

Subject: Database Management System (DBMS)

CHAPTER-1 Introduction

By: Kalpana Karki

Contents

1. Introduction

1.1 Data, Database and DBMS

1.2 Objectives of Database

1.3 Needs of DBMS for organization and others

1.4 Data abstraction, Data Independence

1.5 Schema and Instances

1.6 Three schema Approach

1.7 Database administrator and Users

1.8 DBMS Languages

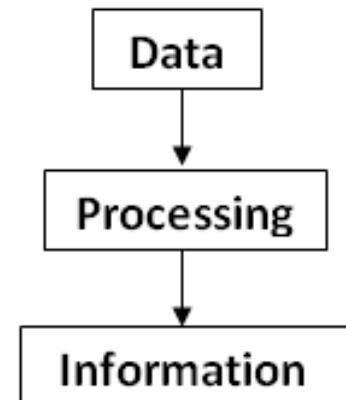
1.1 Data, Database and DBMS

Data:

- Raw and unorganized facts that need to be processed.
- Data is a collection of facts, figures and statistics related to an object. Data can be processed to create useful information.
- It also enables the user to make better decision for future.
- Example: student's test scores

Information:

- Processed data
- It is meaningful and has direct application.
- Example: average score of a class



Database

- Collection of related data.
- A data base is a collection of information that is organization so that it can easily be accessed, managed, and updated.
- It can store huge amount of data and information.

DBMS (Database Management System)

- DBMS is a software package designed to define, manipulate, retrieve and manage data in a database.
- A DBMS generally manipulates the data itself, the data format, field names, record structure and file structure.
- It also defines rules to validate and manipulate data in database.
- It is designed to allow the definition, creation, querying, update and administration of database.
- The data stores in a DBMS package can be accessed by multiple users and by multiple application programs like (SQL Server, Oracle, and Ms-Access).
- Well-known DBMS are:
 - MySQL, MS-SQL, Oracle, Sybase, PostgreSQL, Ms-Access, etc.

- DBMS provides interface database, user and applications.

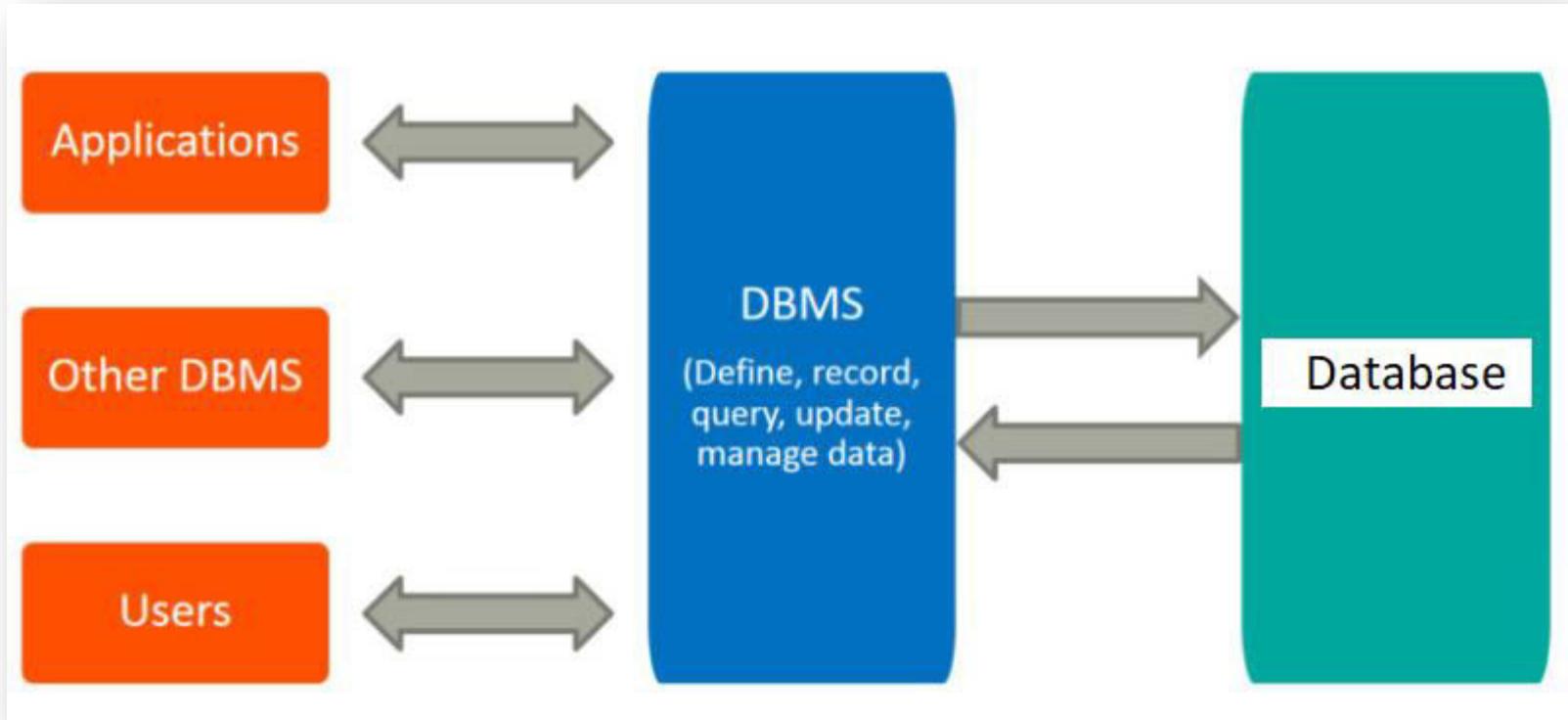


Figure: DBMS

1.2 Objectives of Database/DBMS

Database/DBMS should do following:

- Provide efficient storage, update and retrieval of data.
- Making easy access to data for the authorized user.
- Providing prompt response to users requests for data.
- Eliminate redundantly (Duplicate) data.
- Allow multiple users to be active at one time.
- Provide data integrity.
- Provide security with a user access privilege.
- Combining interrelated data to generate a report
- Provide multiple views for same data.
- Provide backup and recovery during failure.
- Incorporate changes easily and quickly.

Components of DBMS

- Hardware (computer hardware)
- Software (application software like DBMS)
- Data (data of database)
- User (DBA, software developer and end user)
- Procedure (general instructions to use DBMS)
- Database language (Programming languages used to write commands to access, insert, update, delete and store data in database)

Advantages of DBMS:

- Improved data sharing
- Improved data security
- Improved data access
- Improved data integrity
- Reduce data redundancy/duplication
- Inconsistency can be avoided
- Concurrency control
- Backup and recovery
- Standards can be enforced
- Better service to user

Disadvantages of DBMS:

- Cost of hardware and software
- Management complexity
- Larger file size
- Cost of staff (e.g. cost of DBA)
- Need of trained staff
- Performance (due to large size of database, sometime performance degrades)

Types of DBMS

1. On the basis of number of users
 - i. Single-user DBMS
 - ii. Multi-user DBMS
2. On the basis of site location/ DB distribution
 - i. Centralized DBMS
 - ii. Distributed DBMS
 - iii. Parallel DBMS
 - iv. Client-server DBMS
3. On the basis of structure/ data model
 - i. Hierarchical DBMS
 - ii. Network DBMS
 - iii. Relational DBMS
 - iv. Object-oriented DBMS
4. On the basis of purpose
 - i. General purpose DBMS
 - ii. Specific purpose DBMS

1.3 Needs of DBMS for organization and others

- Improved data sharing
- Improved data security
- Effective data integration
 - Integrated picture of organization's operations
- Data consistency is ensured.
- Increase productivity of end user
- Quick decision making
- Backup and recovery, etc.

Database Approach Vs File system

- Database approach stores data in database with different structures.
- **File system** controls how data is stored and retrieved.

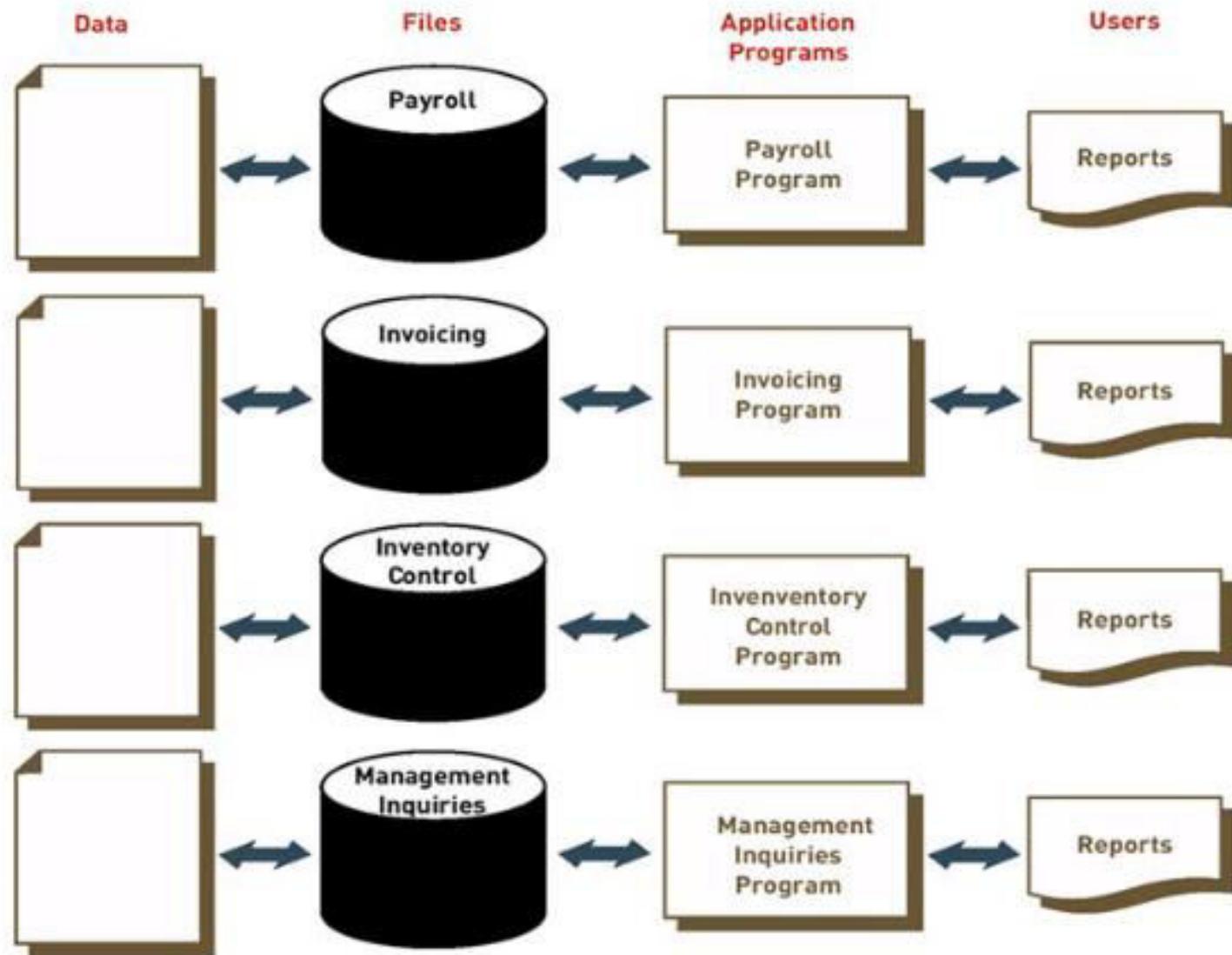


Figure: File system

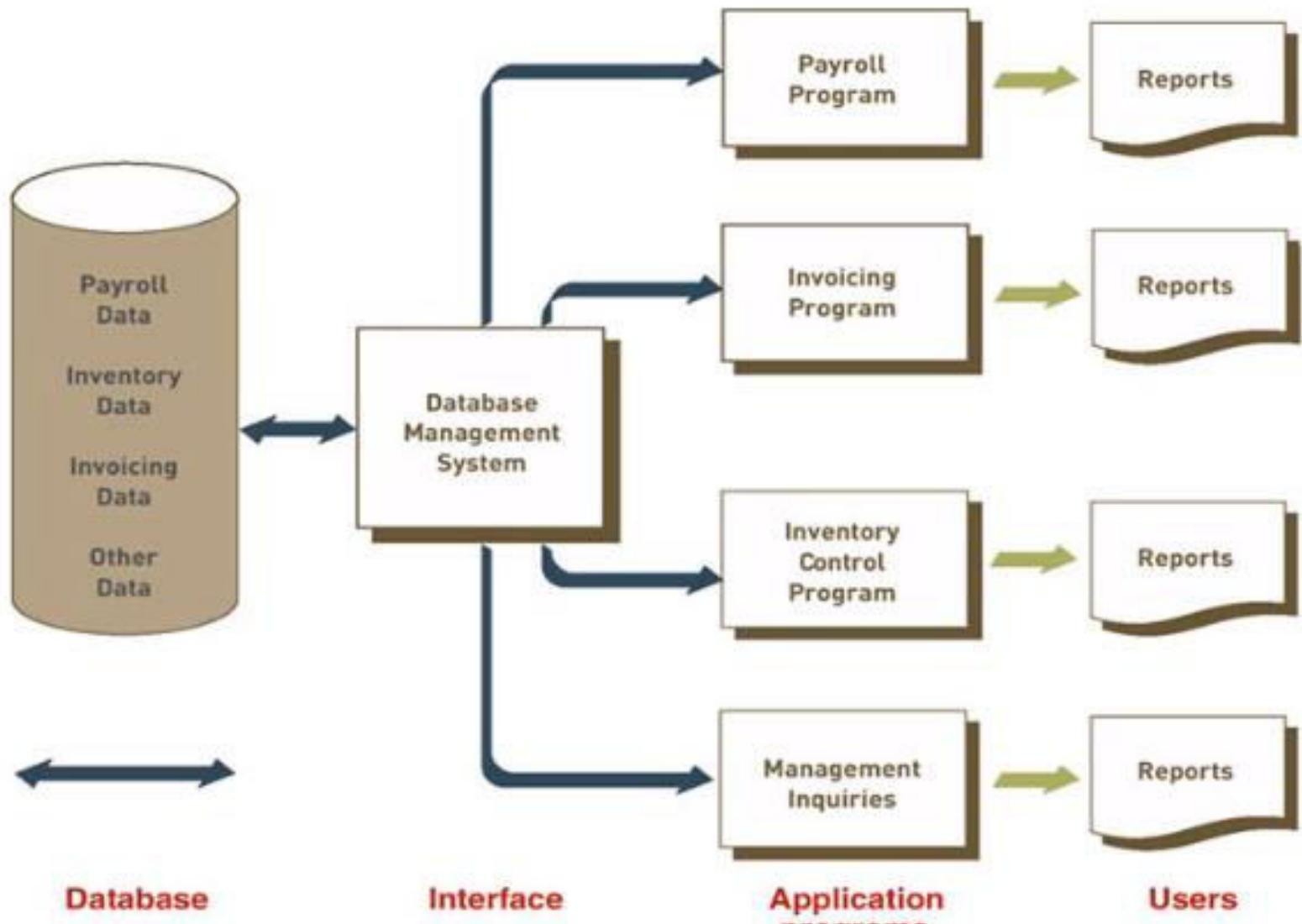


Figure: Database Approach

Assignments:

1. Compare file system and database system.

DBMS Architectures

- This architecture has three levels:
 1. External level
 2. Conceptual level
 3. Internal level

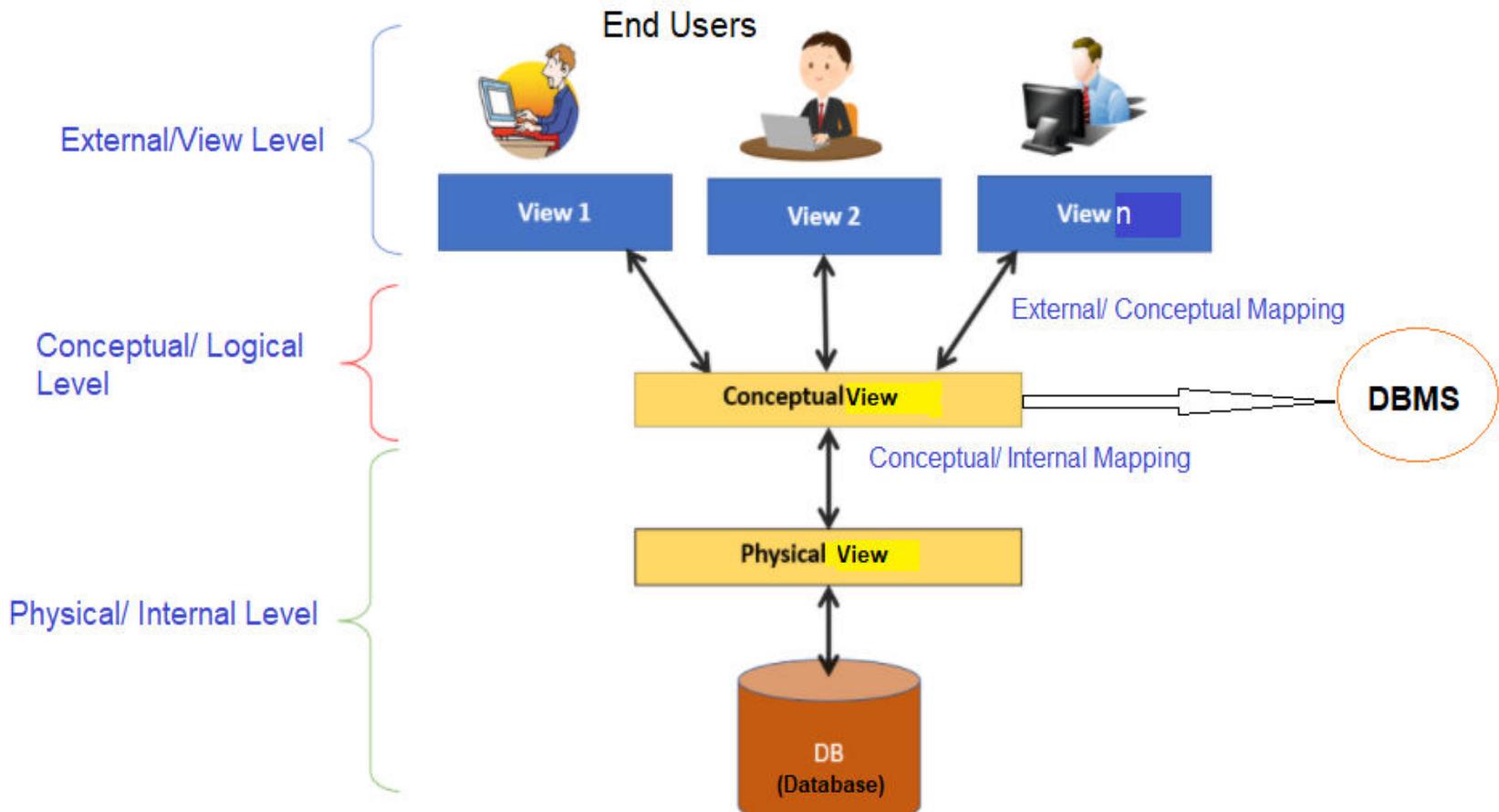


Figure: Three-tier DBMS Architecture

1. External/View level:

- It is the top level of DBMS architecture.
- It includes users of the database.
- It has highest level of abstraction.
- Different views may have different representation of the same data.
- The reason this level is called “view” is because several users can view their desired data from this level which is internally fetched from database with the help of conceptual and internal level mapping.

2. Conceptual/ Logical Level:

- It is the intermediate level of DBMS architecture.
- It hides the details of physical storage structures from external level.
- It is the global view of the database.
- It describes the structures of the whole database for a community of users.
- The whole design of the database such as relationship among data, schema of data etc. are described in this level.
- Database constraints and security are also implemented in this level of architecture. This level is maintained by DBA (database administrator).

3. Physical/ Internal Level:

- This is the lowest level of the DBMS architecture.
- This level describes how the data is actually stored in the storage devices.
- This level is also responsible for allocating space to the data (data storage).
- It describes details of data storage and access paths for the database.

1.4 Data abstraction, Data Independence

Data Abstraction

- The major purpose of DB system is to provide its users with an abstract view of data i.e. the system hides the details of how the data are stored, created and maintained.
- It is a process of hiding the implement details (such as how the data are stored and maintained) and representing only the essential features to simplify user's interaction with the system.
- The DB system developer hides the complexity to DB user through several levels of abstraction.
- To simplify user's interaction with the system, the complexity is hidden from the database users through several levels of abstraction, which are:
 1. View/external level
 2. Logical/conceptual level
 3. Physical/internal level

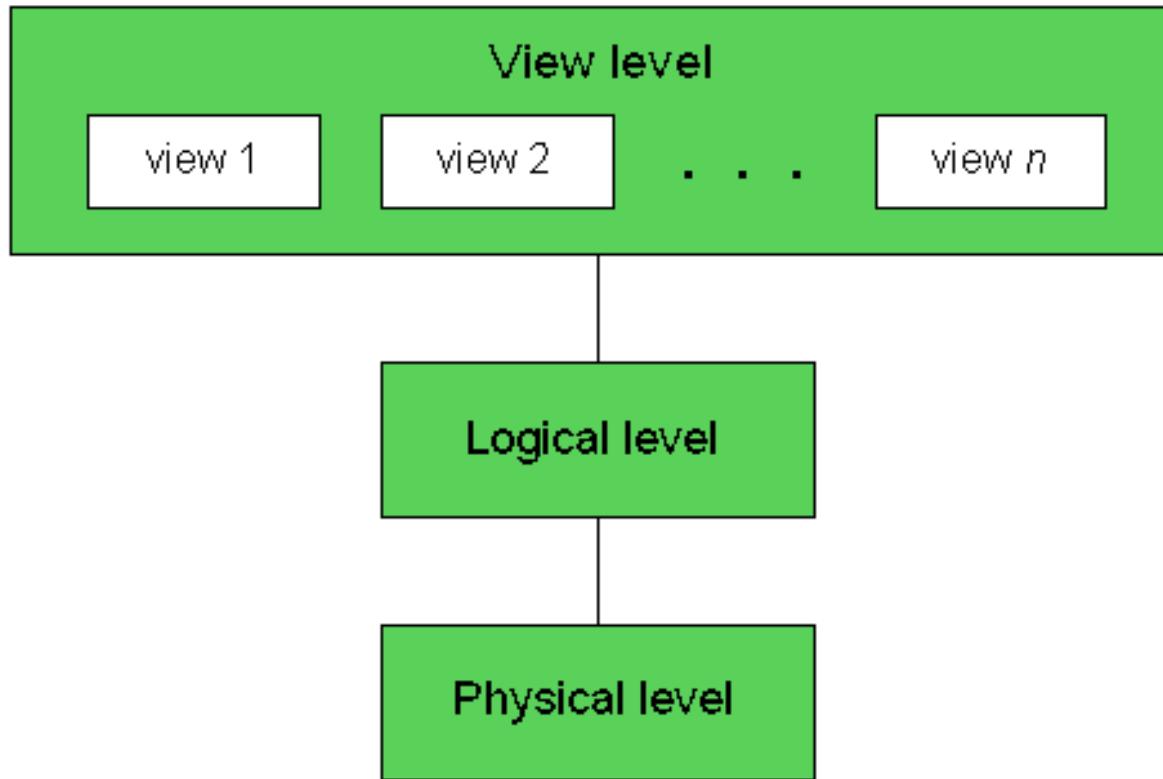


Figure: The three level of data abstraction

- **Physical Level:**
 - Lowest level of abstraction.
 - Describes **how** the data are stored.
 - Complex low-level data structures described in detail.
- **Logical / Conceptual Level:**
 - Next highest level of abstraction.
 - Describes **what** data are stored and **what relationships** exist among those data.
 - Database administrator level.
- **View Level:**
 - Highest level of abstraction.
 - Describes only part of the database for a particular group of users.
 - Can be many different views of a database.

Data Independence

- It is defined as a property of DBMS that helps you to change the Database schema at one level of a database system without requiring to change the schema at the next higher level.
- Data independence helps you to keep data separated from all programs that make use of it.
- **Main purpose of data independence:** achieving data independence in order to save time and cost required when the database is modified or altered.
- Types:
 1. Logical data independence
 2. Physical data independence

1. Logical data independence

- It refers characteristic of being able to modify the logical schema without affecting the external schema or application program.
- The user view of the data would not be affected by any changes to the conceptual view of the data.
 - These changes may include insertion or deletion of attributes, altering table structures entities or relationships to the logical schema etc.

2. Physical data independence

- It refers to the characteristic of being able to modify the physical schema without any alterations to the conceptual or logical schema.
- Example: Conceptual structure of the database would not be affected by any change in storage size of the database system server.

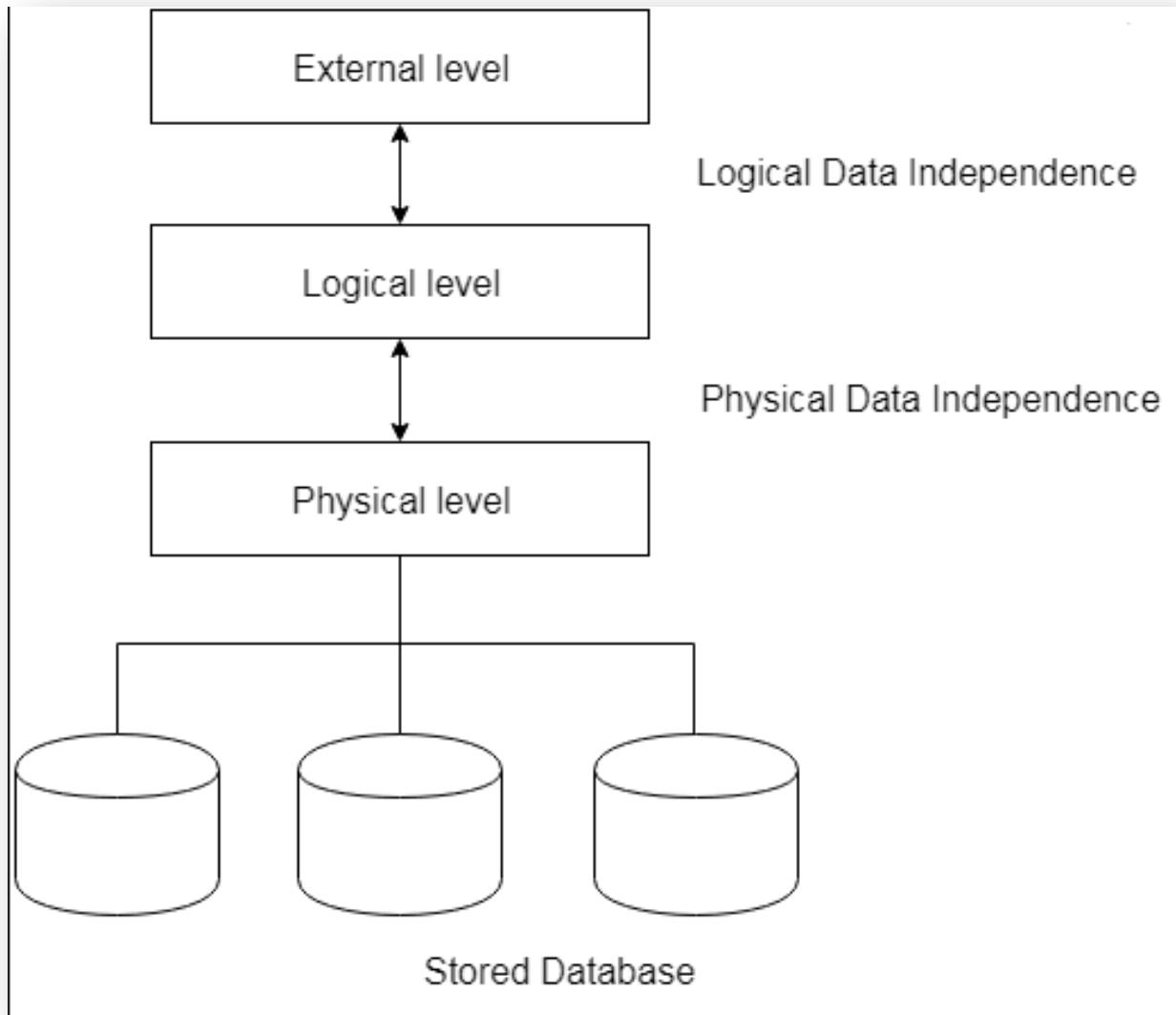


Figure: Data independence

1.5 Schema and Instances

Schema

- Design of a database is called the schema.
- Schema is of three types: Physical schema, logical schema and view schema.

Instance

- The data stored in database at a particular moment of time is called instance of database.
- Actual contents of database at a particular moment of time.

Example of schema:

Student

S_id	S_name	S_address	S_contact
------	--------	-----------	-----------

Example of instance:

Student

S_id	S_name	S_address	S_contact
1	Hari	Kathmandu	98456
2	Rita	Pokhara	98765

1.6 Three schema Approach

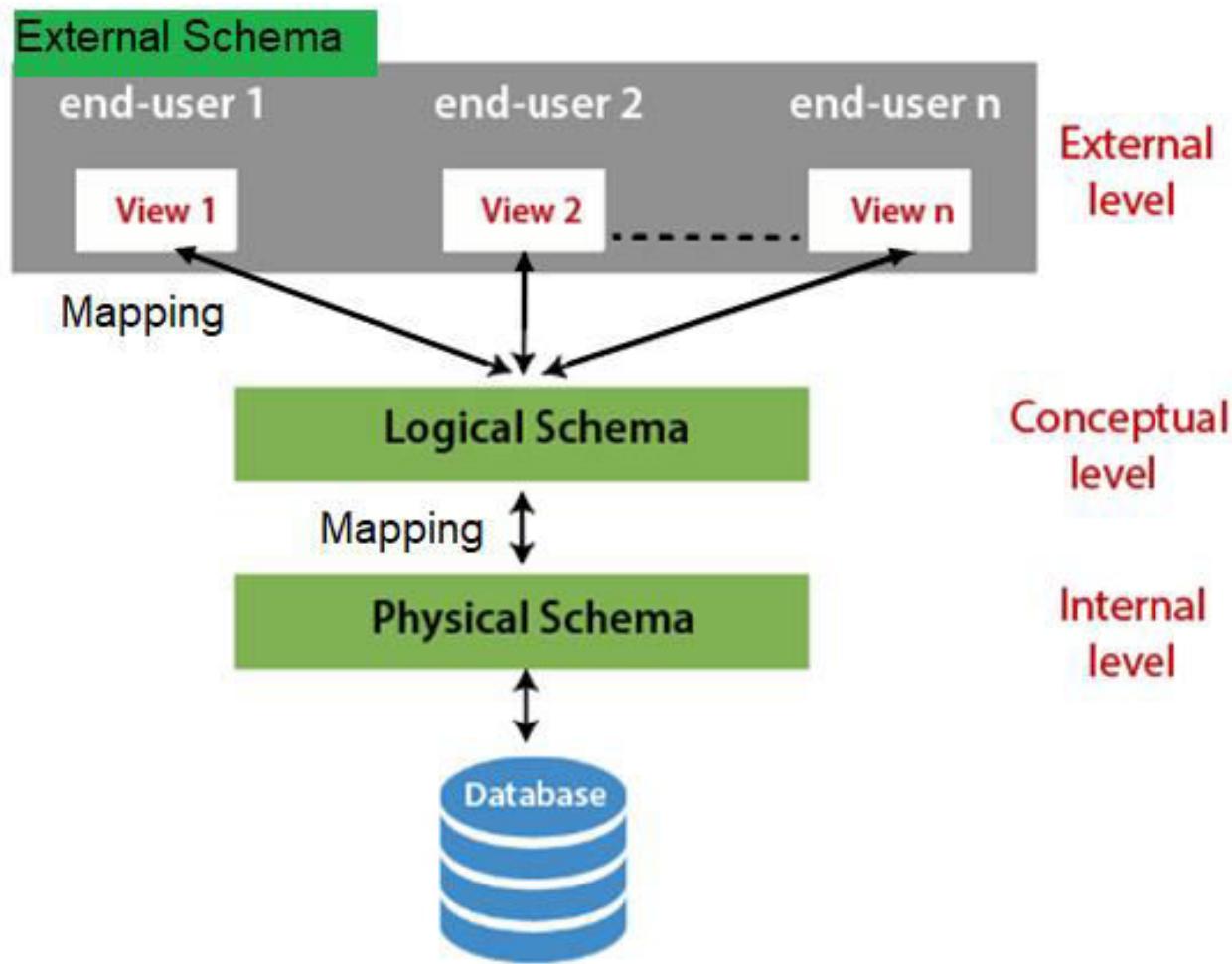


Figure: Three schema architecture

1. Physical/Internal schema

- It describes the DB design/physical structure at the physical level.
- It describes the complete details of data storage and access path for the database.
- The internal schema is a very low-level representation of the entire database.

2. Logical/ conceptual schema

- It describes the structure of the whole database for a community of users.
- It hides the details of physical storage from users.
- It describes entities, data types, relationships, user operations and constraints.
- It also manages security and integrity information.

3. External/view schema

- The external schema describes the segment of the database which is needed for a certain user group and hides the remaining details from the database from the specific user group
- It describes the end user interaction with database systems.

1.7 Database administrator (DBA) and Users

DBA:

- A person who has central control over the DB system.
- It is an IT professional who works on creating, maintaining, querying, and tuning the database of the organization.
- They are also responsible for maintaining data security and integrity.
- This role requires the professionals to have good knowledge and experience in the particular DBMS that the company uses.

- Based on the requirements of the company, there are various types of DBAs including:
 - **Administrative DBA**
 - They maintain and run the databases and servers of the organization.
 - They are mainly concerned with the security patches, replication, and backup of data.
 - **System DBA**
 - This role focuses on technical, rather than business issues.
 - The system DBA is knowledgeable in the arcane technical details of how the database is installed, configured, and modified.
 - **Development DBA**
 - They work on developing SQL queries and stored procedures to meet the requirements of the business.
 - They specialize in database development.
 - **Data Architect**
 - They design schemas, build data structures, table indexes, and relationships.
 - They are mainly responsible for building a structure that meets the business requirements in a specific area.
 - **Data Warehouse DBA**
 - They merge data from numerous data sources and store them in a data warehouse.

Functions of 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.

8. Security

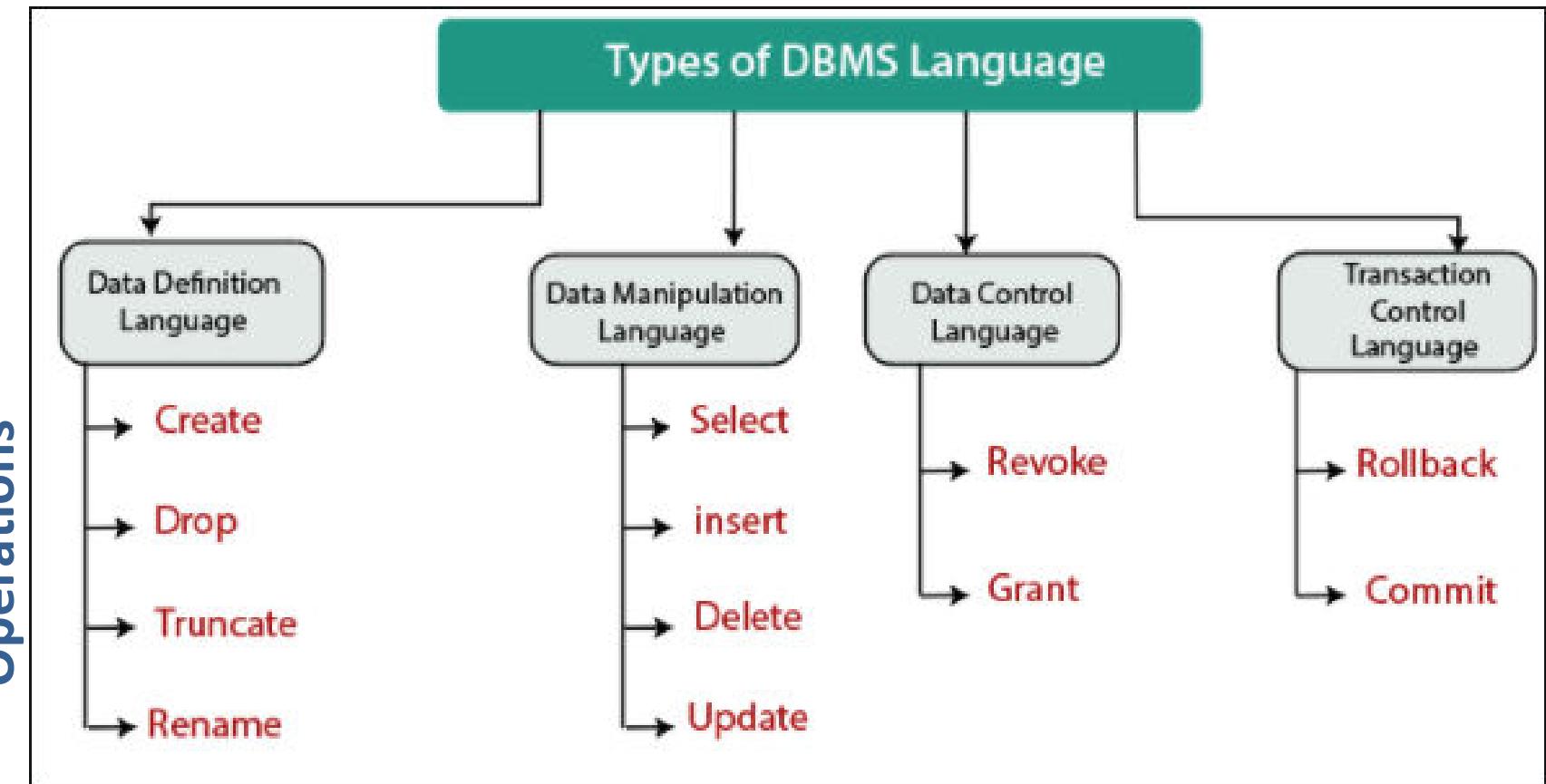
- A DBA needs to know potential weaknesses of the database software and the company's overall system and work to minimize risks.

Database User

- Database users are the persons who interact with the database and take the benefits of database.
- They are differentiated into different types based on the way they expect to interact with the system.
 - **Naive users:**
 - They are the unsophisticated users who interact with the system by using permanent applications that already exist.
 - Example: Online Library Management System, ATMs (Automated Teller Machine), etc.
 - **Application programmers:**
 - They are the computer professionals who interact with system through DML.
 - They write application programs.
 - **Sophisticated users:**
 - They interact with the system by writing SQL queries directly through the query processor without writing application programs.
 - **Specialized users:**
 - They are also sophisticated users who write specialized database applications that do not fit into the traditional data processing framework.
 - Example: Expert System, Knowledge Based System, etc.
 - **DBA**

1.8 DBMS Languages

- **Database Languages** are the set of statements, that are used to define and manipulate a database.
- It provides the tools to **implement** and **manipulate** a database.



1. Data Definition Language (DDL)

- DDL is used for specifying the database schema. It is used for creating tables, schema, indexes, constraints etc. in database.
- Operations/commands:
 - To create the database instance – **CREATE**
 - To alter the structure of database – **ALTER**
 - To drop database instances – **DROP**
 - To rename database instances – **RENAME**

2. Data Manipulation Language (DML)

- DML is used for accessing and manipulating data in a database.
- Operations/commands:
 - To read records from table(s) – **SELECT**
 - To insert record(s) into the table(s) – **INSERT**
 - Update the data in table(s) – **UPDATE**
 - Delete all the records from the table – **DELETE**

3. Data Control language (DCL)

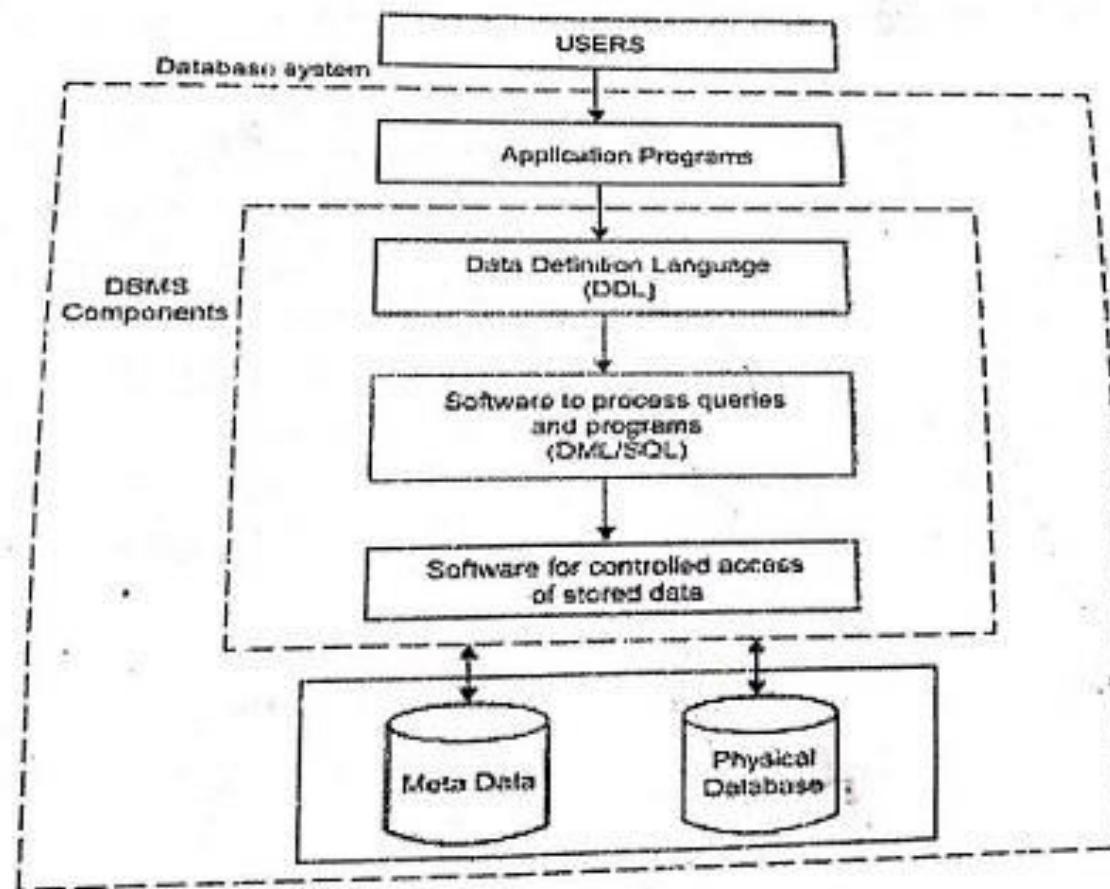
- DCL is used for granting and revoking user access on a database.
- Operations/commands:
 - To grant access to user – **GRANT**
 - To revoke access from user – **REVOKE**

4. Transaction Control Language(TCL)

- The changes in the database that we made using DML commands are either performed or rolled back using TCL.
- Operations/commands:
 - To persist the changes made by DML commands in database – **COMMIT**
 - To rollback the changes made to the database – **ROLLBACK**

Components of DBMS

A DBMS has three main components. These are Data Definition Language (DDL), Data Manipulation Language and Query Facilities (DML/SQL) and software for controlled access of Database as shown in figure and are defined as follows.



Data Definition Language (DDL)

It allows the users to define the database, specify the data types, data structures and the constraints on the data to be stored in the database.

Data Manipulation Language (DML) and Query Language

DML allows users to insert, update, delete and retrieve data from the database. SQL provides general query facility.

Software for Controlled Access of Database

This software provides the facility of controlled access of the database by the users, concurrency control to allow shared access of the database and recovery control system so restore the database in case of hardware or software failure.

Subject: DBMS

CHAPTER-2
Data Models

By: Kalpana Karki

Contents

Data Models

1. Conceptual, Logical and Physical model
2. Hierarchical, Network and Relational Data Models
3. Object-based Model, Entity Relationship Model(ER Model)
4. Components of ER diagram
5. Role of ER diagram
6. Entity Relationship diagram Methodology
7. Converting ER model into relations.

Data Models

- Underlying structure of the database is called as data model.
- It defines how the logical structure of a database is modeled.
- It is a collection of conceptual tools for describing data, data relationships, data semantics, and consistency constraints.
- It defines how data is connected to each other and how they are processed and stored inside the system.
- It consists of set of rules and standards that define how the data is organized in database.
- Data Models are fundamental entities to introduce abstraction in a DBMS.

Types of data model

1. Types on the basis of abstraction level (According to ANSI in 1975)
 - i. Conceptual data model
 - ii. Logical data model
 - iii. Physical data model
2. Other data models (on the basis of structure and used concept)
 - a) Hierarchical data model
 - b) Network data model
 - c) Relational data model
 - d) Object-based data model
 - e) ER(Entity relationship) data model

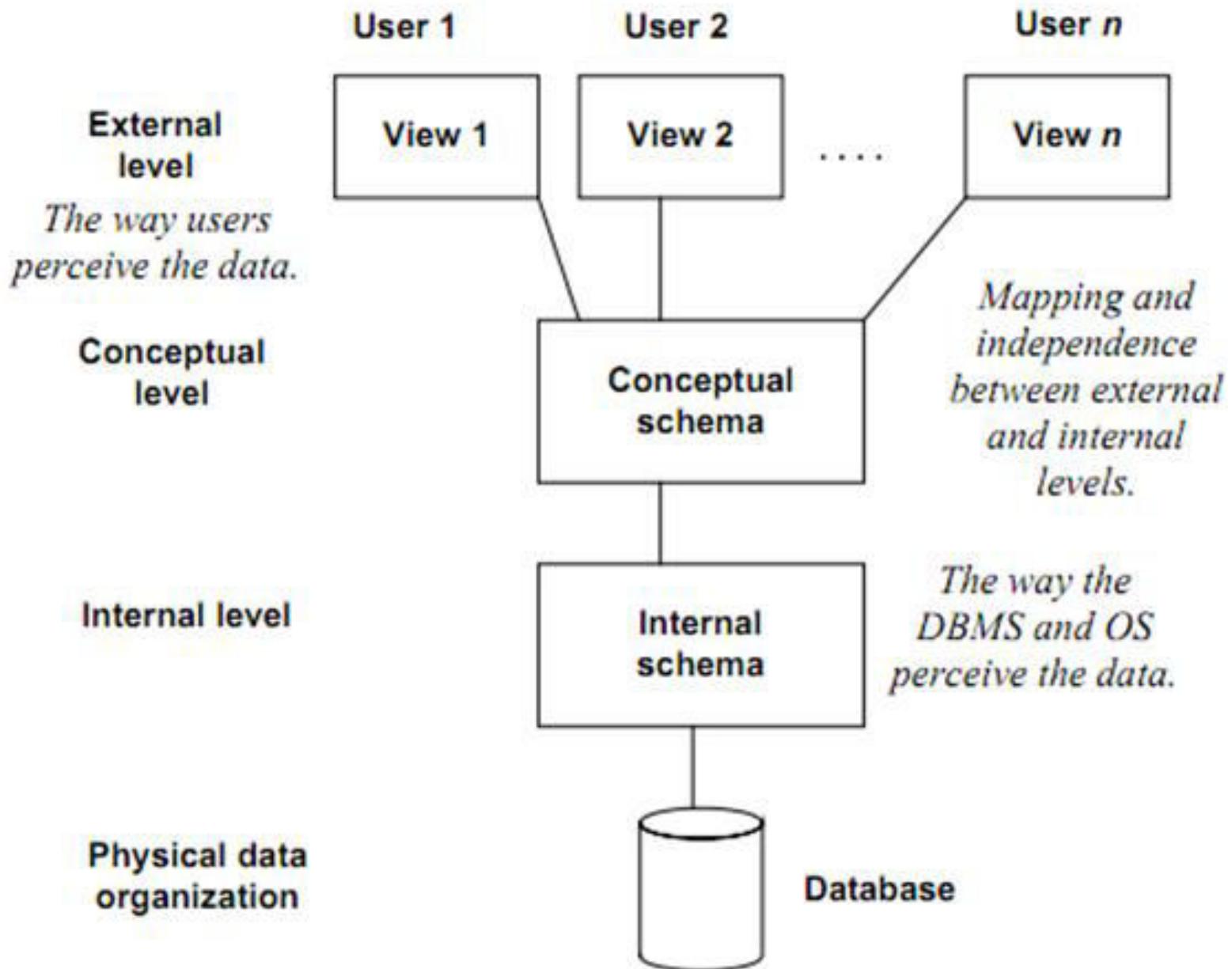
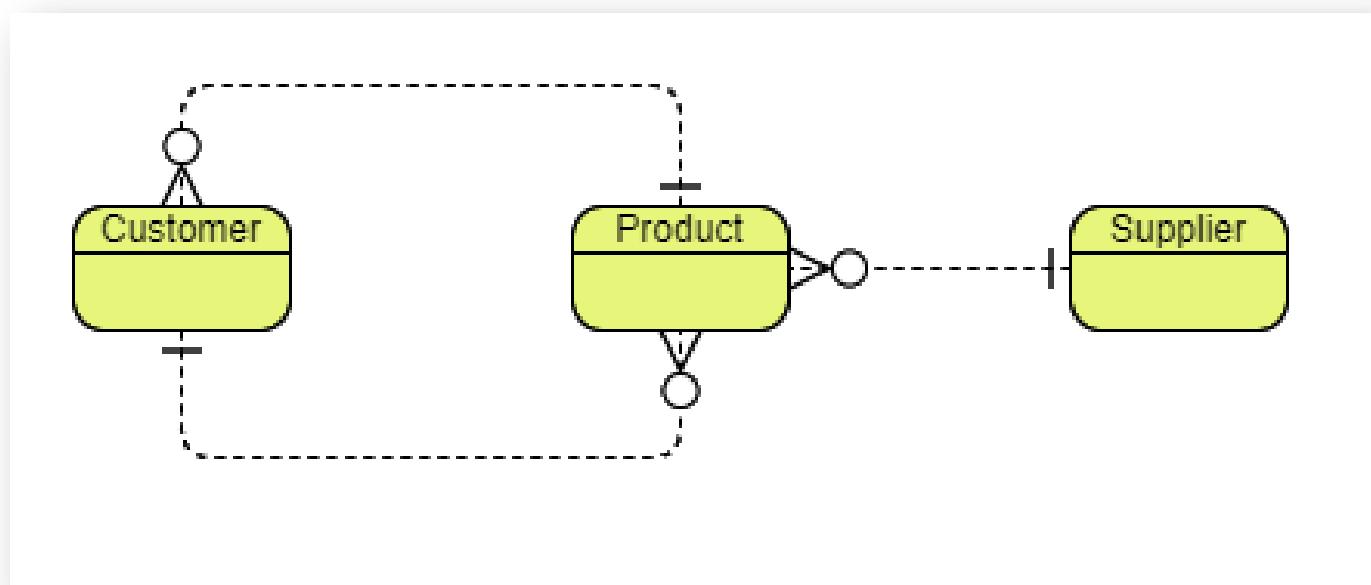


Fig.2.1 Three tier DBMS Architecture

i) Conceptual data model

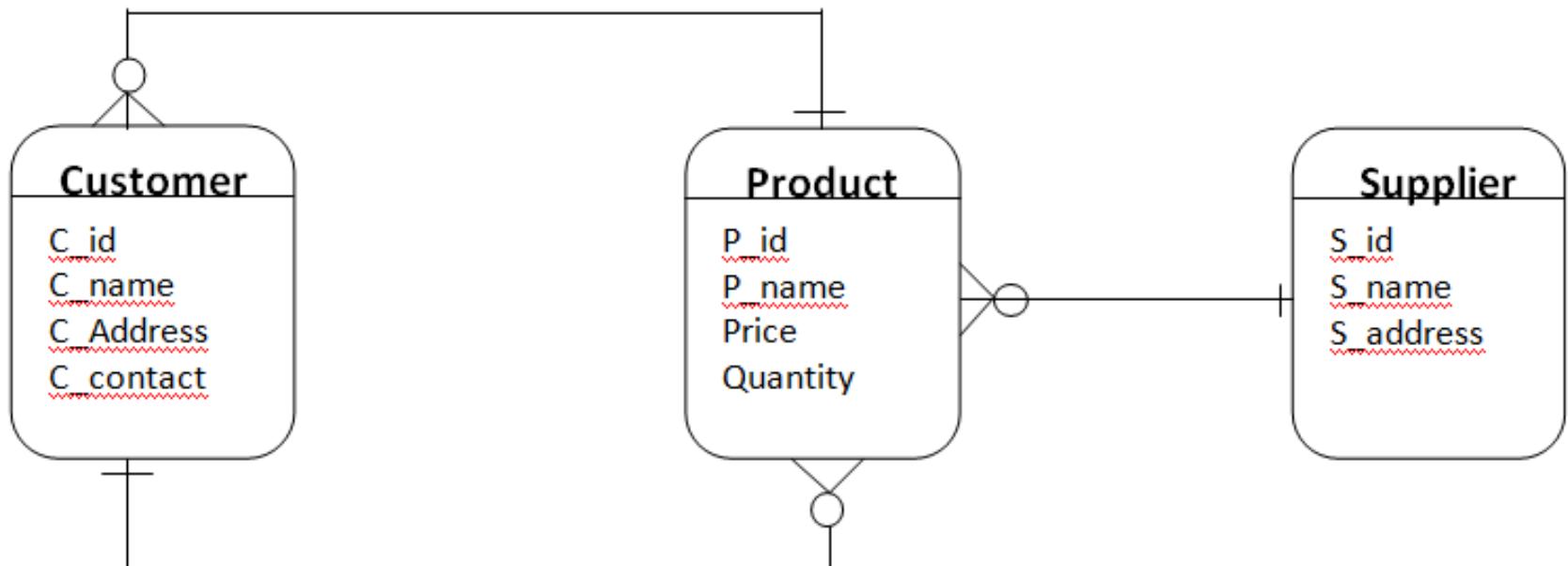
- It identifies the highest-level relationship between the different entities of the system.
- In an organization, it defines high-level, static business structures and concepts.
- Features:
 - It includes entity classes, their attributes and relationship among them.
 - No attribute is specified.
 - No primary key is specified



ii) Logical data model

- It describes the data in much detail as possible, without regard to how they will be physically implemented in DB.
- Features:
 - It includes entities and relationships among them.
 - All attributes for each entity is specified.
 - Primary key is specified.
 - Foreign key is specified.
 - Normalization occurs in this level.

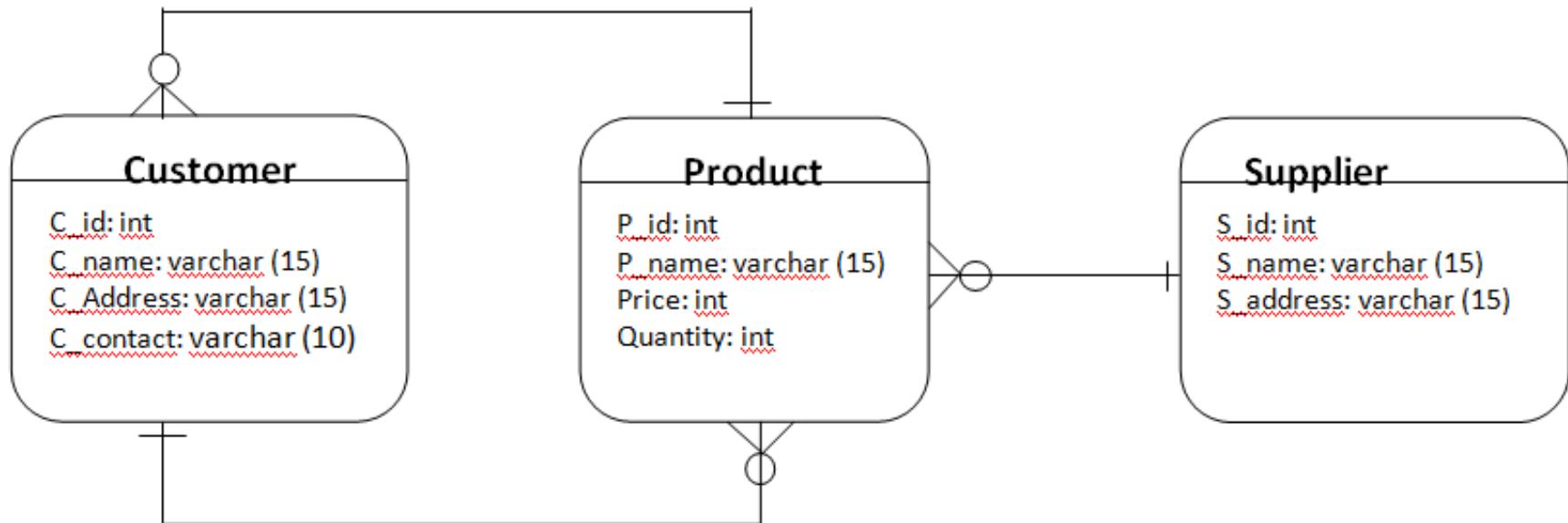
Example of Logical data model



iii) Physical data model

- It describes how the model will be built in DB.
- It shows all table structures including columns, column data type, keys, and relationship between tables.
- Features:
 - It specifies all tables and columns.
 - All keys are specified.
 - De-normalization may occurs at this level.

Example of Physical data model



a) Hierarchical data model

- It is a data model in which the data are organized into a tree-like structure.
- The data are stored as **records** which are connected to one another through **links**.
- A record is a collection of fields, with each field containing only one value.
- Each record is having one parent record and many children.
- The hierarchy starts from the **Root** data, and expands like a tree, adding child nodes to the parent nodes.
- **Characteristics:**
 - Each parent can have many children.
 - Each child has only one parent.

Example of hierarchical data model

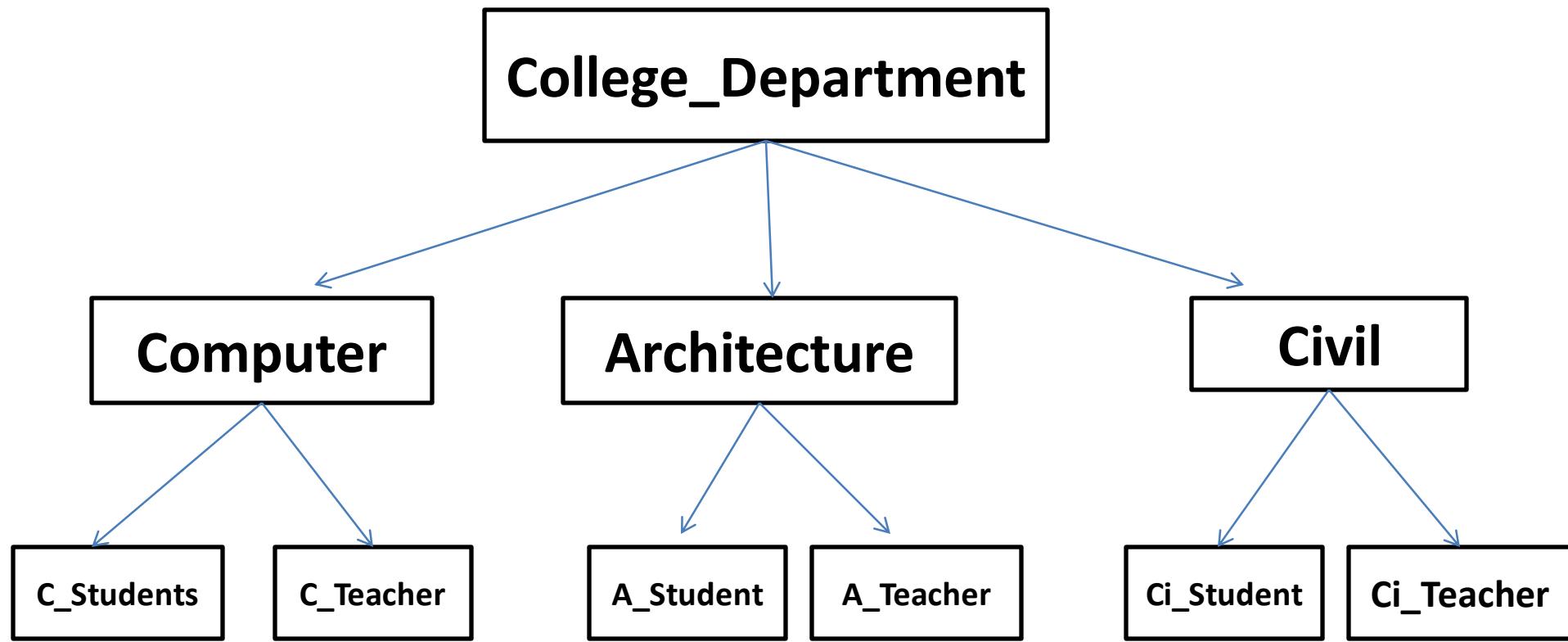


Figure: Hierarchical data model for college

Features of Hierarchical data model

- **One to many relationships**
- **Problem in Deletion**
- **Hierarchy of data**
- **Parent-child relationship**
- **Pointer:** Pointers are used for linking records that tell which is a parent and which child record is.
- **Predefined relationship:** All relations between root, parent and child nodes are predefined in the database schema.
- **Redundancy**

Advantages

- Promotes data sharing.
- It is conceptually simple due to the parent-child relationship.
- Database security is enforced.
- Efficient with 1: N relationships.
- A clear chain of command or authority.

Disadvantages

- Complex relationships are not supported.
- M: N relationship is not supported.
- Lack of standards.
- Poor flexibility
- Communication barriers
- Rigid structure

b) Network data model

- This is an extension of the Hierarchical model.
- In this model data is organized more like a graph, and are allowed to have more than one parent node.
- This data model is created to represent complex data relationships more effectively than the hierarchical model, to improve database performance, and to impose a database standard.
- This database model was used to map many-to-many data relationships.
- In this type of model, a child can be linked to multiple parents, a feature that was not supported by the hierarchical data model.

Basic Structure:

- Set
 - Relationship.
 - Composed of at least two record types.
- Owner
 - Equivalent to the hierarchical model's parent.
- Member
 - Equivalent to the hierarchical model's child.

Example of network data model

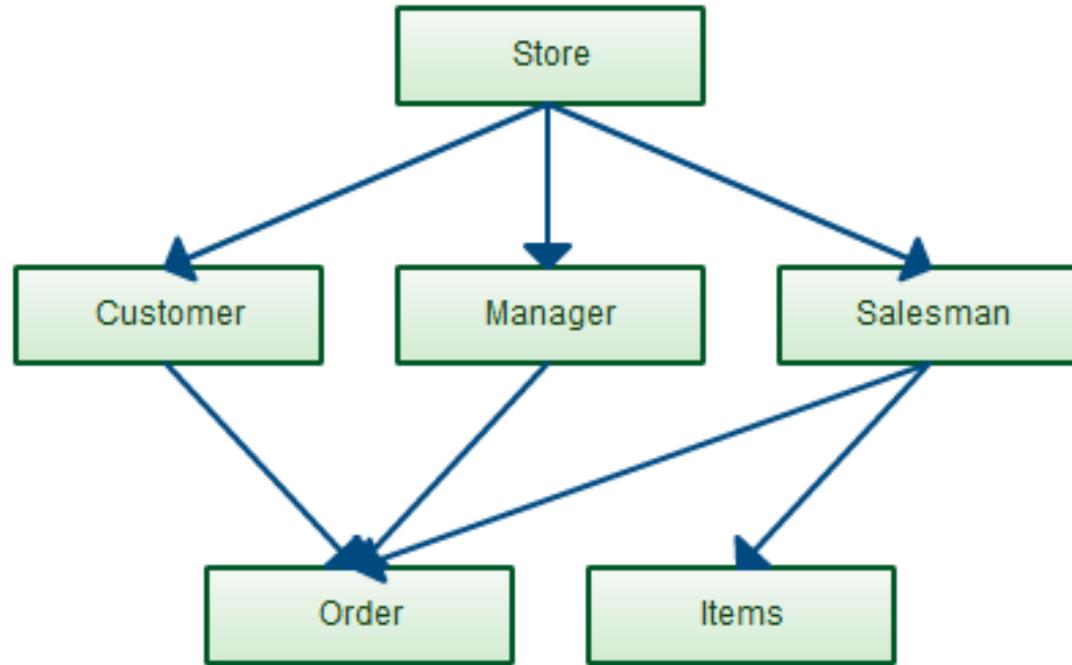


Figure: Network data model for store

Features/ characteristics of network data model

- The network model is better than a [hierarchical model](#).
- Supports many to many relationships.
- Many parents can have many children.
- Many children can have many parents.
- Entities are represented as a connected network with each other.
- Represented as a network and one child can have more than one parent. This model represents a complex structure.

- **Network Data Model Advantage**

- Conceptual simplicity
- Handles more relationship types
- Data access flexibility
- Promotes database integrity

- **Network Date Model Disadvantages**

- System complexity
- Detail structural knowledge is required
- Lack of structural independence

C) Relational data model

- In relational model, the data and relationships are represented by collection of inter-related tables.
- Each table is a group of column and rows, where column represents attribute of an entity and rows represents records.
- It represents the database as a collection of relations.
- It describes the data, relationship between that data, data semantic and constraints on the data in the relational database.
- A relation is table of values.
- Every row in the table represents a collection of related data values of particular person.
- Some terminologies:
 - Relation = table
 - Tuple= row
 - Attribute= column

Some popular Relational Database management systems (RDBMS) are:

- My-SQL
- MS-SQL/ MS-SQL Server
- MS_Access
- Oracle, etc.

Example of relational data model

Student

S_id	S_name	S_address	S_contact
1	Rita	Pokhara	9846789112
2	Rajesh	Kathmandu	9846933321

Key

Marks

Stu_id	DBMS	Math	English
1	76	67	82
2	85	78	54

Figure: Relational model for student

Advantages of Relational model:

- **Simplicity:** A Relational data model in DBMS is simpler than the hierarchical and network model.
- **Structural Independence:** The relational database is only concerned with data and not with a structure. This can improve the performance of the model.
- **Easy to use:** It is easy as tables consisting of rows and columns are quite natural and simple to understand
- **Query capability:** It makes possible for a high-level query language like [SQL](#) to avoid complex database navigation.
- **Data independence:** The Structure of Relational database can be changed without having to change any application.
- **Scalable:** Database could be enlarged easily to enhance as requirement.
- **Greater flexibility**
- **Easy to design**

Disadvantages of Relational model:

- Data anomalies (insertion, update, delete anomalies)
- Need of professional people having knowledge about relational model
- Need of additional software
- Complexity increases with increase in data in DB

d) Object-based data model

General Types:

- a) Object-oriented data model
- b) ER-data model

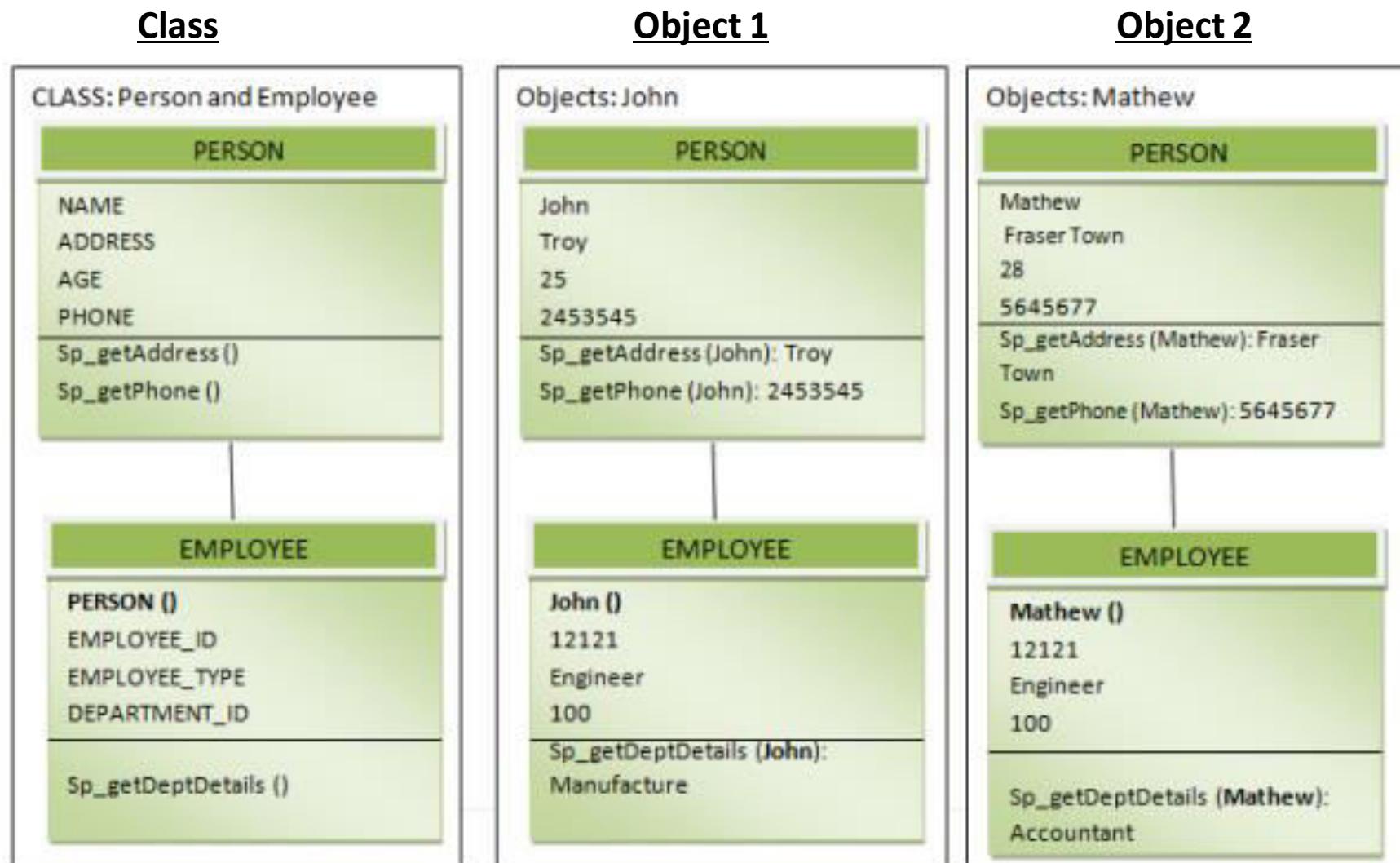
d) Object-based data model

- This data model represents real world objects.
- Increasingly complex real-world problems demonstrated a need for a data model that more closely represented the real world.
- It uses the concepts of object-oriented design.
- It considers each object in the world as objects and isolates it from each other.
- It groups its related functionalities together and allows inheriting its functionality to other related sub-groups.

Components/features of Object-based data model:

- Class
- Object
- Attributes
- Inheritance
- Abstraction
 - Encapsulation
 - Data hiding
- Polymorphism, etc.

Example of object-based data model



- **Advantages**

- Because of its inheritance property, we can re-use the attributes and functionalities.
 - It reduces the cost of maintaining the same data multiple times.
- Due to encapsulated message, there is no fear being class misused by other objects.
- If we need any new feature we can easily add new class inherited from parent class and adds new features.
 - Hence it reduces the overhead and maintenance costs.
- Because of the above feature, it becomes **more flexible** in the case of any changes.
- Codes are re-used because of inheritance.
- Since each class binds its attributes and its functionality, it is same as representing the real world object. We can see each object as a real entity. Hence it is more **understandable**.

- **Disadvantages**

- It is not widely developed and complete to use it in the database systems. Hence it is not accepted by the users.
- Lack of OODM standards
- It is an approach for solving the requirement. It is not a technology. Hence it fails to put it in the database management systems.

e) ER(Entity relationship) data model

- **ER Model** stands for Entity Relationship Model is a high-level conceptual data model diagram.
- It describes interrelated things of interest in a specific domain of knowledge.
- A basic ER model is composed of entity types and specifies relationships that can exist between entities.
- ER model helps to systematically analyze data requirements to produce a well-designed database.
- The ER Model represents real-world entities and the relationships between them.
- Creating an ER Model in DBMS is considered as a best practice before implementing your database.

- ER modeling is based on following concepts:
 - **Entities:** It is defined as tables that hold specific information (data).
 - **Attributes:** Properties/descriptions of entities.
 - **Relationships:** It is defined as the associations or interactions between entities.

Advantages of ER-data model:

- Conceptually it is very simple
- Better visual representation
- Effective communication tool
- Highly integrated with relational model
- Easy conversion to any data model

Disadvantages of ER-data model:

- Limited constraints and specification
- Loss of information content
- Limited relationship representation
- No representation of data manipulation

2.4 ER-Diagram

- An ER diagram shows the relationship among entity sets. An entity set is a group of similar entities and these entities can have attributes.
- It is a visual representation of different entities within a system and how they relate to each other.

ER Diagram consists of the following components:

i. Entity:

- An entity can be a person, place, event, or object that is relevant to a given system.
- For example: a school system may include following entities students, teachers, subjects, etc.

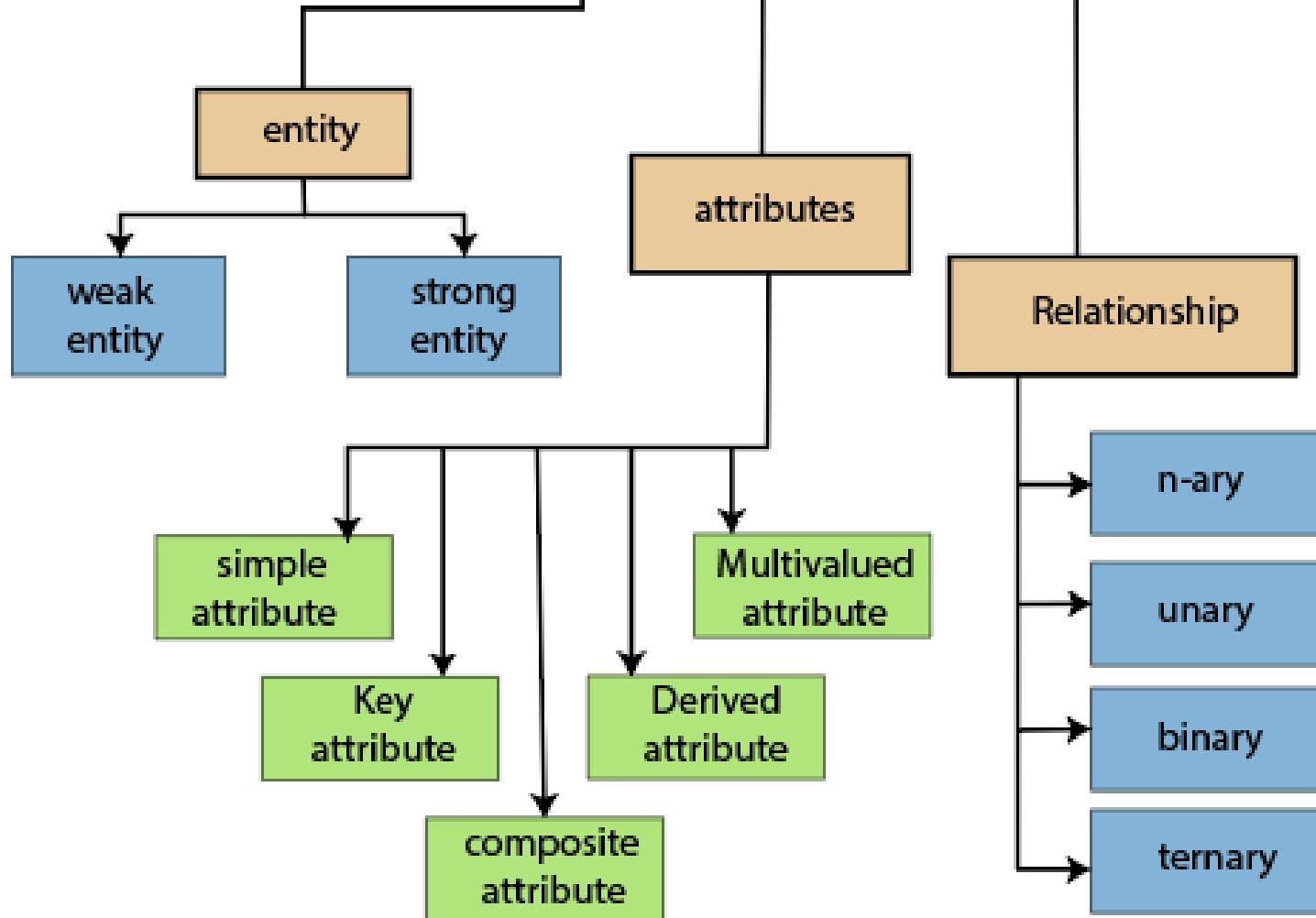
ii. Attributes:

- An attribute is a property or characteristic of an entity.
- Example: attributes of entity student are: S-id, S-name, S-address, etc

iii. Relationships:

- A relationship describes how entities interact.

Component of ER Diagram



Roles of ER-diagram

- It allows us to sketch DB schema designs.
 - It includes some constraints.
- It helps us to understand the system conceptually.
- It is an easy to use graphical tool for modeling data
- It is a GUI representation of the logical structure of a Database
- It helps you to identifies the entities which exist in a system and the relationships between those entities.
- It also helps to design relational DB.

ER-diagram methodology

- **Rectangles:** This Entity Relationship Diagram symbol represents entity types
- **Ellipses :** Symbol represent attributes
- **Diamonds:** This symbol represents relationship types
- **Lines:** It links attributes to entity types and entity types with other relationship types
- **Primary key:** attributes are underlined
- **Double Ellipses:** Represent multi-valued attributes



Figure: ER-diagram Symbols

Components of ERD:

- Entity
- Attributes
- Relationship

- **Cardinality** defines the possible number of occurrences in one entity which is associated with the number of occurrences in another.

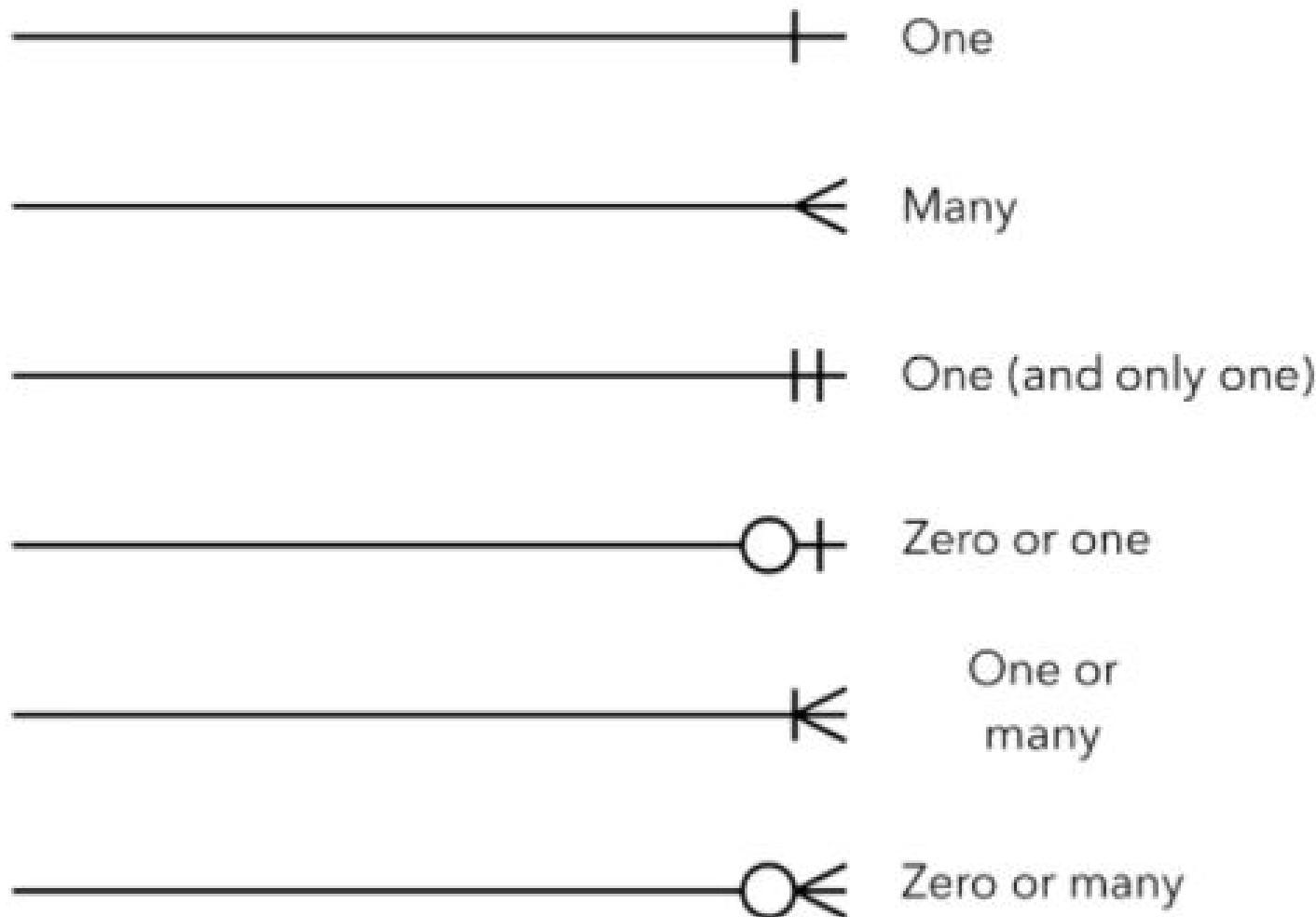


Figure: Types of cardinality

Example of ERD

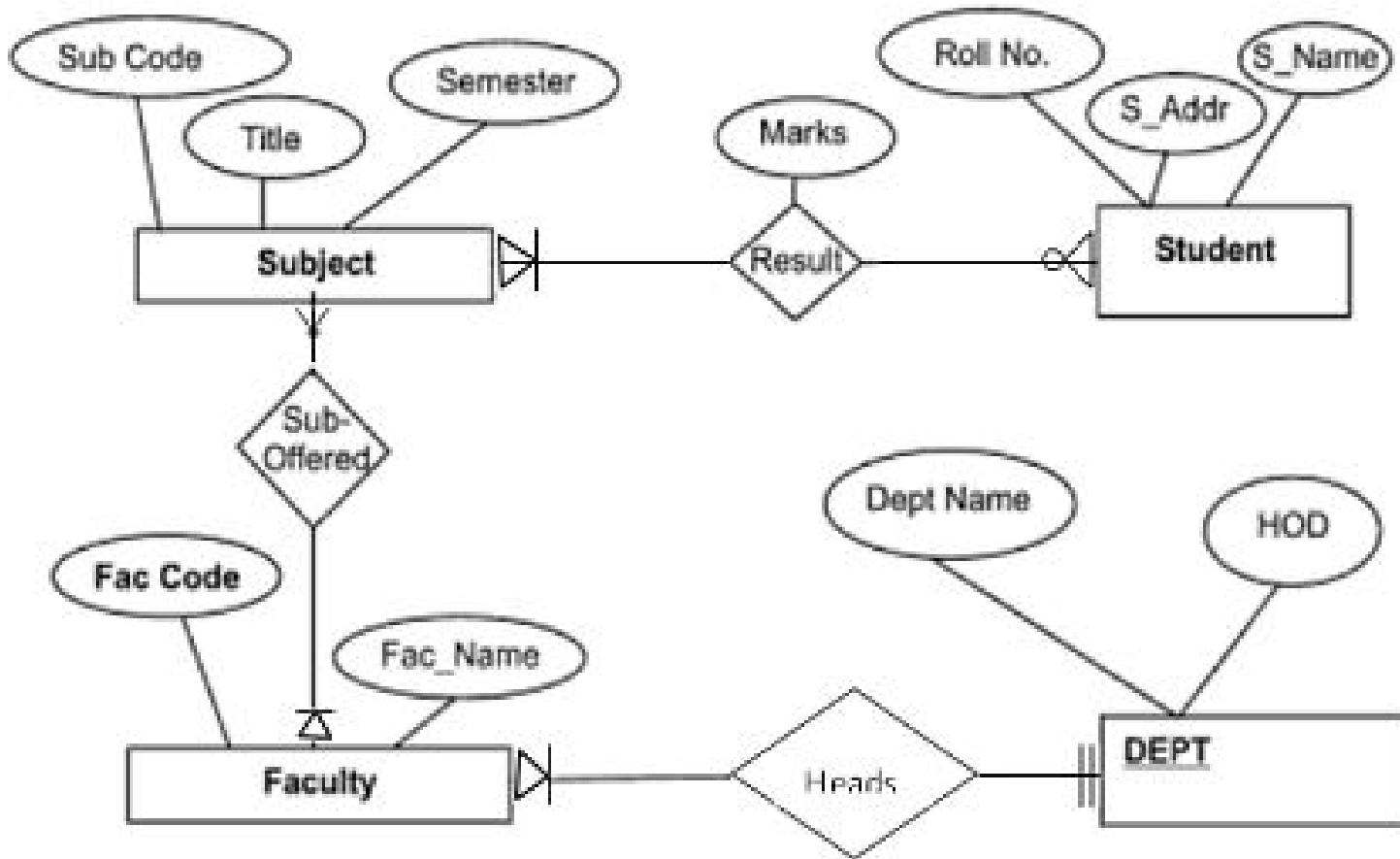


Figure: ERD for College management system

Extended ER-features

1. Generalization:

- Generalization is the process of extracting common properties from a set of entities and create a generalized entity from it.
- It is a bottom-up approach in which two or more entities can be generalized to a higher level entity if they have some attributes in common.

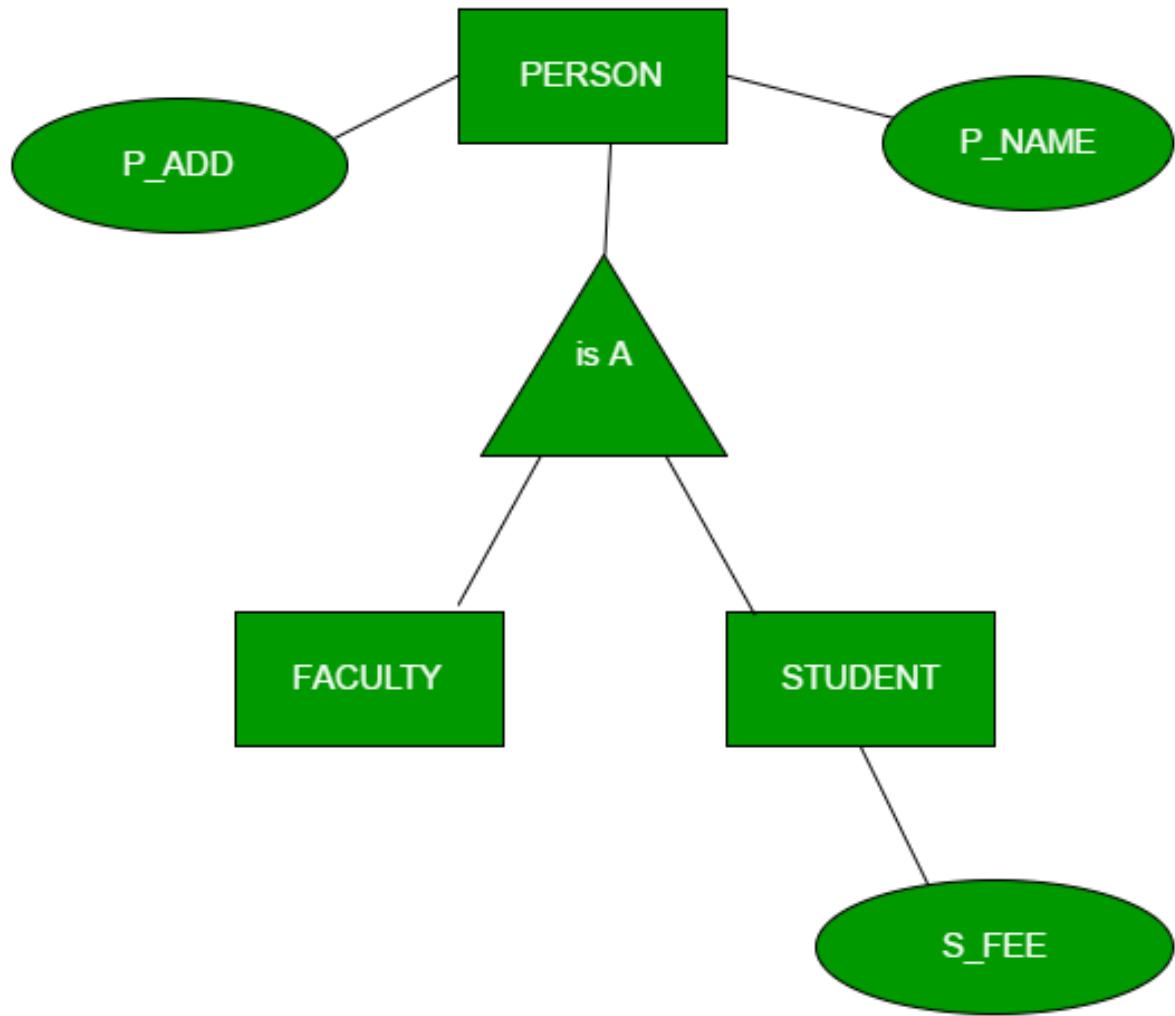


Figure: Generalization example

2. Specialization:

- In specialization, an entity is divided into sub-entities based on their characteristics.
- It is a top-down approach where higher level entity is specialized into two or more lower level entities.

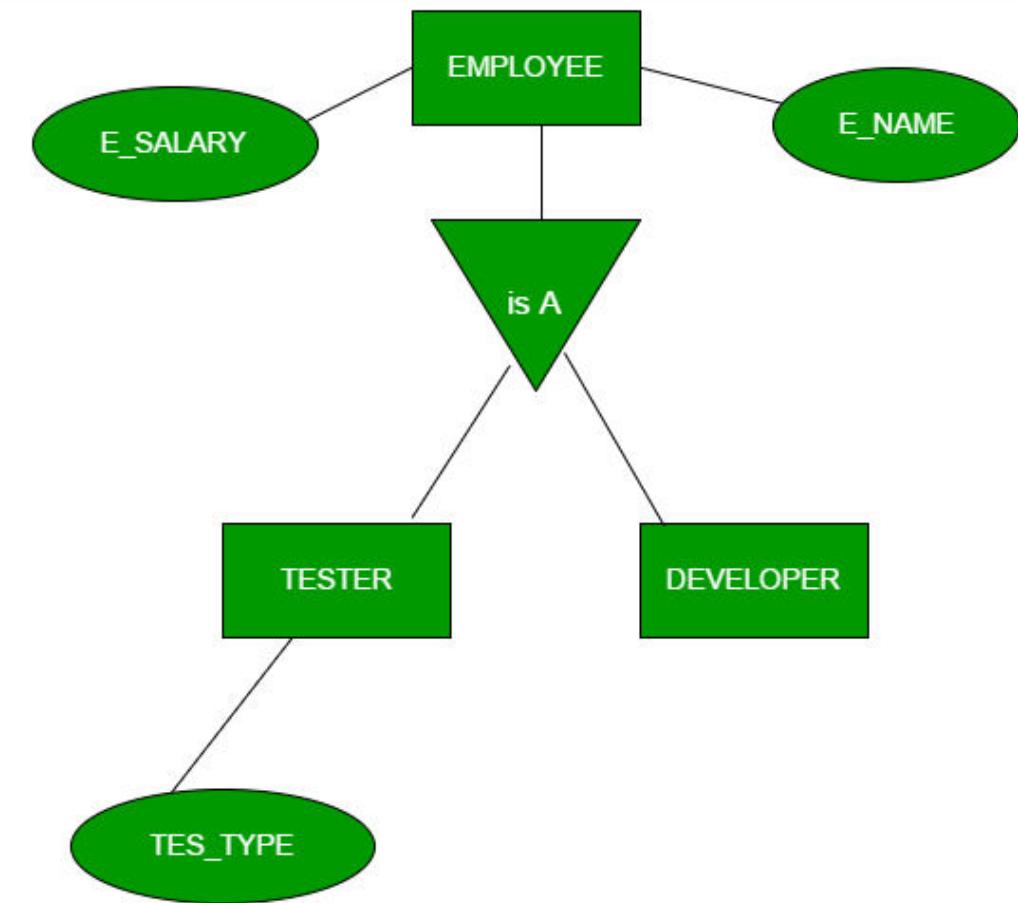
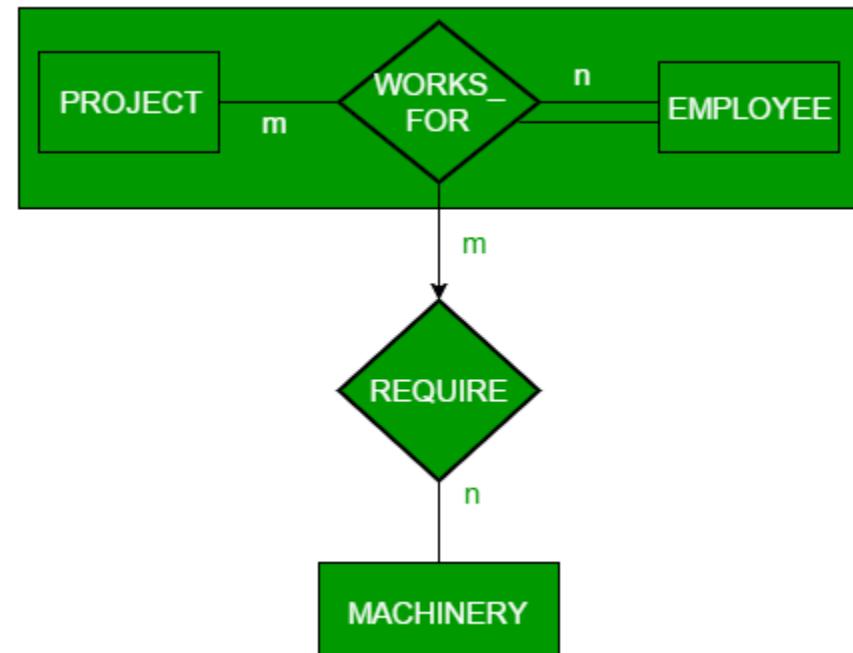


Figure: Specialization example

3. Aggregation:

- An ER diagram is not capable of representing relationship between an entity and a relationship which may be required in some scenarios.
- In those cases, a relationship with its corresponding entities is aggregated into a higher level entity.



Lines: link the attributes to entity sets and entity sets to relationship sets.

Doubles lines: indicate the total participation of an entity in a relationship set.

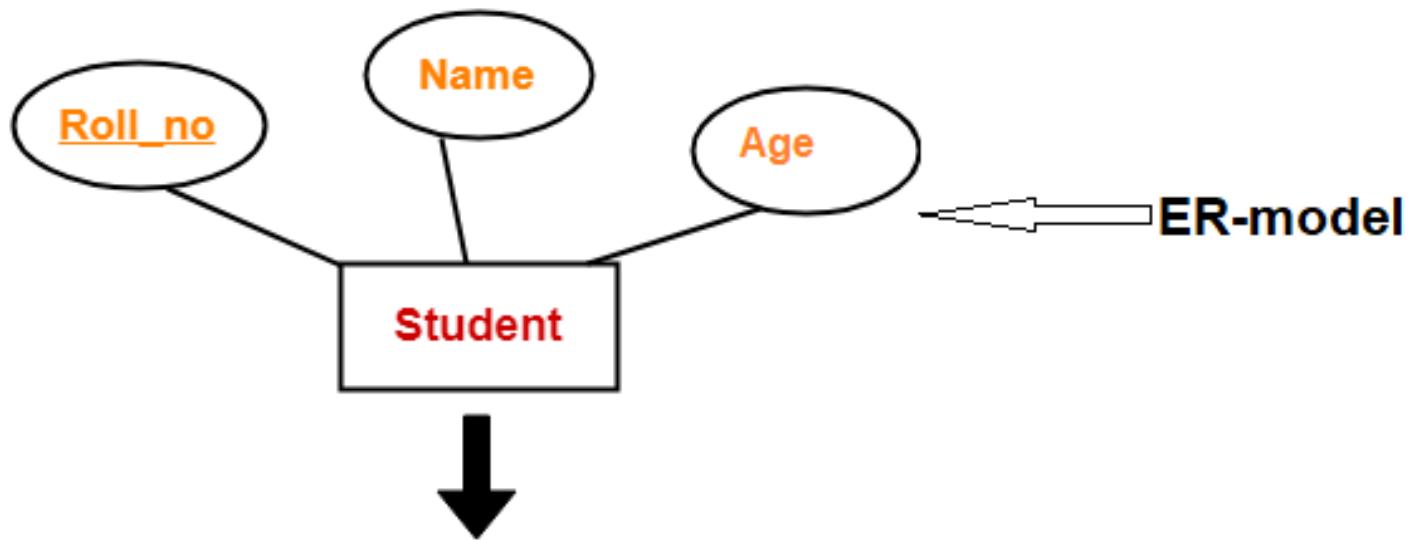
Figure: Aggregation example

2.7 Converting ER model into Relations

- Relation:
 - A relation is a table that holds the data that are interested in.
 - It is two-dimensional and has rows & columns.
- Rules for converting ER-model to Relational model:

Rules for converting ER-model to Relational model:

<u>ER-model</u>	<u>Relational model</u>
1. Entity	1. Relation (table)
2. Simple attributes	2. Columns (fields)
3. Key attributes	3. Primary key
4. Value set	4. Domain
5. 1:1 or 1:N relationship	5. Foreign key
6. M:N relationship	6. A relation and two foreign keys



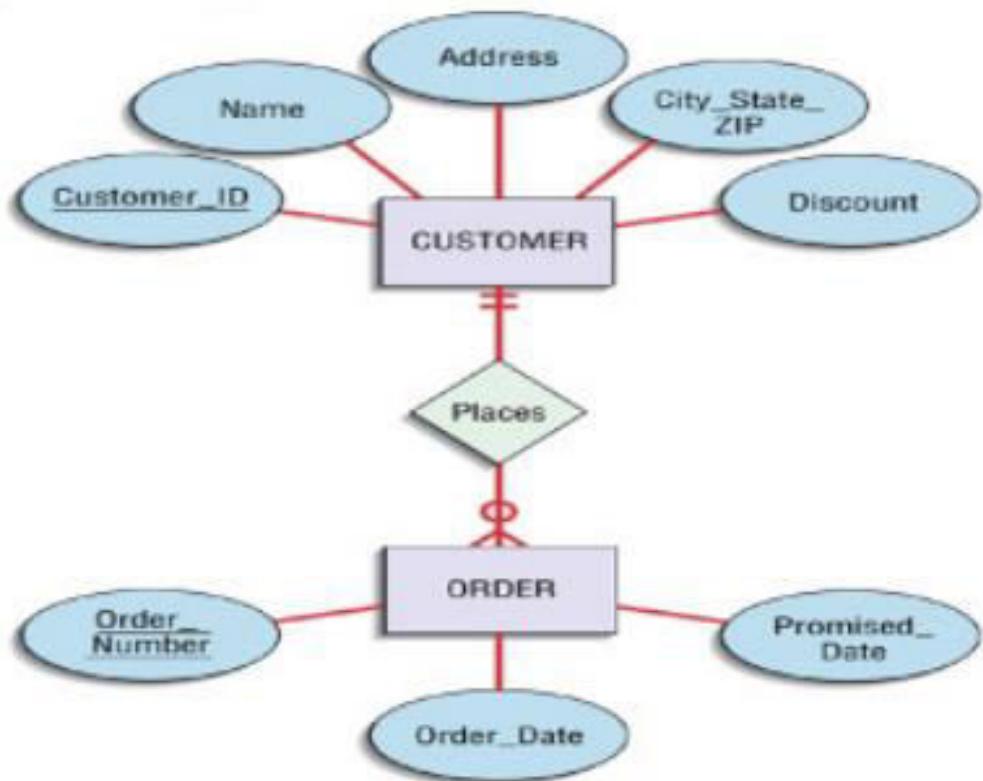
Student

<u>Roll_no</u>	Name	Age
1	Ram	20
2	Rita	22

← Relational model

OR Schema : Student (Roll_no , Name , Age)

Figure: Example of ER-model to Relational-model



CUSTOMER

<u>Customer_ID</u>	Name	Address	City_State_ZIP	Discount
1273	Contemporary Designs	123 Oak St.	Austin, TX 28384	5%
6390	Casual Corner	18 Hoosier Dr.	Bloomington, IN 45821	3%

ORDER

<u>Order_Number</u>	Order_Date	Promised_Date	Customer_ID
57194	3/15/0X	3/28/0X	6390
63725	3/17/0X	4/01/0X	1273
80149	3/14/0X	3/24/0X	6390

Assignment

1. Design ERD for student management system and convert to the Relational model.
2. Design ERD for Bank management system and convert to the Relational model.

Subject: DBMS

Chapter-3 Relational Model

By: Er. Kalpana Karki

Contents

3. Relational Model

- 3.1 Definitions and terminology
- 3.2 Structure of Relational databases
- 3.3 Relational Algebra and calculus
- 3.4 Pitfalls of relational Model

Relational Model

3.1 Definitions and terminology

- **Relational Model (RM)** represents the database as a collection of relations.
- A relational model database is defined as a database that allows you to group its data items into one or more independent tables that can be related to one another by using fields common to each related table.
- The **relational model** for database management is an approach to managing data using a structure and language, where all data is represented in terms of tuples, grouped into relations.
- A database organized in terms of the relational model is a relational database.
- **Some popular Relational Database management systems are:**
 - DB2 and Informix Dynamic Server - IBM
 - Oracle and RDB – Oracle
 - SQL Server and Access - Microsoft

Basic terminology used in relational model

- **Tuples of a Relation:** Each row in data is actually a tuple.
- **Cardinality of a Relation:** The number of tuples in a relation determines its cardinality.
- **Attribute:** Each column in the tuple is called attribute.
- **Degree of relation:** The number of attributes in tuple determines its degree.
- **Domain:** Specifies the type of data represented by the attribute or a domain is all kind of values that an attribute can validly contain.
 - Domains are often confused by data types; data type is physical concept while domain is logical.
 - Number is data type and age is domain.
- **Field:** It stores actual value of an attribute.
- **Keys of a Relation:** it is a set of one or more columns whose combined values are unique. Some different types of keys are:
 - Primary key
 - Foreign Key
- **Relation:** Table with rows and column.
- **Extension :** Set of tuples (records) at any given instance.
- **Intension:** Permanent part of relation.

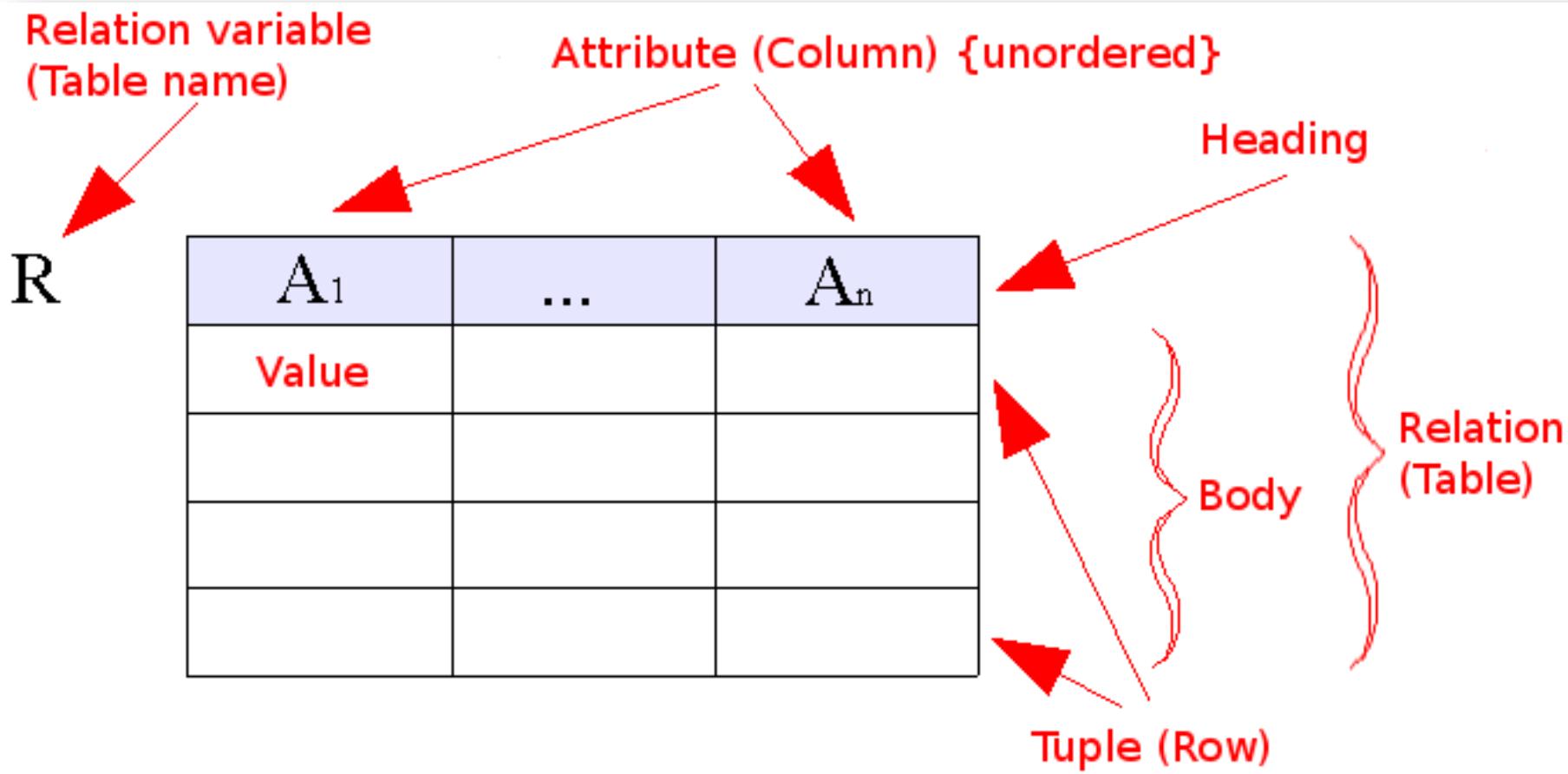


Figure: Relational Model Concept

- Constraints on the Relational database management system is mostly divided into three main categories are:
 - Domain Constraints
 - Key Constraints
 - Referential Integrity Constraints
- There are more constraints in Relational model.

3.2 Structure of Relational databases

- It provides a basis of high-level data language, which will yield maximal independence between programs on the one hand and machine representation on the other.
- In other words, the relational model consists of the following:
 - Data independence from hardware and storage implementation
 - Automatic navigation or nonprocedural language for accessing data at a high-level.
- A **relational database** basically consists of a number of **tables**, each of which is called a relation instance, or simply **relation**.
- Each table contains a number of **rows** and a number of **columns**.
- Each row is called a **tuple**.
- Each column is called an **attribute**.

Customer		
customer_name	customer_street	customer_city
Jones	Main	Harrison
Smith	North	Rye
Curry	North	Rye
Lindsay	Park	Pittsfield

Diagram illustrating the structure of a relational database table:

- The table is labeled "Customer".
- The columns are labeled "customer_name", "customer_street", and "customer_city".
- The rows are labeled "Jones", "Smith", "Curry", and "Lindsay".
- An arrow points from the label "attributes (or columns)" to the column headers.
- An arrow points from the label "tuples (or rows)" to the row data.

3.3 Relational Algebra and calculus

Relational Algebra

A query language is a language in which a user requests information from the database. These languages are usually on a level higher than that of a standard programming languages. Query languages can be categorized as :

i) Procedural language

In procedural language, the user instructs the system to perform a sequence of operations on the database to compute the desired result.

ii) Nonprocedural language

In a nonprocedural language, the user describes the desired information without giving a specific procedure for obtaining that information.

The **relational algebra** is a procedural query language. It consists of a set of operations that take one or two relations as input and produce a new relation as their result.

➤ Formal definition of the relational algebra :

A basic expression in the relational algebra consists of either one of the following :

- A relation in the database
- A constant relation

A constant relation is written by listing its tuples within { }, for example
{(1, 'Ram', 'Kathmandu'), (2, 'Hari', 'Pokhara')}

- Relational algebra is a procedural query language, which takes instances of relations as input and yields instances of relations as output.
 - It uses operators to perform queries. An operator can be either **unary** or **binary**.
 - They accept relations as their input and yield relations as their output.
 - Relational algebra is performed recursively on a relation and intermediate results are also considered relations.
 - The fundamental operations of relational algebra are as follows –
 1. Select
 2. Project
 3. Rename
 4. Union
 5. Set-intersection
 6. Set-difference
 7. Division
 8. Cartesian product
 9. Join
 10. Assignment
 11. Deletion
 12. Insertion
 13. Update
-
- Unary Operations
- Binary Operations

3.4 Pitfalls of relational Model

Advantages of Relational Model:

- **Conceptual Simplicity:** simple than other data models.
- **Structural Independence:** changes in the structure do not affect the data access.
- **Design Implementation:** It achieves both data independence and structural independence.
- **Ad hoc query capability:** the presence of very powerful, flexible and easy to use capability.
- **Scalable:** Regarding a number of records, or rows, and the number of fields, a database should be enlarged to enhance its usability.
- **Query capability:** It makes possible for a high-level query language like [SQL](#) to avoid complex database navigation.

Pitfalls/disadvantages of relational Model

- **Hardware overheads:** relational database systems hide the implementation complexities and the physical data storage details from the user. For doing this, the relational database system need more powerful hardware computers and data storage devices.
- **Ease of design can lead to bad design:** The user needs not to know the complexities of the data storage. This ease of design and use can lead to the development and implementation of the very poorly designed database management system.
- **Relational databases can sometimes become complex:** as the amount of data grows, and the relations between pieces of data become more complicated.

Subject: DBMS

CHAPTER-5

Relational Database Design and Normalization

By: Kalpana Karki

Contents

5. Relational Database Design and Normalization

- 5.1 Integrity Constraints: Domain constraint, Entity Integrity, Referential Integrity
- 5.2 Functional dependency
- 5.3 Inference rules for functional dependency
- 5.4 Decomposition of relation
- 5.5 Closure Set of Functional Dependency and attributes
- 5.6 Dependency preservation
- 5.7 Normalization, Role of Normalization
- 5.8 Normalization(1 F, 2NF, 3NF)
- 5.9 BCNF and 3NF
- 5.10 Multi -valued dependency and 4NF
- 5.11 Join dependency and 5N F

Keys in DBMS

- It is an attribute or set of attributes which helps you to identify a row(tuple) in a relation(table).
- A single attribute or a combination of two or more attributes of an entity set that is used to identify one or more instances (rows) of the set (table) is called as key.
- They allow you to find the relation between two tables.
- Keys help you uniquely identify a row in a table by a combination of one or more columns in that table.
- Key is also helpful for finding unique record or row from more than a table.

Types of Key

1. Super key:

- A super key is a group of single or multiple keys which identifies rows in a table.
- A Super key may have additional attributes that are not needed for unique identification.

Example:

EmpSSN	EmpNum	Empname
9812345098	AB05	Shown
9876512345	AB06	Roslyn
199937890	AB07	James

In the above-given example, EmpSSN and EmpNum name are superkeys.

Possible super key set: {EmpSSN}, {EmpSSN, EmpNum}

2. Primary Key :

- Primary key is one of the candidate keys that uniquely identify each row in the relation.
- A primary key cannot contain null values.
- The Primary Key can't be a duplicate meaning the same value can't appear more than once in the table.
- A table cannot have more than one primary key.

Example:

In the following example, **StuID** is primary key.

StuID	Roll No	First Name	Last Name	Email
1	11	Tom	Price	abc@gmail.com
2	12	Nick	Wright	xyz@gmail.com
3	13	Dana	Natan	mno@yahoo.com

3. Foreign Key :

- An Attribute or combination, in one table whose values must either match the primary Key (PK) in another table or be null.
- Foreign keys are the column of the table which is used to point to the primary key of another table.
- It is a column that creates a relationship between two tables.

Example:

DeptCode	DeptName
001	Science
002	English
005	Computer

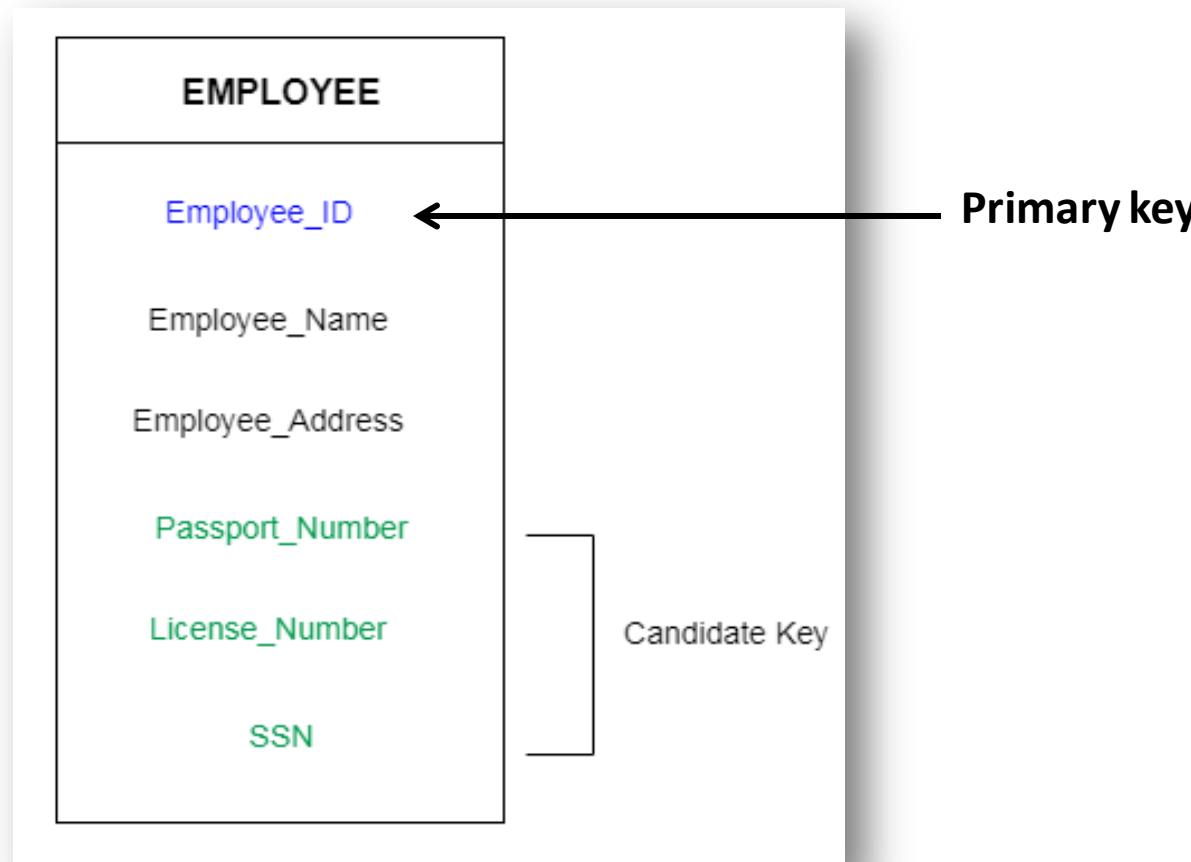
← Parent Table

Teacher ID	DeptCode	Fname	Lname
B002	002	David	Warner
B017	002	Sara	Joseph
B009	001	Mike	Brunton

← Child Table

4. Candidate Key :

- A candidate key of an entity set is a minimal super key that uniquely identifies each row in the relation.
- A candidate key is an attribute or set of an attribute which can uniquely identify a tuple.
- The remaining attributes except for primary key are considered as a candidate key. The candidate keys are as strong as the primary key.



5. Composite key:

- A primary key that consists of more than one attribute that uniquely identify any record in a table is called **Composite key**.
- But the attributes which together form the **Composite key** are not a key independently or individually.
- Example:

Composite Key



student_id	subject_id	marks	exam_name

6. Secondary Key/Alternative key:

- The candidate key which are not selected as primary key are known as secondary keys or alternative keys.
- An attribute or combination, used strictly for data retrieval purposes.

Assignment

1. Compare super key and composite key with example.

5.1 Integrity Constraints: Domain Constraint, Entity Integrity, Referential Integrity

- **Integrity Constraints**

- An *integrity constraint* is a condition specified on a database schema and restricts the data that can be stored in an instance of the database.
- If a database instance satisfies all the integrity constraints specified on the database schema, it is a legal instance.
- A DBMS enforces integrity constraints, in that it permits only legal instances to be stored in the database.
- Three types of integrity constraints are an inherent part of the relational data model:
 - i. Entity integrity
 - ii. Referential integrity
 - iii. Domain integrity.

i. Entity Integrity

- It concerns the concept of a [primary key](#).
- Entity integrity is an integrity rule which states that, “every table must have a primary key and that the column or columns chosen to be the primary key should be unique and not null”.

OR,

The entity integrity constraint states that primary key value can't be null.

- This is because the primary key value is used to identify individual rows in relation and if the primary key has a null value, then we can't identify those rows.
- A table can contain a null value other than the primary key field.
- **NULL** represents the absence of a value.

- Example1:
 - STUDENT(R_no,Name,Add,Age,Sem)
 - In this relation if R_no of the student is the primary key of this table then we should not be able to insert a tuple into this STUDENT relation with a null for R_no.
- Example 2:

Customer

Primary Key	ID	Customer_Name	Age
	1	Andrew	18
	2	Angel	20
		Angel	20

This value cannot be NULL as we will not be able to identify customers uniquely

ii. Referential integrity

- It concerns with the concept of a foreign key.
- A referential integrity constraint is specified between two tables.
- It states that, “if a foreign key exist in a relation, either the foreign key values must match candidate key value of base relation (primary key relation) or foreign key value must be wholly NULL.”

OR,

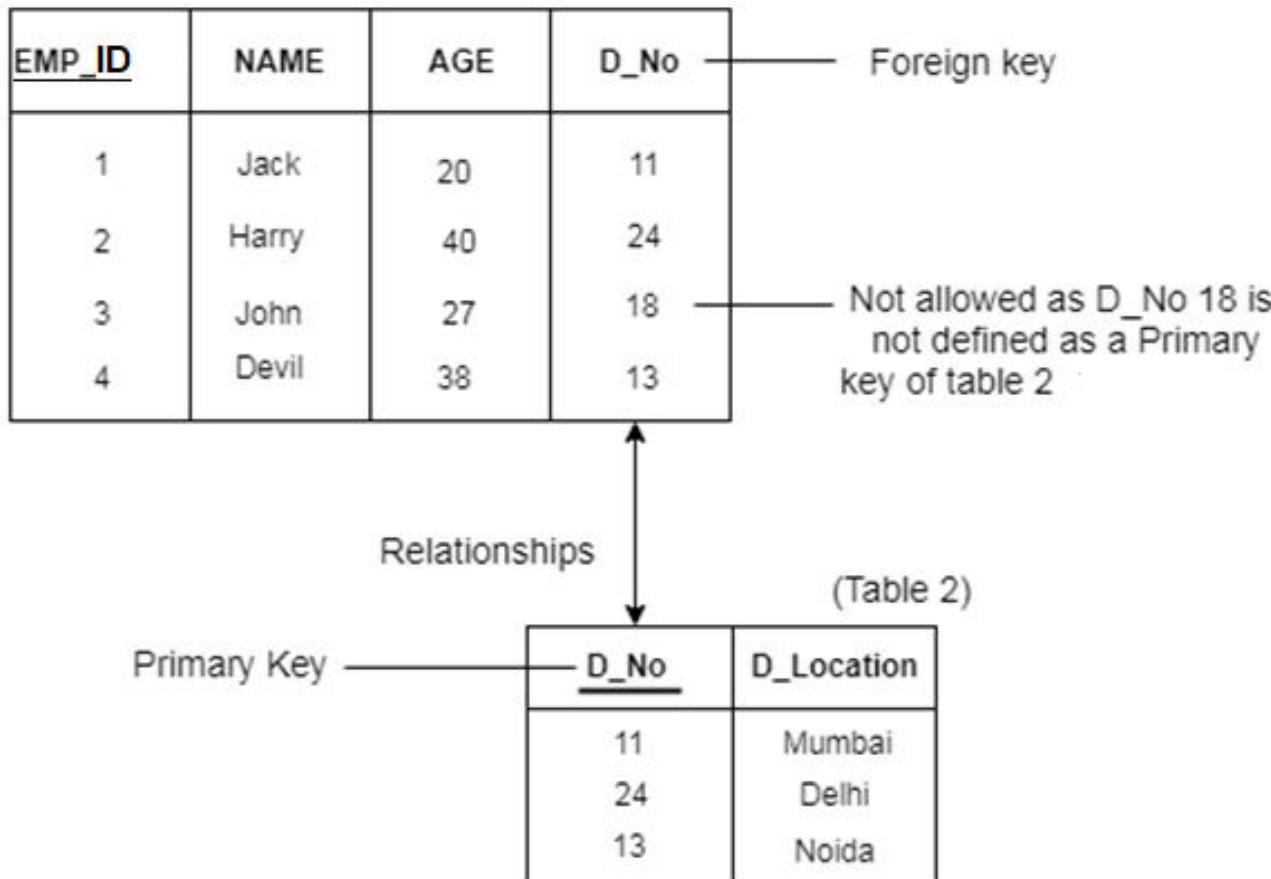
Referential integrity requires that a foreign key must have a matching primary key or it must be null.

- This constraint is specified between two tables (parent and child); it maintains the correspondence between rows in these tables. It means the reference from a row in one table to another table must be valid.

- Example:

Here, Table 1= Foreign key table and Table 2= Primary key table

(Table 1)



In the Referential integrity constraints, if a foreign key in Table 1 refers to the Primary Key of Table 2, then every value of the Foreign Key in Table 1 must be null or be available in Table 2.

iii. Domain integrity

- Domain constraints specify that the value of each attribute must be atomic (single) value and each data must have same data format/type.
- The primary unit of data in the relational data model is the data item. Such data items are said to be non-decomposable or atomic.
- A domain is a set of values of the same type.
- Domains are therefore pools of values from which actual values appearing in the columns of a table are drawn.

- Example:

Student

ID	NAME	SEMENSTER	AGE
1000	Tom	1 st	17
1001	Johnson	2 nd	24
1002	Leonardo	5 th	21
1003	Kate	3 rd	19
1004	Morgan	8 th	A

Not allowed. Because AGE has data type integer.

Relational Constraints

Types of constraints in RDBMS

1. Domain Constraints:

- Domain constraints specify that the value of each attribute must be atomic value and each data must have same data format.

2. Key Constraints:-

- No two tuples can have the same combination of values for all their attributes.

3. NOT NULL Constraints:-

- This constraint prohibits the insertion of a NULL value for this attributes.

4. UNIQUE Constraints:

- The value of attributes for each tuples must be unique. It may permit NULL value.

5. Primary Key Constraints:

- It ensure that certain attributes or set of attributes must have unique value and also it cannot permits null value.

6. CHECK Constraints:

- If ensure that each value of attributes must satisfy the given condition balance decimal(7,2) CHECK(balance >2000)

7. Foreign key Constraints:

- Foreign key always points to primary key of same or other relation. It ensures that foreign key always points to void primary key. i.e. Value of foreign key must be in primary key.

5.2 Functional dependency (FD)

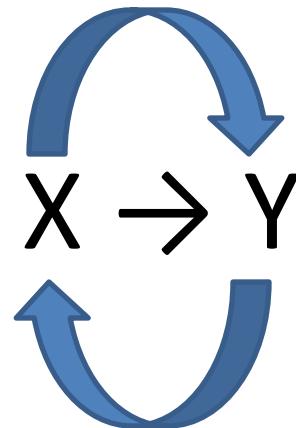
- FD is a relationship between two attributes, typically between the PK and other non-key attributes within a table.
- **Statement:** “An attribute in a relational model table is said to be functionally dependent on another attribute in the table if it can take only one value for a given value of the attribute upon which it is functionally dependent”
- or
- A functional dependency is a property of the semantics of the attributes in a relation.
 - The semantics indicate how attributes relate to one another, and specify the functional dependencies between attributes.
- It helps to maintain the quality of data in the database.
- It plays a vital role to find the difference between good and bad database design.

- A functional dependency is denoted by an arrow " \rightarrow ".
- The functional dependency of X on Y is represented by $X \rightarrow Y$.

Here, Y is functionally dependent on X.

X functionally determines Y.

X functionally determines Y.



Y is functionally dependent on X.

- Example:

Employee number	Employee Name	Salary	City
1	Dana	50000	San Francisco
2	Francis	38000	London
3	Andrew	25000	Tokyo

- In this example, if we know the value of Employee number, we can obtain Employee Name, city, salary, etc. By this, we can say that the city, Employee Name, and salary are functionally depended on Employee number.

- We can represent relationships between attributes as:

Employee number → Employee Name

Employee number → Salary

Employee number → City

OR,

Employee number → Employee Name, Salary, City

Types of Functional Dependency

- a) Full-functional dependency
- b) Partial functional dependency
- c) Trivial functional dependency
- d) Non-trivial functional dependency
- e) Multi-valued functional dependency
- f) Join functional dependency

a) Full-functional dependency

Fully-functionally Dependency (FFD)

- An attribute is fully functional dependent on another attribute, if it is Functionally Dependent on that attribute and not on any of its proper subset.
- Example:

Student (S_id, S_name, Subject_id)

FDs: $(S_id, Subject_id) \rightarrow S_name$

Here,

S_name is functionally dependent on $(S_id, Subject_id)$.

But,

$S_id \rightarrow S_name$ **(It holds.)**

$Subject_id \rightarrow S_name$ **(It doesn't hold.)**

S_name should not functionally dependent on S_id and $Subject_id$ separately.

Therefore, S_name is FFD on $(S_id, Subject_id)$.

So, here full-functional dependency is presence.

b) Partial functional dependency

- Partial Dependency occurs when a non-prime attribute is functionally dependent on part of a candidate key.
- An attribute is partially dependent on another attribute, if it is Functionally Dependent on that attribute and also on any of its proper subset.
- Example:

Employee (E_id, E_name, Citizenship_number)

Here, FDs:

$$(E_id, \text{Citizenship_number}) \rightarrow E_name$$

E_name is Functionally dependent on (E_id, Citizenship_number).

Also, $E_id \rightarrow E_name$ **(It holds.)**

$$\text{Citizenship_number} \rightarrow E_name$$
 (It holds.)

Therefore, E_name is functionally dependent on subset of (E_id, Citizenship_number) also.

So, here partial dependency is presence.

C) Trivial functional dependency

- $A \rightarrow B$ has trivial functional dependency if B is a subset of A.
- The following dependencies are also trivial like:
 $A \rightarrow A, B \rightarrow B$
- **Example:**
- Consider a table with two columns Employee_Id and Employee_Name.
- $\{Employee_id, Employee_Name\} \rightarrow Employee_Id$ is a trivial functional dependency.

d) Non-trivial functional dependency

- $A \rightarrow B$ has a non-trivial functional dependency if B is not a subset of A.
- **Example:**

In Department relation, $Dept_id \rightarrow Dept_name$

e) Multi-valued functional dependency

- Multi-valued dependency occurs when two attributes in a table are independent of each other but, both depend on a third attribute.
- A multi-valued dependency consists of at least two attributes that are dependent on a third attribute that's why it always requires at least three attributes.
- If a relation (r) having attributes A,B,C in which B and C are multi-valued facts about A, then it is represented as $A \rightarrow\!\!> B$ and $A \rightarrow\!\!> C$.
Then, multi-valued dependency exists only in B and C. B and C are independent of each other.

Example:

Car_Detail

Car_model	Maf_year	Color
H001	2017	Metallic
H001	2017	Green
H005	2018	Metallic
H005	2018	Blue

- In this example, Maf_year and Color are independent of each other but dependent on Car_model. In this example, these two columns are said to be multi-valued dependent on car_model.
- This dependence can be represented like this:

Car_model -> Maf_year

Car_model->> Colour

f) Join functional dependency

- If a table can be recreated by joining multiple tables and each of these tables have a subset of the attributes of the table, then the table is in Join Dependency.
- It is a generalization of Multi-valued Dependency.
- Example:

$R(A,B,C,D)$

Here, R=relation/table

A,B,C,D= attribute/column

Then, join dependency: $A \rightarrow B$

$A,B \rightarrow C$

$A,B,C \rightarrow D$

Example 2:

The diagram illustrates a database join operation. At the top, there are two tables, Table 1 and Table 2, each with three rows. Table 1 has columns S and P, with values S1, S1, S2 and P1, P2, P1 respectively. Table 2 has columns P and L, with values P1, P2, P1 and L2, L1, L1 respectively. Two black arrows point from the right side of Table 1 and Table 2 down to a third table, Table 3, located below them. Between the arrows, the word "Join" is written in red. Table 3 has columns S, P, and L, containing the combined data from both tables: S1, P1, L2; S1, P2, L1; S2, P1, L1; and S1, P1, L1.

Table 1		Table 2	
S	P	P	L
S1	P1	P1	L2
S1	P2	P2	L1
S2	P1	P1	L1

Table 3		
S	P	L
S1	P1	L2
S1	P2	L1
S2	P1	L1
S1	P1	L1

Example 3:

Student (S_id, S_name, S_address)

Marks (S_id, DBMS_marks, Math_marks)

Resultant relation after joining above two relations Student and Marks:

New_table (S_id, S_name, S_address, DBMS_marks, Math_marks)

5.3 Inference rules for functional dependency

- These rules are also known as Axioms of functional dependency.
- **Armstrong's axioms** are a set of axioms (inference rules) used to infer all the functional dependencies on a relational database.
- They were developed by William W. Armstrong on his 1974 paper.
- Armstrong's rules:
 - i. Reflexivity rule:
 - If A and B are set of attributes from R and $B \subseteq A$, then $A \rightarrow B$ holds.
 - ii. Augmentation rule:
 - If $A \rightarrow B$ holds and C is a set of attributes then $AC \rightarrow BC$ also holds.
 - iii. Transitivity rule:
 - If $A \rightarrow B$ holds and $B \rightarrow C$ holds, then $A \rightarrow C$ holds.

- Some additional rules:
 - a) Union rule:
 - If $A \rightarrow B$ holds and $A \rightarrow C$ holds then $A \rightarrow BC$ holds.
 - b) Decomposition rule:
 - If $A \rightarrow BC$ holds then $A \rightarrow B$ and $A \rightarrow C$ holds.
 - c) Pseudo-trasitivity rule:
 - If $A \rightarrow B$ holds and $BC \rightarrow D$ holds then $AC \rightarrow D$ holds.

5.4 Decomposition of relation

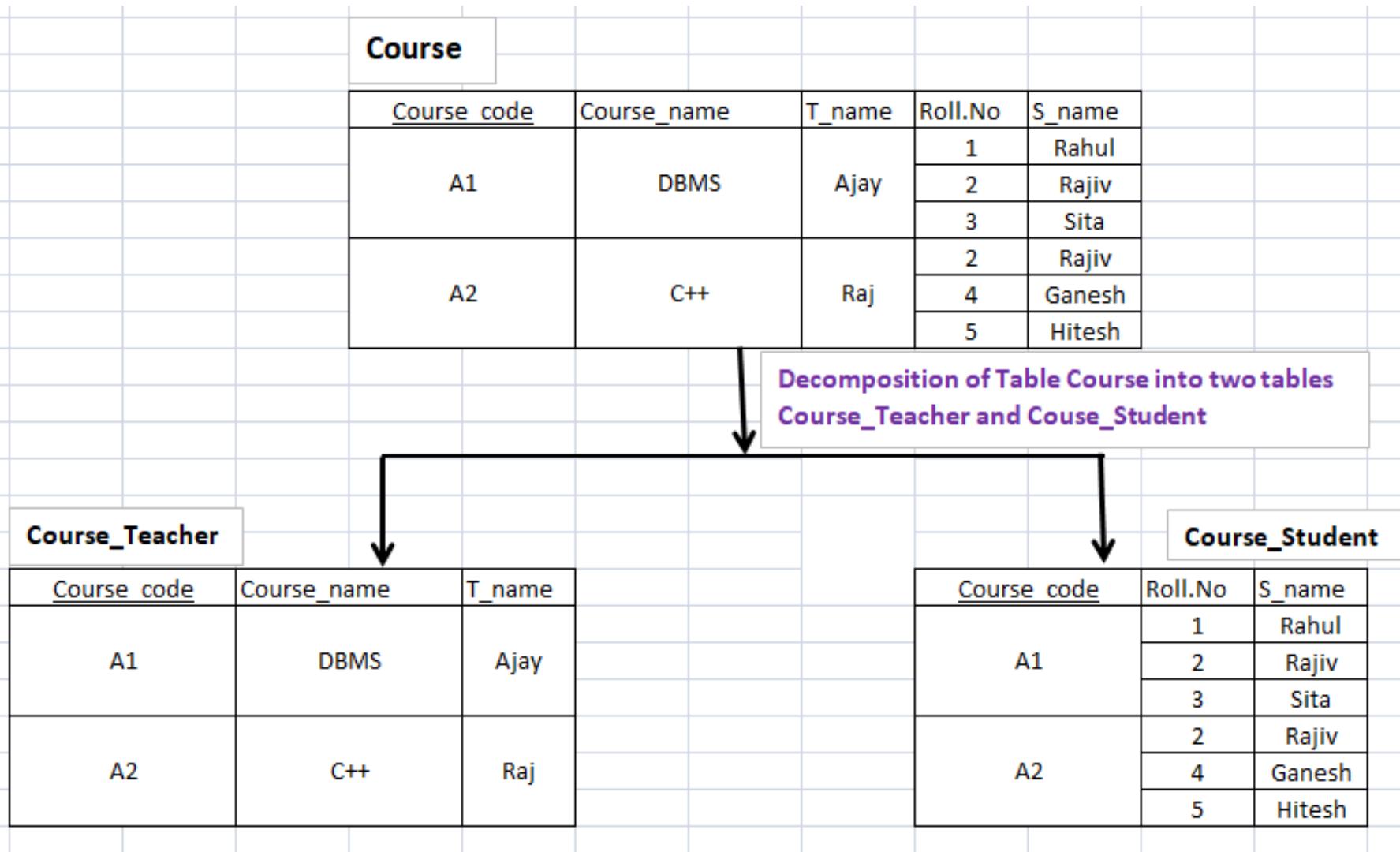
- Decomposition of relation (table) involves separating the attributes (columns) of the relation to create two or more new relations.
- When a relation in the relational model is not in appropriate normal form then the decomposition of a relation is required.
- In a database, it breaks the table into multiple tables.
- If the relation has no proper decomposition, then it may lead to problems like loss of information.
- Decomposition is used to eliminate some of the problems of bad design like anomalies, inconsistencies, and redundancy.

Rules of decomposition:

Rule1: one of two table contains the table identifier (PK) of original table and all of non-repeating attributes.

Rule2: The other table contains the table identifier (PK) of original table and all of repeating attributes.

Example:



5.5 Closure Set of Functional Dependency and attributes

Closure set of FD:

- Closure set of functional dependencies ‘F’ is set of all FDs that includes F as well as all dependencies that can be inferred from F.
- The Closure Of Functional Dependency means the complete set of all possible attributes that can be functionally derived from given functional dependency using the inference rules known as **Armstrong’s Rules**.
- If “F” is a functional dependency then closure of functional dependency can be denoted using “ F^+ ”.
- It is also referred as a complete set of FDs.
- Example1:

R (A, B, C)

FDs/F: { A \rightarrow B }

B \rightarrow C }

Here, R=relation/table

A, B= attributes/Column name

Then,

$F^+ = \{ A \rightarrow C \}$, Using rule of transitivity.

Example2

$R(A, B, C, G, H, I)$

$F: \{ A \rightarrow B \dots \text{(i)}$

$A \rightarrow C \dots \text{(ii)}$

$CG \rightarrow H \dots \text{(iii)}$

$CG \rightarrow I \dots \text{(iv)}$

$B \rightarrow H \dots \text{(v)}$

}

Find F^+ .

Solution:

Using inference rules,

$F^+ : \{ A \rightarrow H \text{ , Using transitivity rule in equations (i) and (v).}$

$CG \rightarrow HI \text{, Using union rule in equations (iii) and (iv).}$

$AG \rightarrow H \text{, Using pseudo-transitivity rule in equations (ii) and (iii).}$

$AG \rightarrow I \text{, Using pseudo-transitivity rule in equations (ii) and (iv).}$

}

Example 3:

Student (S_id, S_name, S_address, S_marks, S_position, S_attendance, S_performance)

F: { S_id \rightarrow S_name(i)

S_id \rightarrow S_address(ii)

S_id \rightarrow S_marks(iii)

S_marks \rightarrow S_position(iv)

S_id, S_marks \rightarrow S_performance(v)

}

Find F⁺.

Solution:

F⁺ :{

S_id \rightarrow S_name, S_address, S_marks ; Union rule in (i), (ii) and (iii).

S_id \rightarrow S_position; Transitivity rule in (iii) and (iv).

}

Example-4

Student (S_id, S_name, S_address, S_marks, S_position, S_attendance,
S_performance)

F: { S_id \rightarrow S_name(i)
S_id \rightarrow S_address(ii)
S_id \rightarrow S_marks(iii)
S_position \rightarrow S_marks(iv)
S_id, S_marks \rightarrow S_performance(v)
}

Find F⁺.

Solution:

F⁺ :{
S_id \rightarrow S_name, S_address, S_marks ; Union rule in (i), (ii) and (iii).
S_id, S_position \rightarrow S_performance ; Pseudo-transitivity rule in (iv) and (v).
}

Questions: (Assignment)

1. R (A, B, C, D, E)

F: { A \rightarrow BC

CD \rightarrow E

B \rightarrow D

E \rightarrow A }

Find F⁺.

2. Consider a relation R (A, B, C, D, E, F, G) with the functional dependencies-

A \rightarrow BC

BC \rightarrow DE

D \rightarrow F

CF \rightarrow G

Find Closure set of functional dependencies.

Closure of Attributes

- Closure of attribute 'A' is the set of attributes which can be determined by attribute A.
- It is denoted by A^+
- Attribute closure of an attribute set can be defined as set of attributes which can be functionally determined from it.
- **How to find attribute closure of an attribute set?**
 - Add elements of attribute set to the result set.
 - Recursively add elements to the result set which can be functionally determined from the elements of the result set.

Example:

$R (A, B, C)$

$F: \{ A \rightarrow B$
 $B \rightarrow C$
}

Find closure of attributes.

- (i) $A^+ ?$ (ii) $B^+ ?$ (iii) $C^+ ?$

Solution:

$A^+ : \{A$
 $A, B ;$ Using $A \rightarrow B$
 $ABC ;$ Using $B \rightarrow C$
}

$B^+ : \{B$
 $BC ;$ Using $B \rightarrow C$
}

$C^+ : \{C$
}

Nothing is determined from C.

Questions: (Assignment)

1. $R(A, B, C, D)$

$F: \{ A \rightarrow B$

$B \rightarrow D$

$C \rightarrow B$

}

Find closure of attributes.

- (i) $A^+ ?$ (ii) $B^+ ?$ (iii) $C^+ ?$ (iv) $D^+ ?$

2. $R(A, B, C, D, E, F, G)$

$F: \{ A \rightarrow B$

$BC \rightarrow DE$

$AEG \rightarrow G$

}

Find $(AC)^+ .$

5.6 Dependency preservation

- It is an important constraint of the database.
- A decomposition of a relation R into R1, R2, R3,, Rn is dependency preserving decomposition with respect to the set of FDs 'F' that hold on R only if the following is hold:

$$(F_1 \cup F_2 \cup F_3 \cup \dots \cup F_n)^+ = F^+$$

Where,

- $F_1, F_2, F_3, \dots, F_n$ = set of FDs of relations R1, R2, R3, ..., Rn.
- $(F_1 \cup F_2 \cup F_3 \cup \dots \cup F_n)^+$ = Closure of union of sets of FDs.
- F^+ = Closure set of FD 'F' of R.
- In the dependency preservation, at least one decomposed table must satisfy every dependency.

Example:

Let a relation $R(A,B,C,D)$ and set a FDs (F) = { $A \rightarrow B$, $A \rightarrow C$, $C \rightarrow D$ } are given.

A relation R is decomposed into - R_1 and R_2

$R_1 = (A, B, C)$ with FDs $F_1 = \{A \rightarrow B, A \rightarrow C\}$,

{ $A \rightarrow B, A \rightarrow C$ } these FDs are preserved in relation R_1 .

$R_2 = (C, D)$ with FDs $F_2 = \{C \rightarrow D\}$.

{ $C \rightarrow D$ } this FD is preserved in relation R_2 .

$F' = F_1 \cup F_2 = \{A \rightarrow B, A \rightarrow C, C \rightarrow D\}$ so, $F' = F$.

And so, $F'^+ = F^+$.

Thus, the decomposition is dependency preserving decomposition.

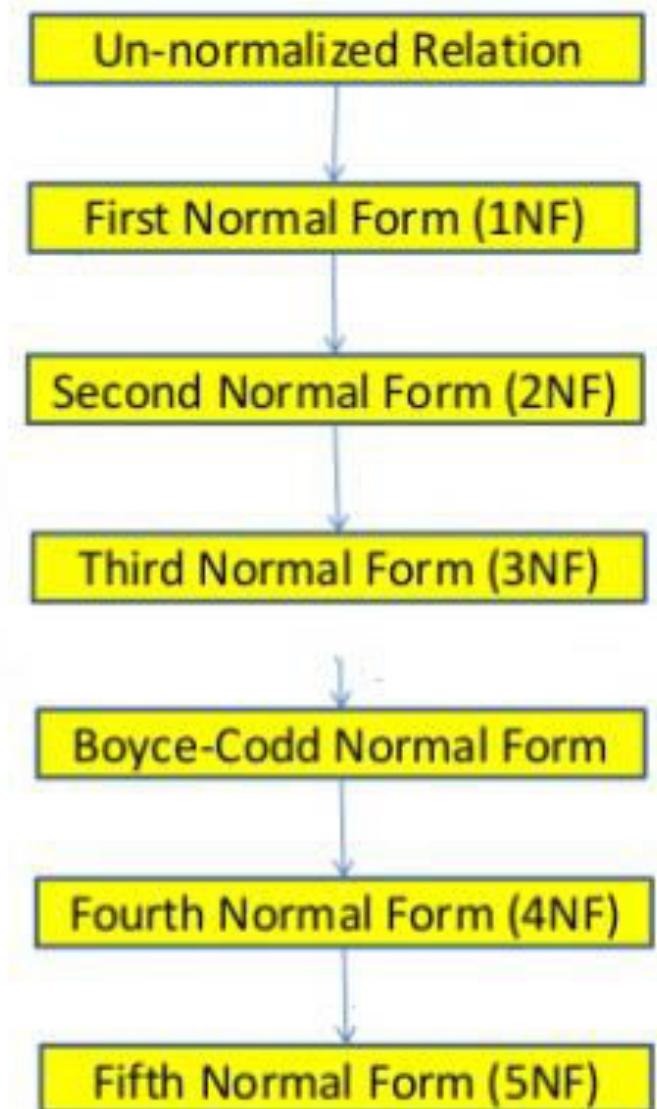
5.7 Normalization, Role of Normalization

Normalization:

- Normalization is the process of organizing the data in the database.
- It is the process of organizing the columns (attributes) and tables (relations) of a relational database to reduce data redundancy and improve data integrity.
- It is also used to eliminate the undesirable characteristics like Insertion, Update and Deletion anomalies.
- Normalization divides the larger table into the smaller table and links them using relationship.
- Normalization is used for mainly two purposes:
 - Eliminating redundant(useless) data.
 - Ensuring data dependencies i.e data is logically stored.

Normal Forms

- The database community sets the few rules or guidelines for database normalization. These rules are called as **normal forms**.
- Normal form starts from the **1NF** till **5NF**.



First Normal Form (1NF)

- As per the rule of first normal form, an attribute (column) of a table cannot hold multiple values.
i.e. It should hold only atomic (single) values.
- Example:

Student

<u>Student_name</u>	<u>Age</u>	<u>Subject</u>
Ram	22	DBMS, Math
Gita	21	Math
Gina	20	Microprocessor
Hari	21	DBMS

Student table in 1NF:

<u>Student_name</u>	<u>Age</u>	<u>Subject</u>
Ram	22	DBMS
Ram	22	Math
Gita	21	Math
Gina	20	Microprocessor
Hari	21	DBMS

Second Normal Form (2NF)

- A table is said to be in 2NF if both the following conditions hold:
 - Table is in 1NF (First normal form)
 - No non-prime attribute is dependent on the proper subset of any candidate key of table.
i.e. The primary key of the table should compose of exactly 1 column.
- An attribute that is not part of any candidate key is known as **non-prime attribute**.

Example of 2NF:

Teacher

teacher_id	subject	teacher_age
111	Maths	38
111	Physics	38
222	Biology	38
333	Physics	40
333	Chemistry	40

Candidate Keys: {teacher_id, subject}

Non prime attribute: teacher_age

Here, teacher_age depends on composite key (teacher_id, subject)

Teacher table in 2NF:

Teacher_Details

teacher_id	teacher_age
111	38
222	38
333	40

In this table, teacher_age depends on only teacher_id (primary key).

Teacher_Subject

teacher_id	subject
111	Maths
111	Physics
222	Biology
333	Physics
333	Chemistry

In this table, subject depends on only teacher_id (primary key).

Third Normal Form (3NF)

- A table design is said to be in 3NF if both the following conditions hold:
 - Table must be in 2NF
 - Transitive functional dependency of non-prime attribute on any super key should be removed.
- Example:

Tournament

Tournament	Year	Winner	Winner DOB
Indiana Invitational	1998	All Fredrickson	1975/5/9
Cleveland	1999	Bob Albertson	1968/9/22
Des Moines	1999	All Fredrickson	1975/5/9
Indiana Invitational	1998	Chip Masterson	1977/7/13

In this table, following transitive functional dependencies are existing:

(Tournament, Year) \rightarrow Winner

Winner \rightarrow Winner DOB

Here, transitive dependency is presence.

Tournament table will be in 3NF if we divide Tournament table as below:

Tournament_Details

<u>Tournament</u>	<u>Year</u>	<u>Winner</u>
Indiana Invitational	1998	All Fredrickson
Cleveland	1999	Bob Albertson
Des Moines	1999	All Fredrickson
Indiana Invitational	1998	Chip Masterson

Tournament_Winner

<u>Winner</u>	<u>Winner_DOB</u>
All Fredrickson	1975/5/9
Bob Albertson	1968/9/22
All Fredrickson	1975/5/9
Chip Masterson	1977/7/13

Note: No Transitive functional dependency

Boyce-Codd Normal Form (BCNF)

- For a table to satisfy the Boyce-Codd Normal Form, it should satisfy the following two conditions:
 - It should be in the **Third Normal Form**.
 - And, for any functional dependency $A \rightarrow B$, A should be a **super key** of the table.

EMPLOYEE table:

EMP_ID	EMP_COUNTRY	EMP_DEPT	DEPT_TYPE	EMP_DEPT_NO
264	India	Designing	D394	283
264	India	Testing	D394	300
364	UK	Stores	D283	232
364	UK	Developing	D283	549

In the above table Functional dependencies are as follows:

$EMP_ID \rightarrow EMP_COUNTRY$

$EMP_DEPT \rightarrow \{DEPT_TYPE, EMP_DEPT_NO\}$

Candidate key: {EMP-ID, EMP-DEPT}

The table is not in BCNF because neither EMP_DEPT nor EMP_ID alone are keys.

To convert the given table into BCNF, we decompose it into three tables:

EMP_COUNTRY table:

<u>EMP_ID</u>	EMP_COUNTRY
264	India
264	India

EMP_DEPT table:

<u>EMP_DEPT</u>	DEPT_TYPE	EMP_DEPT_NO
Designing	D394	283
Testing	D394	300
Stores	D283	232
Developing	D283	549

EMP_DEPT_MAPPING table:

<u>EMP_ID</u>	EMP_DEPT
D394	283
D394	300
D283	232
D283	549

- **Candidate keys:**
 - **For the first table:** EMP_ID
 - For the second table:** EMP_DEPT
 - For the third table:** {EMP_ID, EMP_DEPT}
- Now, this is in BCNF because left side part of both the functional dependencies is a key. i.e. single key determines all other attributes in each table.

Fourth Normal Form (4NF)

- The database must satisfy the following two things;
 - The database must meet all the requirement of 3NF
 - There should be no more than one multi-valued dependencies.
- OR, The table should not have any **Multi-valued Dependency**.
- According to the 4th normal form, a record type should not contain two or more independent multi-valued facts about an entity.

Example of 4NF:

Student		
s_id	course	hobby
1	Science	Cricket
1	Maths	Hockey
1	Science	Hockey
1	Maths	Cricket

- In the table above, there is no relationship between the columns course and hobby. They are independent of each other.
- And both are depends on same attribute s_id.
- So there is multi-value dependency, which leads to un-necessary repetition of data and other anomalies as well.

Course Table

s_id	course
1	Science
1	Maths
2	C#
2	Php

And, **Hobbies Table**,

s_id	hobby
1	Cricket
1	Hockey
2	Cricket
2	Hockey

This tables are in 4NF, because there is no multi-value dependency.

Fifth Normal Form (5NF)

- Table should be in 4NF.
- If we can decompose table further to eliminate redundancy and anomaly, and when we re-join the decomposed tables by means of candidate keys, we should not be losing the original data or any new record set should not arise.
(In simple words, joining two or more decomposed table should not lose records nor create new records.)

OR, Eliminate join dependency.

Example of 5NF:

P

SUBJECT	LECTURER	SEMESTER
Computer	Anshika	Semester 1
Computer	John	Semester 1
Math	John	Semester 1
Math	Akash	Semester 2
Chemistry	Praveen	Semester 1

So to make the above table into 5NF, we can decompose it into three relations P1, P2 & P3:

P1

SEMESTER	SUBJECT
Semester 1	Computer
Semester 1	Math
Semester 1	Chemistry
Semester 2	Math

P2

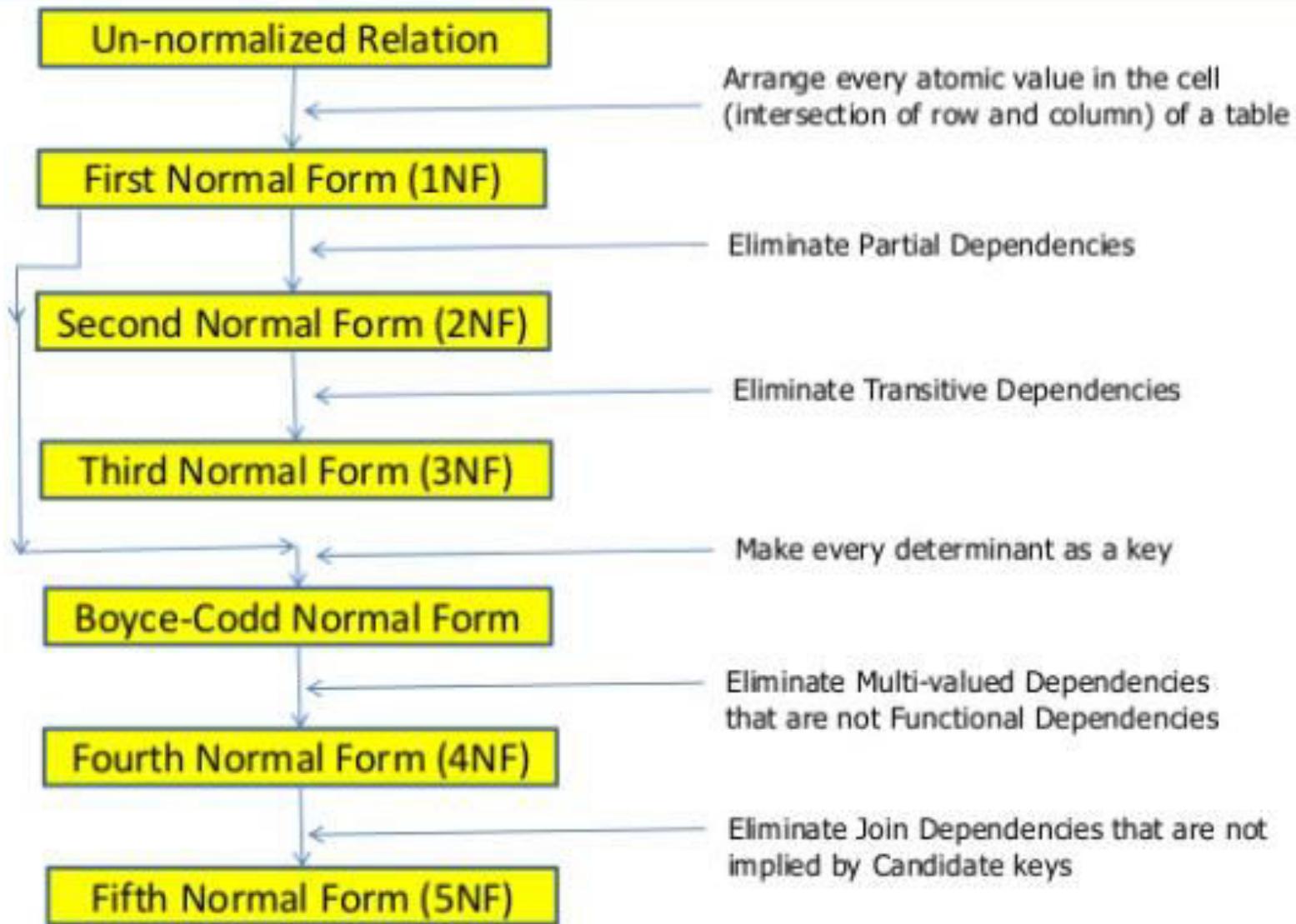
SUBJECT	LECTURER
Computer	Anshika
Computer	John
Math	John
Math	Akash
Chemistry	Praveen

P3

SEMSTER	LECTURER
Semester 1	Anshika
Semester 1	John
Semester 1	John
Semester 2	Akash
Semester 1	Praveen

The join relation of these three relations is equal to the original relation P.

Summary of Normal Forms



Role of Normalization

The following makes Database Normalization a crucial step in database design process :

- **Resolving the database anomalies**
 - The forms of Normalization remove all the Insert, Update and Delete anomalies.
 - **Insertion Anomaly** occurs when you try to insert data in a record that does not exist.
- **Eliminate Redundancy of Data**
- **Data Dependency**
 - The data gets stored in the correct table and ensures normalization.
- **Isolation of Data**
 - A good designed database states that the changes in one table or field do not affect other. This is achieved through Normalization.
- **Data Consistency**
 - While updating if a record is left, it can lead to inconsistent data, Normalization resolves it and ensures Data Consistency.
- **Much more flexible DB design.**
- **A better handle on DB security, etc.**

Subject-DBMS

Chapter-6 Database Security

By: Er. Kalpana Karki

Contents

6. Database Security

6.1 Importance of database security

6.2 Different levels of security

6.3 Confidentiality, Authentication ,Authorization, Non-
Repudiation

6.4 Security and Views

6.5 Access control: Discretionary and Mandatory

6.6 Encryption and Decryption

Database Security

- **Database security** concerns the use of a broad range of information security controls to protect databases against compromises of their confidentiality, integrity and availability.
- It encompasses a range of security controls designed to protect the DBMS.
- Database security procedures are aimed at protecting not just the data inside the database, but the database management system and all the applications that access it from intrusion, misuse of data, and damage.
- Database security covers and enforces security on all aspects and components of databases. This includes:
 - Data stored in database.
 - Database server.
 - Database management system (DBMS).
 - Other database workflow applications.

Some of the ways database security is analyzed and implemented include:

- Restricting unauthorized access
- Physical security of the database server and backup equipment from theft and natural disasters.
- Reviewing the existing system for any known or unknown vulnerabilities and defining and implementing a road map/plan to mitigate them.
- Data encryption can provide an additional layer of security to protect the integrity and confidentiality of data.

Security aspects in DB security:

- a) Confidentiality
- b) Integrity
- c) Availability
- d) Authentication
- e) Authorization
- f) Non-repudiation

CIA Triad

- CIA= Confidentiality, integrity and availability
- It is a model designed to guide policies for information security within an organization.
- **Confidentiality**
 - Confidentiality is roughly equivalent to [privacy](#).
- **Integrity**
 - It involves maintaining the consistency, accuracy, and trustworthiness of data over its entire life cycle.
- **Availability**
 - It refers to the actual availability of your data.



Note: For database security CIA triad must be ensure.

6.1 Importance of database security

- To prevent unauthorized data observation
- To prevent unauthorized data modification
- To ensure the data confidentiality
- To ensure the data integrity
- To ensure the data availability
- To make sure that only authorized user will have access to the data
- Recovery after failure

Database security safeguards defend against a myriad of security threats and can help protect your enterprise from:

- Deployment failure
- Privilege abuse
- Unmanaged sensitive data
- Backup data exposure
- Weak authentication
- Database injection attacks

Threat is a potential negative action or event facilitated by a vulnerability that results in an unwanted impact to a computer system or application.

6.2 Different levels of security

- **Database Level :**
 - Some database-system users may be authorized to access only a limited portion of the database. Other users may be allowed to issue queries, but may be forbidden to modify the data.
 - It is responsibility of the database system to ensure that these authorization restrictions are not violated.
 - database users and authorization
- **Application Level :**
 - Application like MS-office level of security is also important in any organization.
 - information management and processing
- **Operating System Level :**
 - No matter how secure the database system is, weakness in operating system security may serve as a means of unauthorized access to the database.
 - data storage and protection

- **Network Level :**
 - Since almost all database systems allow remote access through terminals or networks, software-level security within the network software is as important as physical security, both on the Internet and in networks private to an enterprise.
 - data transmission
- **Physical Level :**
 - The sites containing the computer systems must be secured against harm by intruders/hackers.
 - computer equipment protection
- **Human Level :**
 - Users must be authorized carefully to reduce the chance of any such user giving access to an intruder in exchange for a bribe or other favors .
 - social engineering protection

6.3 Confidentiality, Authentication ,Authorization, Non-Reputation

- **Confidentiality**
 - It is the property that information is not made available or disclosed to unauthorized individuals, entities, or processes.
 - Confidentiality is keeping information secret or private.
 - Confidentiality might be important for military, business or personal reasons.
- **Authentication** is the property that an entity is what it claims to be. Example: Id and password, finger print, etc.
- **Authorization** is the process that determines and approves the privileges for an authenticated access. Example: person with id and password is authorized person.
- **Non-reputation**
 - It is the presentation of un-forgeable evidence that a message was sent or received.
 - The term is often seen in a legal setting wherein the authenticity of a signature is being challenged.

6.4 Security and Views

Views in DB

- In a **database**, a **view** is the result set of a *stored* query on the data, which the database users can query just as they would in a persistent database collection object.
- Views in SQL are kind of virtual tables.
- A view also has rows and columns as they are in a real table in the database.
- We can create a view by selecting fields from one or more tables present in the database.

Importance of View in Database

1. Provide additional level of table security by restricting access to a predetermined set of rows or columns of a table.
2. Hide Data complexity.
3. Simplify Statements for User: Views allow users to select information from multiple tables without actually knowing how to perform join.
4. Present Data in different perspective: Columns of views can be renamed without effecting the tables on which the views are based.
5. Isolate applications from changes in definitions of base tables.
6. Express query that cannot be expressed without using a view.
7. Saving of complex queries.

1. Syntax to create view:

```
CREATE VIEW view_name AS  
SELECT column1, column2.....  
FROM table_name  
WHERE condition;
```

Example:

```
CREATE VIEW DetailsView AS  
SELECT NAME, ADDRESS  
FROM StudentDetails  
WHERE S_ID < 5;
```

2. Syntax to see result:

```
SELECT * FROM view_name ;
```

Example: `SELECT * FROM DetailsView;`

3. Syntax to drop view:

DROP VIEW view_name;

Example: DROP VIEW DetailsView;

4. Syntax for update view:

CREATE OR REPLACE VIEW view_name AS

SELECT column1,coulmn2,..

FROM table_name

WHERE condition;

Example:

CREATE OR REPLACE VIEW MarksView AS

SELECT StudentDetails.NAME, StudentDetails.ADDRESS,
StudentMarks.MARKS, StudentMarks.AGE

FROM StudentDetails, StudentMarks

WHERE StudentDetails.NAME = StudentMarks.NAME;

6.5 Access control: Discretionary and Mandatory

Access Control

- It is a security technique that can be used to regulate who or what can view or use resource in a computing environment.
- An access control is a security model, which formally defined definition of a set of access control rules that is independent of technology or implementation platform.
- This security models are implemented within operating systems, networks, database management systems and back office, application and web server software.

Access Control Matrix

- It is a table of subjects and objects indicating what actions an individual subject can take upon individual object.

Object Subject	File 1	File 2	File 3	File 4
User 1	Read	Write	Read	-
User 2	Write	Read	-	-
User 3	Write	-	-	Read
User 4	Read	Read	Read	-

Permissions

Here, Subject= Users
Object= Resources

Types of Access Control

1. Mandatory Access Control (MAC):

- This access control is based on a security labeling system.
- User have security clearance and resources have security labels that contains data classification.
- Example: Military

2. Discretionary Access Control (DAC):

- This access control is at the discretion (judgment) of the owner.
- With DAC models, the data owner decides on access.
 - The owner of the company can decide how many people have access to a specific location. Each access control point has a list of authorized users.
 - i.e. The owner of the company decides which subject can access which objects.
- Example: Private organizations

3. Role-based Access Control (RBAC):

- Also known as non-discretionary access control,
- It uses a centrally administered set of controls to determine how subjects and objects interact.
- RBAC largely eliminates discretion when providing access to objects.
- It is the best method for an organization that has high turnover.
- Example: a human resources specialist should not have permissions to create network accounts; this should be a role reserved for network administrators.

4. Rule-based Access Control:

- It uses specific rules that indicate what can and cannot happen between a subject and an object.
- Rule-based access control can change the permissions based on a specific set of rules created by the administrator.
- Not necessarily identity based.
- Example: If your business closes at 5 p.m., there's no need for anyone to have access to your main office, even managers, after closing.

Other types:

- a) Content dependent access control
 - Access to the resource is determined by the content within the resource.
- b) Context based access control
 - It makes access decision based on the context of collection of information rather than contents within an objects.
 - Used in firewalls to decide which packets should enter the network and which not.

6.6 Encryption and Decryption

- Encryption is the process of translating original data (*plaintext*) into something that appears to be random and meaningless (cipher text).
- Decryption is the process of converting cipher text back to plaintext.
- Key is used in both encryption and decryption algorithm.
- The goal of every encryption algorithm is to make it as difficult as possible to decrypt the generated cipher text without using the key.

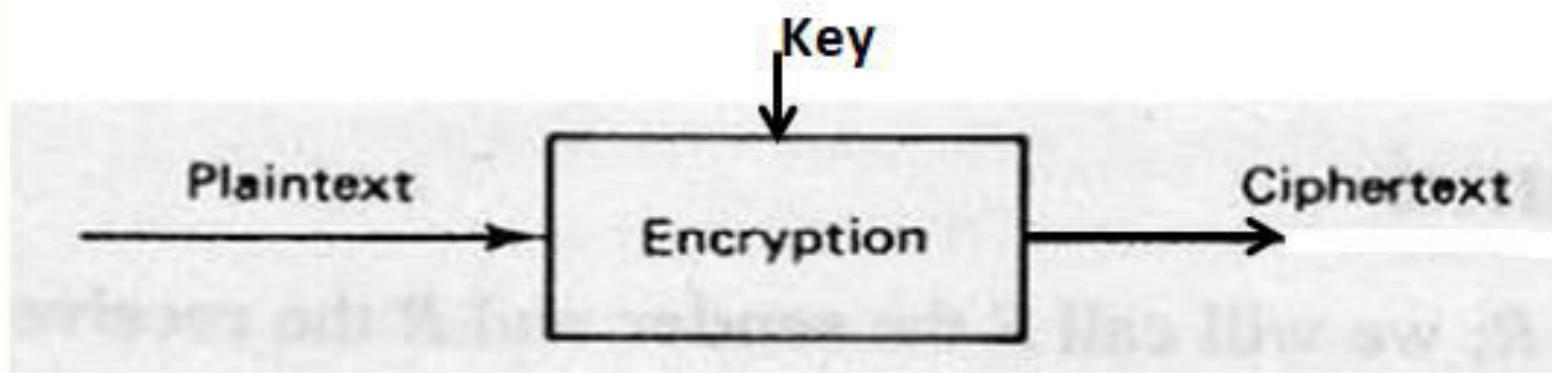


Fig: Encryption Process

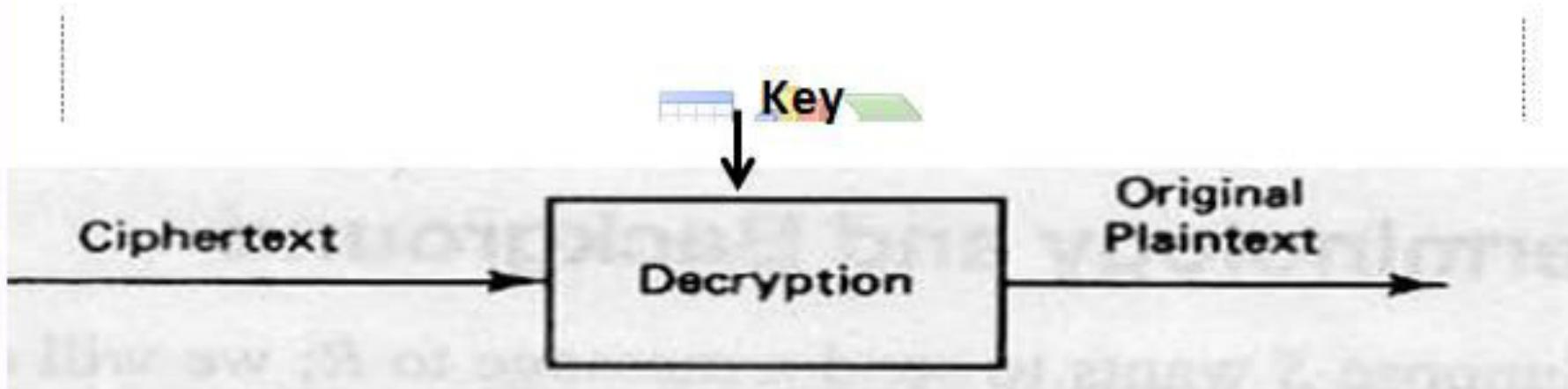


Fig: Decryption Process

Some encryption algorithms:

1. Substitution cipher
2. Transposition cipher
3. Caesar cipher, etc.

Example:

Substitution Method

a) Encryption:

Plain text: Hello

Key: { H -> #

e -> a

l -> 1

o -> S

}

Cipher text: #a11S

b) Decryption:

Cipher text: #a11S

Key: { # -> H

a -> e

1 -> l

S -> o

}

Plain text: Hello

Transposition method example:

Plain text: Hello

Cipher text: elolH

Example of caesar method:

Plain text: ABC

Key: ROT 3

Cipher text: DEF

Subject: DBMS

Chapter-7 Query Processing

By Er. Kalpana Karki

Contents

7. Query Processing

7.1 Introduction to Query Processing

7.2 Query Cost

7.3 Representing Queries using query tree

7.4 Query Optimization

7.5 Query Decomposition

7.1 Introduction to Query Processing

- Query is a statement requesting the retrieval of information from DB.
- Query processing is the procedure of selecting the best plan or strategy to be used in responding to a database request.
- It is the activity performed in extracting data from the database.
- It includes translations on high level Queries into low level expressions that can be used at physical level of file system, query optimization and actual execution of query to get the actual result.
- The component of the DBMS responsible for generating this strategy is called a query processor.
- It is a stepwise process.
- The cost of each access plan is estimated and the optimal one is chosen and executed.
- **What are the objectives of query processing?**
 - The aim of query processing is to transform a query written in a high level language into a correct and efficient execution strategy expressed in a low level language and execute the strategy to retrieve the data.

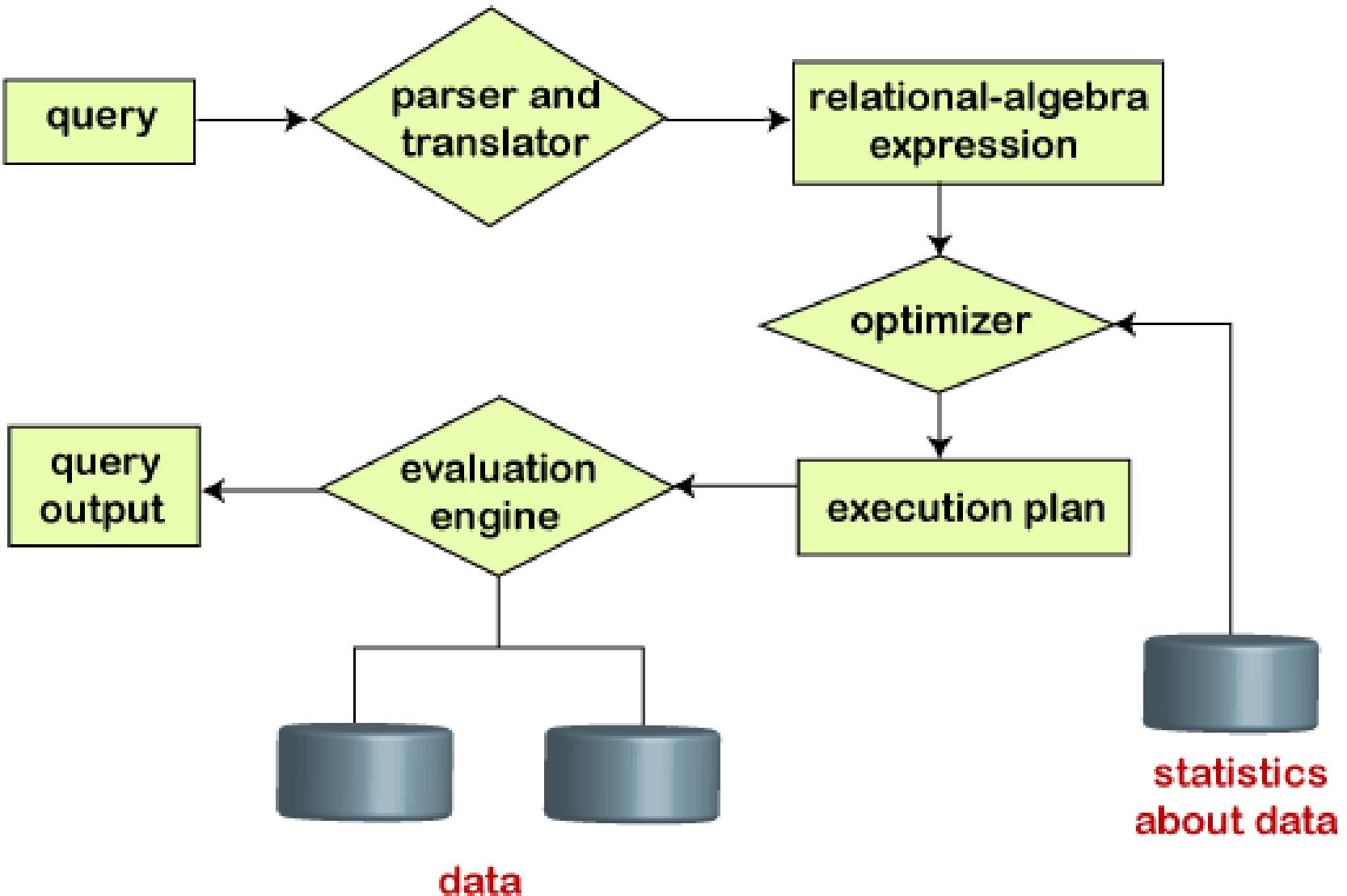


Figure: Steps in Query processing

- In query processing, it takes various steps for fetching the data from the database. The steps involved are:
 1. Parsing and translation
 2. Optimization
 3. Evaluation
- **Input:**
 - A query written in SQL is given as input to the query processor.
- **Step 1: Parsing**
 - In this step, the parser of the query processor module checks the syntax of the query, the user's privileges to execute the query, the table names and attribute names, etc.
 - The correct table names, attribute names and the privilege of the users can be taken from the system catalog (data dictionary).
 - Example: **SELECT * FORM Employee**
Here error of wrong spelling of FROM is given by this check.
Correct query: SELECT * FROM Employee
- **Step 2: Translation**
 - If we have written a valid query, then it is converted from high level language SQL to low level instruction in Relational Algebra.
 - Example: SQL query : **SELECT * FROM Employee**
Relational algebra: $\sigma_{E_name} (Employee)$

- **Step 3: Optimizer**
 - Optimizer uses the statistical data stored as part of data dictionary. The statistical data are information about the size of the table, the length of records, the indexes created on the table, etc.
 - Optimizer also checks for the conditions and conditional attributes which are parts of the query.
 - Optimizer selects best method to execute the query, which has low cost to execute.
- **Step 4: Execution Plan**
 - A query can be expressed in many ways.
 - A query plan (or query execution plan) is an ordered set of steps used to access data in a SQL relational database management system.
 - Finally, in this step, runs the query and display the required result.
- **Step 5: Evaluation**
 - Though we got many execution plans constructed through statistical data, though they return same result (obvious), they differ in terms of Time consumption to execute the query, or the Space required executing the query. Hence, it is mandatory choose one plan which obviously consumes less cost.
 - At this stage, we choose one execution plan of the several we have developed. This Execution plan accesses data from the database to give the final result.
- **Output:**
 - The final result is shown to the user.

7.2 Query Cost

- Cost is generally measured as total elapsed time for answering query.
- Many factors contribute to time cost:
 - disk accesses
 - CPU
 - network communication
- Basic cost parameters:
 - Cost of accessing disk block randomly
 - Data transfer rate
 - Clustering of data tuples on disk
 - Sort order of data tuples on disk
 - Cost of sorting intermediate results

The cost of executing query includes following components:

1. Access cost to secondary storage
2. Storage cost
3. Computation cost
4. Memory usage cost
5. Communication cost

7.3 Representing Queries using query tree

Query tree

- It is a tree data structure that corresponds to a relational algebra expression.
- The execution terminates when the root node is executed and produces the result relation for the query.
- Relational operations/commands used in Query tree:
 - Select
 - Project
 - Where
 - Join
- A Query Tree is also called a relational algebra tree.
 - **Leaf node** of the tree, representing the base input relations of the query.
 - **Internal nodes** result of applying an operation in the algebra.
 - **Root** of the tree representing a result of the query.

Query Graph

- A graph data structure that corresponds to a relational expression.
- It does not indicate an order on which operations to perform first, there is only a single graph corresponding to each query.
- Symbols used in query graph:
 - Single circle = Relation/table node
 - Double circle = Constant value
 - Graph edges = selection/join
 - Square brackets = Attributes

7.4 Query Optimization

- **Query optimization** is a feature of many relational database management systems (RDBMS) and other databases such as graph databases.
- The **query optimizer** attempts to determine the most efficient way to execute a given query by considering the possible [query plans](#).
- Query optimization is the science and the art of applying equivalence rules to rewrite the tree of operators used in a query and produce an optimal plan.

Methods of query optimization

1. Cost based Optimization (Physical)

- This is based on the cost of the query.
- The query can use different paths based on indexes, constraints, sorting methods etc.
- This method mainly uses the statistics like record size, number of records, number of records per block, number of blocks, table size, whether whole table fits in a block, organization of tables, uniqueness of column values, size of columns etc.

2. Heuristic Optimization (Logical)

- This method is also known as rule based optimization.
- This is based on the equivalence rule on relational expressions; hence the number of combination of queries get reduces here. Hence the cost of the query too reduces.
- This method creates relational tree for the given query based on the equivalence rules.
- These equivalence rules by providing an alternative way of writing and evaluating the query, gives the better path to evaluate the query.

7.5 Query Decomposition

- It is the first phase of query processing whose aims are to transform a high-level query into a relational algebra query and to check whether that query is syntactically and semantically correct.
- Thus, a query decomposition phase starts with a high-level query and transforms into a query graph of low-level operations (algebraic expressions), which satisfies the query.
- Process of minimizing the resources usage.
- **Aims of query decomposition:**
 - To transform a high-level query into Relational algebra query.
 - To check the query is syntactically and semantically correct.
 - It is efficient way to retrieve data from database.

Stages in Query Decomposition

1. Analysis

- Query is syntactically analyzed.
- It verifies relation, attribute and operations used in query.

2. Normalization

- It normalizes query for easy manipulation.

3. Semantic analysis

- It rejects normalized queries that are incorrectly formulated or contradictory.

4. Simplification/ Redundancy elimination

- It eliminates redundancy.
- It checks for integrity and security constraints.

5. Query restructuring

- Last stage of query decomposition.
- Query is restructured/re-write to provide a more efficient implementation.

Example of Query Decomposition

Consider Schema: Employee(ID, fname, lname, address, salary)

Question: Find the fname and lname of employees whose salary is greater than maximum salary of the relation.

SQL Query: Select fname, lname From Employee

Where salary > (select Max(salary) From Employee)

Query decomposition:

- Inner Block: (select Max(salary) From Employee)
- Outer Block: Select fname, lname From Employee
Where salary > C

here, C= output of inner block.

DBMS

Chapter-8

Filing and File Structure

Prepared By: Kalpana Karki

Contents

8. Filing and File Structure

8.1 Storage devices

8.2 Buffer Management

8.3 File Organization (sequential , indexed sequential,
hashed file)

8.4 Hash Collision: Detection and Resolution

8.5 Data Dictionary Storage

8.1 Storage Devices

- It is any hardware capable of holding information either temporarily or permanently.
- It is used for storing, porting and extracting data files and objects.
- Types of storage devices:
 - Primary storage device, such as RAM
 - Secondary storage device, such as a hard drive
 - Secondary storage can be removable, internal, or external

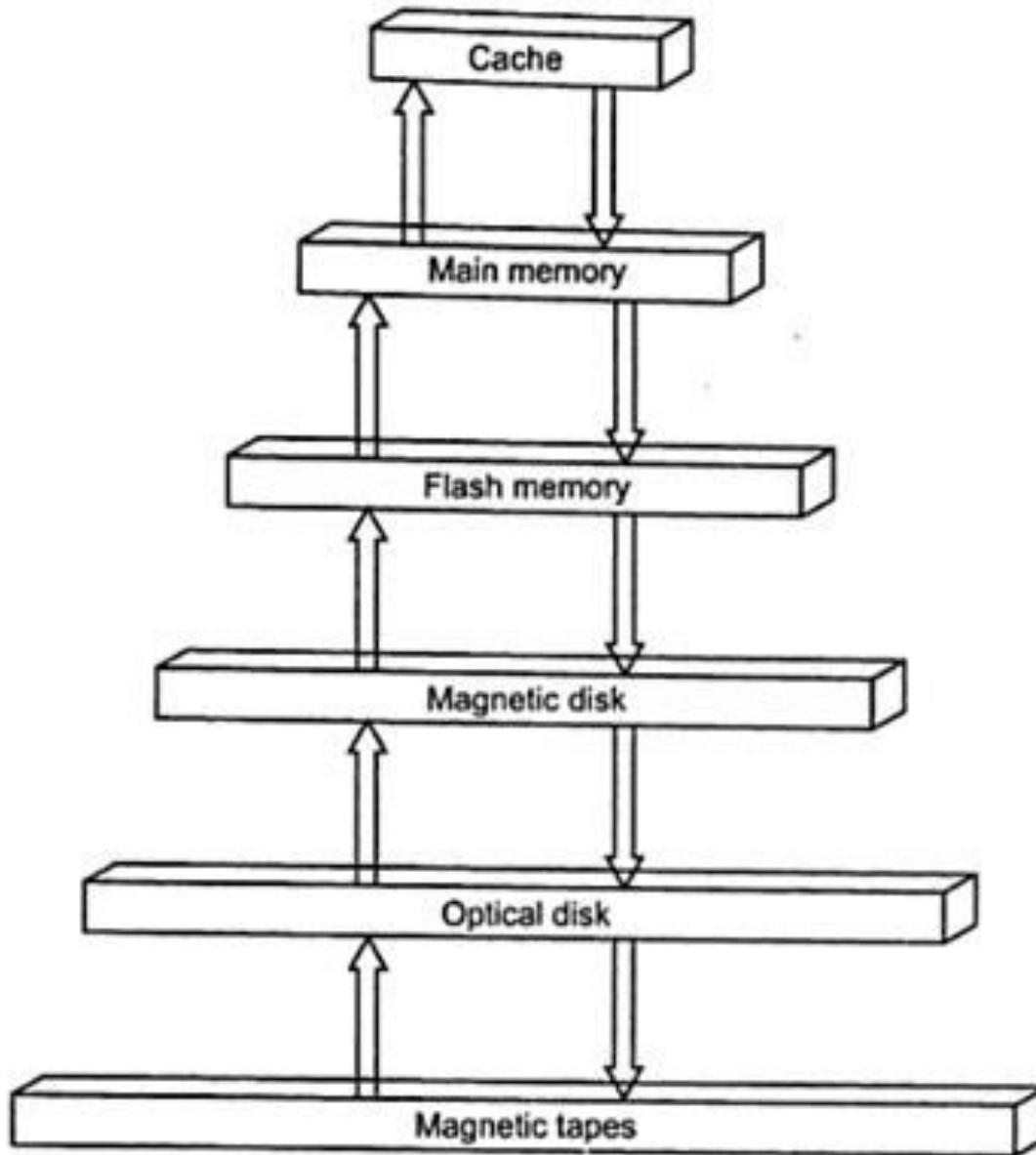


Fig. Storage device hierarchy

- **Cache Memory:**
 - Small-sized type of volatile computer memory that provides high-speed data access to a processor and stores frequently used computer programs, applications and data.
 - It is the fastest memory in a computer, and is typically integrated onto the motherboard and directly embedded in the processor or main random access memory (RAM).
- **Main Memory:**
 - Physical memory that is internal to the [computer](#).
 - Main memory include [RAM](#) and primary storage.
- **Flash Memory:**
 - An electronic (solid-state) non-volatile computer storage medium that can be electrically erased and reprogrammed.
 - Flash memory evolved from erasable programmable read-only memory ([EPROM](#)) and electrically erasable programmable read-only memory ([EEPROM](#)).
 - Example: Pen drive

- **Magnetic disk:**
 - A storage device that uses a magnetization process to write, rewrite and access data. It is covered with a magnetic coating and stores data in the form of tracks, spots and sectors.
 - Hard disks, zip disks and floppy disks are common examples of magnetic disks.
- **Optical disc:**
 - An electronic data storage medium that can be written to and read from using a low-powered laser beam.
 - Example: CD-ROM, DVD, etc.
- **Magnetic Tape:**
 - A system for storing digital information on magnetic tape using digital recording.
 - A medium for magnetic recording, made of a thin, magnetizable coating on a long, narrow strip of plastic film.

RAID(Redundant Array of Independent Disks)



(a) RAID 0 : nonredundant striping



(b) RAID 1 : mirrored disks



(c) RAID 2 : memory-style error-correcting codes



(d) RAID 3 : bit-interleaved parity



(e) RAID 4 : block-interleaved parity



(f) RAID 5 : block-interleaved distributed parity



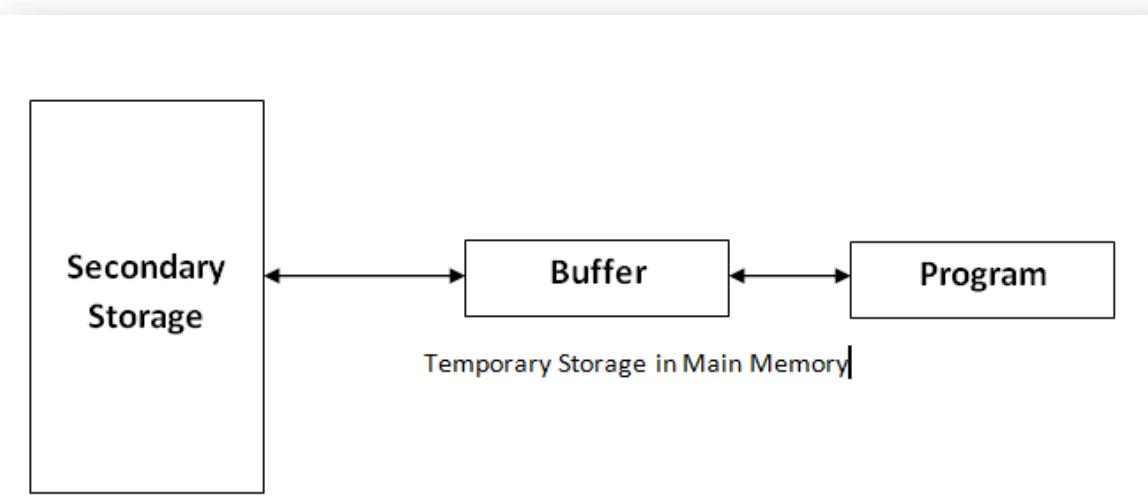
(g) RAID 6 : P + Q redundancy

Assignment: RAID (0-6)

8.2 Buffer Management

- **Buffer/ Data Buffer:**

- It is a region of a physical memory storage used to temporarily store data while it is being moved from one place to another.
- A temporary holding area for data while it's waiting to be transferred to another location.
- It is usually located in the RAM.
- The concept of the buffer was developed in order to prevent data congestion from an incoming to an outgoing port of transfer.



Buffer Management:

- Keep pages in a part of memory (buffer), read directly from there.
- What happens if you need to bring a new page into buffer and buffer is full: you have to remove one page.
- Replacement policy:
 - LRU : Least Recently Used (CLOCK)
 - MRU: Most Recently Used
 - Toss-immediate : remove a page if you know that you will not need it again
 - Pinning (needed in recovery, index based processing, etc)
 - Other DB specific RPs: DBMIN, LRU-k, 2Q

Page Requests from Higher Levels

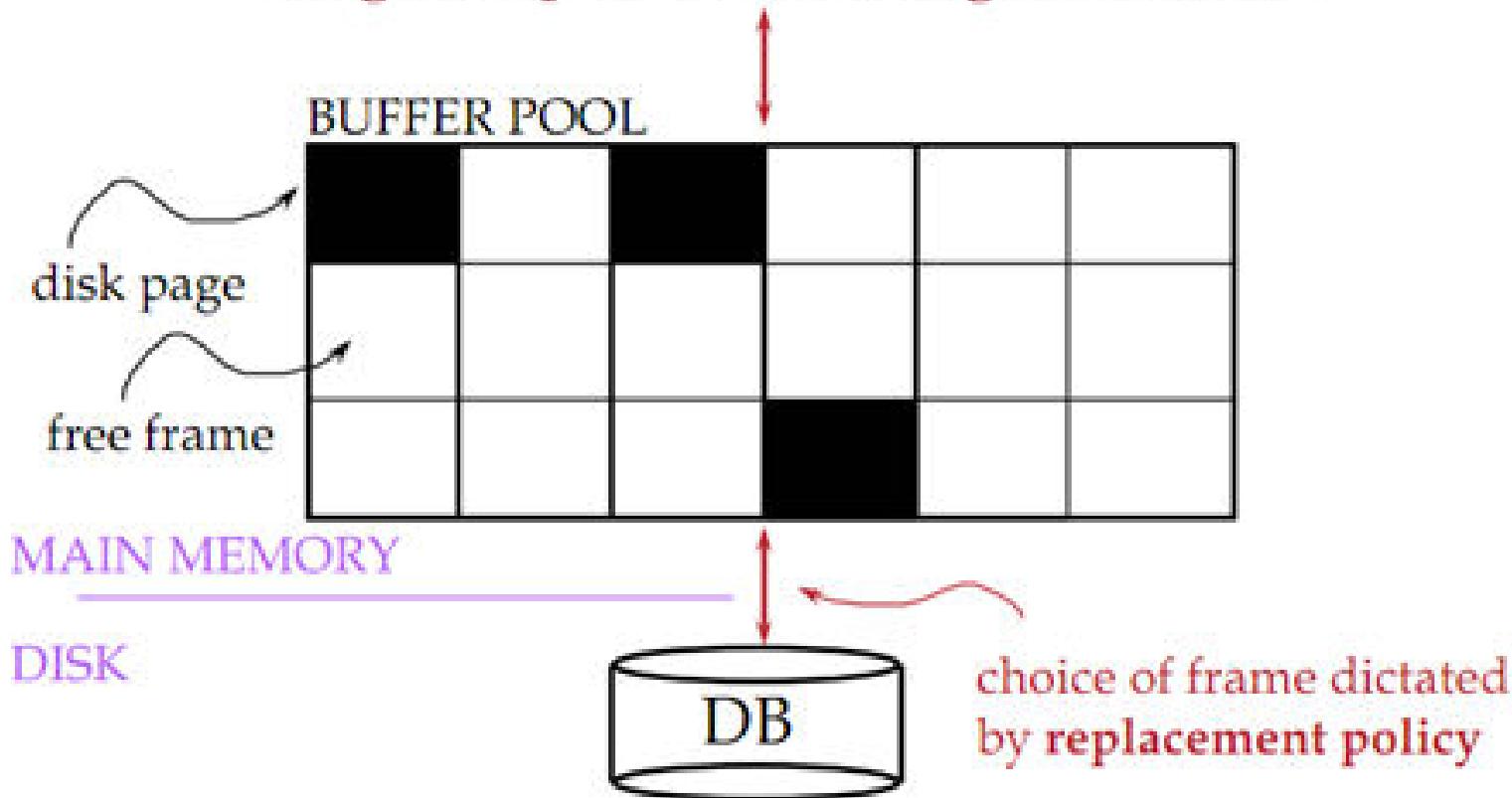


Fig: Concept of Buffer

- Data must be in RAM for DBMS to operate on it!
- Buffer Mgr hides the fact that not all data is in RAM

a. **What are the techniques of buffer management? [2011]**

Buffer manager uses various techniques such as buffer replacement strategy, pinned blocks, buffer cache and so on.

	Data Copying	Offset	Scatter/Gather
Memory BW	High	Low	Low
CPU BW	High	Low	Low
Memory Usage per Layer	Optimal for individual protocol layer because exact amount of space will be allocated	High for application protocol because it must allocate space more than it needs	Compromise, segments are sized depending on requirements
Are Protocol Requirements without copying satisfied?	NA	No, segmentation needs copying	Mostly yes (one copy and segmentation must be done when data leaves node)

8.3 File organization(Sequential, indexed-sequential, hashed file)

File Organization:

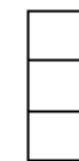
- It refers to the way data is stored in a file.
- It is a method of arranging a file of records on external storage.
- It determines the methods of access, efficiency, flexibility and storage devices to use.
- Components of file:
- Record_id: It is sufficient to physically locate record.
- Index: Data structure that allows us to find the record_id of records with given values in index search key fields.



File/ Table



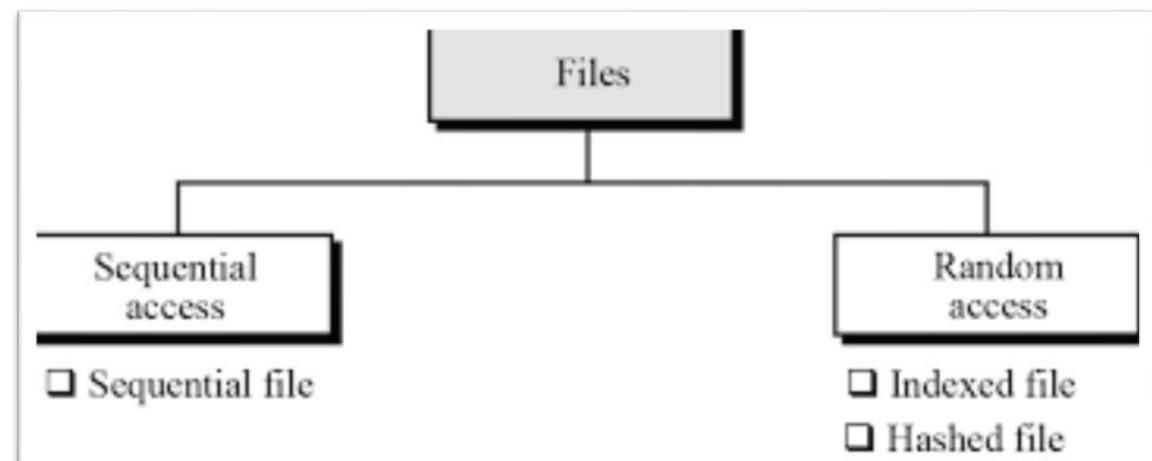
Record/ Row



Field/ Column/ Attributes

Types of file organization:

- Sequential File Organization
 - Pile file organization
 - Sorted File Method
- Indexed Sequential Access Method
- Hash/Direct File Organization
- Other types:
 - Heap File Organization
 - B+ Tree File Organization
 - Cluster File Organization



Sequential File Organization

- Most common form of file structure, where data are stored in sequential manner.
- Types:
 1. Pile file organization
 2. Sorted File Method

A-217	Brighton	750	
A-101	Downtown	500	
A-110	Downtown	600	
A-215	Mianus	700	
A-102	Perryridge	400	
A-201	Perryridge	900	
A-218	Perryridge	700	
A-222	Redwood	700	
A-305	Round Hill	350	

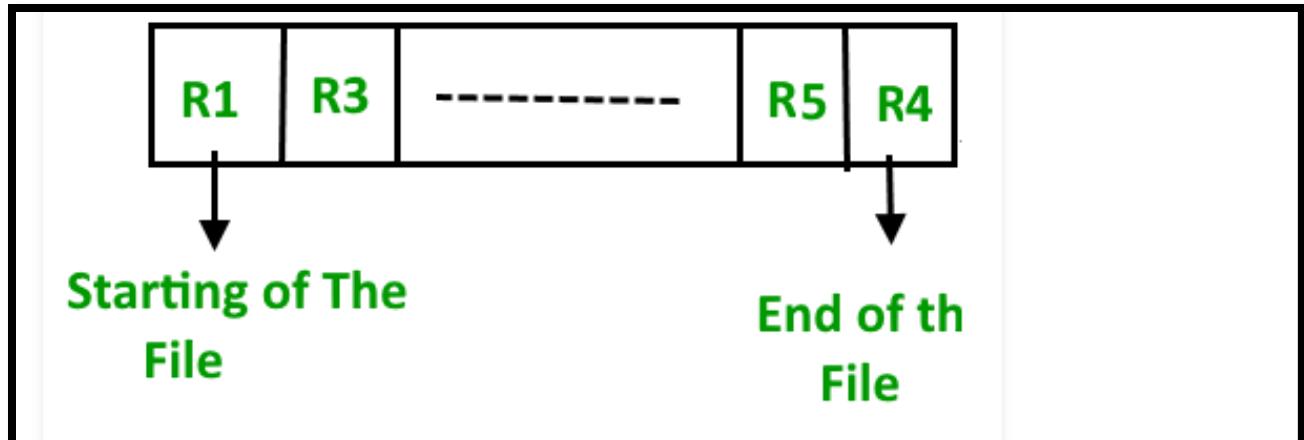


Fig. 4.13 Sequential file for account records

- **Pros and Cons of Sequential File Organization –**
- **Pros –**
 - Fast and efficient method for huge amount of data.
 - Simple design.
 - Files can be easily stored in magnetic tapes i.e cheaper storage mechanism.
- **Cons –**
 - Time wastage as we cannot jump on a particular record that is required, but we have to move in a sequential manner which takes our time.
 - Sorted file method is inefficient as it takes time and space for sorting records.

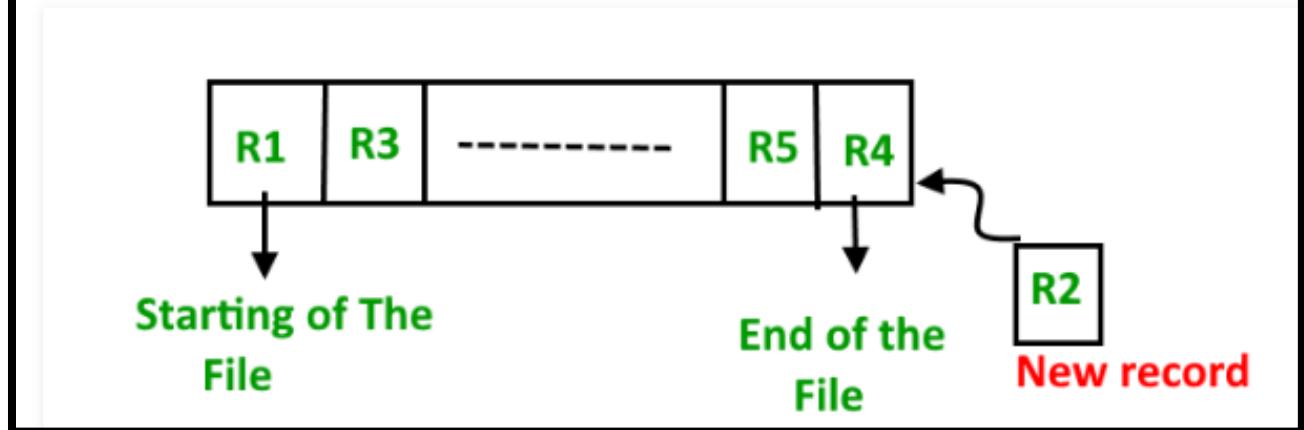
1. Pile file

- This method is quite simple, in which we store the records in a sequence i.e one after other in the order in which they are inserted into the tables.



Insertion of new record –

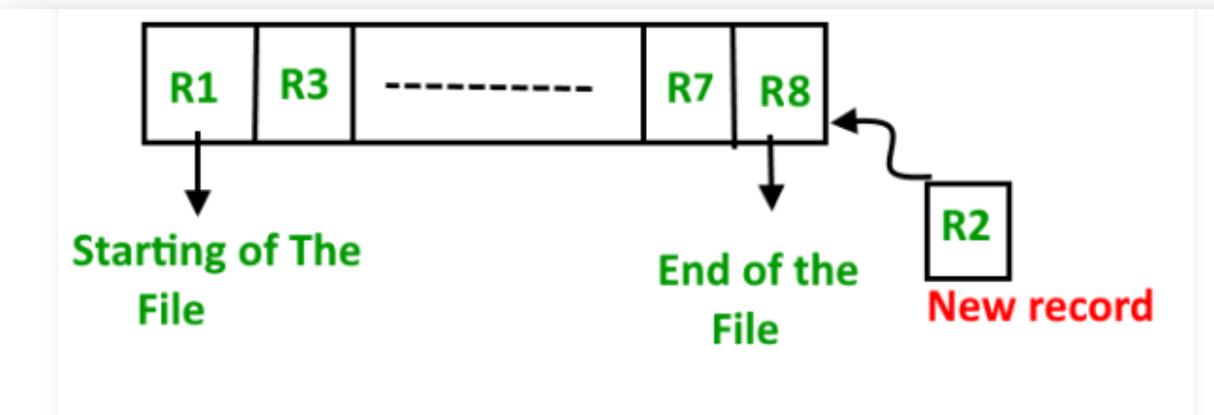
Let the R1, R3 and so on upto R5 and R4 be four records in the sequence. Here, records are nothing but a row in any table. Suppose a new record R2 has to be inserted in the sequence, then it is simply placed at the end of the file.



- **Pros –**
 - Fast and efficient method for huge amount of data.
 - Simple design.
 - Files can be easily stored in magnetic tapes i.e cheaper storage mechanism.
- **Cons –**
 - Time wastage as we cannot jump on a particular record that is required, but we have to move in a sequential manner which takes our time.
 - Sorted file method is inefficient as it takes time and space for sorting records.

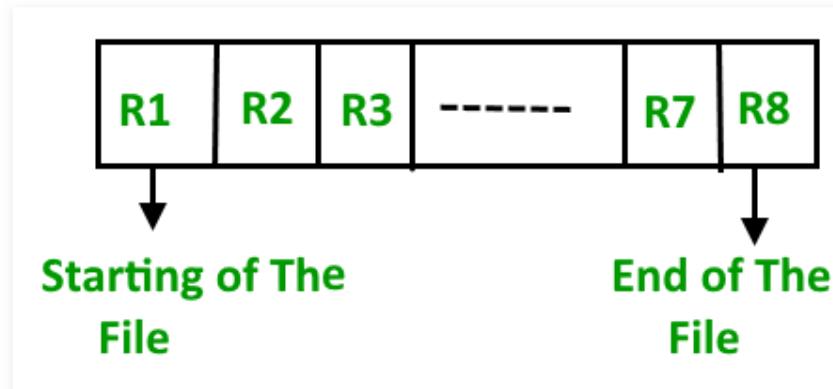
2. Sorted File Method

- In this method, whenever a new record has to be inserted, it is always inserted in a sorted (ascending or descending) manner.
- Sorting of records may be based on any primary key or any other key.



Insertion of new record –

Let us assume that there is a preexisting sorted sequence of four records R1, R3, and so on upto R7 and R8. Suppose a new record R2 has to be inserted in the sequence, then it will be inserted at the end of the file and then it will sort the sequence .



Indexed Sequential Access Method

- When there is need to access records sequentially by some key value and also to access records directly by the same key value, the collection of records may be organized in an effective manner called Indexes Sequential Organization.
- You must be familiar with search process for a word in a language dictionary.
- Almost similar to sequential method only that, an index is used to enable the computer to locate individual records on the storage media.
- **Index** is the key that uniquely identifies a record. If more than one index is present the other ones are called *alternate indexes*. The indexes are created with the file and maintained by the system.
- To search for a word we do not search sequentially.
- Main Concept: Random access of data (no sequential access)

- Types of index:
 - Primary index
 - Single index is used to point unique record.
 - Data file is ordered on a **key field**.
 - Secondary index
 - A secondary index provides a secondary means of accessing a file for which some primary access already exists.
 - The secondary index may be on a field which is a candidate key and has a unique value in every record, or a non-key with duplicate values.

Partial index

bingham	5
callendar	10
	15
..	..

Main file

#	name	tutor	sex
0	ashok	Ebo	m
1	aldham	Ebo	m
2	amdhal	Okl	f
3	azerty	Ebo	m
4			
5	bingham	Okl	f
6	bjalko	Okl	f
7	blantyre	Jhl	m
8	brambell	Ftr	f
9	byzantium	Jhl	m
10	callendar	Ebo	m
..

Block 1

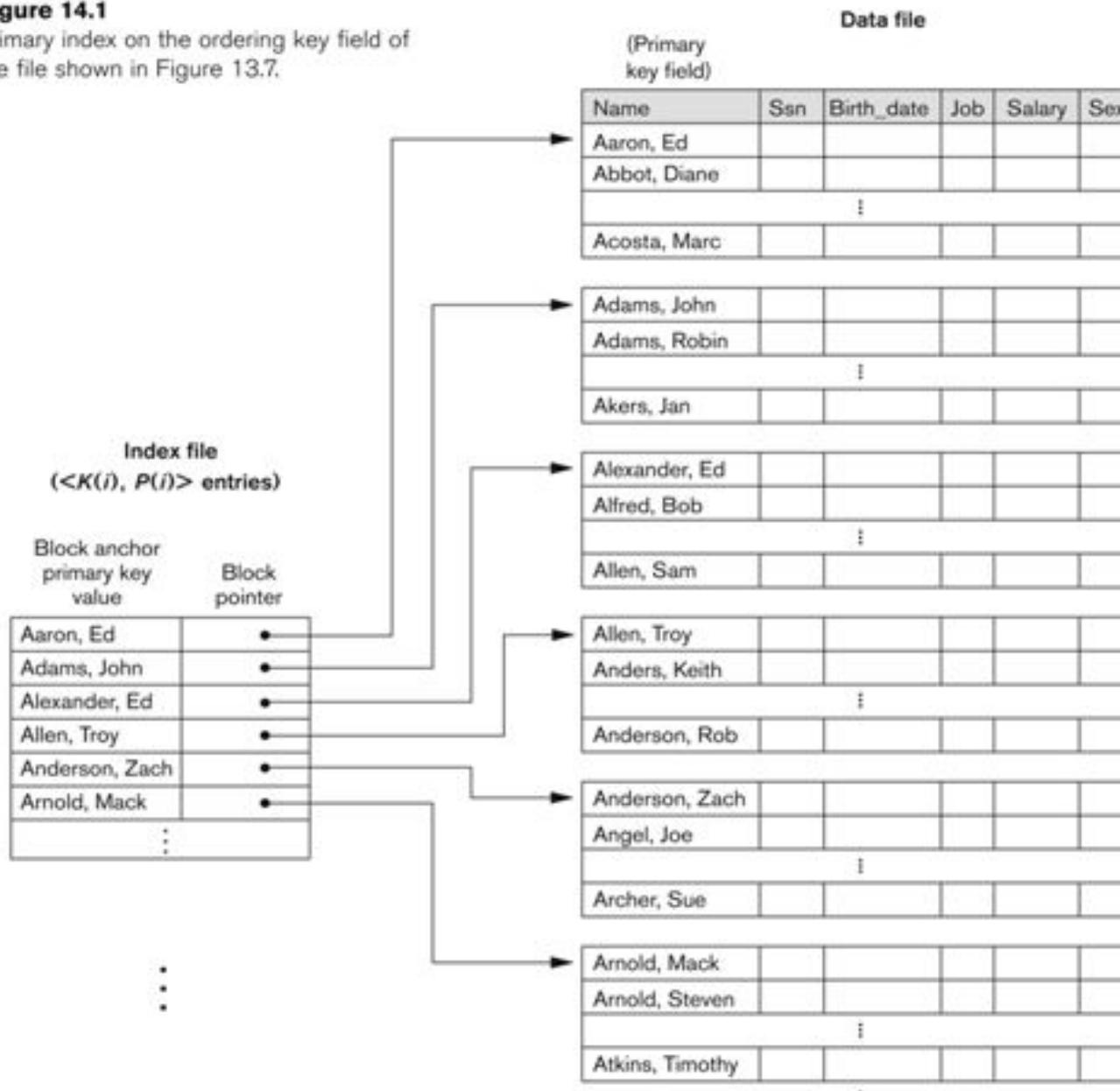
Block 2

Block 3

Fig: Primary Indexed Sequential File Organization

Figure 14.1

Primary index on the ordering key field of the file shown in Figure 13.7.

**Fig: Primary Indexed Sequential File Organization**

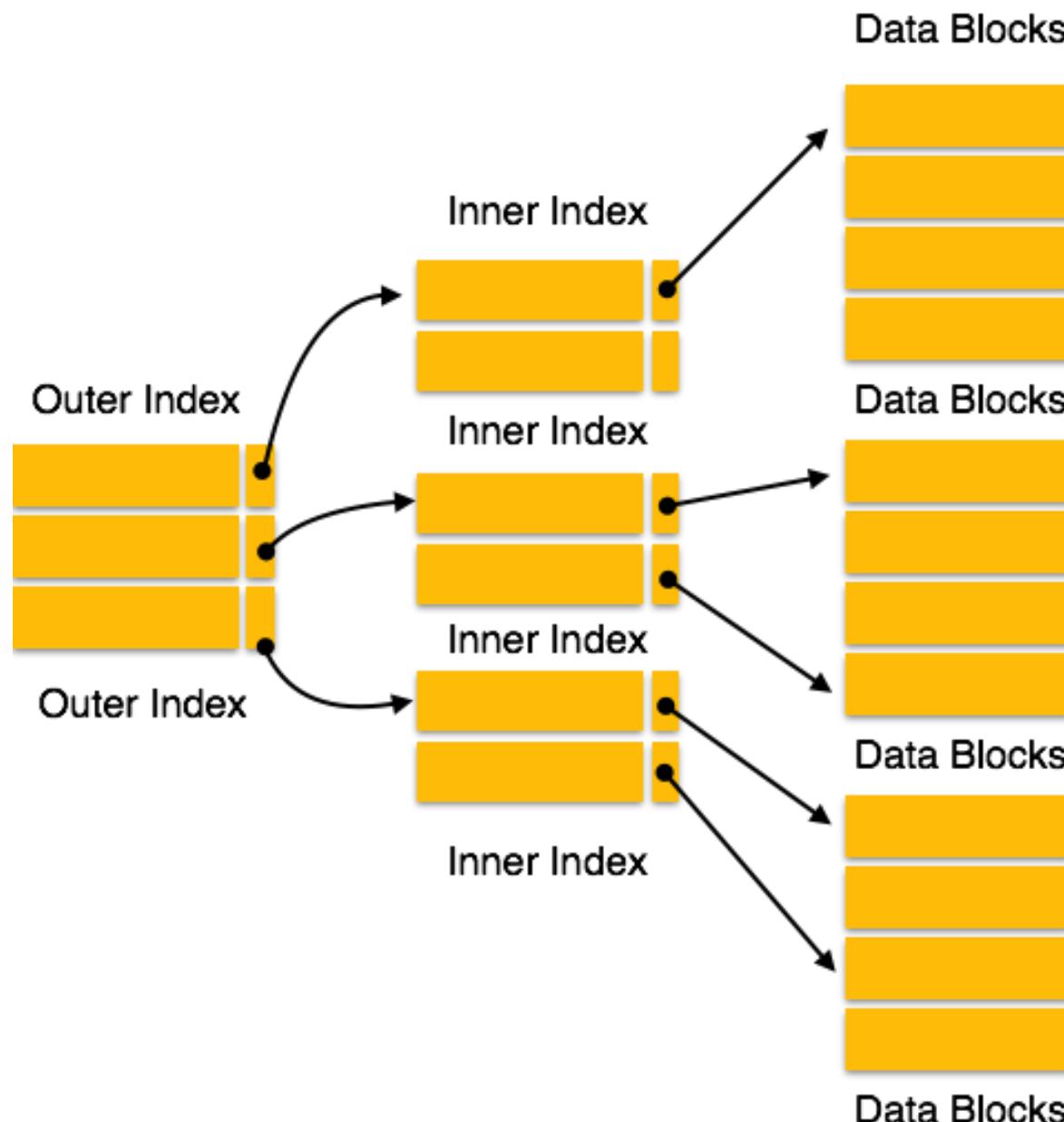
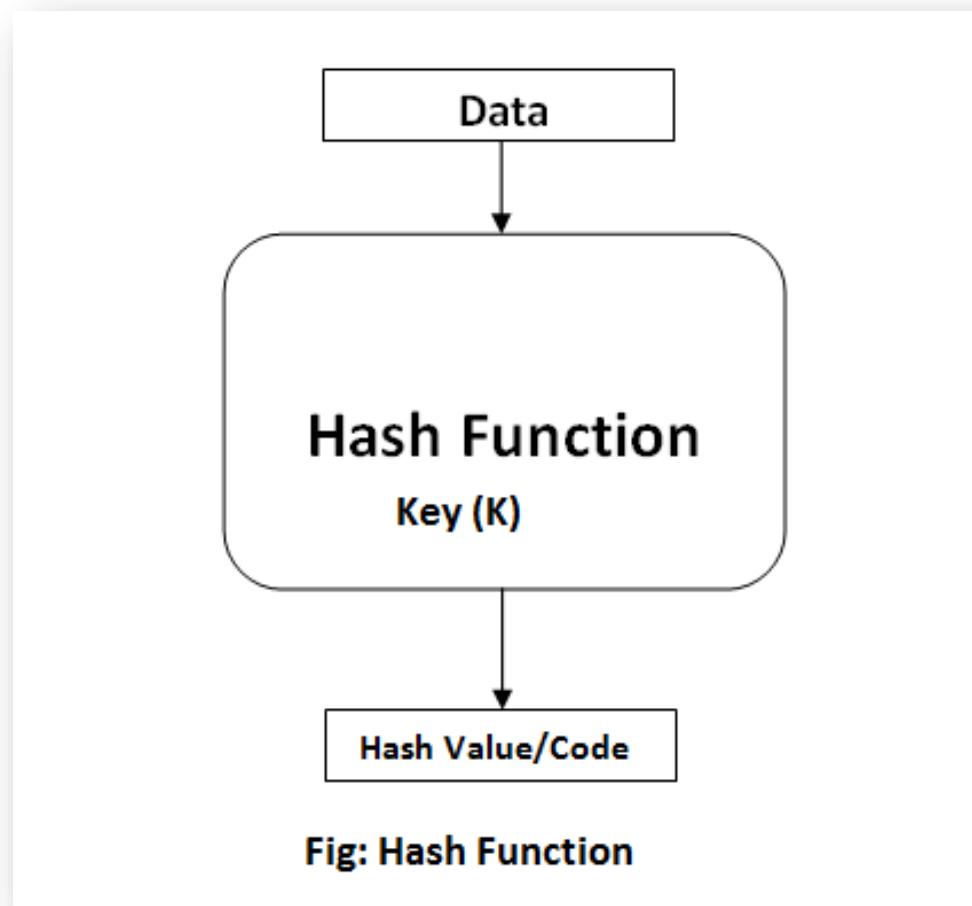


Fig: Secondary Indexed Sequential File Organization

Hashed File Organization

- Hashing involves computing the address of a data item by computing a function on the search key value.
- A hash function h is a function from the set of all search key values K to the set of all bucket addresses B .
- **Hash function is used to calculate the address of the block to store the records.**
- The hash function can be any simple or complex mathematical function.
- **Hashing** is an efficient technique to directly search the location of desired data on the disk without using index structure.
- Data is stored at the data blocks whose address is generated by using hash function.
- The memory location where these records are stored is called as data block or data bucket.

- To perform a lookup on a search key value K_i , we compute h_{K_i} , and search the bucket with that address. If two search keys i and j map to the same address, because $h(K_i)=h(K_j)$, then the bucket at the address obtained will contain records with both search key values.



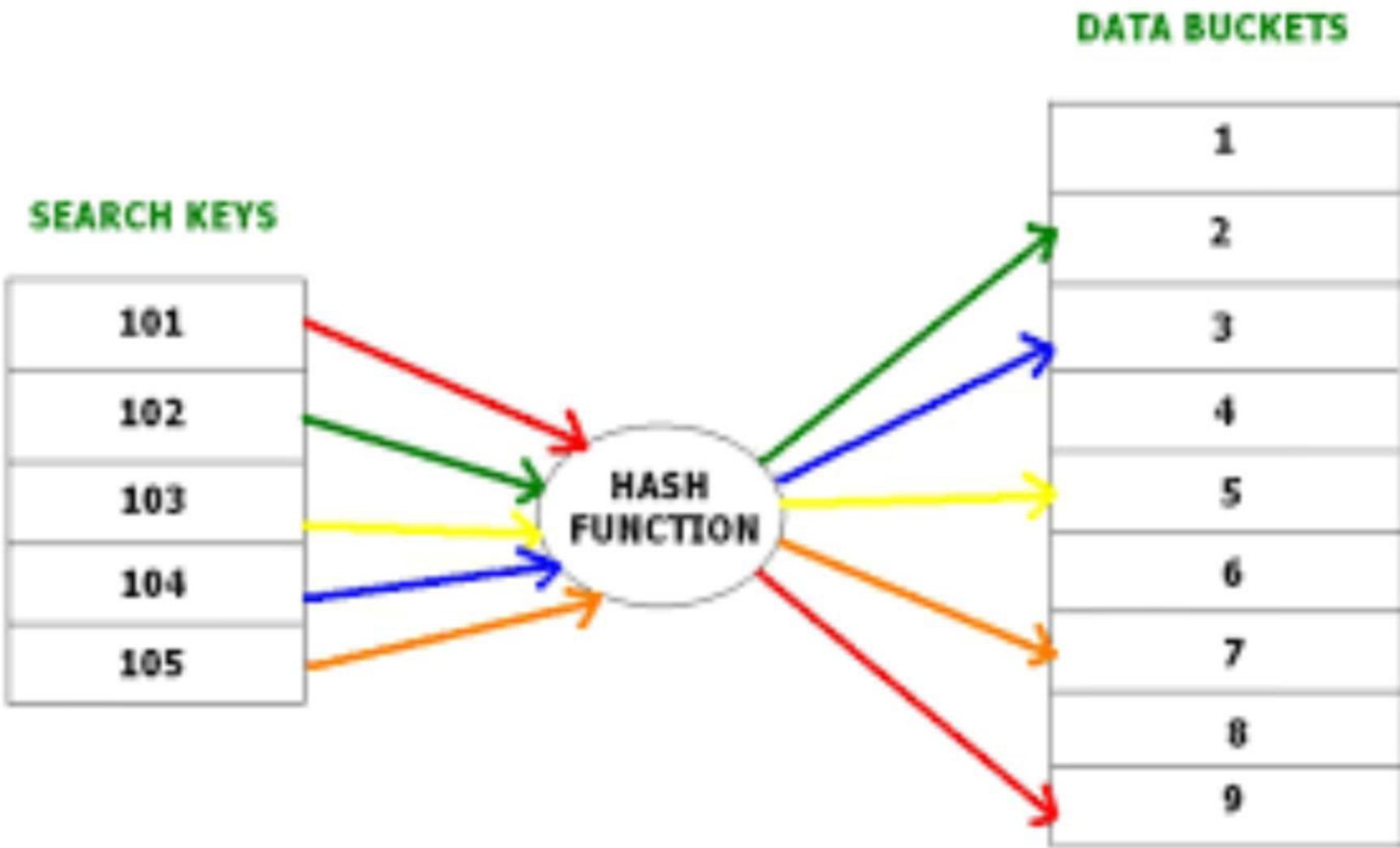


Fig: Hashed File Organization

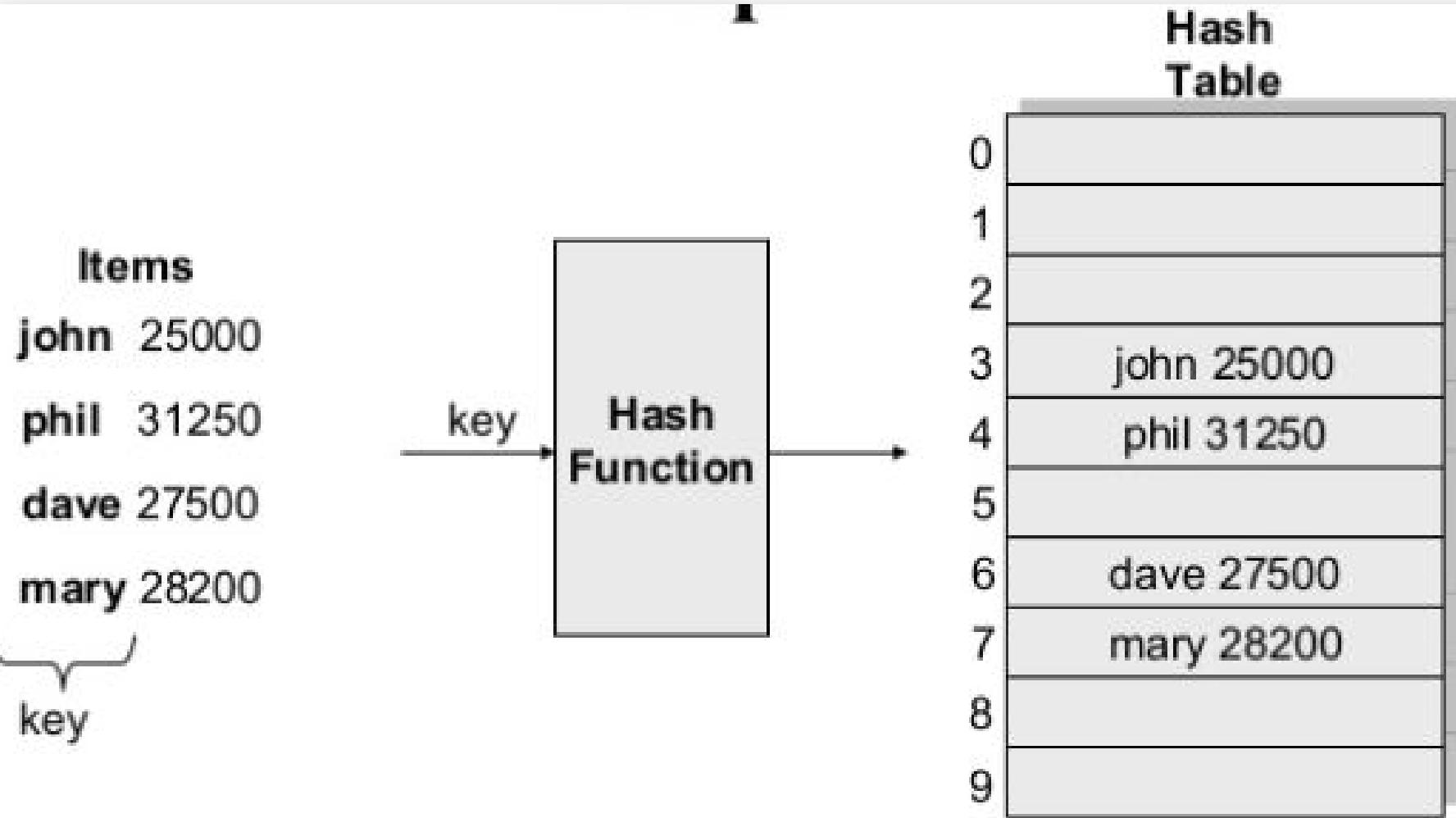


Fig: Hashed File Organization

Advantages of Hash File Organization:

- Records need not be sorted after any of the transaction. Hence the effort of sorting is reduced in this method.
- Since block address is known by hash function, accessing any record is very faster. Similarly updating or deleting a record is also very quick.
- This method can handle multiple transactions as each record is independent of other. i.e.; since there is no dependency on storage location for each record, **multiple records can be accessed at the same time.**
- It is suitable for online transaction systems like online banking, ticket booking system etc.

Disadvantages of Hash File Organization:

- This method may accidentally delete the data.
- Since all the records are randomly stored, they are scattered in the memory. Hence memory is not efficiently used.
- If these hash columns are frequently updated, then the data block address is also changed accordingly. Each update will generate new address. This is also not acceptable.

Other File Organization:

- I) B-tree file organization
- II) B+ tree file organization

B+ Tree File Organization

- It uses a tree like structure to store records in File.
- It uses the concept of Key indexing where the primary key is used to sort the records. For each primary key, an index value is generated and mapped with the record.
- An index of a record is the address of record in the file.
- B+ Tree is very much similar to binary search tree, with the only difference that instead of just two children, it can have more than two.
- All the information is stored in leaf node and the intermediate nodes acts as pointer to the leaf nodes.

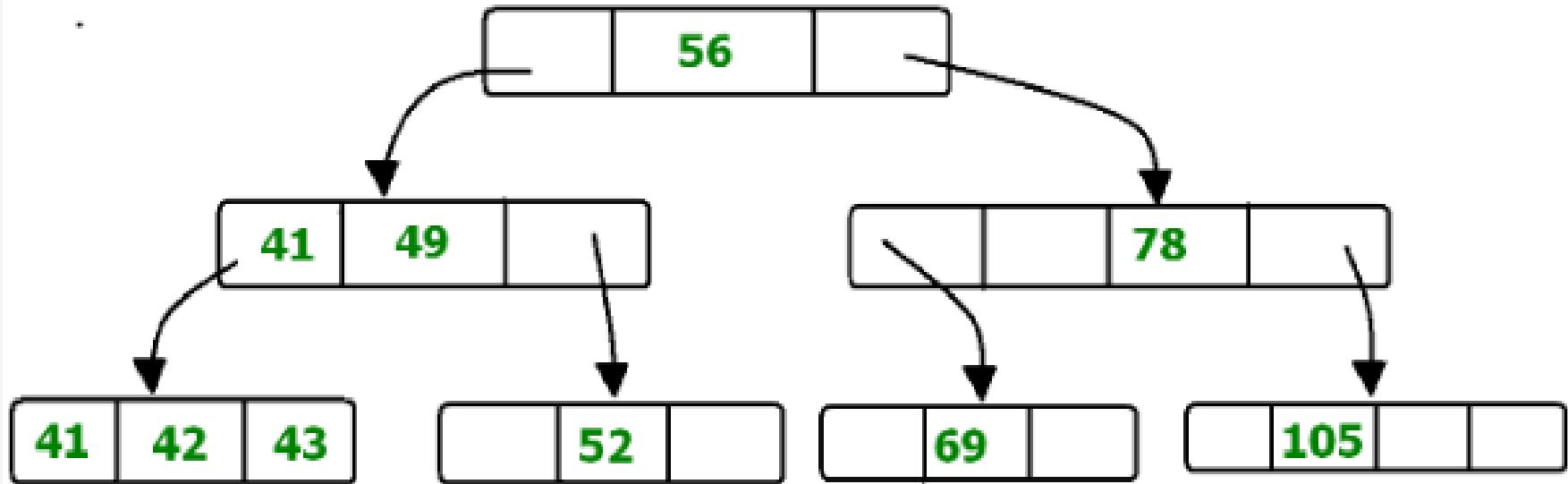


Fig: B+ Tree File Organization

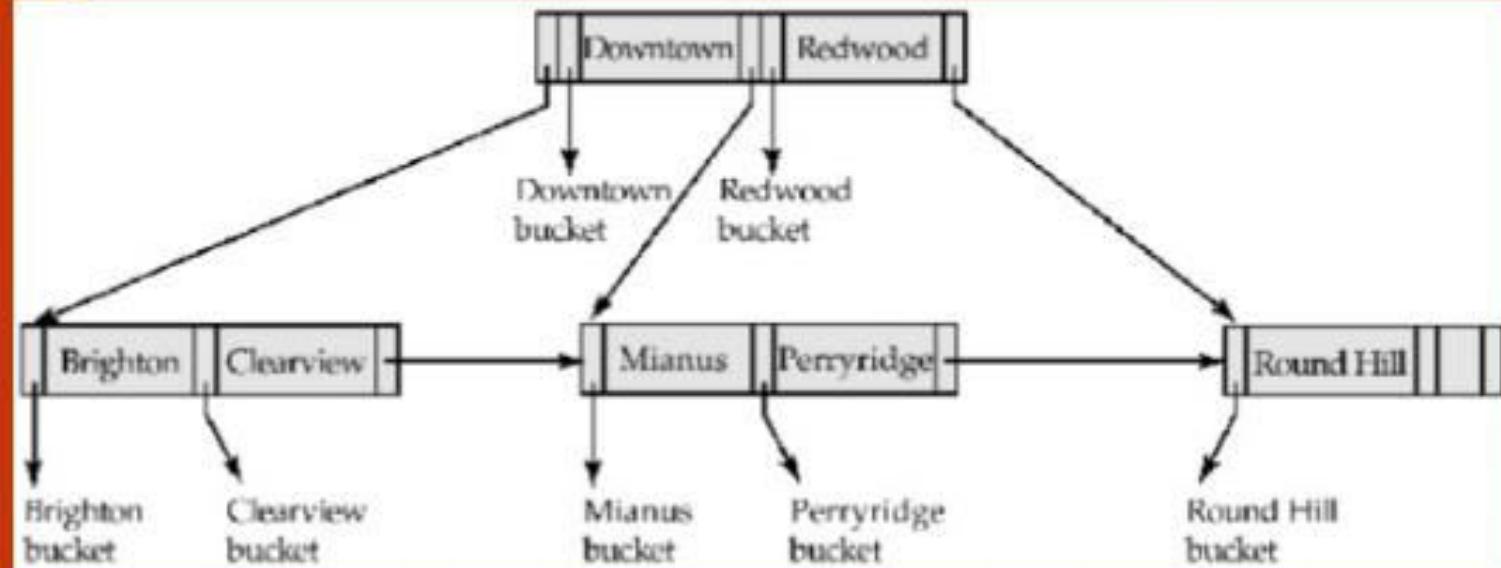
B-tree File Organization

Similar to B+-tree, but B-tree allows search-key values to appear only once; eliminates redundant storage of search keys.

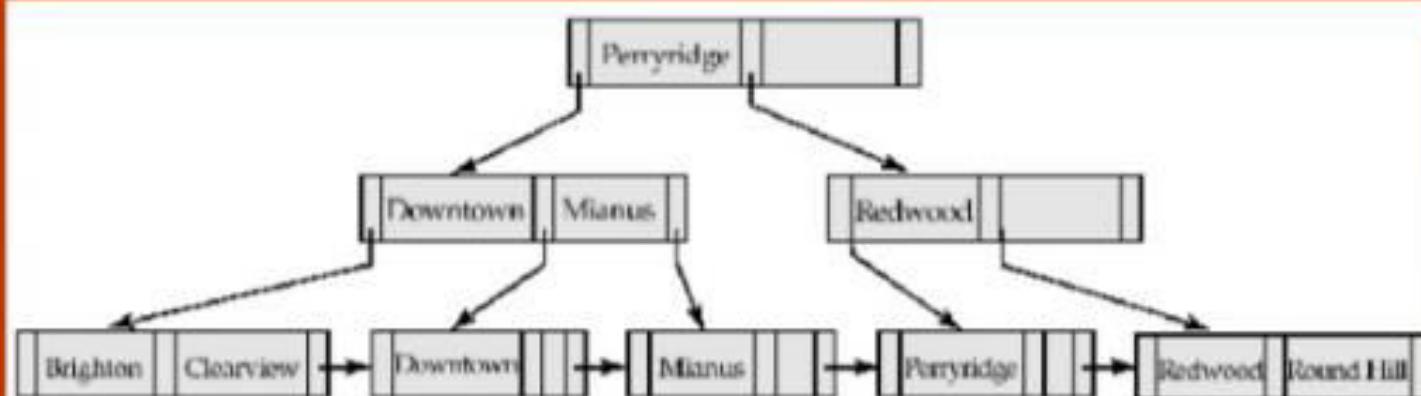
Search keys in nonleaf nodes appear nowhere else in the B-tree; an additional pointer field for each search key in a nonleaf node must be included.

Generalized B-tree leaf node

B-Tree Index File Example



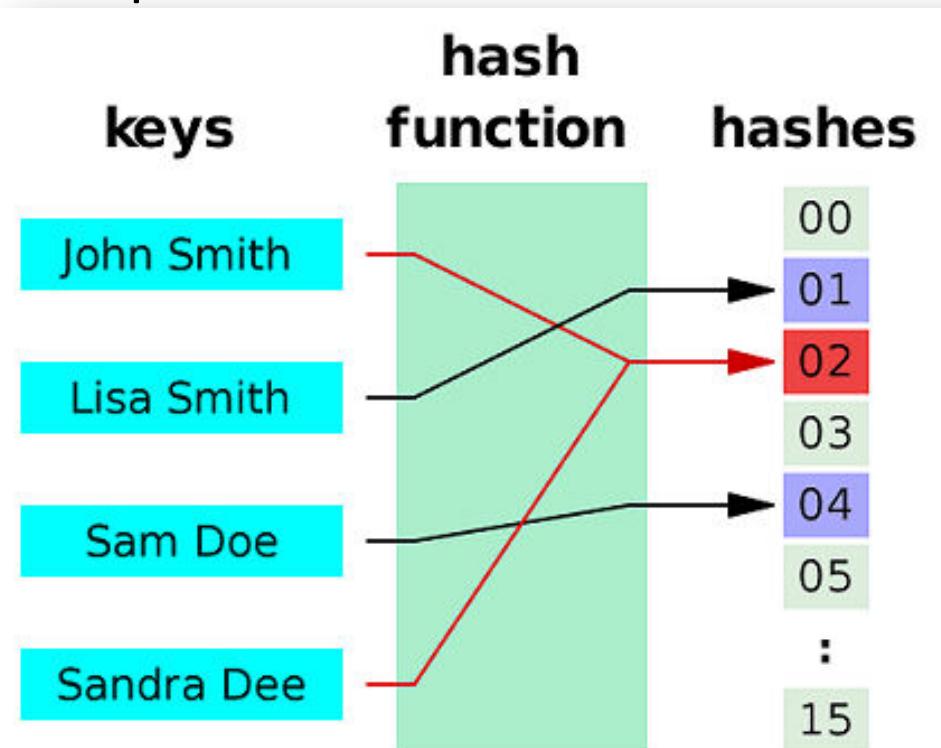
B-tree (above) and B+-tree (below) on same data



8.4 Hash Collision

- In DBMS, hashing is a technique to directly search the location of desired data on the disk without using index structure.
- Hash collision is a state when the resultant hashes from two or more data in the data set, wrongly map the same place in the hash table.
- A situation when the result hash values for two or more data are same i.e both hash values points on same location in table.

Example: Hash Collision



Hash Collision Detection

- When the result hash values for two or more data are same i.e. both hash values points on same location in table, this implies that there is hash collision.

Hash Collision Resolution

- If, when an element is inserted, it hashes to the same value as an already inserted element, then we have a collision and need to resolve it.
- There are several methods for dealing with this:
 - **Separate chaining**
 - **Open addressing**
 - Linear Probing
 - Quadratic Probing
 - Double Hashing

1. Separate Chaining

- A method by which linked lists of values are built in association with each location within the hash table when a collision occurs.
- Collision can be resolved by creating a list of keys that map/ hold same values.

Figure 1: Hash Table: Empty / Collision Occurrence

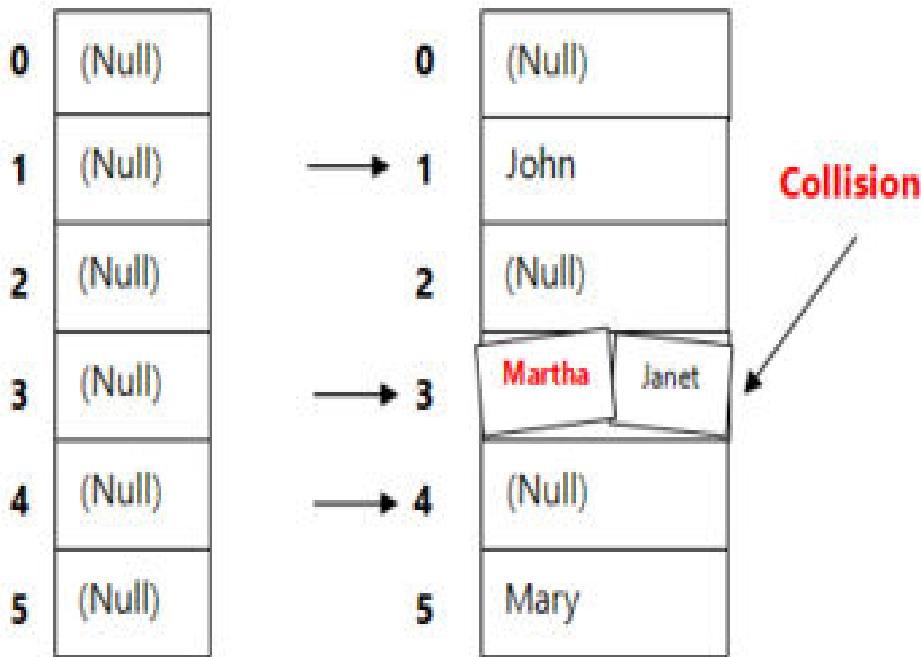
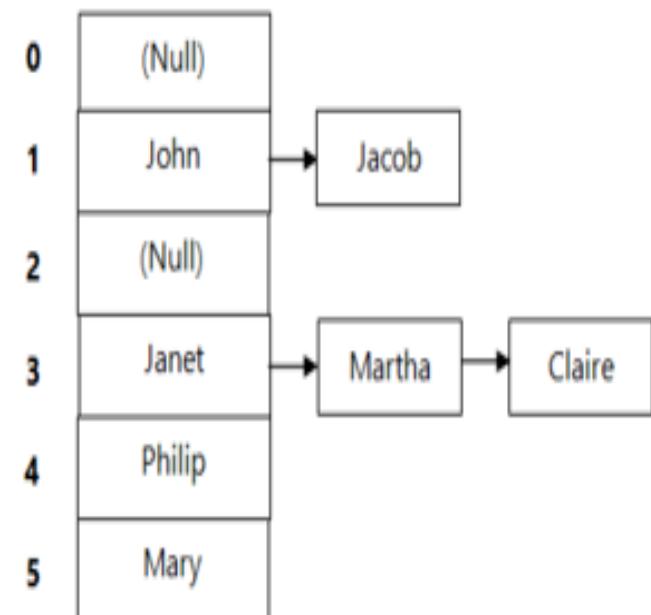


Figure 2: Separate Chaining



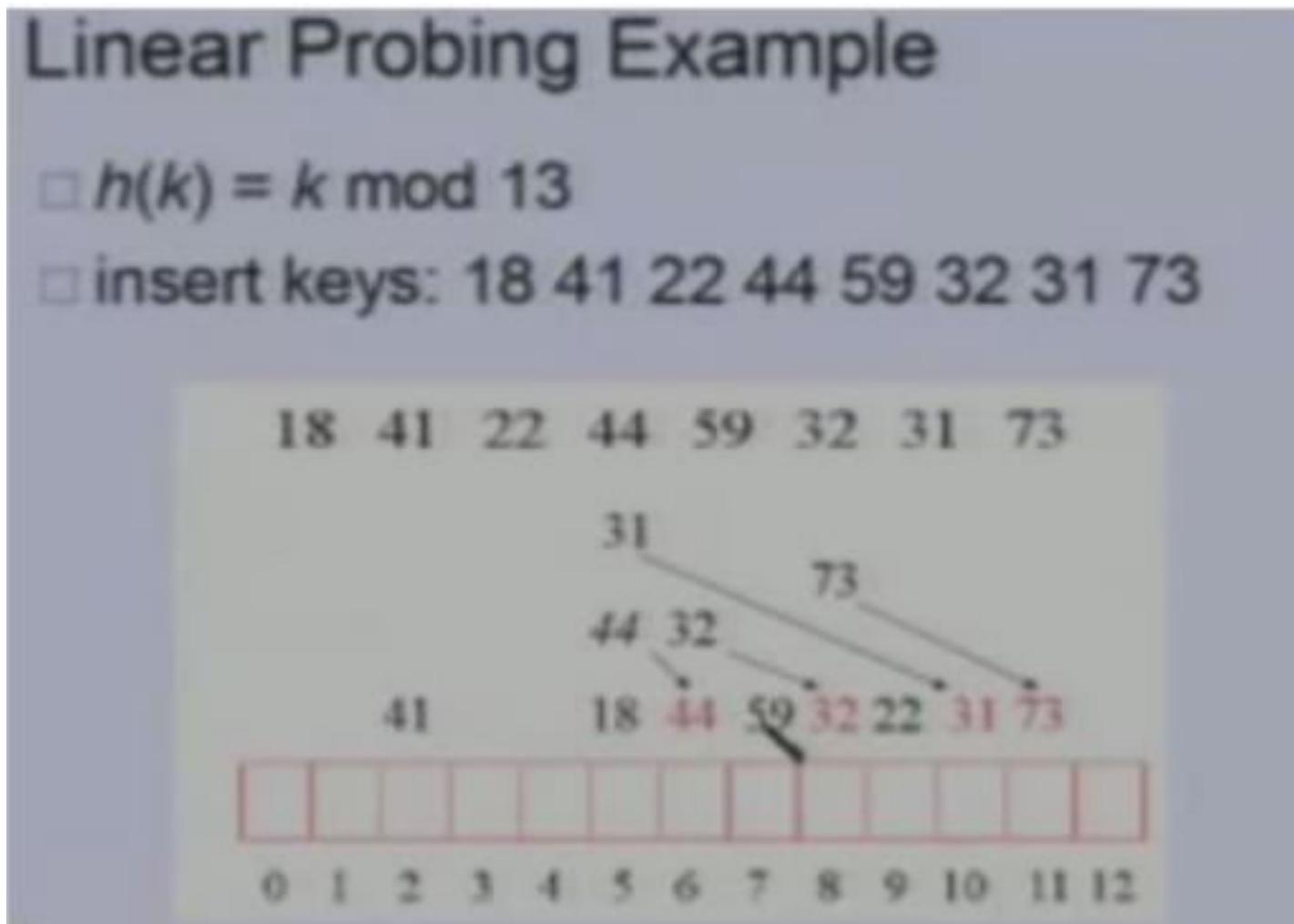
2. Open Hashing

- In Open hashing method, Instead of overwriting older one the next available data block is used to enter the new record.
- Alternative cell is tried until empty cell is found to store same value.
- There are three common collision resolution strategies:
 - Linear Probing
 - Quadratic probing
 - Double hashing

Linear probing

- It scans for next empty slot to store same values.
- Here, K= keys of data, and n= size of table

$$H(K) = K \bmod n$$



Quadratic Probing

- It operates by taking the original hash index and adding successive values of an arbitrary quadratic polynomial until an open slot is found.
- It also use same formula as linear probing. $H(k) = k \bmod \text{table_size}$.
- **If collision occurs then follow steps mention below:**
- Steps:
 1. Set counter=j where ($j=0,1,2,\dots$)
 2. Get hash value, $H(k) = (k+j^2) \bmod \text{table_size}$.
 3. If $H(k)=$ empty then insert value and stop.
 - Else (space is occupied) then do:
 - Increment j by 1.
 - Compute: $H(k) = (k+j^2) \bmod \text{table_size}$.
 - Repeat step-3 until $j=\text{size of table}$.
 4. Hash table is full.
 5. Stop.

Example of Quadratic probing:

size of table = 7

Numbers / keys = 76, 40, 48, 5, 20

48	5	40	40	76		
0	1	2	3	4	5	6

We have,

$$H(k) = k \bmod \text{table-size}$$

- $H(76) = 76 \bmod 7 = 6$
- $H(40) = 40 \bmod 7 = 5$
- $H(48) = 48 \bmod 7 = 6$ (Hash collision)
 - Now, we need to find another empty slot to insert this value '48'

We have, Set: $I=0$

$$\begin{aligned} H(k) &= (k + I^2) \bmod 7 \\ &= (48 + 0^2) \bmod 7 \\ &= 48 \bmod 7 \\ &= 0 \quad \checkmark \end{aligned}$$

$$\begin{aligned} H(k) &= (k + 0)^2 \bmod 7 \\ &= (48 + 0)^2 \bmod 7 \\ &= 6 \quad X \end{aligned}$$

- $H(5) = 5 \bmod 7 = 5$ (Hash collision)

Now, $I=0$

$$H(5) = (5 + 0^2) \bmod 7 = 6 \quad X$$

$$H(5) = (5 + 1^2) \bmod 7 = 9 \bmod 7 = 2 \quad \checkmark$$

- $H(20) = 20 \bmod 7 = 6$ (Hash collision)

Now, $I=0$

$$H(20) = (20 + 0^2) \bmod 7 = 0 \quad X$$

$$= (20 + 1^2) \bmod 7 = 21 \bmod 7 = 3$$

Double Hashing

- It uses hashing function twice to find two possible slots to store same values. Here, k= keys of data.

Here, $H_1(K)$ = position in table where we first check for storage of key

$H_2(K)$ = It checks for another storage location in table if first location is already occupied.

- **Formulas:**
- $H_1(K) = k \bmod n$, where n= size of table
- $H_2(K) = M - (key \bmod M)$, where M= prime number less than size of table

Note: place key/data at that position which is $H_2(K) = (\text{result})$ steps after from first/original hash position.

Double Hashing Example

keys (k) = 37, 90, 45, 17, 43, 55

(n)table-size = 10

We have formula:

$$H_1(k) = k \bmod n = k \% n$$

$$H_2(k) = M - (k \bmod M)$$

where, M = prime number less than table-size.

Now

- $H_1(37) = 37 \% 10 = 7$

- $H_1(90) = 90 \% 10 = 0$

- $H_1(45) = 45 \% 10 = 5$

- $H_1(17) = 17 \% 10 = 7$ (hash collision occurs)

- $H_1(43) = 43 \% 10 = 3$

Now, we need to apply 2nd hash function

$$H_2(k) = M - (k \bmod M)$$

Here, M = prime number less than 10.

= 7 (because it is 1st less prime number than 10)

$$H_2(17) = 7 - (17 \% 7)$$

$$= 7 - 3 = 4$$

(so, we should move 4-steps from 1st/original hash value)

- $H_2(90) = 49 \% 10 = 9$

- $H_2(55) = 55 \% 10 = 5$

Now, apply 2nd hash function:

$$H_2(55) = 7 - (55 \% 7)$$

$$= 7 - 6$$

$$= 1$$

remove only position from 1st hash position, i.e. place at 6th position.

Table

30	0
17	1
43	2
	3
17	4
45	5
55	6
37	7
	8
49	9

This is the location where we place 17.

8.5 Data Dictionary Storage

- It contains metadata i.e. data about the database.
- It is very important as it contains information such as what is in the database, who is allowed to access it, where is the database physically stored etc.
- The data dictionary in general contains information about the following:
 - Names of all the database tables and their schemas.
 - Details about all the tables in the database, such as their owners, their security constraints, when they were created etc.
 - Physical information about the tables such as where they are stored and how.
 - Table constraints such as primary key attributes, foreign key information etc.
 - Information about the database views that are visible.

Types of data dictionary:

- **Active Data Dictionary**
 - If the structure of the database or its specifications change at any point of time, it should be reflected in the data dictionary. This is the responsibility of the database management system in which the data dictionary resides.
 - So, the data dictionary is automatically updated by the database management system when any changes are made in the database. This is known as an active data dictionary as it is self updating.
- **Passive Data Dictionary**
 - This is not as useful or easy to handle as an active data dictionary. A passive data dictionary is maintained separately to the database whose contents are stored in the dictionary.
 - That means that if the database is modified the database dictionary is not automatically updated as in the case of Active Data Dictionary.

"Data dictionary is a repository of information describing the data in the database, that is the **metadata** or the '*data about the data*'."

The DBMS system catalog is one of the fundamental components of the system. Many of the software components that rely on the system catalog for information. For example, the authorization control module uses the system catalog to check whether a user has the necessary authorization to carry out the requested operation.

To perform this check, the system catalog has to store :

- The names of users authorized to use the DBMS.
- The names of the data items in the database.
- The data items that each user can access, the types of access allowed : for example, insert, update, delete or read access.

As another example, the integrity check module uses the system catalog to check that the requested operation satisfies all necessary integrity constraints. To perform this check, the system catalog has to store -

- The names of the data items in the database.
- The types and sizes of the data items.
- The constraint on each data item.

Data dictionary system can be classified as active or passive. An active system is always consistent with the database structure, because it is maintained automatically by the system. On the other hand, a passive system may not be consistent with the database, as changes are initiated by the users. If the data dictionary is part of the DBMS, we refer to it as an integrated data dictionary. A stand alone data dictionary has two its own specialized DBMS.

End of Chapter-8

Chapter-9

Concurrency Control

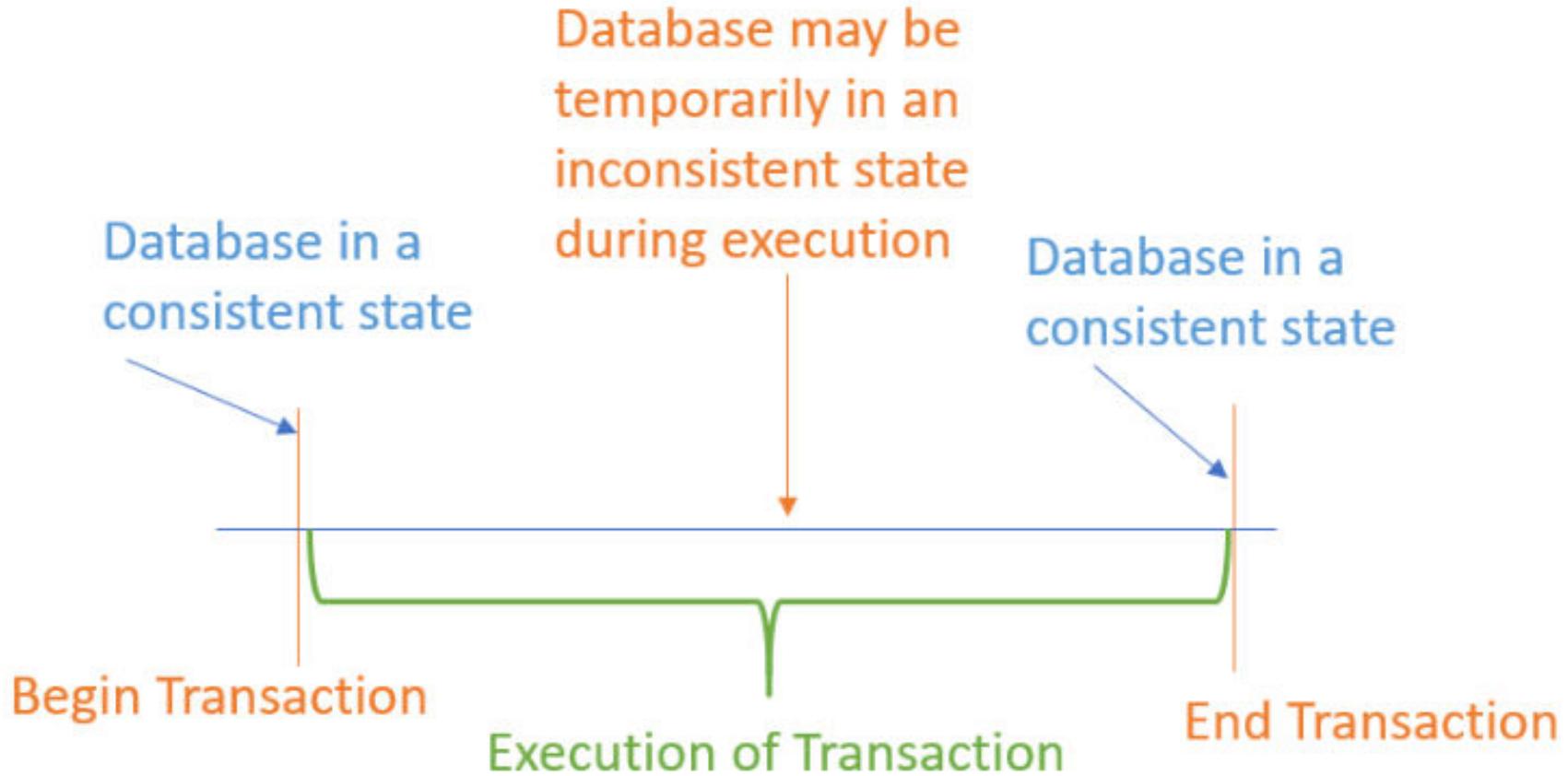
Contents:

- 9.1 Database transaction, transaction properties and states
- 9.2 Needs of concurrency control
- 9.3 Scheduling
- 9.4 Characterizing schedule: Based on serializability and Recoverability
- 9.5 Concurrency control techniques: Lock based, Two-phase locking and Time-stamp based locking
- 9.6 Multiple granularity
- 9.7 Deadlock handling

9.1 Database transaction, transaction properties and states

Transaction:

- A unit of work performed within a DBMS.
- It can be defined as a group of tasks. A single task is the minimum processing unit which cannot be divided further.
- It is a logical unit that is independently executed for data retrieval or updates.
- In relational databases, database transactions must follow ACID property.
- A program consists of begin transaction and end transaction and all operations execute between these two start and end transactions.



➤ Transactions access data using two operations :

- i) *read* (x), which transfers the data item x from the database to a local buffer belonging to the transaction that executed the read operation.
- ii) *write* (x), which transfers the data item x from the local buffer to the transaction that executed the write back to the database.

Example : Let T_i be a transaction that transfers \$ 50 from account A to account B. This transaction can be defined as :

$T_i : \text{read}(A)$

$A := A - 50$

$\text{write}(A)$

$\text{read}(B)$

$B := B + 50$

$\text{write}(B)$

Transaction properties/ ACID Property:

- **Atomicity** : Either all operations of the transaction are reflected properly in the database, or none are.
- **Consistency** : Execution of a transaction in isolation preserves the consistency of the database.
- **Isolation** : Even though multiple transactions execute concurrently, each transaction is unaware of other transactions executing concurrently in the system.
- **Durability** : After a transaction completes successfully, the changes it has made to the database persist, even if there are system failures.

- **Atomicity:**
 - This property states that a transaction must be treated as an atomic unit, that is, either all of its operations are executed or none.
 - There must be no state in a database where a transaction is left partially completed. States should be defined either before the execution of the transaction or after the execution/abortion/failure of the transaction.
- **Consistency :**
 - The database must remain in a consistent state after any transaction.
 - No transaction should have any adverse effect on the data residing in the database.
 - If the database was in a consistent state before the execution of a transaction, it must remain consistent after the execution of the transaction as well.

- **Isolation :**
 - In a database system where more than one transaction are being executed simultaneously and in parallel, the property of isolation states that all the transactions will be carried out and executed as if it is the only transaction in the system.
 - **Concept:** each transaction execute individually, no one effect each other.
 - No transaction will affect the existence of any other transaction.
- **Durability :**
 - The database should be durable enough to hold all its latest updates even if the system fails or restarts. If a transaction updates a chunk of data in a database and commits, then the database will hold the modified data.
 - If a transaction commits but the system fails before the data could be written on to the disk, then that data will be updated once the system springs back into action.

Transaction states

In a database, the transaction can be in one of the following states :

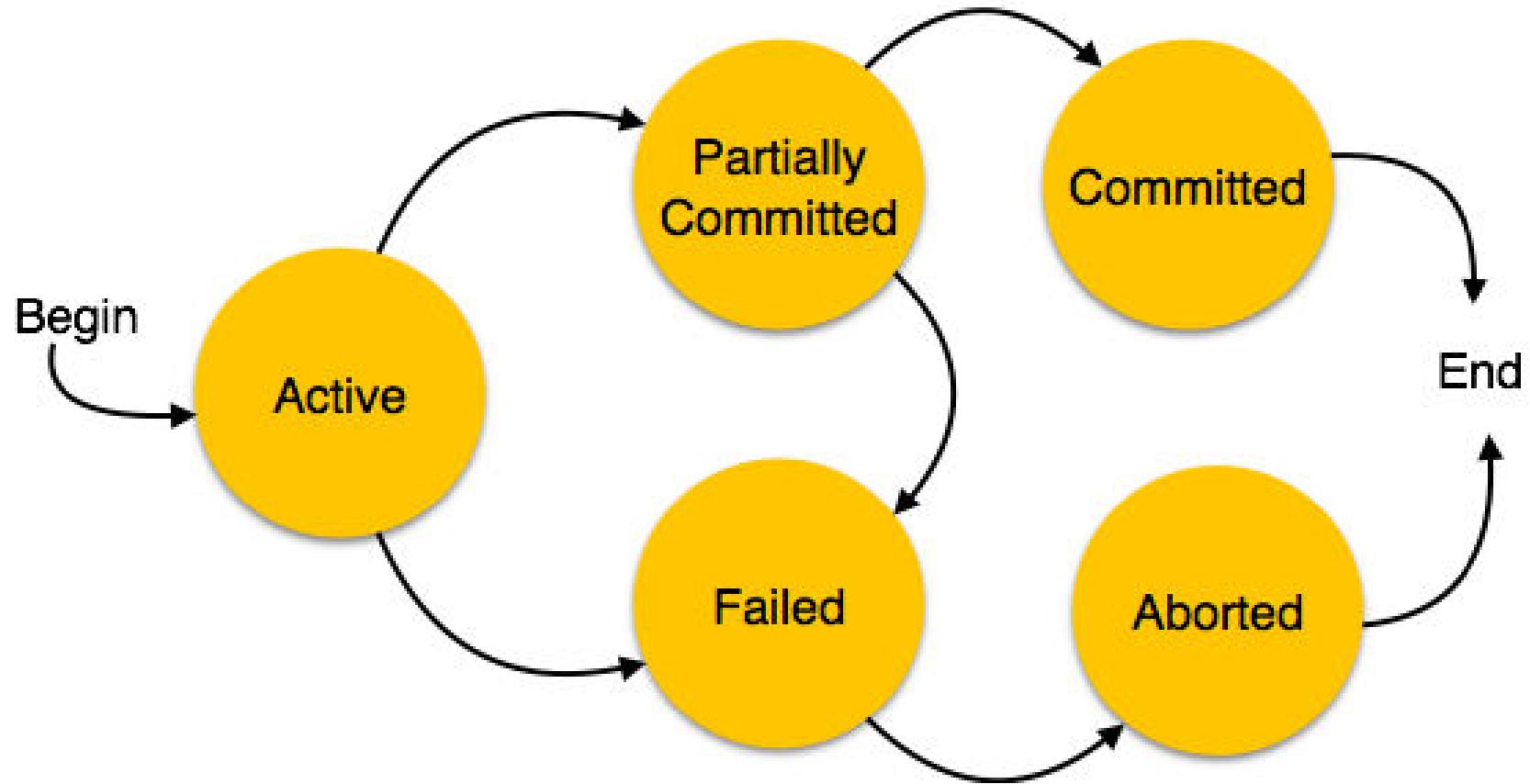


Fig: Stages in transaction execution

1. Active state

- The active state is the first state of every transaction. In this state, the transaction is being executed.
- For example: Insertion or deletion or updating a record is done here. But all the records are still not saved to the database.

2. Partially committed

- In the partially committed state, a transaction executes its final operation, but the data is still not saved to the database.
- In the total mark calculation example, a final display of the total marks step is executed in this state.

3. Committed

- A transaction is said to be in a committed state if it executes all its operations successfully. In this state, all the effects are now permanently saved on the database system.

4. Failed state

- If any of the checks made by the database recovery system fails, then the transaction is said to be in the failed state.
- In the example of total mark calculation, if the database is not able to fire a query to fetch the marks, then the transaction will fail to execute.

5. Aborted

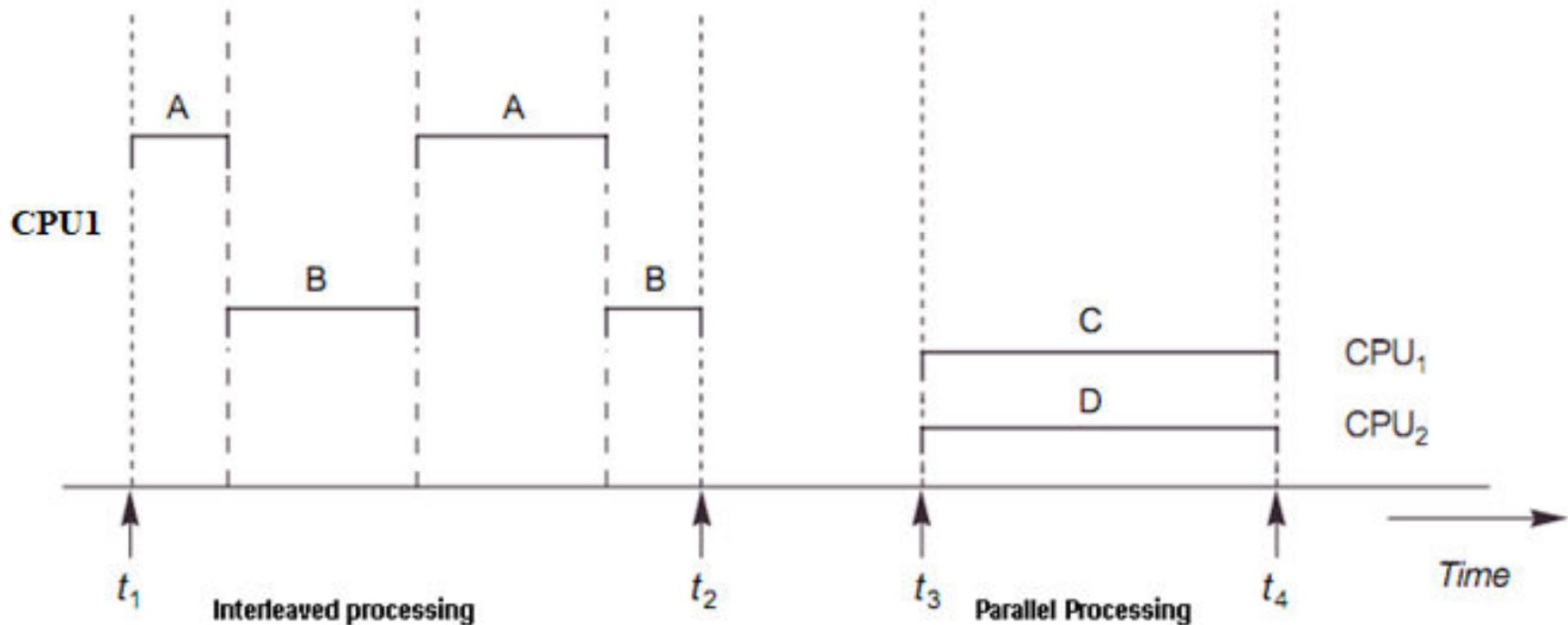
- If any of the checks fail and the transaction has reached a failed state then the database recovery system will make sure that the database is in its previous consistent state. If not then it will abort or roll back the transaction to bring the database into a consistent state.
- If the transaction fails in the middle of the transaction then before executing the transaction, all the executed transactions are rolled back to its consistent state.
- After aborting the transaction, the database recovery module will select one of the two operations:
 - Re-start the transaction
 - Kill the transaction

9.2 Needs of concurrency control

Concurrency:

- It means multiple computations are happening at the same time.
- Multiple queries are executed at the same time.
- Many users access the database at the same time.

Interleaved processing versus parallel processing of concurrent transactions.



- **Advantages of concurrent processing:**
 - Speedup the process
 - Multitasking
 - Multiprocessing, etc.
- **Problems that may arise due to concurrent processing:**
 - Problem in data integrity
 - Problem in data consistency, etc.
- **Note: To reduce those problem there is need of concurrency control techniques.**

Concurrency Control

- Concurrency control is the process of managing simultaneous execution of transactions (such as queries, updates, inserts, deletes and so on) in a multiprocessing database system without having them interfere with one another.
- This property of DBMS allows **many transactions to access the same database at the same time without interfering with each other.**
- **Goal of concurrency control:** To ensure the atomicity (or serializability) of the execution of transactions in a multi-user database environment.
- Concurrency controls mechanism attempt to interleave (parallel) READ and WRITE operations of multiple transactions so that the interleaved execution yields results that are identical to the results of a serial schedule execution.
- **This interleaving creates the impression that the transactions are executing concurrently.**
- Concurrency control is important because the simultaneous execution of transactions over shared database can create several **data integrity and consistency problems.**

Needs of concurrency control

Why Concurrency Control is needed: (to remove following problem)

- **The Lost Update Problem**
 - This occurs when two transactions that access the same database items have their operations interleaved in a way that **makes the value of some database item incorrect**.
- **The Temporary Update (or Dirty Read) Problem**
 - This occurs when one transaction updates a database item and then the transaction fails for some reason.
 - **The updated item is accessed by another transaction before it is changed back to its original value.**
- **The Incorrect Summary Problem**
 - If one transaction is calculating an aggregate summary function on a number of records while other transactions are updating some of these records, the aggregate function may calculate some values before they are updated and others after they are updated.

(a)

	T_1	T_2
Time ↓	<pre>read_item(X); X := X - N; write_item(X); read_item(Y); Y := Y + N; write_item(Y);</pre>	<pre>read_item(X); X := X + M; write_item(X);</pre>

← Item X has an incorrect value because its update by T_1 is lost (overwritten).

Fig. The lost update problem

Example of Last update problem:

T_1	T_2	
Let, $A = 1000$ RCA) $A = A - 50$		
$w(A)$ $A = 950$	RCA) $A = A + 100 = 1100$	\leftarrow To read value of 'A' before written by T_2 . (lost update)
$w(A) = 1100$		T_2 calculates wrong value because accurate value is $A = 950 + 100 = 1050$ (T_2 must read after write operation performed by T_1)

Time	T ₁	T ₂	Comments
1	read_item(X);		Read operation is performed in T ₁ at time step 1.
2	X:=X-N;		Value of data item X is modified.
3		read_item(X);	Read in value of X (Which value is read in?)
4		X:=X+M	Value of data item X is modified.
5	write_item(X);		Write operation is performed in T ₁ .
6	read_item(Y);		Read operation is performed in T ₁ .
7		write_item(X);	Write data item X to database (What value is written in?)
8	Y:=Y+N;		Value of data item is modified.
9	write_item(Y);		Write data item Y to database.

Example:

Time	T ₁	T ₂	Value
1	read_item(X);		X = 80
2	X:=X-N;		X = 80 - 5 = 75 (which is not written into database)
3		read_item(X);	X = 80 (T ₂ still reads in the original value of X, the updated value of X is lost.) (Update is lost)
4		X:=X+M	X = 80 + 4 = 84
5	write_item(X);		X = 75 is written into database
6	read_item(Y);		
7		write_item(X);	X = 84 over writes X = 75, a wrong record is written in database
8	Y:=Y+N;		
9	write_item(Y);		

(b)

T_1	T_2
<code>read_item(X); $X := X - N;$ write_item(X);</code>	
Time ↓	<code>read_item(X); $X := X + M;$ write_item(X);</code>

← Transaction T_1 fails and must change the value of X back to its old value; meanwhile T_2 has read the temporary incorrect value of X .

Fig. The temporary update problem

Example of Temporary update / Dirty read problem:

T_1	T_2	
<p>Let, $A=100$, $R(A)$ $A = A+20$ $W(A)$</p> <p>$R(B)$</p> <p>power failure</p>	<p>$R(A)$ $A = A+20$ $W(A)$</p> <p>Commit</p>	<p>Here, $A=100$ $T_1 : A=100+20$ $=120$</p> <p>$T_2 : A=W(A)$ $A=120+10$ $=130$</p> <p>$W(A)=130$ Commit (C.stored in DB)</p>

↳ Transaction is discarded
 ↳ rollback to previous state.

Then, $T_1 \Rightarrow A=100$
 $A=100+20=120$
 $W(A)=120$

$T_2 \Rightarrow$ Previous task performed by T_2 is stored
 in DB.

(this can't be rollback)

i.e. T_2 will again add 10 in 120.
 Result \Rightarrow inaccurate.

Time	T ₁	T ₂	Comment
1	read_item(X);		
2	X:=X-N;		
3	write_item(X);		X is temporarily updated
4		read_item(X);	
5		X:=X+M	
6		write_item(X);	
7	read_item(Y);		
...	
	ROLLBACK		T ₁ fails and must change the value of X back to its old value; meanwhile T ₂ has read the temporary incorrect value of X

(c)

T_1	T_3
	$sum := 0;$ $read_item(A);$ $sum := sum + A;$ \vdots \vdots
$read_item(X);$ $X := X - N;$ $write_item(X);$	$read_item(X);$ $sum := sum + X;$ $read_item(Y);$ $sum := sum + Y;$
$read_item(Y);$ $Y := Y + N;$ $write_item(Y);$	

T_3 reads X after N is subtracted and reads Y before N is added; a wrong summary is the result (off by N). 

Fig. The incorrect summary problem

Example of Incorrect Summary Problem

T_1	T_2	
	Sum = 0	Let,
	R(A)	$A = 100 \neq B = 200$
	Sum = sum + A	$T_2 : \text{sum} = 0 + 100 = 100$
	R(B)	$\text{Sum} = 100 + 100 = 200$
	Sum = sum + B	
	B = B + 20	
$\rightarrow R(B)$		
$B = B + 100$		
$W(B)$		

Here, value of 'B' is read before update i.e. before write operation, which leads to the incorrect result summary.

$$T_1 : R(B) = 200$$

$$B = 200 + 100 = 300$$

$$\therefore W(B) = 300$$

But, actual result must be: $R(B) = 200 + 10 = 210$

$$B = 210 + 100 = 310$$

$$\therefore W(B) = 310$$

=

Time	T_1	T_3	Comment
1		$\text{sum}:=0;$	
2		$\text{read_item}(A);$	
3		$\text{sum}:=\text{sum}+A;$	
4	$\text{read_item}(X);$		
5	$X:=X-N;$		
6	$\text{write_item}(X);$		
7		$\text{read_item}(X);$	T_3 reads X after N is subtracted from it
8		$\text{sum}:=\text{sum}+X;$	
9		$\text{read_item}(Y);$	T_3 reads Y before N is added to it
10		$\text{sum}:=\text{sum}+Y;$	A wrong summary is resulted (off by N)
11	$\text{read_item}(Y);$		
12	$Y:=Y+N;$		
13	$\text{write_item}(Y);$		

9.3 Scheduling

- Schedule a sequence of the operations by a set of concurrent transactions that preserves the order of operations in each of the individual transactions.
- A chronological execution sequence of a transaction is called a schedule. A schedule can have many transactions in it, each comprising of a number of instructions/tasks.
- It must preserves the order in which the instructions appear in each individual transaction.
- Scheduling is the process of making schedule.

Example of schedule:



T ₁	T ₂
read(A); A := A - N; write(A); read(B); B := B + N; write(B);	read(A); A := A + M; write(A);

- **Two operations in a schedule are said to conflict if they satisfy all three of the following conditions:**
 - They belong to different transactions.
 - They access the same item X
 - At least one of the operations is a `write_item(X)`.
- **A schedule S of n transactions T_1, T_2, \dots, T_n is said to be a complete schedule if the following conditions hold:**
 - The operations in S are exactly those operations in T_1, T_2, \dots, T_n including a commit or abort operation as the last operation for each transaction in the schedule.
 - For any pair of operations from the same transaction T_i , their order of appearance in S is the same as their order of appearance in T_i .
 - For any two conflicting operations, one of the two must occur before the other in the schedule.

Consider the following two transactions:

T1: read item(X)

X := X + 10;

write_item(X)

T2: read_item(X)

X := X * 1.1;

write_item(X)

- If initially X=50 and T1 then T2 is “effective” order, X would be 66.
 - $66 = (50+10) * 1.1$
- If initially X=50 and T2 then T1 is “effective” order, X would be 65.
 - $65 = 50 * 1.1 + 50$
- Both are valid.

9.4 Characterizing schedule: Based on serializability and Recoverability

1. Characterizing schedule: Based on serializability

Serializability:

- It is the classical concurrency scheme.
- It checks for correctness of schedule.
- It helps to ensure that a schedule that is going to perform concurrent transaction is equivalent to one that executes the transactions serially in some order.

Example: Let pick schedule: R2,R1,W1,W2

Now, compare this schedule with a serial schedule. If this schedule is equivalent to some serial schedule, then this schedule is known as **serial serializable schedule (serial serializability)**.

Definition: (Serializable schedule)

- A schedule ‘s’ of n-transactions is serializable if it is equivalent to some serial schedule of the same n-transaction.

Conflicting Transactions/ Operations

Operations are said to be conflicting if:

- They belongs to different transactions
- Access to same DB item
- At least one of then is a write operation, such as: RW, RW, WW.

Instructions l_i and l_j of transactions T_i and T_j , respectively, **conflict** if and only if there exists some item Q accessed by both l_i and l_j , and at least one of these instructions wrote Q .

1. $l_i = \text{read}(Q)$, $l_j = \text{read}(Q)$. l_i and l_j don't conflict.
2. $l_i = \text{read}(Q)$, $l_j = \text{write}(Q)$. They conflict.
3. $l_i = \text{write}(Q)$, $l_j = \text{read}(Q)$. They conflict
4. $l_i = \text{write}(Q)$, $l_j = \text{write}(Q)$. They conflict

Equivalent schedule: Two schedules are said to be equivalent if they produce the same final DB state.

2-major concepts in serializability:

- I. **Conflict serializability:** if the schedule is conflict equivalent to some serial schedule then it is known as conflict serializable.
- II. **View serializability:** If the schedule is view equivalent to some serial schedule then it is known as view serializable.

Conflict Equivalent: Two schedules are said to be conflict equivalent if the order of any two conflicting operations is same in both schedules.

View equivalent:

- Schedules S1 and S2 are called view equivalent if the following three conditions hold true for them:
 - For each data item X, if transaction T_i reads X from the database initially in schedule S1, then in schedule S2 also, T_i must perform the initial read of X from the database.
 - If transaction T_i reads a data item that has been updated by the transaction T_j in schedule S1, then in schedule S2 also, transaction T_i must read the same data item that has been updated by the transaction T_j .
 - For each data item X, if X has been updated at last by transaction T_i in schedule S1, then in schedule S2 also, X must be updated at last by transaction T_i .

Conflict Serializability (Cont.)

- Schedule 3 can be transformed into Schedule 6, a serial schedule where T_2 follows T_1 , by series of swaps of non-conflicting instructions.
 - Therefore Schedule 3 is conflict serializable.

T_1	T_2
read(A)	
write(A)	read(A)
	write(A)
read(B)	
write(B)	
	read(B)
	write(B)

Schedule 3

T_1	T_2
read(A)	
write(A)	
read(B)	
write(B)	
	read(A)
	write(A)
	read(B)
	write(B)

Schedule 6

View Serializability (Cont.)

- A schedule S is **view serializable** if it is view equivalent to a serial schedule.
- Every conflict serializable schedule is also view serializable.
- Below is a schedule which is view-serializable but *not* conflict serializable.

T_3	T_4	T_6
read(Q)	write(Q)	
write(Q)		write(Q)

2. Characterizing schedule: Based on Recoverability

Classification of schedule in recoverability:

- i. **Recoverable schedule:** If T2 reads a data item written by T1 commit operation then T1 should appear before commit operation of T2.
- ii. **Cascading Rollback schedule:** A single transaction failure leads to a series of transaction failures.
- iii. **Cascadeless schedule:** If T2 reads a data item written by T1 commit operation, then T1 should appear before read operation of T2.
- iv. **Strict schedule:** If write operation of T1 precedes a conflicting operation of T2 (read/write), then commit event of T1 also precedes the conflicting operation of T2.

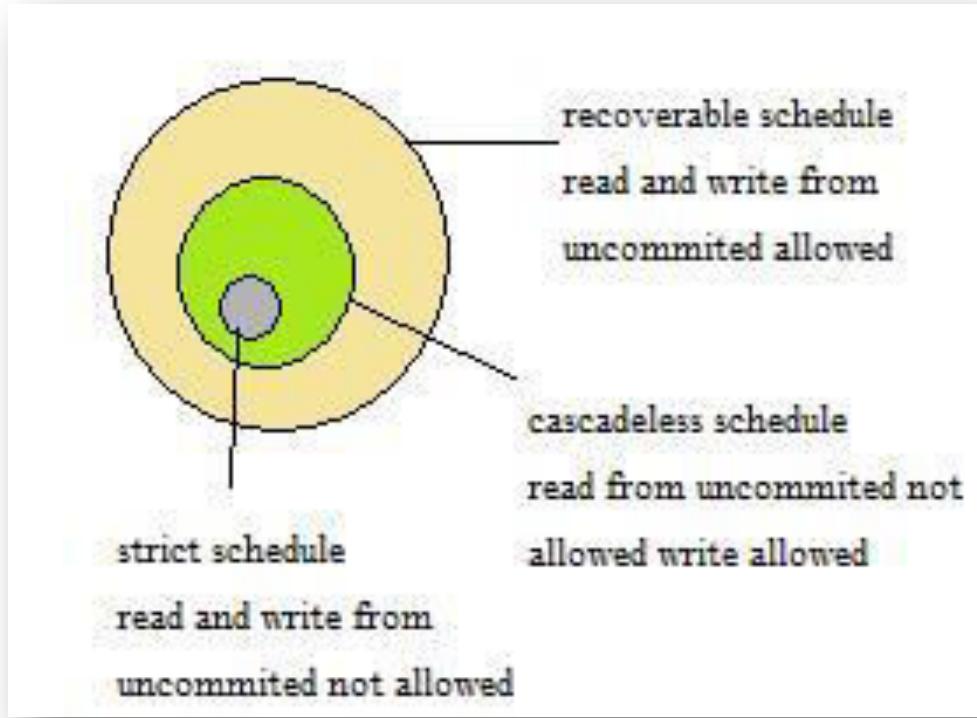


Fig: Relationship between different types of schedule recoverability

Precedence Graph

- It is used to test conflict serializability.
- A schedule is conflict serializable if and only if its precedence graph is acyclic (no cycle formation).

Precedence Graph Example

- No cycles: conflict serialisable schedule

T1 reads X before T2 writes X and
T1 writes X before T2 reads X and
T1 writes X before T2 writes X



T1	T2
Read(X) Write(X)	Read(X) Write(X)

Precedence Graph Example

- The lost update schedule has the precedence graph:

T1 Write(X) followed by T2 Write(X)



T2 Read(X) followed by T1 Write(X)

T1	T2
Read(X) $X = X - 5$ Write(X) COMMIT	Read(X) $X = X + 5$ Write(X) COMMIT

Cycle formation so, transactions T1 and T2 are not conflict serializable.

9.5 Concurrency control techniques:

1. Lock based
2. Two-phase locking
3. Time-stamp based locking

1. Lock based protocols

- A lock is a mechanism to control concurrent access to a data item.
- Lock is a data variable which is associated with a data item.

Types lock used:

- I. **Binary Locks:** A Binary lock on a data item can either locked or unlocked states.
- II. **Shared/ Exclusive locks:**
 - Data items can be locked in two modes :
 - **exclusive (X) mode:** Data item can be both read as well as written. X-lock is requested using lock-X instruction.
 - **shared (S) mode:** Data item can only be read. S-lock is requested using lock-S instruction.
 - Lock requests are made to concurrency-control manager.
 - Transaction can proceed only after request is granted.
 - **Locking protocol:** It is a set of rules followed by all transactions while requesting and releasing locks.

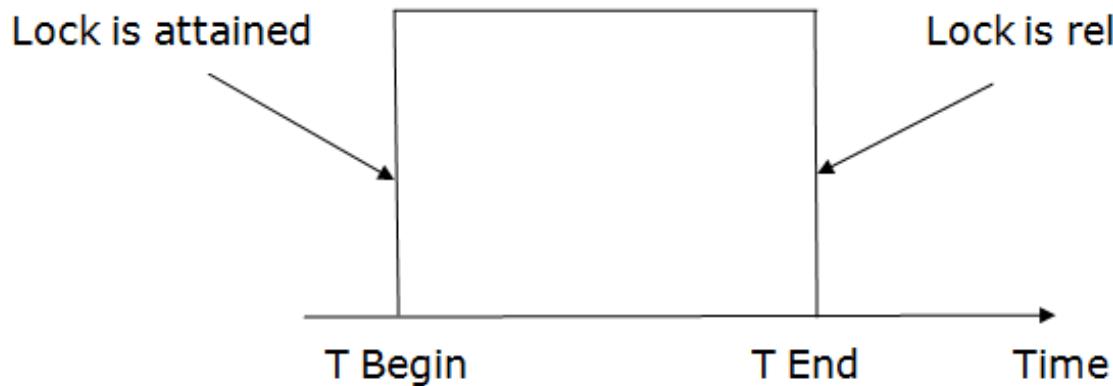


Fig: Transaction execution using lock

Pitfalls of Lock-based protocol:

a) Starvation

- Starvation is the situation when a transaction needs to wait for an indefinite period to acquire a lock.
- Following are the reasons for Starvation:
 - When waiting scheme for locked items is not properly managed
 - In the case of resource leak
 - The same transaction is selected as a victim repeatedly

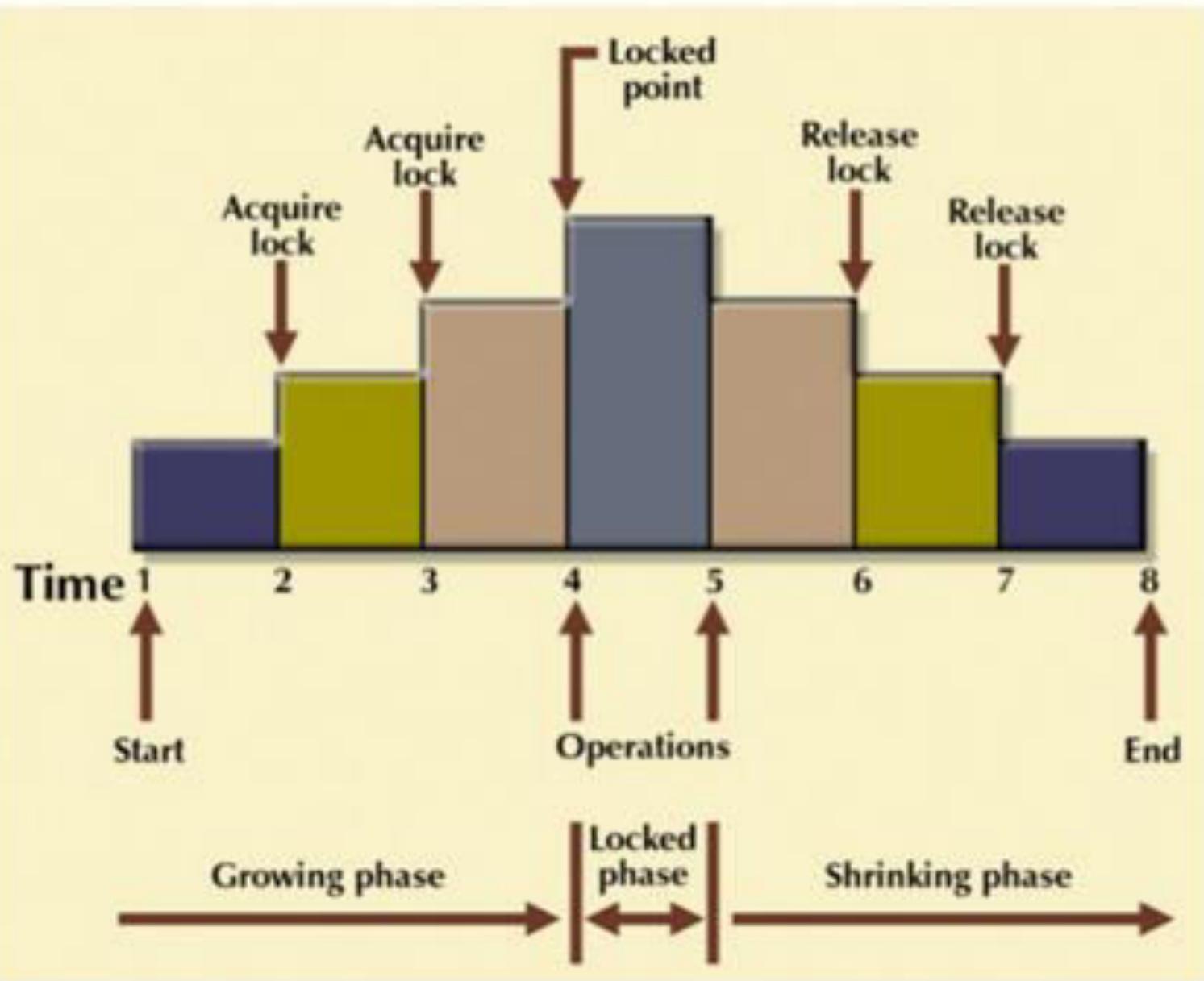
b) Deadlock

- Deadlock refers to a specific situation where two or more processes are waiting for each other to release a resource or more than two processes are waiting for the resource in a circular chain.

2. Two-phase locking (2PL)

- It is also known as a 2PL protocol. It is also called P2L.
- In this type of locking protocol, the transaction should acquire a lock after it releases one of its locks.
- This locking protocol divides the execution phase of a transaction into three different parts.
 - In the first phase, when the transaction begins to execute, **it requires permission for the locks it needs.**
 - The second part is where the transaction obtains all the locks. When a transaction releases its first lock, the third phase starts.
 - In this third phase, the transaction cannot demand any new locks. Instead, it only releases the acquired locks.

- This is a protocol which ensures conflict-serializable schedules.
- **Phase 1: Growing Phase**
 - transaction may obtain locks
 - transaction may not release locks
- **Phase 2: Shrinking Phase**
 - transaction may release locks
 - transaction may not obtain locks
- In between these two phases actually transactions are executed within locked time. [Locked phase]
- The protocol assures serializability.
 - It can be proved that the transactions can be
 - serialized in the order of their lock points
 - i.e. the point where a transaction acquired its final lock.
-



3. Time-stamp based locking

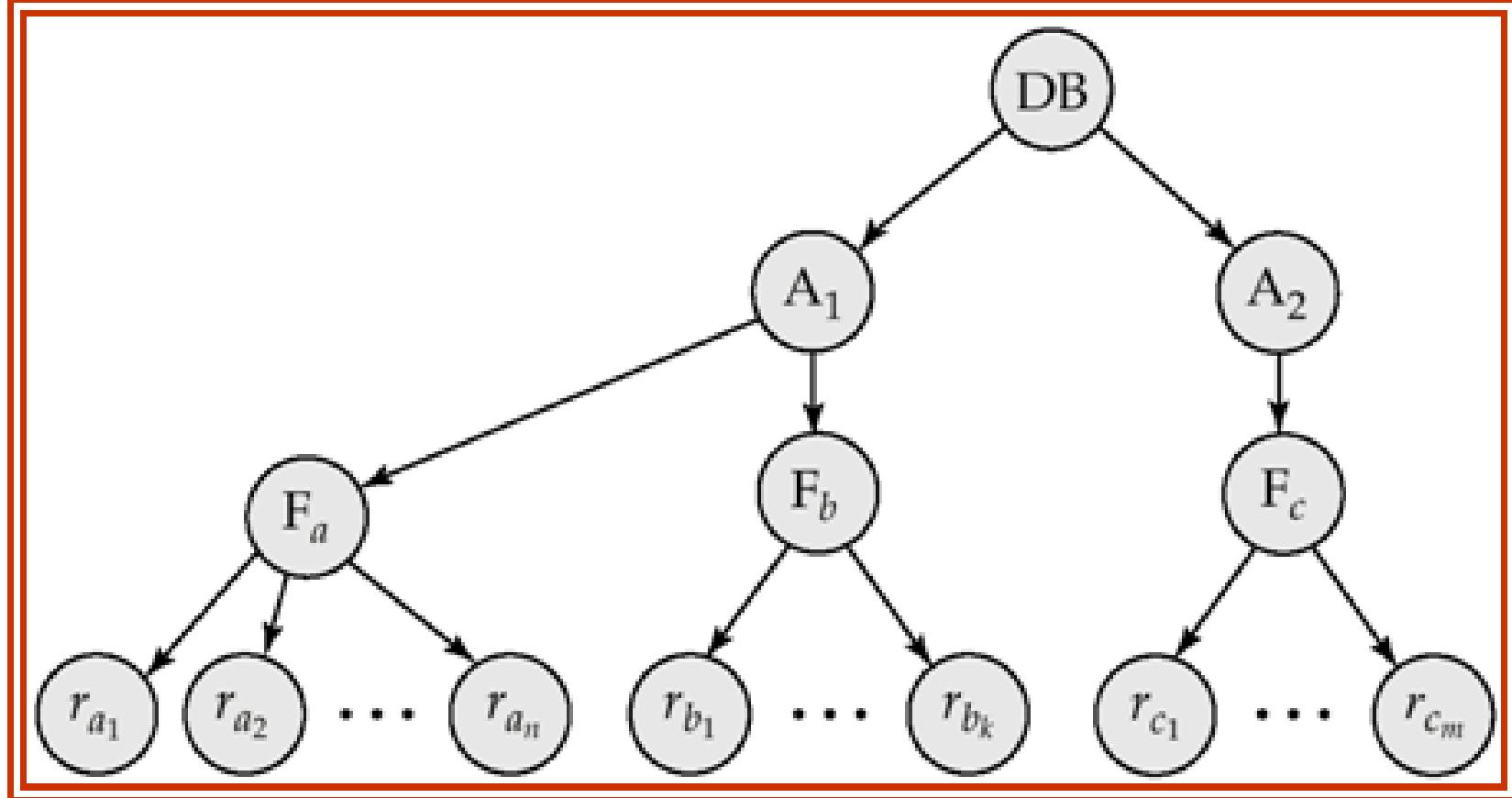
- The timestamp-based algorithm uses a **timestamp to serialize the execution of concurrent transactions.**
- **Timestamp: fixed period of time to execute transaction.**
- This protocol ensures that every conflicting read and write operations are executed in timestamp order.
- **Non-lock concurrency control technique.**
- The protocol uses the **System Time or Logical Count** as a Timestamp.
- The older transaction is always given priority in this method. It uses system time to determine the time stamp of the transaction. This is the most commonly used concurrency protocol.
- Lock-based protocols help you to manage the order between the conflicting transactions when they will execute. Timestamp-based protocols manage conflicts as soon as an operation is created.

- In order to assure such behavior,
 - the protocol maintains for each data Q two timestamp values:
 - **W-timestamp(Q)** is the largest time-stamp of any transaction that executed $\text{write}(Q)$ successfully.
 - **R-timestamp(Q)** is the largest time-stamp of any transaction that executed $\text{read}(Q)$ successfully.
- The timestamp ordering protocol ensures that:
 - any conflicting read and write operations are:
 - executed in timestamp order.

9.6 Multiple granularity

- Multiple Granularity is another **specialized locking strategy** is called multiple-granularity locking.
- **It allows us to efficiently set locks on objects that contain other objects.**
- For instance, a database contains several files, a file is a collection of pages, and a page is a collection of records.
 - A transaction that expects to access most of the pages in a file should probably set a lock on the entire file, rather than locking individual pages (or records') as and when it needs them.
- **Can be represented graphically as a tree**
- When a transaction locks a node in the tree *explicitly*,
 - it *implicitly* locks all the node's descendants in the same mode.
- **Granularity of locking (level in tree where locking is done):**
 - fine granularity (lower in tree):
 - high concurrency, high locking overhead
 - coarse granularity (higher in tree):
 - low locking overhead, low concurrency

Example of Granularity Hierarchy



The levels, starting from the coarsest (top) level are

- *database*
- *area*
- *file*
- *record*

9.7 Deadlock handling

Deadlock:

- In a database, a deadlock is an unwanted situation in which two or more transactions are waiting indefinitely for one another to give up locks.
- Example: Suppose, Transaction T1 holds a lock on some rows in the Students table and **needs to update** some rows in the Grades table. Simultaneously, Transaction T2 **holds** locks on those rows (Which T1 needs to update) in the Grades table **but needs** to update the rows in the Student table **held by Transaction T1**. [waiting for one another indefinitely.]

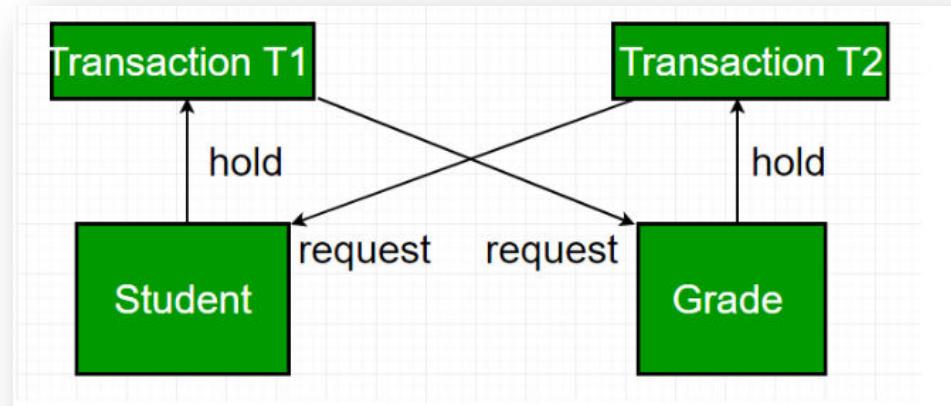
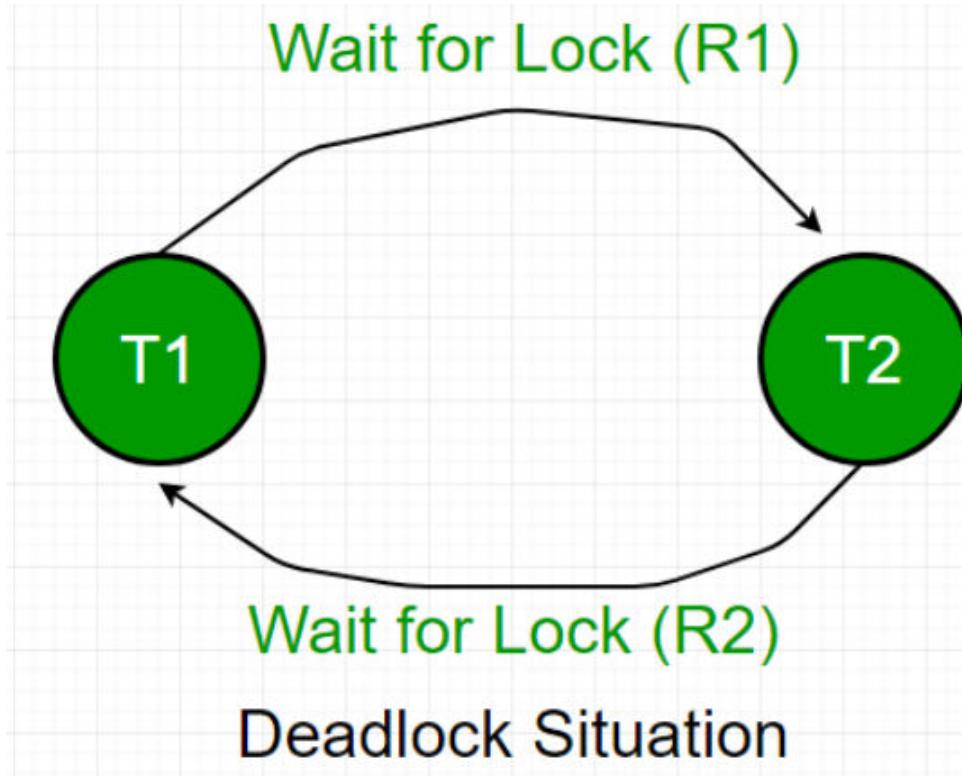


Fig: Example of Deadlock

Deadlock Detection

- **Wait-for-graph** is one of the methods for detecting the deadlock situation. This method is suitable for smaller database. In this method a graph is drawn based on the transaction and their lock on the resource. If the graph created has a closed loop or a cycle, then there is a deadlock.



Deadlock Conditions

- **Following are the deadlock conditions,**

1. Mutual Exclusion
2. Hold and Wait
3. No Preemption
4. Circular Wait

A deadlock may occur, if the above conditions hold true.

In Mutual exclusion states that at least one resource cannot be used by more than one process at a time. The resources cannot be shared between processes.

Hold and Wait states that a process is holding a resource, requesting for additional resources which are being held by other processes in the system.

No Preemption states that a resource cannot be forcibly taken from a process. Only a process can release a resource that is being held by it.

Circular Wait states that one process is waiting for a resource which is being held by second process and the second process is waiting for the third process and so on and the last process is waiting for the first process. It makes a circular chain of waiting.

Following are the requirements to free the deadlock/ Dealing with deadlock condition:

- 1. No Mutual Exclusion :** No Mutual Exclusion means removing all the resources that are sharable.
- 2. No Hold and Wait :** Removing hold and wait condition can be done if a process acquires all the resources that are needed before starting out.
- 3. Allow Preemption :** Allowing preemption is as good as removing mutual exclusion. The only need is to restore the state of the resource for the preempted process rather than letting it in at the same time as the preemptor.
- 4. Removing Circular Wait :** The circular wait can be removed only if the resources are maintained in a hierarchy and process can hold the resources in increasing the order of precedence.

Deadlock prevention

- Deadlock Prevention ensures that the system never enters a deadlock state.
- **Deadlock prevention mechanism proposes two schemes :**

1. Wait-Die Scheme

- In this method, if a transaction request for the resource which is already locked by other transaction, then the DBMS checks for the timestamp of both the transaction and allows older transaction to wait until the resource is available for execution.

2. Wound Wait Scheme

- In this method, if an older transaction requests for a resource held by younger transaction, then older transaction forces younger transaction to kill the transaction and release the resource. The younger transaction is restarted with minute delay but with same timestamp. If the younger transaction is requesting a resource which is held by older one, then younger transaction is asked to wait till older releases it.

- ▶ **Wait-Die** – If T_1 is older than T_2 , T_1 is allowed to wait. Otherwise, if T_1 is younger than T_2 , T_1 is aborted and later restarted.
- ▶ **Wound-Wait** – If T_1 is older than T_2 , T_2 is aborted and later restarted. Otherwise, if T_1 is younger than T_2 , T_1 is allowed to wait.

Deadlock Avoidance

- Deadlock Avoidance helps in avoiding the rolling back conflicting transactions.
- It is not good approach to abort a transaction when a deadlock occurs.
- Rather deadlock avoidance should be used to detect any deadlock situation in advance.

Chapter-10

Database Recovery

Contents

10. Database Recovery

10.1 Importance of database recovery

10.2 Failure Classification

10.3 Log based recovery: Deferred & Immediate In
Single/Multi User Environment

10.4 Write Ahead Logging Protocol

10.5 Shadow paging

10.6 Backup-recovery

10.7 Dumping

Database Recovery

- **Data recovery** is the process of restoring data that has been lost, accidentally deleted, corrupted or made inaccessible.
- DB recovery is the process of restoring the database to a correct (consistent) state in the event of failure.
- It is a process of restoring the database to most recent consistent state.

Purpose of Database Recovery:

- To bring the database into the last consistent state, which existed prior to the failure.
- To preserve transaction properties (Atomicity, Consistency, Isolation and Durability).

States of DB Recovery:

- **Pre-condition:** Recovery when DB is in consistent state.
- **Condition:** Recovery when some kind of failure occurs in system.
- **Post- condition:** Restore DB to the consistent state that existed before the failure. (Recovery after failure.)

Importance of Database Recovery

- Data recovery after failure.
- No data loss.

Why recovery is needed: (What causes a Transaction to fail)

1. A computer failure (system crash):

- A hardware or software error occurs in the computer system during transaction execution. If the hardware crashes, the contents of the computer's internal memory may be lost.

2. A transaction or system error:

- Some operation in the transaction may cause it to fail, such as integer overflow or division by zero. Transaction failure may also occur because of erroneous parameter values or because of a logical programming error. In addition, the user may interrupt the transaction during its execution.

3. Local errors or exception conditions detected by the transaction:

- Certain conditions necessitate cancellation of the transaction. For example, data for the transaction may not be found. A condition, such as insufficient account balance in a banking database, may cause a transaction, such as a fund withdrawal from that account, to be canceled.
- A programmed abort in the transaction causes it to fail.

4. Concurrency control enforcement:

- The concurrency control method may decide to abort the transaction, to be restarted later, because it violates serializability or because several transactions are in a state of deadlock

5. Disk failure:

- Some disk blocks may lose their data because of a read or write malfunction or because of a disk read/write head crash. This may happen during a read or a write operation of the transaction.

6. Physical problems and catastrophes:

- This refers to an endless list of problems that includes power or air-conditioning failure, fire, theft, sabotage, overwriting disks or tapes by mistake, and mounting of a wrong tape by the operator.

Backup and Recovery

- In general, backup and recovery refers to the various strategies and procedures involved in protecting your database against data loss and reconstructing the database after any kind of data loss.
- A **backup** is a copy of data from your database that can be used to reconstruct that data.
- Backups can be divided into physical backups and logical backups.
 - **Physical backups** are backups of the **physical files** used in storing and recovering your database, such as data files, control files, and archived redo logs.
 - Ultimately, every physical backup is a copy of files storing database information to some other location, whether on disk or some offline storage such as tape.
 - **Logical backups** contain **logical data** (for example, tables or stored procedures) exported from a database with an Oracle export utility and stored in a binary file, for later re-importing into a database using the corresponding Oracle import utility.

Some of the backup techniques are as follows :

- **Full database backup :**
 - In this full database including data and database, Meta information needed to restore the whole database, including full-text catalogs are backed up in a predefined time series.
- **Differential backup :**
 - It stores only the data changes that have occurred since last full database backup. When same data has changed many times since last full database backup, a differential backup **stores the most recent version of changed data**.
 - For this first, we need to restore a full database backup.
- **Transaction log backup :**
 - In this, all events that have occurred in the database, like a **record of every single statement executed is backed up**.
 - It is the backup of transaction log entries and contains all transaction that had happened to the database. Through this, the database can be recovered to a specific point in time.
 - It is even possible to perform a backup from a transaction log if the data files are destroyed and not even a single committed (permanently stored) transaction is lost.

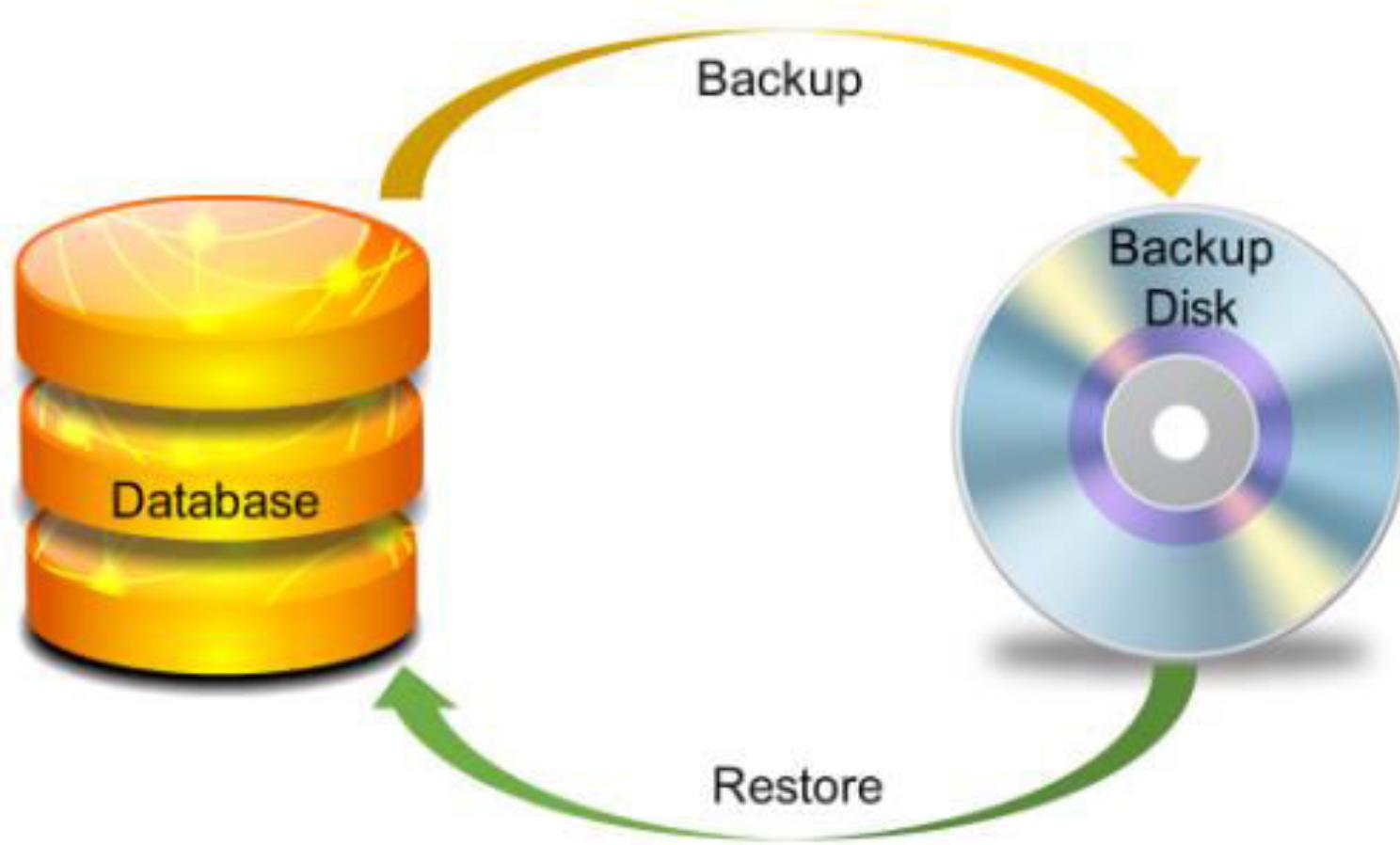


Fig: Backup and Recovery

Failure Classification

- There are many types of failures that can affect database processing. Some failures affect the main memory only, while others involve secondary storage.
- Failure refers to the state of system, where system is not able to do any task accurately.
- Effects of system failure:
 - Loss of data from main memory
 - Loss of data from disk
 - System in a inconsistent state i.e. inaccurate results.

- **Hardware failures:** Hardware failures may include memory errors, disk crashes, bad disk sectors, disk full errors and so on. Hardware failures can also be attributed to design errors, inadequate (poor) quality control during fabrication, overloading (use of under-capacity components) and wearout of mechanical parts.
- **Software failures:** Software failures may include failures related to softwares such as, operating system, DBMS software, application programs and so on.
- **System crashes:** System crashes are due to hardware or software errors, resulting in the loss of main memory. There could be a situation that the system has entered an undesirable state, such as deadlock, which prevents the program from continuing with normal processing. This type of failure may or may not result in corruption of data files.
- **Network failures:** Network failures can occur while using a client-server configuration or a distributed database system where multiple database servers are connected by common networks. Network failures such as communication software failures or aborted asynchronous connections will interrupt the normal operation of the database system.
- **Media failures:** Such failures are due to head crashes or unreadable media, resulting in the loss of parts of secondary storage. They are the most dangerous failures.
- **Application software errors:** These are logical errors in the program that is accessing the database, which cause one or more transactions to fail.
- **Natural physical disasters:** These are failures such as fires, floods, earthquake or power failures.
- **Carelessness:** These are failures due to unintentional destruction of data or facilities by operators or users.
- **Sabotage:** These are failures due to intentional corruption or destruction of data, hardware, or software facilities.

Media (Disk) Failure

- An error can arise when trying to write or read a file that is required to operate the database. This occurrence is called *media failure* because there is a physical problem reading or writing to files on the storage medium.
- A common example of media failure is a **disk head crash**, which causes the loss of all files on a disk drive. All files associated with a database are vulnerable to a disk crash, including data files, online redo log files, and control files.

10.3 Log based recovery

- A **database recovery log** keeps a record of all changes made to a database, including the addition of new tables or updates to existing ones.
- **It is most popular mechanism for implementing recovery algorithm.**
- A log is kept on stable storage.
 - The log is a sequence of log records, and maintains a record of update activities on the database.
- When transaction T_i starts, it registers itself by writing a $\langle T_i \text{ start} \rangle$ log record
- Before T_i executes $\text{write}(X)$, a log record $\langle T_i, X, V1, V2 \rangle$ is written, where $V1$ is the value of X before the write, and $V2$ is the value to be written to X .
 - Log record notes that T_i has performed a write on data item X_j X_j had value $V1$ before the write, and will have value $V2$ after the write.
- When T_i finishes its last statement, the log record $\langle T_i \text{ commit} \rangle$ is written.
- We assume for now that log records are written directly to stable storage (that is, they are not buffered)

- Basic update strategies in log based recovery:
 1. Deferred update
 2. Immediate update

Concepts/ terminologies used in log-based recovery are:

- **Transaction log:** history of actions executed by a DBMS used to guarantee ACID properties over crashes or hardware failure.
- **Undo:** cancel, reverse the effects/Performed task.
- **Redo:** Do something again.
- **Checkpoint:** It is a type of mechanism where all the previous logs are removed from the system and permanently stored in the storage disk.
 - When it reaches to the checkpoint, then the transaction will be updated into the database, and till that point, the entire log file will be removed from the file.

1. Deferred updates techniques(NO-UNDO/REDO algorithm):

- Do not physically update the database on disk **until after a transaction reaches its commit point.**
- Before reaching the commit point, all transaction updates are recorded in the local transaction workspace (or buffers).
- During commit, the updates are first recorded persistently in the log and then written to the DB.
- If a transaction fails before reaching its commit point, no UNDO is needed because it will not have changed the database anyway.
- If there is a crash, it may be necessary to REDO the effects of committed transactions from the Log because their effect may not have been recorded in the database.
- **Main concept:** It need only to redo the committed transaction and no undo is needed in case of failure.
- **Deferred update also known as NO-UNDO/REDO algorithm**

Recovery Using Deferred Update in a Single-User and multi- user environment

i) Recovery Using Deferred Update in a Single-User environment

- The technique is simple because we have only one active transaction (no concurrency).
- **RDU_S:**
 - Use a list for committed transaction and another for active transaction since the last checkpoint.
 - Apply the REDO operation to all write-item of committed transactions in the order of recording them in log.
 - Restart the active transaction.
- REDO(write-operation):
 - Examine the log entry of the operation to get the item and its new value .
 - Set the value of the item in the DB to its new value .

ii) Recovery Using Deferred Update in a multi-User environment:

- The techniques may be more complex depending on the used **concurrency control**.
- Assuming strict 2PL (2 phase locking) concurrency protocol and a checkpoint are used, the recovery technique may be as:
- **RDU_M(with checkpoint)**:
 - Use a list for the committed transactions and another for the active transactions since the last checkpoint.
 - Redo the operations of the committed transactions according to their order in the log.
 - Cancel the active the transactions and resubmit them again.

Ex:

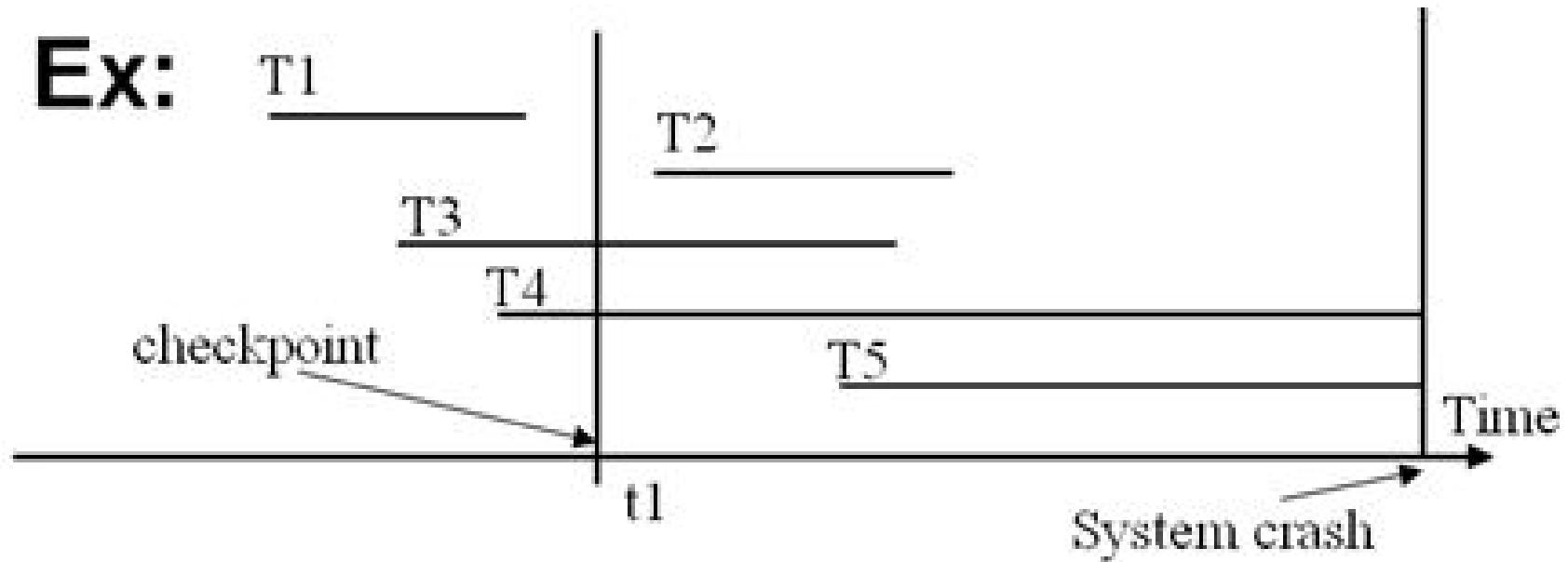


Fig: Example of deferred update in multi-user environment

Notes:

T1 is not redone because it commits before the checkpoint.

T2 and T3 are redone because they commit after the checkpoint.

T4 and T5 are ignored (canceled) and restarted again.

2. Immediate update techniques (UNDO/REDO algorithm)

- The DB may be updated by some operations of **a transaction before the transaction reaches its commit point.**
- To make recovery possible, force writes the changes on the log before to apply them to the DB.
- If a transaction fails before reaching commit point, it must be rolled back by undoing the effect of its operations on the DB.
- It is also required to redo the effect of the committed transactions.
- **Immediate update also known as UNDO/REDO algorithm.**
 - A variation of the algorithm where all updates are recorded in the database before a transaction commits requires only redo –UNDO/NO-REDO algorithm

Recovery Using Deferred Update in a Single-User and multi- user environment

i) Recovery Using Immediate Update in a Single-User Environment:

- **RIU_S algorithm:**
 - Use a list for the committed transactions and another for the active transaction since the last checkpoint.
 - Undo all the write operations of the **active transaction** using the UNDO procedure.
 - Redo the write operations of the **committed transactions** in the order in which they were written in the log using the REDO procedure.
- **Procedure UNDO(write-operation):**
 - Examine the log entry of the operation to get the item and its old value.
 - Set the value of the item in the DB to its old value.

ii) Recovery Using Immediate Update in a Multi-User Environment

- The process depends on the used **concurrency control protocol**.
- Assuming strict 2PL and checkpoint, the technique will be as:

Procedure RIU_M:

- Use a list for the committed transactions and another for the active transaction since the last checkpoint.
- Undo all write operations of the active transactions using the UNDO procedure (undo in reverse order).
- Redo all write operations of committed transactions in their log order (this step can be enhanced as RDU_M).

Algorithm: same as that in the single-user environment.

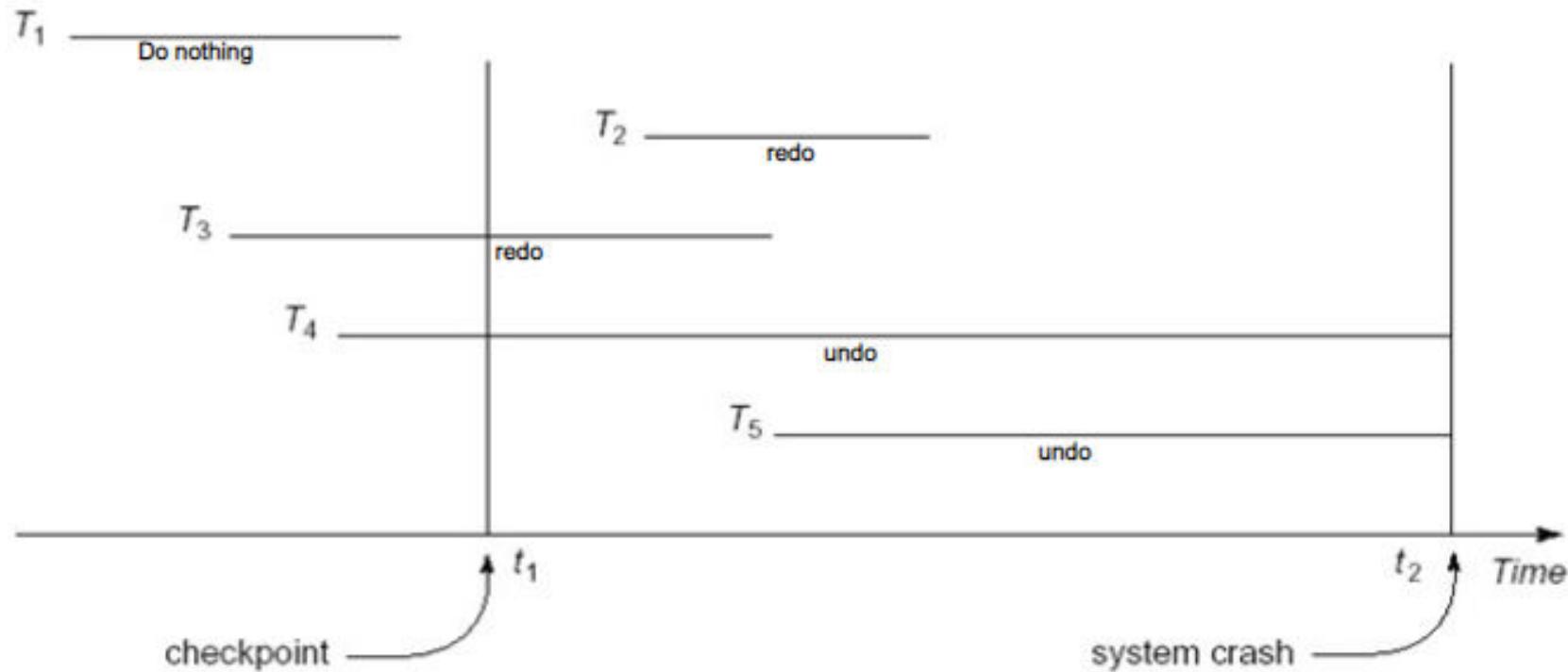


Fig: Example of recovery Using Immediate Update in a Multi-User Environment (immediate update with concurrency control)

10.4 Write Ahead Logging Protocol(WAL)

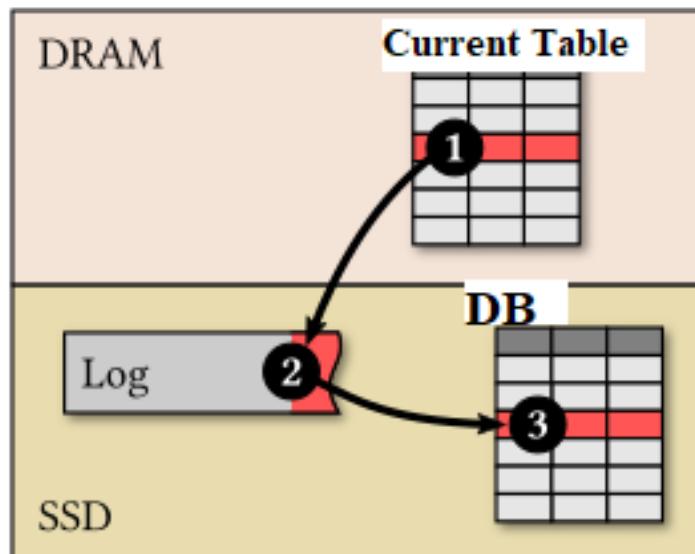
- When in-place update (immediate or deferred) is used then log is necessary for recovery and it must be available to recovery manager. This is achieved by Write-Ahead Logging (WAL) protocol.
- It is a family of techniques for providing atomicity and durability (two of the ACID properties) in database systems. The changes are first recorded in the log, which must be written to stable storage, before the changes are written to the database.
- **WAL states that:**
 - For Undo: Before a data item's AFIM is flushed to the database disk (overwriting the BFIM) its BFIM must be written to the log and the log must be saved on a stable store (log disk).
i.e. **before adding new data/updates into the database previously stored data must be copied to another disk to store permanently.**
 - For Redo: Before a transaction executes its commit operation, all its AFIMs must be written to the log and the log must be saved on a stable store.
i.e. **before permanent execution of transaction, all its new updates must be saved into stable/permanent storage.**

Note: BFIM(Before Image: Before modification)

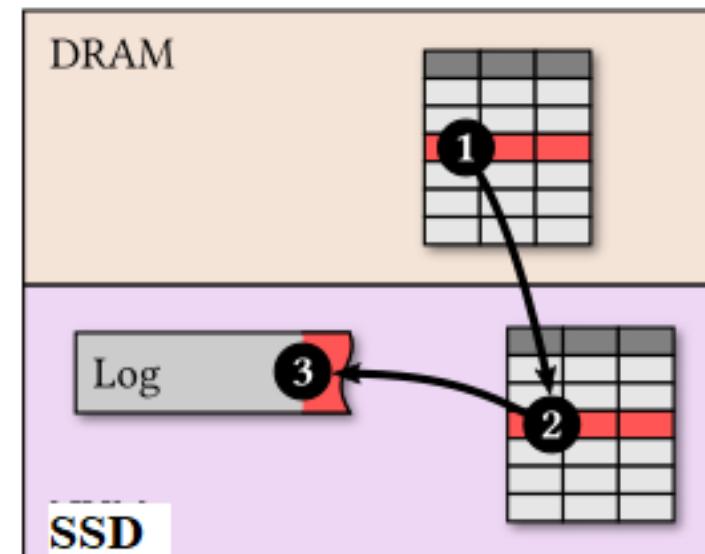
AFIM (After Image: After modification)

Objectives:

- With UNDO: It guarantees atomicity.
- With REDO: It guaranteed durability.



Write-ahead logging



Write-behind logging

The changes are first recorded in the log, before the changes are written to the database.

The changes are first written in the database then recorded in the log.

10.5 Shadow Paging

- It is an alternative to log-based recovery; this scheme is useful if transactions execute serially.
- Concept: No-undo and No-redo
- **Idea:** maintain *two* page tables during the lifetime of a transaction –the current page table, and the shadow page table
- Store the shadow page table in nonvolatile storage, such that state of the database prior to transaction execution may be recovered.
 - Shadow page table is never modified during execution
- To start with, both the page tables are identical. Only current page table is used for data item accesses during execution of the transaction.
- Whenever any page is about to be written for the first time
 - A copy of this page is made onto an unused page. [to be a new current page]
 - The current page table is then made to point to the copy
 - The update is performed on the copy

Two tables are used:

- Current page table: pointing to the most recent DB pages on disk
- Shadow page table: pointing to the DB pages before transaction execution.

While crash occurs: discard current pages; back to the state before transaction executes (using the data referenced by the shadow page table); hence, No-Undo.

While transaction commits: discard shadow pages; hence, No-Redo.

- Advantages:
 - No-redo and no-undo
 - No overhead of writing log records
- Disadvantages:
 - The size of tables can be large because size for garbage collection is also needed.
 - Copying the entire page table is very expensive

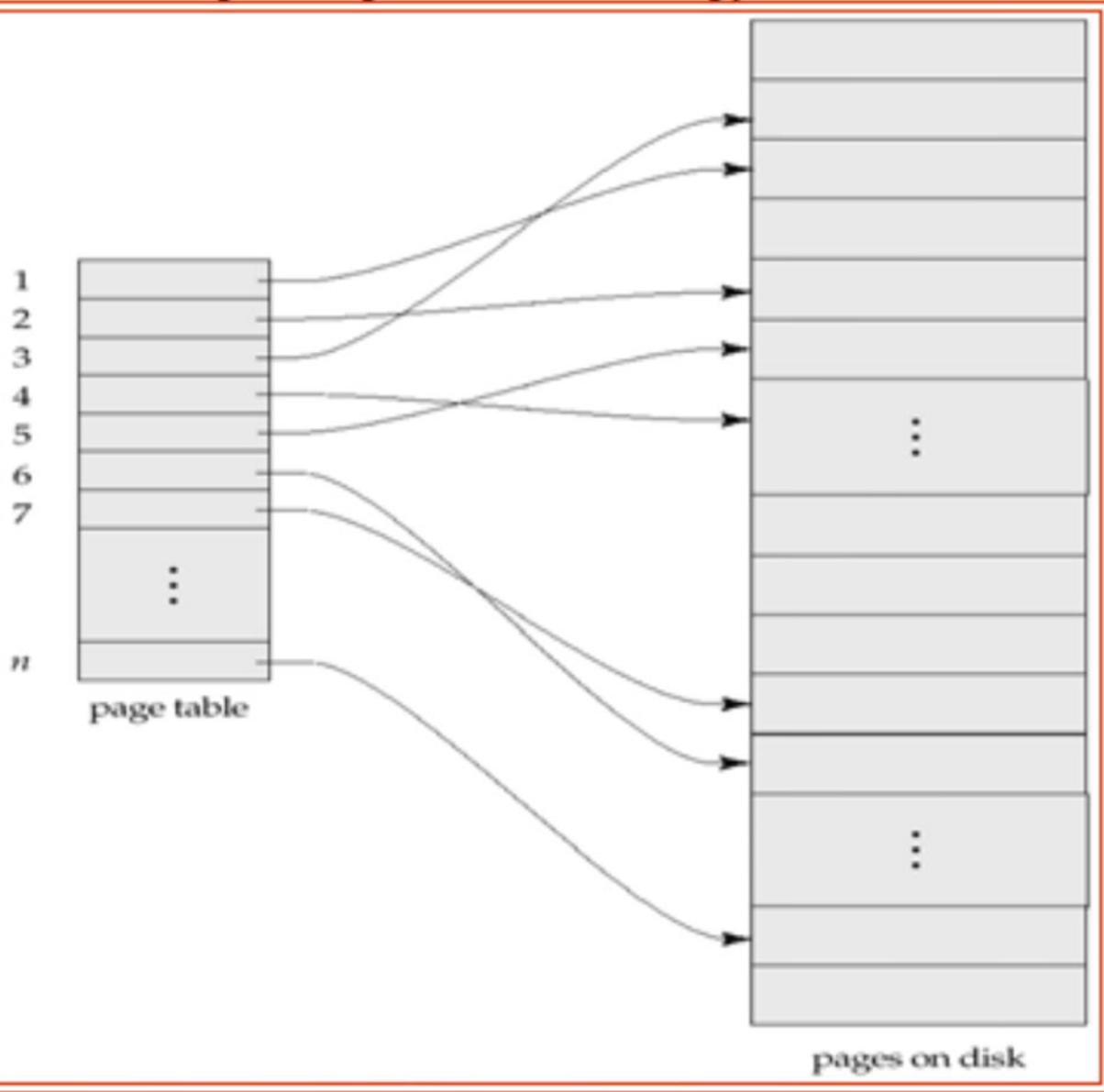


Fig: Current page table with disk

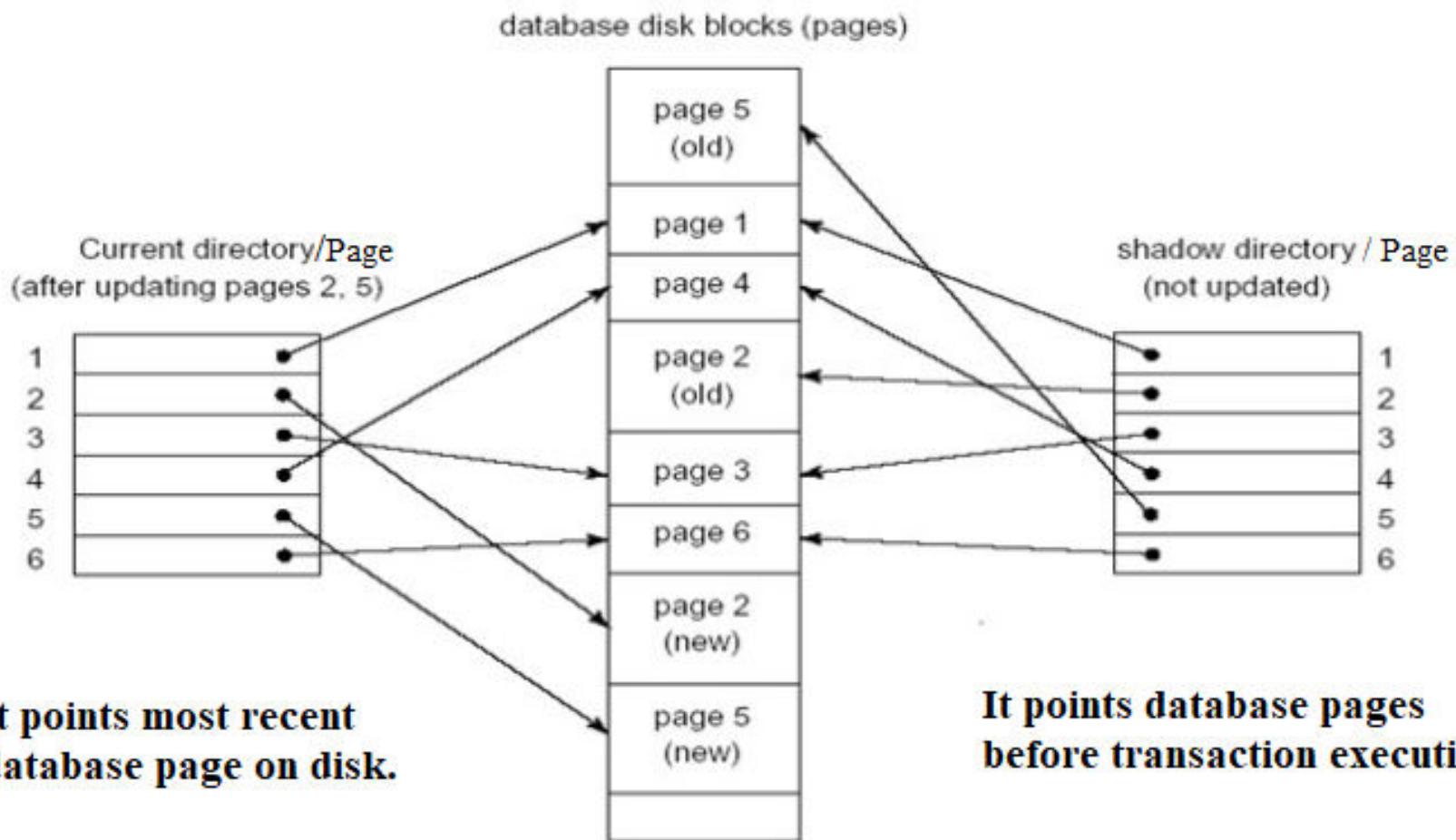


Fig: Example of Shadow page table with current page table

10.6 Backup and Recovery (already covered)

10.7 Dumping:

- A **database dump/ SQL dump** contains a record of the table structure and/or the data from a database and is usually in the form of a list of SQL statements.
- A database dump is most often used for backing up a database so that its contents can be restored in the event of data loss.
- Corrupted databases can often be recovered by analysis of the dump.
- Database dumps are often published by free software and free content projects, to allow reuse or forking of the database.
- It is a major output of data that can help users to either back up or duplicate a database.
- ***How Does MySQL Dump Work?***
 - For each database and table the backup program (e.g. mysqldump) executes:
 - LOCK TABLE *table name*
 - SHOW CREATE TABLE *table name*
 - SELECT * INTO OUTFILE *temporary file*
 - Write the contents of the temporary file to the end of the dump file
 - UNLOCK TABLES

Types of Dumping:

1. Small memory dump:

- It is useful when space is greatly limited.

2. Kernel memory dump:

- It contains all the memory in use by kernel at the time of the crash.
- Kernel is the computer program that constitutes the central core of the computer's operating system.

Advantages of dumping:

- Corrupted databases can be corrected by looking at the contents of the dump.
- There is good assurance that the raw database are not corrupt as all the data is read and it is read using standard SQL queries.

Disadvantages of dumping:

- Requires a Full backup be done every time. Backup can be very time consuming especially on larger databases.
- Locking can be optional. If locking is not performed there is no consistency in the backup.
- Restoring only desired databases or tables requiring editing the SQL Dump file before restoring from it. **Editing raw database dumps is very time consuming and error prone.**

Chapter-11

Advanced Database models

Contents:

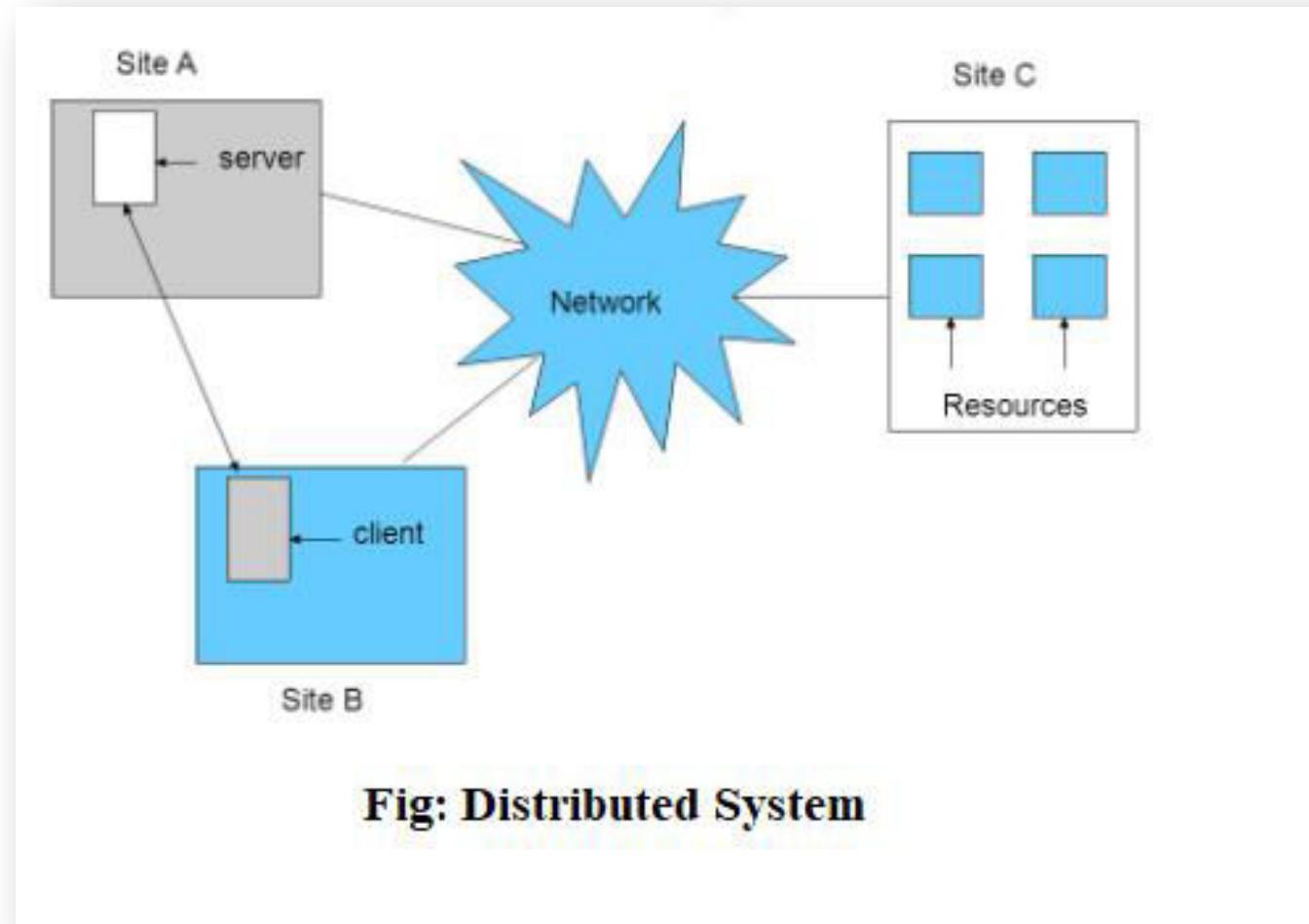
11.1 Distributed Model

11.2 Multimedia model

11.3 ORDBMS(Object Relational Database Management System)

Distributed System

- A *distributed system* is a system whose components are located on different networked computers, which communicate and coordinate their actions by passing messages to one another.



11.1 Distributed Model

- A **distributed database management system** (DDBMS)
 - It is a [software system](#) that permits the management of a [distributed database](#) and makes the distribution transparent to the users.
 - Concept: It is a DB in which not all storage devices are attached to a common processor.
- A distributed database is a collection of multiple, logically interrelated databases distributed over a computer network.
- Distributed database system consists of loosely coupled sites that share no physical component.
- Database systems that run on each site are independent of each other.
- Or
- **Distributed database management systems** is a software for managing databases stored on multiple computers in a network.
- A distributed database is a set of databases stored on multiple computers that typically appears to applications on a single database.
- Consequently, an application can simultaneously access and modify the data in several databases in a network.
- A DDBMS mainly classified into two types:
 1. **Homogeneous Distributed database management systems**
 2. **Heterogeneous Distributed database management systems**

1. Homogeneous Distributed database management systems:-

- In a homogeneous distributed database **all sites have identical software** and are aware of each other and agree to cooperate in processing user requests.
- Each site surrenders part of its autonomy in terms of right to change schema or software.
- Homogeneous DDBMS **appears to user as a single system**.
- The homogeneous system is **much easier to design and manage**.
- The following conditions must be satisfied for homogeneous database:
 - The operating system used, at each location must be same or compatible.
 - The data structures used at each location must be same or compatible.
 - The database application (or DBMS) used at each location must be same or compatible.

2. Heterogeneous Distributed database management systems

- In a heterogeneous distributed database **different sites may use different schema and software**.
- Difference in schema is a major problem for query processing and transaction processing.
- Sites may not be aware of each other and may provide only limited facilities for cooperation in transaction processing.
- The following conditions must be satisfied for heterogeneous system:
 - Different nodes may have different hardware & software and data structures at various nodes or locations are also incompatible.
 - Different computers and operating systems, database applications or data models may be used at each of the locations.

- **Characteristics**
 - All sites are interconnected.
 - Fragments can be replicated.
 - Logically related shared data can be collected.
 - Data at each and every site is controlled by the DBMS.
 - Each Distributed Database Management System takes part in at least one global application.
- **Functionality**
 - Security
 - Keeping track of data
 - Replicated data management
 - System catalog(record) management
 - Distributed transaction management
 - Distributed query processing
 - Distributed database recovery

Manufacturing

Distributed Database

Headquarters

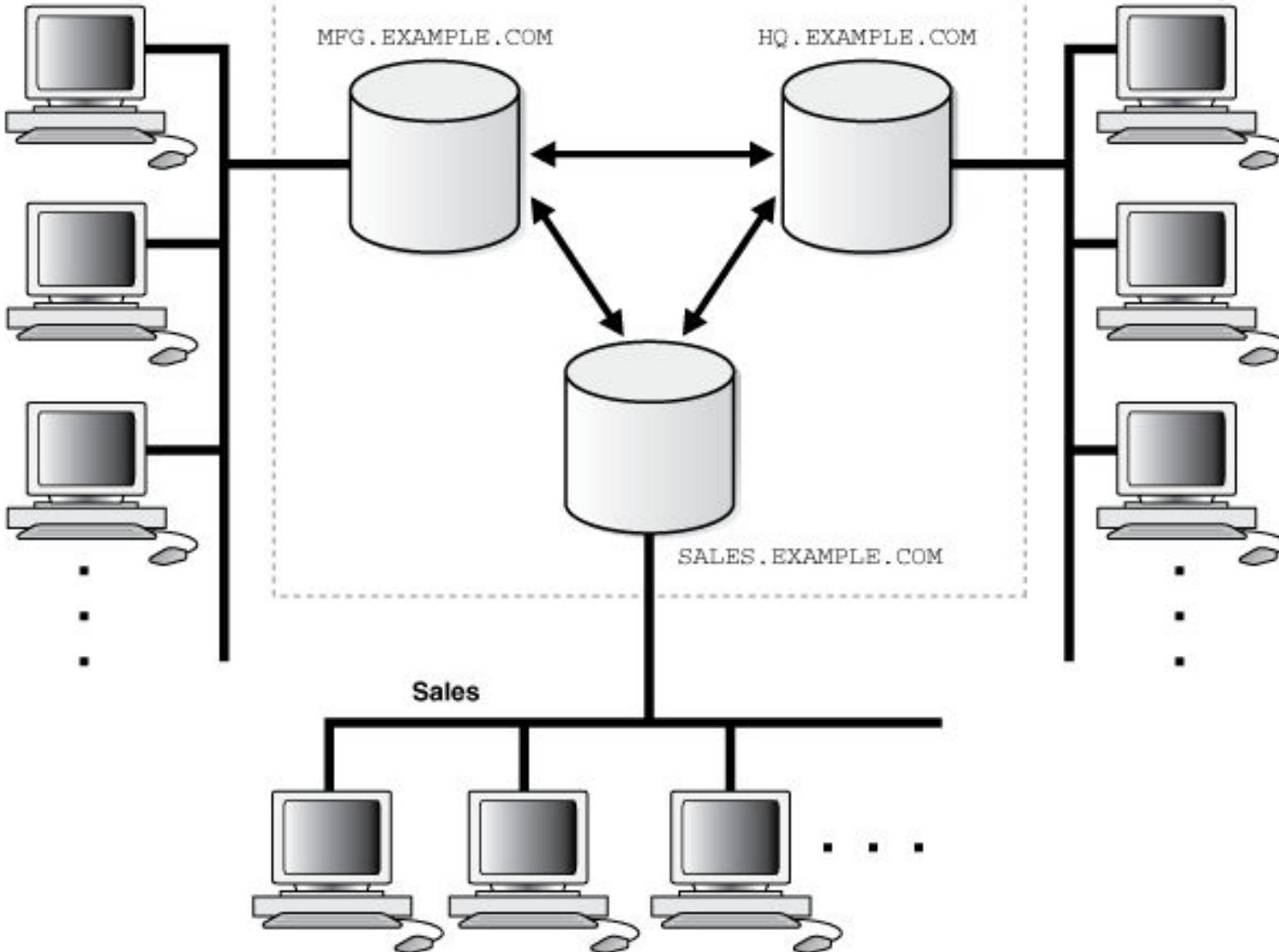


Fig: Distrubutes DBMS

Why DDBMS?/ Advantages of DDBMS:

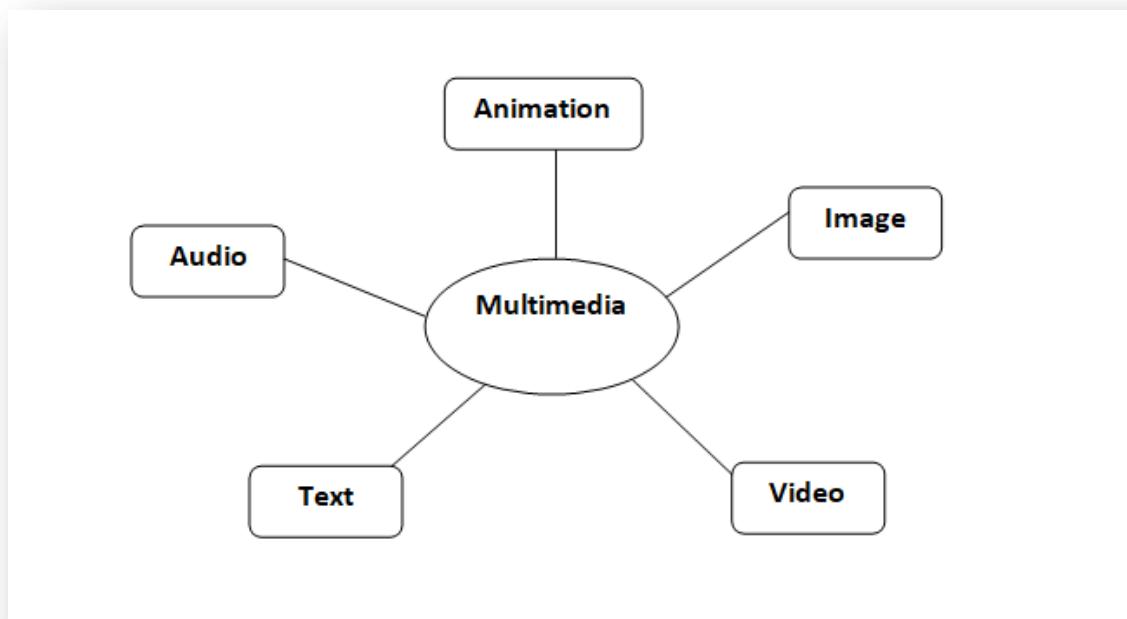
- Distribution and Autonomy of Business Units
 - Departments/Facilities are geographically distributed
 - Each has the authority to create and control own data
 - Business mergers create this environment
- Data sharing
 - Consolidate data across local databases on demand.
- Data communication costs and reliability
 - Economical and reliable to locate data where needed.
 - High cost for remote transactions / large data volumes
 - Dependence on data communications can be risky
- Multiple application vendor environment
 - Each unit may have different vendor applications
 - A distributed DBMS can provide functionality that cuts across separate applications
- Database recovery
 - Replicating data on separate computers may ensure that a damaged database can be quickly recovered

Disadvantages of DDBMS:

- Need of sophisticated software to suit distributed environment results in high cost and complexity.
- Processing overhead to exchange information among the sites.
- Difficult to maintain Data integrity as data are dispersed across various locations.
- Slow response if data are not uniformly distributed

11.2 Multimedia model

- **Multimedia** is content that uses a combination of different content forms such as text, audio, images, animations, video and interactive content.
- Example of multimedia: Video (combination of audio and images)
- Multimedia: multi+ media
 - Multi: many, more
 - Media/ Medium: means of representation of information. Example: text, image, audio/sound, etc.



- **Multimedia database (MMDB)** is a collection of related for multimedia data.
- The multimedia data includes text, images, graphic objects , audio and video,etc.
- **Multimedia Database Management System (MMDBMS)** is a framework that manages different types of data potentially represented in a wide diversity of formats on a wide array of media sources.
- MMDBMS provides support for multimedia data types, and facilitate for creation, storage, access, query and control of a multimedia database.

Application of MMDB/ MMDBMS:

- Digital library
- Video conferencing
- Music DB
- GIS(Geographic information system), etc.

Requirements of MMDB:

- **Integration**
 - Data items do not need to be duplicated for different programs invocations
- **Data independence**
 - Separate the database and the management from the application programs
- **Concurrency control**
 - Allows concurrent transactions
- **Persistence**
 - Data objects can be saved and re-used by different transactions and program invocations
- **Privacy**
 - Access and authorization control
- **Integrity control**
 - Ensures [database](#) consistency between transactions
- **Recovery**
 - Failures of transactions should not affect the [persistent](#) data storage
- **Query support**
 - Allows easy querying of multimedia data

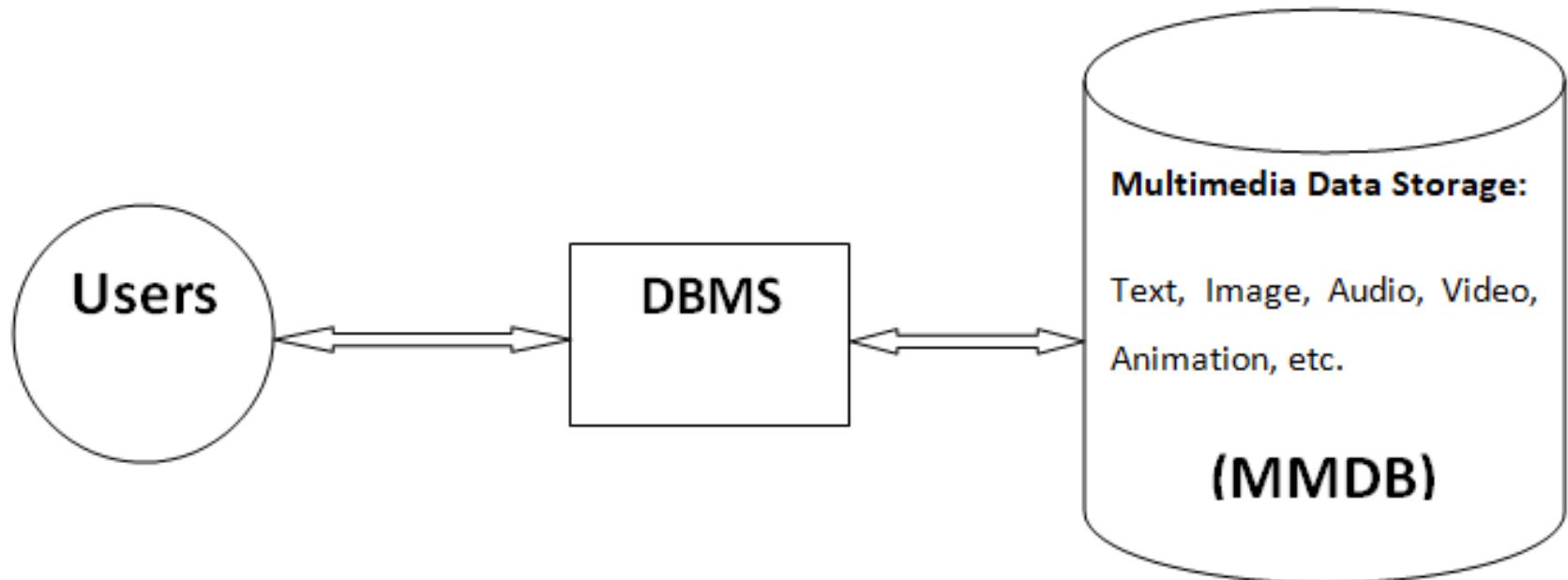


Fig: Architecture of MMDBMS

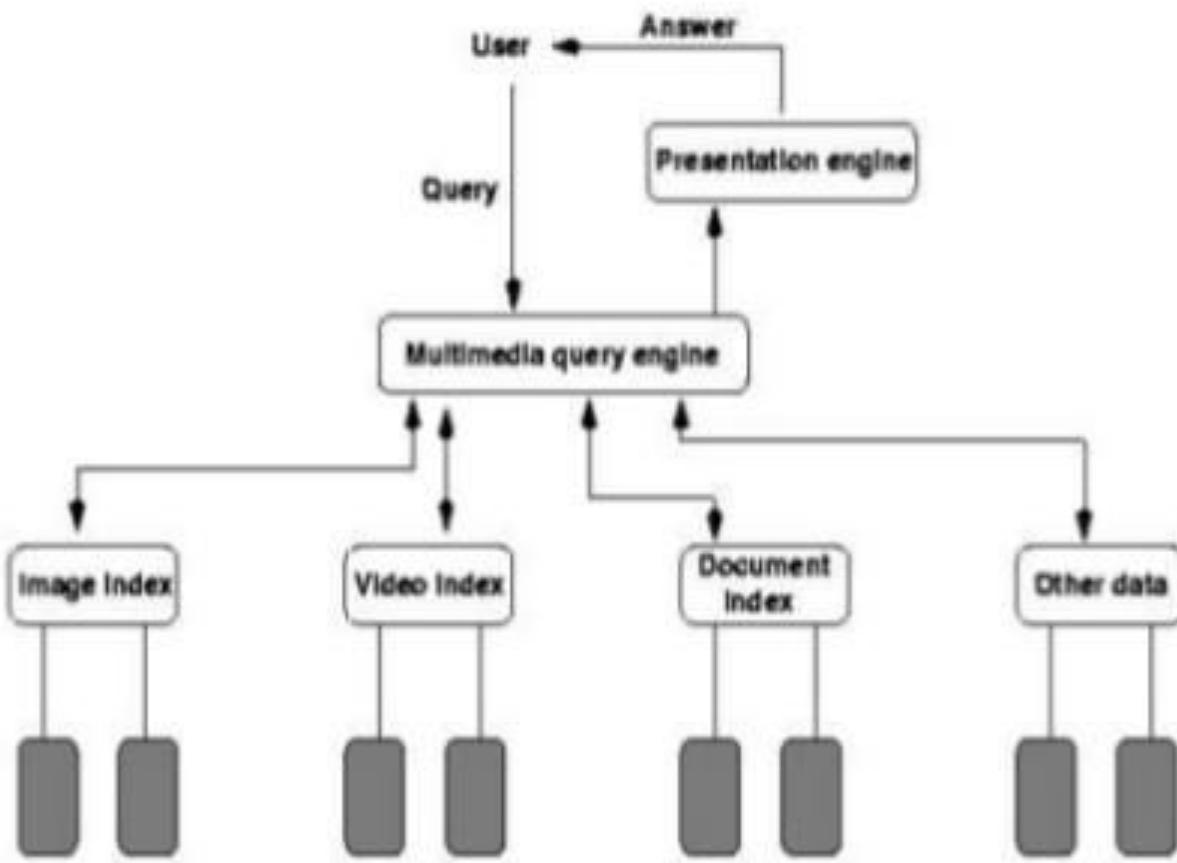


Fig: MMDB Architecture

11.3 ORDBMS(Object Relational Database Management System)

- An **object-relational database (ORD)**, or **object-relational database management system (ORDBMS)**, is a DBMS similar to a relational database, but with an object-oriented database model: objects, classes and inheritance are directly supported in database schemas and in the query language.
- In addition, just as with proper relational systems, it supports extension of the data model with custom data-types and methods.
- It **directly supports object oriented paradigms objects, classes, inheritances , polymorphism in the database schemas & query language**. Also has the ability to extensible with user-defined types & methods.
- Major characteristics:
 - Data stored in the form of **table as well as class and objects** in DB.
 - It uses object-oriented concepts such as class, object, inheritance, abstraction, etc.

Relational Model	Object-Oriented Model
Relation	Class
Tupel / Row	Instance/ Object
Column/ Attribute	Attribute
Stored procedure/ Query	Method

Advantages

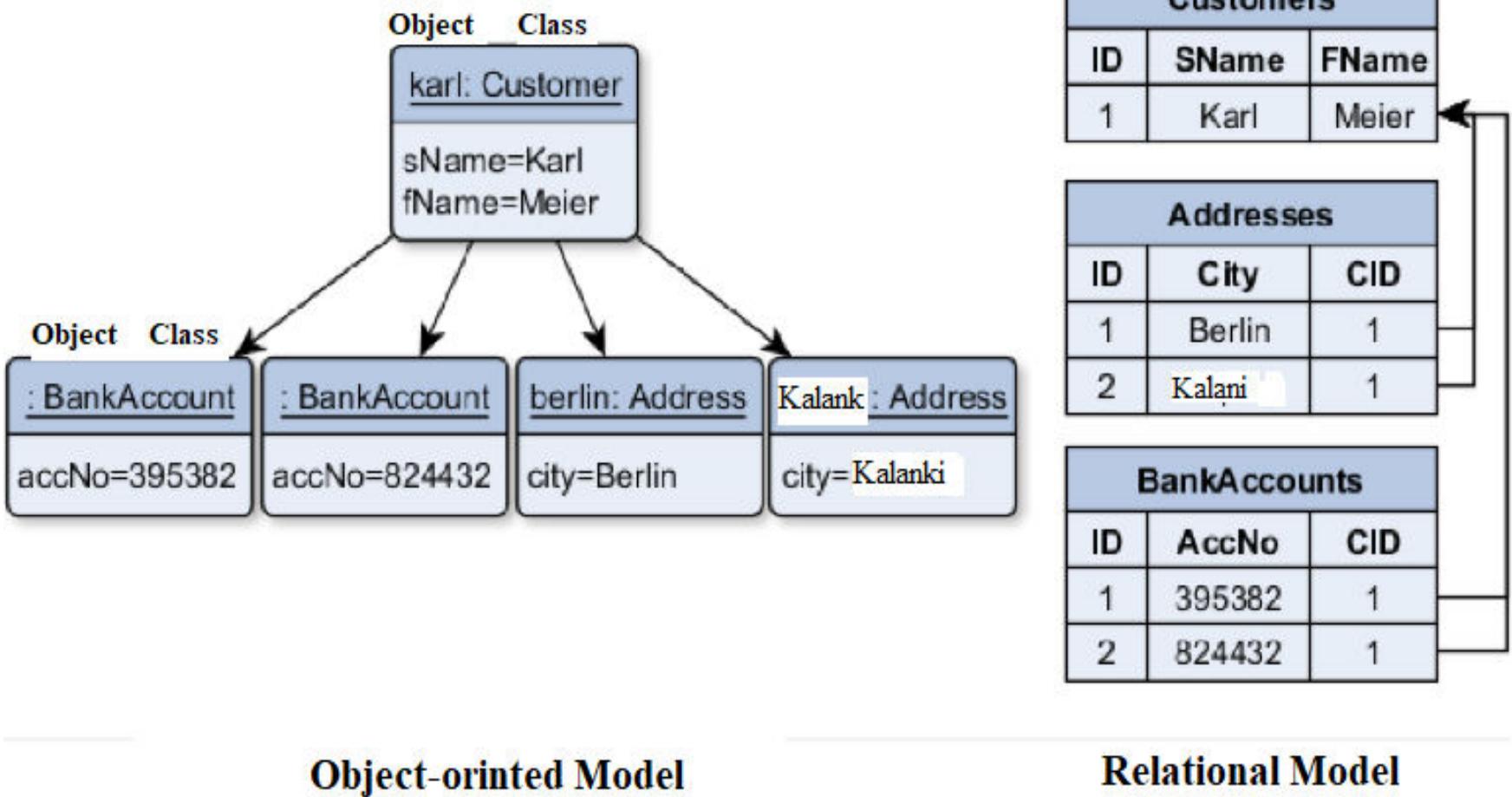
- Supports rich data types such as audio, video and image files.
- Access and manipulate the complex data types in the database without breaking the rule of relational model.
- It provides facilities of both relational and object-oriented model.
- Extensibility
 - The functionality of the system can be extended in Object relational data model. This can be achieved using complex data types as well as advanced concepts of object oriented model such as inheritance.
- Increased Productivity
- Reuse and Sharing

Disadvantages

- Complexity
- Increased costs

Examples

ORDBMS vendors in the market : Oracle, SQL Server, [PostgreSQL](#), Informix. Among these, [PostgreSQL](#) is an Open Source ORDBMS.



Object-oriented Model

Relational Model

Fig: Example of Object- Relational Model