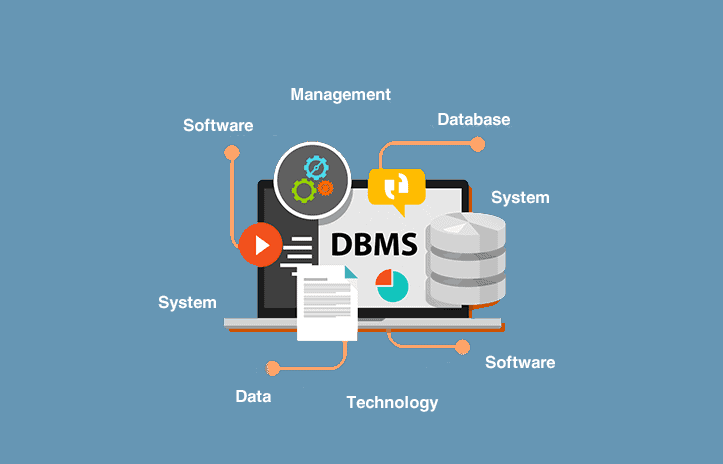
**Database Management System**

****

**By:**

**BANDANA**

**KUMARI**

**TRAINEE AT INNOLABZ VENTURE**

|  |  |  |
| --- | --- | --- |
| **INDEX** | | |
| **S.No.** | **Topic** | **Page No.** |
| 1 | Introduction | 3-14 |
| 2 | Data Modeling | 15-37 |
| 3 | Relational Data Modeling | 38-43 |
| 4 | Normalisation | 44-55 |
| 5 | Transaction Processing | 56-64 |
| 6 | Concurrency Control | 65-68 |
| 7 | File Organization | 69-81 |
| 8 | Indexing and B+ Tree | 82-84 |
| 9 | Hashing | 85-86 |
| 10 | SQL | 87-93 |

# Introduction

A 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 this data. It is also used to organize the data in the form of a table, schema, views, and reports, etc.

**For example:** The college Database organizes the data about the admin, staff, students and faculty etc.

Using the database, you can easily retrieve, insert, and delete the information.

## Database Management System

* Database management system is a software which is used to manage the database. For example: [MySQL](https://www.javatpoint.com/mysql-tutorial), [Oracle](https://www.javatpoint.com/oracle-tutorial), etc are a very popular commercial database which is used in different applications.
* DBMS provides an interface to perform various operations like database creation, storing data in it, updating data, creating a table in the database and a lot more.
* It provides protection and security to the database. In the case of multiple users, it also maintains data consistency.

**DBMS allows users the following tasks:**

* **Data Definition:** It is used for creation, modification, and removal of definition that defines the organization of data in the database.
* **Data Updation :** It is used for the insertion, modification, and deletion of the actual data in the database.
* **Data Retrieval:** It is used to retrieve the data from the database which can be used by applications for various purposes.
* **User Administration:** It is used for registering and monitoring users, maintain data integrity, enforcing data security, dealing with concurrency control, monitoring performance and recovering information corrupted by unexpected failure.

**Advantages of DBMS**

* **Controls database redundancy:** It can control data redundancy because it stores all the data in one single database file and that recorded data is placed in the database.
* **Data sharing:** In DBMS, the authorized users of an organization can share the data among multiple users.
* **Easily Maintenance:** It can be easily maintainable due to the centralized nature of the database system.
* **Reduce time:** It reduces development time and maintenance need.
* **Backup:** It provides backup and recovery subsystems which create automatic backup of data from hardware and software failures and restores the data if required.
* **Multiple user interface:** It provides different types of user interfaces like graphical user interfaces, application program interfaces

**Disadvantages of DBMS**

* **Cost of Hardware and Software:** It requires a high speed of data processor and large memory size to run DBMS software.
* **Size:** It occupies a large space of disks and large memory to run them efficiently.
* **Complexity:** Database system creates additional complexity and requirements.
* **Higher impact of failure:** Failure is highly impacted the database because in most of the organization, all the data stored in a single database and if the database is damaged due to electric failure or database corruption then the data may be lost forever.

# **Types of Databases**

There are various types of databases used for storing different varieties of data:



## 1) Centralized Database

It comforts the users to access the stored data from different locations through several applications. An example of a Centralized database can be Central Library that carries a central database of each library in a college/university.

### **Advantages of Centralized Database**

* It has decreased the risk of data management, i.e., manipulation of data will not affect the core data.
* Data consistency is maintained as it manages data in a central repositor

### **Disadvantages of Centralized Database**

* The size of the centralized database is large, which increases the response time for fetching the data.
* It is not easy to update such an extensive database system

## 2) Distributed Database

Unlike a centralized database system, in distributed systems, data is distributed among different database systems of an organization. These database systems are connected via communication links. Such links help the end-users to access the data easily. **Examples** of the Distributed database are Apache Cassandra, HBase, Ignite, etc.

### **Advantages of Distributed Database**

* Modular development is possible in a distributed database, i.e., the system can be expanded by including new computers and connecting them to the distributed system.
* One server failure will not affect the entire data set.

## 3) Relational Database

This database is based on the relational data model, which stores data in the form of rows(tuple) and columns(attributes), and together forms a table(relation). A relational database uses SQL for storing. **Examples** of Relational databases are MySQL, Microsoft SQL Server, Oracle, etc.

### **Properties of Relational Database**

There are following four commonly known properties of a relational model known as ACID properties, where:

**A means Atomicity:** This ensures the data operation will complete either with success or with failure. It follows the 'all or nothing' strategy. For example, a transaction will either be committed or will abort.

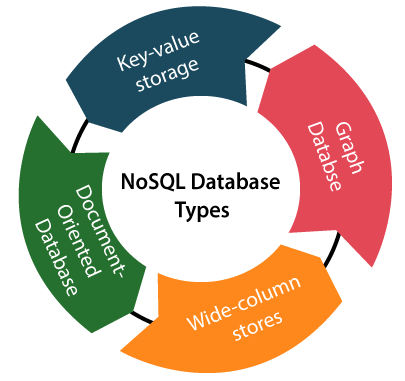
**C means Consistency:** If we perform any operation over the data, its value before and after the operation should be preserved. For example, the account balance before and after the transaction should be correct, i.e., it should remain conserved.

**I means Isolation:** There can be concurrent users for accessing data at the same time from the database. Thus, isolation between the data should remain isolated. For example, when multiple transactions occur at the same time, one transaction effects should not be visible to the other transactions in the database.

**D means Durability:** It ensures that once it completes the operation and commits the data, data changes should remain permanent.

## 4) NoSQL Database

Non-SQL/Not Only SQL is a type of database that is used for storing a wide range of data sets. It is not a relational database as it stores data not only in tabular form but in several different ways. It came into existence when the demand for building modern applications increased. We can further divide a NoSQL database into the following four types:



**a. Key-value storage:** It is the simplest type of database storage where it stores every single item as a key (or attribute name) holding its value, together.

**b. Document-oriented Database:** A type of database used to store data as JSON-like document. It helps developers in storing data by using the same document-model format as used in the application code.

**c. Graph Databases:** It is used for storing vast amounts of data in a graph-like structure. Most commonly, social networking websites use the graph database.

**d. Wide-column stores:** It is similar to the data represented in relational databases. Here, data is stored in large columns together, instead of storing in rows.

### **Advantages of NoSQL Database**

* It enables good productivity in the application development as it is not required to store data in a structured format.
* It is a better option for managing and handling large data sets.
* It provides high scalability.
* Users can quickly access data from the database through key-value.

## 5) Cloud Database

A type of database where data is stored in a virtual environment and executes over the cloud computing platform. It provides users with various cloud computing services (SaaS, PaaS, IaaS, etc.) for accessing the database. There are numerous cloud platforms, but the best options are:

* Amazon Web Services (AWS)
* Microsoft Azure,Google Cloud SQL, etc.

## 6) Object-oriented Databases

The type of database that uses the object-based data model approach for storing data in the database system. The data is represented and stored as objects which are similar to the objects used in the object-oriented programming language.

**Object-oriented programming properties**

* Objects
* Classes
* Inheritance
* Polymorphism
* Encapsulation

## 7) Hierarchical Databases

It is the type of database that stores data in the form of parent-children relationship nodes. Here, it organizes data in a tree-like structure.



Data get stored in the form of records that are connected via links. Each child record in the tree will contain only one parent. On the other hand, each parent record can have multiple child records.

# **RDBMS**

**RDBMS** stands for Relational Database Management Systems.

An RDBMS is a [DBMS](https://techterms.com/definition/dbms) designed specifically for [relational databases](https://techterms.com/definition/relational_database). Therefore, RDBMSs are a subset of DBMSs.

A relational database refers to a [database](https://techterms.com/definition/database) that stores data in a structured format, using [rows](https://techterms.com/definition/row) and [columns](https://techterms.com/definition/column). This makes it easy to locate and access specific values within the database.

All modern database management systems like SQL, MS SQL Server, IBM DB2, ORACLE, My-SQL and Microsoft Access are based on RDBMS.

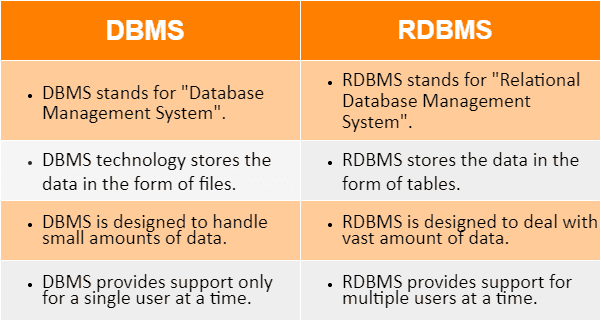
Relational DBMS owes its foundation to the fact that the values of each table are related to others. It has the capability to handle larger magnitudes of data and simulate queries easily.

Relational Database Management Systems maintains data integrity by simulating the following features:

* **Entity Integrity:** No two records of the database table can be completely duplicate.
* **Referential Integrity:** Only the rows of those tables can be deleted which are not used by other tables. Otherwise, it may lead to data inconsistency.
* **User-defined Integrity:** Rules defined by the users based on confidentiality and access.
* **Domain integrity :** The columns of the database tables are enclosed within some structured limits, based on default values, type of data or ranges.

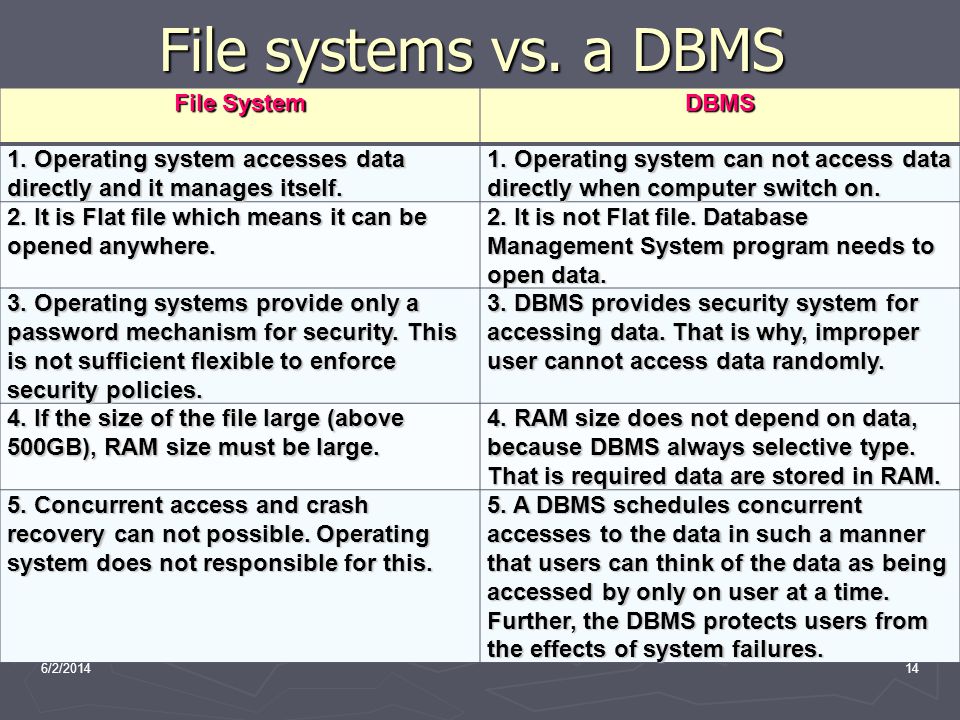
**Difference between DBMS and RDBMS**

The main differences between DBMS and RDBMS are given below:



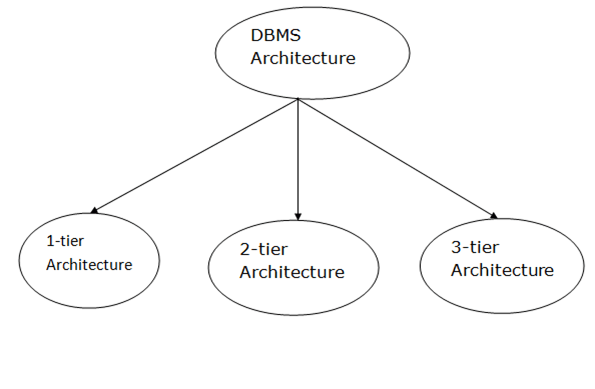
After observing the differences between DBMS and RDBMS, we can say that RDBMS is an extension of DBMS. There are many software products in the market today who are compatible for both DBMS and RDBMS. Means today a RDBMS application is DBMS application and vice-versa.

# **DBMS vs. File System**

There are following differences between DBMS and File system:

# **DBMS Architecture**

* The DBMS design depends upon its architecture. The basic client/server architecture is used to deal with a large number of PCs, web servers, database servers and other components that are connected with networks.
* The client/server architecture consists of many PCs and a workstation which are connected via the network.
* DBMS architecture depends upon how users are connected to the database to get their request done.



### **1-Tier Architecture**

* In this architecture, the database is directly available to the user. It means the user can directly sit on the DBMS and uses it.
* This is the simplest **architecture** of Database in which the client, server, and Database all reside on the same machine.

### **2-Tier Architecture**

* There are 2 layer in 2-tier architecture

1. Client Layer
2. Database Server Layer

* Client has a machnine which has an interface . The interface makes a connection with the database server through JDBC & ODBC

1. JDBC - Java Database Connectivity
2. ODBC – Open Database Connectivity

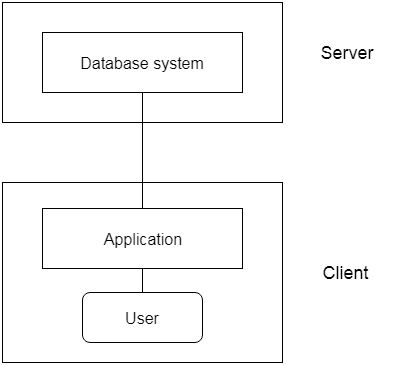
**Advantages :-**

* Limited number of Clients
* Maintainance – Easy to maintain

**Disadvantages :-**

* Security - Security is low because client has direct contact with database server .
* Scalability – Users are not limited . So, when large number of user are there at that time 2-tier architecture fails. To overcome this problem 3-tier architecture is used in new days.

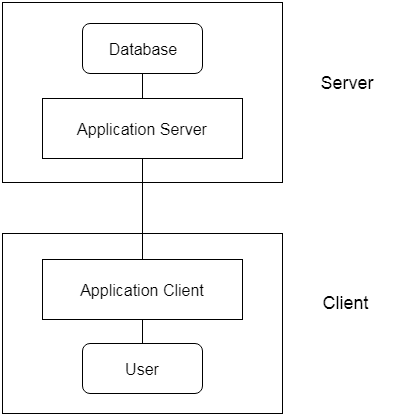
**2-tier Architecture**



### **3-Tier Architecture**

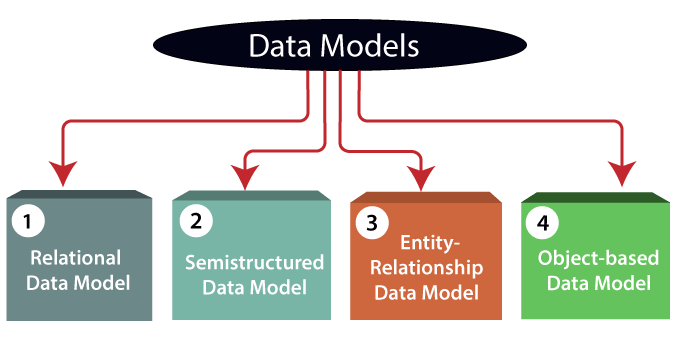
* The 3-Tier architecture contains another layer between the client and server. In this architecture, client can't directly communicate with the server.
* The application on the client-end interacts with an application server which further communicates with the database system.
* The 3-Tier architecture is used in case of large web application.

**Fig: 3-tier Architecture**



# **Data Models**

Data Model is the modeling of the data description, data semantics, and consistency constraints of the data. It provides the conceptual tools for describing the design of a database at each level of data abstraction. Therefore, there are following four data models used for understanding the structure of the database:



**1) Relational Data Model:** It represents the database as a collection of relations. A relation is nothing but a table of values. Every row in the table represents a collection of related data values. These rows in the table denote a real-world entity or relationship.

**2) Entity-Relationship Data Model:** An **Entity–relationship model (ER model)** describes the structure of a database with the help of a diagram, which is known as **Entity Relationship Diagram (ER Diagram)**. An ER model is a design or blueprint of a database that can later be implemented as a database. The main components of E-R model are: entity set and relationship set.

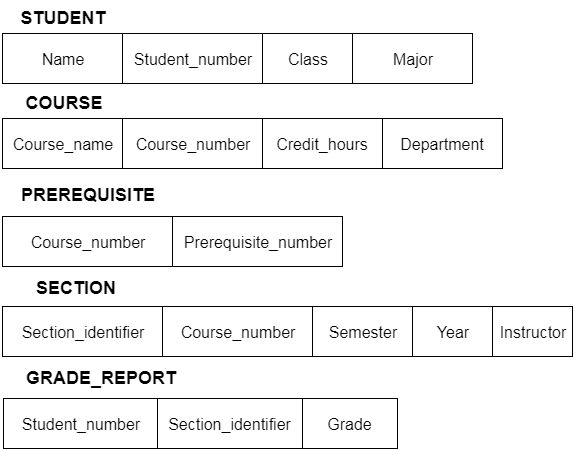
**3) Object-based Data Model:** In Object Oriented Data Model, data and their relationships are contained in a single structure which is referred as object in this data model. In this, real world problems are represented as objects with different attributes. All objects have multiple relationships between them. Basically, it is combination of Object Oriented programming and Relational Database Model

**4) Semi-structured Data Model :** It is the data which does not conforms to a data model but has some structure. It lacks a fixed or rigid schema. It is the data that does not reside in a rational database but that have some organisational properties that make it easier to analyse. With some process, we can store them in the relational database.

**Data model Schema and Instance**

A schema diagram can display only some aspects of a schema like the name of record type, data type, and constraints. Other aspects can't be specified through the schema diagram. For example, the given figure neither show the data type of each data item nor the relationship among various files.

In the database, actual data changes quite frequently. For example, in the given figure, the database changes whenever we add a new grade or add a student. The data at a particular moment of time is called the instance of the database.



# **Data Independence**

* Data independence can be explained using the three-schema architecture.
* Data independence refers characteristic of being able to modify the schema at one level of the database system without altering the schema at the next higher level.

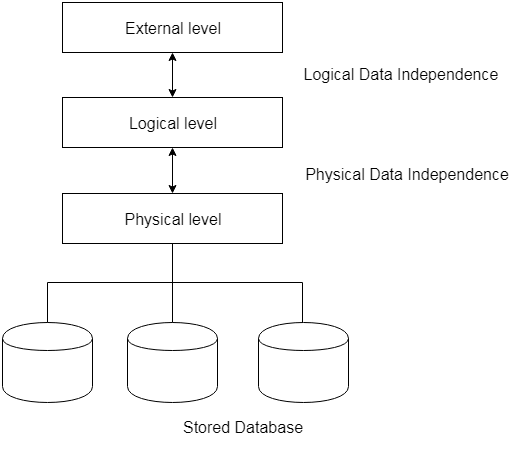
There are two types of data independence:

## 1. Logical Data Independence

* Logical data independence refers characteristic of being able to change the conceptual schema without having to change the external schema.
* Logical data independence is used to separate the external level from the conceptual view.
* Logical data independence occurs at the user interface level.

## 2. Physical Data Independence

* Physical data independence can be defined as the capacity to change the internal schema without having to change the conceptual schema.
* Physical data independence is used to separate conceptual levels from the internal levels.
* Physical data independence occurs at the logical interface level.



# **Database Language**

* A DBMS has appropriate languages and interfaces to express database queries and updates.
* Database languages can be used to read, store and update the data in the database.

## Types of Database Language

## DBMS Language

### **1. Data Definition Language**

* **DDL** stands for **D**ata **D**efinition **L**anguage. It is used to define database structure or pattern.
* It is used to create schema, tables, indexes, constraints, etc. in the database

Here are some tasks that come under DDL:

* **Create:** It is used to create objects in the database.
* **Alter:** It is used to alter the structure of the database.
* **Drop:** It is used to delete objects from the database.
* **Truncate:** It is used to remove all records from a table.
* **Rename:** It is used to rename an object.
* **Comment:** It is used to comment on the data dictionary.

### **2. Data Manipulation Language**

**DML** stands for **D**ata **M**anipulation **L**anguage. It is used for accessing and manipulating data in a database. It handles user requests.

Here are some tasks that come under DML:

* **Select:** It is used to retrieve data from a database.
* **Insert:** It is used to insert data into a table.
* **Update:** It is used to update existing data within a table.
* **Delete:** It is used to delete all records from a table.
* **Merge:** It performs UPSERT operation, i.e., insert or update operations.
* **Call:** It is used to call a structured query language or a Java subprogram.
* **Explain Plan:** It has the parameter of explaining data.
* **Lock Table:** It controls concurrency.

### **3. Data Control Language**

* **DCL** stands for **D**ata **C**ontrol **L**anguage. It is used to retrieve the stored or saved data.
* The DCL execution is transactional. It also has rollback parameters.

Here are some tasks that come under DCL:

* **Grant:** It is used to give user access privileges to a database.
* **Revoke:** It is used to take back permissions from the user.

There are the following operations which have the authorization of Revoke : JDK, JRE, and JV

CONNECT, INSERT, USAGE, EXECUTE, DELETE, UPDATE and SELECT.

### **4. Transaction Control Language**

TCL is used to run the changes made by the DML statement. TCL can be grouped into a logical transaction.

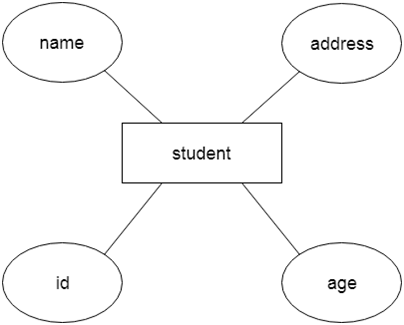
Here are some tasks that come under TCL:

* **Commit:** It is used to save the transaction on the database.
* **Rollback:** It is used to restore the database to original since the last Commit.

# **ER model**

* ER model stands for an Entity-Relationship model. It is a high-level data model. This model is used to define the data elements and relationship for a specified system.
* It develops a conceptual design for the database. It also develops a very simple and easy to design view of data.
* In ER modeling, the database structure is portrayed as a diagram called an entity-relationship diagram.
* ER diagrams are created based on three basic concepts: entities, attributes and relationships.

**For example,** Suppose we design a school database. In this database, the student will be an entity with attributes like address, name, id, age, etc. The address can be another entity with attributes like city, street name, pin code, etc and there will be a relationship between them.



## Component of ER Diagram

## DBMS ER model concept

### **1. Entity:**

An entity may be any object, class, person or place. In the ER diagram, an entity can be represented as rectangles.

Consider an organization as an example- manager, product, employee, department etc. can be taken as an entity.



**a. Weak Entity**

An entity that depends on another entity called a weak entity. The weak entity doesn't contain any key attribute of its own. The weak entity is represented by a double rectangle.



### **2. Attribute**

The attribute is used to describe the property of an entity. Eclipse is used to represent an attribute.C++ vs Java

**For example,** id, age, contact number, name, etc. can be attributes of a student.



**a. Key Attribute**

The key attribute is used to represent the main characteristics of an entity. It represents a primary key. The key attribute is represented by an ellipse with the text underlined.



**b. Composite Attribute**

An attribute that composed of many other attributes is known as a composite attribute. The composite attribute is represented by an ellipse, and those ellipses are connected with an ellipse.



**c. Multivalued Attribute**

An attribute can have more than one value. These attributes are known as a multivalued attribute. The double oval is used to represent multivalued attribute.

**For example,** a student can have more than one phone number.



**d. Derived Attribute**

An attribute that can be derived from other attribute is known as a derived attribute. It can be represented by a dashed ellipse.

**For example,** A person's age changes over time and can be derived from another attribute like Date of birth.



### **3. Relationship**

A relationship is used to describe the relation between entities. Diamond or rhombus is used to represent the relationship.



Types of relationship are as follows:

**a. One-to-One Relationship**

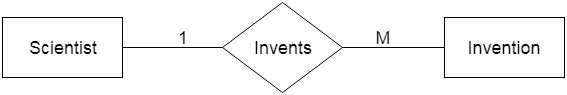
When only one instance of an entity is associated with the relationship, then it is known as one to one relationship.

**For example,** A female can marry to one male, and a male can marry to one female.

  
**b. One-to-many relationship**

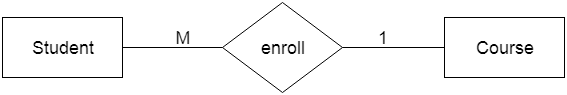
When only one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then this is known as a one-to-many relationship.

**For example,** Scientist can invent many inventions, but the invention is done by the only specific scientist.

  
**c. Many-to-one relationship**

When more than one instance of the entity on the left, and only one instance of an entity on the right associates with the relationship then it is known as a many-to-one relationship.

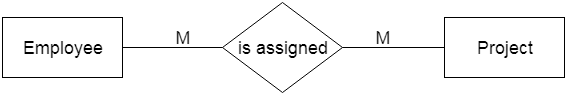
**For example,** Student enrolls for only one course, but a course can have many students



**d. Many-to-many relationship**

When more than one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then it is known as a many-to-many relationship.

**For example,** Employee can assign by many projects and project can have many employees.

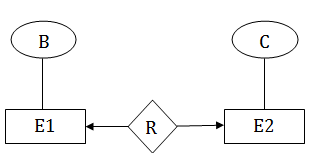


# **Mapping Constraints**

* A mapping constraint is a data constraint that expresses the number of entities to which another entity can be related via a relationship set.
* It is most useful in describing the relationship sets that involve more than two entity sets.
* For binary relationship set R on an entity set A and B, there are four possible mapping cardinalities. These are as follows:
  1. One to one (1:1)
  2. One to many (1:M)
  3. Many to one (M:1)
  4. Many to many (M:M)

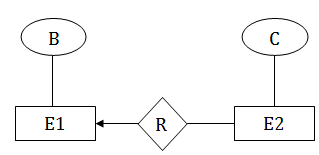
## One-to-one

In one-to-one mapping, an entity in E1 is associated with at most one entity in E2, and an entity in E2 is associated with at most one entity in E1.



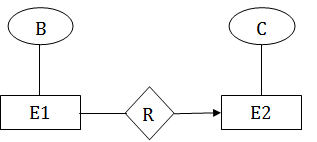
## One-to-many

In one-to-many mapping, an entity in E1 is associated with any number of entities in E2, and an entity in E2 is associated with at most one entity in E1.



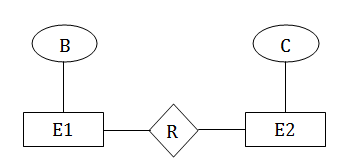
## Many-to-one

In one-to-many mapping, an entity in E1 is associated with at most one entity in E2, and an entity in E2 is associated with any number of entities in E1.



## Many-to-many

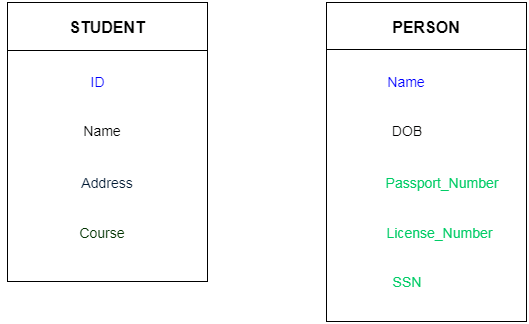
In many-to-many mapping, an entity in E1 is associated with any number of entities in E2, and an entity in E2 is associated with any number of entities in E1.



# **Keys**

* Keys play an important role in the relational database.
* It is used to uniquely identify any record or row of data from the table. It is also used to establish and identify relationships between tables.

**For example:** In Student table, ID is used as a key because it is unique for each student. In PERSON table, passport\_number, license\_number, SSN are keys since they are unique for each person.



## Types of key : DBMS Keys

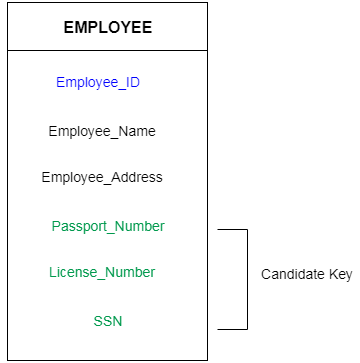
### **1. Primary key**

* A primary key is a field in a table which uniquely identifies each row/record in a database table.
* Primary keys must contain unique values.
* A primary key column cannot have NULL values.

### **2. Candidate key**

* 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.

**For example:** In the EMPLOYEE table, id is best suited for the primary key. Rest of the attributes like SSN, Passport\_Number, and License\_Number, etc. are considered as a candidate key.



### **3. Super Key**

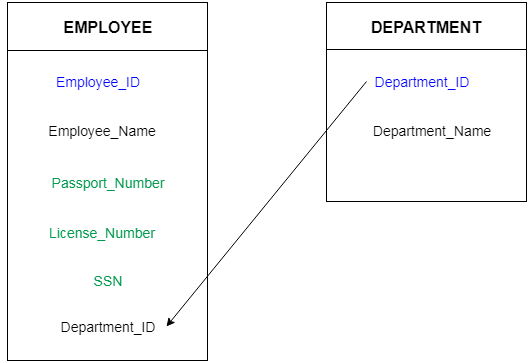
Super key is a set of an attribute which can uniquely identify a tuple. Super key is a superset of a candidate key.

**For example:** In the above EMPLOYEE table, for(EMPLOEE\_ID, EMPLOYEE\_NAME) the name of two employees can be the same, but their EMPLYEE\_ID can't be the same. Hence, this combination can also be a key.

The super key would be EMPLOYEE-ID, (EMPLOYEE\_ID, EMPLOYEE-NAME), etc.

### **4. Foreign key**

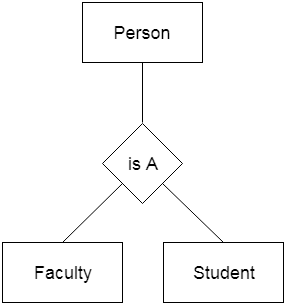
* Foreign keys are the column of the table which is used to point to the primary key of another table.
* We add the primary key of the DEPARTMENT table, Department\_Id as a new attribute in the EMPLOYEE table.
* Now in the EMPLOYEE table, Department\_Id is the foreign key, and both the tables are related.



# **Generalization**

* Generalization is like a bottom-up approach in which two or more entities of lower level combine to form a higher level entity if they have some attributes in common.
* In generalization, an entity of a higher level can also combine with the entities of the lower level to form a further higher level entity.
* In generalization, entities are combined to form a more generalized entity, i.e., subclasses are combined to make a superclass.

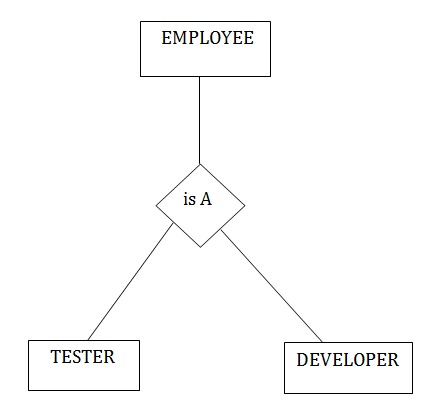
**For example,** Faculty and Student entities can be generalized and create a higher level entity Person.



# **Specialization**

* Specialization is a top-down approach, and it is opposite to Generalization. In specialization, one higher level entity can be broken down into two lower level entities.
* Specialization is used to identify the subset of an entity set that shares some distinguishing characteristics.
* Normally, the superclass is defined first, the subclass and its related attributes are defined next, and relationship set are then added.

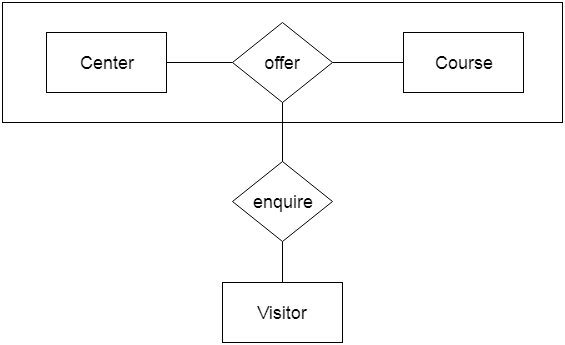
**For example:** In an Employee management system, EMPLOYEE entity can be specialized as TESTER or DEVELOPER based on what role they play in the company.



# **Aggregation**

In aggregation, the relation between two entities is treated as a single entity. In aggregation, relationship with its corresponding entities is aggregated into a higher level entity.

**For example:**  If a visitor visits a coaching center then he will never enquiry about the Course only or just about the Center instead he will ask the enquiry about both.

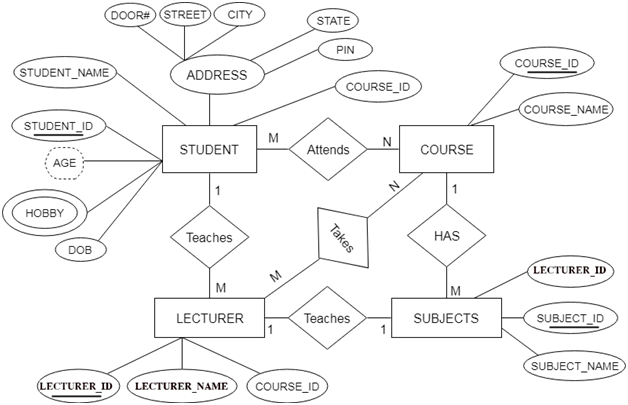


# **Reduction of ER diagram to Table**

The database can be represented using the notations, and these notations can be reduced to a collection of tables.

In the database, every entity set or relationship set can be represented in tabular form.

**The ER diagram is given below:**



There are some points for converting the ER diagram to the table:

* **Entity type becomes a table.**

In the given ER diagram, LECTURE, STUDENT, SUBJECT and COURSE forms individual tables.

* **All single-valued attribute becomes a column for the table.**

In the STUDENT entity, STUDENT\_NAME and STUDENT\_ID form the column of STUDENT table. Similarly, COURSE\_NAME and COURSE\_ID form the column of COURSE table and so on. to find Nth Highest Salary in SQL

* **A key attribute of the entity type represented by the primary key.**

In the given ER diagram, COURSE\_ID, STUDENT\_ID, SUBJECT\_ID, and LECTURE\_ID are the key attribute of the entity.

* **The multivalued attribute is represented by a separate table.**

In the student table, a hobby is a multivalued attribute. So it is not possible to represent multiple values in a single column of STUDENT table. Hence we create a table STUD\_HOBBY with column name STUDENT\_ID and HOBBY. Using both the column, we create a composite key.

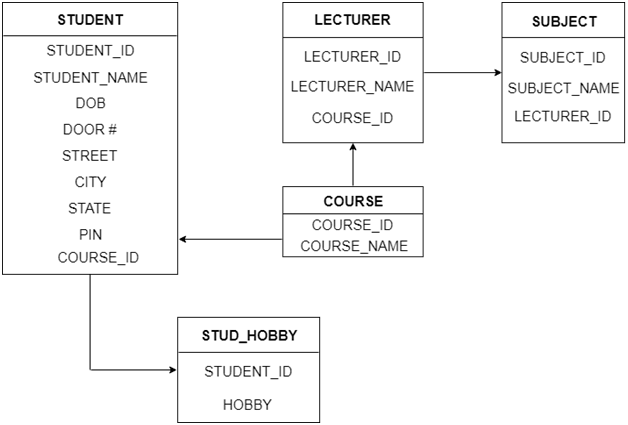
* **Composite attribute represented by components.**

In the given ER diagram, student address is a composite attribute. It contains CITY, PIN, DOOR#, STREET, and STATE. In the STUDENT table, these attributes can merge as an individual column.

* **Derived attributes are not considered in the table.**

In the STUDENT table, Age is the derived attribute. It can be calculated at any point of time by calculating the difference between current date and Date of Birth.

Using these rules, you can convert the ER diagram to tables and columns and assign the mapping between the tables. Table structure for the given ER diagram is as below:



RELATIONAL DATA MODEL

Relational Data Model uses a collection of ‘Tables’ to represent both data and the Relationships among those data.

Each table has multiple columns and each column has unique name.

STUDENT (Relation)

|  |  |  |  |
| --- | --- | --- | --- |
| SID | S\_NAME | S-AGE | S\_CLASS |
| 001 | DEEPAK | 30 | 8 |
| 002 | SANJAY | 29 | 7 |
| 003 | MEHTA | 29 | 7 |
| 004 | RAHUL | 28 | 7 |

Attributes: - SID, S\_Name, S-AGE, S\_CLASS

Tuples: - 003, Mehta, 29, 7, etc...

The data is arranged in a relation which is visually represented in a Two Dimensional Table.

The data is inserted into the table in the form of tuples (rows). A tuple is formed by one or more than one attributes. Tuple in this e.g is a row (complete row).

Attributes are used as basic building blocks in the formation of various expression that are used to derive the meaningful information.

There can be number of types in the table (relation), but all the type contains fixed and same attributes with varying values.

A Relation is represented by a table,

A tuple is represented by a row

An attribute is represented by a column of the table

Attribute value contains the value for the column in the row.

CONSTRAINTS

(Set of rules and limitation)

1. To select a particular row/tuple from table / relation we Use attribute / columns name with the help of unique value a field of an attribute.

2. This field which are unique from other fields are used

As indexes which helps in searching fast.

The given table has attribute with every unique fields

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| E\_ID | E\_NAME | E\_AGE | E\_GEN | E\_SAL |
| 0006 | Deepak | 29 | M | 18000 |
| 0010 | Ajay | 28 | M | 15000 |
| 0020 | Deepak | 31 | M | 18000 |
| 0310 | Kapil | 42 | M | 17000 |
| 0289 | Rahul | 52 | M | 14000 |
| 0628 | Deepak | 29 | M | 16000 |

3. All the relational algebra operations, like Select Intersection, product union, join, division, merge can also be performed on the relation data model.

4. Operations on RDM (Relational Data Model) are facilitated with the help of different conditional expressions, various key attribute & pre-defined constraints etc.

5. Data Integrity: - It is maintained by process like normalization.

6. Description of data in terms of this model is called a Schema.

7. Schema for relation specified, its name, name of each field

Student (S\_id: Integer,

S\_name: string,

S\_login: string etc.)

RELATIONAL CALCULAS

1. It is non procedural query language.
2. Here user is considered with the details of how to obtain the end results.
3. The relational calculus tells, the system what data to be retrieved from the table but doesn’t tell how to retrieved it.

There are two different types of relational calculus

1. Tuples relational calculus: - variables ranges over tuple like sql.

2. Domain relational calculus: - variable ranges over domain like query by example

NOTE: - Relational calculus is more declarative and it is user defined queries in terms of what they want not in terms of how to compute it.

RELATIONAL ALGEBRA

It is used in the design of transaction and forms the conceptual basis of SQL.

It consists of operators and manipulates which apply to relations and given relation as a result but do not change the actual relation in database.

It is a procedural intermediate language used within the DBMS.

It has different operators namely:- Basic operator: - projector, selector, cross product, union, rename, set difference.

Derived operator: - join, intersect division

JOIN OPERATION

A join clause is used to combine rows from two or more tables based on related column between them. Several operators can be used to join tables, such as =, <, >, <>, <=, =>, != , BETWEEN , LIKE, and NOT; they all are used to join tables . Details are discus in SQL.

Type of SQL JOIN: -

1. (Inner join): - it return records when the both table have matching value.
2. (left outer join): - it return records from the left table , and match record from the right table
3. (right outer join): - it return records from the right table, and match record from the left table

Now a days, data is structured in a very large number of tables and which is needed to join these multiple tables based on logical relationships between them.

INTERGRITY CONSTRAINTS

It is a set of rules used to maintain quality information

It has four types: - 1. Domain constraints

2. Entity Integrity constraints

3. Referential integrity constraints

4. Key constraints

1. Domain constraints: -

|  |  |  |  |
| --- | --- | --- | --- |
| E\_ID | NAME | ROOM NO | AGE |
| P2C-21-062 | BANDANA | 102 | 21 |
| P2C-21-078 | RAHUL | 201 | 34 |
| P2C-21-034 | ATUL | 202 | 22 |
| P2C-21-98 | RITIKA | 204 | A |

IN the above table the alphabetical value will not be allowed in AGE. The column type should be same.

2. Entity Integrity Constraint: - It checks the primary key. If any P.K is missing then it will show the error. Primary key value should not be null.

3. Referential Integrity Constraints: - Suppose we have two table in 1st table P\_no is foreign key but for 2nd table it is a primary key and by using referential integrity we can know about the employee location.

|  |  |  |  |
| --- | --- | --- | --- |
| E\_ID | NAME | AGE | P\_NO |
| 1 | RT | 32 | 979 |
| 2 | GH | 27 | 989 |
| 3 | BX | 24 | 999 |

|  |  |
| --- | --- |
| P\_NO | LOCATION |
| 979 | Mumbai |
| 989 | Noida |
| 999 | Punjab |

1. Key constraints: - Repetition of primary key will not allowed in the table.

|  |  |  |
| --- | --- | --- |
| E\_ID | NAME | ROM |
| T8U | HY | 405 |
| B9P | JP | 560 |
| P2C | ER | 103 |
| P2C | GB | 789 |

The repetition of P2C in last row will show the error.

NORMALISATION

Functional Dependency:-

A specific constraint relation between two set of attribute where the 1st set can determine the value of 2nd set. I.e X 🡪 Y

where x is determinant and y is dependent.

If in two tuple t1.x = t2.y then y value must be same as

t1.y = t2.y

|  |  |
| --- | --- |
| X | Y |
| 1 | 1 |
| 2 | 1 |
| 3 | 2 |
| 4 | 3 |
| 2 | 5 |

Suppose as in given table, t1.x=t2.x are equal then only we need to check t1.y=t2.y. Here t1.x=2, t2.x is also 2 but t1.y not equal as to t2.y. So, this relation is not in F.D.

Every value of X is unique then we can definitely determine the value of Y from X.

X and Y can have set of set of attributes.

|  |  |
| --- | --- |
| X | Y |
| 1 | 1 |
| 2 | 1 |
| 3 | 2 |
| 4 | 3 |

This given table is an example of F.D because here each value of X is unique and we can easily find the value of Y.

Let’s see another example:-

Consider an Employee table is given

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| R.no | Name | Street | Dept. | Course |
| 1 | a | 78 | CS | C1 |
| 2 | b | 60 | EE | C1 |
| 3 | a | 78 | CS | C2 |
| 4 | b | 60 | EE | C2 |
| 5 | c | 80 | IT | C3 |
| 6 | d | 80 | EC | C2 |

F.D: X🡪Y

If t1.x=t2.y then, t1.y=t2.y

(R.NO 🡪Name), ((R.no, name) 🡪Street), (Name🡪Street), (Name, 🡪Dept., all these sets are F.D.

(Name, street)🡪(Dept., Course), (Name🡪R.no),

(Dept. 🡪course), (course 🡪 Dept.) these sets of tuples are not in F.D.

NORMALISATION: - It is a process of making the table free from insert, update and delete anomaly and save by reducing the redundant data or duplicate.

Benefits of using normalization:-

1. To reduce the redundancy from the table.
2. To save the space.
3. To reduce anomalies.
4. Normalization minimize the null value.
5. Searching, sorting and creating index will be faster after applying the normalization.

Normalization has different level:-

1. 1nf
2. 2nf
3. 3nf
4. BCNF
5. 4NF
6. 5NF

1ST NORMAL FORM: - A relation is in 1NF only when it follows the following condition

1. All attribute having atomic domains.
2. Each column contains atomic value.
3. A column should contain value from same domain as like if the Sid value is in integer then every Sid must be in integer value.
4. Each column should have a unique name.
5. No duplicate rows.
6. No ordering of rows and columns.

Let’s see an example

|  |  |  |
| --- | --- | --- |
| Sid | Sector | room |
| 1 | 11,15 | 102 |
| 2 | 44 | 110 |
| 3 | 52 | 201 |

So, the above table is not in 1st NF because it have many multivalued attribute and composite value attribute.

|  |  |
| --- | --- |
| Sid | Sector |
| 1 | 11 |
| 1 | 15 |
| 2 | 44 |
| 3 | 52 |

|  |  |
| --- | --- |
| SID | Room |
| 1 | 102 |
| 2 | 110 |
| 3 | 201 |

The above two tables is now in 1NF and the value of each attribute contains only a single value it also one to many relation.

NOTE: - When ER diagram is converted into relational schema then definitely that will be in 1NF by default.

2nd Normal Form: -

A relation will only be in 2nf when it meets the below two condition

1. It should be in 1NF.
2. No partial dependency in the relation.
3. Occur when there is composite key.

The given table below shows how a relation will convert in 2nd:-

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sid | Name | Marks | Course | Teacher |
| 201 | Ravi | 17 | CSS | A |
| 202 | Suraj | 17 | C++ | B |
| 201 | Saras | 15 | HTML | C |
| 204 | Alex | 16 | J.S | D |
| 205 | Ram | 16 | C++ | E |

NON KEY COLUMN

Composite key = Sid + course

Now, take away the partially dependent to new table so, the table will be in 2nd normal form.

|  |  |
| --- | --- |
| Course | Teacher |
| 201 | A |
| 202 | B |
| 201 | C |
| 204 | D |
| 205 | E |

Here, teacher is partially depend on course.

3RD NORMAL FORM

A table will only be in 3rd normal form if it has the following three property:-

1. It should be in 1nf , 2nf
2. No TRANSITIVE DEPENDENCY
3. Occurs when you guess value of any column from Non key column.

So, considering an example we will understand about 3rd normal form.

Considering the data of student: -

|  |  |  |  |
| --- | --- | --- | --- |
| Sid | Name | Examtype | Maxmarks |
| 201 | Saghir | Viva | 20 |
| 202 | Harris | Theory | 100 |
| 203 | Maxwell | practical | 50 |
| 204 | Bandy | Practical | 50 |
| 205 | Simon | viva | 20 |
| 206 | Sam | Theory | 100 |
| 207 | Jim | Theory | 100 |

Here we can see that if any new students want to take the theory paper then we can guess that the marks will be 100. So, here we can use the 3rd normal form.

Now, we can separate the column which we guess and split it.

|  |  |
| --- | --- |
| Examtype | Marks |
| Viva | 20 |
| Theory | 100 |
| practical | 50 |

BCNF: - It is a special case of 3RD normal form and the L.H.S of every functional dependency should be candidate key of super key.

Ex: - Let’s considered employees details: -

|  |  |  |  |
| --- | --- | --- | --- |
| Room | name | Id | age |
| 1 | Ravi | P2C | 20 |
| 2 | Kiran | P1C | 19 |
| 3 | Preety | P9E | 34 |
| 4 | garima | K8I | 24 |

Candidate Key: - {room, id}

Foreign key: -{room🡪name},{room🡪id},{id🡪age},{id🡪room}

4NF

It should be in BCNF and it should not have Multivalued Dependency.

4nf is a level of database normalization where there are non- trivial multivalued dependencies other than a candidate key.

AS in the below table MVD exists so this table is suffering from all anomalies so to solve this splitting of table is required.

|  |  |  |
| --- | --- | --- |
| course | instructor | Textbook |
| Eng. | White  Green  black | Python  C++ |
| CSE | Blue | Basic C  JAVA |

Course🡪🡪instructor

Course🡪🡪textbook

|  |  |
| --- | --- |
| COURSE | Instructor |
| Eng. | White |
| Eng. | Green |
| Eng. | Black |
| CSE | Blue |

|  |  |
| --- | --- |
| Course | Textbook |
| Eng. | Python |
| Eng. | C++ |
| CSE | Basic C |
| CSE | Java |

In this way we have resolved our multivalued resolution.

5NF: - A relation is in 5nf if it is in 4nf and not having any join dependency and joining should be lossless.

A relation that has a join dependency cannot be divided into two (or more) relation such that the resulting tables can be combined to form the original table.

|  |  |  |
| --- | --- | --- |
| subject | professor | semester |
| Java | Amit | 5 |
| C | Asha | 3 |
| DBMS | Asha | 5 |
| DSA | Neha | 3 |
| DSA | Sumit | 3 |

|  |  |
| --- | --- |
| Subject | Prof. |
| java | Amit |
| C | Asha |
| DBMS | Asha |
| DSA | Neha |
| DSA | Sumit |

|  |  |
| --- | --- |
| Sub | Sem. |
| JAVA | 5 |
| C | 3 |
| DBMS | 5 |
| DSA | 3 |
| DSA | 3 |

|  |  |
| --- | --- |
| Professor | Sem. |
| Amit | 5 |
| Asha | 3 |
| Asha | 5 |
| Neha | 3 |
| Sumit | 3 |

|  |  |  |
| --- | --- | --- |
| SUB | PROF. | SEM. |
| JAVA | Amit | 5 |
| C | Asha | 3 |
| C | Asha | 5 |
| DBMS | Asha | 3 |
| DBMS | Asha | 5 |
| DSA | Neha | 3 |
| DSA | Sumit | 5 |

Let 1st table be R and then decompose it into table R1, R2, R3, and rejoining of R1, R2, and R3 create a new table named a Table 4. Here in this table we see that row 3 and row 4 is the extra row which is not in the actual table. So after removing these two row we will get the actual Table and now these table cannot further be breakable.

RELATIONAL DECOMPOSITION

Let R be a relation schema A set of relation schema (R1, R2, R3……Rn) is a decomposition of R if

R = R1 U R2 U R3……Rn.

Ex:-Relation R(x, y, z) there can be two subset R1(X, Y) & R2(Y, Z) we get, R=R1 U R2.

TYPES OF DECOMPOSITION:-

1. Lossless Decomposition
2. Dependency preserving

Lossless Decomposition: - A decomposition {R1, R2 ....Rn}

of a relation R is called lossless decomposition for R .

When we can say this lossless decomposition

If the natural join of R1, R2....Rn produces exactly the relation R.

Let a relation R (A, B, C) and after the decomposition it gives R1 (A, B) and R2 (A, C) then the

R’ will be R’ (A, B, C) thus R = R’ then we can say the relation is in lossless decomposition.

NOTE: -This is effective in removing redundancy from database while preserving the original data.

DEPENDENCY PRESERVING: -

FD = {A🡪B, B🡪C, A🡪D}

R (A, B, C)

R1 (A B) R2 (B C D)

R1 = (A B) It gives A🡪B (It’s not holding A🡪D)

R2 = (B C D) It gives B🡪C (it’s not holding A🡪D)

The above two relation only give the 2 dependency but not the A🡪D and also after taking the union of these relation it will not give the A🡪D.

Here, one functional dependency is lost so, this decomposition is not dependency preserving.

MULTIVALED DEPENDENCY

A multivalued dependency (MD) arises when a relation R having non- atomic attribute is converted to a normalized form.

Ex: - R\_author

|  |  |  |
| --- | --- | --- |
| ISBN | A\_ID | Phone |
| 001 | A001  A003  A001 | 925  923  911 |
| 002 | A005 | 911  912  913 |

|  |  |  |
| --- | --- | --- |
| ISBN | A\_id | Phone |
| 001 | A001 | 925 |
| 001 | A003 | 923 |
| 002 | A005 | 911 |
| 002 | A005 | 912 |
| 002 | A005 | 913 |
| 001 | A001 | 924 |

(ISBN 🡪🡪 A\_ID), (A\_ID 🡪🡪Phone) are multivalued dependency.

JOIN DEPENDENCY

Generalization of Multivalued dependency is known as JOIN DEPENDENCY. If {r1, r2…rn} hold over a relation R if r1, r2, r3 …., rn is lossless join decomposition of r. There is no set of sound and complete inference rule for JD.

OR

If a table can be recreated by joining multiple tables and each table is have a subset of the attributes of table, then that is in join dependency.

INCLUSION DEPENDENCE

To design a database inclusion dependency has insignificant influence on it. Inclusion is a statement in which few of the columns of a relation are in other columns. Generally, it is based on keys.

Ex: - Foreign key

**Transaction Processing**

The transaction is a collection of logically related operations by which the database will reach from one consistency state to another consistency state.

**ACID Properties in DBMS**

ACID(atomicity, consistency, isolation, durability) is a set of properties that guarantee database transactions are processed reliability. In the context of databases, a single logical operation on the data is called a transaction . For example, a transfer of funds from one bank to another, even though that might involve multiple changes(such as debiting one account and crediting another) is a single transaction.

Property of Transaction

1. Atomicity
2. Consistency
3. Isolation
4. Durability

Atomicity

Atomicity requires the database modifications must follow an “all or nothing” rule. Each transaction is said to atomic. If one part of the transaction fails, the entire transaction fails and the database state is left unchanged. An atomic transfer cannot be subdivided and must be processed in its entirely or not at all. Atomicity means that the user do not have to worry about the effect of incomplete transactions.

Consistency

The consistency property ensures that any transaction the database performs will take it from one consistent state to another consistent state.

**For example:** The total amount must be maintained before or after the transaction.

1. Total before T occurs = 600+300=900
2. Total after T occurs= 500+400=900

Isolation

Isolation refers to the requirement that other operations cannot access data that has been modified during a transaction that has not yet completed. The question of isolation occurs in case of concurrent transactions (multiple transactions occurring at same time) . Each transaction must remain unaware of other concurrently executing transactions, except that one transaction may be forced to wait for the completion of another transaction that has modified data that the waiting transaction requires.

Durability

Durability is the ability of the DBMS to recover the committed transaction updates against any kind of system failure. Durability is the DBMS’s guarantee that once the user has been notified of a transaction’s success the transaction will not be lost, the transaction’s data changes will survive the system failure, and that all integrity constraints have been satisfied, so the DBMS won’t need to reverse the transaction.

# States of Transaction

Following are the different states of database:

Commit

Transaction

Begin

Transaction

Partially

Committed

Committed

Active

Abort

Abort

Failed

Terminated

### **Active state**

The active state is the first state of every transaction. In this state, the transaction is being executed.

### **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.

### **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.

### **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.

### **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:
  1. Re-start the transaction
  2. Kill the transaction

**Schedule**

A series of operation from one transaction to another transaction is known as schedule. It is used to preserve the order of the operation in each of the individual transaction.

Schedule

Serializable

Schedule

Serial

Schedule

Non-Serial

Schedule

**Schedule**

It is chronological execution sequence of multiple transactions.

1. **Serial Schedule :** Only one transaction at a time.
2. **Parallel Schedule :** Multiple transactions at a time.

(Note: **Throughput**: Number of transactions executed per unit time)

**Read Write Conflict or Unrepeatable read**

T2

T1

R(A)

**Conflict**

**occur**

Commit

W(A)

Commit

W(A)

A=A-1

R(A)

A=A-1

**Irrecoverable Schedule**

T2

T1

R(A)

**Serializability**

rollback

Fail

R(B)

Commit

W(A)

A=A-2

R(A)

W(A)

A=A-5

We have a parallel schedule and we have to find a clone serial schedule which is equivalent to given parallel schedule.

**Conflict Equivalent**

|  |  |  |
| --- | --- | --- |
| R(A) | W(A) | Conflict  Pairs |
| W(A) | R(A) |
| W(A) | W(A) |

|  |  |  |
| --- | --- | --- |
| R(B) | R(A) | Non-Conflict |
| W(B) | W(A) |
| R(B) | W(A) |
| W(A) | W(B) |

**Precedence Graph**

Check conflict pairs in other transactions and draw edges

T3

T2

T1

R(x)

T2 -> T3 ->T1

No Loop/Cycle

So it is a Conflict Serializable

Thus, it is serializable and consistent also.

T1

T2

T3

W(x)

W(z)

R(z)

R(x)

R(y)

R(y)

W(z)

R(z)

**View Serializability**

Check whether schedule is conflict serializable or not

T3

T2

T1

R(A)

Loop/Cycle Present

So it is a not Conflict Serializable

What leads to the rise of view serializability

W(A)

W(A)

W(A)

T3

T1

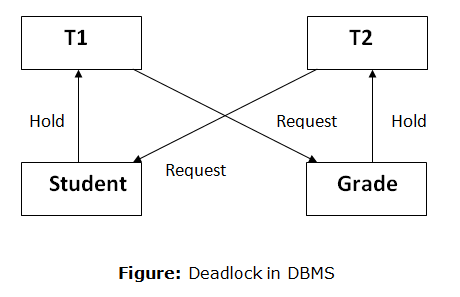
T2

# Deadlock in DBMS

A deadlock is a condition where two or more transactions are waiting indefinitely for one another to give up locks. Deadlock is said to be one of the most feared complications in DBMS as no task ever gets finished and is in waiting state forever.

**For example:** In the student table, transaction T1 holds a lock on some rows and needs to update some rows in the grade table. Simultaneously, transaction T2 holds locks on some rows in the grade table and needs to update the rows in the Student table held by Transaction T1.

Now, the main problem arises. Now Transaction T1 is waiting for T2 to release its lock and similarly, transaction T2 is waiting for T1 to release its lock. All activities come to a halt state and remain at a standstill. It will remain in a standstill until the DBMS detects the deadlock and aborts one of the transactions.



**Concurrency Control**

**Shared Exclusive Locking**

A **database lock** is **used** to “**lock**” some data in a **database** so that only one **database** user/session may update that particular data. So, **database locks** exist to prevent two or more **database** users from updating the same exact piece of data at the same exact time.

**Shared lock(s)** : if the transaction locked data item in shared mode then allowed to read only.

**Exclusive lock**: if transaction locked data item in exclusive mode then allowed to read and write both.

Locking provides serializable schedule.

Request🡪

Grant->

|  |  |  |
| --- | --- | --- |
|  | Shared Lock | Exclusive Lock |
| Shared lock | Yes | No |
| Exclusive Lock | No | No |

|  |
| --- |
| Problems in Serializable Schedule: |
| 1. May not sufficient to produce only serializable schedule |
| 1. May not free from ir-recoverability |
| 1. May not be free from deadlock(infinite waiting) |
| 1. May not be free from starvation(due to finite waiting) |

# Lock-Based Protocol

**2-Phase Locking(2PL)**

**Growing phase:** locks are acquired and no lock are released.

**Shrinking phase:** locks are released and no locks are required.

|  |  |
| --- | --- |
| T1 | Growing cells |
| X(A) |
| S(B) |
| R(A) |
| W(A) |
| R(B) |
| S(A) |
| R(C) |
| S(A) |
| R(A) |
| U(A) | Shrinking Phase |

Lock-point: where first lock is released or first unlock starts.

**2 PL( 2 Phase Locking)**

**Advantages:**

2-PL always ensures the serializability

**Disadvantages:**

|  |
| --- |
| 1. May not free from ir-recoverability |
| 1. May not be free from deadlock |
| 1. May not be free from starvation 2. May not be from cascading rollback. |

**Strict 2PL**: It should satisfy the basic 2PL and all exclusive locks should hold until commit/abort. It is cascadeless and strict recoverable.

**Regret 2PL**: it should satisfy the basic 2PL and all shared, exclusive locks should hold until commit/abort.

**Timestamp Ordering Protocol**

* The Timestamp Ordering Protocol is used to order the transactions based on their Timestamps. The order of transaction is nothing but the ascending order of the transaction creation.
* The priority of the older transaction is higher that's why it executes first. To determine the timestamp of the transaction, this protocol uses system time or logical counter.
* The lock-based protocol is used to manage the order between conflicting pairs among transactions at the execution time. But Timestamp based protocols start working as soon as a transaction is created.
* Let's assume there are two transactions T1 and T2. Suppose the transaction T1 has entered the system at 007 times and transaction T2 has entered the system at 009 times. T1 has the higher priority, so it executes first as it is entered the system first.
* The timestamp ordering protocol also maintains the timestamp of last 'read' and 'write' operation on a data.

**Basic Timestamp ordering protocol works as follows:**

1. Check the following condition whenever a transaction Ti issues a **Read (X)** operation:

* If W\_TS(X) >TS(Ti) then the operation is rejected.
* If W\_TS(X) <= TS(Ti) then the operation is executed.
* Timestamps of all the data items are updated.

2. Check the following condition whenever a transaction Ti issues a **Write(X)** operation:

* If TS(Ti) < R\_TS(X) then the operation is rejected.
* If TS(Ti) < W\_TS(X) then the operation is rejected and Ti is rolled back otherwise the operation is executed.

**Where,**

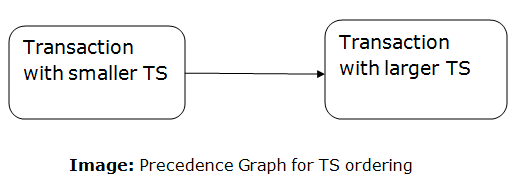
**TS(TI)** denotes the timestamp of the transaction Ti.

**R\_TS(X)** denotes the Read time-stamp of data-item X.

**W\_TS(X)** denotes the Write time-stamp of data-item X.

Advantages and Disadvantages of TO protocol:

* TO protocol ensures serializability since the precedence graph is as follows:



* TS protocol ensures freedom from deadlock that means no transaction ever waits.
* But the schedule may not be recoverable and may not even be cascade- free.

**File Organization**

* The **File** is a collection of records. Using the primary key, we can access the records. The type and frequency of access can be determined by the type of file organization which was used for a given set of records.
* File organization is a logical relationship among various records. This method defines how file records are mapped onto disk blocks.
* File organization is used to describe the way in which the records are stored in terms of blocks, and the blocks are placed on the storage medium.
* The first approach to map the database to the file is to use the several files and store only one fixed length record in any given file. An alternative approach is to structure our files so that we can contain multiple lengths for records.
* Files of fixed length records are easier to implement than the files of variable length records.

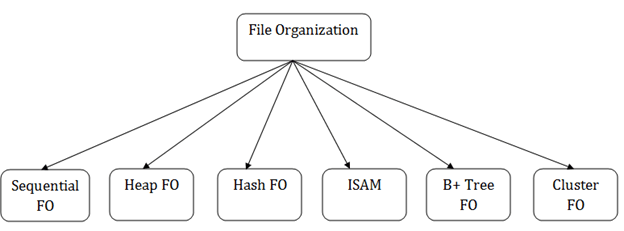
Objective of file organization

* It contains an optimal selection of records, i.e., records can be selected as fast as possible.
* To perform insert, delete or update transaction on the records should be quick and easy.
* The duplicate records cannot be induced as a result of insert, update or delete.
* For the minimal cost of storage, records should be stored efficiently.

Types of file organization:

File organization contains various methods. These particular methods have pros and cons on the basis of access or selection. In the file organization, the programmer decides the best-suited file organization method according to his requirement.

Types of file organization are as follows:



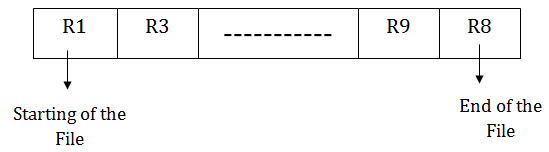
* [Sequential file organization](https://www.javatpoint.com/dbms-sequential-file-organization)
* [Heap file organization](https://www.javatpoint.com/dbms-heap-file-organization)
* [Hash file organization](https://www.javatpoint.com/dbms-hash-file-organization)
* [B+ file organization](https://www.javatpoint.com/dbms-b-plus-file-organization)
* [Indexed sequential access method (ISAM)](https://www.javatpoint.com/dbms-indexed-sequential-access-method)
* [Cluster file organization](https://www.javatpoint.com/dbms-cluster-file-organization)

**Sequential File Organization**

This method is the easiest method for file organization. In this method, files are stored sequentially. This method can be implemented in two ways:

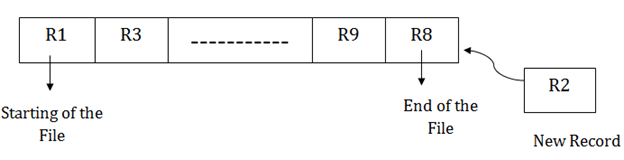
1. Pile File Method:

* It is a quite simple method. In this method, we store the record in a sequence, i.e., one after another. Here, the record will be inserted in the order in which they are inserted into tables.
* In case of updating or deleting of any record, the record will be searched in the memory blocks. When it is found, then it will be marked for deleting, and the new record is inserted.



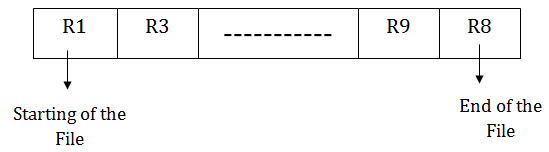
Insertion of the new record:

Suppose we have four records R1, R3 and so on upto R9 and R8 in a sequence. Hence, records are nothing but a row in the table. Suppose we want to insert a new record R2 in the sequence, then it will be placed at the end of the file. Here, records are nothing but a row in any table.



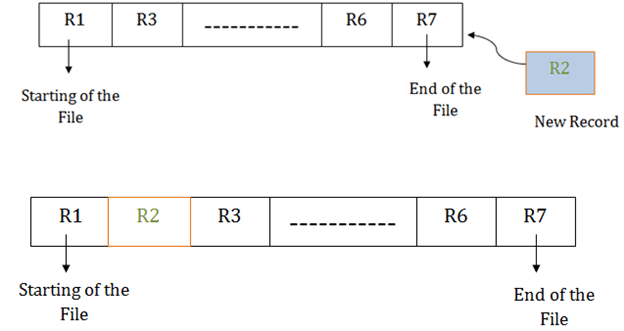
2. Sorted File Method:

* In this method, the new record is always inserted at the file's end, and then it will sort the sequence in ascending or descending order. Sorting of records is based on any primary key or any other key.
* In the case of modification of any record, it will update the record and then sort the file, and lastly, the updated record is placed in the right place.



Insertion of the new record:

Suppose there is a pre-existing sorted sequence of four records R1, R3 and so on upto R6 and R7. 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.



Pros of sequential file organization

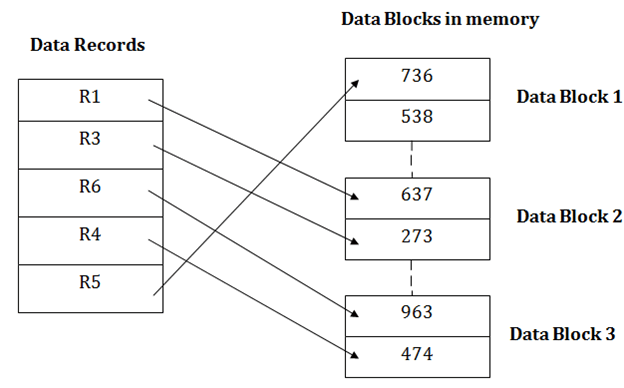
* It contains a fast and efficient method for the huge amount of data.
* In this method, files can be easily stored in cheaper storage mechanism like magnetic tapes.
* It is simple in design. It requires no much effort to store the data.
* This method is used when most of the records have to be accessed like grade calculation of a student, generating the salary slip, etc.
* This method is used for report generation or statistical calculations.

Cons of sequential file organization

* It will waste time as we cannot jump on a particular record that is required but we have to move sequentially which takes our time.
* Sorted file method takes more time and space for sorting the records.

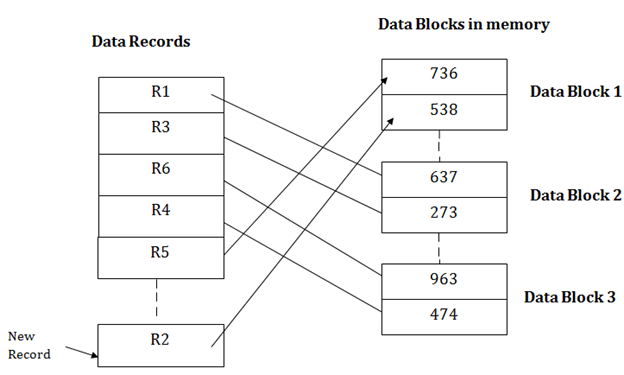
**Heap file organization**

* It is the simplest and most basic type of organization. It works with data blocks. In heap file organization, the records are inserted at the file's end. When the records are inserted, it doesn't require the sorting and ordering of records.
* When the data block is full, the new record is stored in some other block. This new data block need not to be the very next data block, but it can select any data block in the memory to store new records. The heap file is also known as an unordered file.
* In the file, every record has a unique id, and every page in a file is of the same size. It is the DBMS responsibility to store and manage the new records.



Insertion of a new record

Suppose we have five records R1, R3, R6, R4 and R5 in a heap and suppose we want to insert a new record R2 in a heap. If the data block 3 is full then it will be inserted in any of the database selected by the DBMS, let's say data block 1.



If we want to search, update or delete the data in heap file organization, then we need to traverse the data from staring of the file till we get the requested record.

If the database is very large then searching, updating or deleting of record will be time-consuming because there is no sorting or ordering of records. In the heap file organization, we need to check all the data until we get the requested record.

Pros of Heap file organization

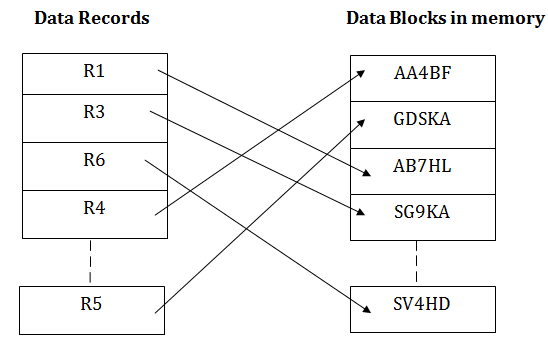
* It is a very good method of file organization for bulk insertion. If there is a large number of data which needs to load into the database at a time, then this method is best suited.
* In case of a small database, fetching and retrieving of records is faster than the sequential record.

Cons of Heap file organization

* This method is inefficient for the large database because it takes time to search or modify the record.
* This method is inefficient for large databases.

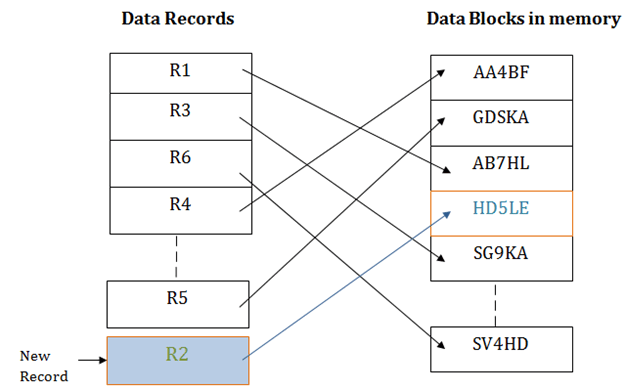
# Hash File Organization

Hash File Organization uses the computation of hash function on some fields of the records. The hash function's output determines the location of disk block where the records are to be placed.



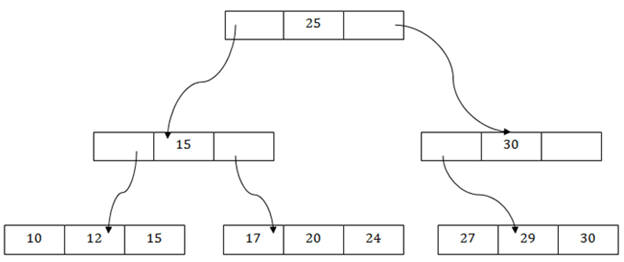
When a record has to be received using the hash key columns, then the address is generated, and the whole record is retrieved using that address. In the same way, when a new record has to be inserted, then the address is generated using the hash key and record is directly inserted. The same process is applied in the case of delete and update.

In this method, there is no effort for searching and sorting the entire file. In this method, each record will be stored randomly in the memory.



B+ File Organization

* B+ tree file organization is the advanced method of an indexed sequential access method. It uses a tree-like structure to store records in File.
* It uses the same concept of key-index where the primary key is used to sort the records. For each primary key, the value of the index is generated and mapped with the record.
* The B+ tree is similar to a binary search tree (BST), but it can have more than two children. In this method, all the records are stored only at the leaf node. Intermediate nodes act as a pointer to the leaf nodes. They do not contain any records.



The above B+ tree shows that:

* There is one root node of the tree, i.e., 25.
* There is an intermediary layer with nodes. They do not store the actual record. They have only pointers to the leaf node.
* The nodes to the left of the root node contain the prior value of the root and nodes to the right contain next value of the root, i.e., 15 and 30 respectively.
* There is only one leaf node which has only values, i.e., 10, 12, 17, 20, 24, 27 and 29.
* Searching for any record is easier as all the leaf nodes are balanced.
* In this method, searching any record can be traversed through the single path and accessed easily.

Pros of B+ tree file organization

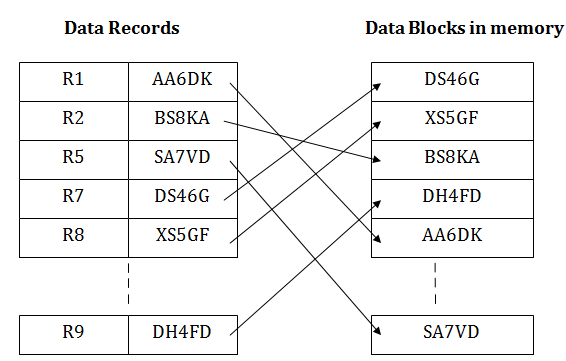
* In this method, searching becomes very easy as all the records are stored only in the leaf nodes and sorted the sequential linked list.
* Traversing through the tree structure is easier and faster.
* The size of the B+ tree has no restrictions, so the number of records can increase or decrease and the B+ tree structure can also grow or shrink.
* It is a balanced tree structure, and any insert/update/delete does not affect the performance of tree.

Cons of B+ tree file organization

* This method is inefficient for the static method.

**Indexed sequential access method (ISAM)**

ISAM method is an advanced sequential file organization. In this method, records are stored in the file using the primary key. An index value is generated for each primary key and mapped with the record. This index contains the address of the record in the file.



If any record has to be retrieved based on its index value, then the address of the data block is fetched and the record is retrieved from the memory.

Pros of ISAM:

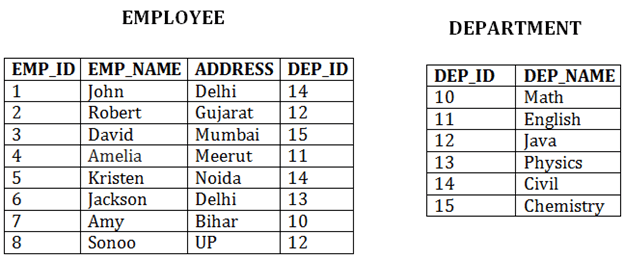
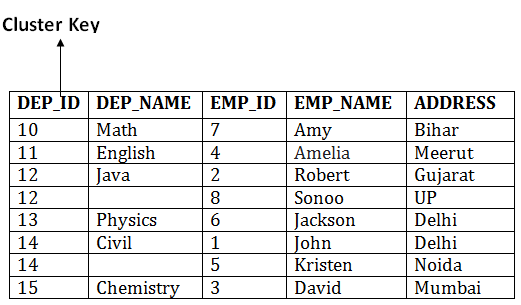
* In this method, each record has the address of its data block, searching a record in a huge database is quick and easy.
* This method supports range retrieval and partial retrieval of records. Since the index is based on the primary key values, we can retrieve the data for the given range of value. In the same way, the partial value can also be easily searched, i.e., the student name starting with 'JA' can be easily searched.

Cons of ISAM

* This method requires extra space in the disk to store the index value.
* When the new records are inserted, then these files have to be reconstructed to maintain the sequence.
* When the record is deleted, then the space used by it needs to be released. Otherwise, the performance of the database will slow down.

**Cluster file organization**

* When the two or more records are stored in the same file, it is known as clusters. These files will have two or more tables in the same data block, and key attributes which are used to map these tables together are stored only once.
* This method reduces the cost of searching for various records in different files.
* The cluster file organization is used when there is a frequent need for joining the tables with the same condition. These joins will give only a few records from both tables. In the given example, we are retrieving the record for only particular departments. This method can't be used to retrieve the record for the entire department.

In this method, we can directly insert, update or delete any record. Data is sorted based on the key with which searching is done. Cluster key is a type of key with which joining of the table is performed.

Types of Cluster file organization:

Cluster file organization is of two types:

1. Indexed Clusters:

In indexed cluster, records are grouped based on the cluster key and stored together. The above EMPLOYEE and DEPARTMENT relationship is an example of an indexed cluster. Here, all the records are grouped based on the cluster key- DEP\_ID and all the records are grouped.

2. Hash Clusters:

It is similar to the indexed cluster. In hash cluster, instead of storing the records based on the cluster key, we generate the value of the hash key for the cluster key and store the records with the same hash key value.

Pros of Cluster file organization

* The cluster file organization is used when there is a frequent request for joining the tables with same joining condition.
* It provides the efficient result when there is a 1:M mapping between the tables.

Cons of Cluster file organization

* This method has the low performance for the very large database.
* If there is any change in joining condition, then this method cannot use. If we change the condition of joining then traversing the file takes a lot of time.
* This method is not suitable for a table with a 1:1 condition.

**Indexing and B+ Tree:-**

* ***Indexing in Databases*:-**
* Indexing is the way to fast to access the database by taking the recode with there point address and it is a additional way to access the file.
* If indexing file is smaller to size as to compare to the main file then the searching time is less. Its means that memory is directionally proportional to time.
* The two way to decrease the size of index are:-

1. Store only attribute and pointer address.
2. Store important data.

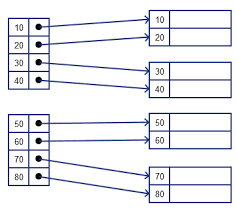
Vxcz\*

* **There are mainly 4 types of indexing methods are**:-

1. Primary Indexing.
2. Secondary Indexing.
3. Cluster Indexing.
4. Multilevel Indexing.
5. **Primary Indexing** :- Main database is shorted and will be access on the primary key , it will be search on primary key.

**Primary Indexing is further divided into two types are:-**

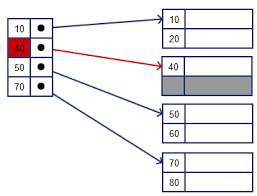
1. Dense Index.
2. Sparse Index.
3. **Dense Index** :- In dense index, every search value in the database, there is an index recode. There recode contains the search key.



**Dense Index**

* When database is not shorted in proper way then we use dense index because every data is recode in there address.

1. **Sparse Index :-** In sparse index, not every data is recode only few value in the index. That record point to by the index recode.



* When database is not shorted in proper way then sparse index is failed.

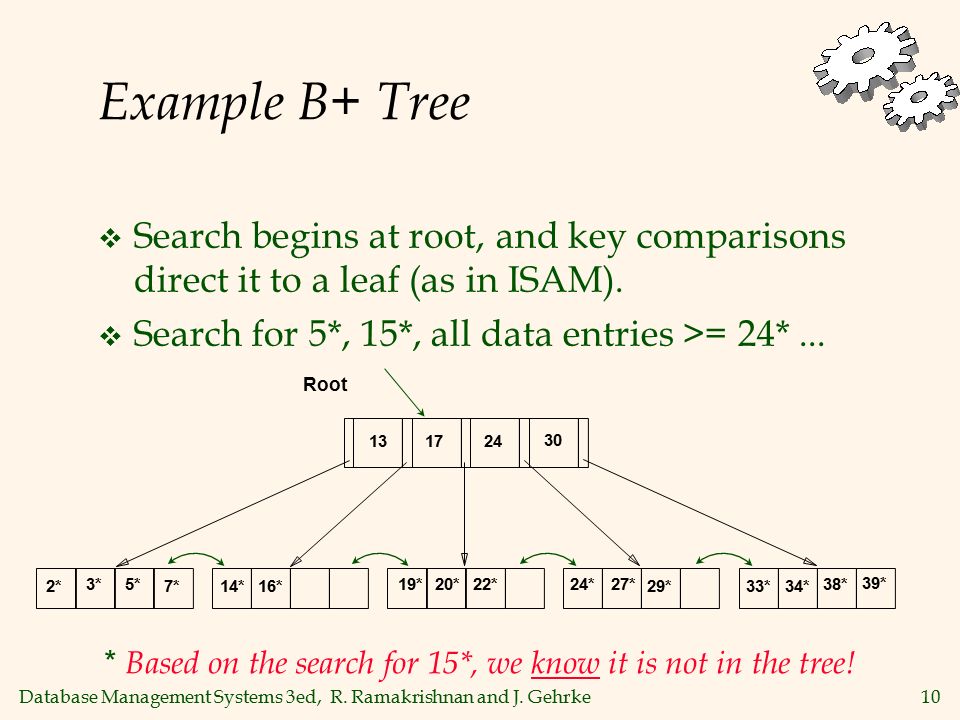
1. **Secondary Indexing:-** Main database is not shorted then we recode all attribut in index.
2. **Cluster Indexing :-** Main database is shorted and it is access will done on non-key attribute, it will search non-key and value is repeted.
3. **Multi-level Indexing** :- when indexing is done more then one time.

* ***B+Tree*:-**

Is is a balanced tree search tree. Leaf node denote actual data points. It ensures that all leaf nodes remain at the same height. It is the leaf nodes are linked using a link list. Therefore, a B+ tree can support random access as well as sequential access.

* **Structure of B+ Tree:-**

Every leaf node is al equal distance from the root node. It is of the order n whene n is fixed for every B+ tree. It contains an internal node and lesf node.



* **Interal Node:-** It is of the B+ tree can contain at least n/2 record pointers excepted the root node.ate most, an internal node of the tree conyains n pointers.
* **Leaf Node:-** It is of the B+ tree can contain at teast n/2 record point and n/2 key value. Evert leafe node of the B+ tree contain one block point P to point to next leaf node.
* **Searching a record in B+ Tree:-** In B+ tree, we will find the recode we will first compare to the root node when leaf node is greater then then the root node the go riht side and when leaf node is less then the root not then go to the left side, the process is go on. At the end we will find the node when search number is same to the find node.

**Hashing**

* It is a technique to directly search the location of desired data on the disk without using index structure.
* **Data Bucket:-**  Memory location- recode are store(unit of storage).
* **Key:-** An attribute or set of attribute with helps you to identify a row in a relation.
* **Hash Key:-**  Mapping function that map all the set of search key to the address where actual recodes are placed.
* **Types of hashing:-**

1. **Static Hashing.**
2. **Dynamic Hashing.**
3. **Static Hashing:-**  When a search key value provided, the hash function always compute the same address.

**Mod(x) =0 🡪 x-1**

**Example:-**

If we want to generate address for a Key using mod(5) hash function, it always result in the same no. of data bucket in the memory for this static hashing function remain coustant.

* **Operations of Static Hashing:-**
* **Searching a record:-**

A record need to be searched, then the same hash function retrieves the address of the bucket where the data is stored.

* **Insert a record:-**

New record is inserted then we will generate an address for new record based on the hash key and stored in that location.

* **Delete a record:-**

First fetch the recode which is supposed to deleted then we will delete the records for that address in memory.

* **Update a record:-**

First search it using a hash function then data record is updated.

1. **Dynamic hashing:-**

It is used to overcome the problem of static hashing like bucket overflow. Data bucket grow or shrink as increases. This methocd is also known as Extendable hashing method.

# **SQL**

SQL stand for Structured Query Language. Used for storing and managing data in relation database management system (RDMS).

* **Datatype of SQL:-**
  + **Six type of datatype of SQL are:-**
    1. **Binary Datatype.**
    2. **Numeric Datatype.**
    3. **Extract Numeric Datatype.**
    4. **String Datatype.**
    5. **Date Datatype.**

1. **Binary Datatype:-** There are three type of binary datatype are:-
2. **Binary**
3. **Varbinary**
4. **Image**
5. **Numeric Datatype:-** There are two type of numeric datatype are:-
6. **Float**
7. **Real**
8. **Extract Numeric Datatype:-** There are five type of extract numeric datatype are:-
9. **Int**
10. **Smallint**
11. **Bit**
12. **Decimal**
13. **Number**
14. **String Datatype:-** There are three type of string datatype are:-
15. **Char**
16. **Varchar**
17. **Text**
18. **Date Datatype:-** There are three type of date datatype are:-
19. **Date**
20. **Time**
21. **Timestamp**

* **SQL Operator:-**

**There are various type of sql operation are:-**

1. **Arithmetic operation:-**  +,-,\*,/,%.
2. **Comparison Operation:-** =,!=,<>,>,<,>=,<=,!<,!>.
3. **Logical Operation:-** ALL,AND,ANY,BETWEEN,IN,

NOT,OR,EXISTS,LIKE.

* **SQL table:-** It is a collection of data in the form of row and columns.
* **Operation on table are:-**

1. **Create table.**
2. **Alter table.**
3. **Drop table.**
4. **Delete table.**
5. **Rename table.**
6. **Create table:-**  Create table is used to create a table in database. you should define the name of the table and also defined its columns and column’s data type.

**Syntax:-**

cerate table “table name”(“column1” “data type”,

“column2” “data type”,

“column3” “data type”,

“column4” “data type”,

……….

………..

“columnN” “data type”);

**Example:-**

CREATE TABLE EMPLOYEE(“ID” “INT” NOT NULL,

“NAME” “VAECHAR(25)” NOT NULL,

“PHONE” “INT” NOT NULL,

“ADDRESS” “CHAR (30)” PRIMARY KEY (ID));

1. **Alter Table:-** Its is used change the structure of the table.

**Syntex:-**

ALTER table TABLE NAME;

1. **Drop table:-** Drop table is used to delete a table defination and data from table.

**Syntax:-**

DROP TABLE “table name”;

**Example:-**

DROP TABLE EMPLOYEE;

1. **DELETED table:-** DELETE statement is used to delete rows from table.

**Syntax:-**

DELETE FROM table\_name WHERE condition;

**Example:-**

DELETE FROM EMPLOYEE WHERE EMP\_ID=3;

1. **Rename table:-** It is used to chnge the name of the name of the column of the table.

**Syntex:-**

ALTER TABLE “TABLE NAME” RENAME TO “NEW TABLE NAME”;

* **SELECT statement:-** SELECT is used to query or retrieve data from a table in the database.

**Syntax:-**

SELECT column1,coumn2,……

FROM table\_name;

**Example:-**

SELECT \* FROM table\_name;

* **INSERT statement:-** It is used to insert a single data in a table. In this statement insert the data in without specifying column name and by specifying column name.

**Syntax:-**

INSERT INTO TABLE\_NAME  
VALUES (value1, value2, value3, ….. value n);

**EXAMPLE:-**

INSERT INTO Customers

(Customer Name, Contact Name, Address, City, Postal code, Country)

VALUES (‘cardinal’, ‘Tom B.Erichsen’ , ‘Skagen 21’, ‘Stavanger’, ‘4006’, ‘Norway’);

* **UPDATE STATEMENT:-** It is used to modify the data that is already in the database. The condition in the WHERE clause decides that which row is to be update.

**SYNTAX:-**

UPDATE table-name

SET column1 = value1, column2 = value2, …

WHERE condition;

**EXAMPLE:-**

UPDATE Customer

SET Contact Name=’Juan’

WHERE Country= ‘Mexico’;

* **DELETE STATEMENT:-** It is used to delete rows from a table. Basically DELETE statement removes one or more records from a table.

**SYNTAX:-**

DELETE FROM table-name WHERE some-condition;

**EXAMPLE:-**

DELETE FROM Customers WHERE

Customer Name= ‘Alfreds Futterkiste’;

* **IN:-**  It is use to help in reduces to need to use multiple comdition.

**Syntax:-**

SELECT column\_name(s) FROM table\_name WHERE column\_name IN (value1, value2,…..);

* **SQL Clauses:-**

It is three type of SQL clause are:-

1. Group by clause.
2. Having clause.
3. Order by clause.
4. **Group by clause:-** This statement is used to arrange identical data into groups.

**Syntax:-**

SELECT “COLUMN NAME” FROM “TABLE NAME” WHERE “CONDITION” GROUP BY “COLUMN NAME”

1. **Having clause:-** It is used to specify a search condition for a group or an aggregate.

**Syntax:-**

SELECT column1, column2

FROM table\_name

WHERE conditions

GROUP BY column1, column2

HAVING conditions

ORDER BY column1, column2;

1. **Order by clause:-**  Short the result-set in ascending or descending order.

**Syntax:-**

SELECT column1, column2

FROM table\_name

WHERE condition

ORDER BY column1,column2…. ASC|DESC;

* **SQL Join:-**

SQL join is to combine two or more table. It is used to combine the records from two or more table in a database.

* **Type of SQL join:-**

1. **Inner Join.**
2. **Left Join.**
3. **Right Join.**
4. **Full Join.**
5. **Inner Join:-**  Selects recodes that have matching value in both table as long as the condition to satisfied.

**Syntax:-**

SELECT column name

FROM table\_name1

INNER JOIN table\_name2

ON table1. Column = table 2 . column;

1. **Left Join :-** Return all the row from the left table even if there are no matches in might table.

**Syntax:-**

SELECT table1.column1, table1.column2, table2.column1,....

FROM table1

LEFT JOIN table2

ON table1.matching\_column = table2.matching\_column;

1. **RIGHT Join:-** The functionality is same as lift join. It just put the data to right side.

**Syntax:-**

SELECT table1.column1, table1.column2, table2.column1,....

FROM table1

RIGHT JOIN table2

ON table1.matching\_column = table2.matching\_column;