DATABASE MANAGEMENT SYSTEM

BRANCH- COMPUTER SCIENCE ENGG.

SUBJECT:- DBMS

SEM:- 4TH

PREPARED BY:- PRADEEP KUMAR GIRI

Q 1. Explain the comparison of file management system and database system. (2017(w)-5(b)-5marks)

Ans:-

File Management System	*Database Management System*
File System is a general, easy-to-use system to store general files which require less security and constraints.	Database management system is used when security constraints are high.
Data Redundancy is more in file management system.	Data Redundancy is less in database management system.
Data Inconsistency is more in file system.	Data Inconsistency is less in database management system.
Centralisation is hard to get when it comes to File Management System.	Centralisation is achieved in Database Management System.
User locates the physical address of the files to access data in File Management System.	In Database Management System, user is unaware of physical address where data is stored.
Security is low in File Management System.	Security is high in Database Management System.
File Management System stores unstructured data as isolated data files/entities.	Database Management System stores structured data which have well defined constraints and interrelation.

Q.2 Define Database, Explain advantages and disadvantages of DBMS. (7-Mark-2017-7(c))

Ans:- Am integration of related data /information ,which can be referred by an user or a group of users for individual/organization activities like creation, addition, deletion, updation and even sharing of information etc. .

"A database management system (DBMS) is a collection of programs that manage the database structure and controls access to the data stored in the database".

- The DBMS serves as the intermediate between the user and the database. The database structure is stored as a collection of files. These data can be accessed in those files through the DBMS.
- The DBMS hides much of the database's internal complexity from the application programs and users.

Advantages of DBMS

1. Controlling of Redundancy :

Data redundancy refers to the duplication of data (i.e storing same data multiple times). In a database system, by having a centralized database and centralized control of data by the DBA the

unnecessary duplication of data is avoided. It also eliminates the extra time for processing the large volume of data. It results in saving the storage space.

2. Improved Data Sharing :

DBMS allows a user to share the data in any number of application programs.

3. Data Integrity :

Integrity means that the data in the database is accurate. Centralized control of the data helps in permitting the administrator to define integrity constraints to the data in the database.

4. Security :

Having complete authority over the operational data, enables the DBA in ensuring that the only mean of access to the database is through proper channels. The DBA can define authorization checks to be carried out whenever access to sensitive data is attempted.

5. Data Consistency :

By eliminating data redundancy, we greatly reduce the opportunities for inconsistency. For example: is a customer address is stored only once, we cannot have disagreement on the stored values. Also updating data values is greatly simplified when each value is stored in one place only. Finally, we avoid the wasted storage that results from redundant data storage.

6. Efficient Data Access :

In a database system, the data is managed by the DBMS and all access to the data is through the DBMS providing a key to effective data processing

7. Enforcements of Standards :

With the centralized of data, DBA can establish and enforce the data standards which may include the naming conventions, data quality standards etc.

8. Data Independence :

Ina database system, the database management system provides the interface between the application programs and the data. When changes are made to the data representation, the Meta data obtained by the DBMS is changed but the DBMS is continues to provide the data to application program in the previously used way. The DBMs handles the task of transformation of data wherever necessary.

Disadvantages of DBMS

The disadvantages of the database approach are summarized as follows:

1. Complexity: The provision of the functionality that is expected of a good DBMS makes the DBMS an extremely complex piece of software. Database designers, developers, database administrators and end-users must understand this functionality to take full advantage of it. Failure to understand the system can lead to bad design decisions, which can have serious consequences for an organization.

2. Size: The complexity and breadth of functionality makes the DBMS an extremely large piece of software, occupying many megabytes of disk space and requiring substantial amounts of memory to run efficiently.

3. Performance: Typically, a File Based system is written for a specific application, such as invoicing. As result, performance is generally very good. However, the DBMS is written to be more general, to cater for many applications rather than just one. The effect is that some applications may not run as fast as they used to.

4. Higher impact of a failure: The centralization of resources increases the vulnerability of the system. Since all users and applications rely on the ~vailabi1ity of the DBMS, the failure of any component can bring operations to a halt.

5. Cost of DBMS: The cost of DBMS varies significantly, depending on the environment and functionality provided. There is also the recurrent annual maintenance cost.

6. Additional Hardware costs: The disk storage requirements for the DBMS and the database may necessitate the purchase of additional storage space. Furthermore, to achieve the required performance it may be necessary to purchase a larger machine, perhaps even a machine dedicated to running the DBMS. The procurement of additional hardware results in further expenditure.

7. Cost of Conversion: In some situations, the cost of the DBMS and extra hardware may be insignificant compared with the cost of converting existing applications to run on the new DBMS and hardware. This cost also includes the cost of training staff to use these new systems and possibly the employment of specialist staff to help with conversion and running of the system. This cost is one of the main reasons why some organizations feel tied to their current systems and cannot switch to modern database technology.

Q3. Explain three levels of database architecture. (2019(w)-3-10MARKS)

OR Explain the three levels of data abstraction.

Define is data abstraction? (2 mark (2019(w)-1(j))

This architecture is used to provide framework which is extremely useful in describing general database concepts and for explaining the structure of individual systems.

The purpose of designing a generalized database is so that it must have the capability to transform the query asked by the user into programming form so that the system can understand it and will be able to retrieve back the answer of the query.

It is divided into three levels.

I. External level or view level or user view

II. Conceptual level or logical level or global view

III. Physical level or internal level or internal view

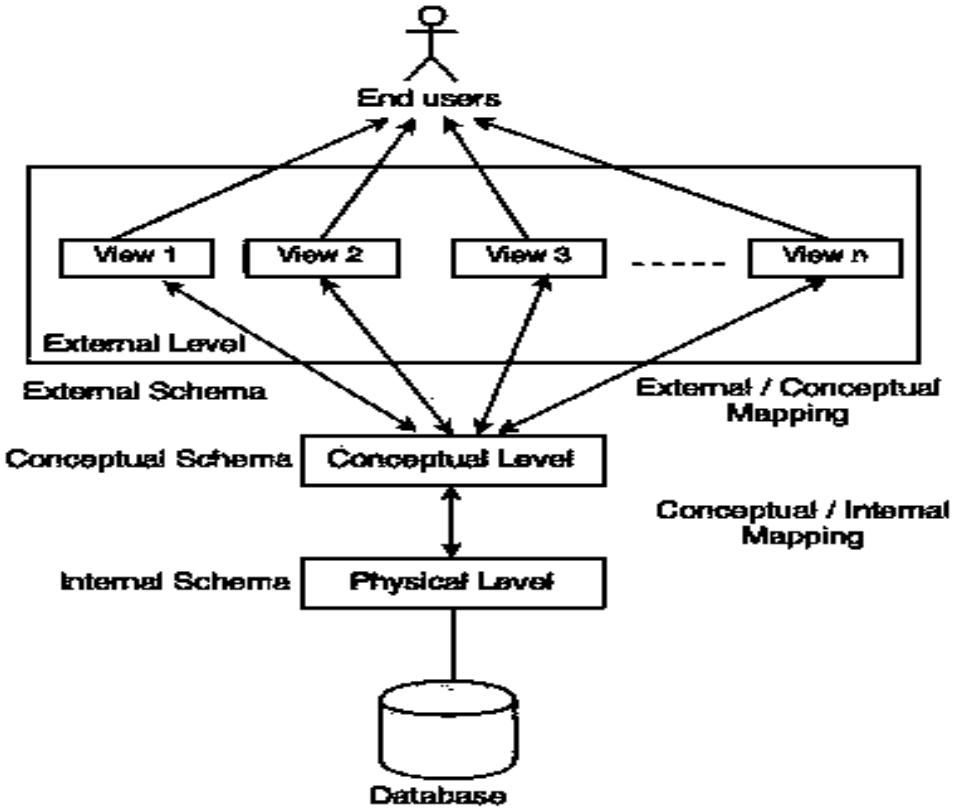


Fig. Three Level Architechture of DBMS

1. Physical Level:-

The lowest level of abstraction describes how the data are actually stored. The physical level describes complex low level data structure in detail. This level is expressed by physical schema which contains the definition of stored record, the method of representing the data fields and access aids used.

2. Logical Level:-

i) The next higher level of abstraction describes what data are stored in the database and what relationship exists among those data. The logical level thus describes the entire database in terms of a small number of relatively simple structures.

ii) It is defined by the logical schema. It describes all the records and relationships. There is one logical schema per database.

3. View Level:-

i) The highest level of data abstraction describes only part of the entire database. The system may provide many views for the same database.

ii) Each view is described by a schema called external schema. The external schema consists of the definition of logical records and relationships in the view level.

iii) The external schema also contains the method of deriving the objects in the external view from the objects in the conceptual view.

iv) It is the level closest to the users i.e. it deals with the way in which data is viewed by the users.

Q4. Explain Data Definition Languages. (2018(S)-7(C-iii))

Data Definition Language (DDL) statements are used to classify the database structure or schema. It is a type of language that allows the DBA or user to depict and name those entities, attributes, and relationships that are required for the application along with any associated integrity and security constraints. Here are the lists of tasks that come under DDL:

- CREATE - used to create objects in the database
- ALTER - used to alters the structure of the database
- DROP - used to delete objects from the database
- TRUNCATE - used to remove all records from a table, including all spaces allocated for the records are removed.
- RENAME - used to rename an object.

DDL is used to define the database. This definition includes all the entity sets and their associated attributes as well as the relationship among the entity sets. It also includes any constraints that have to be maintained.

❖ The DDL used at the external schema is called the view definition language (VDL) from where the defining process starts.

❖ A data dictionary contains metadata. A database system consults data dictionary before reading or modifying actual data. The schema of a table is an example of a metadata i.e. data about data.

❖ We specify the storage structure and access methods used by the database system by a set of statements in a special type of DDL called data storage and definition language (DSDL). These statements define the implementation details of the database schemas which are usually hidden from the users.

❖ The DDL provides facilities to specify such consistency constraints. The database system checks these constraints every time the database is updated.

Creating a Table

CREATE command can also be used to create tables. Now when we create a table, we have to specify the details of the columns of the tables too. We can specify the **names** and **data types** of various columns in the create command itself.

Following is the syntax,

```
CREATE TABLE <TABLE_NAME>
(
   column_name1 datatype1,
```

```
   column_name2 datatype2,
   column_name3 datatype3,
   .....................................
   Column_name4 datatypeN
);
```

ALTER Command: Add a new Column

Using ALTER command we can add a column to any existing table.

Syntax:

ALTER TABLE<table name>

ADD (Column name Data_type);

ALTER Command: Modify an existing Column

ALTER command can also be used to modify data type of any existing column.

Syntax:

ALTER TABLE <table name >

MODIFY (Column name Data_type);

ALTER Command: Rename a Column

Using ALTER command you can rename an existing column.

Syntax:

ALTER TABLE <table name>

RENAME <old_column_name> TO < new_column_name>;

ALTER Command: Drop a Column

ALTER command can also be used to drop or remove columns. Following is the syntax,

ALTER TABLE <table name>

DROP (column_name);

TRUNCATE command

TRUNCATE command removes all the records from a table. But this command will not destroy the table's structure.

Syntax:

TRUNCATE TABLE < table name>;

DROP command

DROP command completely removes a table from the database. This command will also destroy the table structure and the data stored in it.

Syntax:

DROP TABLE <table name>;

RENAME Table

RENAME command is used to set a new name for any existing table. Following is the syntax,

RENAME TABLE < old_table_name> to < new_table_name>;

Q5. What are the various database users? Describe in details (6-Mark 2019(w)-2(f))

There are five different types of database system users differentiated by the way they expect to interact with the system.

- **Naive Users:-**

❖ They are unsophisticated users who interact with the system by invoking one of the application programs that have been written previously. Example:- ATM user

❖ The typical user interface for a naive user is a forms interface where the user can fill in appropriate fields of the form. Naive users may also simply read reports generated from the database.

- **Application Programmers:-**

❖ Application programmers are computer professionals who write application programs.

❖ Application programmers can choose from many tools like RAD tools, programming languages, fourth generation languages etc. to develop user interfaces.

- **Sophisticated Users:-**

❖ Sophisticated users interact with the system without writing programs. Instead they form their requests in database query language. They submit each such query to query processor whose function is to break down DML statements into instructions that the storage manager understands.

❖ Analysts who submit queries to explore data in the database fall in this category.

- **Specialized Users:-**

❖ Specialized users are sophisticated users who write specialized database applications that do not fit into the traditional data processing framework.

❖ Among these applications are computer aided design systems, knowledge base& expert systems that store data with complex structure (Example:- Graphics and audio data) and environment modelling systems

- **Database Administrators (DBA):-**

A person who has central control over the entire database system is called database administrator (DBA).

❖ The functions of DBA include

i. Schema definition
ii. Storage structure and access method definition
iii. Schema and physical organization modification
iv. Granting of authorization for data access
v. Routine maintenance

Q7. Define data dictionary.(4-mark 2017(w)-7-(c-i))

A data dictionary is a file or a set of files that contains a database's metadata. The data dictionary contains records about other objects in the database, such as data ownership, data relationships to other objects, and other data.

The data dictionary is a crucial component of any relational database. Ironically, because of its importance, it is invisible to most database users. Typically, only database administrators interact with the data dictionary.

Metadata (Meta content) is defined as data providing information about one or more aspects of the data, such as:

・Means of creation of the data

・Purpose of the data

・Time and date of creation

・Creator or author of data

ADVANTAGES OF DATA DICTIONARY

- IT EXUDES CLARITY ON THE REST OF THE DATABASE DOCUMENTATION.
- WHEN A NEW USER IS INTRODUCED TO THE SYSTEM, IDENTIFYING TABLE STRUCTURE AND TYPES BECOMES SIMPLE.

DISADVANTAGES OF DATA DICTIONARY

- It needs careful planning, defining the exact requirements
- Designing its contents, testing, implementation and evaluation.

Q8. Distinguish between DROP and DELETE command. (2-Mark 2019-1(f))

Delete: The DELETE command is used to remove rows from a table. A WHERE clause can be used to only remove some rows. If no WHERE condition is specified, all rows will be removed. After performing a DELETE operation you need to COMMIT or ROLLBACK the transaction to make the change permanent or to undo it.

Drop: The DROP command removes a table from the database. All the tables' rows, indexes and privileges will also be removed. The operation cannot be rolled back.

Q9. Write down the syntax and give examples for the following

1. Alter
2. Create
3. Drop

ALTER Command: Add a new Column

Syntax:

ALTER TABLE<table name>

ADD (Column name Data_type);

Ex:

ALTER TABLE Persons

ADD DateOfBirth date;

ALTER Command: Modify an existing Column

Syntax:

ALTER TABLE <table name >

MODIFY Column name Data_type;

Ex:

ALTER TABLE Persons

MODIFY Roll varchar2 (5);

ALTER Command: Rename a Column

Syntax:

ALTER TABLE <table name>

RENAME <old_column_name> TO < new_column_name>;

Ex:

```
ALTER TABLE Persons
RENAME Roll to Rollno;
```

ALTER Command: Drop a Column

```
ALTER TABLE <table name>
DROP (column name);
Ex:
ALTER TABLE Persons
DROP DateOfBirth;
```

Creating a Table

```
Syntax:
CREATE TABLE <TABLE_NAME>
(
   column_name1 datatype1,
   column_name2 datatype2,
   column_name3 datatype3,
   .....................................
   Column_name4 datatypeN
);
Ex:
CREATE TABLE Persons (
   PersonID int,
   LastName varchar2 (255),
   FirstName varchar2 (255),
   Address varchar2(255),
   City varchar(255)
);
```

DROP command

```
Syntax:
DROP TABLE <table name>;
Ex:
DROP TABLE Shippers;
```

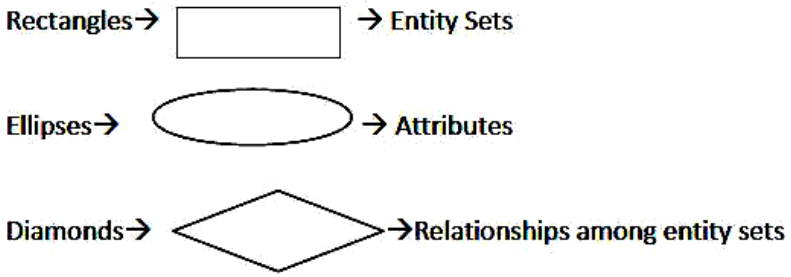
Chapter-2

Q1. Explain the E-R Model with suitable example. (5-Mark 2019(w)-2(e))

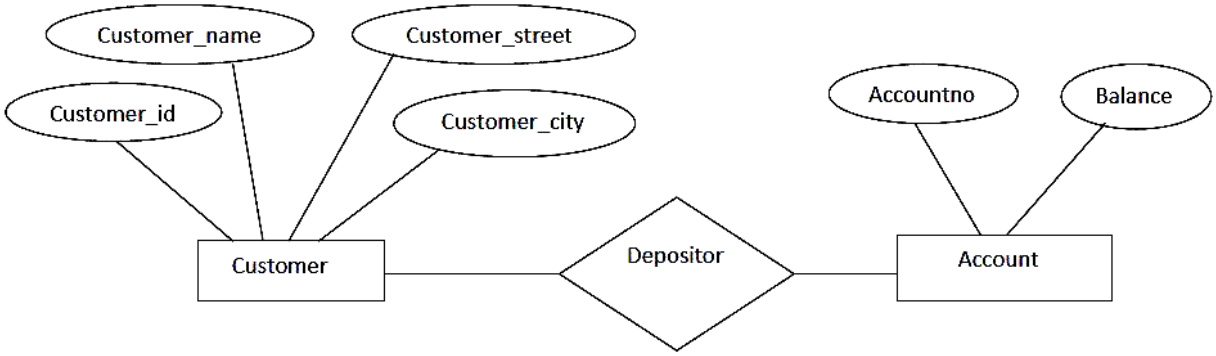
The E-R data model is based on a perception of a real world that consists of a collection of basic objects called entities and of relationships among these objects.

- An entity is a 'thing' or 'object' in the real world that is distinguishable from other objects. Example: - person, bank account etc.
- Entities are described in a database by a set of attributes. For example attributes like account number and balance describe the entity bank account. In some cases an extra attribute is used to define uniquely an entity set.
- A relationship is an association among several entities. For example a depositor relationship associates a customer with each account he has.

❖ The set of all entities of same type and the set of all relationships of the same type are termed as entity set and relationship set respectively.
❖ The overall logical structure (schema) of a database can be expressed graphically by an E-R diagram which is built up from the following components.



Lines→ ———— →Link attributes to entity sets and entity sets to relationships.
Each component is labeled with the entity or relationship that it represents.



(E-R Diagram)

❖ In addition to entities and relationships, the E-R model represents certain constraints to which the contents of a database must conform. One important constraint is ***mapping cardinalities***, which express the number of entities to which another entity can be associated via a relationship set. For example, if each account must belong to only one customer the E-R model can express that constraint.
❖ The E-R model is widely used in database design.

Q2. What are different types of data model available? Explain them.(10 marks-2019(W)-5)

Data model:
Data models define how the logical structure of a database is modelled. Data Models are fundamental entities to introduce abstraction in a DBMS. Data models define how data is connected to each other and how they are processed and stored inside the system.

The very first data model could be flat data-models, where all the data used are to be kept in the same plane. Earlier data models were not so scientific, hence they were prone to introduce lots of duplication and update anomalies.

RELATIONAL DATA MODEL:-

- This model has the advantage of being simple in principle; users can express their queries in a powerful query language.
- In this model the relation is the only construct required to represent the associations among the attributes of an entity as well as relationships among different entities.
- One of the main reasons for introducing this model was to increase the productivity of the application programmer by eliminating the need to change application programs when a change is made to the database. Users need not know the exact physical structures to use the database and are protected from any changes made to these structures. They are however still required to know how the data has been partitioned into various relations.

NETWORK DATA MODEL:-

- The network data model was formalized in the late 1960's by the Database Task Group of Conference on data system languages (DBTG/CODASYL). Hence it is also known as DBTG model.
- The network model uses two different data structures to represent the database entities and relationship between the entities named *record type* and *set type*. A record type is used to represent an entity type. It is made up of a number of data items that represents the attributes of an entity.

A set type is used to represent a directed relationship between two record types, the so Called owner record type and the member record type.

- The set type like the record type is named and specifies that there is a one to many relationship (1:M) between the owner and member record types. The set type can have more than one record type as its member, but only one record type is allowed to be the owner in a given set type.
- A database could have one or more occurrences of each of its record types and set types. An occurrence of a set type consists of an occurrence of the owner record type and any number of occurrences of each of its member record type. A record type can't be a member of two distinct occurrences of the same type.

HIERARCHICAL DATA MODEL:-

- A tree may be defined as a set of nodes such that there is one specially designated node called root node and the remaining nodes are partitioned into disjoint sets, each of which in turn is a tree, the sub trees of the root. If the relative order of the sub trees is significant the tree is an ordered tree.
- In a hierarchical database the data is organized in a hierarchical or ordered tree structure and the database is a collection of such disjoint trees (sometimes referred to as forests or spanning trees). The nodes of the tree represent record types. Each tree effectively represents a root record type and all its dependent record types. If we define the root record type at level 0, then the level of its dependent record types can be defined at level 1. The dependents of the record types at level 1 are said to be at level 2 and so on.
- An occurrence of a hierarchical tree type consists of one occurrence of the root record type along with zero or more occurrences of its dependent sub tree types. Each dependent sub tree is in turn, hierarchical and consists of a record type as its root node.

Q3.What is data independence. (2-Mark).

Data Independence:-

- Three levels of abstraction along with the mappings from internal to conceptual and from conceptual to external level provide two distinct levels of data independence: logical data independence and physical data independence.
- Logical data independence indicates that the conceptual schema can be changed without affecting the existing external schemas. The change would be absorbed by the mapping between the external and conceptual level.
- Logical data independence also insulates application programs from operations such as combining two records into one or splitting an existing record into two or more records.
- Logical data independence is achieved by providing the external level or user view of the database. The application programs or users see the database as described by their respective external views.
- Physical data independence is achieved by the presence of the internal level of the database and the mapping or transformation from conceptual level of database to internal level.

Q4. What are mapping Constraints? (2-Mark-2019(w)-1(h))

An E-R enterprise schema may define certain constraints to which the contents of the database must conform. Mapping cardinalities and participation constraints are two of the most important types of constraints.

i.) One to one:-

An entity in A is associated with at most one entity in B and an entity in B is associated with at most one entity set in A.

ii.) One to many:-

An entity set in A is associated with any number (zero or more) of entities in B. An entity in B however can be associated with at most one entity in A.

iii.) Many to one:-

An entity in A is associated with at most one entity in B. An entity in B however can be associated with any number (zero or more) of entities in A.

iv.) Many to many:-

An entity in A is associated with any number (zero or more) of entities in B and an entity in B is associated with any number (zero or more) of entities in A.

Q.5 What are Database attributes?(2-Mark)

ATTRIBUTES:-

- An entity is represented by a set of attributes. Attributes are descriptive properties possessed by each member of an entity set.
- Each entity has a value for each of its attributes. For example a particular customer entity may have the value 321-12 for customer id, value Anil for customer name etc.
- For each attribute there is a set of permitted values called domain or value set for that attribute.

Q6.Define Relationship set.(2-Mrak)

RELATIONSHIP SETS:-

- A relationship is an association among several entities.
- A relationship set is a set of relationships of the same type.
- The association between entity sets is referred to as participation that is the entity sets E1, E2, ……., En participate in a relationship set R.

❖ A relationship instance in an E-R schema represents an association between the named entities of the real world enterprise that is being modeled.
❖ A relationship may also have attributes called ***descriptive attributes***. Consider a relationship set depositor with entity sets customer and account. We could associate the attribute access date to that relationship to specify the most recent date on which the customer accessed an account.
❖ A relationship instance in a given relationship set must be uniquely identifiable from its participating entities, without using descriptive attributes. For example instead of using a single access date we use access dates.

Q7.Explain difference between hierarchical, relational and network data model. (10-Mrak)

Comparison	Hierarchical	Network	Relational
Description	Hierarchical orientation & navigation; hierarchies of related records & standard interfaces	Network orientation & navigation; uses hierarchically-arranged data with the exception that child tables can have more than one parent; standard interfaces	Relational orientation; data retrieved by unique keys; relationships expressed through matching keys; physical organization of data managed by RDBMS
Physical Structure	Tree; parent-child relationships; single table acts as the "root" of the database from which other tables "branch" out; child can only have one parent, but a parent can have multiple children	Network of interrelated lists; looks like several trees that share branches; children have multiple parents and parents have multiple children	Data is stored in relations (tables); relationships maintained by placing a key field value of one record as an attribute in the related record
Structural Changes	Inflexible (once data is organized in a particular way, difficult to change); data reorganization complicated; requires careful design	Inflexible (once data is organized in a particular way, difficult to change); data reorganization complicated; requires careful design	Flexible; because tables are subject-specific & key fields relate one entity to another, both the data & the database structure can be easily modified & manipulated; programs independent of data format which yields flexibility when modifications are needed

Comparison	Hierarchical	Network	Relational
Relationships	Linked lists using pointers stored in the parent/child records to navigate through the records; pointers could be a disk address, the key field, or other random access technique; start at root and work down the tree to reach target data; supports one-to-one & one-to-many relationships	Uses series of linked lists to implement relationships between records; each list has an owner record & possibly many member records; a single record can either be the owner or a member of several lists of various types; supports one-to- one, one-to-many, & many-to-many relationships	Uses key fields to link data in many different ways; supports one-to-one, one-to-many & many-to-many relationships

Q8. What is data independence? Explain it types.(5-Mark-2019(w)-2(a))

Data Independence:-

- Three levels of abstraction along with the mappings from internal to conceptual and from conceptual to external level provide two distinct levels of data independence: logical data independence and physical data independence.
- Logical data independence indicates that the conceptual schema can be changed without affecting the existing external schemas. The change would be absorbed by the mapping between the external and conceptual level.
- Logical data independence also insulates application programs from operations such as combining two records into one or splitting an existing record into two or more records.
- Logical data independence is achieved by providing the external level or user view of the database. The application programs or users see the database as described by their respective external views.
- Physical data independence is achieved by the presence of the internal level of the database and the mapping or transformation from conceptual level of database to internal level.
- The physical data independence criterion requires that the conceptual level does not specify storage structures or the access methods (indexing, hashing etc) used to retrieve the data from the physical storage medium.
- Another aspect of data independence allows different interpretation of the same data. The storage of data is in bits and may change from EBCDIC to ASCII coding.

Logical Data Independence

Logical data is data about database, that is, it stores information about how data is managed inside. For example, a table (relation) stored in the database and all its constraints, applied on that relation. Logical data independence is a kind of mechanism, which liberalizes itself from actual data stored on the disk. If we do some changes on table format, it should not change the data residing on the disk.

Physical Data Independence

All the schemas are logical, and the actual data is stored in bit format on the disk. Physical data independence is the power to change the physical data without impacting the schema or logical data.

For example, in case we want to change or upgrade the storage system itself − suppose we want to replace hard-disks with SSD − it should not have any impact on the logical data or schemas.

Chapter-3

Q1.What is join? Explain different types of join? (10-amrk-2019(w)-7)

Join is a combination of a Cartesian product followed by a selection process. A Join operation pairs two tuples from different relations, if and only if a given join condition is satisfied.

We can classify joins basically into two types

1. **INNER JOINS: These joins are the one that has the tuples that satisfy some conditions and rest are discarded . Further they are classified as**
- **Theta join**
- **Equi join**
- **Natural join**

2. **OUTER JOINS: These have all the tuples from either or both the relations. Further they are classified as**
- **Left outer join**
- **Right outer join**
- **Full outer join**

Let us now move on to the study the classified types with examples in detail.

INNER JOINS

1. **Theta join(θ) – They have tuples from different relations if and only if they satisfy the theta condition, here the comparison operators (≤, ≥, <, >, =, ¯)come into picture. Let us consider simple example to understand in a much better way, suppose we want to buy a mobile and a laptop, based on our budget we have thought of buying both such that mobile price should be less than that of laptop.**

MOBILE

MODEL	PRICE
Asus	10k
Samsung	20k
Iphone	50k

LAPTOP

MODEL	PRICE
Acer	20k
HP	35k
Apple	80k

AFTER JOINS

MODEL	PRICE
Asus	Acer
Asus	HP
Asus	Apple
Samsung	HP
Samsung	Apple
Iphone	Apple

2. Equi join – As the name itself indicates, if suppose the join uses only the equality operator then it is called as equi join.

3. Natural join – It does not utilize any of the comparison operator. Here the condition is that the attributes should have same name and domain. There has to be at least one common attribute between between two relations. It forms the cartesian product of two arguments, performs selection forming equality on those attributes that appear in both relations and eliminates the duplicate attributes.

EMPLOYMENT

NAME	EMPID	DPT_NAME
A	11	Sales
B	12	Finance
C	13	Finance

DEPARTMENT

DPT_NAME	MANAGER
Finance	M1
Sales	M2

Name	EMPID	DPT_NAME	MANAGER
A	11	Sales	M2
B	12	Finance	M1
C	13	Finance	M1

Outer join

1.Left outer join – All the tuples of left table is displayed irrespective of whether it satisfies the matching conditions. Thus in the left all the tuples have been displayed but in the right only those are present that satisfy the matching conditions.

COUNTRY

COUNTRY_ID	COUNT_NME
1	India
2	Pakistan
4	Nepal

STATE

STATE_ID	COUNTRY_ID	STATE_NME
1	1	Karnataka
2	1	Tamil Nadu
3	2	Islamabad
4	NULL	Bangladesh

So for left join, left side table have all the values but right side only has those whose COUNTRY_ID has been matched.

COUNTRY_ID	COUNT_NME	STATE_ID	COUNTY_ID	STATE_NME
1	India	1	1	Karnataka
1	India	2	1	Tamil nadu
2	Pakistan	3	2	Islamabad
4	Nepal	NULL	NULL	NULL

LEFT OUTER JOIN

Bangladesh has not occurred since there is no match found.

2. Right outer join – All the tuples of right table are displayed irrespective of whether it satisfies the matching conditions or not.. Thus in the right, all the tuples have been displayed but in the left only those are present that satisfy the matching conditions. The previous example can be implemented here as well.

RIGHT OUTER JOIN

COUNTRY_ID	COUNT_NME	STATE_ID	COUNTY_ID	STATE_NME
1	India	1	1	Karnataka
1	India	2	1	Tamil nadu
2	Pakistan	3	2	Islamabad
NULL	NULL	4	NULL	Bangladesh

3. Full outer join– Tuples from both the relations takes part irrespective of whether it has the matching or non-matching conditions. Example is as shown

FULL OUTER JOIN

COUNTRY_ID	COUNT_NME	STATE_ID	COUNTY_ID	STATE_NME
1	India	1	1	Karnataka
1	India	2	1	Tamil nadu
2	Pakistan	3	2	Islamabad
4	Nepal	NULL	NULL	NULL
NULL	NULL	4	NULL	Bangladesh

Q2. What are different relational operators used in Relational algebra? (10-Mark)

RELATIONAL ALGEBRA:-

- The relational algebra is a procedural query language. It consists of a set of operations that take one or two relations as its operands and produces a new relation as its result.
- The fundamental operations in the relational algebra are select, project, union, set difference, rename and Cartesian product.

❖ In addition to the fundamental operations there are several other operations namely set intersection, natural join, division and assignment.

3.2 Fundamental Operations:-

The select, project and rename operations are called unary operations because they operate on one relation. The other three operations union, set difference and Cartesian product operate on pairs of relations and therefore called binary relations.

Select Operation:-

The select operation selects tuples that satisfy a given predicate. We use the lower case Greek letter sigma (σ) to denote selection. The predicate appears as a subscript to σ. Thus to select those tuples of the loan relationship where the branch is "Berhampur", we write

σ branch_name="Berhampur" (loan)

- In general we allow comparisons using =,≠,≤,<,≥,> in the selection predicate,
- Further we can combine several predicates into a larger predicate by using the connectors ^ (AND) , ˅ (OR) and ~ (NOT). For example

σ branch_name="Berhampur"^ amount>1200

- The selection predicate may include comparisons between two attributes.

Project Operation:-

- The project operation is a unary operation that returns its argument relation, with certain attributes left out. Since a relationship is a set, any duplicate rows are eliminated.
- Projection is denoted by upper case Greek letter Π (pi). Suppose we want to list all loan numbers and the amounts of loans but do not care about the branch name.

Π loan_number, amount (loan)

Composition of relational operations:-

For example we want to find those customers who live in Berhampur?

Π customer_name(σ customer_city="Berhampur"(customer))

Here instead of giving the name of a relation as the argument of the projection operation, we give an expression that evaluates to a relation.

- In general since the result of a relational algebra operation is of the type relation as its inputs, relational algebra operations can be composed together into a relational algebra expression.

- Composing relational algebra operations into relational algebra expressions is just like composing arithmetic operations i.e. Inner parentheses will be evaluated first and then the outer parentheses and so on.

Union operation:-

- Consider a query to find the names of all bank customers who have either an account or a loan account or both.

The customer relation does not contain information about loan and the loan relation does not contain information about customer. So to find all the customer of a particular bank we have to join both relations by the following projection operation.

Π customer_name(depositor) ∪ Π borrower_name(loan)

- To have union operation we require two conditions.

1. The relations R and S must be of same arity i.e. they must have same number of attributes.

2. The domains of the ith attribute of R and the ith attribute of S must be same for all i. Set Difference operation:-

- The set difference operation is denoted by "–"allows us to find tuples that are in one relation but not are in the other one. The expression R-S produces a relation containing those tuples that are in R but not in S
- We can find all customers of the bank who have an account but not a loan.

Π customer_name(deositor)-Π customer_name(borrower)

We must ensure that set difference are taken between compatible relations. Therefore a set difference operation R-S to be valid we require that the relations R and S be of same arity and the domains of ith attribute of R and the ith attribute of S be same for all i.

Cartesian product operation:-

- The Cartesian product operation denoted by "X" allows us to combine information from any two relations. We write the Cartesian product of relations R1 and R2 as R1 X R2
- A relation is by definition a subset of a Cartesian product of a set of domains.
- However since the same attribute name may appear in both R1 and R2 we need to devise a naming schema to distinguish between these attributes.
- We do it by attaching to an attribute the name of the relation from which the attribute originally came. For example the relational schema R= borrower X depositor is

(depositor.name, depositor.accno, borrower.name, borrower.loan no, borrower.accno)

Set intersection operation:-

- We want to find out the customers who have both a loan and an account. Using set intersection operation we write

Π depositor_name(depositor)∩ Π borrower_name(borrower)

- We can rewrite any relational algebra expression that uses set intersection operation with a pair of set difference operation.

R∩S=R-(R-S)

Natural Join operation:-

- The natural join is a simple operation that allows us to combine certain selections and a Cartesian product into one operation. It is denoted by the join symbol .The natural join operation forms a Cartesian product of its two arguments, performs a selection forcing equality on those attributes that appear in both relation schemas and finally removes duplicate attributes.

Chapter-4

Q1. What do you mean by lossless join? (2-Mark 2019-2(i))

LOSS LESS DECOMPOSITION:-

- When we decompose a relation into a number of smaller relations, it is crucial that the decomposition be lossless. We must first present criteria for determining whether a composition is lossy.
- Let R be relation schema and F be a set of functional dependencies on R. Let R1 and R2 form a decomposition of R. This composition is a loss less join decomposition of R if at least one of the following functional dependencies is in F_+:

(i) R1 ∩R2 → R1
(ii) R1 ∩R2 → R2

- In other words if R1 ∩ R2 forms a super key of either R1 or R2 the decomposition of R is a loss less decomposition.
- For the general case of decomposition of a relation into multiple parts at once the test for lossless join decomposition is more complicated.
- While the test for binary decomposition is clearly a sufficient condition for loss less join, it is a necessary condition only if all constraints are functional dependencies.

Q2.Explain with suitable example what is functional dependency.(5-Mark 2017-6(b))(2019(w)-1(D).
Or-Define FD in a Relation. (2 marks-2019(w)-1(D).

A *functional dependency* (FD) is a relationship between two attributes, typically between the PK and other non-key attributes within a table. For any relation R, attribute Y is functionally dependent on attribute X (usually the PK), if for every valid instance of X, that value of X uniquely determines the value of Y. This relationship is indicated by the representation below:

X ————> Y

The left side of the above FD diagram is called the *determinant*, and the right side is the *dependent*. Here are a few examples.

In the first example, below, SIN determines Name, Address and Birthdate. Given SIN, we can determine any of the other attributes within the table.

SIN ————-> Name, Address, Birthdate

For the second example, SIN and Course determine the date completed (DateCompleted). This must also work for a composite PK.

SIN, Course ————> DateCompleted

The third example indicates that ISBN determines Title.

ISBN ————> Title

Rules of Functional Dependencies

Consider the following table of data r(R) of the relation schema R(ABCDE) shown in Table 11.1.

A	B	C	D	E
a1	b1	c1	d1	e1
a2	b1	C2	d2	e1
a3	b2	C1	d1	e1
a4	b2	C2	d2	e1
a5	b3	C3	d1	e1

Table R

Table 11.1. Functional dependency example, by A. Watt.

As you look at this table, ask yourself: *What kind of dependencies can we observe among the attributes in Table R?* Since the values of A are unique (a1, a2, a3, etc.), it follows from the FD definition that:

$A \rightarrow B, \quad A \rightarrow C, \quad A \rightarrow D, \quad A \rightarrow E$

- It also follows that $A \rightarrow BC$ (or any other subset of ABCDE).
- This can be summarized as $A \rightarrow BCDE$.
- From our understanding of primary keys, A is a primary key.

Since the values of E are always the same (all e1), it follows that:

$A \rightarrow E, \quad B \rightarrow E, \quad C \rightarrow E, \quad D \rightarrow E$

However, we cannot generally summarize the above with $ABCD \rightarrow E$ because, in general, $A \rightarrow E, \quad B \rightarrow E, \quad AB \rightarrow E$.

Other observations:

1. Combinations of BC are unique, therefore $BC \rightarrow ADE$.
2. Combinations of BD are unique, therefore $BD \rightarrow ACE$.
3. If C values match, so do D values.

a. Therefore, C → D

b. However, D values don't determine C values

c. So C does not determine D, and D does not determine C.

Looking at actual data can help clarify which attributes are dependent and which are determinants.

Q3. What do you mean by normalization? Explain different normal forms?
Q4. Describe the 1st, 2nd, 3rd NF, BCNF with example. (10-marks 2019(w)-4)

NORMALIZATION

If a database design is not perfect it may contain anomalies, which are like a bad dream for database itself. Managing a database with anomalies is next to impossible.

- **Update anomalies:** if data items are scattered and are not linked to each other properly, then there may be instances when we try to update one data item that has copies of it scattered at several places, few instances of it get updated properly while few are left with there old values. This leaves database in an inconsistent state.
- **Deletion anomalies:** we tried to delete a record, but parts of it left undeleted because of unawareness, the data is also saved somewhere else.
- **Insert anomalies:** we tried to insert data in a record that does not exist at all.

Normalization is a method to remove all these anomalies and bring database to consistent state and free from any kinds of anomalies.

- The first of the normal forms that we study, first normal form imposes a very basic requirement on relations; unlike the other normal forms, it does not require additional information such as functional dependency.
- A domain is atomic if elements of the domain are considered to be individual units. We say that a relation schema R is in first normal form (1NF) if the domains of all the attributes of R are atomic.
- In other words only one value is associated with each attribute and the value is not set of values or list of values.
- A database schema is in first normal form if every relation schema included in the database schema is in 1NF.
- The first normal form pertains to the tabular format of the relation.
- Sometimes non atomic values can be useful. For example composite valued attributes and set valued attributes. In many domains where entities have a complex structure, forcing a 1NF representation represents an unnecessary burden on the application programmer who has to write code to convert data into atomic form.

For example consider a table which is not in first normal form.

Student	Age	Subject
Adam	17	Biology, Math
Alex	14	Math
Stuart	15	Math

In First Normal Form, any row must not have a column in which more than one value is saved, like separated with commas. Rather than that, we must separate such data into multiple rows.
Student Table following 1NF will be

Student	Age	Subject
Adam	17	Biology
Adam	17	Math
Alex	14	Math
Stuart	15	Math

SECOND NORMAL FORM (2NF):-

- A relation scheme R<S, F> is in 2NF if it is in 1NF and if all non prime attributes are fully functional dependent on the relation keys.
- A database schema is in 2NF if every relation schema included in the database schema is in 2NF
- A 2NF does not permit partial dependency between a non prime attribute and the relation keys.
- Even though 2NF does not permit partial dependency between a non prime attribute and the relation keys it does not rule out the possibility that a non prime attribute may also be functionally dependent on another non prime attribute. This type of dependency between non prime attributes also causes anomalies. In example of First Normal Form there are two rows for Adam, to include multiple subjects that he has opted for. While this is searchable, and follows First normal form, it is an inefficient use of space. Also in the above Table in First Normal Form, while the candidate key is {**Student**, **Subject**}, **Age** of Student only depends on Student column, which is incorrect as per Second Normal Form. To achieve second normal form, it would be helpful to split out the subjects into an independent table, and match them up using the student names as foreign keys.

New Student Table following 2NF will be :

Student	Age
Adam	17
Alex	14
Stuart	15

In Student Table the candidate key will be **Student** column, because all other column i.e **Age** is dependent on it.
New Subject Table introduced for 2NF will be:

Student	Subject
Adam	Biology
Adam	Math
Alex	Math
Stuart	Math

In Subject Table the candidate key will be {**Student**, **Subject**} column. Now, both the above tables qualify for Second Normal Form and will never suffer from Update Anomalies. Although there are a few complex cases in which table in Second Normal Form suffers Update Anomalies, and to handle those scenarios Third Normal Form is there

THIRD NORMAL FORM (3NF):-

❖ BCNF requires that all non trivial dependencies of the form $\alpha \rightarrow \beta$, where α is a super key. Third normal form (3NF) relaxes this constraint slightly by allowing certain non trivial functional dependencies whose left side is not a super key.

❖ A relation schema R is in third normal form with respect to a set F of functional dependencies if, for all functional dependencies in F^+ of the form $\alpha \rightarrow \beta$, where α C R and β C R, at least one of the following holds:

1. $\alpha \rightarrow \beta$ is a trivial functional dependency
2. α is a super key for R
3. Each attribute A in $\beta - \alpha$ is contained in a candidate key for R

❖ Note that the third condition above does not say that a single candidate key must contain all the attributes in $\beta - \alpha$; each attribute A in $\beta - \alpha$ may be contained in a different candidate key.

❖ **Third Normal form** applies that every non-prime attribute of table must be dependent on primary key. The *transitive functional dependency* should be removed from the table. The table must be in **Second Normal form**.

For example, consider a table with following fields.

Student_Detail Table :

Student_id	Student_name	DOB	Street	City	State	PIN

In this table Student_id is Primary key, but street, city and state depends upon PIN. The dependency between PIN and other fields is called **transitive dependency**. Hence to apply **3NF**, we need to move the street, city and state to new table, with **PIN** as primary key.

New Student_Detail Table :

Student_id	Student_name	DOB	PIN

Address Table:

PIN	Street	City	State

The advantage of removing transitive dependency is:-

- ❖ Amount of data duplication is reduced.
- ❖ Data integrity achieved.

BOYCE- CODD NORMAL FORM (BCNF):-

- ❖ Boyce- Codd normal form eliminates all redundancy that can be discovered based on functional dependencies.
- ❖ A relational schema R is in BCNF with respect to a set F of functional dependencies in F_+ of the form $\alpha \rightarrow \beta$, where $\alpha C R$ and $\beta C R$, at least one of the following holds:

1. $\alpha \rightarrow \beta$ is a trivial functional dependency (that is $\beta C \alpha$)
2. α is a super key for schema R

- ❖ A database design is in BCNF if each member of the set of relation schemas that constitute the design is in BCNF.
- ❖ When we decompose a schema that is not in BCNF, it may be that one or more of the resulting schemas are not in BCNF. In such cases further decomposition is required, the eventful result of which is a set of BCNF schemas.

- ❖ The BCNF imposes a stronger constraint on the types of functional dependencies allowed in a relation. The only non trivial functional dependencies allowed in a relation. The only non trivial functional dependencies allowed in the BCNF are those functional dependencies whose determinants are candidate super keys of the relation.
- ❖ Any schema that satisfies BCNF also satisfies 3NF, since each of its functional dependencies would satisfy one of the first two alternatives. BCNF is therefore a more restrictive normal form than is 3NF.
- ❖ A 3NF table which does not have multiple overlapping candidate keys is said to be in BCNF.

In the above normalized tables in 3NF, Student_id is super-key in Student_Detail relation and PIN is super-key in Address relation. So,
Student_id → Student_name, DOB, PIN
And
PIN → Street, City, State
confirms, that both relations are in BCNF.

Q4) Define Primary key. How it is different from candidate key. (2019(w)-2 marks-1(e))

A primary key is a special relational database table column (or combination of columns) designated to uniquely identify all table records.

A super key with no redundant attribute is known as candidate key. Candidate keys are selected from the set of super keys, the only thing we take care while selecting candidate key is that the candidate key should not have any redundant attributes. That's the reason they are also termed as minimal super key.

Chapter-5

Q1) Explain various types of database language used in SQL. Write Syntax and example queries for I) CREATE 2) INSERT 3) UPDATE (2019(w)-10marks-6)

Types of Language:

1. Data Definition Language
2. Data Manipulation Language

3. Data Query language
4. Data Control Language
5. Transaction Control Language

1.Data Definition Language:

CREATE: Create new database, table, etc.
ALTER: Alter existing database, table, etc.
DROP: Drop the database
RENAME: Set a new name for the table.

2.Data Manipulation Language:

INSERT: Insert data
UPDATE: Update data
DELETE: Delete all records

3.Data Query language

SELECT: Retrieve data from the database

4.Data Control Language:

GRANT: Give privilege to access the database.
REVOKE: Take back the privilege to access the database.

5.Transaction Control Language:

COMMIT: Save the work.
SAVEPOINT: Set a point in transaction to rollback later
ROLLBACK: Restores since last commit

Command to Create a Table:-

Syntax:-

```
create table tablename
 ( colname datatype(size),
colname datatype (size)
……,
 colname datatype (size)
);
```

Example:-

```
Create table student
(rollno varchar2(20),
 name varchar2(30), address varchar2(50),
semester varchar2(10)
);
```

Command to insert data into a table:-

While inserting a single row of data into a table, the insert operation first creates a new row (empty in the database table) and then loads the value passed into the column specified.

Syntax:-

```
Insert into table name (col1, col2, col3,……., coln)
 Values (expr1, expr2, expr3,……., exprn);
```

Example:

```
 Insert into student
Values ('f1201207005','radhika','bbsr','3rd cse');
```

Insert into student
Values ('f1201207002','rupak','ctc','5th cse');

Insert into student
Values ('f1201207003','jyoti','rkl','5th it');

Insert into student
Values ('f1201207008','ajay','bam','3rd it');

Insert into student
Values ('f1201207001','manoj','khd','5th cse');

Updating the contents of a table:-

The update command is used to change or modify data value of a table.
Syntax:-
Sql> update table name set col1=<expression>, col2=<expression>;
Example:-
Update student set address='bam';

To update records conditionally we can use the following syntax.
Syntax:-
Sql> update table name set col. name=<expression> where <condition>;
Example:-
Update student set name='bidya' where rollno='f1201207001';

Q2 short notes on DML. (4-Mark)

Command to Create a Table:-

Syntax:-
create table tablename
(colname datatype(size),
colname datatype (size)
……,
colname datatype (size)
);
Example:-

Create table student
(rollno varchar2(20),
name varchar2(30), address varchar2(50),
semester varchar2(10)
);

Command to insert data into a table:-

While inserting a single row of data into a table, the insert operation first creates a new row (empty in the database table) and then loads the value passed into the column specified.

Syntax:-
Insert into table name (col1, col2, col3,……., coln)
Values (expr1, expr2, expr3,……., exprn);
Example:
Insert into student
Values ('f1201207005','radhika','bbsr','3rd cse');

Insert into student
Values ('f1201207002','rupak','ctc','5th cse');

Insert into student
Values ('f1201207003','jyoti','rkl','5th it');

Insert into student
Values ('f1201207008','ajay','bam','3rd it');

Insert into student
Values ('f1201207001','manoj','khd','5th cse');

Updating the contents of a table:-

The update command is used to change or modify data value of a table.
Syntax:-
Sql> update table name set col1=<expression>, col2=<expression>;
Example:-
Update student set address='bam';

To update records conditionally we can use the following syntax.
Syntax:-
Sql> update table name set col. name=<expression> where <condition>;
Example:-
Update student set name='bidya' where rollno='f1201207001';

Deleting data from a table:-

- To remove all the rows from a table the syntax is as follows

Syntax:-
Sql> delete from table name;
Example:-
delete from student;

- To remove a set of rows from a table the syntax is as follows

Syntax:-
Sql> delete from table name where <condition>;
Example:-
delete from student where name='radhika';

Chapter-6

Q1. Define Transaction. Explain various properties of transaction. (5-Mark 2019(w)-2(c))

Or- List out all the states of a transaction. (2-mark-2019(w)-1(b))

TRANSACTION PROCESSING:-

- A transaction is a program unit whose execution may change the contents of a database
- If the database was in a consistent state before a transaction, then on completion of the transaction the database will be in a consistent state.
- This requires that the transaction be considered atomic, it is executed successfully or in case of errors the user can view the transaction as not having been executed at all.

Properties of transaction:-

The database system must maintain the following properties of transactions to show all the characteristics. The properties referred to ACID (atomicity, consistency, isolation and durability) test represent the transaction paradigm.

Atomicity:-

The atomicity property of a transaction implies that it will run to completion as an individual unit, at the end of which either no changes have occurred to the database or the database has been changed in a consistent manner. At the end the updates made by the transaction will be accessible to other transactions and processes.

Consistency:-

The consistency property of a transaction implies that if the database was in a consistent state before the start of a transaction, then on termination of a transaction the database will also be in a consistent state.

Isolation:-

The isolation property of a transaction indicates that actions performed by a transaction will be isolated or hidden from outside the transaction until the transaction terminates. This property gives the transactions a measure of relative independence.

Durability:-

The durability property of a transaction ensures that the commit action of a transaction, on its termination, will be reflected in the database. The permanence of the commit action of a transaction requires that any failures after the commit operation will not cause loss of updates made by the transaction.

Chapter-7

Q1. Define concurrency control. Explain the concept like Lock and dead lock in Database. (5-Mark 2017-3(b))

Concurrency Control concepts:-

If all schedules in a concurrent environment are restricted to serializable schedules, the result obtained will be consistent with some serial execution of transactions and will be considered correct. However using only serial schedules unnecessarily limits the degree of concurrency. Furthermore, testing for serializability of a schedule is not only computationally expensive but it is an

after the fact technique and impractical. Thus concurrency control scheme is applied in a concurrent database environment to ensure that the schedules produced by concurrent transactions are serializable. The most frequently used schemes are locking and time stamp based order.

LIVE LOCK:-

- A **live lock** is similar to a deadlock, except that the states of the processes involved in the live lock constantly change with regard to one another, none progressing. This term was defined formally at some time during the 1970s – in Babich's 1979 article on program correctness. Live lock is a special case of resource starvation; the general definition only states that a specific process is not progressing.
- A real-world example of live lock occurs when two people meet in a narrow corridor, and each tries to be polite by moving aside to let the other pass, but they end up swaying from side to side without making any progress because they both repeatedly move the same way at the same time.
- Live lock is a risk with some algorithms that detect and recover from deadlock. If more than one process takes action, the deadlock detection algorithm can be repeatedly triggered. This can be avoided by ensuring that only one process (chosen arbitrarily or by priority) takes action.

DEAD LOCK:-

- A **deadlock** is a situation in which two or more competing actions are each waiting for the other to finish, and thus neither ever does.
- In a transactional database, a deadlock happens when two processes each within its own transaction updates two rows of information but in the opposite order. For example, process A updates row 1 then row 2 in the exact timeframe process B updates row 2 then row 1. Process A can't finish updating row 2 until process B is finished, but it cannot finish updating row 1 until process A finishes. No matter how much time is allowed to pass, this situation will never resolve itself and because of this database management systems will typically kill the transaction of the process that has done the least amount of work.
- Deadlock is a common problem in multiprocessing systems, parallel computing and distributed systems, where software and hardware locks are used to handle shared resources and implement process synchronization.
- In telecommunication systems, deadlocks occur mainly due to lost or corrupt signals instead of resource contention.
- Example:-

A simple computer-based example is as follows. Suppose a computer has three CD drives and three processes. Each of the three processes holds one of the drives. If each process now requests another drive, the three processes will be in a deadlock. Each process will be waiting for the "CD drive released" event, which can be only caused by one of the other waiting processes. Thus, it results in a circular chain.

Q2. Explain how Serializability can be done in concurrency control. (5-Mark 2017-7(b))

SERIALIZABILITY:

- **Serializability** is a property of a transaction schedule (history). It relates to the *isolation* property of a database transaction.

❖ **Serializability** of a schedule means equivalence to a *serial schedule* (i.e., sequential with no transaction overlap in time) with the same transactions. It is the major criterion for the correctness of concurrent transactions' schedule, and thus supported in all general purpose database systems.

❖ **The rationale behind serializability** is the following:

If each transaction is correct by itself i.e. meets certain integrity conditions, then a schedule that comprises any *serial* execution of these transactions is correct (its transactions still meet their conditions): "Serial" means that transactions do not overlap in time and cannot interfere with each other, i.e. complete *isolation* between each other exists. Any order of the transactions is legitimate, if no dependencies among them exist, which is assumed. As a result, a schedule that comprises any execution (not necessarily serial) that is equivalent to any serial execution of these transactions is correct.

❖ Schedules that are not serializable are likely to generate erroneous outcomes. Examples are with transactions that debit and credit accounts with money: If the related schedules are not serializable, then the total sum of money may not be preserved. Money could disappear, or be generated from nowhere. It does not happen if serializability is maintained.

❖ If any specific order between some transactions is requested by an application, then it is enforced independently of the underlying serializability mechanisms. These mechanisms are typically indifferent to any specific order, and generate some unpredictable partial order that is typically compatible with multiple serial orders of these transactions.

❖ Two major types of serializability exist: *view-serializability*, and *conflict-serializability*. View-serializability matches the general definition of serializability given above. Conflict-serializability is a broad special case, i.e., any schedule that is conflict-serializable is also view-serializable, but not necessarily the opposite. Conflict-serializability is widely utilized because it is easier to determine and covers a substantial portion of the view-serializable schedules.

❖ **View-serializability** of a schedule is defined by equivalence to a serial schedule (no overlapping transactions) with the same transactions, such that respective transactions in the two schedules read and write the same data values ("view" the same data values).

❖ **Conflict-C**is defined by equivalence to a serial schedule (no overlapping transactions) with the same transactions, such that both schedules have the same sets of respective chronologically ordered pairs of conflicting operations (same precedence relations of respective conflicting operations).

Q3. What do you mean by serializability and recoverability of a schedule? (2019(W)-2(d))

Serializability is a property of a transaction schedule (history). It relates to the *isolation* property of a database transaction.

❖ **Serializability** of a schedule means equivalence to a *serial schedule* (i.e., sequential with no transaction overlap in time) with the same transactions. It is the major criterion for the correctness of concurrent transactions' schedule, and thus supported in all general purpose database systems.

❖ **The rationale behind serializability** is the following:

If each transaction is correct by itself i.e. meets certain integrity conditions, then a schedule that comprises any *serial* execution of these transactions is correct (its transactions still meet their conditions): "Serial" means that transactions do not overlap in time and cannot interfere with each other, i.e. complete *isolation* between each other exists. Any order of the transactions is legitimate, if no dependencies among them exist, which is assumed. As a result, a schedule that comprises any execution (not necessarily serial) that is equivalent to any serial execution of these transactions is correct.

RECOVERABILITY:-

Crash Recovery

Though we are living in highly technologically advanced era where hundreds of satellite monitor the earth and at every second billions of people are connected through information technology, failure is expected but not every time acceptable.
DBMS is highly complex system with hundreds of transactions being executed every second. Availability of DBMS depends on its complex architecture and underlying hardware or system software. If it fails or crashes amid transactions being executed, it is expected that the system would follow some sort of algorithm or techniques to recover from crashes or failures.

Q5. What is lock? List the type of Lock.(2019(w)-2(g))

Lock:-

A lock is a data variable which is associated with a data item. This lock signifies that operations that can be performed on the data item. Locks help synchronize access to the database items by concurrent transactions.

DEAD LOCK:-

- A **deadlock** is a situation in which two or more competing actions are each waiting for the other to finish, and thus neither ever does.
- In a transactional database, a deadlock happens when two processes each within its own transaction updates two rows of information but in the opposite order. For example, process A updates row 1 then row 2 in the exact timeframe process B updates row 2 then row 1. Process A can't finish updating row 2 until process B is finished, but it cannot finish updating row 1 until process A finishes. No matter how much time is allowed to pass, this situation will never resolve itself and because of this database management systems will typically kill the transaction of the process that has done the least amount of work.
- Deadlock is a common problem in multiprocessing systems, parallel computing and distributed systems, where software and hardware locks are used to handle shared resources and implement process synchronization.
- In telecommunication systems, deadlocks occur mainly due to lost or corrupt signals instead of resource contention.
- Example:-

A simple computer-based example is as follows. Suppose a computer has three CD drives and three processes. Each of the three processes holds one of the drives. If each process now requests another drive, the three processes will be in a deadlock. Each process will be waiting for the "CD drive released" event, which can be only caused by one of the other waiting processes. Thus, it results in a circular chain.

Types of Locks

Several types of locks are used in concurrency control. To introduce locking concepts gradually, we first discuss binary locks, which are simple but restrictive and so are not used in practice. We then discuss shared/exclusive locks, which provide more general locking capabilities and are used in practical database locking schemes.

Binary Locks

A binary lock can have two states or values: locked and unlocked.

A distinct lock is associated with each database item *A.* If the value of the lock on *A* is 1, item *A* cannot be accessed by a database operation that requests the item. If the value of the lock on *A* is 0 then item can be accessed when requested. We refer to the current value of the lock associated with item *A* as *LOCK (A).* There are two operations, lock item and unlock item are used with binary locking A transaction requests access to an item *A* by first issuing a lock *item (A)* operation. If LOCK (A) = 1, the transaction is forced to wait. If LOCK (A) = 0 it is set to 1 (the transaction locks the item) and the transaction is allowed to access item *A.* When the transaction is through using the item, it issues an unlock *item (A)* operation, which sets *LOCK (A)* to 0 (unlocks the item) so that *A* may be accessed by other transactions. Hence binary lock enforces mutual exclusiol1 on the data item

Share/Exclusive (for Read/Write) Locks

We should allow several transactions to access the same item *A* if they all access *A'* for reading purposes only. However, if a transaction is to write an item A, it must have exclusive access to *A.* For this purpose, a different type of lock called a multiple-mode lock is used. In this scheme there are shared/exclusive or read/write locks are used.

Locking operations

There are three locking operations called read_lock(A), write_lock(A) and unlock(A) represented as lock-S(A), lock-X(A), unlock(A) (Here, S indicates shared lock, X indicates exclusive lock)can be performed on a data item. A lock associated with an item A, LOCK (A), now has three possible states: "read-locked", "write-locked," or "unlocked." A read-locked item is also called share-locked item because other transactions are allowed to read the item, whereas a write-locked item is caused exclusive-locked, because a single transaction exclusively holds the lock on the item.

Q6) what do you mean by schedule?(2 mark-2019(w)-1(c))

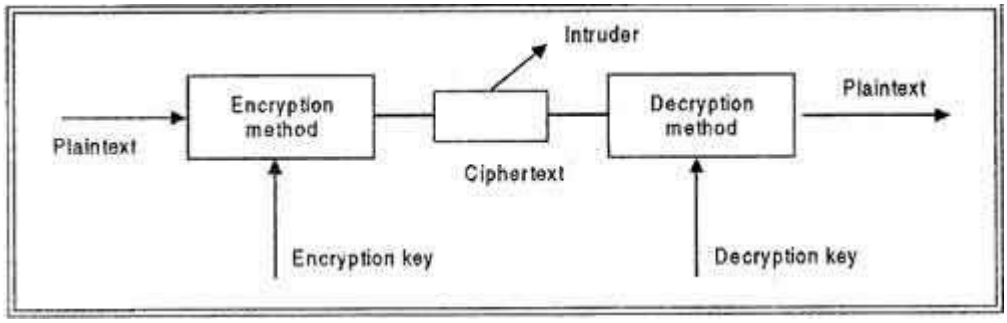
Schedule − A chronological execution sequence of a transaction is called a schedule. A schedule can have many transactions in it, each comprising of a number of instructions/tasks.

Chapter-8

Q1. Explain different types of technique used for Encryption process.

A DBMS can use encryption to protect information in certain situations where the normal security mechanisms of the DBMS are not adequate. For example, an intruder may steal tapes containing some data or tap a communication line. By storing and transmitting data in an encrypted form, the

DBMS ensures that such stolen data is not intelligible to the intruder. Thus, encryption is a technique to provide privacy of data.



In encryption, the message to be encrypted is known as plaintext. The plaintext is transformed by a function that is parameterized by a key. The output of the encryption process is known as the cipher text. Ciphertext is then transmitted over the network. The process of converting the plaintext to ciphertext is called as Encryption and process of converting the ciphertext to plaintext is called as Decryption. Encryption is performed at the transmitting end and decryption is performed at the receiving end. For encryption process we need the encryption key and for decryption process we need decryption key as shown in figure. Without the knowledge of decryption key intruder cannot break the ciphertext to plaintext. This process is also called as Cryptography.
The basic idea behind encryption is to apply an encryption algorithm, which may' be accessible to the intruder, to the original data and a user-specified or DBA-specified encryption key, 'which is kept secret. The output of the algorithm is the encrypted version of the data. There is also a decryption algorithm, which takes the encrypted data and the decryption key as input and then returns the original data. Without the correct decryption key, the decryption algorithm produces gibberish. Encryption and decryption keys may be same or· different but there must be relation between the both which must me secret.

Techniques used for Encryption

There are following techniques used for encryption process:

• Substitution Ciphers

• Transposition Ciphers

Substitution Ciphers: In a substitution cipher each letter or group of letters is replaced by another letter or group of letters to mask them For example: a is replaced with D, b with E, c with F and z with C. In this way attack becomes DWWDFN. The substitution ciphers are not much secure because intruder can easily guess the substitution characters.
Transposition Ciphers: Substitution ciphers preserve the order of the plaintext symbols but mask them-;-The transposition cipher in contrast reorders the letters but do not mask them. For this process a key is used. For example: iliveinqadian may be coded as divienaniqnli. The transposition ciphers are more secure as compared to substitution ciphers.

It uses both a substitution of characters and a rearrangement of their order on the basis of an encryption key. The main weakness of this approach is that authorized users must be told the encryption key, and the mechanism for communicating this information is vulnerable to clever intruders.

Public Key Encryption

Another approach to encryption, called public-key encryption, has become increasingly popular in recent years. The encryption scheme proposed by Rivest, Shamir, and Adheman, called RSA, is a well-knm.vnexample of public-key encryption. Each authorized user has a public encryption key, known to everyone and a private decryption key (used by the decryption algorithm), chosen by the user and known only to him or her. The encryption and decryption algorithms themselves are assumed to be publicly known.

Consider user called Suneet. Anyone can send Suneet a secret message by encrypting the message using Sunset's publicly known encryption key. Only Suneet can decrypt this secret message because the decryption algorithm required Suneet's decryption key, known only to Suneet. Since users choose their own decryption keys, the weakness 0f DES is avoided.

The main issue for public-key encryption is how encryption and decryption keys are chosen. Technically, public-key encryption algorithms rely on the existence of one-way functions, which are functions whose inverse is computationally very hard to determine.

The RSA algorithm, for example is based on the observation that although checking whether a given number of prime is easy, determining the prime factors of a nonprime number is extremely hard. (Determining the prime factors of a number with over 100 digits can take years of CPU-time on the fastest available computers today.)

We now sketch the intuition behind the RSA algorithm, assuming that the data to be encrypted is an integer 1. To choose an encryption key and a decryption key, our friend Suneet-- create a public key by computing the product of two large prime numbers: PI and P2. The private key consists of the pair (PI, P2) and decryption algorithms cannot be used if the product of PI and P2 is known. So we publish the product PI *P2, but an unauthorized user would need to be able to factor PIP2 to steal data. By choosing PI and P2 to be sufficiently large (over 100 digits), we can make it very difficult (or nearly impossible) for an intruder to factorize it.

Q2. What are various security level used protect a database.

Types of Security

Database security is a broad area that addresses many issues, including the following:

■Various legal and ethical issues regarding the right to access certain information

— For example, some information may be deemed to be private and cannot be accessed legally by unauthorized organizations or persons. In the United States, there are numerous laws governing privacy of information.

■Policy issues at the governmental, institutional, or corporate level as to what kinds of information should not be made publicly available

—for example, credit ratings and personal medical records.

■System-related issues such as the system level at which various security functions should be enforced

— for example, whether a security function should be handled at the physical hardware level, the operating system level, or the DBMS level.

■The need in some organizations to identify multiple security levels and to categorize the data and users based on these classifications

—for example, top secret, secret, confidential, and unclassified. The security policy of the organization with respect to permitting access to various classifications of data must be enforced.

Threats to Databases.

Threats to databases can result in the loss or degradation of some or all of the following commonly accepted security goals: integrity, availability, and confidentiality.

■Loss of integrity.

Database integrity refers to the requirement that information be protected from improper modification. Modification of data includes creation, insertion, updating, changing the status of data, and deletion. Integrity is lost if unauthorized changes are made to the data by either intentional or accidental acts. If the loss of system or data integrity is not corrected, continued use of the contaminated system or corrupted data could result in inaccuracy, fraud, or erroneous decisions.

■Loss of availability.

Database availability refers to making objects available to a human user or a program to which they have a legitimate right.

■Loss of confidentiality.

Database confidentiality refers to the protection of data from unauthorized disclosure. The impact of unauthorized disclosure of confidential information can range from violation of the Data Privacy Actto the jeopardization of national security. Unauthorized, unanticipated, or unintentional disclosure could result in loss of public confidence, embarrassment, or legal action against the organization. To protect databases against these types of threats, it is common to implement

Q3) Define Encryption. Name the different types of encryption process.(2mark-2019(w)-1(g))

Database encryption is the process of converting data, within a database, in plain text format into a meaningless cipher text by means of a suitable algorithm.

Database decryption is converting the meaningless cipher text into the original information using keys generated by the encryption algorithms.