Royal Education Society's

COCSIT LATUR

FACULTY OF COMPUTER STUDIES

BCA (Second Year) (Third Semester)

Revised CBCS

Name of the Paper-Database Management System (BCA-303)

Prepared by: Kawade P.P.

Q.1 Attempt any FIVE of the following (3 Marks each) 15

a) Explain Characteristics of DBMS?
b) Explain Users of DBMS?
c) What is Index with their types?
d) Explain Entity, entity set and Relationship?
e) Explain ER to Relational conversion (any 2 rule)?
f) Explain Project Operation?
g) Explain 2NF Normal Form?

Q.2 Attempt any Three of the following (5 Marks each) 15

a) Explain keys with their types?
b) Explain Structure of DBMS?
c) Explain Important terms of relational databases?
d) Explain in detail types of anomaly?
e) Explain types of Relationship?

Q.3 Attempt any Three of the following (5 Marks each) 15

a) What is Attribute with their types?
b) Explain Extended E-R features?
c) Explain Union and Select Algebra Operation?
d) Explain Normalization with 1NF Conversion?
e) Explain Components of E-R diagram?

Q.4 Attempt any Three of the following (5 Marks each) 15

a) Explain Constraints with their types?
b) What is Functional Dependency?
c) Explain 3NF Normal Form?
d) Explain Cartesian product and Intersection Operation?
e) Difference between File processing systems Vs DBMS.

.

Q.5 Write short notes on any three of the following (5 Marks each)15

i) Natural Join operation
ii) BCNF Normal Form
iii) Types of Data Models

iv) Types of File Organization

v) Advantages and Disadvantages of DBMS

Q.1 Attempt any FIVE of the following (3 Marks each) 15

a) Explain Characteristics of DBMS?

Ans. Characteristics of DBMS

A database management system is able to store any kind of data in a database.

The database management system has to support ACID (atomicity, consistency, isolation, durability) properties.

The Database management system allows so many users to access databases at the same time.

Backup and recovery are the two main methods which allow users to protect the data from damage or loss.

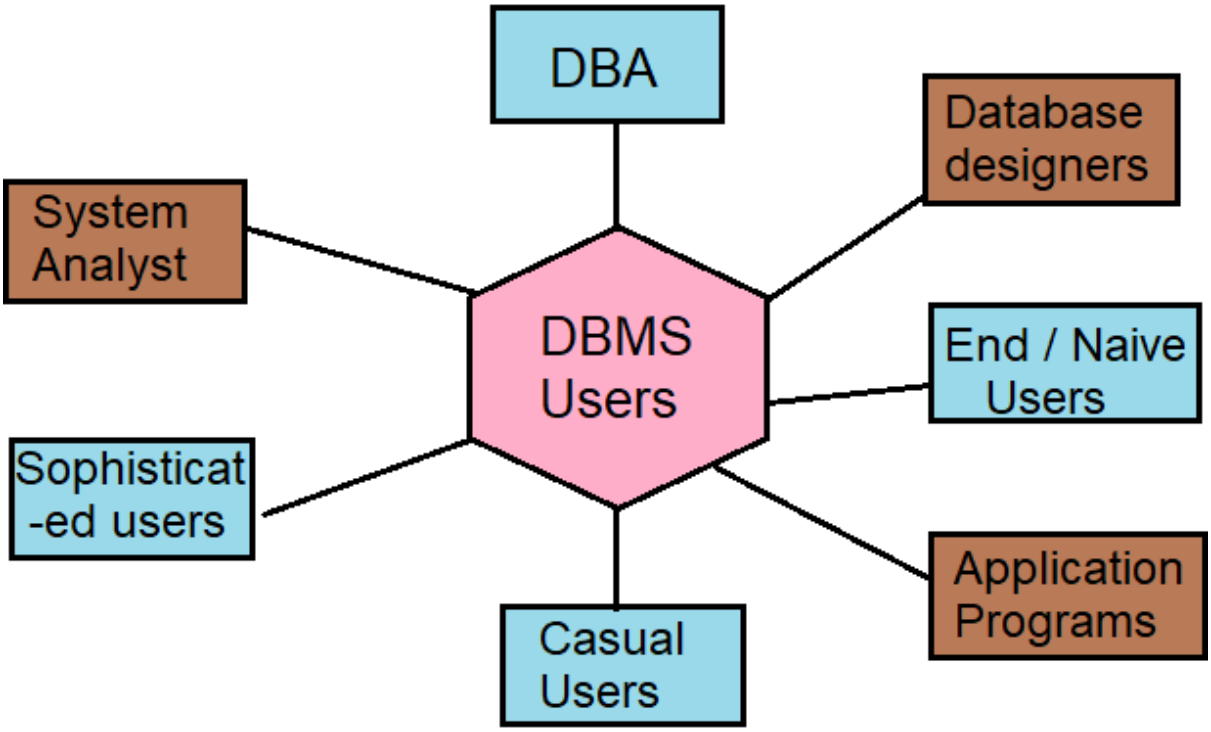
It also provides multiple views for different users in a single organization.

It follows the concept of normalization which is helpful to minimize the redundancy of a relation.

It also provides users query language, helpful to insert, retrieve, update, and delete the data in a database.

b) Explain Users of DBMS?

Ans. Users in DBMS



These are seven types of data base users in DBMS.

1. Database Administrator (DBA) :

Database Administrator (DBA) is a person/team who defines the schema and also controls the 3 levels of database.

The DBA will then create a new account id and password for the user if he/she need to access the data base. DBA is also responsible for providing security to the data base and he allows only the authorized users to access/modify the data base.

- DBA also monitors the recovery and back up and provide technical support.
- The DBA has a DBA account in the DBMS which called a system or superuser account. DBA repairs damage caused due to hardware and/or software failures.

2. Naive / Parametric End Users :

Parametric End Users are the unsophisticated who don't have any DBMS knowledge but they frequently use the data base applications in their daily life to get the desired results.

For examples, Railway's ticket booking users are naive users.

Clerks in any bank is a naive user because they don't have any DBMS knowledge but they still use the database and perform their given task.

3. System Analyst :

System Analyst is a user who analyzes the requirements of parametric end users. They check whether all the requirements of end users are satisfied.

4. Sophisticated Users :

Sophisticated users can be engineers, scientists, business analyst, who are familiar with the database. They can develop their own data base applications according to their requirement. They don't write the program code but they interact the data base by writing SQL queries directly through the query processor.

5. Data Base Designers :

Data Base Designers are the users who design the structure of data base which includes tables, indexes, views, constraints, triggers, stored procedures. He/she controls what data must be stored and how the data items to be related.

6. Application Program :

Application Program are the back end programmers who writes the code for the application programs. They are the computer professionals. These programs could be written in Programming languages such as Visual Basic, Developer, C, FORTRAN, COBOL etc.

7. Casual Users / Temporary Users :

Casual Users are the users who occasionally use/access the data base but each time when they access the data base they require the new information, for example, Middle or higher level manager.

c) What is Index with their types?

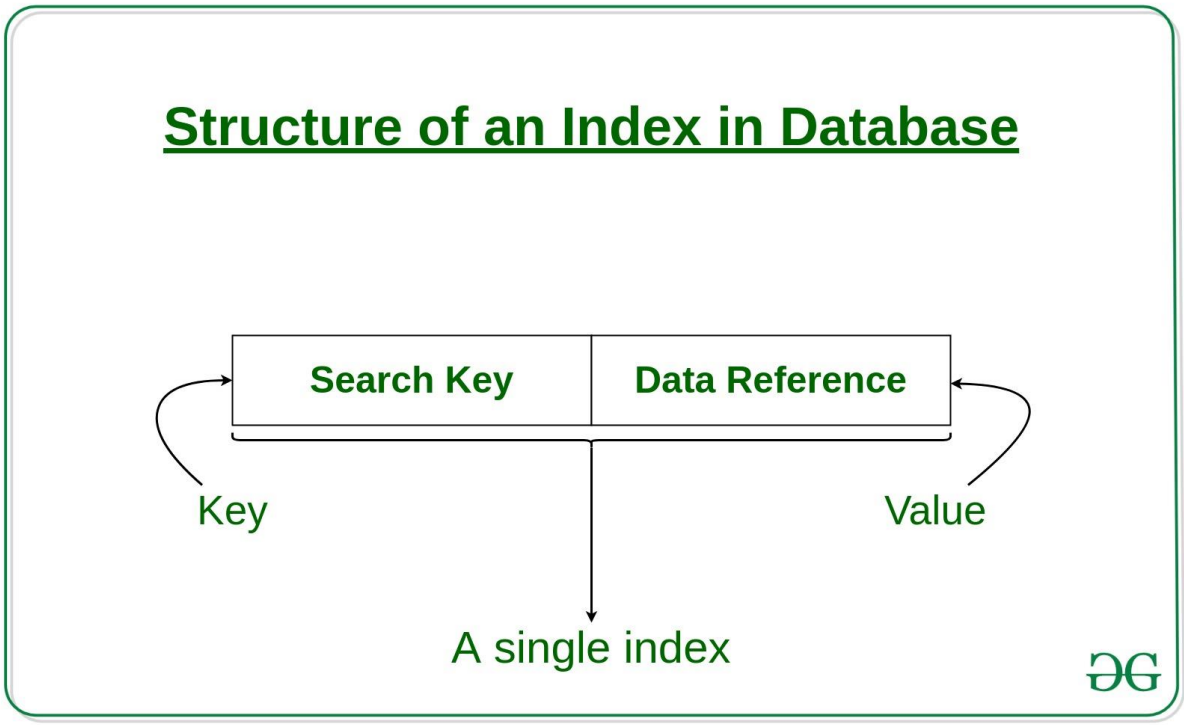
Ans. Indexing is a way to optimize the performance of a database by minimizing the number of disk accesses required when a query is processed. It is a data structure technique which is used to quickly locate and access the data in a database.

Indexes are created using a few database columns.

- The first column is the Search key that contains a copy of the primary key or candidate key of the table. These values are stored in sorted order so that the corresponding data can be accessed quickly.

  Note: The data may or may not be stored in sorted order.

- The second column is the Data Reference or Pointer which contains a set of pointers holding the address of the disk block where that particular key value can be found.

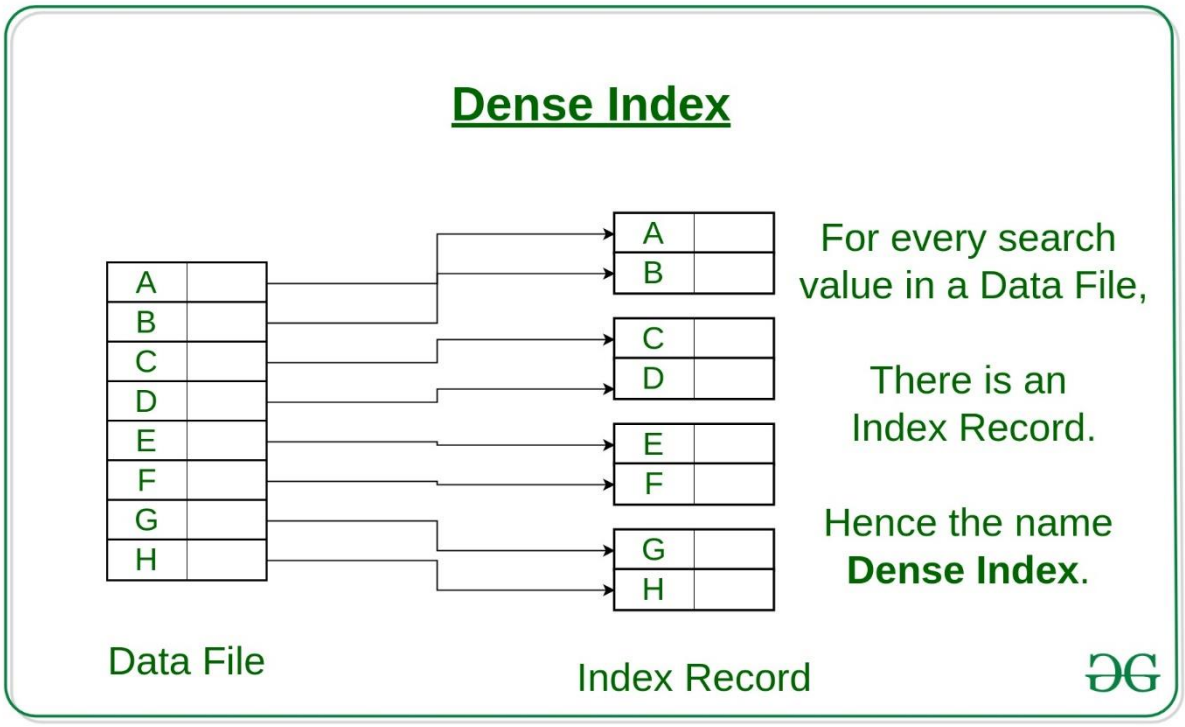


In general, there are two types of file organization mechanism which are followed by the indexing methods to store the data:

1. Sequential File Organization or Ordered Index File: In this, the indices are based on a sorted ordering of the values. These are generally fast and a more traditional type of storing mechanism. These Ordered or Sequential file organization might store the data in a dense or sparse format:

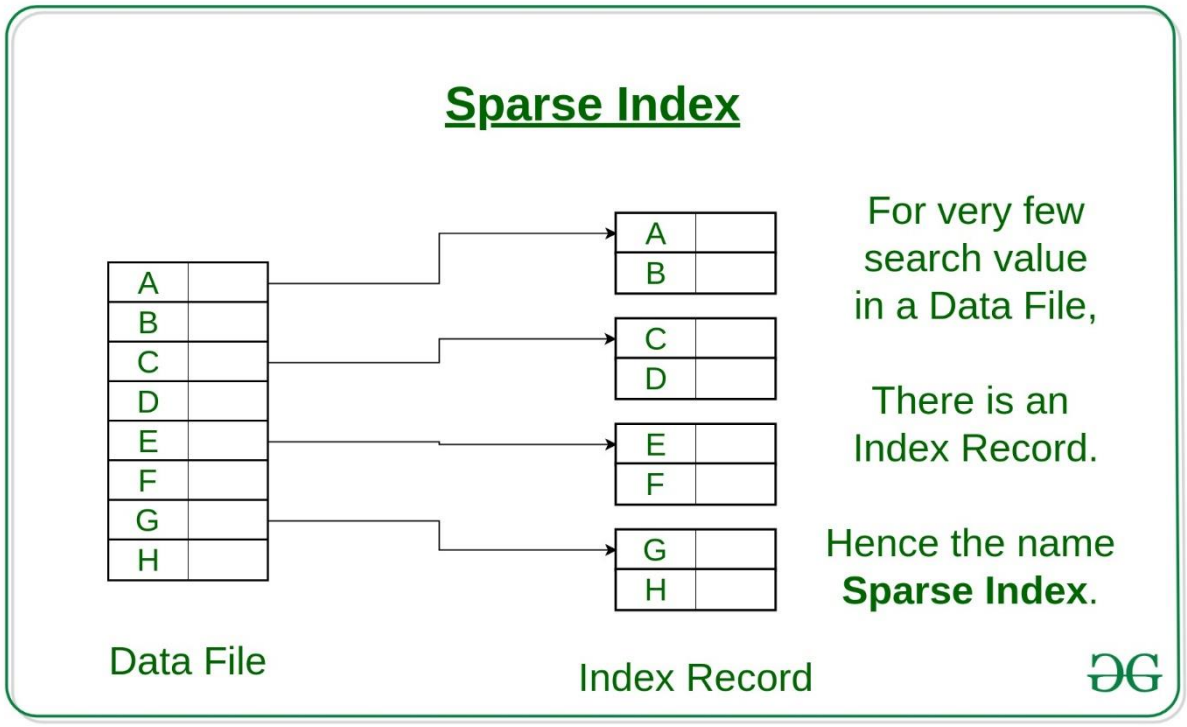
(i) Dense Index:

- For every search key value in the data file, there is an index record.
- This record contains the search key and also a reference to the first data record with that search key value.



(ii) Sparse Index:

- The index record appears only for a few items in the data file. Each item points to a block as shown.
- To locate a record, we find the index record with the largest search key value less than or equal to the search key value we are looking for.
- We start at that record pointed to by the index record, and proceed along with the pointers in the file (that is, sequentially) until we find the desired record.



2. Hash File organization:

Indices are based on the values being distributed uniformly across a range of buckets. The buckets to which a value is assigned is determined by a function called a hash function.

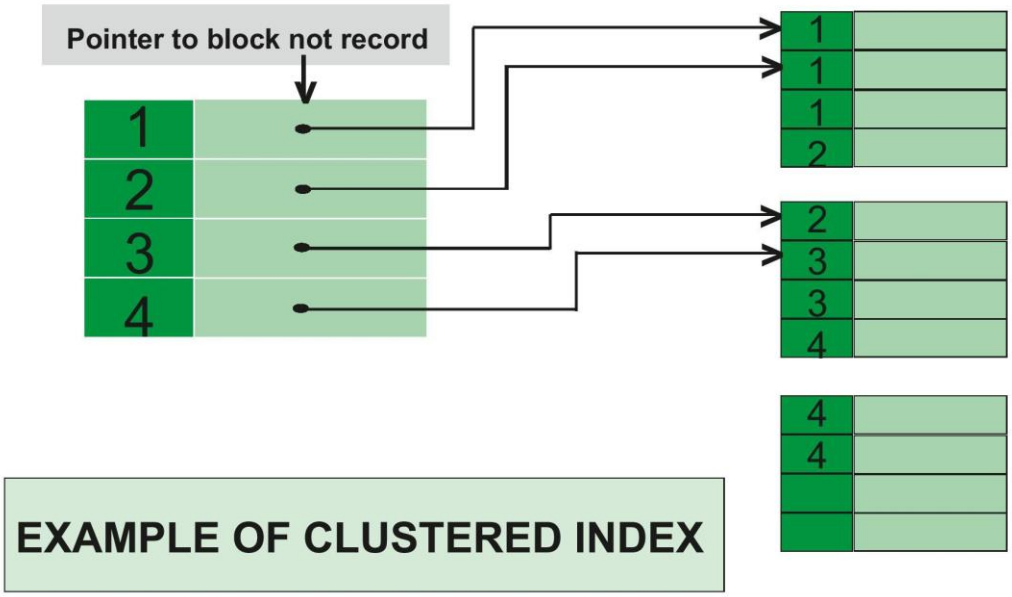
There are primarily three methods of indexing:

(i)Clustered Indexing

When more than two records are stored in the same file these types of storing known as cluster indexing. By using the cluster indexing we can reduce the cost of searching reason being multiple records related to the same thing are stored at one place and it also gives the frequent joining of more than two tables (records).

Clustering index is defined on an ordered data file. The data file is ordered on a non-key field. In some cases, the index is created on non-primary key columns which may not be unique for each record. In such cases, in order to identify the records faster, we will group two or more columns together to get the unique values and create index out of them. This method is known as the clustering index. Basically, records with similar characteristics are grouped together and indexes are created for these groups.

For example, students studying in each semester are grouped together. i.e. 1st Semester students, 2nd semester students,3rd semester students etc. are grouped.

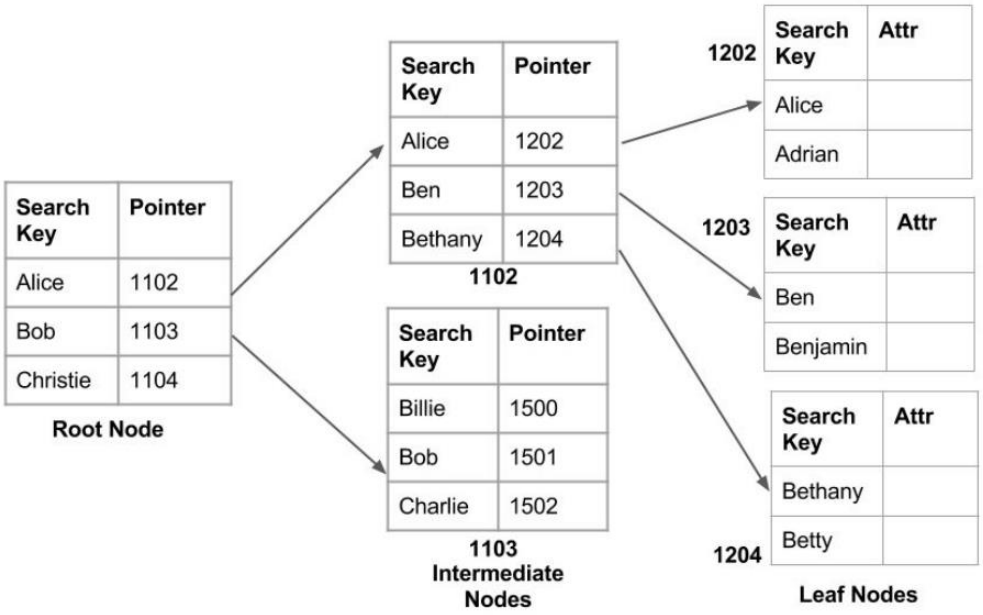


Clustered index sorted according to first name (Search key) Primary Indexing:

This is a type of Clustered Indexing wherein the data is sorted according to the search key and the primary key of the database table is used to create the index. It is a default format of indexing where it induces sequential file organization. As primary keys are unique and are stored in a sorted manner, the performance of the searching operation is quite efficient.

2. Non-clustered or Secondary Indexing

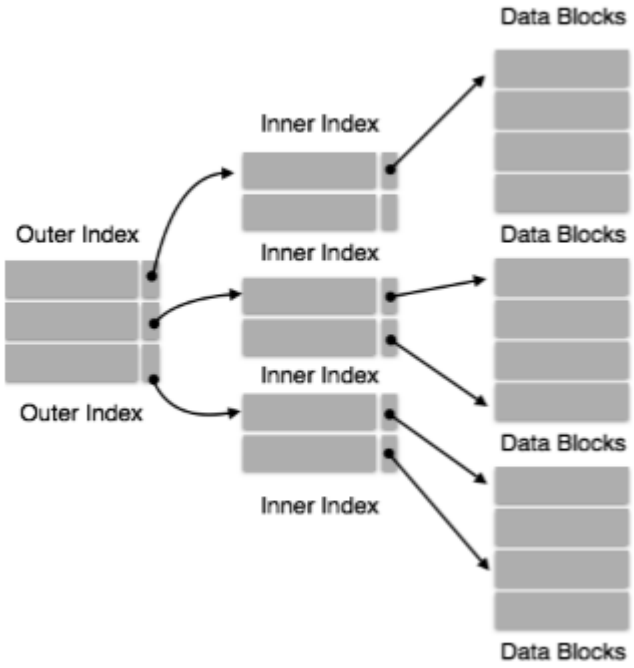
A non-clustered index just tells us where the data lies, i.e. it gives us a list of virtual pointers or references to the location where the data is actually stored. Data is not physically stored in the order of the index. Instead, data is present in leaf nodes. For ex. The contents page of a book. Each entry gives us the page number or location of the information stored. The actual data here (information on each page of the book) is not organized but we have an ordered reference (contents page) to where the data points actually lie. We can have only dense ordering in the non-clustered index as sparse ordering is not possible because data is not physically organized accordingly. It requires more time as compared to the clustered index because some amount of extra work is done in order to extract the data by further following the pointer. In the case of a clustered index, data is directly present in front of the index.



Non clustered index

3. Multilevel Indexing

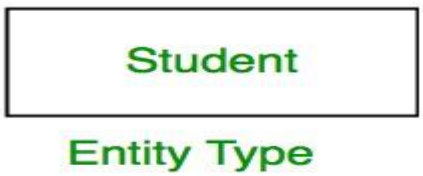
With the growth of the size of the database, indices also grow. As the index is stored in the main memory, a single-level index might become too large a size to store with multiple disk accesses. The multilevel indexing segregates the main block into various smaller blocks so that the same can stored in a single block. The outer blocks are divided into inner blocks which in turn are pointed to the data blocks. This can be easily stored in the main memory with fewer overheads.



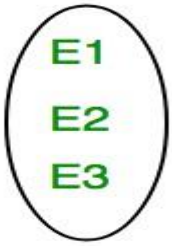
d) Explain Entity, entity set and Relationship?

Ans. Entity may be an object with a physical existence – a particular person, car, house, or employee – or it may be an object with a conceptual existence – a company, a job, or a university course.

An Entity is an object of Entity Type and set of all entities is called as entity set. e.g.; E1 is an entity having Entity Type Student and set of all students is called Entity Set. In ER diagram, Entity Type is represented as:



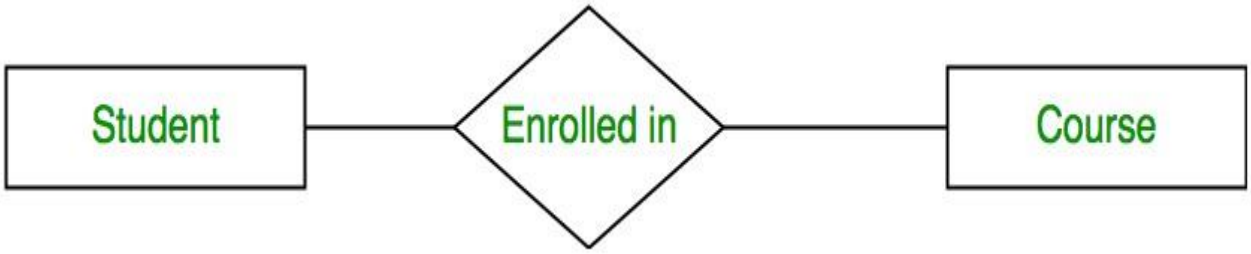
Entity Type



Entity Set

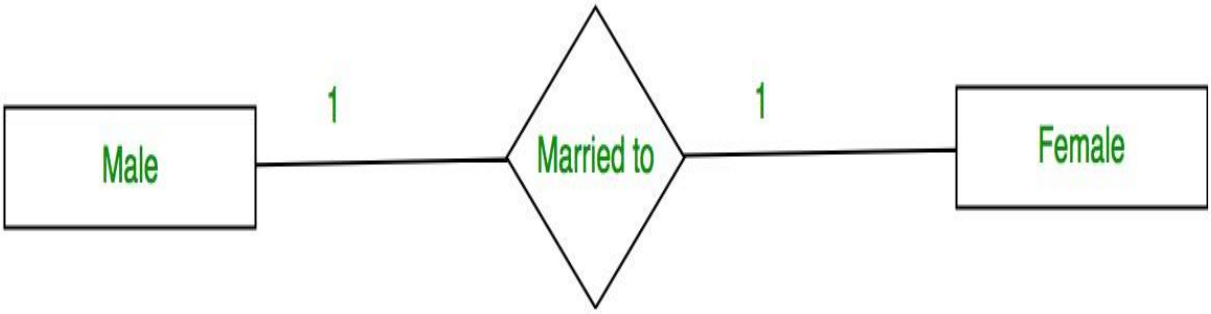
Relationship

A relationship type represents the **association between entity types**. For example,'Enrolled in' is a relationship type that exists between entity type Student and Course. In ER diagram, relationship type is represented by a diamond and connecting the entities with lines.

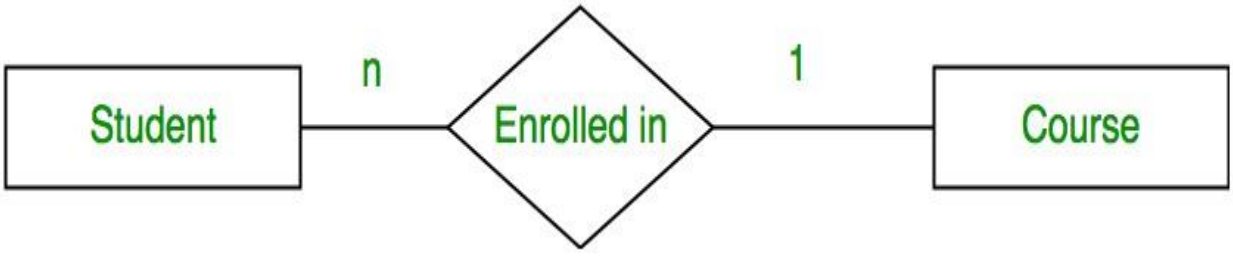


Relationship types:-

1. One to one – Such a relationship exists when one record of one table is related to only one record of the other table.



1. **Many to one –** Such a relationship exists when number of records of one table is related to one record of the other table.



One to many- Such a relationship exists when one record of one table is related to number of records of the other table. Ex. Students gives no. of exams.

3. Many to many – Such a relationship exists when number of records of one table is related to number of records of the other table.

e) Explain Project Operation?

Ans.Projection operation

The projection eliminates all attributes of the input relation but those mentioned in the projection list. The projection method defines a relation that contains a vertical subset of Relation.

It displays the specific column of a table. It is denoted by pie (∏). It is a vertical subset of the original relation. It eliminates duplicate tuples.

Syntax

The syntax is as follows −

$\prod_{regno}(student)$

Example

Consider the student table:

Regno	Branch	Section
1	CSE	A
2	ECE	B
3	CIVIL	B
4	IT	A

To display regno column of student table, we can use the following command –

$\prod_{regno}(student)$

Output

RegNo
1
2
3
4

To display branch, section column of student table, use the following command −

$\prod branch,section(student)$

The result is as follows −

Branch	Section
CSE	A
ECE	B
CIVIL	B
IT	A

f) Explain 2NF Normal Form?

Ans. Second Normal Form

- In the 2NF, relational must be in 1NF.
- In the second normal form, all non-key attributes are fully functional dependent on the primary key

Example: Let's assume, a school can store the data of teachers and the subjects they teach. In a school, a teacher can teach more than one subject.

TEACHER table

TEACHER_ID	SUBJECT	TEACHER_AGE
25	Chemistry	30
25	Biology	30
47	English	35
83	Math	38
83	Computer	38

In the given table, non-prime attribute TEACHER_AGE is dependent on TEACHER_ID which is a proper subset of a candidate key. That's why it violates the rule for 2NF.

To convert the given table into 2NF, we decompose it into two tables:

TEACHER_DETAIL table:

TEACHER_ID	TEACHER_AGE
25	30
47	35
83	38

TEACHER_SUBJECT table:

TEACHER_ID	SUBJECT
25	Chemistry
25	Biology

47	English
83	Math
83	Computer

Q.2 Attempt any Three of the following (5 Marks each)

a) Explain keys with their types?

Ans. Keys in DBMS is an attribute or set of attributes which helps you to identify a row(tuple) in a relation(table). 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 the table. Database key is also helpful for finding unique record or row from the table.

Example:

Employee ID	FirstName	LastName
11	Andrew	Johnson
22	Tom	Wood
33	Alex	Hale

In the above-given example, employee ID is a primary key because it uniquely identifies an employee record. In this table, no other employee can have the same employee ID.

Types of Keys

There are four types of Keys in DBMS

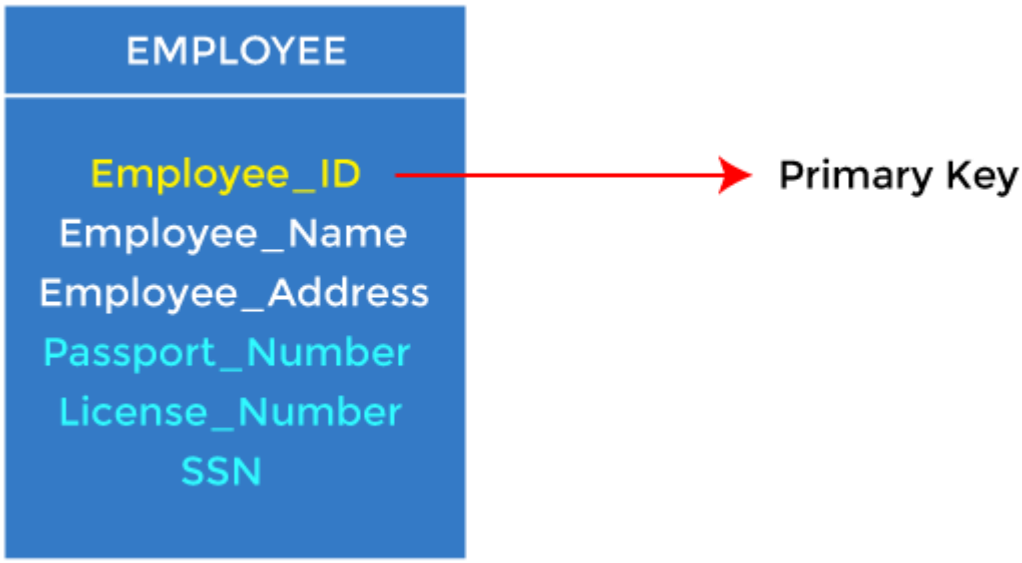
1. Primary Key
2. Candidate Key
3. Super Key
4. Foreign Key

1. Primary key

o It is the first key used to identify one and only one instance of an entity uniquely. An entity can contain multiple keys, as we saw in the PERSON table. The key which is most suitable from those lists becomes a primary key.

o In the EMPLOYEE table, ID can be the primary key since it is unique for each employee. In the EMPLOYEE table, we can even select License_Number and Passport_Number as primary keys since they are also unique.

o For each entity, the primary key selection is based on requirements and developers.

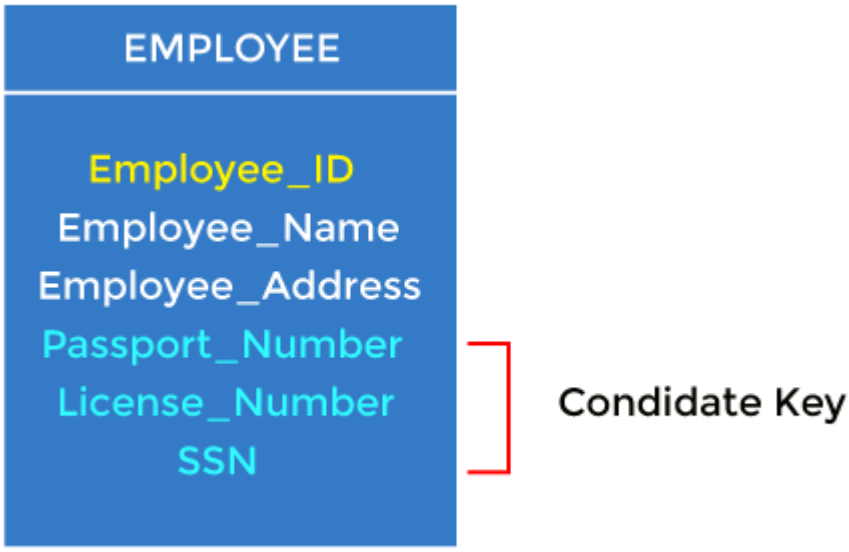


2. Candidate key

o A candidate key is an attribute or set of attributes that can uniquely identify a tuple.

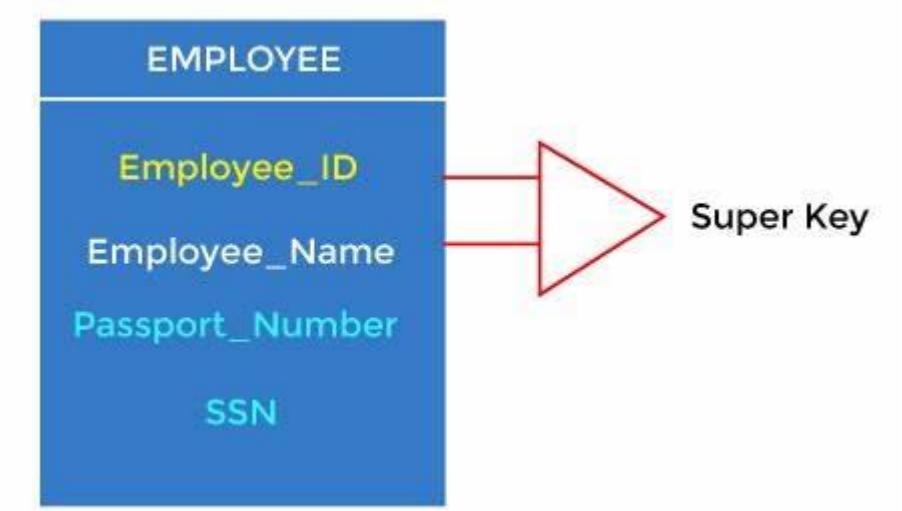
o Except for the primary key, the remaining attributes are considered a candidate key. The candidate keys are as strong as the primary key.

For example: In the EMPLOYEE table, id is best suited for the primary key. The rest of the attributes, like SSN, Passport_Number, License_Number, etc., are considered a candidate key.



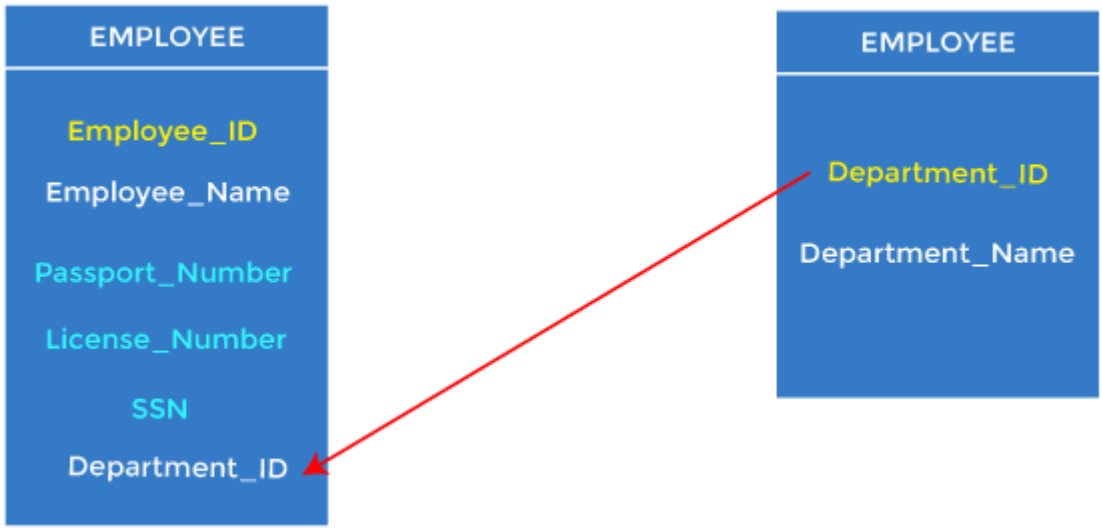
3. Super Key

Super key is an attribute set that can uniquely identify a tuple. A 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
o Foreign keys are the column of the table used to point to the primary key of another table.
o Every employee works in a specific department in a company, and employee and department are two different entities. So we can't store the department's information in the employee table. That's why we link these two tables through the primary key of one table.
o We add the primary key of the DEPARTMENT table, Department_Id, as a new attribute in the EMPLOYEE table.
o In the EMPLOYEE table, Department_Id is the foreign key, and both the tables are related.



b) Explain Structure of DBMS?

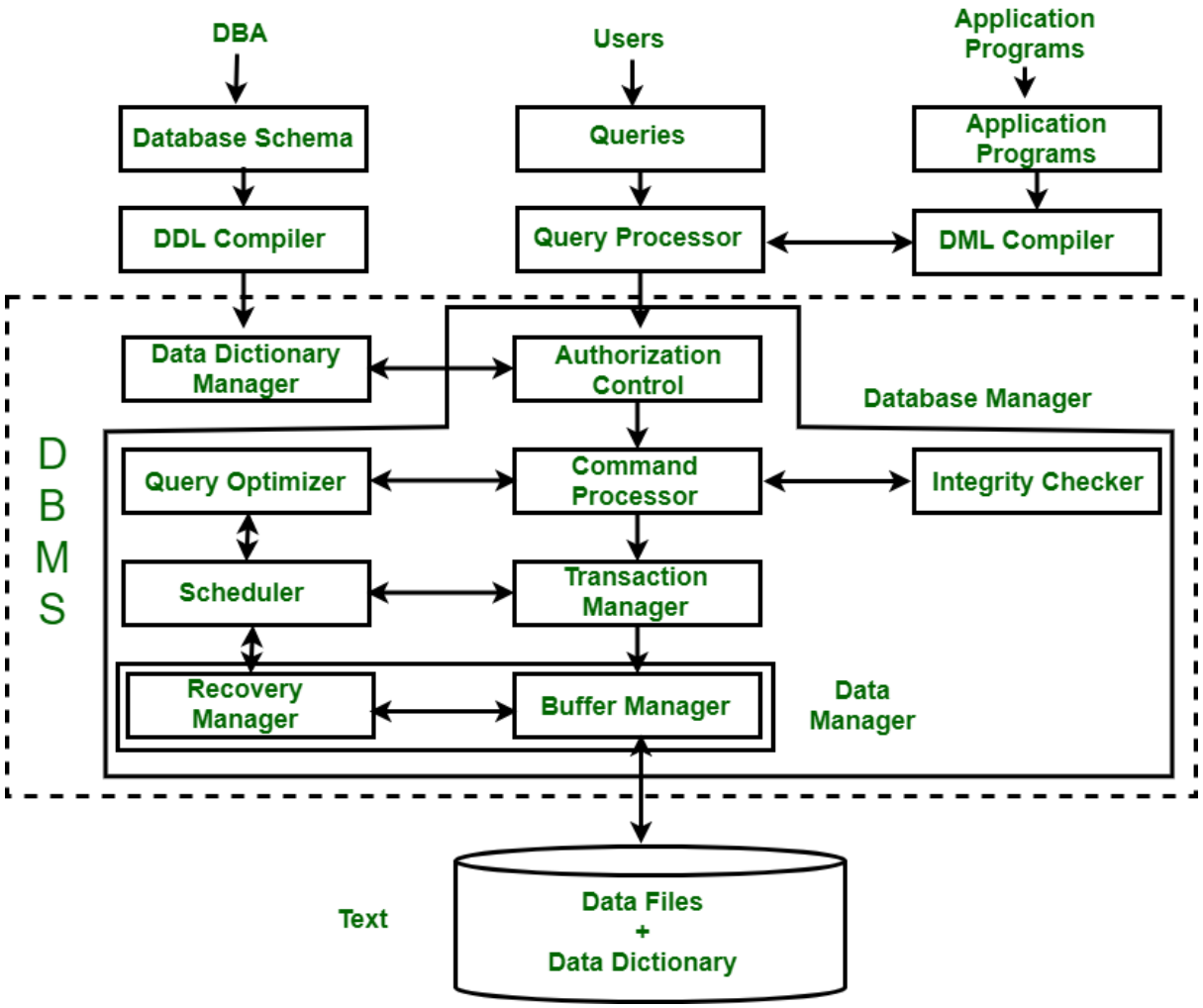
Ans. Structure of DBMS

Database Management System (DBMS) is a software that allows access to data stored in a database and provides an easy and effective method of –

- Defining the information.
- Storing the information.
- Manipulating the information.
- Protecting the information from system crashes or data theft.
- Differentiating access permissions for different users.

Please be note that the Structure of Database Management System is also referred to as Overall System Structure or Database Architecture but it is different from the tier architecture of Database.

The database system is divided into three components: Query Processor, Storage Manager, and Disk Storage. These are explained as following below.



Architecture of DBMS

1. Query Processor:
It interprets the requests (queries) received from end user via an application program into instructions. It also executes the user request which is received from the DML compiler.
Query Processor contains the following components –

- DML Compiler –

It processes the DML statements into low level instruction (machine language), so that they can be executed.

- DDL Interpreter –

It processes the DDL statements into a set of table containing meta data (data about data).

- Embedded DML Pre-compiler –

It processes DML statements embedded in an application program into procedural calls.

- Query Optimizer –

It executes the instruction generated by DML Compiler.
2. Storage Manager:

Storage Manager is a program that provides an interface between the data stored in the database and the queries received. It is also known as Database Control System. It maintains the consistency and integrity of the database by applying the constraints and executes the DCL statements. It is responsible for updating, storing, deleting, and retrieving data in the database.

It contains the following components –

- Authorization Manager –

It ensures role-based access control, i.e,. checks whether the particular person is privileged to perform the requested operation or not.

- Integrity Manager –

It checks the integrity constraints when the database is modified.

- Transaction Manager –

It controls concurrent access by performing the operations in a scheduled way that it receives the transaction. Thus, it ensures that the database remains in the consistent state before and after the execution of a transaction.

- File Manager –

It manages the file space and the data structure used to represent information in the database.

- Buffer Manager –It is responsible for cache memory and the transfer of data between the secondary storage and main memory.

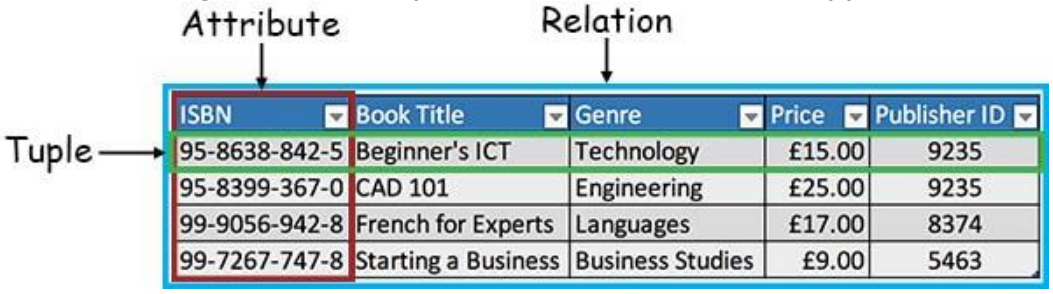
3. Disk Storage: It contains the following components –

- Data Files – It stores the data.
- Data Dictionary –It contains the information about the structure of any database object. It is the repository of information that governs the metadata.
- Indices –It provides faster retrieval of data item.

c) Explain Important terms of relational databases?

Ans.

- Relations – Most of us know what a table is from using spreadsheet software. It is a grid made up of rows and columns. Well in relational databases we often refer to these as “relations”.
- Tuples – A row in a table is known as a record, or a tuple. A tuple holds all the data on a single item. For example, if we have a table to store book details for a shop, then each tuple is an individual book the shop sells, and it will store all the data on that individual book such as the ISBN, book title and price.
- Attributes – An individual piece of data in a record is known as a field, or attribute. In the previous example we said that each record for a book will store the ISBN, book title and price. Well these three things are all examples of attributes. Attributes appear as columns in a database table.



ISBN	Book Title	Genre	Price	Publisher ID
95-8638-842-5	Beginner's ICT	Technology	£15.00	9235
95-8399-367-0	CAD 101	Engineering	£25.00	9235
99-9056-942-8	French for Experts	Languages	£17.00	8374
99-7267-747-8	Starting a Business	Business Studies	£9.00	5463

- Domains – All the possible allowable values for an attribute. This is slightly different to the data type of the attribute. For example, a field may have an integer number data type, which defines that it can only allow whole numbers to be entered. However, there may be additional rules applied,

such as that the number must be between 1 & 10. The domain would therefore be this range of whole numbers.

- Cardinality – How unique an attribute is in terms of its data values. Some attributes will have a wide range of different data values entered. For example, the primary key field will have a completely unique value for every record. Where there is a large percentage of unique values, this is known as “High Cardinality”. Where there are a lot of repeated values across the entities tuples, this is known as having “Low Cardinality”.

d) Explain in detail types of anomaly?

Ans.Normalization is the process of splitting relations into well-structured relations that allow users to insert, delete, and update tuples without introducing database inconsistencies. Without normalization, many problems can occur when trying to load an integrated conceptual model into the DBMS. These problems arise from relations that are generated directly from user views are called anomalies. There are three types of anomalies: update, deletion, and insertion anomalies.

1)Update Anomaly-An update anomaly is a data inconsistency that results from data redundancy and a partial update. For example, each employee in a company has a department associated with them as well as the student group they participate in.

Employee_ID	Name	Department	Student_Group
123	J. Longfellow	Accounting	Beta Alpha Psi
234	B. Rech	Marketing	Marketing Club
234	B. Rech	Marketing	Management Club
456	A. Bruchs	CIS	Technology Org.
456	A. Bruchs	CIS	Beta Alpha Psi

If A. Bruchs' department is an error it must be updated at least 2 times or there will be inconsistent data in the database. If the user performing the update does not realize the data is stored redundantly the update will not be done properly.

2)Deletion Anomaly-A deletion anomaly is the unintended loss of data due to deletion of other data. For example, if the student group Beta Alpha Psi disbanded and was deleted from the table above, J. Longfellow and the Accounting department would cease to exist. This results in database inconsistencies and is an example of how combining information that does not really belong together into one table can cause problems.

3)Insertion Anomaly-An insertion anomaly is the inability to add data to the database due to the absence of other data. For example, assume Student Group is defined so that null values are not allowed. If a new employee is hired but not immediately assigned to a Student Group, then this employee could not be entered into the database. This results in database inconsistencies due to omission.

Update, deletion, and insertion anomalies are very undesirable in any database. Anomalies are avoided by the process of normalization.

Q.3 Attempt any Three of the following (5 Marks each)

a) What is Attribute with their types?

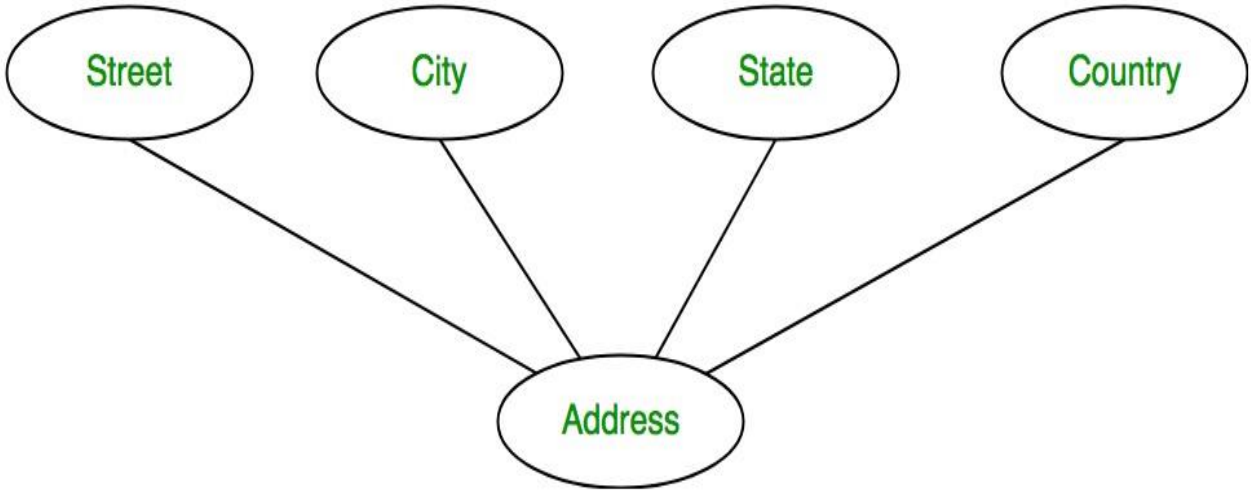
Attributes are the **properties which define the entity type**. For example, Roll No, Name, DOB, Age, Address, Mobile No are the attributes which defines entity type Student. In ER diagram, attribute is represented by an oval.

1. Key Attribute –

The attribute which **uniquely identifies each entity** in the entity set is called key attribute. For example, Roll No will be unique for each student. In ER diagram, key attribute is represented by an oval with underlying lines.

2. Composite Attribute –

An attribute **composed of many other attribute** is called as composite attribute. For example, Address attribute of student Entity type consists of Street, City, State, and Country. In ER diagram, composite attribute is represented by an oval comprising of ovals.



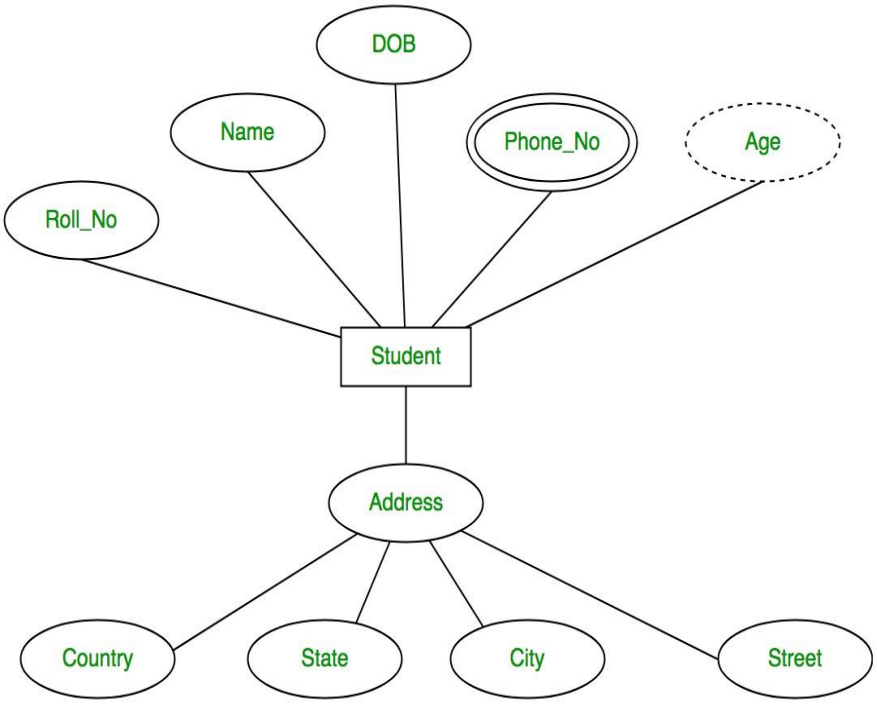
3. Multivalued Attribute –

An attribute consisting **more than one value** for a given entity. For example, Phone_No (can be more than one for a given student). In ER diagram, multivalued attribute is represented by double oval.

4. Derived Attribute –

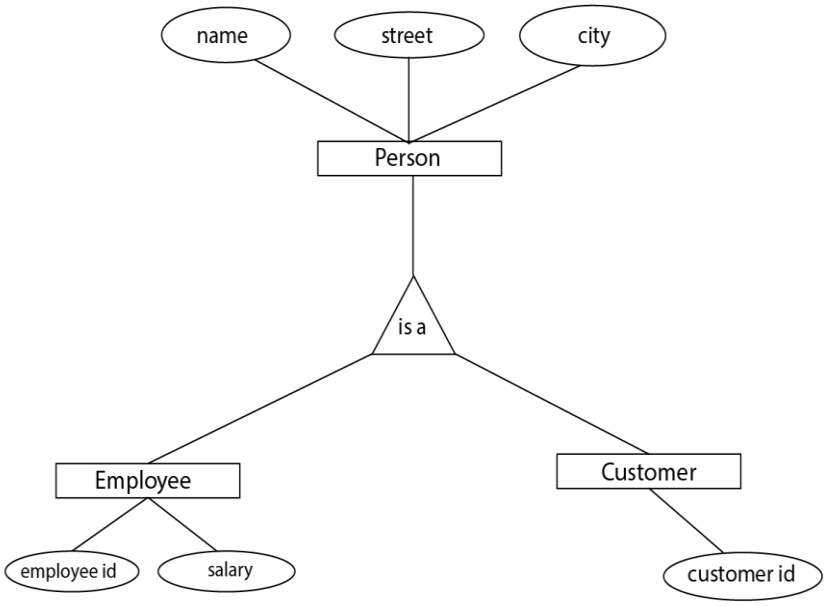
An attribute which can be **derived from other attributes** of the entity type is known as derived attribute. e.g.; Age (can be derived from DOB). In ER diagram, derived attribute is represented by dashed oval.

The complete entity type **Student** with its attributes can be represented as:



b) Explain Extended E-R features?

Take an example - a person (name, street, city) be an entity set which is further classified among “customer” and “employee”.



Each of these person type is describe by a set of attributes that include all the attribute of entity set “person” plus possible additional attribute like “customer id” of “customer” entity. “employee id”, “salary” of “employee” entity.

1)Specialization – The process of designating to sub grouping within an entity set is called specialization. In above figure, the “person” is distinguish in to whether they are “employee” or “customer”.

Formally in above figure specialization is depicted by a triangle component labelled (is a), means the *customer* is a *person*.

Sometime this ISA (is a) referred as a superclass-subclass relationship. This is also used to emphasize on to creating the distinct lower level entity sets.

2)Generalization – generalization is relationship that exist between higher level entity set and one or more lower level entity sets. Generalization synthesizes these entity sets into single entity set.

3)Higher level and lower level entity sets – This property is created by specialization and generalization. The attributes of higher level entity sets are inherited by lower level entity sets.
For example: In above figure “customers” and “employee” inherits the attributes of “person”.

4)Attribute inheritance- When given entity set is involved as a lower entity set in only one “ISA” (is a) relationship, it is referred as a *single attribute inheritance*. If lower entity set is involved in more than one ISA (is a) relationship, it is referred as a *multi attribute inheritance.*

5)Aggregation- there is a one limitation with E-R model that it cannot express relationships among relationships. So aggregation is an abstraction through which relationship is treated as *higher level entities*.

c) Explain Union and Select Algebra Operation?

Ans.

1)Union: A union of two relations (R1 U R2) can only be performed if the two relations are *union compatible*. This essentially means that both relations have the exact same attributes. A union combines the rows of the two relations and outputs a new relation that has both input relations’ rows in it. Suppose, taking the example table above, we want to union with this new table below:

PERSON

Name	Age	FavoriteFood
John	15	Pizza
Stacy	32	Curry
Glenn	27	Swordfish

Example PERSON relation with attributes Name, Age, and FavoriteFood

$PERSON_2$

Name	Age	FavoriteFood
Amy	5	Cake

The end result would look like this:

PERSON ∪ PERSON$_2$

Name	Age	FavoriteFood
John	15	Pizza
Stacy	32	Curry
Glenn	27	Swordfish
Amy	5	Cake

2) Select operation

It displays the records that satisfy a condition. It is denoted by sigma (σ) and is a horizontal subset of the original relation.

Syntax

Its syntax is as follows −

$\sigma_{condition}(\text{table name})$

Example

Consider the student table given below −

Regno	Branch	Section
1	CSE	A
2	ECE	B
3	CIVIL	B
4	IT	A

Now, to display all the records of student table, we will use the following command −

$\sigma(\text{student})$

In addition to this, when we have to display all the records of CSE branch in student table, we will use the following command −

$\sigma_{branch=cse}(student)$

Hence, the result will be as follows −

RegNo	Branch	Section
1	CSE	A

To display all the records in student tables whose regno>2, we will use the below mentioned command −

$\sigma RegNo>2(student)$

The output will be as follows −

RegNo	Branch	Section
3	CIVIL	B
4	IT	A

To display the record of ECE branch section B students, use the given command −

$\sigma_{branch=ECE \wedge section=B}(student)$

To display the records of section B CSE and IT branch, use the following command −

$\sigma_{Section=B \wedge Branch=cse \vee branch=IT}(student)$

d) Explain Normalization with 1NF Conversion?

Ans.

- Normalization is the process of organizing the data in the database.
- Normalization is used to minimize the redundancy from a relation or set of relations. It is also used to eliminate undesirable characteristics like Insertion, Update, and Deletion Anomalies.
- Normalization divides the larger table into smaller and links them using relationships.
- The normal form is used to reduce redundancy from the database table.

A relation is in 1NF if it contains atomic values. It states that an attribute of a table cannot hold multiple values. It must hold only single-values attributes. First normal form disallows the multi-valued attributes, composite attributes, and their combinations.

Requirements

The requirements to be considered while designing 1 NF are explained below −

- Each table has a primary key (minimal set of attributes which can uniquely identify a record).

- The values in each column of a table are atomic (no multivalue attributes are allowed).
- There are no repeating groups (two columns do not store similar information in the same table).

Example 1

Consider a relation student (rollno, name, branch, address, phone).

Rollno	Name	Branch	Address	Phone
1	AAA	CSE	Hyderabad	3242344,4564555,3112453
2	BBB	ECE	Delhi	3452245,4323245

The above relation is not in 1NF because the phone is a multivalued attribute, which is having multiple numbers for a single person.

Now we represent the above table by creating a new row for each phone number as shown below −

Rollno	Name	Branch	Address	Phone
1	AAA	CSE	Hyderabad	3242344
1	AAA	CSE	Hyderabad	4564555
1	AAA	CSE	Hyderabad	3112453
2	BBB	ECE	Delhi	3452245
2	BBB	ECE	Delhi	4323245

The above table contains redundant data due to phone numbers, as for each phone number, we have to repeat all the information of the student. So the phone attribute should be separated from the above table.

We divide or decompose the above table R into two tables which is the concept of normalization −

R1(key, multivalued attribute), R2(R-multivalued attribute)

=>R1(rollno, phone), R2(rollno, name, branch, address).

Steps to decompose the 1NF table are as follows −

- Place all items that appear in the repeating group in a new table.
- Find a primary key for each new table produced.

- Duplicate in a new table the primary key of the table from which the repeating group was extracted or vice versa.

R1

Rollno	Phone
1	3242344
1	4564555
1	3112453
2	3452245
2	4323245

R2

Rollno	Name	Branch	Address
1	AAA	CSE	Hyderabad
2	BBB	ECE	Delhi

R1 and R2 are in 1NF.

Key of R1 = rollno

Key of R2 = (rollno, phone)

Example 2

Consider another example to check whether the given table is 1NF or not, if not try to convert into 1NF.

Un-normalized table R1

Course	Content
Programming	C,C++,java

Course	Content
Scripting language	HTML,javascript

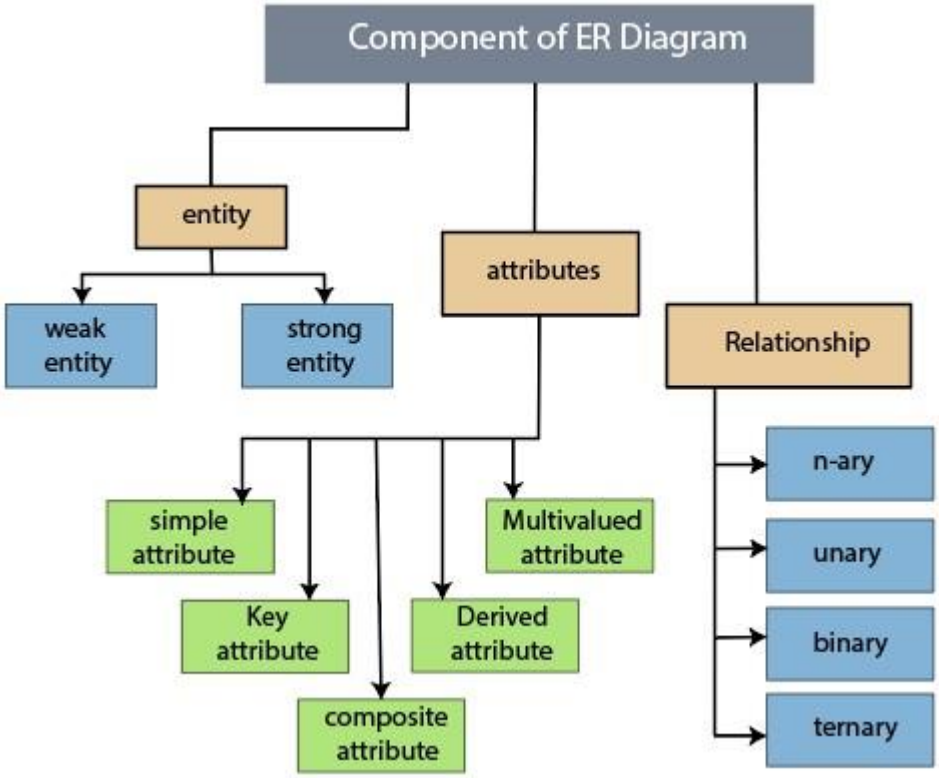
Normalized table (1NF)

Course	Content
Programming	C
Programming	C++
Programming	Java
Scripting language	HTML
Scripting language	javascript

e) Explain Components of E-R diagram?

Ans.

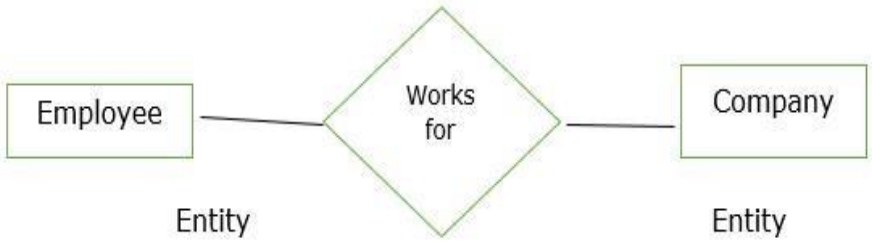
Components of E-R Diagram:-



Entity

It may be an object, person, place or event that stores data in a database. In a relationship diagram an entity is represented in rectangle form. For example, students, employees, managers, etc.

The entity is pictorially depicted as follows −



Entity set

It is a collection of entities of the same type which share similar properties. For example, a group of students in a college and students are an entity set.

Entity is characterised into two types as follows −

1)Strong entity set

2)Weak entity set

1)Strong entity set

The entity types which consist of key attributes or if there are enough attributes for forming a primary key attribute are called a strong entity set. It is represented by a single rectangle.

For Example:

Roll no of student

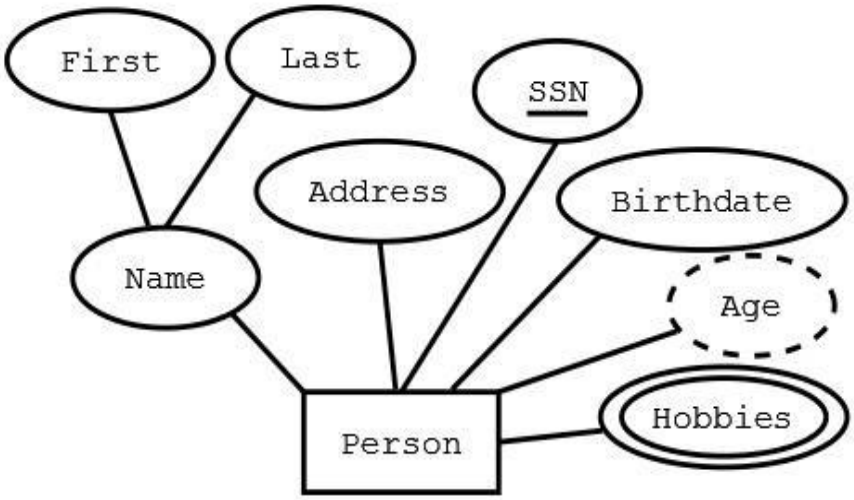
Emp ID of Employee

2)Weak entity set

An entity does not have a primary key attribute and depends on another strong entity via foreign key attribute. It is represented by a double rectangle.

Attributes

It is the name, thing etc. These are the data characteristics of entities or data elements and data fields.



Types of attributes

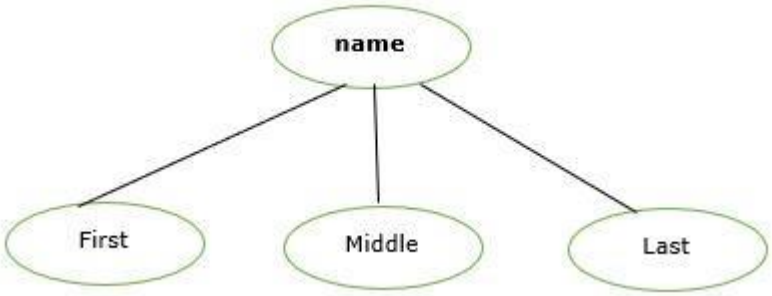
The types of attributes in the Entity Relationship (ER) model are as follows –

Single value attribute − These attributes contain a single value. For example, age, salary etc.

Multivalued attribute − They contain more than one value of a single entity. For example, phone numbers.

Composite attribute − The attributes which can be further divided. For example, Name-> First name, Middle name, last name

Derived attribute − The attribute that can be derived from others. For example, Date of Birth.

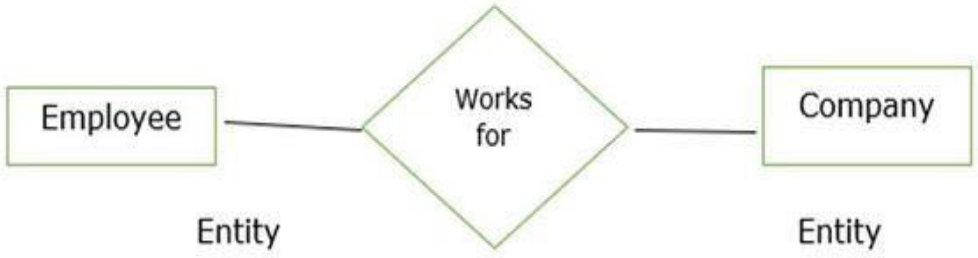


Relationship

It is used to describe the relation between two or more entities. It is represented by a diamond shape.

For Example, students study in college and employees work in a department.

The relationship is pictorially represented as follows −



Here works for is a relation between two entities.

Degree of Relationship

A relationship where a number of different entities set participate is called a degree of a relationship.

It is categorized into the following −

1)Unary Relationship

2)Binary Relationship

3)Ternary Relationship

4)n-ary Relationship

Q.4) Attempt any Three of the following (5 Marks each) 15

a) Explain Constraints with their types?

Ans.

- Relational constraints are the restrictions imposed on the database contents and operations.
- They ensure the correctness of data in the database.

Types of Constraints in DBMS-

1)_Domain constraint

2)Tuple Uniqueness constraint

3)Key constraint

4)Entity Integrity constraint

5)Referential Integrity constraint

1. Domain Constraint-

Domain constraint defines the domain or set of values for an attribute.

It specifies that the value taken by the attribute must be the atomic value from its domain.

Example-

Consider the following Student table-

STU_ID	Name	Age
S001	Akshay	20
S002	Abhishek	21

S003	Shashank	20
S004	Rahul	**A**

Here, value **'A'** is not allowed since only integer values can be taken by the age attribute.

2. Tuple Uniqueness Constraint-

Tuple Uniqueness constraint specifies that all the tuples must be necessarily unique in any relation.

Example-01:

Consider the following Student table-

STU_ID	**Name**	**Age**
S001	Akshay	20
S002	Abhishek	21
S003	Shashank	20
S004	Rahul	20

This relation satisfies the tuple uniqueness constraint since here all the tuples are unique.

Example-02:

Consider the following Student table-

STU_ID	**Name**	**Age**
S001	**Akshay**	**20**
S001	**Akshay**	**20**
S003	Shashank	20
S004	Rahul	20

This relation does not satisfy the tuple uniqueness constraint since here all the tuples are not unique.

3. Key Constraint-

Key constraint specifies that in any relation-

- All the values of primary key must be unique.
- The value of primary key must not be null.

Example-

Consider the following Student table

STU_ID	**Name**	**Age**

S001	Akshay	20
S001	Abhishek	21
S003	Shashank	20
S004	Rahul	20

This relation does not satisfy the key constraint as here all the values of primary key are not unique.

4. Entity Integrity Constraint-

- Entity integrity constraint specifies that no attribute of primary key must contain a null value in any relation.
- This is because the presence of null value in the primary key violates the uniqueness property.

Example-

Consider the following Student table-

STU_ID	**Name**	**Age**
S001	Akshay	20
S002	Abhishek	21
S003	Shashank	20
	Rahul	20

This relation does not satisfy the entity integrity constraint as here the primary key contains a NULL value.

5. Referential Integrity Constraint-

- This constraint is enforced when a foreign key references the primary key of a relation.
- It specifies that all the values taken by the foreign key must either be available in the relation of the primary key or be null.

Important Results-

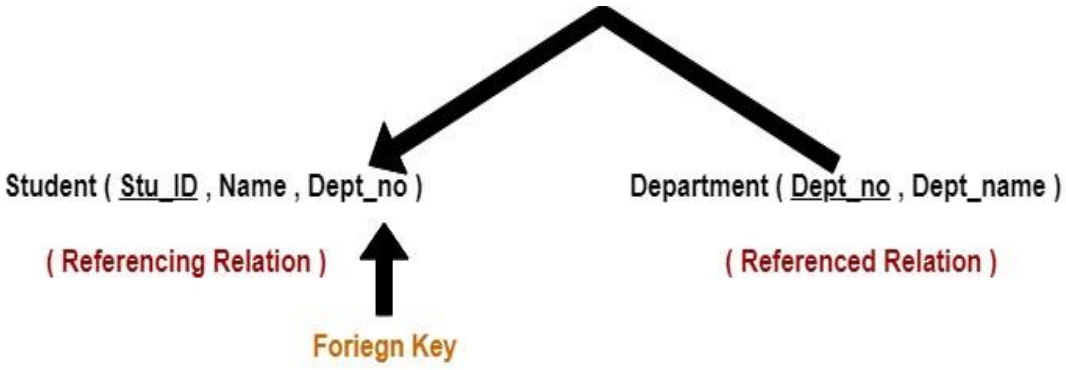
The following two important results emerges out due to referential integrity constraint-

- We can not insert a record into a referencing relation if the corresponding record does not exist in the referenced relation.
- We can not delete or update a record of the referenced relation if the corresponding record exists in the referencing relation.

Example-

Consider the following two relations- 'Student' and 'Department'.

Here, relation 'Student' references the relation 'Department'.



Student

STU_ID	**Name**	**Dept_no**
S001	Akshay	D10

S002	Abhishek	D10
S003	Shashank	D11
S004	Rahul	**D14**

Department

Dept_no	**Dept_name**
D10	ASET
D11	ALS
D12	ASFL
D13	ASHS

Here,

- The relation ‘Student’ does not satisfy the referential integrity constraint.
- This is because in relation ‘Department’, no value of primary key specifies department no. 14.
- Thus, referential integrity constraint is violated.

b) What is Functional Dependency?

Ans.

Functional Dependency

The functional dependency is a relationship that exists between two attributes. It typically exists between the primary key and non-key attribute within a table.

X → Y

The left side of FD is known as a determinant; the right side of the production is known as a dependent.

For example:

Assume we have an employee table with attributes: Emp_Id, Emp_Name, Emp_Address

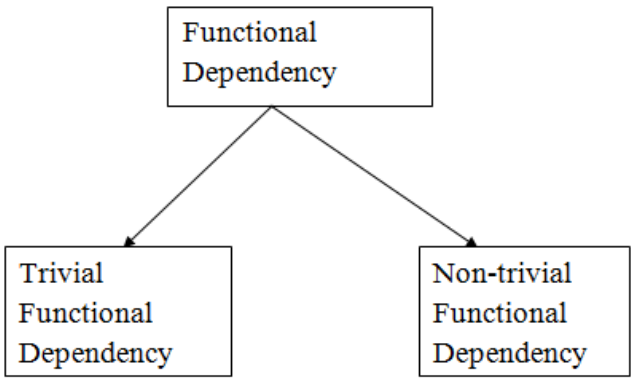
Here Emp_Id attribute can uniquely identify the Emp_Name attribute of employee table because if we know the Emp_Id, we can tell that employee name associated with it.

Functional dependency can be written as:

Emp_Id → Emp_Name

We can say that Emp_Name is functionally dependent on Emp_Id.

Types of Functional dependency



1. Trivial functional dependency

- A → B has trivial functional dependency if B is a subset of A.
- The following dependencies are also trivial like: A → A, B → B

Example:

Consider a table with two columns Employee_Id and Employee_Name.

1. {Employee_id, Employee_Name} → Employee_Id is a trivial functional dependency as
2. Employee_Id is a subset of {Employee_Id, Employee_Name}.
3. Also, Employee_Id → Employee_Id and Employee_Name → Employee_Name are trivial dependencies too.

2. Non-trivial functional dependency

- A → B has a non-trivial functional dependency if B is not a subset of A.

- When A intersection B is NULL, then A → B is called as complete non-trivial.

Example:

ID → Name,

Name → DOB

c) Explain 3NF Normal Form?
Ans.
Third Normal Form (3NF)

A relation will be in 3NF if it is in 2NF and not contain any transitive partial dependency.

3NF is used to reduce the data duplication. It is also used to achieve the data integrity.

If there is no transitive dependency for non-prime attributes, then the relation must be in third normal form.

A relation is in third normal form if it holds atleast one of the following conditions for every non-trivial function dependency X → Y.

X is a super key.

Y is a prime attribute, i.e., each element of Y is part of some candidate key.

Example:

EMPLOYEE_DETAIL table:

EMP_ID	EMP_NAME	EMP_ZIP	EMP_STATE	EMP_CITY
222	Harry	201010	UP	Noida
333	Stephan	02228	US	Boston
444	Lan	60007	US	Chicago
555	Katharine	06389	UK	Norwich
666	John	462007	MP	Bhopal

Super key in the table above:

{EMP_ID}, {EMP_ID, EMP_NAME}, {EMP_ID, EMP_NAME, EMP_ZIP}....so on

Candidate key: {EMP_ID}

Non-prime attributes: In the given table, all attributes except EMP_ID are non-prime.

Here, EMP_STATE & EMP_CITY dependent on EMP_ZIP and EMP_ZIP dependent on EMP_ID. The non-prime attributes (EMP_STATE, EMP_CITY) transitively dependent on super key(EMP_ID). It violates the rule of third normal form.

That's why we need to move the EMP_CITY and EMP_STATE to the new <EMPLOYEE_ZIP> table, with EMP_ZIP as a Primary key.

EMPLOYEE table:

EMP_ID	EMP_NAME	EMP_ZIP
222	Harry	201010
333	Stephan	02228
444	Lan	60007
555	Katharine	06389
666	John	462007

EMPLOYEE_ZIP table:

EMP_ZIP	EMP_STATE	EMP_CITY
201010	UP	Noida
02228	US	Boston
60007	US	Chicago
06389	UK	Norwich
462007	MP	Bhopal

d) Explain Cartesian product and Intersection Operation?

Ans.

Cartesian product operation-

It combines R1 and R2 without any condition. It is denoted by X.

Degree of R1 XR2 = degree of R1 + degree of R2

{degree = total no of columns}

Example

Consider R1 table −

RegNo	Branch	Section
1	CSE	A
2	ECE	B
3	CIVIL	A
4	IT	B

Table R2

Name	RegNo
Bhanu	2
Priya	4

R1 X R2

RegNo	Branch	Section	Name	RegNo
1	CSE	A	Bhanu	2
1	CSE	A	Priya	4
2	ECE	B	Bhanu	2
2	ECE	B	Priya	4
3	CIVIL	A	Bhanu	2
3	CIVIL	A	Priya	4

RegNo	Branch	Section	Name	RegNo
4	IT	B	Bhanu	2
4	IT	B	Priya	4

Intersection operation

It displays the common values in R1 & R2. It is denoted by ∩.

Syntax

$\prod_{regno}(R1) \cap \prod_{regno}(R2)$

Consider two sets,

A={1,2,4,6} and B={1,2,7}

Intersection of A and B

A ∩ B ={1,2}

Elements that are present in both sets A and B are present in the set obtained by intersection of A and B.

In relational algebra if R1 and R2 are two instances of relation then,

R1 ∩ R2 ={ x | x€ R1 and x € R2}

That is, the intersection of R1 and R2 only those tuples will be present that are in both R1 and R2

Example

Find all the customers whose account is in the bank and have taken out a loan.

The expression is as follows −

$\prod Name(Depositor) \cap \prod Name(Borrower)$

Depositor

ID	Name
1	A
2	B

ID	Name
3	C

Borrower

ID	Name
2	B
3	A
5	D

So, the intersection of depositor and borrower is as follows –

A
B

e) Difference between File processing systems Vs DBMS.

Ans.

Basis	File System	DBMS
Structure	The file system is software that manages and organizes the files in a storage medium within a computer.	DBMS is software for managing the database.
Data Redundancy	Redundant data can be present in a file system.	In DBMS there is no redundant data.
Backup and Recovery	It doesn't provide backup and recovery of data if it is lost.	It provides backup and recovery of data even if it is lost.
Query processing	There is no efficient query processing in the file system.	Efficient query processing is there in DBMS.
Consistency	There is less data consistency in the file system.	There is more data consistency because of the process of normalization.
Complexity	It is less complex as compared to DBMS.	It has more complexity in handling as compared to the file system.
Security Constraints	File systems provide less security in comparison to DBMS.	DBMS has more security mechanisms as compared to file systems.
Cost	It is less expensive than DBMS.	It has a comparatively higher cost than a file system.
Data Independence	There is no data independence.	In DBMS data independence exists.
User Access	Only one user can access data at a time.	Multiple users can access data at a time.
Meaning	The user has to write procedures for managing databases	The user not required to write procedures.
Sharing	Data is distributed in many files. So, not easy to share data	Due to centralized nature sharing is easy
Data Abstraction	It give details of storage and representation of data	It hides the internal details of Database
Integrity Constraints	Integrity Constraints are difficult to implement	Integrity constraints are easy to implement
Example	Cobol, C++	Oracle, SQL Server

Q.5 Write short notes on any three of the following (5 Marks each)15

i) Natural Join operation

Ans.

Natural Join (⋈)

Natural join does not use any comparison operator. It does not concatenate the way a Cartesian product does. We can perform a Natural Join only if there is at least one common attribute that exists between two relations. In addition, the attributes must have the same name and domain.

Natural join acts on those matching attributes where the values of attributes in both the relations are same.

Courses

CID	Course	Dept
CS01	Database	CS
ME01	Mechanics	ME
EE01	Electronics	EE

HoD

Dept	Head
CS	Alex
ME	Maya
EE	Mira

Courses ⋈ HoD

Dept	CID	Course	Head
CS	CS01	Database	Alex
ME	ME01	Mechanics	Maya

EE	EE01	Electronics	Mira

ii) BCNF Normal Form

Boyce Codd normal form (BCNF)

BCNF is the advance version of 3NF. It is stricter than 3NF.

A table is in BCNF if every functional dependency X → Y, X is the super key of the table.

For BCNF, the table should be in 3NF, and for every FD, LHS is super key.

Example: Let's assume there is a company where employees work in more than one department.

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 → EMP_COUNTRY

EMP_DEPT → {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:

EMP_ID	EMP_COUNTRY
264	India
264	India

EMP_DEPT table:

EMP_DEPT	DEPT_TYPE	EMP_DEPT_NO
Designing	D394	283
Testing	D394	300
Stores	D283	232
Developing	D283	549

EMP_DEPT_MAPPING table:

EMP_ID	EMP_DEPT
D394	283
D394	300
D283	232
D283	549

Functional dependencies:

EMP_ID → EMP_COUNTRY

EMP_DEPT → {DEPT_TYPE, EMP_DEPT_NO}

Candidate keys:

For the first table: EMP_ID
For the first table: EMP_DEPT
For the first table: {EMP_ID, EMP_DEPT}
Now, this is in BCNF because left side part of both the functional dependencies is a key.

iii) Types of Data Models
Ans.

Data Model

Data Model gives us an idea that how the final system will look like after its complete implementation. It defines the data elements and the relationships between the data elements. Data Models are used to show how data is stored, connected, accessed and updated in the database management system. Here, we use a set of symbols and text to represent the information so that members of the organisation can communicate and understand it. Though there are many data models being used nowadays but the Relational model is the most widely used model. Apart from the Relational model, there are many other types of data models about which we will study in details in this blog. Some of the Data Models in DBMS are:

1. Hierarchical Model
2. Network Model
3. Entity-Relationship Model
4. Relational Model

5.Object-Oriented Data Model

6.Object-Relational Data Model

7.Flat Data Model

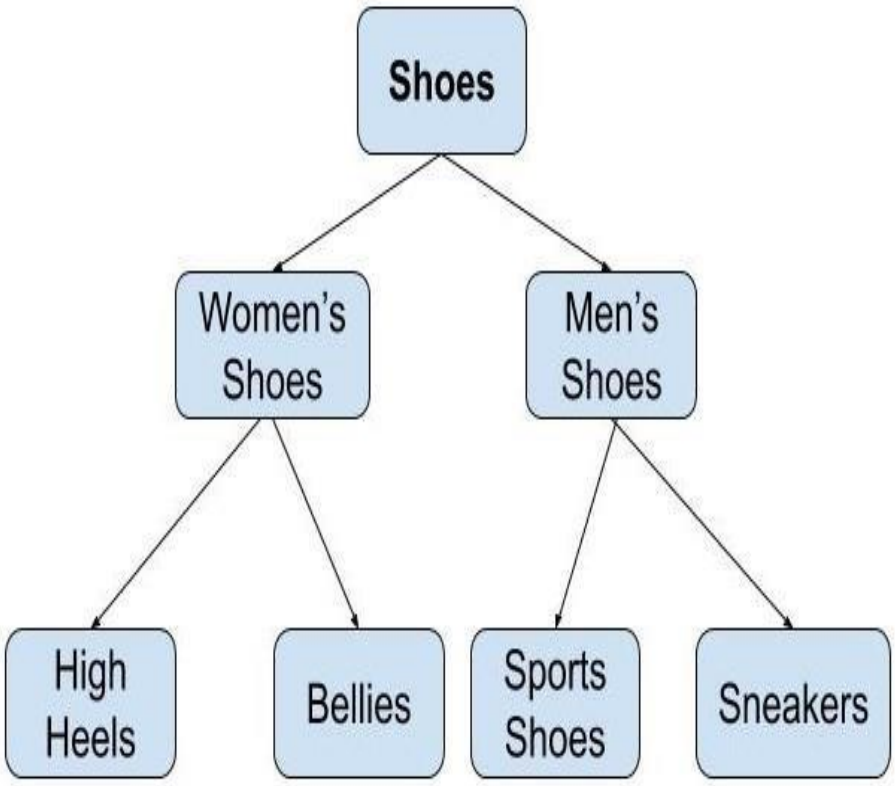
8.Semi-Structured Data Model

9.Associative Data Model

10.Context Data Model

1)*Hierarchical Model-*

Hierarchical Model was the first DBMS model. This model organises the data in the hierarchical tree structure. The hierarchy starts from the root which has root data and then it expands in the form of a tree adding child node to the parent node. This model easily represents some of the real-world relationships like food recipes, sitemap of a website
etc. ***Example:*** We can represent the relationship between the shoes present on a shopping website in the following way:



Hierarchical Model

Features of a Hierarchical Model

1. ***One-to-many relationship:*** The data here is organised in a tree-like structure where the one-to-many relationship is between the datatypes. Also, there can be only one path from parent to any node. ***Example:*** In the above example, if we want to go to the node *sneakers* we only have one path to reach there i.e through men's shoes node.
2. ***Parent-Child Relationship:*** Each child node has a parent node but a parent node can have more than one child node. Multiple parents are not allowed.
3. ***Deletion Problem:*** If a parent node is deleted then the child node is automatically deleted.
4. ***Pointers:*** Pointers are used to link the parent node with the child node and are used to navigate between the stored data. *Example:* In the above example the '*shoes*' node points to the two other nodes '*women shoes*' node and '*men's shoes*' node.

Advantages of Hierarchical Model

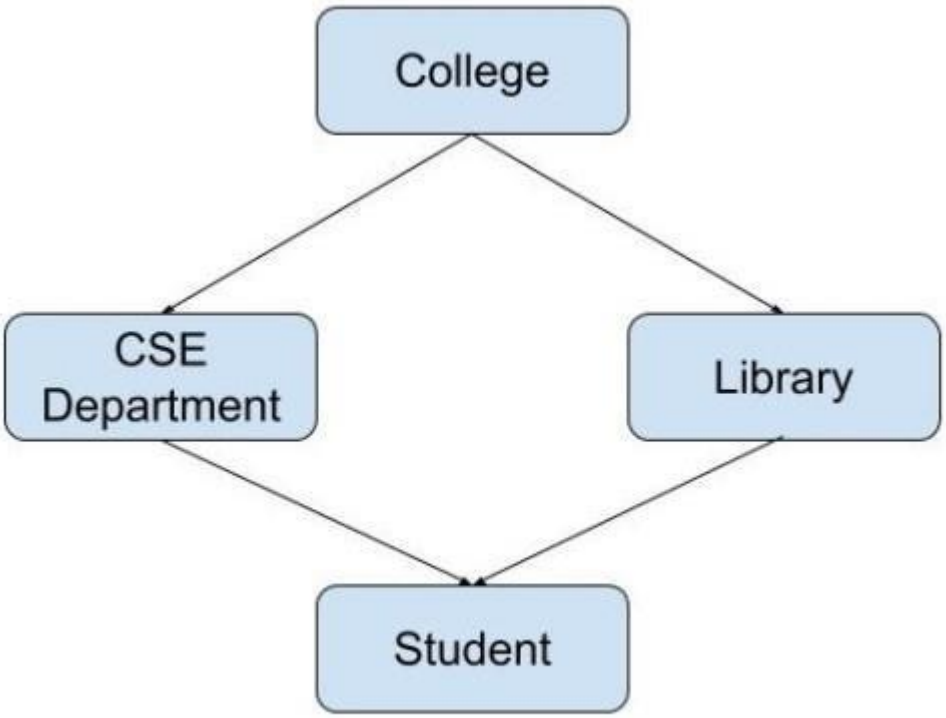
- It is very simple and fast to traverse through a tree-like structure.
- Any change in the parent node is automatically reflected in the child node so, the integrity of data is maintained.

Disadvantages of Hierarchical Model

- Complex relationships are not supported.
- As it does not support more than one parent of the child node so if we have some complex relationship where a child node needs to have two parent node then that can't be represented using this model.
- If a parent node is deleted then the child node is automatically deleted.

2)Network Model

This model is an extension of the hierarchical model. It was the most popular model before the relational model. This model is the same as the hierarchical model, the only difference is that a record can have more than one parent. It replaces the hierarchical tree with a graph. ***Example:*** In the example below we can see that node student has two parents i.e. CSE Department and Library. This was earlier not possible in the hierarchical model.



Network Model

Features of a Network Model

1. ***Ability to Merge more Relationships:*** In this model, as there are more relationships so data is more related. This model has the ability to manage one-to-one relationships as well as many-to-many relationships.
2. ***Many paths:*** As there are more relationships so there can be more than one path to the same record. This makes data access fast and simple.
3. ***Circular Linked List:*** The operations on the network model are done with the help of the circular linked list. The current position is maintained with the help of a program and this position navigates through the records according to the relationship.

Advantages of Network Model

- The data can be accessed faster as compared to the hierarchical model. This is because the data is more related in the network model and there can be more than one path to reach a particular node. So the data can be accessed in many ways.
- As there is a parent-child relationship so data integrity is present. Any change in parent record is reflected in the child record.

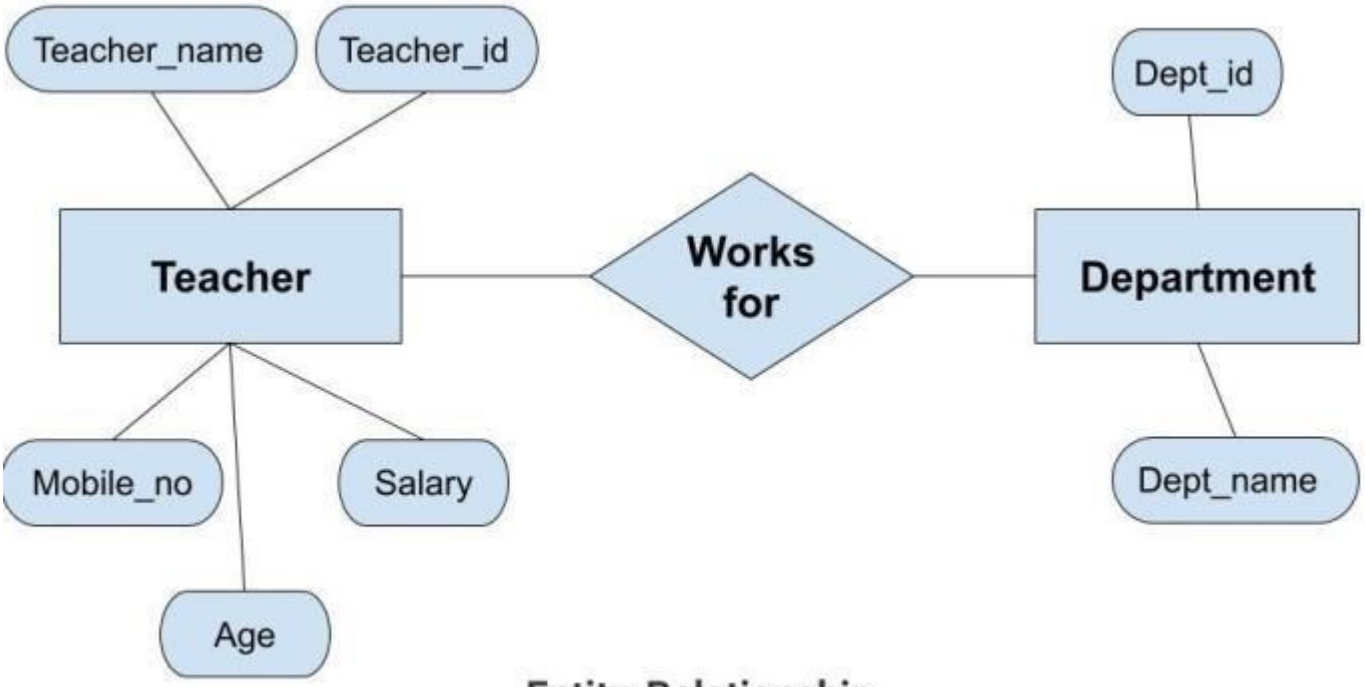
Disadvantages of Network Model

- As more and more relationships need to be handled the system might get complex. So, a user must be having detailed knowledge of the model to work with the model.
- Any change like updation, deletion, insertion is very complex.

3)Entity-Relationship Model

Entity-Relationship Model or simply ER Model is a high-level data model diagram. In this model, we represent the real-world problem in the pictorial form to make it easy for the stakeholders to understand. It is also very easy for the developers to understand the system by just looking at the ER diagram. We use the ER diagram as a visual tool to represent an ER Model. ER diagram has the following three components:

- ***Entities:*** Entity is a real-world thing. It can be a person, place, or even a concept. *Example:* Teachers, Students,Course, Building, Department, etc are some of the entities of a School Management System.
- ***Attributes:*** An entity contains a real-world property called attribute. This is the characteristics of that
  attribute. *Example:* The entity teacher has the property like teacher id, salary, age, etc.
- ***Relationship:*** Relationship tells how two attributes are related. *Example:* Teacher works for a department.
- Example



Entity-Relationship Model

In the above diagram, the entities are Teacher and Department. The attributes of ***Teacher*** entity are Teacher_Name, Teacher_id,

Age, Salary, Mobile_Number. The attributes of
entity ***Department*** entity are Dept_id, Dept_name. The two entities are connected using the relationship. Here, each teacher works for a department.

Features of ER Model

- ***Graphical Representation for Better Understanding:*** It is very easy and simple to understand so it can be used by the developers to communicate with the stakeholders.
- ***ER Diagram:*** ER diagram is used as a visual tool for representing the model.
- ***Database Design:*** This model helps the database designers to build the database and is widely used in database design.

Advantages of ER Model

- ***Simple:*** Conceptually ER Model is very easy to build. If we know the relationship between the attributes and the entities we can easily build the ER Diagram for the model.
- ***Effective Communication Tool***: This model is used widely by the database designers for communicating their ideas.
- ***Easy Conversion to any Model***: This model maps well to the relational model and can be easily converted relational model by converting the ER model to the table. This model can also be converted to any other model like network model, hierarchical model etc.

Disadvatages of ER Model

- ***No industry standard for notation:*** There is no industry standard for developing an ER model. So one developer might use notations which are not understood by other developers.

might use notations which are not understood by other developers.

- ***Hidden information:*** Some information might be lost or hidden in the ER model. As it is a high-level view so there are chances that some details of information might be hidden.

4)Relational Model

Relational Model is the most widely used model. In this model, the data is maintained in the form of a two-dimensional table. All the information is stored in the form of row and columns. The basic structure of a relational model is tables. So, the tables are also called *relations* in the relational model. ***Example:*** In this example, we have an Employee table.

Emp_id	Emp_name	Job_name	Salary	Mobile_no	Dep_id	Project_id
AfterA001	John	Engineer	100000	9111037890	2	99
AfterA002	Adam	Analyst	50000	9587569214	3	100
AfterA003	Kande	Manager	890000	7895212355	2	65

EMPLOYEE TABLE

Features of Relational Model

- ***Tuples***: Each row in the table is called tuple. A row contains all the information about any instance of the object. In the above example, each row has all the information about any specific individual like the first row has information about John.
- ***Attribute or field:*** Attributes are the property which defines the table or relation. The values of the attribute should be from the same domain. In the above example, we have different attributes of the *employee* like Salary, Mobile_no, etc.

Advnatages of Relational Model

- ***Simple:*** This model is more simple as compared to the network and hierarchical model.
- ***Scalable:*** This model can be easily scaled as we can add as many rows and columns we want.
- ***Structural Independence:*** We can make changes in database structure without changing the way to access the data. When we can make changes to the database structure without affecting the capability to DBMS to access the data we can say that structural independence has been achieved.

Disadvantages of Relatinal Model

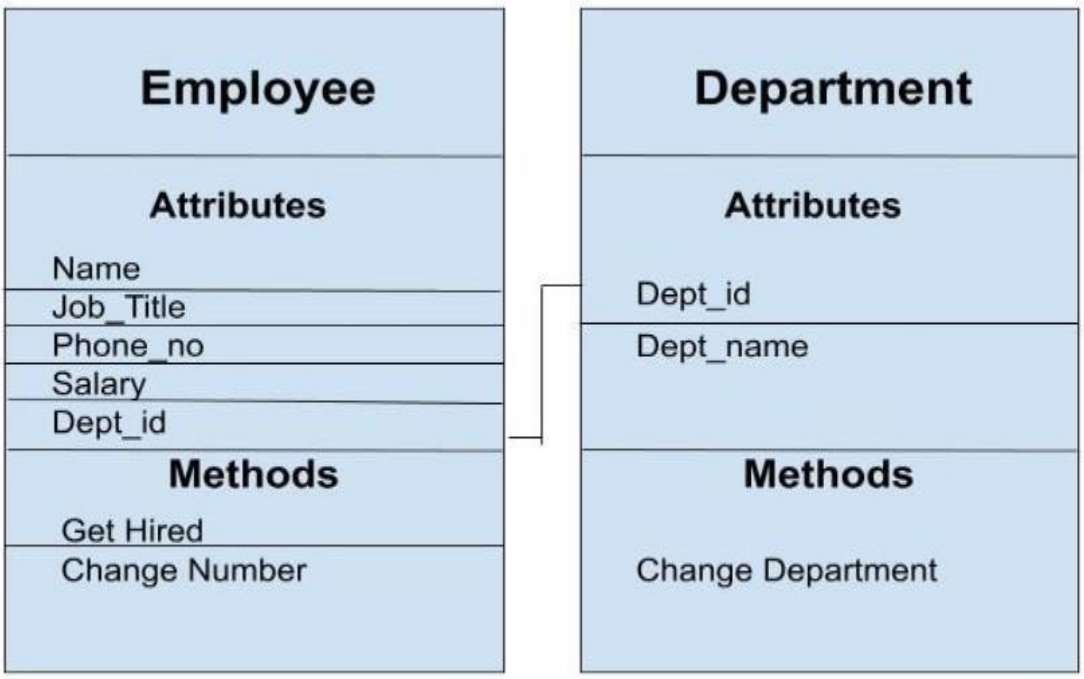
- ***Hardware Overheads:*** For hiding the complexities and making things easier for the user this model requires more powerful hardware computers and data storage devices.

- ***Bad Design:*** As the relational model is very easy to design and use. So the users don't need to know how the data is stored in order to access it. This ease of design can lead to the development of a poor database which would slow down if the database grows.

But all these disadvantages are minor as compared to the advantages of the relational model. These problems can be avoided with the help of proper implementation and organisation.

5)Object-Oriented Data Model

The real-world problems are more closely represented through the object-oriented data model. In this model, both the data and relationship are present in a single structure known as an object. We can store audio, video, images, etc in the database which was not possible in the relational model(although you can store audio and video in relational database, it is adviced not to store in the relational database). In this model, two are more objects are connected through links. We use this link to relate one object to other objects. This can be understood by the example given below.



Object_Oriented_Model

In the above example, we have two objects Employee and Department. All the data and relationships of each object are contained as a single unit. The attributes like Name, Job_title of the employee and the methods which will be performed by that object are stored as a single object. The two objects are connected through a common attribute i.e the Department_id and the communication between these two will be done with the help of this common id.

6)Object-Relational Model

As the name suggests it is a combination of both the relational model and the object-oriented model. This model was built to fill the gap between object-oriented model and the relational model. We can have many advanced features like we can make complex data types according to our requirements using the existing data types. The problem with this model is that this can get complex and difficult to handle. So, proper understanding of this model is required.

7)Flat Data Model

It is a simple model in which the database is represented as a table consisting of rows and columns. To access any data, the computer has to read the entire table. This makes the modes slow and inefficient.

8)Semi-Structured Model

Semi-structured model is an evolved form of the relational model. We cannot differentiate between data and schema in this model. ***Example:*** Web-Based data sources which we can't differentiate between the schema and data of the website. In this model, some entities may have missing attributes while others may have an extra attribute. This model gives flexibility in storing the data. It also gives flexibility to the
attributes. ***Example:*** If we are storing any value in any attribute then that value can be either atomic value or a collection of values.

9)Associative Data Model

Associative Data Model is a model in which the data is divided into two parts. Everything which has independent existence is called as an *entity* and the relationship among these entities are called *association*. The data divided into two parts are called items and links.

- ***Item***: Items contain the name and the identifier(some numeric value).
- ***Links:*** Links contain the identifier, source, verb and subject.

Example: Let us say we have a statement "The world cup is being hosted by London from 30 May 2020". In this data two links need to be stored:

1. The world cup is being hosted by London. The source here is 'the world cup', the verb 'is being' and the target is 'London'.
2. ...from 30 May 2020. The source here is the previous link, the verb is 'from' and the target is '30 May 2020'.

This is represented using the table as follows:

Items

Identifiers	Name
89	The world cup
95	Is being hosted
40	By London
44	from
10	30 May 2020

Links

Identifiers	Source	Verb	Target
70	89	95	40
75	70	44	10

ASSOCIATIVE MODEL

10)Context Data Model

Context Data Model is a collection of several models. This consists of models like network model, relational models etc. Using this model we can do various types of tasks which are not possible using any model alone.

iv) Types of File Organization

What is File Organization?

File Organization refers to the logical relationships among various records that constitute the file, particularly with respect to the means of identification and access to any specific record. In simple terms, Storing the files in certain order is called file Organization. **File Structure** refers to the format of the label and data blocks and of any logical control record.

Types of File Organizations –

Various methods have been introduced to Organize files. These particular methods have advantages and disadvantages on the basis of access or selection . Thus it is all upon the programmer to decide the best suited file Organization method according to his requirements.

Some types of File Organizations are :

- Sequential File Organization
- Heap File Organization
- Hash File Organization
- B+ Tree File Organization
- Clustered File Organization

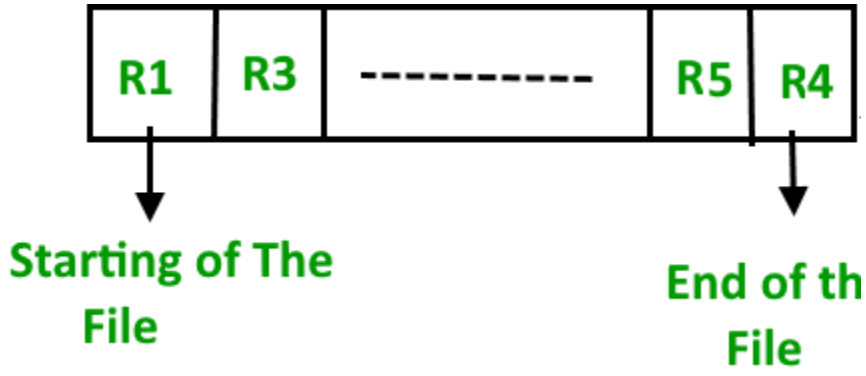
We will be discussing each of the file Organizations in further sets of this article along with differences and advantages/

disadvantages of each file Organization methods.

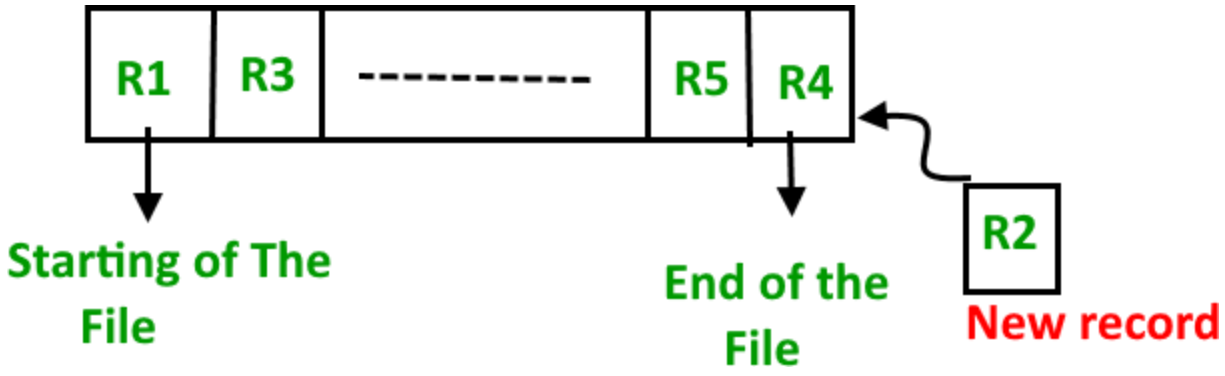
Sequential File Organization –

The easiest method for file Organization is Sequential method. In this method the file are stored one after another in a sequential manner. There are two ways to implement this method:

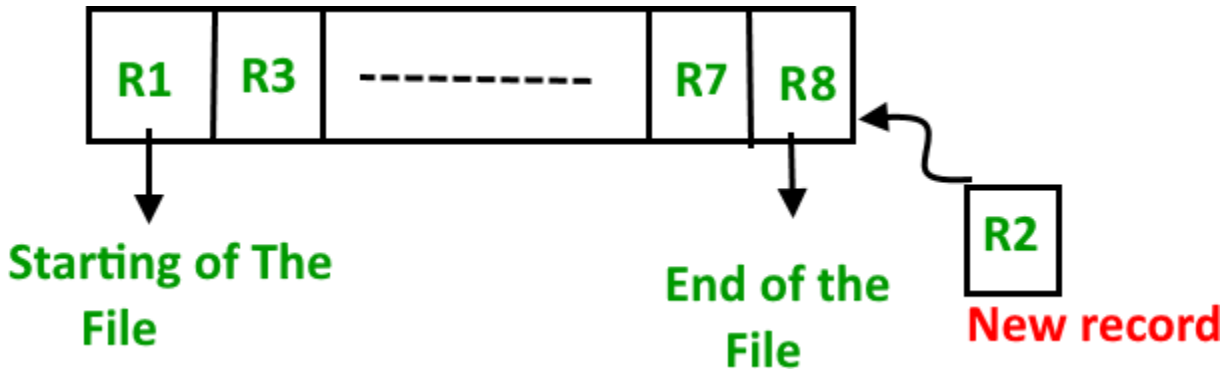
- **Pile File Method –** This method is quite simple, in which we store the records in a sequence i.e one after other in the order in which they are inserted into the tables.



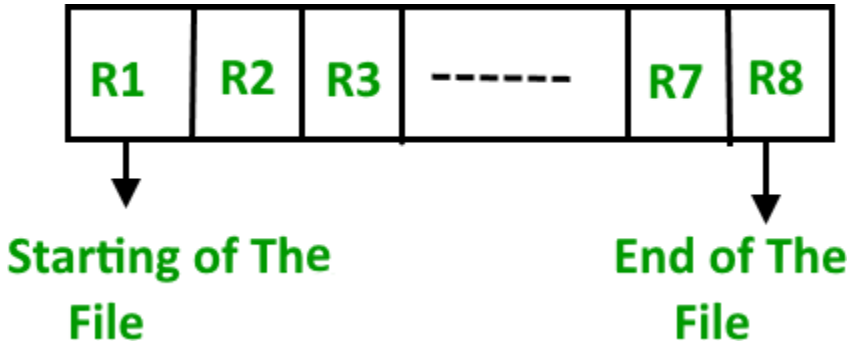
1. **Insertion of new record –**
   Let the R1, R3 and so on upto R5 and R4 be four records in the sequence. Here, records are nothing but a row in any table.
   Suppose a new record R2 has to be inserted in the sequence, then it is simply placed at the end of the file.



- **Sorted File Method –**In this method, As the name itself suggest 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.



1. **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 .



Pros and Cons of Sequential File Organization – Pros –

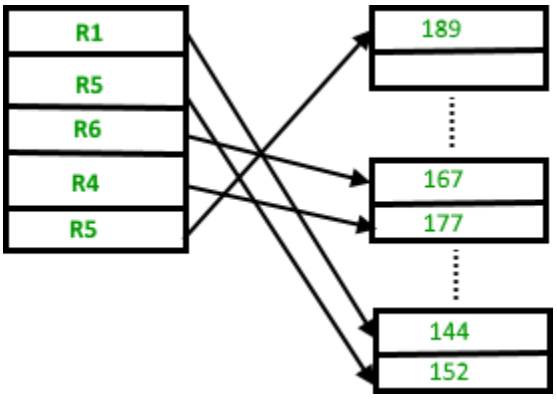
- Fast and efficient method for huge amount of data.
- Simple design.
- Files can be easily stored in magnetic tapes i.e cheaper storage mechanism.

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.

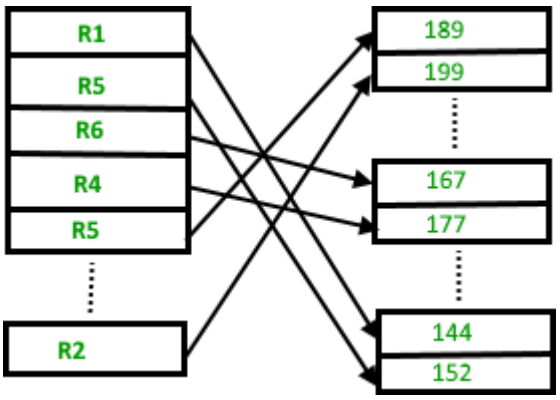
Heap File Organization –

Heap File Organization works with data blocks. In this method records are inserted at the end of the file, into the data blocks. No Sorting or Ordering is required in this method. If a data block is full, the new record is stored in some other block, Here the other data block need not be the very next data block, but it can be any block in the memory. It is the responsibility of DBMS to store and manage the new records.



Insertion of new record –

Suppose we have four records in the heap R1, R5, R6, R4 and R3 and suppose a new record R2 has to be inserted in the heap then, since the last data block i.e data block 3 is full it will be inserted in any of the data blocks selected by the DBMS, lets say data block



If we want to search, delete or update data in heap file Organization the we will traverse the data from the beginning of the file till we get the requested record. Thus if the database is very huge, searching, deleting or updating the record will take a lot of time.

Pros and Cons of Heap File Organization – Pros –

- Fetching and retrieving records is faster than sequential record but only in case of small databases.
- When there is a huge number of data needs to be loaded into the database at a time, then this method of file Organization is best suited.

Cons –

- Problem of unused memory blocks.
- Inefficient for larger databases.

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.

Hash File Organization:

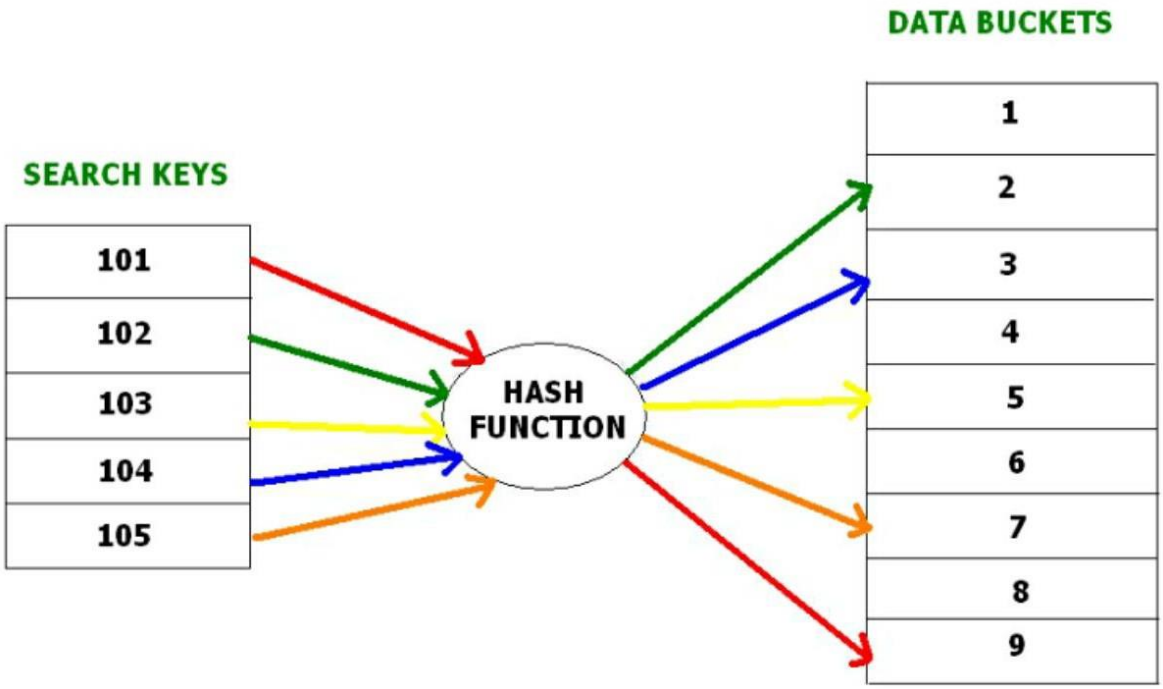
- **Data bucket –** Data buckets are the memory locations where the records are stored. These buckets are also considered

  as *Unit Of Storage*.

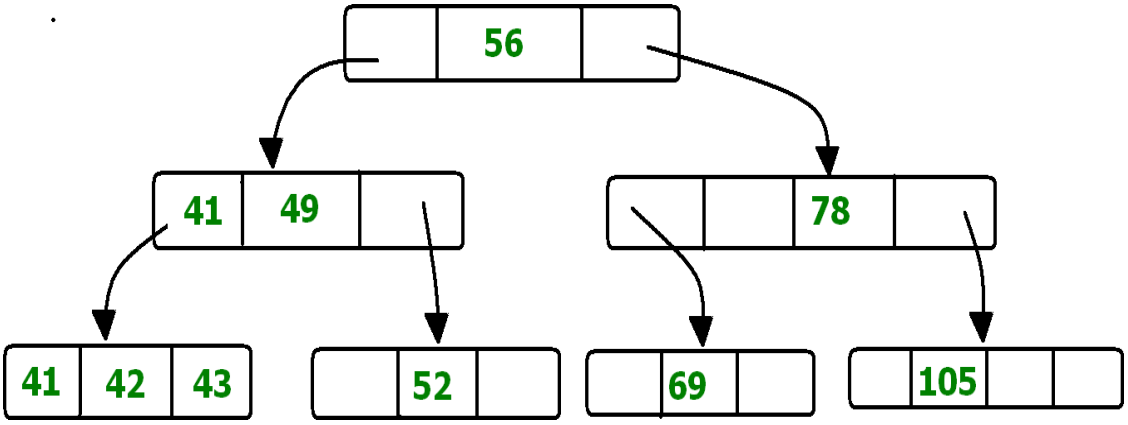
- **Hash Function –** Hash function is a mapping function that maps all the set of search keys to actual record address. Generally, hash function uses the primary key to generate the hash index – address of the data block. Hash function can be simple mathematical function to any complex mathematical function.
- **Hash Index-**The prefix of an entire hash value is taken as a hash index. Every hash index has a depth value to signify how many bits are used for computing a hash function. These bits can address 2n buckets. When all these bits are consumed ? then the depth value is increased linearly and twice the buckets are allocated.

B+ Tree File Organization –

B+ Tree, as the name suggests, 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. The information in leaf nodes always remain a sorted sequential linked list.



In the above diagram 56 is the root node which is also called the main node of the tree.
The intermediate nodes here, just consist the address of leaf nodes. They do not contain any actual record. Leaf nodes consist of the actual record. All leaf nodes are balanced.

Pros and Cons of B+ Tree File Organization – Pros –

- Tree traversal is easier and faster.
- Searching becomes easy as all records are stored only in leaf nodes and are sorted sequential linked list.
- There is no restriction on B+ tree size. It may grows/shrink as the size of data increases/decreases.

Cons –

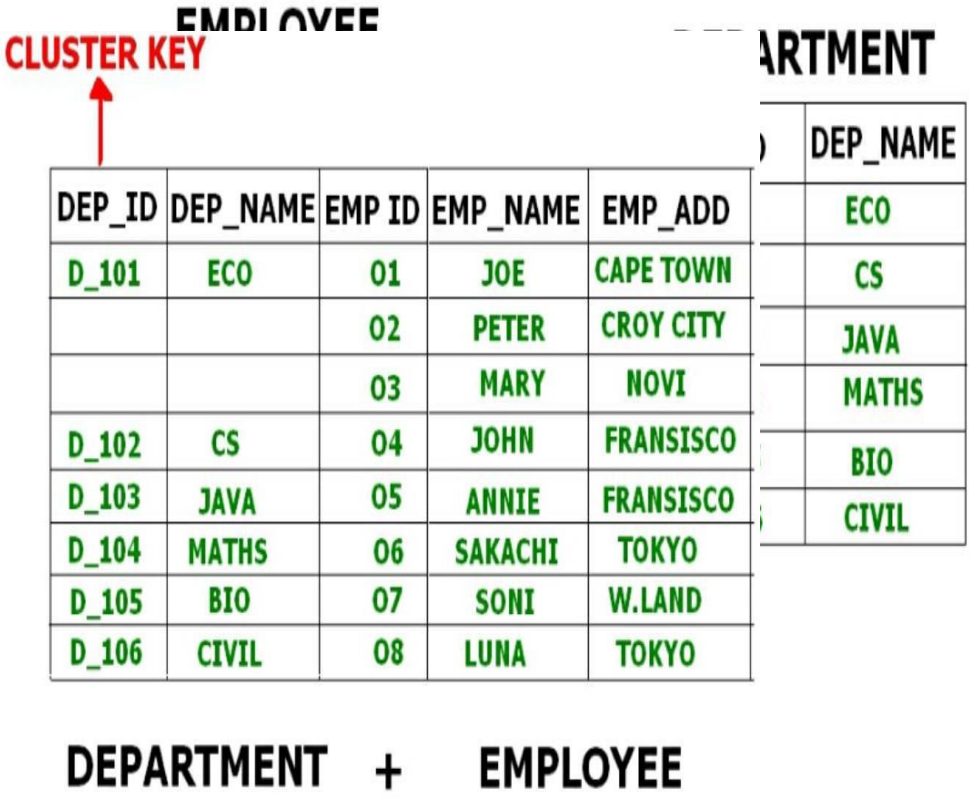
- Inefficient for static tables.

Cluster File Organization –

In cluster file organization, two or more related tables/records are stored within same file known as clusters. These files will have two or more tables in the same data block and the key attributes which are used to map these table together are stored only once.

Thus it lowers the cost of searching and retrieving various records in different files as they are now combined and kept in a single cluster.
For example we have two tables or relation Employee and Department. These table are related to each other.

Therefore these table are allowed to combine using a join operation and can be seen in a cluster file.



If we have to insert, update or delete any record we can directly do so. Data is sorted based on the primary key or the key with which searching is done. **Cluster key** is the key with which joining of the table is performed.

Types of Cluster File Organization – There are two ways to implement this method:

1.Indexed Clusters –

In Indexed clustering the records are group based on the cluster key and stored together. The above mentioned example of the Employee and Department relationship is an example of Indexed Cluster where the records are based on the Department ID.

2.Hash Clusters –

This is very much similar to indexed cluster with only difference that instead of storing the records based on cluster key, we generate hash key value and store the records with same hash key value.

v) Advantages and Disadvantages of DBMS?

Ans.

Advantages of DBMS

- **Data redundancy and inconsistency:** Redundancy is the concept of repetition of data i.e. each data may have more than a single copy. The file system cannot control the redundancy of

data as each user defines and maintains the needed files for a specific application to run. There may be a possibility that two users are maintaining the data of the same file for different applications. Hence changes made by one user do not reflect in files used by second users, which leads to inconsistency of data. Whereas DBMS controls redundancy by maintaining a single repository of data that is defined once and is accessed by many users. As there is no or less redundancy, data remains consistent.

- **Data sharing:** The file system does not allow sharing of data or sharing is too complex. Whereas in DBMS, data can be shared easily due to a centralized system.
- **Data concurrency:** Concurrent access to data means more than one user is accessing the same data at the same time. Anomalies occur when changes made by one user get lost because of changes made by another user. The file system does not provide any procedure to stop anomalies. Whereas DBMS provides a locking system to stop anomalies to occur.
- **Data searching:** For every search operation performed on the file system, a different application program has to be written. While DBMS provides inbuilt searching operations. The user only has to write a small query to retrieve data from the database.
- **Data integrity:** There may be cases when some constraints need to be applied to the data before inserting it into the database. The file system does not provide any procedure to check these constraints automatically. Whereas DBMS maintains data integrity by enforcing user-defined constraints on data by itself.
- **System crashing:** In some cases, systems might have crashed due to various reasons. It is a bane in the case of file systems because once the system crashes, there will be no recovery of the data that's been lost. A DBMS will have the recovery manager which retrieves the data making it another advantage over file systems.
- **Data security:** A file system provides a password mechanism to protect the database but how long can the password be
  protected? No one can guarantee that. This doesn't happen in the case of DBMS. DBMS has specialized features that help provide shielding to its data.

- **Backup:** It creates a backup subsystem to restore the data if required.
- **Interfaces**: It provides different multiple user interfaces like graphical user interface and application program interface.
- **Easy Maintenance**: It is easily maintainable due to its centralized nature.

Disadvantage of DBMS

1. Increased Cost:

These are different types of costs:

a)Cost of Hardware and Software –

This is the first disadvantage of the database management system. This is because, for DBMS, it is mandatory to have a high-speed processor and also a large memory size. After all, nowadays there is a large amount of data in every field which needs to be store safely and with security.

The requirement of this large amount of space and a high-speed processor needs expensive hardware and expensive software too. That is, there is a requirement for sophisticated hardware and software which means that we need to upgrade the hardware which is used for the file-based system. Hardware and Software, both require maintenance which costs very high. All the operating, Training (all levels including programming, application development, and database administration), licensing, and regulatory compliance costs very high.

b) Cost of Staff Training –

Educated staff (database administrator, application programmers, data entry operations) who maintains the database management system also requires a good amount. We need the database system designers to be hired along with application programmers. Alternatively, the services of some software houses need to be taken. So there is a lot of money which needs to be spent on developing software.

c)Cost of Data Conversion –

We need to convert our data into a database management system, there is a requirement of a lot of money as it adds to the cost of the database management system. This is because for this conversion we need to hire database system designers whom we have to pay a lot of money and also services of some software house will be required. All this shows that a high initial investment for hardware, software, and trained staff is required by DBMS. So, altogether Database Management System results in a costlier system

.

2. Complexity:

As we all know that nowadays all companies are using the database management system as it fulfills lots of requirements and also solves the problem. But a problem arises, that is all this functionality has made the database management system an extremely complex software. For the proper requirement of DBMS, it is very important to have a good knowledge of it by the developers, DBA, designers, and also the end-users. This is because if any one of them does not acquire proper and complete skills then this may lead to data loss or database failure.

These failures may lead to bad design decisions due to which there may be serious and bad consequences for the organization. So this complex system needs to be understood by everyone using it. As it cannot be managed very easily. All this shows that a database management system is not a child's game as it cannot be managed very easily. It requires a lot of management. A good staff is needed to manage this database at times when it becomes very complicated to decide where to pick data from and where to save it.

3. Currency Maintenance:

This is very necessary to keep your system current because efficiency which is one of the biggest factors and needs to be overlooked must be maximized. That is we need to maximize the efficiency of the database system to keep our system current. For this, frequent updation must be performed on all the components as new threats come daily. DBMS should be updated according to the current scenario. Also, security measures must be implemented. Due to advancement in database technology, training cost tends to be significant.

4. Performance:

The traditional file system is written for small organizations and for some specific applications due to which performance is generally very good. But for the small-scale firms, DBMS does not give a good performance as its speed is very slow. As a result, some applications will not run as fast as they could. Hence it is not good to use DBMS for small firms. Because performance is a factor that is overlooked by everyone. If performance is good then everyone (developers, designers, end-users) will use it easily and it will be user-friendly too. As the speed of the system totally depends on the performance so performance needs to be good.

5. Frequency Upgrade/Replacement Cycles:

Nowadays in this world, we need to stay up-to-date about the latest technologies, developments arriving in the market.

Frequent upgrade of the products is done by the DBMS vendors to add new functionality to the systems. New upgrade versions of the software often come bundled. Sometimes these updates also need hardware upgrades. Sometimes these changes and updates are so fast that the users find it difficult to work with that system because it is not easy to learn new commands and understand them again when the new upgrades are done. All these upgrades also cost money to train users, designers, etc. to use the new feature.