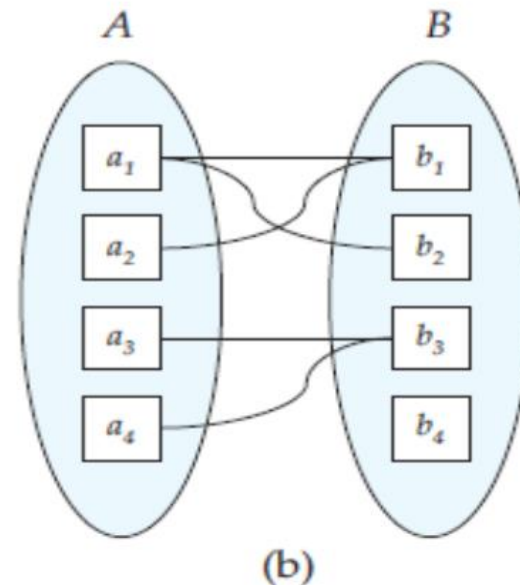
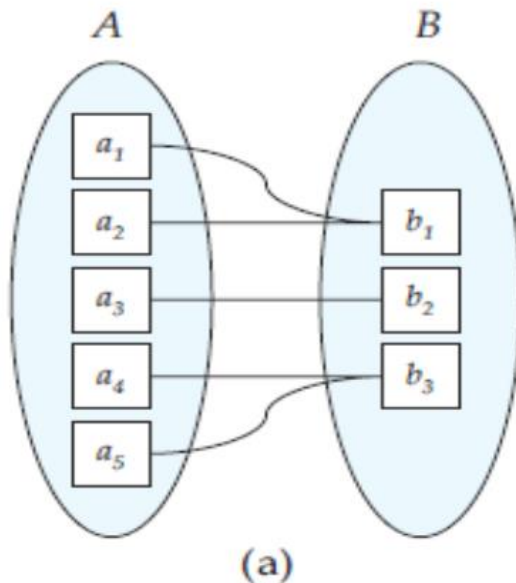


# E-R Model

- Many-to-Many. An entity in A is associated with any number (zero or more) of entities in B, and an entity in B is associated with any number (zero or more) of entities in A. (See Figure 4(b))

Examples:

1. Teacher - Course (One teacher can teach more than one courses and one particular course can be taught by more than one teachers)
2. Employee – Project

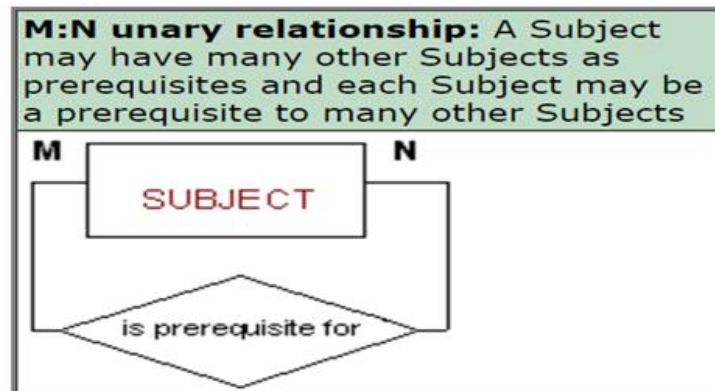
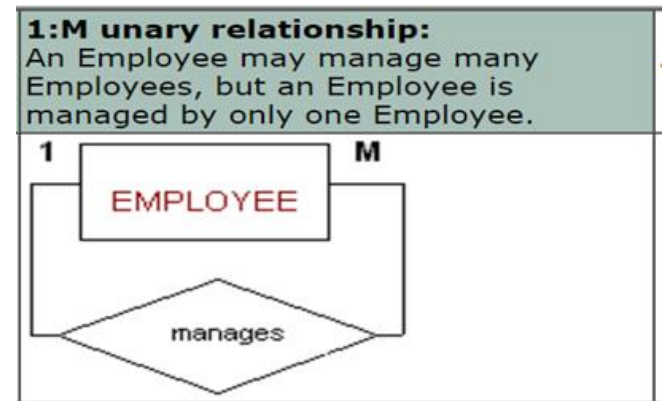
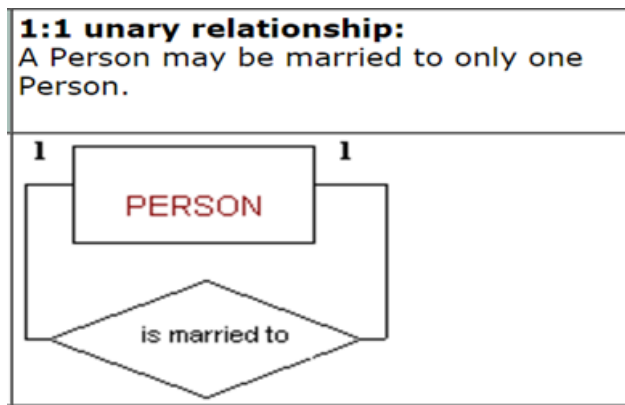


# E-R Model


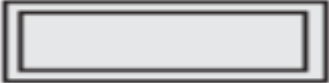




## Recursive Relationships:

Recursive relationships occur within unary relationships. ( same entity set) The relationship may be one to one, one- to- many or many- to- many. That is the cardinality of the relationship is unary. The connectivity may be 1:1, 1: M, or M: N.

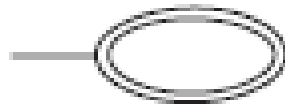
For example:



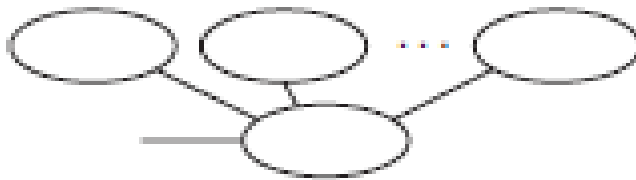
# E-R Diagram Symbol

Symbol	Meaning
	Entity
	Weak Entity
	Relationship
	Identifying Relationship
	Attribute
	Key Attribute

# E-R Diagram Symbol



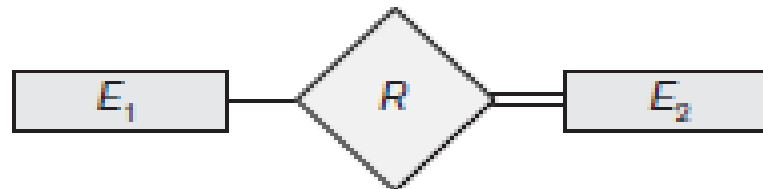
Multivalued Attribute



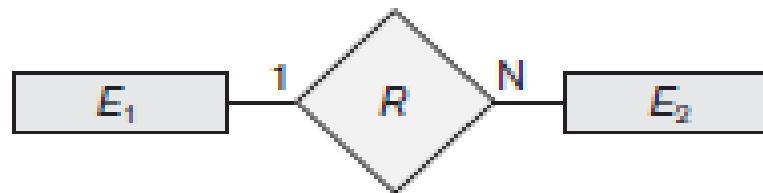
Composite Attribute



Derived Attribute



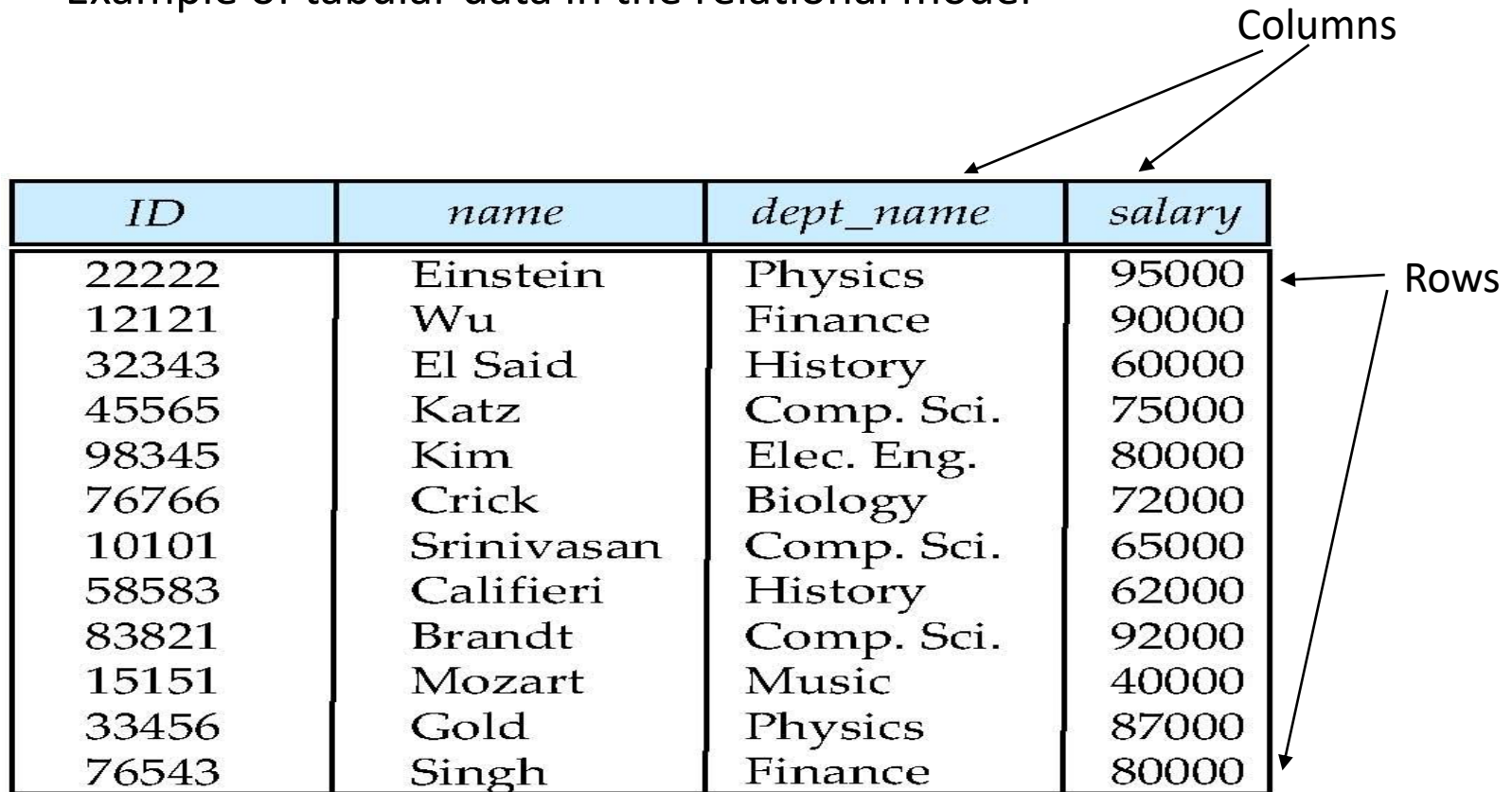
Total Participation of  $E_2$  in  $R$



Cardinality Ratio 1 : N for  $E_1:E_2$  in  $R$

# Relational Model

- Represents data and relationships among data by a collection of tables.
- Example of tabular data in the relational model



The diagram shows a table with four columns and twelve rows. Two arrows labeled 'Columns' point to the top of the 'dept\_name' and 'salary' columns. Two arrows labeled 'Rows' point to the right side of the first and last rows of the table.

<i>ID</i>	<i>name</i>	<i>dept_name</i>	<i>salary</i>
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

<i>ID</i>	<i>name</i>	<i>dept_name</i>	<i>salary</i>
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

<i>dept_name</i>	<i>building</i>	<i>budget</i>
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 Model

## Advantages

- Structural Independence: database structure can be changed without affecting DBMS' capability to access data
- Conceptual Simplicity: Designer has to concentrate only on logical view of the database not on the physical data storage details
- Design, Implementation, Maintenance and Usage ease
- Good for ad hoc requests
- Simpler to navigate
- Greater flexibility

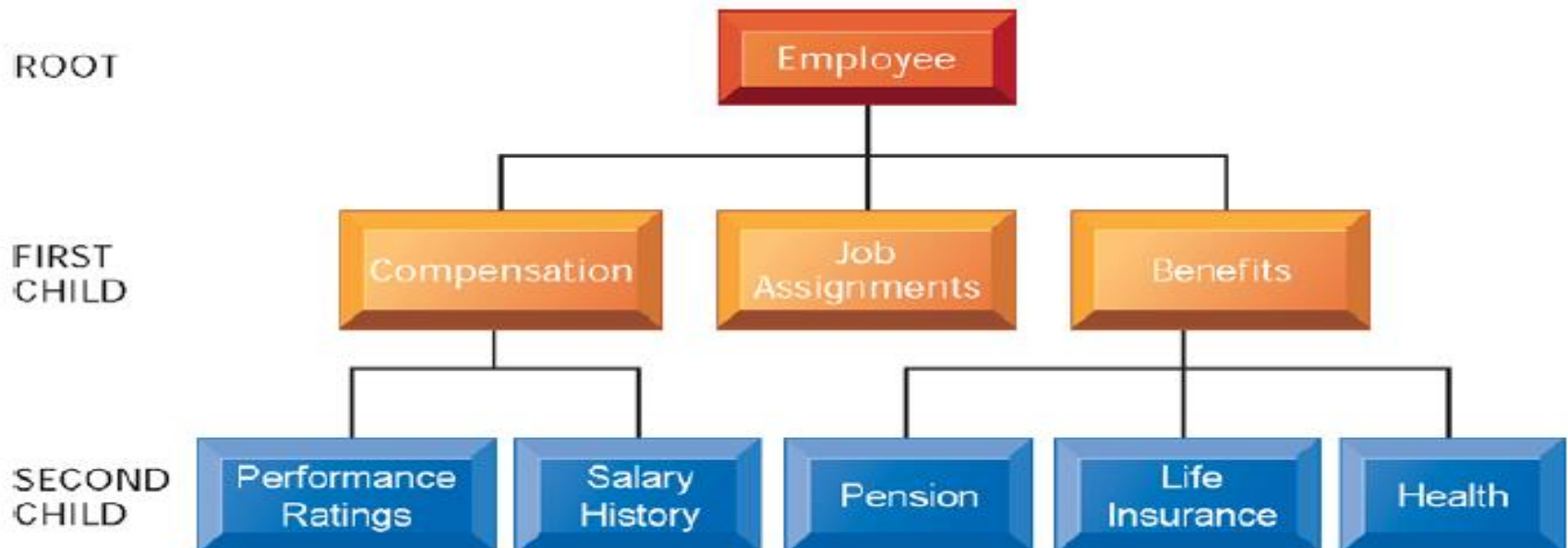
## Disadvantages

- H/W and S/W overheads
- Not good for Transaction processing modeling as hierarchical and network models
- Slower processing time than hierarchical and network models

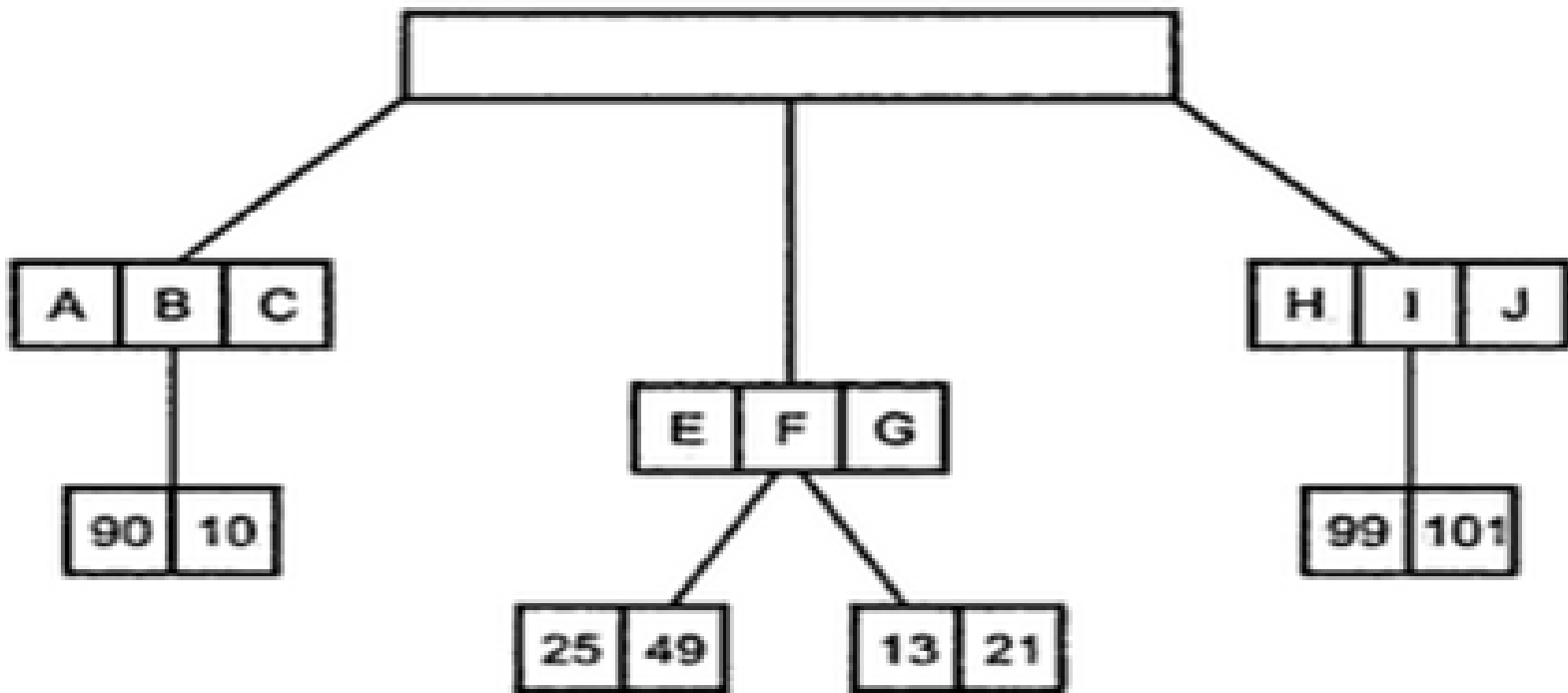


# Hierarchical Model

- The hierarchical DBMS is used to model one-to-many relationships, presenting data to users in a treelike structure
- Within each record, data elements are organized into pieces of records called segments
- Top level segment is called the root. An upper segment is connected logically to a lower segment in a parent–child relationship
- A parent segment can have more than one child, but a child can have only one parent

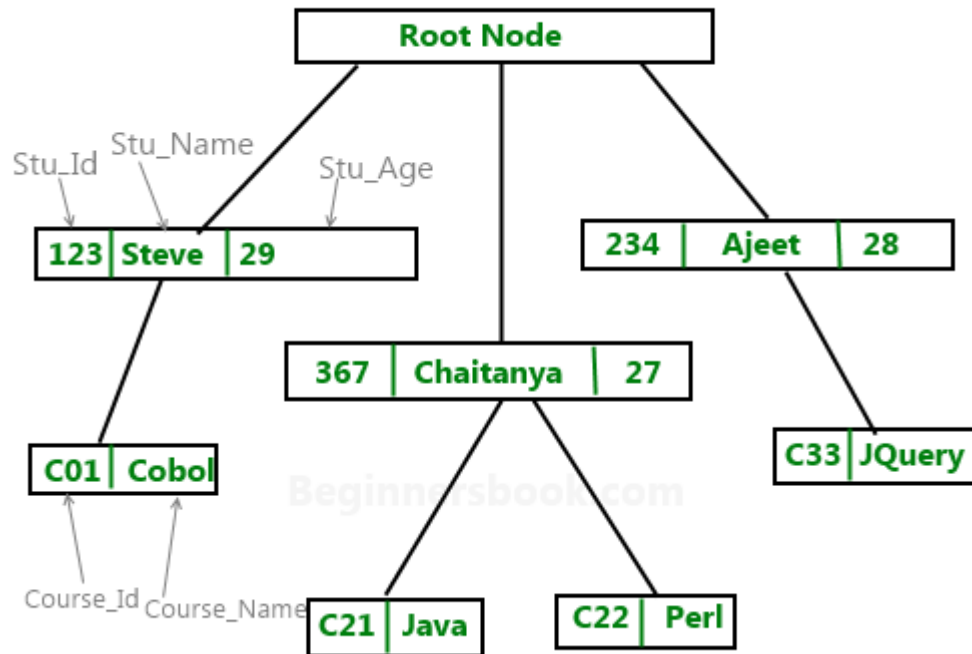


# Hierarchical Model



Bank account holders records in hierarchical Model

## Example of hierarchical model



**Example of hierarchical data represented as relational tables:** The above hierarchical model can be represented as relational tables like this:

# Hierarchical Model

## Advantages

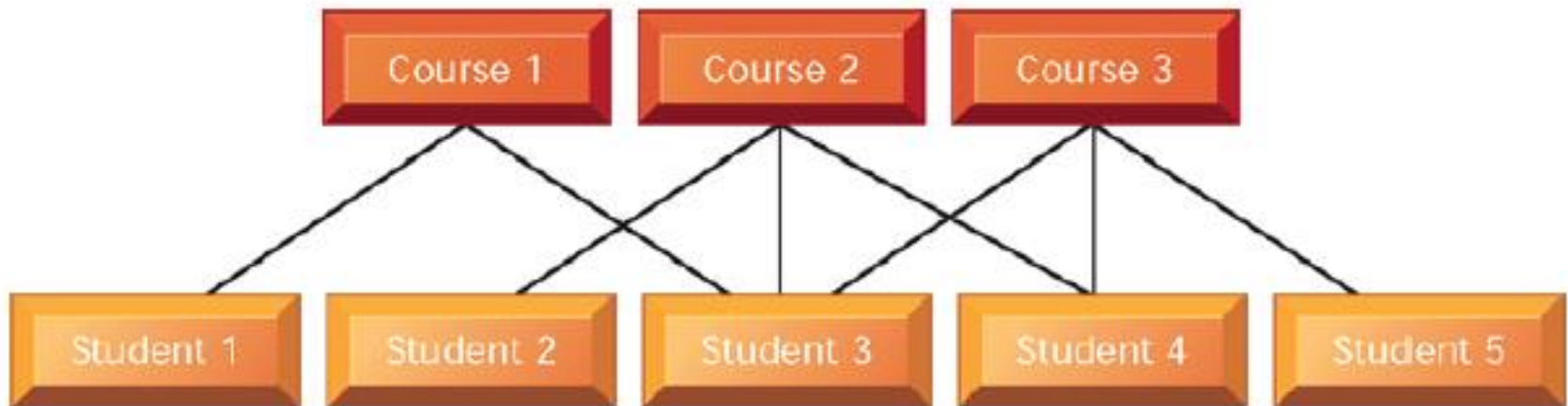
- High speed of access to large data sets
- Ease of updates
- Design is simple
- First model to introduce the security in DBMS
- Very efficient when the users require large number of transactions using data whose relationships are fixed.

## Disadvantages

- Implementation Complexity
- Change in database requires to make necessary change in application programs which accesses the database
- Uses physical storage paths to navigate the different segments. So if physical structure is changed, the application has to be modified too. (lack of structural independence)

# Network Model

- The network DBMS depicts data logically as many-to-many relationships. That is parents can have multiple children, and a child can have more than one parent.
- Example :There are many courses in a university and many students. A student takes many courses, and a course has many students.
- Banks, insurance companies, etc. Are using this legacy data model.



Student course relationship

# Network Model

## Advantages

- Conceptually simple and easy to design
- Can handle 1:n as well as n:n relationships
- The changes in data do not require changes to the application programs

## Disadvantages

- Detailed structured knowledge is required
- Lack of structural independence (physical path)

# Database Engine

A **database engine** (or **storage engine**) is the underlying software component that a database management system (DBMS) uses to create, read, update and delete (CRUD) data from a database.

- Storage manager
- Query processing
- Transaction manager

# Storage Management

- **Storage manager** is a program module that provides the interface between the low-level data stored in the database and the application programs and queries submitted to the system.
- The storage manager is responsible to the following tasks:
  - Interaction with the OS file manager
  - Efficient storing, retrieving and updating of data



The storage manager components includes:

**Authorization and integrity manager:**

Which test for the satisfaction of integrity constraints and checks the authority of users to access the data.

**Transaction Manager:**

Which ensures that the database remains in a consistent state despite system failure, and that concurrent transaction execution proceed without conflicting.

**File Manager:**

which manages the allocation of space on disk and the data structures used to represent information stored on disk.

**Buffer Manager:**

Which is responsible for fetching data from disk storage into main memory, and deciding what data to cache in main memory.

# Query Processing (Cont.)

The query processor components includes;

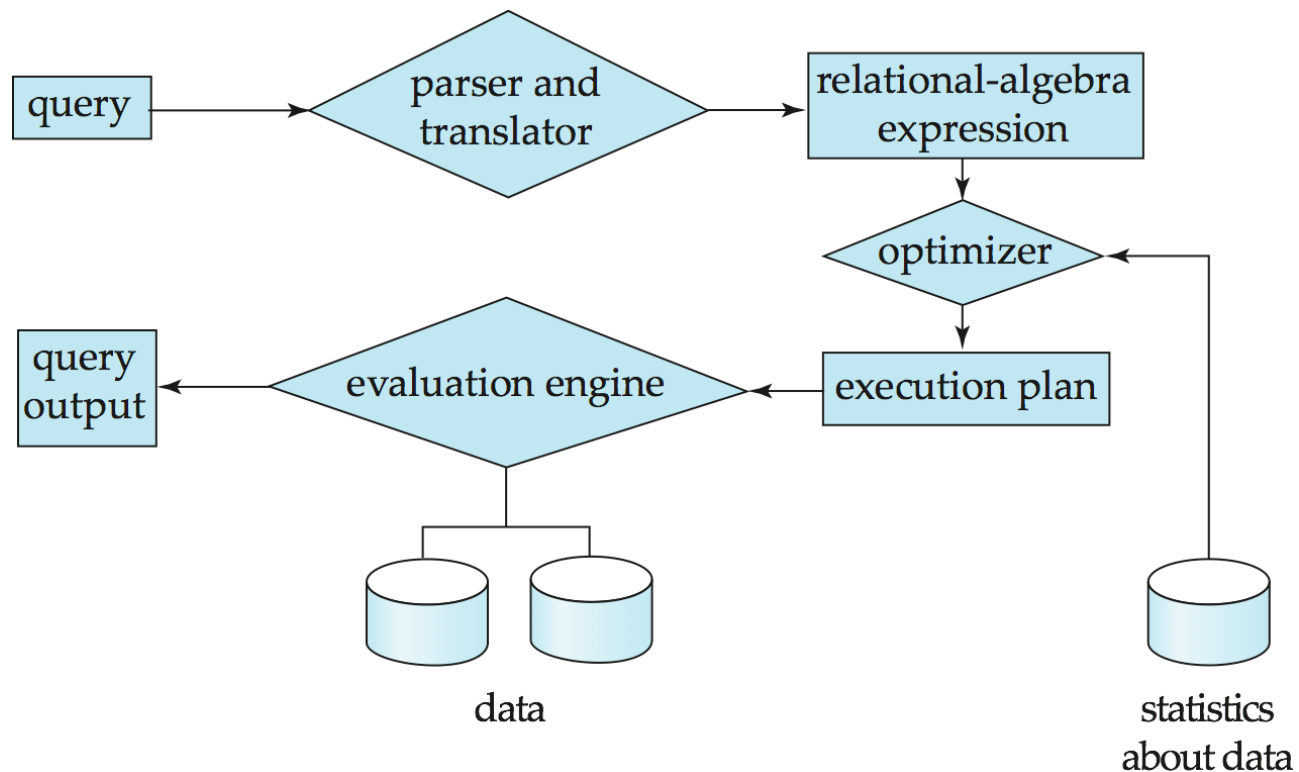
**DDL Interpreter:** which interprets DDL statements and records the definitions in the data dictionary.

**DML compiler:** which translates DML statements in a query language into an evaluation plan consisting of low level instructions that the query evaluation engine understand. It also performs query optimization-finding best cost evaluation plan.

**Query Evaluation:** which evaluates low level instruction generated by the DML compiler.

# Query Processing

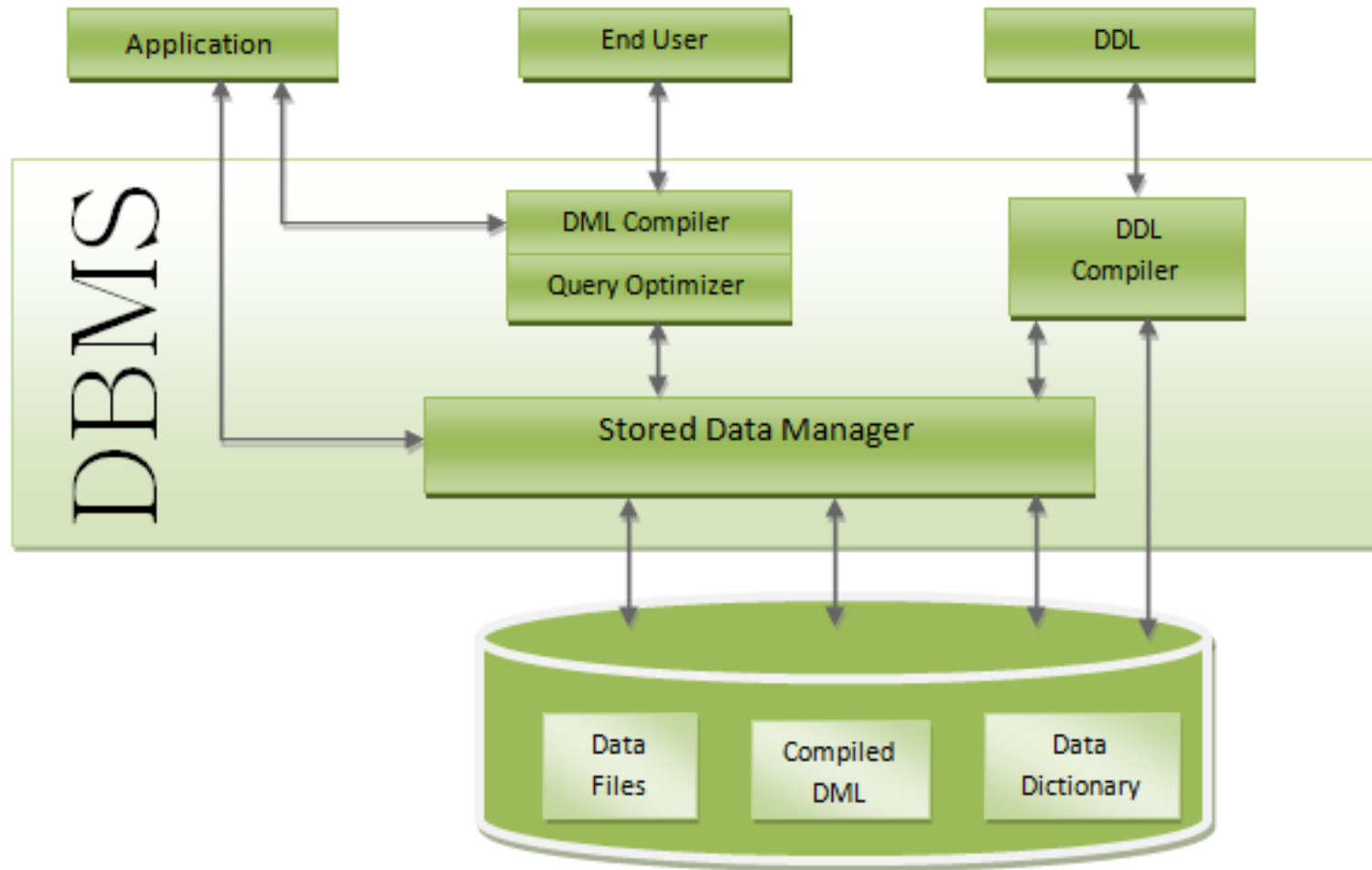
1. Parsing and translation
2. Optimization
3. Evaluation



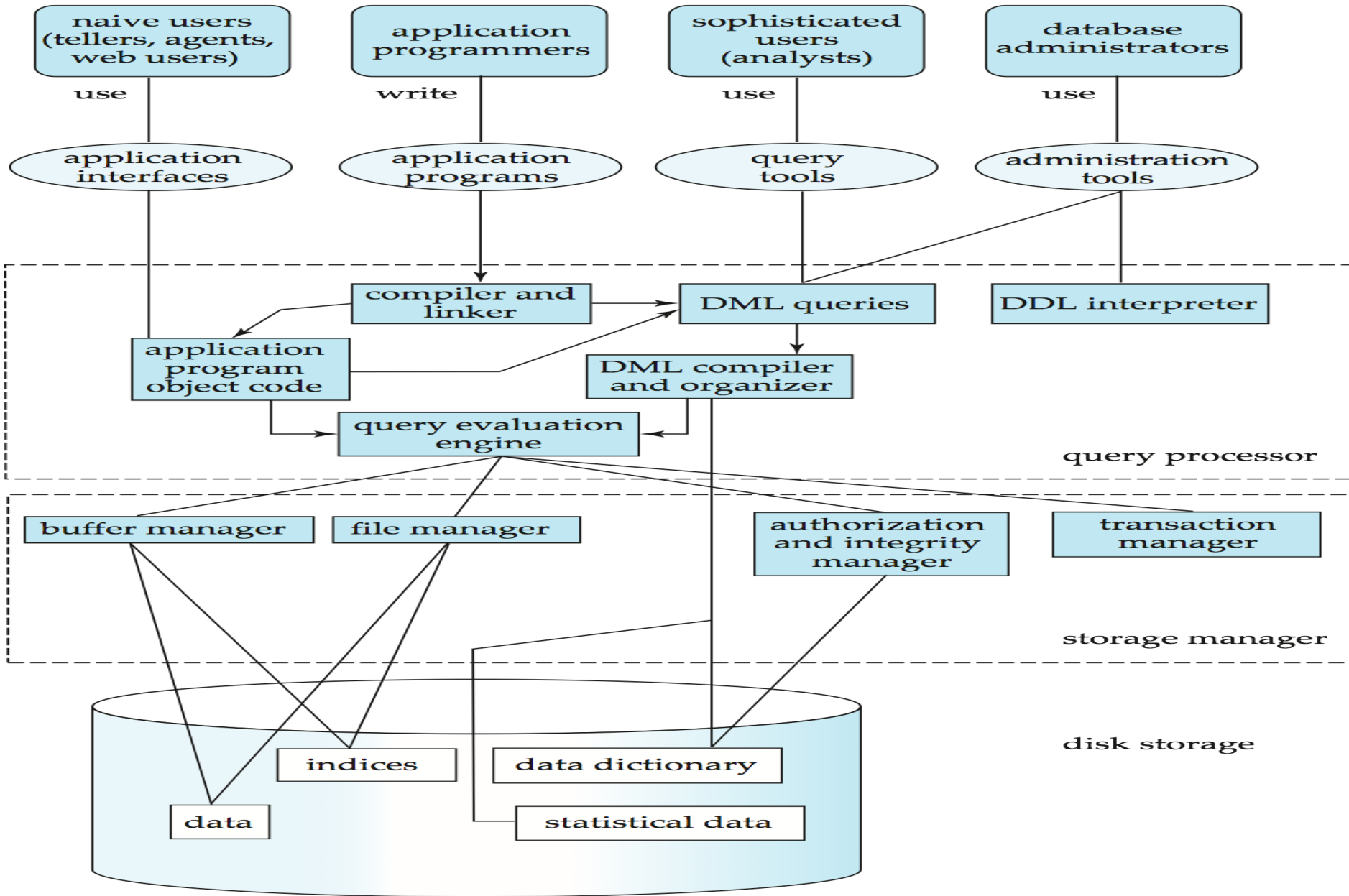
# Transaction Management

- What if the system fails?
- What if more than one user is concurrently updating the same data?
- A **transaction** is a collection of operations that performs a single logical function in a database application
- **Transaction-management component** ensures that the database remains in a consistent (correct) state despite system failures (e.g., power failures and operating system crashes) and transaction failures.
- **Concurrency-control manager** controls the interaction among the concurrent transactions, to ensure the consistency of the database.

# Database System Architecture



# Database System Architecture



# Database System Architecture

## **DDL Compiler:**

Data Description Language compiler processes schema definitions specified in the DDL.

It includes metadata information such as the name of the files, data items, storage details of each file, mapping information and constraints etc.

## **DML Compiler and Query optimizer:**

The DML commands such as insert, update, delete, retrieve from the application program are sent to the DML compiler for compilation into object code for database access.

The object code is then optimized in the best way to execute a query by the query optimizer and then send to the data manager.

# Database System Architecture

## Data Manager:

The Data Manager is the central software component of the DBMS also known as Database Control System.

The Main Functions Of Data Manager Are:

- Convert operations in user's Queries coming from the application programs or combination of DML Compiler and Query optimizer which is known as Query Processor from user's logical view to physical file system.
- Controls DBMS information access that is stored on disk.
- It also controls handling buffers in main memory.
- It also enforces constraints to maintain consistency and integrity of the data.
- It also synchronizes the simultaneous operations performed by the concurrent users.
- It also controls the backup and recovery operations.



# Database System Architecture

## Data Dictionary:

Data Dictionary, which stores metadata about the database, in particular the schema of the database.

names of the tables, names of attributes of each table, length of attributes, and number of rows in each table.

Detailed information on physical database design such as storage structure, access paths, files and record sizes.

Usage statistics such as frequency of query and transactions.

Data dictionary is used to actually control the data integrity, database operation and accuracy. It may be used as a important part of the DBMS

# Database System Architecture

## **Data Files:**

Which store the database itself.

## **Compiled DML:**

The DML compiler converts the high level Queries into low level file access commands known as compiled DML.

## **End Users:**

The second class of users then is end user, who interacts with system from online workstation or terminals.

Use the interface provided as an integral part of the database system software.

User can request, in form of query, to access database either directly by using particular language, such as SQL, or by using some pre-developed application interface.

Such request are sent to query evaluation engine via DML pre-compiler and DML compiler

The query evaluation engine accepts the query and analyses it.

It finds the suitable way to execute the compiled SQL statements of the query.

Finally, the compiled SQL statements are executed to perform the specified operation

# Database System Architecture

## **Query Processor Units:**

Interprets DDL statements into a set of tables containing metadata. Translates DML statements into low level instructions that the query evaluation engine understands.

Converts DML statements embedded in an application program into procedure calls into the host language.

Executes low level instructions generated by DML compiler.

DDL Interpreter

DML Compiler

Embedded DML Pre-compiler

Query Evaluation Engine

## **Storage Manager Units**

- Checks the authority of users to access data.
- Checks for the satisfaction of the integrity constraints.
- Preserves atomicity and controls concurrency.
- Manages allocation of space on the disk.