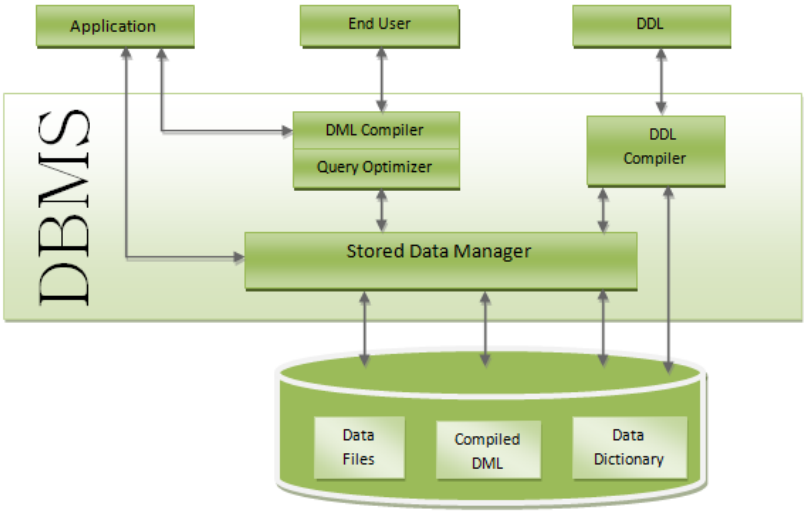
UNIT-I

1. Describe the Structure of Database Management system with a neat diagram.

ANS:



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

Integrity Manager

Transaction Manager

File Manager

Buffer Manager

3. Disk Storage:

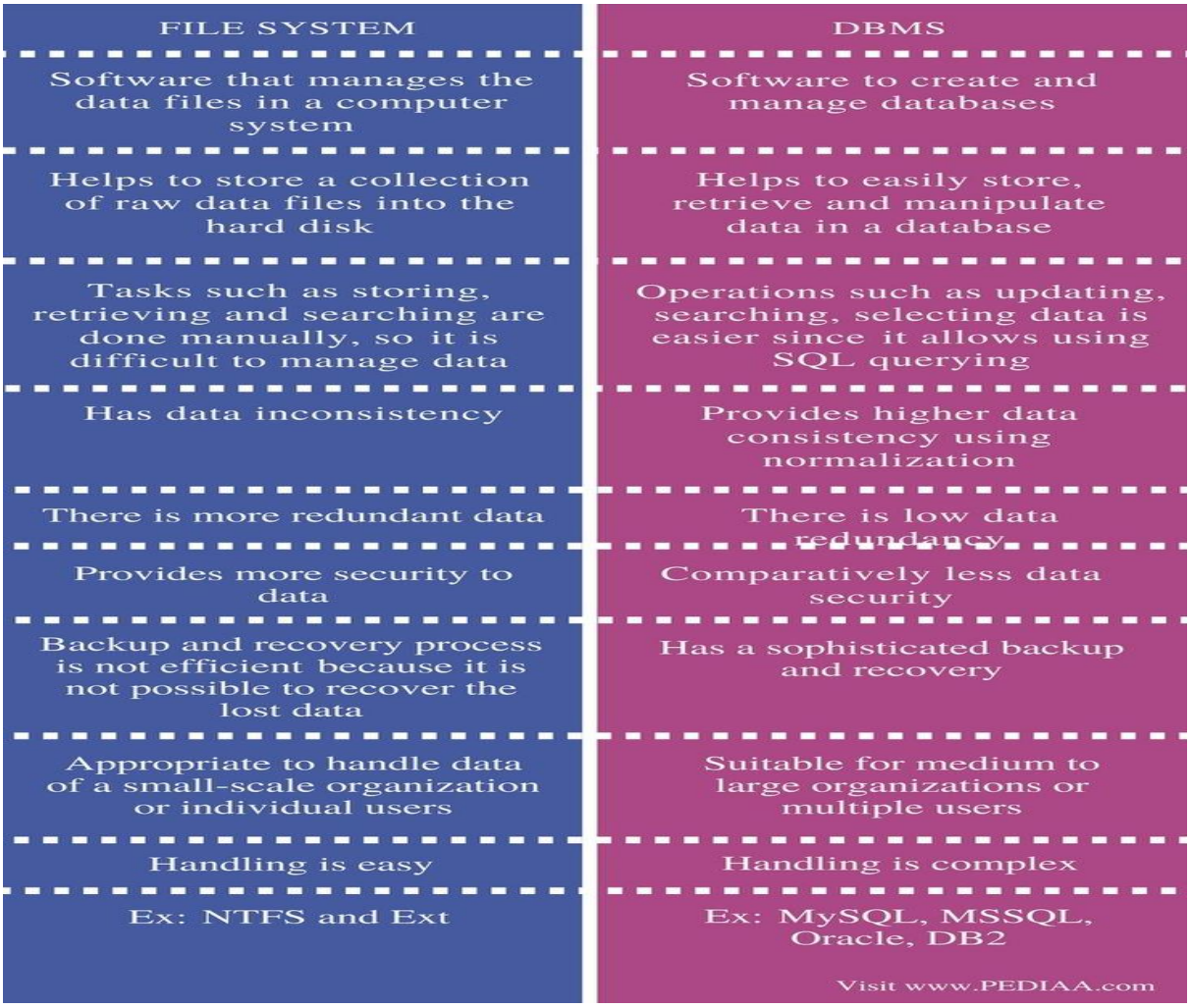
It contains the following components –

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**.**

2. Differentiate between File System and Database Management System.



3. What is Data Model? List out various Data Model and explain with suitable example.

ANS:

Data Models define how the logical structure of a database is modeled. 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.

FOUR TYPES:

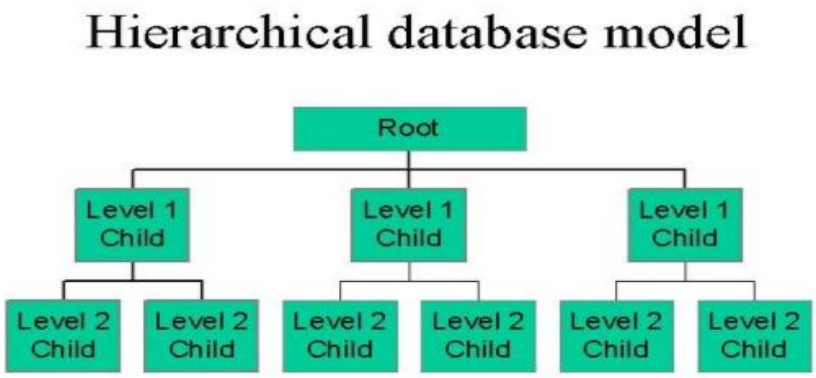
1. Hierarchical database
2. Network database
3. Relational database
4. Object-Oriented database

1. Hierarchical database: In a Hierarchical database model, data is organized in a tree-like structure.

- Data is Stored Hierarchically (top down or bottom up) format.
- Data is represented using a parent-child relationship.
- In Hierarchical DBMS, parent may have many children, but children have only one parent.

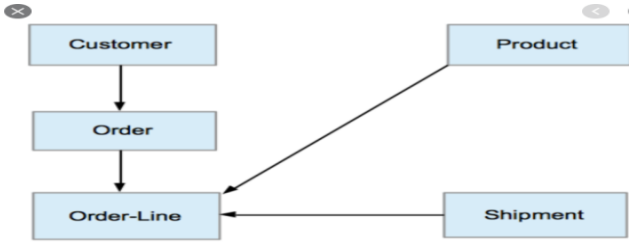
Note: This is old model, presently not using.

Because of Single DB concept**.**

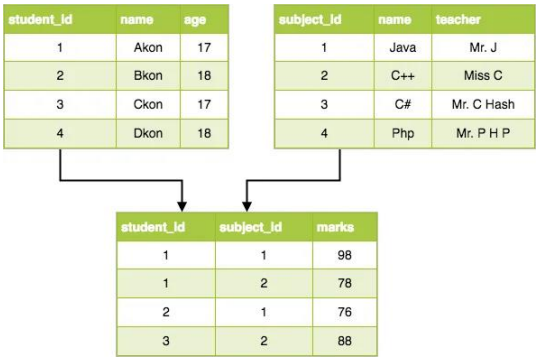


2. Network database: The network database model allows each child to have multiple parents.

- It helps you to address the need to model more complex relationships like as the orders/parts many-to-many relationship.
- In this model, entities are organized in a graph which can be accessed through several paths.
- Entity in DBMS can be a real-world object with an existence, For example, in a College database, the entities can be Professor, Students, Courses, etc. ... The attribute value gets stored in the database.

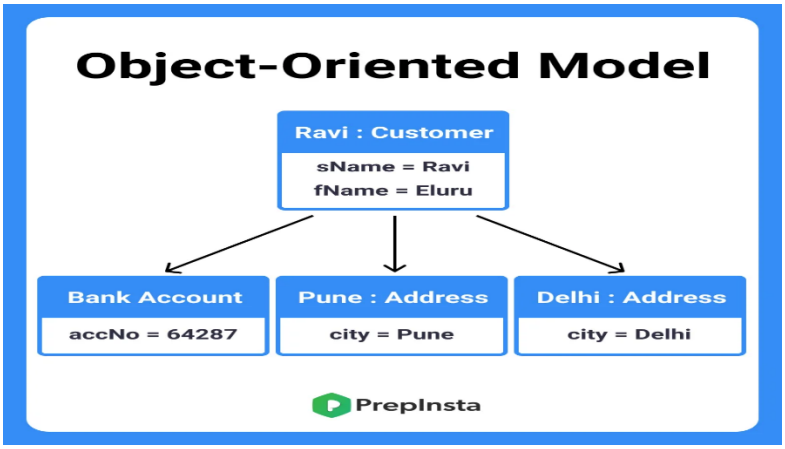


3. Relational database: In this model, data is organized in two-dimensional tables and the relationship is maintained by storing a common field.



4. Object-Oriented database:

- **In Object-oriented Model data stored in the form of objects.**
- **The structure which is called classes which display data within it.**
- **It defines a database as a collection of objects which stores both data members values and operations.**



4. a) Define Database Management System (DBMS). List out the advantages of DBMS.

ANS:

Database Management System (DBMS) Definition - What does Database Management System (DBMS) mean? A database management system (DBMS) is a software package designed to define, manipulate, retrieve and manage data in a database. A DBMS generally manipulates the data itself, the data format, field names, record structure and file structure.

Applications of DBMS:

- Banking.
- Airlines.
- Universities.
- Manufacturing and selling.
- Human resourses.

Advantages of DBMS:

1. Data Redundancy

Unlike traditional file-system storage, Data Redundancy in DBMS is very less or not present.

2. Data Inconsistency

In traditional file system storage, the changes made by one user in one application doesn't update the changes in other application, given both have the same set of details.

3. Data Sharing

Data Sharing is the primary advantage of Database management systems. DBMS system allows users and applications to share Data with multiple applications and users.

4. Data Searching

Searching and retrieving of data is very easy in DBMS systems.

5. Data Security

DBMS systems provide a strong framework to protect data privacy and security.

6. Data Concurrency

In DBMS, Data are stored in one or more servers in the network and that there is some software locking mechanism that prevents the same set of data from being changed by two people at the same time.

7. Data Integration

Data integration is a process of combining the data residing at different locations and present the user with a unified view of data.

8. Data Backup and Recovery

This is another advantage of DBMS as it provides a strong framework for Data backup, users are not required to back up their data periodically and manually, it is automatically taken care by DBMS. Moreover, in case of a server crash, DBMS restores the Database to its previous condition.

b) List out various applications of Database Management Systems.

Ans:

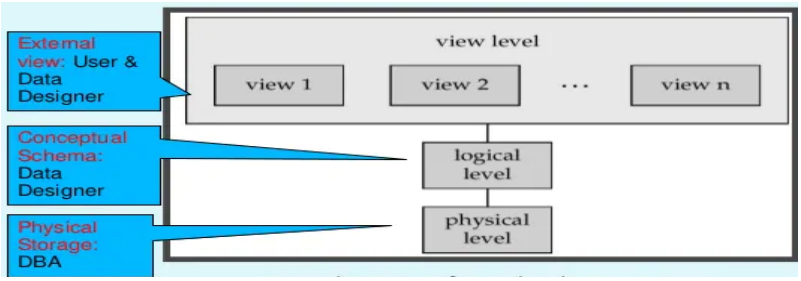
- Railway Reservation System
- Library Management System
- Banking
- Universities and colleges
- Credit card transactions
- Social Media Sites
- Telecommunications
- Finance
- Military
- Online Shopping
- Human Resource Management
- Manufacturing
- Airline Reservation system
- Agriculture

5. a) What is meant by Data Abstraction? Describe various levels of Data Abstraction.

ANS:

Data Abstraction: We use Data Structures (DS) to store(represent) data in the Database (DB). Managing the DB efficiently in order to access data and perform calculations and manipulations to get required report/output is known as DBMS. Or Data Abstraction is a process of hiding unwanted or irrelevant details from the end user. It provides a different view and helps in achieving data independence which is used to enhance the security of data.

Levels of Data Abstractions in a DBMS – There are 3 View Levels of Abstractions –



Physical level/Internal level : The physical representation of the database on the computer. This level describes *how* the data is stored in the database.

- It includes :
  - Where the data is located
  - File structures
  - Access methods
  - Indexes.

The physical schema is managed by the DBA.

Logical level/Conceptual level: The community view of the database. This level describes *what* data is stored in the database and the relationships among the data.

- What are the entities and Relationships in organization.
- What information these entities and relationships should store in database.
- What integrity constraints/business rules it should have?
- It consists of the schemas we have described with CREATE TABLE statements.

View level/ External Level: The users view of the database. This level describes that part of the database that is relevant to each user.

- Each external schema is a combination of base tables and views, tailored to the needs of a single user.
- It is managed by the data designer and the user.

b) Define Data Independence. Describe Physical Data Independence and Logical Data

Independence with a neat diagram.

Ans:

Data Independence

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

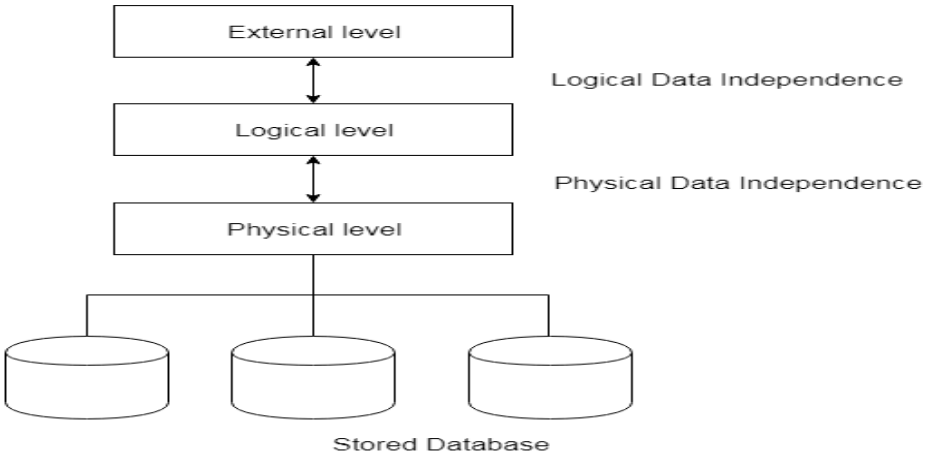
There are two types of data independence:

1. Logical Data Independence

- Logical data independence refers characteristic of being able to change the conceptual schema without having to change the external schema.
- Logical data independence is used to separate the external level from the conceptual view.
- If we do any changes in the conceptual view of the data, then the user view of the data would not be affected.
- Logical data independence occurs at the user interface level.

2. Physical Data Independence

- Physical data independence can be defined as the capacity to change the internal schema without having to change the conceptual schema.
- If we do any changes in the storage size of the database system server, then the Conceptual structure of the database will not be affected.
- Physical data independence is used to separate conceptual levels from the internal levels.
- Physical data independence occurs at the logical interface level.



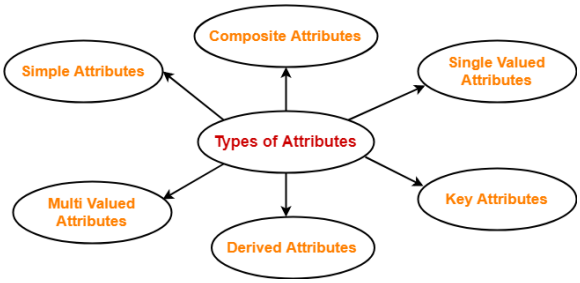
UNIT-II

1. a) Illustrate various types of attributes with notations.

Ans:

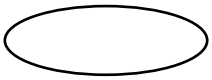
Attribute: A relationship is an important part of any Entity relationship diagram as it shows the relation between two different entities.

The six types of attributes:



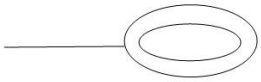
Single Valued Attributes: It is an attribute with only one value.

Symbol:



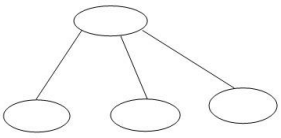
Multi Valued Attributes: These are the attributes which can have multiple values for a single or same entity.

Symbol:



Compound / Composite attributes: This attribute can be further divided into more attributes.

Symbol:



simple / Atomic Attributes: The attributes which cannot be further divided are called as simple / atomic attributes.

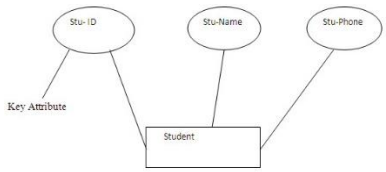
Symbol:

Derived Attributes: These attributes are derived from other attributes. It can be derived from multiple attributes and also from a separate table.

Symbol:

Key Attributes: This attribute represents the main characteristic of an entity i.e. primary key. Key attribute has clearly different value for each element in an entity set.

Symbol:



b) What is meant by Mapping Cardinality? List and explain types of Cardinality Ratios.

Ans: In the view of databases, cardinality refers to the uniqueness of data values that are contained in a column. High cardinality is nothing but the column contains a large percentage of totally unique values. Low cardinality is nothing but the column which has a lot of “repeats” in its data range.

Cardinality between the tables can be of type one-to-one, many-to-one or many-to-many.

Mapping Cardinality

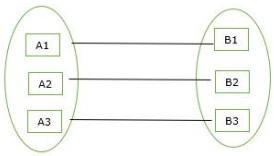
It is expressed as the number of entities to which another entity can be associated via a relationship set.

For binary relationship set there are entity set A and B then the mapping cardinality can be one of the following −

- One-to-one
- One-to-many
- Many-to-one
- Many-to-many

One-to-one relationship

One entity of A is associated with one entity of B.

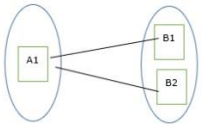


Example

Given below is an example of the one-to-one relationship in the mapping cardinality. Here, one department has one head of the department (HOD).

One-to-many relationship

An entity set A is associated with any number of entities in B with a possibility of zero and entity in B is associated with at most one entity in A

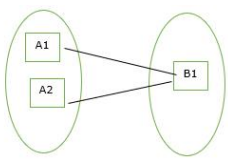


Example

Given below is an example of the one-to-many relationship in the mapping cardinality. Here, one department has many faculties.

Many-to-one relationship

An entity set A is associated with at most one entity in B and an entity set in B can be associated with any number of entities in A with a possibility of zero.



Example

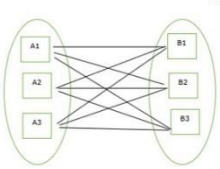
Given below is an example of the many-to-one relationship in the mapping cardinality. Here, many faculties work in one department.

Many-to-many relationship

Many entities of A are associated with many entities of B.

An entity in A is associated with many entities of B and an entity in B is associated with many entities of A.

Many to many=many to one + one to many



Example

Given below is an example of the many-to-many relationship in the mapping cardinality. Here, many employees work on many projects.

2. Construct an Entity Relationship diagram for a University database. Assume your own entities (Minimum of 5 entities), attributes and relations.

Entity: A thing in the real world with independent existence. Any particular row (a record) in a relation(table) is known as an entity.

An entity is written in

1. Strong entity

2. Weak entity

Relation: A relationship is an important part of any Entity relationship diagram as it shows the relation between two different entities.

There are four types of relations:

One to one – 1:1

One to many – 1:M

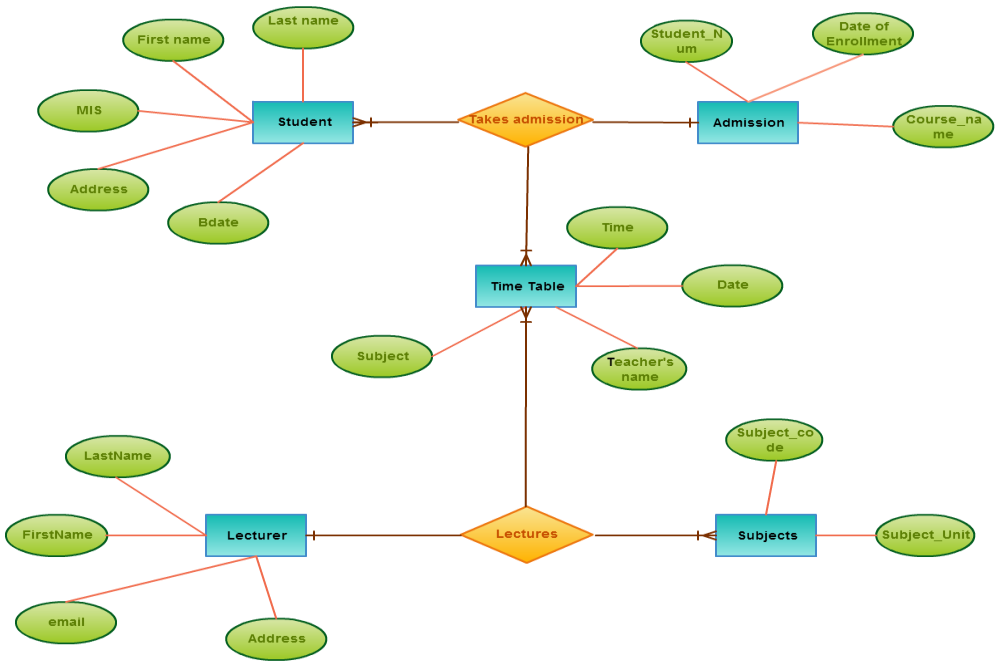
Many to one – M:1

Many to many – M:M

Attribute: A relationship is an important part of any Entity relationship diagram as it shows the relation between two different entities.

- Simple attribute
- Composite attribute
- Single valued attribute
- Multi valued attribute
- Derived attribute
- Key attributes

Entity Relationship diagram for an University database:

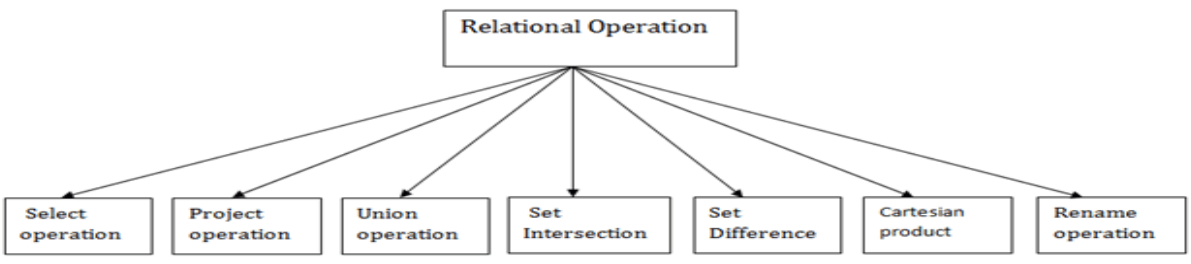


3. a) What is Relational Algebra? List out various types of Relational operations and explain with an example.

Ans:

Relational algebra is a procedural query language. It gives a step by step process to obtain the result of the query (SQL). It uses operators to perform queries.

Types of Relational operation – 7



1. Select Operation: The select operation selects tuples that satisfy a given predicate.(codition)

* It is denoted by sigma (σ).

Notation: σ p(r)

Ex: **σ BRANCH_NAME="Hyderabad" (LOAN)**

BRANCH_NAME	LOAN_NO	AMOUNT
Hyderabad	L-17	1000
Bangalore	L-23	2000
Chennai	L-15	1500
Vijayawada	L-14	1500
Hyderabad	L-13	500
Delhi	L-11	900
Bombay	L-16	1300

Output:

BRANCH_NAME	LOAN_NO	AMOUNT
Hyderabad	L-17	1000
Hyderabad	L-13	500

2. Project Operation: This operation shows the list of those attributes that we wish to appear in the result. Rest of the attributes are eliminated from the table.

* It is denoted by ∏

Notation: ∏ A1, A2, An (r)

Ex: ∏ NAME, CITY (CUSTOMER)

NAME	STREET	CITY
Jones	Main	Harrison
Smith	North	Rye
Hays	Main	Harrison
Curry	North	Rye
Johnson	Alma	Brooklyn
Brooks	Senator	Brooklyn

Output:

NAME	CITY
Jones	Harrison
Smith	Rye
Hays	Harrison
Curry	Rye
Johnson	Brooklyn
Brooks	Brooklyn

3. Union Operation: Suppose there are two tuples in R and S table. The union operation contains all the tuples that are either in R or S or both .

* It eliminates the duplicate tuples (records/rows). It is denoted by ∪.

Notation: R ∪ S

Ex: DEPOSITOR RELATION(R table) **BARROW RELATION (S table)**

CUSTOMER_NAME	ACCOUNT_NO
Johnson	A-101
Smith	A-121
Mayes	A-321
Turner	A-176
Johnson	A-273
Jones	A-472
Lindsay	A-284

CUSTOMER_NAME	LOAN_NO
Jones	L-17
Smith	L-23
Hayes	L-15
Jackson	L-14
Curry	L-93
Smith	L-11
Williams	L-17

∏ CUSTOMER_NAME (BORROW) ∪ ∏ CUSTOMER_NAME (DEPOSITOR)

CUSTOMER_NAME
Johnson
Smith
Hayes
Turner
Jones
Lindsay
Jackson
Curry
Williams
Mayes

4. Set Intersection Operation: Suppose there are two tuples (records/rows) in table R and S. The set intersection operation contains all tuples that are in both R & S.

* It is denoted by intersection ∩.

Notation: R ∩ S

Ex: **∏ CUSTOMER_NAME (BORROW) ∩ ∏ CUSTOMER_NAME (DEPOSITOR)**

CUSTOMER_NAME
Smith
Jones

5. Set Difference Operation: Suppose there are two tuples in R and S table. The set difference operation contains all tuples that are in R but not in S.

* It is denoted by intersection minus (-).

Notation: R - S

Ex: **∏ CUSTOMER_NAME (BORROW) - ∏ CUSTOMER_NAME (DEPOSITOR)**

CUSTOMER_NAME
Jackson
Hayes
Willians
Curry

6. Cartesian Product Operation: The Cartesian product is used to combine each row in one table with each row in the other table. It is also known as a cross product.

* It is denoted by X.

Notation: E X D

EMPLOYEE (table) **DEPARTMENT: (table)**

EMP_ID	EMP_NAME	EMP_DEPT
1	Smith	A
2	Harry	C
3	John	B

DEPT_NO	DEPT_NAME
A	Marketing
B	Sales
C	Legal

EMPLOYEE X DEPARTMENT

EMP_ID	EMP_NAME	EMP_DEPT	DEPT_NO	DEPT_NAME
1	Smith	A	A	Marketing
1	Smith	A	B	Sales
1	Smith	A	C	Legal
2	Harry	C	A	Marketing
2	Harry	C	B	Sales
2	Harry	C	C	Legal
3	John	B	A	Marketing
3	John	B	B	Sales
3	John	B	C	Legal

7. Rename Operation:

*** The rename operation is used to rename the output relation.**

***It is denoted by rho (ρ).**

Input:

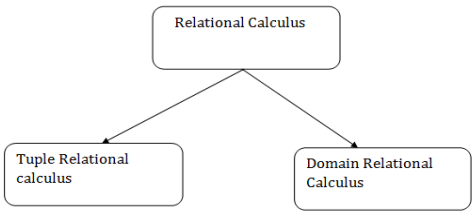
ρ(STUDENT1, STUDENT)

b) What is Relational Calculus? Describe types of Relational Calculus with suitable example.

Ans:

Relational Calculus: Relational calculus is a non-procedural query language. In the non-procedural query language, the user is concerned with the details of how to obtain the end results.

Types of Relational calculus:



Tuple Relational Calculus (TRC):

- The tuple relational calculus is specified to select the tuples in a relation. In TRC, filtering variable uses the tuples of a relation.

The result of the relation can have one or more tuples.

- Notation:

  {T | P (T)} or {T | Condition (T)}

- P(T) – logically uses OR, AND, NOT connecttors.

- It also uses quantifiers– FORALL, THERE EXIST

Example:

FN	LN	Age
A	AA	35
B	BB	60

{t.LN/ student(t) AND t.age>30}

output:

LN
AA
BB

It returns a tuple with "name" from author who has written on article as "database"

Display the lastname of those students where age is greater than 30.

Domain Relational Calculus (DRC):

In domain relational calculus, filtering is done based on the domain of the attributes and not based on the tuple values.

Syntax: { c1, c2, c3, ..., cn | F(c1, c2, c3, ... ,cn)}

where, c1, c2... etc represents domain of attributes(columns)

and F defines the formula including the condition for fetching the data.

For example,

name	age
AA	19
BB	20

{< name, age > | ∈ Student ∧ age > 17}

OUTPUT:

name	age
AA	19
BB	20

Again, the above query will return the names and ages of the students in the table Student who are older than 17.

4. Translate Entity Relationship diagram into a collection of tables with associated

constraints to a relational database schema.

5. Define view. Write the syntax for creating,updating and destroying a view with suitable example for Simple and Complex Views.

Ans: **Views:** Views in SQL are kind of virtual tables. A view also has rows and columns as they are in a real table in the database.

Simple views:

CREATE VIEW : We can create View using **CREATE VIEW** statement. A View can be created from a single table or multiple tables.

Syntax:

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

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

SELECT * FROM DetailsView;

- Output:

NAME	ADDRESS
Harsh	Kolkata
Ashish	Durgapur
Pratik	Delhi
Dhanraj	Bihar

CREATE VIEW StudentNames AS SELECT S_ID, NAME FROM StudentDetails ORDER BY NAME;

SELECT * FROM StudentNames;

Output:

S_ID	NAMES
2	Ashish
4	Dhanraj
1	Harsh
3	Pratik
5	Ram

DELETING VIEWS: We have learned about creating a View, but what if a created View is not needed any more? Obviously we will want to delete it. SQL allows us to delete an existing View. We can delete or drop a View using the DROP statement.

Syntax:

- **DROP VIEW view_name;**

Ex:DROP VIEW MarksView;

UPDATING VIEWS: There are certain conditions needed to be satisfied to update a view. If any one of these conditions is **not** met, then we will not be allowed to update the view.

Syntax:CREATE OR REPLACE VIEW view_name AS SELECT column1,coulmn2,.. FROM table_name ;

COMPOSITE VIEWS

CREATE VIEW:

In this example we will create a View named MarksView from two tables StudentDetails and StudentMarks. To create a View from multiple tables we can simply include multiple tables in the SELECT statement.

->CREATE VIEW MarksView AS SELECT StudentDetails.NAME, StudentDetails.ADDRESS, StudentMarks.MARKS FROM StudentDetails, StudentMarks WHERE StudentDetails.NAME = StudentMarks.NAME;

- SELECT * FROM MarksView;

Output:

NAME	ADDRESS	MARKS
Harsh	Kolkata	90
Pratik	Delhi	80
Dhanraj	Bihar	95
Ram	Rajasthan	85

UPDATING VIEWS:

For example, if we want to update the view **MarksView** and add the field AGE to this View from **StudentMarks** Table, we can do this as:

->CREATE OR REPLACE VIEW MarksView AS SELECT StudentDetails.NAME, StudentDetails.ADDRESS, StudentMarks.MARKS, StudentMarks.AGE FROM StudentDetails, StudentMarks WHERE StudentDetails.NAME = StudentMarks.NAME;

->SELECT * FROM MarksView;

Output:

NAME	ADDRESS	MARKS	AGE
Harsh	Kolkata	90	19
Pratik	Delhi	80	19
Dhanraj	Bihar	95	21
Ram	Rajasthan	85	18

UNIT-III

1. Identify and list various Data definition Language (DDL) commands. Create a table by specifying key and referential constraints in Structured Query Language.

Ans:

DDL stands for Data Definition Language. As the name suggests, the DDL commands help to define the structure of the databases or schema. When we execute DDL statements, it takes effect immediately. The

changes made in the database using this command are saved permanently because its commands are auto-committed. The following commands come under DDL language:

- **CREATE:** It is used to create a new database and its objects such as table, views, function, stored procedure, triggers, etc.

Syntax –

CREATE TABLE table_name (column_1 datatype, column_2 datatype, column_3 datatype, ...);

Example –

CREATE TABLE Student_info (College_Id number(2), College_name varchar(30), Branch varchar(10));

- **DROP:** It is used to delete the database and its objects, including structures, from the server permanently.

Syntax –

DROP TABLE table_name;

Example –

If the College Authority wants to change their Database by deleting the Student_info Table.

DROP TABLE Student_info;

- **ALTER:** It's used to update the database structure by modifying the characteristics of an existing attribute or adding new attributes.

Syntax –

Syntax to add a column to an existing table.

ALTER TABLE table_name ADD column_name datatype;

Example –

In our Student_info table, we want to add a new column for CGPA. The syntax would be as below as follows.

ALTER TABLE Student_info ADD CGPA number;

- **TRUNCATE:** It is used to completely remove all data from a table, including their structure and space allocates on the server.

Syntax –

Syntax to remove an existing table.

TRUNCATE TABLE table_name;

Example –

TRUNCATE TABLE Student_info;

- **RENAME:** This command renames the content in the database.

Syntax:

alter table oldtable-name rename to newtablename;

Example query:

Alter table abc rename to def;

2. List various Integrity Constraints. Explain the constraints with suitable example.

Ans: Integrity constraints in Database Management Systems (DBMS) are a set of rules that are applied on the table columns or relationships to ensure that the overall validity, integrity, and consistency (i.e. the quality) of the data present in the database table is maintained. Each and every time a table insert, update, delete, or alter operation is performed, it is evaluated against the terms or rules mentioned in the integrity constraint. The data is inserted, updated, deleted, or altered only if the result of the constraint comes out to be True. Thus, integrity constraint prevents accidental damage to the database by an authorized user.

Types of Integrity Constraints in DBMS

- Domain Integrity Constraint
- Entity Integrity Constraint
- Referential Integrity Constraint
- Key Constraints

1. Domain Integrity Constraint

A domain integrity constraint is a set of rules that restricts the kind of attributes or values a column or relation can hold in the database table. For example, we can specify if a particular column can hold null values or not, if the values have to be unique or not, the data type or size of values that can be entered in the column, the default values for the column, etc.

```
CREATE TABLE customer_details(customer_id character varying(255) NOT NULL,customer_name character varying(255) NOT NULL,quantity integer NOT NULL, date_purchased date);
```

2. Entity Integrity Constraint

Entity Integrity Constraint is used to ensure the uniqueness of each record or row in the data table. There are primarily two types of integrity constraints that help us in ensuring the uniqueness of each row, namely, UNIQUE constraint and PRIMARY KEY constraint.

```
CREATE TABLE Students(Student_ID int NOT NULL,Student_Name varchar(255) NOT NULL,Class_Name varchar(255) UNIQUE,Age int,PRIMARY KEY (Student_ID));
```

3. Referential Integrity Constraint

Referential Integrity Constraint ensures that there always exists a valid relationship between two tables.This makes sure that if a foreign key exists in a table relationship then it should always reference a corresponding value in the second table or it should be null.

```
CREATE TABLE Department(Department_ID int NOT NULL,Department_Name varchar(255) NOT NULL,PRIMARY KEY(Department_ID));

CREATE TABLE Employees(Employee_ID int NOT NULL,Employee_Name varchar(255) NOT NULL,
Department int NOT NULL,Age int,FOREIGN KEY (Department) REFERENCES Department(Department_ID));
```

4. Key Constraints

There are a number of key constraints in SQL that ensure that an entity or record is uniquely or differently identified in the database. There can be more than one key in the table but it can have only one primary key.

Some of the key constraints in SQL are :

1. Primary Key Constraint
2. Foreign Key Constraint
3. Unique Key Constraint

3. Identify and list various Data Manipulation Language (DML) commands. Explain DML commands with suitable example.

Ans:

- It stands for Data Manipulation Language. The DML commands deal with the manipulation of existing records of a database. It is responsible for all changes that occur in the database. The changes made in the database using this command can't save permanently because its commands are not auto-committed. Therefore, changes can be rollback. The following commands come under DML language:

SELECT: This command is used to extract information from a table.

Syntax:

```
SELECT *from  table-name;
```

Example:

```
SELECT *from student;
```

INSERT: It is a SQL query that allows us to add data into a table's row.

Syntax:

```
INSERT INTO table-name values(colvalues);
```

Example:

```
INSERT INTO student values (name, lastname) VALUES (‘cde', ‘abc');
```

UPDATE: This command is used to alter or modify the contents of a table.

Syntax:

update tablename set col=‘new value’ where col=‘oldvalue’;

Example:

UPDATE student SET name = 'Dima' WHERE lastname = 'Shiva';

DELETE: This command is used to delete records from a database table, either individually or in groups.

Syntax:

DELETE FROM tablename WHERE colmn =‘value’;

Example:

DELETE FROM student WHERE name = 'Dima';

4. Make use of Structured Query Language in solving the following with proper example

i) Sub queries & Correlated Queries

ii) Joins

iii) Aggregate functions

Ans:

i) Sub queries & Correlated Queries

sub query:

query with in a query. A subquery is a select statement that is nested within another select statement.

Syntax: query1(query2);

Query 1 is outer query and query2 is called inner query.

Ex: **a) retrieve employee names from emptable.**

b) who are having highest salary.

-> select empname from emp where sal=(select max(sal) from emp);

Correlated Queries

Any query is called as correlated subquery when both the inner query and the outer query are independent.

Syntax:

Select *from TN table alias name1 where n-1=(select count(col name) from TN table alias name 2 where table alias name2.colname/table alias name1.column

Ex: **1st highest salary of employee using correlated subquery**

-> select *from emp E1 where 0=(select count (sal) from emp E2 where E2.sal>E1.sal;

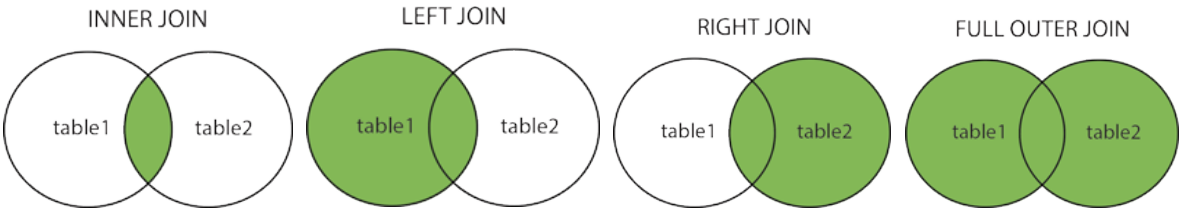
ii) Joins

JOINS:

JOIN means to combine something. In case of SQL, JOIN means "to combine two or more tables".In SQL, JOIN clause is used to combine the records from two or more tables in a database.

Here are the different types of the JOINs in SQL:

- (INNER) JOIN: Returns records that have matching values in both tables
- LEFT (OUTER) JOIN: Returns all records from the left table, and the matched records from the right table
- RIGHT (OUTER) JOIN: Returns all records from the right table, and the matched records from the left table
- FULL (OUTER) JOIN: Returns all records when there is a match in either left or right table



Following is the **class** table

ID	NAME
1	abhi
2	adam
4	alex

class_info table

ID	ADDRESS
1	Delhi
2	Mumbai
3	Chennai

1. INNER JOIN

SYNTAX:

SELECT table1.column1, table1.column2, table2.column1,.... FROM table1 INNER JOINtable2 ON table1.matching_column = table2.matching_column;

EXAMPLE: SELECT * from class INNER JOIN class_info where class.id = class_info.id;

ID	NAME	ID	ADDRESS
1	abhi	1	Delhi
2	adam	2	Mumbai

2. LEFT JOIN

Syntax:

SELECT table1.column1, table1.column2, table2.column1,....FROM table1 LEFT JOIN table2 ON table1.matching_column = table2.matching_column;

EXAMPLE: SELECT * FROM class LEFT OUTER JOIN class_info ON (class.id = class_info.id);

ID	NAME	ID	ADDRESS

1	abhi	1	Delhi
2	adam	2	Mumbai
4	alex	NULL	NULL

3. RIGHT JOIN

Syntax:
SELECT table1.column1, table1.column2, table2.column1,....FROM table1 RIGHT JOIN table2 ON table1.matching_column = table2.matching_column;

EXAMPLE: SELECT * FROM class RIGHT OUTER JOIN class_info ON (class.id = class_info.id);

ID	NAME	ID	ADDRESS
1	abhi	1	Delhi
2	adam	2	Mumbai
NULL	NULL	3	Chennai

4. FULL JOIN

Syntax:SELECT table1.column1, table1.column2, table2.column1,....FROM table1
FULL JOIN table2 ON table1.matching_column = table2.matching_column;

EXAMPLE: SELECT * FROM class FULL OUTER JOIN class_info ON (class.id = class_info.id);

ID	NAME	ID	ADDRESS
1	abhi	1	Delhi
2	adam	2	Mumbai
NULL	NULL	3	Chennai
4	alex	NULL	NULL

iii) Aggregate functions

In database management an aggregate function is a function where the values of multiple rows are grouped together as input on certain criteria to form a single value of more significant meaning.

Various Aggregate Functions

1) Count() 2) Sum() 3) Avg() 4) Min() 5) Max()

student

Id	Name	Marks
1	A	95
2	B	93
3	c	92

COUNT FUNCTION

COUNT function is used to Count the number of rows in a database table. It can work on both numeric and non-numeric data types.

Syntax

COUNT(*) or COUNT([ALL|DISTINCT] expression)

Ex: select count(*) from student;

Output:

3

SUM Function

- Sum function is used to calculate the sum of all selected columns. It works on numeric fields only.

Syntax:

SUM([ALL|DISTINCT] expression)

Ex: select sum(Marks) from student;

Output:

280

AVG function

The AVG function is used to calculate the average value of the numeric type. AVG function returns the average of all non-Null values.

Syntax

AVG() or AVG([ALL|DISTINCT] expression)

Ex: select avg(marks) from student;

Output: 93.33

MAX Function

MAX function is used to find the maximum value of a certain column. This function determines the largest value of all selected values of a column.

Syntax

MAX() or MAX([ALL|DISTINCT] expression)

Ex: select max(marks) from student;

Output: 95

MIN Function

MIN function is used to find the minimum value of a certain column. This function determines the smallest value of all selected values of a column.

Syntax

MIN() or MIN([ALL|DISTINCT] expression)

Ex: select min(marks) from student;

Output: 92

5. List and explain various commands on Transaction Control Language and Data Control language.

ANS: Data Control Language:

- DCL commands are used to grant and take back authority from any database user.
- Here are some commands that come under DCL:
- Grant
- Revoke

a. Grant: It is used to give user access privileges to a database.

Syntax: grant <permissions>on <tablename to <user list>;

GRANT SELECT, UPDATE ON std TO SOME_USER, ANOTHER_USER;

b. Revoke: It is used to take back permissions from the user.

Syntax: Revoke <permissions>on <tablename from <user list>;

Example

REVOKE SELECT, UPDATE ON std FROM USER1, USER2;

Transaction Control Language:

TCL commands can only use with DML commands like INSERT, DELETE and UPDATE only.

These operations are automatically committed in the database that's why they cannot be used while creating tables or dropping them.

Here are some commands that come under TCL:

- COMMIT
- ROLLBACK
- SAVEPOINT

a. Commit: Commit command is used to save all the transactions to the database.

Syntax:

- COMMIT;

Example:

- DELETE FROM CUSTOMERS
- WHERE AGE = 25;
- COMMIT;

b. Rollback: Rollback command is used to undo transactions that have not already been saved to the database.

Syntax:

ROLLBACK;

Example:

DELETE FROM CUSTOMERS

WHERE AGE = 25;

ROLLBACK;

c. SAVEPOINT: It is used to roll the transaction back to a certain point without rolling back the entire transaction.

Syntax:

- SAVEPOINT SAVEPOINT_NAME;

UNIT-IV

1. Define Schema Refinement. Analyze the problems caused by redundancy.

ANS: 1. Schema Refinement:

The Schema Refinement refers to refine the schema by using some technique. The best technique of schema refinement is decomposition.

Normalisation or Schema Refinement is a technique of organizing the data in the database.

It is a systematic approach of decomposing tables to eliminate data redundancy and undesirable characteristics like Insertion, Update and Deletion Anomalies.

Redundancy refers to repetition of same data or duplicate copies of same data stored in different locations.

Anomalies: Anomalies refers to the problems occurred after poorly planned and normalised databases where all the data is stored in one table which is sometimes called a flat file database.

Anomalies or problems facing without normalization(problems due to redundancy) :

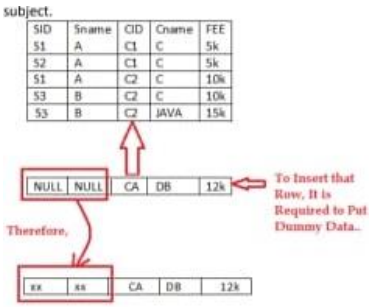
Anomalies refers to the problems occurred after poorly planned and unnormalised databases where all the data is stored in one table which is sometimes called a flat file database. Let us consider such type of schema –

Here all the data is stored in a single table which causes redundancy of data or say anomalies as SID and Sname are repeated once for same CID . Let us discuss anomalies one by one.

Due to redundancy of data we may get the following problems, those are-

1.insertion anomalies : It may not be possible to store some information unless some other

information is stored as well.



2.update anomalies: If one copy of redundant data is updated, then inconsistency is created

unless all redundant copies of data are updated.

3.deletion anomalies: It may not be possible to delete some information without losing some

other information as well.

SID	Sname	CID	Cname	FEE
S1	A	C1	C	5k
S2	A	C1	C	5k
S1	A	C2	C	10k
S3	B	C2	C	10k
S3	B	C3	JAVA	15k

Problem in updation / updation anomaly – If there is updation in the fee from 5000 to 7000,

then we have to update FEE column in all the rows, else data will become inconsistent.

Insertion Anomaly and Deletion Anomaly- These anomalies exist only due to redundancy,

otherwise they do not exist.

2. Define Functional Dependency. List and explain various Functional Dependencies.

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.

1. 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.

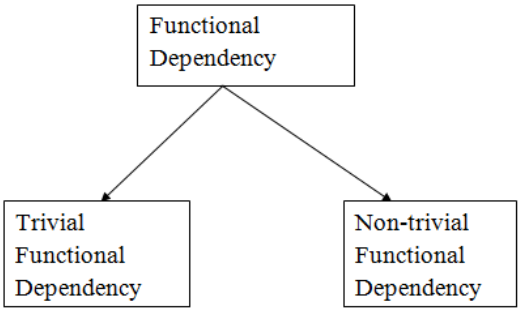
Here Emp_Id attribute can uniquely identify the Emp_Name attribute of employee table because if we know the Emp_Id, we can tell that employee name associated with it.

Functional dependency can be written as:

1. 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:

1. Consider a table with two columns Employee_Id and Employee_Name.
2. {Employee_id, Employee_Name} → Employee_Id is a trivial functional dependency as
3. Employee_Id is a subset of {Employee_Id, Employee_Name}.
4. 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:

1. ID → Name,
2. Name → DOB

3. Define normalization. Interpret the data by applying various normalization techniques such as 1Normal Form (1NF), 2NF and 3NF to reduce redundancy in database tables.

Normalization

* The process of minimizing "Redundancy"
* It removes → Insertion Anomaly
  → Updation Anomaly
  → Deletion Anomaly

Def :- "Normalization" divides the larger table into smaller tables and links them using relationships.

Example :-

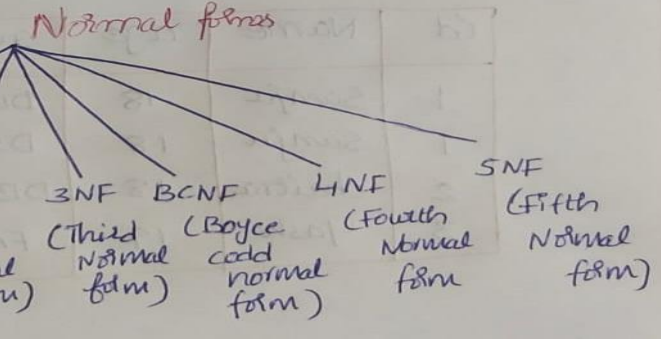
id	Name	age	Branch	HOD
1	Vaishu	18	AIML	TK
2	Vinod	18	AIML	TK
3	Chandu	19	AIML	TK

id	Name	age	branch-id
1	Vaishu	18	101
2	Vinod	18	101
3	Chandu	19	101

branchid	branch	HOD
101	AIML	TK

1) Insertion Anomaly :-
   1) Repetition of data
   2) If the branch is not alloted it has to be put as "NULL".
2) Updation Anomaly :-
   1) If HOD is changed we need to change in all HOD columns.
   2) After decomposition we need to change in one place.
3) Deletion Anomaly :-
   1) If 2 different informations are kept together
   eg:- Student records like Student id, Branch
   If we delete Student-id the Branch will also delete.

Types of Normal forms :-



First Normal Form (1NF) :-

Rules for 1NF :-

① Single valued attribute (Atomic values). Every cell having only one value.
② Attribute domain should not change
③ Unique name for attributes (or) columns.
④ Order doesn't matter.

id	Name	age	Subject
1	Sanju	18	DBMS, DSP
2	chaitra	18	DBMS
3	yash	19	FAM

⇩

id	Name	age	subject
1	Sanju	18	DBMS
1	Sanju	18	DSP
2	chaitra	18	DBMS
3	Yash	19	FAM

Second Normal Form (2NF) :-

Rules for 2NF :-

① The table should be in 1-NF
② No partial dependency

Dependency = Functional dependency = All columns are dependent on one "primary key"

Partial dependency :- Partial dependency happens if there are two or more primary keys in one table. That keys are called as "candidate key" partial dependendency will occur any one part of a candidate key.

Functional dependency :-

Any attribute depends only on a part of a "candidate key".

Example :- Consider 3 tables as student, subject score.

Student

Stud-id	studname	address

Subject

Sub-id	Subname

Score

Score id	Stud-id	Sub-id	Marks	Teacher

After performing 2-NF the tables becomes as follows:

Student

Stud-id	Stud-name	address

Subject

Sub-id	Sub-name	Teacher

Score

candidate key

Score-id	Stud-id	Sub-id	Marks	~~Teacher~~

Remove this col name and add in subject table

Third Normal Form (3NF):-

A Relation is said to be in 3NF, if and only if it follows the following rules.

1. Relation should be in 2NF.
2. It should not contain any "transtive dependency".

Transtive Dependency:-

It is a type of fuctional dependency which has been formed indirectly from two fuctional dependencies.

Consider 3 attributes named as A, B, C

$A \rightarrow B$

$B \rightarrow C$

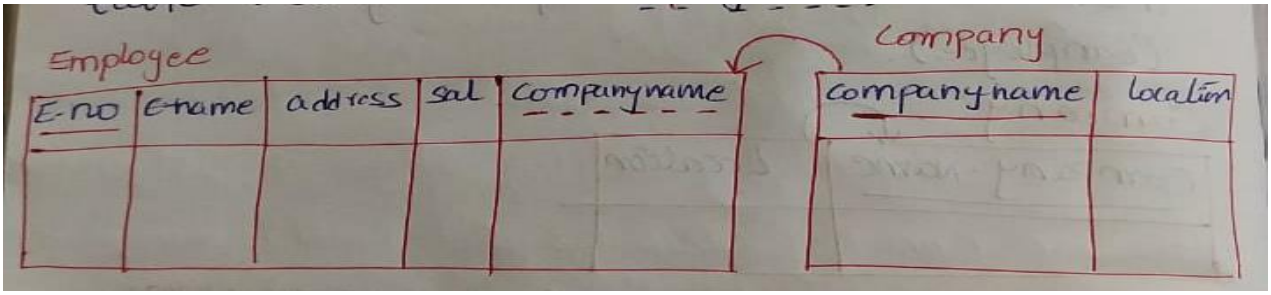
$A \rightarrow C$ (A in directly determines C)

The above relations are called as "transtive dependency"

Transtive Dependency is also defined as "A dependency between non-key attributes. That should be removed in 3NF.

Example:- Employee

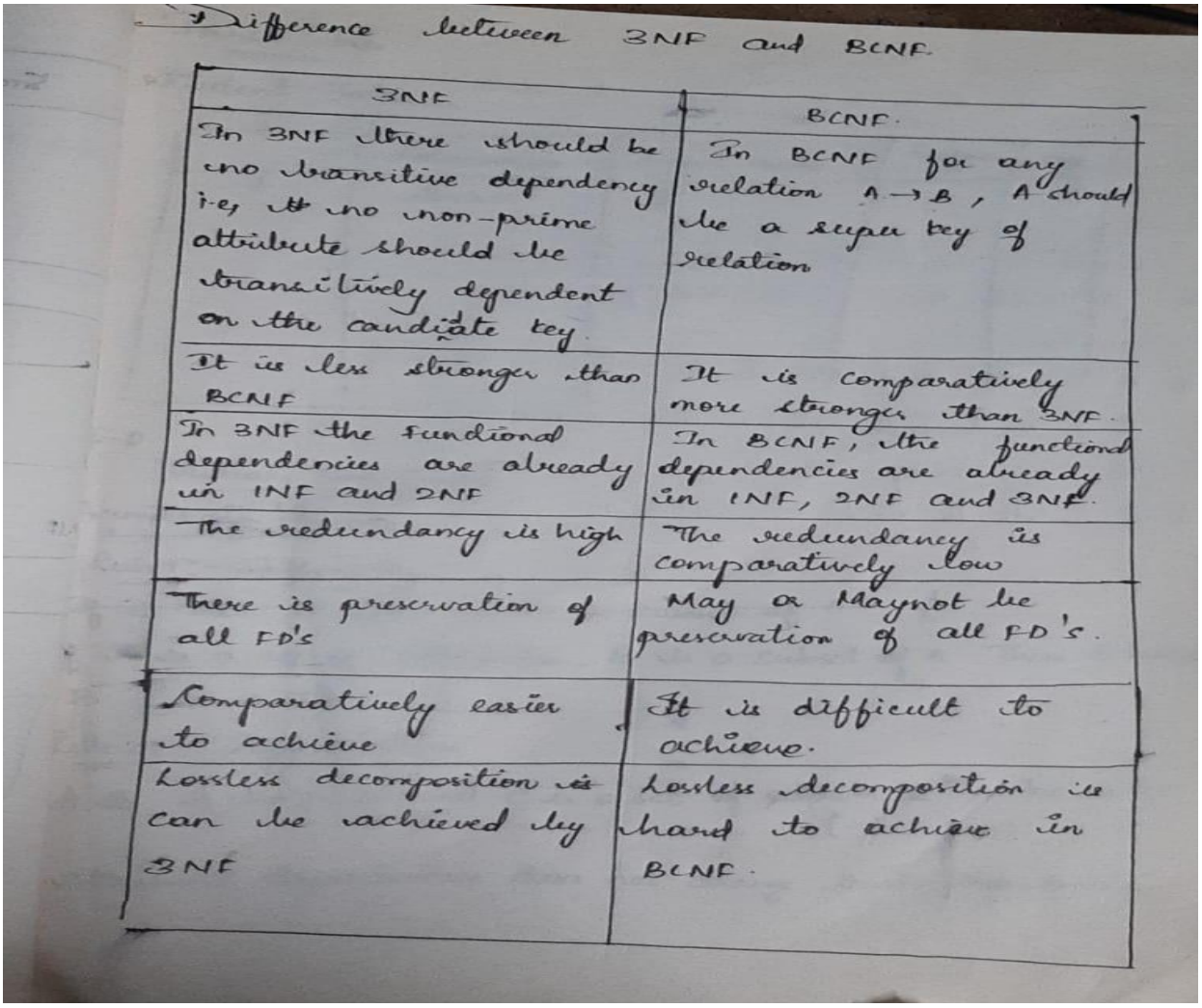
Emp-no	E-name	Address	Salary	Company name	Loc
101	A	AP	40,000	TCS	Hyd
102	B	TS	42,000	Amazon	Banglore
103	C	AP	48,000	Wipro	Hyd
104	D	KA	50,000	TCS	Hyd



4**. What is the relationship exists between 3Normal Form (3NF) and Boyce Codd Normal Form (BCNF). Distinguish between 3NF and BCNF. Consider any one relation schema which is 2NF and convert into BCNF.**

Ans:

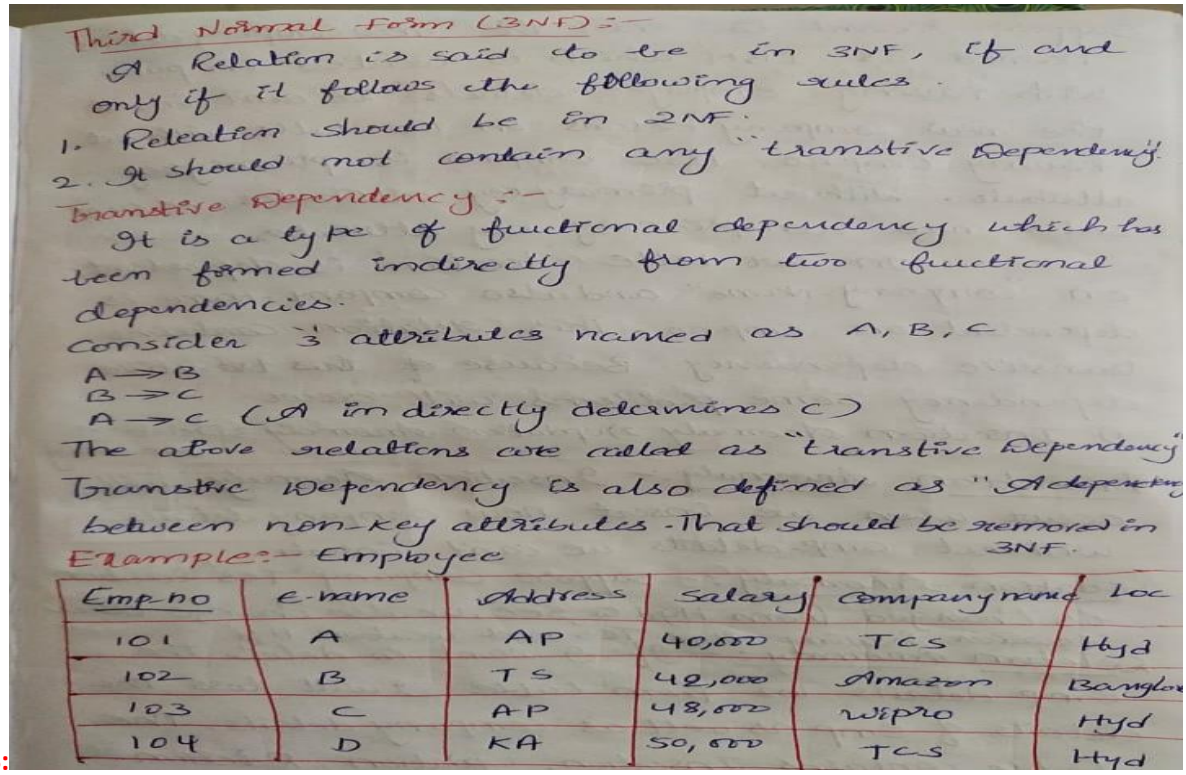
Boyce and Codd Normal Form is a **higher version of the Third Normal form. This form deals with certain type of anomaly that is not handled by 3NF**. A 3NF table which does not have multiple overlapping candidate keys is said to be in BCNF.



Difference between 3NF and BCNF

3NF	BCNF
In 3NF there should be no transitive dependency i.e. if no non-prime attribute should be transitively dependent on the candidate key.	In BCNF for any relation A → B, A should be a super key of relation
It is less stronger than BCNF	It is comparatively more stronger than 3NF.
In 3NF the functional dependencies are already in 1NF and 2NF	In BCNF, the functional dependencies are already in 1NF, 2NF and 3NF.
The redundancy is high	The redundancy is comparatively low
There is preservation of all FD's	May or Maynot be preservation of all FD's.
Comparatively easier to achieve	It is difficult to achieve.
Lossless decomposition is can be achieved by 3NF	lossless decomposition is hard to achieve in BCNF.

ANS:



Third Normal Form (3NF):-

A Relation is said to be in 3NF, if and only if it follows the following rules.

1. Relation should be in 2NF.
2. It should not contain any "transitive Dependency"

Transitive Dependency :-

It is a type of functional dependency which has been formed indirectly from two functional dependencies.

Consider 3 attributes named as A, B, C

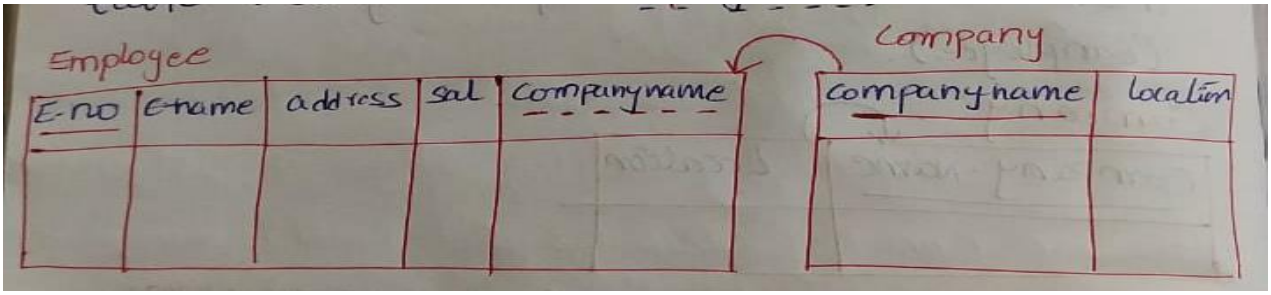
A → B
B → C
A → C (A in directly determines C)

The above relations are called as "transitive Dependency"

Transitive Dependency is also defined as "A dependency between non-key attributes. That should be removed in 3NF.

Example:- Employee

Emp-no	E-name	Address	Salary	Company name	Loc
101	A	AP	40,000	TCS	Hyd
102	B	TS	42,000	Amazon	Banglo
103	C	AP	48,000	Wipro	Hyd
104	D	KA	50,000	TCS	Hyd



Employee

E-no	Ename	address	sal	companyname

Company

companyname	location

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(SK-is a combination of col's that uniquely identifies any row in a table.
- E.F CODD and Raymond F-Boyce developed BCNF

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

EMPLOYEE table:

In the above table Functional dependencies are as follows:

1. EMP_ID → EMP_COUNTRY

2. 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

2NF (Second Normal Form):

Employees Table:

empNum	lastName	firstName
1001	Andrews	Jack
1002	Schwatz	Mike
1009	Beker	Harry
1007	Harvey	Parker
1007	Harvey	Parker

Departments Table:

deptNum	deptName	deptCity	deptCountry
1	Accounts	New York	United States
2	Technology	New York	United States
3	HR	Berlin	Germany
4	Admin	London	United Kingdom

Empdept table:

empDeptID	empNum	deptNum
1	1001	1
2	1002	2
3	1009	3
4	1007	4
5	1007	3

Boyce-Codd Normal Form (3.5 Normal Form):

Employees Table:

empNum	firstName	empCity	deptNum
1001	Jack	New York	D1
1001	Jack	New York	D2
1002	Harry	Berlin	D1
1007	Parker	London	D3
1007	Parker	London	D4

Department Table:

deptNum	deptName	deptHead
D1	Accounts	Raymond
D2	Technology	Donald
D1	Accounts	Samara
D3	HR	Elizabeth
D4	Infrastructure	Tom

5. Define Decomposition. Describe various types of Decomposition with suitable example.

ANS: Decomposition

A functional **decomposition** is the process of breaking down the functions of an organization into progressively greater (finer and finer) levels of detail.
In decomposition, one function is described in greater detail by a set of other supporting functions.
The decomposition of a relation scheme R consists of replacing the relation schema by two or more relation schemas that each contain a subset of the attributes of R and together include all attributes in R.
Decomposition helps in eliminating some of the problems of bad design such as redundancy, inconsistencies and anomalies.
There are two types of decomposition :

1. Lossy Decomposition
2. Lossless Join Decomposition

Lossy Decomposition :

"The decompositio of relation R into R1 and R2 is **lossy** when the join of R1 and R2 does not yield the same relation as in R."
One of the disadvantages of decomposition into two or more relational schemes (or tables) is that some information is lost during retrieval of original relation or table.
Consider that we have table STUDENT with three attribute roll_no , sname and department.

STUDENT:

Roll_no	Sname	Dept
111	parimal	COMPUTER
222	parimal	ELECTRICAL

This relation is decomposed into two relation no_name and name_dept :

No_name:

Roll_no	Sname
111	parimal
222	parimal

Name_dept :

Sname	Dept
parimal	COMPUTER
parimal	ELECTRICAL

In lossy decomposition ,spurious tuples are generated when a natural join is applied to the relations in the decomposition.

stu_joined :

Roll_no	Sname	Dept
111	parimal	COMPUTER
111	parimal	ELECTRICAL
222	parimal	COMPUTER
222	parimal	ELECTRICAL

The above decomposition is a bad decomposition or Lossy decomposition.

Lossless Join Decomposition :

"The decompositio of relation R into R1 and R2 is **lossless** when the join of R1 and R2 yield the same relation as in R."

A relational table is decomposed (or factored) into two or more smaller tables, in such a way that the designer can capture the precise content of the original table by joining the decomposed parts. This is called lossless-join (or non-additive join) decomposition.

This is also refferd as non-additive decomposition.

The lossless-join decomposition is always defined with respect to a specific set F of dependencies.

Consider that we have table STUDENT with three attribute roll_no , sname and department.

STUDENT :

Roll_no	Sname	Dept
111	parimal	COMPUTER
222	parimal	ELECTRICAL

This relation is decomposed into two relation Stu_name and Stu_dept :

Stu_name:

Roll_no	Sname
111	parimal
222	parimal

Stu_dept :

Roll_no	Dept
111	COMPUTER
222	ELECTRICAL

Now ,when these two relations are joined on the comman column 'roll_no' ,the resultant relation will look like stu_joined.

stu_joined :

Roll_no	Sname	Dept
111	parimal	COMPUTER
222	parimal	ELECTRICAL

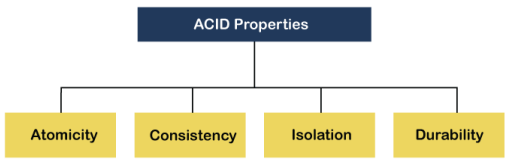
In lossless decomposition, no any spurious tuples are generated when a natural joined is applied to the relations in the decomposition.

UNIT-V

1. List out the ACID properties. Describe the transaction states with a neat diagram.

ACID Properties in DBMS

DBMS is the management of data that should remain integrated when any changes are done in it. It is because if the integrity of the data is affected, whole data will get disturbed and corrupted. Therefore, to maintain the integrity of the data, there are four properties described in the database management system, which are known as the **ACID** properties. The ACID properties are meant for the transaction that goes through a different group of tasks, and there we come to see the role of the ACID properties.



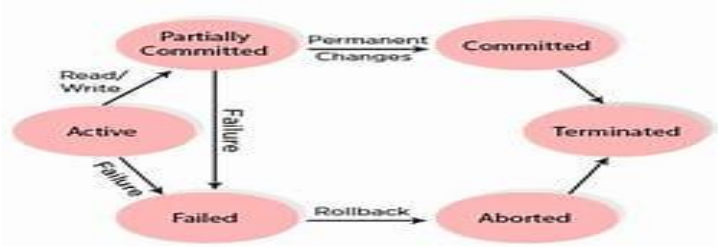
1) Atomicity: The term atomicity defines that the data remains atomic. It means if any operation is performed on the data, either it should be performed or executed completely or should not be executed at all. It further means that the operation should not break in between or execute partially. In the case of executing operations on the transaction, the operation should be completely executed and not partially.

2) Consistency: The word **consistency** means that the value should remain preserved always. In DBMS, the integrity of the data should be maintained, which means if a change in the database is made, it should remain preserved always. In the case of transactions, the integrity of the data is very essential so that the database remains consistent before and after the transaction. The data should always be correct.

4) Isolation: The term 'isolation' means separation. In DBMS, Isolation is the property of a database where no data should affect the other one and may occur concurrently. In short, the operation on one database should begin when the operation on the first database gets complete. It means if two operations are being performed on two different databases, they may not affect the value of one another. In the case of transactions, when two or more transactions occur simultaneously, the consistency should remain maintained. Any changes that occur in any particular transaction will not be seen by other transactions until the change is not committed in the memory.

4) Durability: Durability ensures the permanency of something. In DBMS, the term durability ensures that the data after the successful execution of the operation becomes permanent in the database. The durability of the data should be so perfect that even if the system fails or leads to a crash, the database still survives. However, if gets lost, it becomes the responsibility of the recovery manager for ensuring the durability of the database. For committing the values, the COMMIT command must be used every time we make changes.

States of transaction



Active State: Active state is the initial state in each transaction life cycle. ...

Partially Committed State: This state is also known as the execution phase. ...
Committed State: Some transactions accomplished the implementation adequately, and because all the modifications made in the local memory through the partially committed state are forever saved in the database.

Failed State: A transaction enters this state when it is not able to proceed in a normal manner, either due to its internal error or due to some system failure.

Aborted State: This transaction state appears after the transaction has been rolled back, and the database has been reconstructed to its state before the beginning of the transaction.

Terminated State: A transaction is said to be terminated when it has either committed successfully or has aborted after a failure.

2. Describe Lock based concurrency control with suitable example.

Ans:

Concurrency:- Accessing same data by number of users at the same time.Transaction May success/May not be success.
To prevent this problems Locking mechanisms are used.

Locking Methods:A procedure used to control concurrent access to data when one transaction is accessing the database.
Locking mechanisms:----

Binary Lock:-

it has 2 states/values associated with each data item. These values are:

1.Locked-1

2.Unlocked-0

if data is locked then it can't be accessed by other transactions.

These locks are applied and removing using Lock() & Unlock() operation respectively.

- In binary locks,at a particular point in time,only one transaction can hold a lock on the data item.
- No other transaction will be able to access the same data concurrently.
- Hence, binary locks are very simple to apply but are not used pratically…..

Shared lock:-

If a transaction has shared lock on a data item, it can read the item, but can't update/add the data in it.

It is represented with "S"(lock-s)

It is also known as "Read Only Lock"

Exclusive Lock:-

If a transaction has Exclusive lock on a data item can be performed both read as well as write operations.

It is represented with "X"(lock-X)

In this multiple transactions don't modify the same data simultaneously.

Two Phase Locking (2PL):-

it is a concurrency control technique.

*Both locks & Unlocks

2PL Divides into 2 phases

1.Growing phase

2.Shrinking phase

Growing Phase:

locks are obtained but Not released.

Shrinking Phase:

locks are obtained but no new locks are acquired.

holding lock un-necessarily

*locking too early

*penalty to other transactions

These reduce the concurrency in 2PL

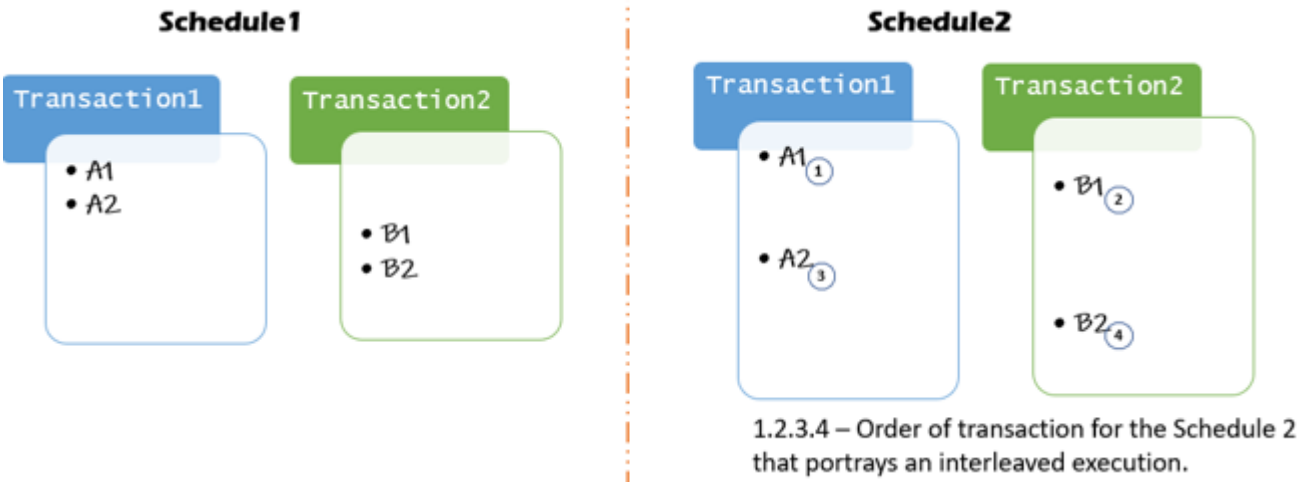
3. Define Serializability. Explain Conflict Serializability and View Serializability with an example.

Ans:

Serial schedule both by definition and execution means that the transactions bestowed upon it will take place serially, that is, one after the other. This leaves no place for inconsistency within the database. But, when a set of transactions are scheduled non-serially, they are interleaved leading to the problem of concurrency within the database. Non-serial schedules do not wait for one transaction to complete for the other one to begin. Serializability in DBMS decides if an interleaved non-serial schedule is serializable or not.

Example of Serializability

Consider 2 schedules, Schedule1 and Schedule2:



1.2.3.4 – Order of transaction for the Schedule 2 that portrays an interleaved execution.

- Schedule1 is a serial schedule consisting of Transaction1 and Transaction2 wherein the operations on data item A (A1 and A2) are performed first and later the operations on data item B (B1 and B2) are carried out serially.
- Schedule2 is a non-serial schedule consisting of Transaction1 and Transaction2 wherein the operations are interleaved.

Explanation: In the given scenario, schedule2 is serializable if the output obtained from both Schedule2 and Schedule1 are equivalent to one another In a nutshell, a transaction within a given non-serial schedule is serializable if its outcome is equivalent to the outcome of the same transaction when executed serially.

types of Serializability

. Conflict Equivalent Schedule

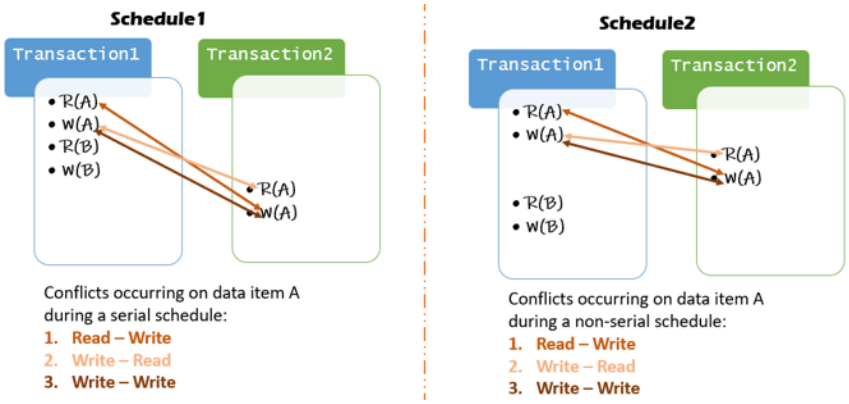
When either of a conflict operation such as Read-Write or Write-Read or Write-Write is implemented on the same data item at the same time within different transactions then the schedule holding such transactions is said to be a conflict schedule. The prerequisites for such conflict schedule are:

1. The conflict operations are to be implemented on the same data item.
2. The conflict operations (RW, WR, WW) must take place within different transactions.
3. At least one of the conflict operations must be the write operation.
4. Two Read operations will not create any conflict.

Two schedules (one being serial schedule and another being non-serial) are said to be conflict serializable if the conflict operations in both the schedules are executed in the same order.

Example:

Consider 2 schedules, Schedule1 and Schedule2,

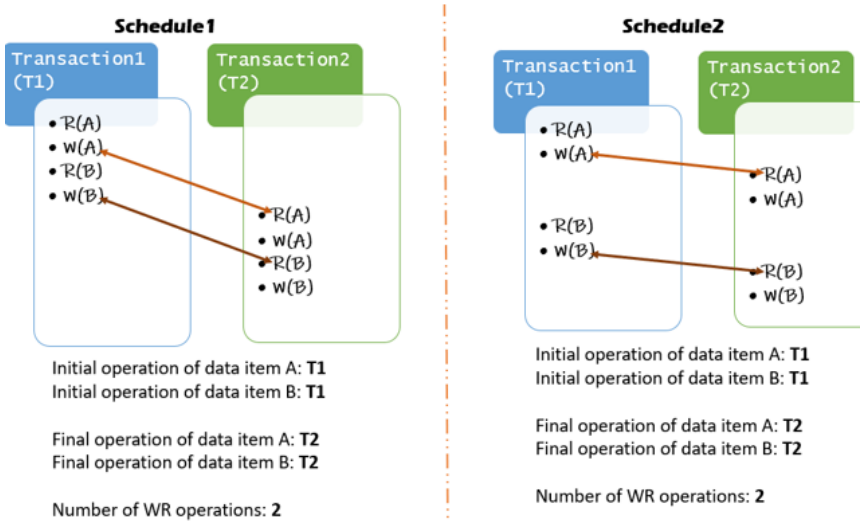


Schedule2 (a non-serial schedule) is considered to be conflict serializable when its conflict operations are the same as that of Shedule1 (a serial schedule).

View Equivalent Schedule

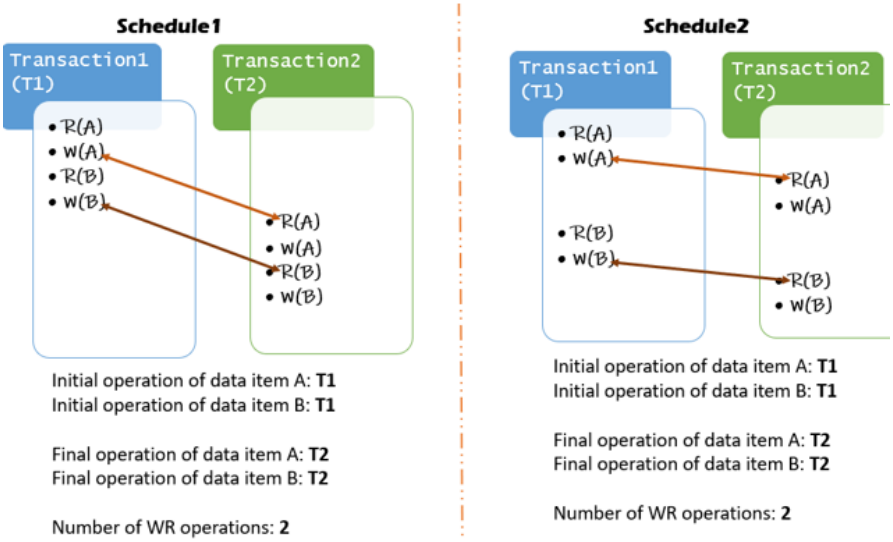
Two schedules (one being serial schedule and another being non-serial) are said to be view serializable if they satisfy the rules for being view equivalent to one another.

The rules to be upheld are:



1. Initial values of the data items involved within a schedule must be the same.
2. Final values of the data items involved within a schedule must be the same.
3. The number of WR operations performed must be equivalent for the schedules involved.

Example:



The (non-serial) Schedule2 is considered as a view equivalent of the (serial) Schedule1, when the 3 rules of view serializability are satisfied. For the example shown above,

- The Initial transaction of read operation on the data items A and B both begin at T1
- The Final transaction of write operations on the data items A and B both end at T2
- The number of updates from write-read operations are 2 in both the cases

4. Describe concurrency control with Time stamp based locking protocols.

Ans:

Lock-based protocols manage the order between the conflicting pairs among transactions at the time of execution, whereas timestamp-based protocols start working as soon as a transaction is created.

The timestamp-ordering protocol ensures serializability among transactions in their conflicting read and write operations. This is the responsibility of the protocol system that the conflicting pair of tasks should be executed according to the timestamp values of the transactions.

- The timestamp of transaction T_i is denoted as $TS(T_i)$.
- Read time-stamp of data-item X is denoted by R-timestamp(X).
- Write time-stamp of data-item X is denoted by W-timestamp(X).

Timestamp ordering protocol works as follows −

- **If a transaction Ti issues a read(X) operation −**
  - If TS(Ti) < W-timestamp(X)
    - Operation rejected.
  - If TS(Ti) >= W-timestamp(X)
    - Operation executed.
  - All data-item timestamps updated.
- **If a transaction Ti issues a write(X) operation −**
  - If TS(Ti) < R-timestamp(X)
    - Operation rejected.

- If TS(Ti) < W-timestamp(X)
  - Operation rejected and Ti rolled back.
- Otherwise, operation executed.

T1	T2
10	20
R(A)	
	W(A)

RTS = 10 RTS>TS(Ti) -> 10>20 -.> FALSE

WTS= 20 WTS > TS(Ti) -> 20>20 -> FALSE

TS(Ti)=20 THE OPERATION EXECUTED

5. Explain concurrency control with optimistic methods

All data items are updated at the end of the transaction, at the end, if any data item is found inconsistent with respect to the value in, then the transaction is rolled back.

Check for conflicts at the end of the transaction. No checking while the transaction is executing. Checks are all made at once, so low transaction execution overhead. Updates are not applied until end-transaction. They are applied to local copies in a transaction space.

Phases

The optimistic concurrency control has three phases, which are explained below −

Read Phase

Various data items are read and stored in temporary variables (local copies). All operations are performed in these variables without updating the database.

Validation Phase

All concurrent data items are checked to ensure serializability will not be validated if the transaction updates are actually applied to the database. Any changes in the value cause the transaction rollback. The transaction timestamps are used and the write-sets and read-sets are maintained.

To check that transaction A does not interfere with transaction B the following must hold −

- TransB completes its write phase before TransA starts the read phase.
- TransA starts its write phase after TransB completes its write phase, and the read set of TransA has no items in common with the write set of TransB.
- Both the read set and write set of TransA have no items in common with the write set of TransB and TransB completes its read before TransA completes its read Phase.

Write Phase

The transaction updates applied to the database if the validation is successful. Otherwise, updates are discarded and transactions are aborted and restarted. It does not use any locks hence deadlock free, however starvation problems of data items may occur.

Problem

S: W1(X), r2(Y), r1(Y), r2(X).

T1 -3

T2 – 4

Check whether timestamp ordering protocols allow schedule S.

Solution

Initially for a data-item X, RTS(X)=0, WTS(X)=0

Initially for a data-item Y, RTS(Y)=0, WTS(Y)=0


```
For W1(X) : TS(Ti)<RTS(X) i.e.
   TS(T1)<RTS(X)
TS(T1)<WTS(X)
3<0 (FALSE)
```

=>goto else and perform write operation w1(X) and WTS(X)=3

```
For r2(Y): TS(T2)<WTS(Y)
   4<0 (FALSE)
```

=>goto else and perform read operation r2(Y) and RTS(Y)=4

```
For r1(Y) :TS(T1)<WTS(Y)
   3<0 (FALSE)
```

=>goto else and perform read operation r1(Y).

```
For r2(X) : TS(T2)<WTS(X)
   4<3 (FALSE)
```

=>goto else and perform read operation r2(X) and RTS(X)=4