Unit III

3 Basics of SQL

3.1 DDL, DML, DCL Structure

Q.1 What is SQL ? What are the characteristics and advantages of SQL ?

Ans. : SQL : SQL stands for Structured Query Language.

- SQL makes use of query. A Query is a set of instruction given to the database management system. It tells any database what information we would like to get from the database.

Characteristics and Advantages :

1) SQL is a standard computer language for creating and manipulating databases.
2) SQL is very simple and easy to learn.
3) SQL allows the users to create, update, delete and retrieve data from the database.
4) SQL is used to create view, stored procedures and functions in a database.
5) SQL allows the users to set the permissions on the tables, procedures and views in the database.

Q.2 Explain DDL, DML and DCL structure.

Ans. : Data Definition Language

Sr. No.	Command	Purpose
1.	CREATE	This command is used to create database, tables, views or any other database objects.

2.	ALTER	It modifies the existing tables.
3.	DROP	This command deletes complete table, view or any other database object.

Data Manipulation Language

Sr. No.	Command	Purpose
1.	SELECT	This command is used to retrieve either all or desired records from one or more tables.
2.	INSERT	For inserting the records in the table, this command is used.
3.	UPDATE	For updating one or more fields of the table, this command is used.
4.	DELETE	This command is used for deleting the desired record.

Data Control Language

Sr. No.	Command	Purpose
1.	GRANT	This command is used to give access rights or privileges to the database.
2.	INVOKE	The revoke command removes user access rights or privileges to the database objects.

Q.3 What is the difference between DDL and DML ?

Ans. :

Difference between DDL and DML

DDL	DML
DDL stands for Data Definition Language.	DML stands for Data Manipulation Language.
DDL commands are used to define database structure.	DML commands are used for managing data within the database.
It works on whole table.	It works on one or more rows.
It cannot be classified further.	It can be classified as - procedural and non-procedural language.

3.2 Creation and Alteration

Q.4 Explain creation of Table using SQL.

Ans. :

- A database can be considered as a container for tables and a table is a grid with rows and columns to hold data.
- Individual statements in SQL are called **queries.**
- We can execute SQL queries for various tasks such as creation of tables, insertion of data into the tables, deletion of record from table and so on.

In this section we will discuss how to create a table.

Step 1 : We normally create a database using following SQL statement

Syntax

```
CREATE DATABASE database_name;
```

Example

```
CREATE DATABASE Person_DB
```

Step 2 : The table can be created inside the database as follows -

```
CREATE TABLE table_name (
    col1_name datatype,
    col2_name datatype,
    ...
    coln_name datatype
);
```

Example

```
CREATE TAELE person_details (
        AadharNo int,
        FirstName VARCHAR(20),
        MiddleName VARCHAR(20),
        LastName VARCHAR(20),
        Address VARCHAR(30),
        City VARCHAR(10)
)
```

Q.5 Give the syntax and example of insertion of data into the table using SQL.

Ans. :

- We can insert data into the table using INSERT statement.

Syntax

```
INSERT INTO table_name (col1,col2,....,coln)
VALUES (value1,value2,....,valuen)
```

Example

```
INSERT INTO person_details (AadharNo, FirstName, MiddleName,
LastName, Address, City)
VALUES (111,'AAA','BBB','CCC','M.G. Road','Pune')
```

The above query will result into –

AadharNo	FirstName	MiddleName	LastName	Address	City
111	AAA	BBB	CCC	M.G. Road	Pune

Q.6 How to delete a record from a table ? Explain it with SQL syntax and example.

Ans. :

- We can delete one or more records based on some condition. The syntax is as follows -

Syntax

```
DELETE FROM table_name WHERE condition;
```

Example

```
DELETE FROM person_details
WHERE AadharNo=333
```


We can delete all the records from table. But in this deletion, all the records get deleted without deleting table. For that purpose the SQL statement will be

```
DELETE FROM person_details;
```

Q.7 Explain the use of DISTINCT keyword in SQL.

Ans. :

- The keyword DISTINCT is used along with the SELECT statements.
- It is used to obtain unique values from the table. This query does not allow duplication of element.

Syntax :

```
SELECT DISTICT Column-name FROM table-name;
```

Example :

Consider following database table **Student**

Roll No	Name	City
1	Ankita	Pune
2.	Rohit	Hyderabad
3.	Prajkta	Chennai
4.	Sunil	Pune
5.	Sharda	Chennai

SQL Statement

```
SELECT DISTICT City
FROM Student;
```

This will result into

Pune
Hyderabad
Chennai

3.3 : Defining Constraints

Q.8 How to apply primary key constraints ? Explain it with suitable example.

Ans. :

(1) Primary key : The primary key constraint is defined to uniquely identify the records from the table.

The primary key must contain unique values. Hence database designer should choose primary key very carefully.

For example

Consider that we have to create a **perons_details** table with AdharNo, FirstName, MiddleName, LastName, Address and City.

Now making AdharNo as a primary key is helpful here as using this field it becomes easy to identify the records correctly.

The result will be

```
CREATE TABLE person_details (
AadharNo int,
FirstName VARCHAR(20),
MiddleName VARCHAR(20),
LastName VARCHAR(20),
Address VARCHAR(30),
City VARCHAR(10),
PRIMARY KEY(AadharNo)
);
```

We can create a composite key as a primary key using CONSTRAINT keyword. For example

```
CREATE TABLE person_details (
AadharNo int NOT NULL,
FirstName VARCHAR(20),
MiddleName VARCHAR(20),
LastName VARCHAR(20) NOT NULL,
```



```
Address VARCHAR(30),
City VARCHAR(10),
CONSTRAINT PK_person_details PRIMARY
KEY(AadharNo, LastName)
);
```

Q.9 Explain the use of IN operator with suitable example.

Ans. :

The IN operator is just similar to OR operator.

It allows to specify multiple values in WHERE clause.

Syntax

```
SELECT col1,col2,...
FROM table_name
WHERE column-name IN (value1, value2, ...);
```

Example

Consider following table

Employee

empID	empName	Salary	DeptID
1	AAA	1000	D101
2	BBB	2000	D102
3	CCC	3000	D103
4	DDD	4000	D104
5	EEE	5000	D105

```
SELECT * FROM Employee
WHERE empID IN (1, 3);
```

The result will be

empID	empName	Salary	DeptID
1	AAA	1000	D101
2	BBB	2000	D102
3	CCC	3000	D103

SELECT col1, col2, ... coln
from table-name
ORDER BY col1, col2, ... ASC|DESC

3.4 Use of Group By, Having and Order By Clause

Q.10 Explain Group By, Order By and Having clause with suitable example.

Ans. :

(1) Order By

- For getting the sorted records in the table we use **ORDER BY** command.
- The ORDER BY keyword sorts the records in ascending order by default.

Syntax

```
SELECT col1, col2,...,coln
FROM table_name
ORDER BY col1,col2,... ASC|DESC
```

Here ASC is for ascending order display and DESC is for descending order display.

```
SELECT *
FROM person_details
ORDER BY AadharNo DESC;
```

(2) Group By

- The GROUP BY clause is a SQL command that is used to group rows that have the same values.
- The GROUP BY clause is used in the SELECT statement.
- This query returns a single row for every grouped item.
- **Syntax :**

SELE SELECT column-names
from table-name
where condition
GROUP by column-names

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
```

Query : *Find the total marks of each student in each city*

```
SELECT SUM(marks), city
FROM Student
GROUP BY city
```


(3) Having

- HAVING filters records that work on summarized GROUP BY results.
- HAVING applies to summarized group records, whereas WHERE applies to individual records.
- Only the groups that meet the HAVING criteria will be returned.
- HAVING requires that a GROUP BY clause is present.
- **Syntax :**

```
SELECT column-names
FROM table-name
WHERE condition
GROUP BY column-names
HAVING condition
```

- **Example**

```
SELECT SUM(marks), city
FROM Student
GROUP BY city
HAVING city IN('Pune','Mumbai')
```

3.5 Schema Change Statements

Q.11 Describe DROP TABLE command of SQL with both the options CASCADE and RESTRICT. ☞ **[SPPU : Nov.-18, Marks 5]**

Ans. :

- The DROP command is used to remove the object (table, domains and constraints) from the database. There are two options for the DROP command - CASCADE and RESTRICT.
- To use the RESTRICT option, the user must first individually drop each element in the schema, then drop the schema itself. That means, the schema is dropped only if it has no elements in it, otherwise the DROP command can not be executed.

- Otherwise to remove completely some database schema CASCADE option is chosen. For example – to remove the Student_database

```
DROP SCHEMA Student_database CASCADE;
```

- If the table is to be deleted then the SQL command would be -

```
DROP TABLE Student CASCADE;
```

- The DROP TABLE command not only deletes all the records in the table if successful, but also removes the table definition from the catalog. If it is desired to delete only the records but to leave the table definition for future use, then the DELETE command. For example -
- The following SQL statement deletes all rows in the "Students" table, without deleting the table :

```
DELETE FROM Students;
```

3.6 Aggregate Functions

Q.12 What is aggregate function ? Explain with suitable examples.

Ans. : • An aggregate function allows you to perform a calculation on a set of values to return a single scalar value.

- SQL offers five built-in aggregate functions :
  1. Average : avg
  2. Minimum : min
  3. Maximum : max
  4. Total : sum
  5. Count :
- Syntax of all the Aggregate Functions

```
AVG( [ DISTINCT | ALL ] expression)
COUNT(*)
COUNT( [ DISTINCT | ALL ] expression )
MAX( [ DISTINCT | ALL ] expression)
MIN( [ DISTINCT | ALL ] expression)
```



```
SUM( [ DISTINCT | ALL ] expression)
```

- The avg function is used to compute average value. For example - To compute average marks of the students we can use

```
SQL Statement
SELECT AVG(marks)
FROM Students
```

- The **Count** function is used to count the total number of values in the specified field. It works on both numeric and non-numeric data type. COUNT (*) is a special implementation of the COUNT function that returns the count of all the rows in a specified table. COUNT (*) also considers Nulls and duplicates. For example Consider following table

Test

id	value
11	100
22	200
33	300
NULL	400

SQL Statement

```
SELECT COUNT(*)
FROM Test
Output
4
```

- The **min** function is used to get the minimum value from the specified column. For example - Consider the above created **Test** table

```
SQL Statement
SELECT Min(value)
FROM Test
Output
100
```

- The max function is used to get the maximum value from the specified column. For example - Consider the above created **Test** table

```
SQL Statement
SELECT Max(value)
FROM Test
Output
400
```

- The sum function is used to get total sum value from the specified column. For example - Consider the above created **Test** table

```
SQL Statement
SELECT sum(value)
FROM Test
Output
1000
```

Q.13 Consider, the following database,
Student(RollNo, Name, Address)
Subject(Sub_code, Sub_Name)
Marks (Roll_no, Sub_code, Marks)
Write following queries in SQL.
Find average marks of each student, along with the name of Student. [SPPU : Nov.-17, (End Sem), Marks 2

Solution:

```
SELECT Name, AVG(Marks)
FROM Student,Marks
WHERE Student.Roll_No=Marks.Roll_No
```

3.7 Built-in Functions

Q.14 List and explain any five mathematical functions.

Ans. : Mathematical Functions

Function	Value Returned
ABS (m)	Absolute value of m

MOD (m, n)	Remainder of m divided by n
POWER (m, n)	m raised to the nth power
ROUND (m [, n])	m rounded to the nth decimal place
TRUNC (m [, n])	m truncated to the nth decimal place
SIN (n)	sine (n)
COS (n)	cosine (n)
TAN (n)	tan (n)
SQRT (n)	positive square root of n

Q.15 List and explain any five string functions.

Ans. : String Functions

Function	Value Returned
INITCAP (s)	First letter of each word is changed to uppercase and all other letters are in lower case.
LOWER (s)	All letters are changed to lowercase.
UPPER (s)	All letters are changed to uppercase.
CONCAT (s1, s2)	Concatenation of s1 and s2. Equivalent to s1 || s2.
LTRIM (s [, set])	Returns s with characters removed up to the first character not in set; defaults to space.
RTRIM (s [, set])	Returns s with final characters removed after the last character not in set; defaults to space.
REPLACE (s, search_s [, replace_s])	Returns s with every occurrence of search_s in s replaced by replace_s; default removes search_s.
LENGTH (s)	Returns the number of characters in s.

3.8 Set Operations

Q.16 Explain various set operations using SQL.

Ans. :

1) Union : To use this UNION clause, each SELECT statement must have

i) The same number of columns selected

ii) The same number of column expressions

iii) The same data type and

iv) Have them in the same order

This clause is used to **combine two tables** using UNION operator. It replaces the OR operator in the query. The union operator **eliminates duplicate** while the union all query will retain the duplicates.

Syntax :

The basic syntax of a UNION clause is as follows -

```
SELECT column1 [, column2 ]
FROM table1 [, table2 ]
[WHERE condition]
UNION
SELECT column1 [, column2 ]
FROM table1 [, table2 ]
[WHERE condition]
```

Here, the given condition could be any given expression based on your requirement.

Example : Find the names of the students who have reserved the 'DBMS' book or 'OS' Book

The query can then be written by considering the Student, Reserve and Book table as


```
SELECT S.sname
FROM Student S, Reserve R, Book B
WHERE S.sid=R.sid AND R.isbn=B.isbn AND B.bname='DBMS'
UNION
SELECT S.sname
FROM Student S, Reserve R, Book B
WHERE S.sid=R.sid AND R.isbn=B.isbn AND B.bname='OS'
```

2) Intersect : The common entries between the two tables can be represented with the help of Intersect operator. It replaces the AND operator in the query.

Syntax :

The basic syntax of a INTERSECT clause is as follows -

```
SELECT column1 [, column2 ]
FROM table1 [, table2 ]
[WHERE condition]
INTERSECT
SELECT column1 [, column2 ]
FROM table1 [, table2 ]
[WHERE condition]
```

Example : Find the students who have reserved both the 'DBMS' book and 'OS' Book

The query can then be written by considering the Student, Reserve and Book table as

```
SELECT S.sid, S.sname
FROM Student S, Reserve R, Book B
WHERE S.sid=R.sid AND R.isbn=B.isbn AND B.bname='DBMS'
INTERSECT
SELECT S.sname
FROM Student S, Reserve R, Book B
WHERE S.sid=R.sid AND R.isbn=B.isbn AND B.bname='OS'
```

3) Except : The EXCEPT clause is used to represent the set-difference in the query. This query is used to represent the entries that are present in one table and not in other.

Syntax :

The basic syntax of a EXCEPT clause is as follows -

```
SELECT column1 [, column2 ]
FROM table1 [, table2 ]
[WHERE condition]
EXCEPT
SELECT column1 [, column2 ]
FROM table1 [, table2 ]
[WHERE condition]
```

Example : Find the students who have reserved both the 'DBMS' book but not reserved 'OS' Book

The query can then be written by considering the Student, Reserve and Book table as

```
SELECT S.sid, S.sname
FROM Student S, Reserve R, Book B
WHERE S.sid=R.sid AND R.isbn=B.isbn AND B.bname='DBMS'
EXCEPT
SELECT S.sname
FROM Student S, Reserve R, Book B
WHERE S.sid=R.sid AND R.isbn=B.isbn AND B.bname='OS'
```

3.9 Nested Queries

Q.17 What is co-related query ? Explain it with suitable example.

Ans. : In co-related nested queries, the output of inner query depends on the row which is being currently executed in outer query. For example To find names of employee who are department id 103

- If we want to find out sname of Student who live in city with cid as 101, it can be done with the help of co-related nested query as :

```
SELECT sname
FROM Student S
WHERE EXISTS
( SELECT *
    FROM Student_City SC
    WHERE S.sid=SC.sid and SC.cid=101
)
```

SELECT E.ename
FROM Employee E
Where EXISTS(SELECT * FROM DepartmentD
WHERE D.id=103 AND D.id=E.id)

- Here For each row of **Student S**, it will find the rows from **Student_City** where S.sid = SC.sid and SC.cid=101.
- If for a **sid** from **Student S,** atleast a row exists in **Student_City** SC with **cid**=101, then inner query will return true and corresponding **sid** will be returned as output.

3.10 Join and its Types

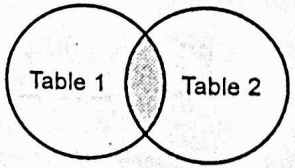
Q.18 Explain join operations with example.

☞ [SPPU : Nov.-19, End Sem, Marks 6, Nov.-17, End Sem, Marks 5]

Ans. :

1) Inner Join :

- The most important and frequently used of the joins is the INNER JOIN. They are also known as an EQUIJOIN.
- The INNER JOIN creates a new result table by combining column values of two tables (Table1 and Table2) based upon the join-predicate.
- The query compares each row of table1 with each row of Table2 to find all pairs of rows which satisfy the join-predicate.
- When the join-predicate is satisfied, column values for each matched pair of rows of A and B are combined into a result row. It can be represented as :



- **Syntax :** The basic syntax of the INNER JOIN is as follows.

```
SELECT Table1.column1, Table2.column2...
```

```
FROM Table1
INNER JOIN Table2
ON Table1.common_field = Table2.common_field;
```

- **Example :** For above given two tables namely Student and City, we can apply inner join. It will return the record that are matching in both tables using the common column cid.

The query will be

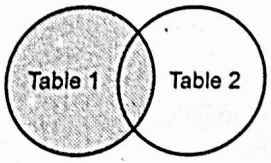
```
SELECT *
FROM Student Inner Join City on Student.cid=City.cid
```

The result will be

sid	cid	sname	cid	cname
1	101	Ram	101	Pune
2	101	Shyam	101	Pune
3	102	Seeta	102	Mumbai

2) Left Join(Outer Join) :

- The SQL LEFT JOIN returns all rows from the left table, even if there are no matches in the right table. This means that if the ON clause matches 0 (zero) records in the right table; the join will still return a row in the result, but with NULL in each column from the right table.
- This means that a left join returns all the values from the left table, plus matched values from the right table or NULL in case of no matching join predicate.
- It can be represented as -



- **Syntax :** The basic syntax of a LEFT JOIN is as follows.

```
SELECT
SELECT Table1.column1, Table2.column2...
```



```
FROM Table1
LEFT JOIN Table2
ON Table1.common_field = Table2.common_field;
```

- **Example :** For above given two tables namely Student and City, we can apply Left join. It will Return all records from the left table, and the matched records from the right table using the common column cid. The query will be

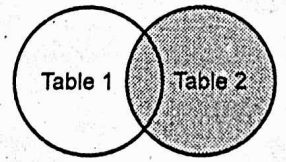
```
SELECT *
FROM Student Left Join City on Student.cid=City.cid
```

The result will be

sid	cid	sname	cid	cname
1	101	Ram	101	Pune
2	101	Shyam	101	Pune
3	102	Seeta	102	Mumbai
4	NULL	Geeta	NULL	NULL

3) Right Join(Outer Join) :

- The SQL RIGHT JOIN returns all rows from the right table, even if there are no matches in the left table.
- This means that if the ON clause matches 0 (zero) records in the left table; the join will still return a row in the result, but with NULL in each column from the left table.
- This means that a right join returns all the values from the right table, plus matched values from the left table or NULL in case of no matching join predicate.
- It can be represented as follows :



- **Syntax :** The basic syntax of a RIGHT JOIN is as follow -

```
SELECT Table1.column1, Table2.column2...
```

```
FROM Table1
RIGHT JOIN Table2
ON Table1.common_field = Table2.common_field;
```

- **Example :** For above given two tables namely Student and City, we can apply Right join. It will return all records from the right table, and the matched records from the left table using the common column cid. The query will be

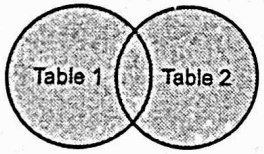
```
SELECT *
FROM Student Right Join City on Student.cid=City.cid
```

The result will be -

sid	cid	sname	cid	cname
1	101	Ram	101	Pune
2	101	Shyam	101	Pune
3	102	Seeta	102	Mumbai
NULL	NULL	NULL	103	Chennai

4) Full Join (Outer Join) :

- The SQL FULL JOIN combines the results of both left and right outer joins.
- The joined table will contain all records from both the tables and fill in NULLs for missing matches on either side.
- It can be represented as,



- **Syntax :** The basic syntax of a FULL JOIN is as follows :

```
SELECT Table1.column1, Table2.column2...
FROM Table1 FULL JOIN Table2 ON Table1.common_field
= Table2.common_field;
```

The result will be -

- **Example :** For above given two tables namely **Student** and **City**, we can apply full join. It will return returns rows when there is a match in one of the tables using the common column cid. The query will be -

```
SELECT *
FROM Student Full Join City on Student.cid=City.cid
```

The result will be -

sid	cid	sname	cid	cname
1	101	Ram	101	Pune
2	101	Shyam	101	Pune
3	102	Seeta	102	Mumbai
4	NULL	Geeta	NULL	NULL
NULL	NULL	NULL	103	Chennai

3.11 Exist, Any, All

Q.19 Explain Exists operator in SQL with suitable example.

☞ [SPPU : Nov.-19, Marks 5]

Ans. :

- The **EXISTS** operator is used to test for the existence of any record in a subquery. The **EXISTS** operator returns true if the subquery returns one or more records.
- Syntax

```
SELECT column_name(s)
FROM table_name
WHERE EXISTS
(SELECT column_name FROM table_name WHERE condition);
```

- Example - Consider two tables namely **Student** and **Student_Score** table.
- If we want to get all students that have received a 10 grade in the subject 'Maths'.

- The SQL query using EXISTS can be written as follows -

```
SELECT
  RollNo, FName, LName
FROM
  Student
WHERE EXISTS (
  SELECT Student_Score.RollNo
  FROM
    Student_Score
  WHERE
    Student_Score.RollNo = Student.RollNo AND
    Student_Score.grade = 10 AND
    Student_Score.Subject = 'Maths'
)
```

Q.20 Explain Any and All Operators.

Ans. :

- The Any operator returns a Boolean value as a result. It returns true if any of the subquery meets the condition. When we use ANY operator then that means the condition will be true if the operation is true for any value in the range.
- Syntax

```
SELECT column_name(s)
FROM table_name
WHERE column_name operator ANY
 (SELECT column_name
   FROM table_name
   WHERE condition);
```

- Query : Find the name of the students who have scored the grade greater than 9.5 in any test.

We will write the query as

```
SELECT FName,LName
FROM Student
WHERE RollNo = ANY
 (SELECT RollNo
 FROM Student_Score
```



```
 WHERE Grade > 9.5);
```

- The ALL operator returns a boolean value as result. It returns True if all of the subquery values meet the condition. When we use ALL operator then that means the condition will be true if the operation is true for all values in the range.
- Syntax

```
SELECT column_name(s)
FROM table_name
WHERE column_name operator ALL
 (SELECT column_name
   FROM table_name
   WHERE condition);
```

- **For example -** We will again consider the above specified **Student** and **Student_Score** tables. **Query :** Find the name of the students who have appeared only for three tests
- The SQL statement will be as follows -

```
SELECT FName,LName
FROM Student
WHERE RollNo = ALL
 (SELECT RollNo
 FROM Student_Score
 WHERE count(Test_no) = 3);
```

3.12 View and its Types

Q.21 Explain the concept of view along with its operations.

Ans. :

- Views in SQL are kind of virtual tables.
- A view also has rows and columns as they are in a real table in the database.
- We can create a view by selecting fields from one or more tables present in the database.

- A view can either have all the rows of a table or specific rows based on certain condition.

Creating View

We can create a view using CREATE VIEW statement. The syntax is

```
CREATE VIEW name_of_view AS
SELECT column1,column2,...
FROM table_name1,table_name2,...
WHERE condition;
```

Example

i) **Creating a view using single table :** Consider table Empolyee and create a view EmployeeDetails whose Salary is <10000

EmpID	EName	Salary
101	Archana	20000
102	Madhura	5000
103	Poonam	8000
104	Sharda	15000
105	Monika	7000

```
CREATE VIEW EmployeeDetails(EmpID, EName) AS
SELECT E.EmpID, E.EName
OM Employee E
ERE E.Salary > 10000
```

The Output will be -

EmpID	EName
102	Madhura
103	Poonam

105	Monika

View Update

- The SQL UPDATE VIEW command can be used to modify the data of a view.
- All views are not updatable. So, UPDATE command is not applicable to all views.
- An updatable view is one which allows performing a UPDATE command on itself without affecting any other table.
- **Syntax for updating view**

```
UPDATE < view_name >
SET<column1>=<value1>,<column2>=<value2>,....
WHERE <condition>;
```

- **Example**

```
UPDATE VIEW Employee_dept_Details
SET Salary=1000
WHERE EName='Monika';
This view can be viewed by using following query
SELECT * FROM Employee_dept_Details;
```

3.13 Transaction Control Commands

Q.22 What are three types of TCL commands ? Illustrate.

Ans. : The COMMIT, ROLLBACK, and SAVEPOINT are collectively considered as Transaction Commands

1) **COMMIT :** The COMMIT command is used to save permanently any transaction to database. When we perform, Read or Write operations to the database then those changes can be undone by rollback operations. To make these changes permanent, we should make use of **commit**

2) **ROLLBACK :** The ROLLBACK command is used to undo transactions that have not already saved to database. For example

used to get back to previous permanent status of the table. similar to undo com

- Consider the database table as

RollNo	Name
1	AAA
2	BBB
3	CCC
4	DDD
5	EEE

Fig. 3.23.1 Student table

- Following command will delete the record from the database, but if we immediately performs ROLLBACK, then this deletion is undone.

For instance -

```
DELETE FROM Student
WHERE RollNo =2;
ROLLBACK;
```

Then the resultant table will be

RollNo	Name
1	AAA
2	BBB
3	CCC
4	DDD
5	EEE

3) SAVEPOINT : A SAVEPOINT is a point in a transaction when you can roll the transaction back to a certain point without rolling back the entire transaction. The SAVEPOINT can be created as

```
SAVEPOINT savepoint_name;
```

- Then we can ROLLBACK to SAVEPOIT as

```
ROLLBACK TO savepoint_name;
```

used along with the ROLLBACK command
It is used to mark a trasaction in table

- For example - Consider Student table as follows -

RollNo	Name
1	AAA
2	BBB
3	CCC
4	DDD
5	EEE

Fig. 3.23.2 Student table

- Consider Following commands

```
SQL> SAVEPOINT S1
SQL>DELETE FROM Student
Where RollNo=2;
SQL> SAVEPOINT S2
SQL>DELETE FROM Student
Where RollNo=3;
SQL> SAVEPOINT S3
SQL>DELETE FROM Student
Where RollNo =4
SQL> SAVEPOINT S4
SQL>DELETE FROM Student
Where RollNo=5
SQL> ROLLBACK TO S3;
```

- Then the resultant table will be

RollNo	Name
1	AAA
2	BBB
3	CCC

- Thus the effect of deleting the record having RollNo 2, and RollNo 3 is undone.

Q.23 Schema definition for supplier-and-parts database.
Supplier = (supplier_number, supplier_name, status, city)
Parts = (part_number, part_name, colour, weight, city)
Shipments = (supplier_number, part_number, quantity)
Write SQL query for following requirements (any 2).
i) Find shipment information (supplier_number, supplier_name, part_number, part_name, quantity) for those having quantity less than 150.
ii) List supplier_number, suppler_name, part_number, part_name for those suppliers who made shipment of parts whose quantity is larger than the average quantity.
iii) Find aggregate quantity of part_number 'A692' of colour 'GREEN' for which shipment made by supplier_number who reside in 'MUMBAI'.
Ans. : ☞ [SPPU : May-18, End Sem, Marks 5]

```
i)  SELECT
    supplier_number,supplier_name,part_number,part_name,quantity
    FROM Supplier, Parts, Shipments
    WHERE Supplier.supplier_number = Shipments.supplier_number AND
    Parts.part_number=Shipments.part_number
    AND Shipments.quantity<150
ii) SELECT supplier_number, suppler_name, part_number, part_name
    FROM Supplier, Parts
    WHERE Supplier.supplier_number = Shipments.supplier_number AND
    Part.part_number= Shipments.part_number AND
    Shipments.quantity>(SELECT AVG(quantity) FROM Shipments)
iii) SELECT COUNT(part_number)
    FROM Parts, Supplier, Shipments
    WHERE Supplier.supplier_number = Shipments.supplier_number AND
    Part.part_number= Shipments.part_number AND
    Parts.part_number = 'A692' AND Parts.colour='GREEN' AND
    Supplier.city='MUMBAI'
```

Q.24 Consider the following database schema :
Loan (loan_no, Branch_name, Amount)
Borrower (Cust_name, Loan no).
Write SQL queries for following requirements (any two) :
i) Find all customers who have a loan from bank. Find their names, loan nos. and Loan amount.
ii) Find names of customers in alphabetical order who have a loan at Pune branch.

iii) Find all loan nos. for loan made at Pune branch with loan amount greater than 20000. ☞ [SPPU : Oct.-18, End Sem, Marks 5]
Solution :

```
(i)  SELECT Cust_name Loan_no, Amount
     FROM Loan, Borrower
     WHERE Loan.Loan_no = Borrower.Loan_no
(ii) SELECT Cust_name
     FROM Borrower, Loan
     WHERE Loan.Branch_name = 'Pune'
     ORDER BY Cust_name ASC
(iii) SELECT Loan_no
     FROM Loan, Borrower
     WHERE Loan.Loan_no = Borrower.Loan_no AND Loan.Branch_name
= 'Pune' AND
     Loan.Amount>20000
```

Q.25 Consider Employee database with following schema :
Employee(Emp_Id, First_Name, Last_Name, Salary, Joining_Date, Department)
Bonus(Emp_Ref_Id, Bonus_Amount, Bonus_Date)
Designation(Emp_Ref_Id, Emp_Designation, Affected_From)
Write queries in SQL for following requirements (any 2) :
i) To fetch the departments that have less than five people in it.
ii) To print the name of employees having the highest salary in each department.
iii) Write an SQL query to print details of the employee who are also Managers. ☞ [SPPU : Dec.-18, End Sem, Marks 5]
Solution :

```
(i)  SELECT Department, COUNT(Emp_Id) as 'Number of People'
     FROM Employee GROUP BY Department
     HAVING COUNT(Emp_Id) <5
(ii) SELECT E.Department,t.First_Name,t.Salary
     FROM
     (SELECT MAX(Salary) AS TotalSalary, Department FROM
     Employee GROUP BY Department)
     AS temp
     INNER JOIN ON temp.Department=E.Department
     AND temp.TotalSalary=E.Salary;
(iii) SELECT DISTINCT E.FIRST_NAME, D.Emp_Designation
     FROM Employee E, Designation D
```

```
WHERE E.Emp_Id = D.Emp_Ref_Id
AND D.Emp_Designation IN ('Manager');
```

Q.26 Consider following schema :
account (acct-no, branch - name, balance)
Depositor (cust - name, acct - no)
borrower (cust-name, loan-no)
loan (loan - no, branch - name, amount)
Write following queries using SQL (any 2)
i) Find names of all customers who have a loan at the redwood branch.
ii) Find all customers who are having an account and loan or both.
iii) Find average account balance at each branch.

☞ [SPPU : May-19, End Sem, Marks 5]

Ans. :

```
(i) SELECT DISTINCT cust-name
    FROM borrower, loan
    WHERE borrower.loan-no = loan.loan-no AND loan.branch-name
    = 'redwood'
(ii) SELECT cust-name FROM Depositor
    UNION
    SELECT cust-name FROM borrower
(iii) SELECT branch-name, AVG(balance)
    FROM account
    GROUP BY branch-name
```

Q.27 Emp(E_number, E_name, Dept_no)
Dept(Dept_no, Dept_name).
Consider the schema given above. Consider above tables are created without considering the Dept_no as primary key in Dept table and foreign key in Emp table. Assuming tables are already created write SQL queries for following requirements.
i) Create primary key in dept table considering above situations.
ii) Create foreign key considering EMP as child table and dept as master table also consider the above situation.
iii) Add column salary with appropriate data type in EMP table.

☞ [SPPU : Oct.-19, In Sem, Marks 5]

Solution :

```
i) ALTER TABLE Dept
   ADD PRIMARY KEY(Dept_no)
ii) ALTER TABLE Emp
   ADD CONSTRAINT FK_deptno
```



```
   FOREIGN KEY (Dept_no) REFERENCES Dept(Dept_no);
iii) ALTER TABLE Emp
   ADD Salary int;
```

3.14 PL/SQL Concepts

Q.28 What is PL/SQL ?
Ans. :

- PL/SQL stands for Procedural Language extensions to the Structured Query Language (SQL).
- PL/SQL is a combination of SQL along with the procedural features of programming languages.
- Oracle uses a PL/SQL engine to processes the PL/SQL statements.
- PL/SQL includes procedural language elements like conditions and loops. It allows declaration of constants and variables, procedures and functions, types and variable of those types and triggers.

Q.29 Write the PL/SQL block of code to calculate the factorial value of a number. ☞ [SPPU : May-19, Dec.-19, End Sem, Marks 5]

Solution :

```
SET SERVEROUTPUT ON;
DECLARE
-- The factorial f is initialized to 1
 f NUMBER := 1;
--user inputs the number here in variable n
 n NUMBER :=&1;
 num NUMBER := n;
BEGIN
 WHILE n>0 LOOP
  f:=n*f;
  n:=n-1;
 END LOOP;
 DBMS_OUTPUT.put_line('Factorial of ' || num || ' is ' || f);
END;
/
```

```
SET Serveroutpu on;
Declare
nnumber;
fact number := 1;
number;
begin
n := &n;
for i in 1 .. n
loop
fact := fact*i;
end loop;
Dbms_output.put_line('Factorial
'||fact
end;
/
```

Output

```
Factorial of 7 is 5040
```

Q.30 Write PI/SQL block of code for following requirement :
Student_fees (PRN, S_name, class, fees_paid).
Accept the PRN of student from user, check the fees paid by student, if fees paid is less than 30,000 then display the message on screen not paid full fees and display the total fees due. If fees_paid is greater than or equal to 30,000 then display message no fees due. ☞ [SPPU : Oct.-19, In Sem, Marks 5]

Ans. :

PL/SQL Program

```
SET SERVEROUTPUT ON;
DECLARE
 temp_PRN NUMBER(5);
 temp_fees NUMBER(10);
BEGIN
 temp_PRN := &temp_PRN;
 SELECT fees_paid INTO temp_fees FROM STUDENT_FEES WHERE PRN
= temp_PRN;
 DBMS_OUTPUT.put_line('PRN = ' || temp_prn);
 DBMS_OUTPUT.put_line('Fees(Rs.) = '|| temp_fees);
 IF(temp_fees<30000) THEN
  DBMS_OUTPUT.put_line('The Feees not Paid Fully');
 ELSE
  DBMS_OUTPUT.put_line('No Fees Due');
 END IF;
END;
/
```

Output(Run1)

```
PRN = 101
Fees(Rs.) = 30000
No Fees Due
```


3.15 Cursors

Q.31 Write short note on - Cursor.
Ans. :

- When an SQL statement is processed, Oracle creates a memory area known as **context area.** A cursor is a **pointer to this context area.**
- It contains all information needed for processing the statement.
- In PL/SQL, the context area is controlled by cursor.
- A cursor contains information on a SELECT statement and the rows of data accessed by it.
- The cursor is used to fetch and process the rows returned by SQL statement one at a time.

Q.32 Explain Explicit cursor.
Ans. :

- Explicit cursors are used when you are executing a SELECT statement query that will return more than one row.
- Cursors can process one record at a given point of time even though it stores more than one record.
- An explicit cursor is defined in the declaration section of the PL/SQL Block.
- Explicit cursors are used if you need to have better control over the Context Area via Cursor.

Syntax for Declaration of Explicit Cursor

```
CURSOR cusrsor_name
IS
SELECT statement
```

For example

```
CURSOR MyCursor
IS
SELECT * FROM Student;
```

3.16 Stored Procedures

Q.33 Write a PL/SQL code for addition of two numbers using procedure.

Ans. : Following example show the procedure for addition of two numbers. The result is stored in third variable which is an output variable.

Step 1 : We will create a procedure and store it in an sql file as follows

```
CREATE OR REPLACE PROCEDURE AddProc(x IN NUMPER, y IN
NUMBER, z OUT NUMBER)
IS
BEGIN
 z:=x+y;
END;
/
```

Two Input and One output parameter

Step 2 : The call to the procedure can be made from another file. The code for calling the procedure for execution is as follows -

```
SET SERVEROUTPUT ON;
DECLARE
a NUMBER;
b NUMBER;
c NUMBER;

BEGIN
 a := 10;
 b := 20;
 AddProc(a,b,c);
 dbms_output.put_line('Addition of two numbers:'||c);
END;
```

3.17 Stored Function

Q.34 How to write functions in PL/SQL ? Give its syntax and examples.

Ans. :

- Stored function is a named block or subprogram in PL/SQL.

- In PL/SQL, a function takes one or more parameter and returns one value.
- The syntax for a function in PL/SQL is as follows :

```
CREATE [Or REPLACE] Function Function_Name
[(Parameter,..)]
Return Datatype
[IS | AS]
[Declaration Section]
BEGIN
[Executable Section]
END Function_Name;
```

Q.35 Write PL/SQL block of code which accepts the roll.no. from user, the attendance of roll no entered by user will be checked in stud_att (Roll_no, Att) table. Attendance of Roll no entered is displayed on screen. ☞ [SPPU : Aug.-17, End Sem, Marks 5]

Ans. : PL/SQL Program

Step 1 : Create a function in a sql file as

```
example.sql
CREATE OR REPLACE FUNCTION Display(roll NUMBER)
 RETURN NUMBER IS
  temp_att NUMBER(5);
BEGIN
  SELECT att INTO temp_att FROM TESTUDENTS WHERE roll_no = roll;
  RETUFN temp_att;
END;
/
```

Step 2 : Following is a driver program that calls above created function

```
driver.sql
SET SERVEROUTPUT ON;
DECLARE
 temp_rollno NUMBER(3);
 temp_att NUMBER(5);
 total_days NUMBER(5) := 200;
BEGIN
  temp_rollno := &temp_rollno;
```

```
  temp_att := Display(temp_rollno);
  DBMS_OUTPUT.put_line('Roll No = ' || temp_rollno||' Attendance = '||
temp_att);
END;
/
```

Output

```
Roll No = 4 Attendance = 100
```

3.18 Database Triggers

Q.36 What is trigger ?
Ans. :

- Trigger is something that is invoked automatically when some event occurs.
- PL/SQL triggers are **block structures or pre-defined programs,** which may be in-built or even explicitly developed by the programmers for a particular task.
- Trigger is **stored into database** and invoked repeatedly, when specific condition match.
- Triggers are stored programs, which are automatically executed or fired when some event occurs.
- Triggers are associated with response-based events such as a,
  - Database Definition Language (DDL) statement such as CREATE, DROP or ALTER .
  - Database Manipulation Language (DML) statement such as UPDATE, INSERT or DELETE.
  - Any other database operation such as a Startup, Shutdown, Logging in and Logging Out.

Syntax

```
CREATE OR REPLACE TRIGGER Trigger_Name
BEFORE or AFTER or INSTEAD OF
INSERT or UPDATE or DELETE
```



```
of Column_Name
ON Table_Name
[REFERENCING OLD AS O New AS N]
FOR EACH ROW
WHEN (Condition)
DECLARE
Declaration Section
BEGIN
Execution Section
END;
```

Where

- CREATE [OR REPLACE] TRIGGER trigger_name: It creates or replaces an existing trigger with some trigger_name.
- {BEFORE | AFTER | INSTEAD OF} : This specifies when the trigger would be executed. The INSTEAD OF clause is used for creating trigger on a view.
- {INSERT [OR] | UPDATE [OR] | DELETE} : This specifies the DML operation.
- [OF col_name] : This specifies the column name that would be updated.
- [ON table_name] : This specifies the name of the table associated with the trigger.
- [REFERENCING OLD AS o NEW AS n]: This allows you to refer new and old values for various DML statements, like INSERT, UPDATE, and DELETE.
- [FOR EACH ROW] : This specifies a row level trigger, i.e., the trigger would be executed for each row being affected. Otherwise the trigger will execute just once when the SQL statement is executed, which is called a table level trigger.

- WHEN (condition) : This provides a condition for rows for which the trigger would fire. This clause is valid only for row level triggers.

Q.37 Write a PL/SQL trigger for following requirement :
Event : Deletion of row from stud (roll_no, name, class) table.
Action : after deletion of values from stud table, values should be inserted into cancel_admission (roll_no, name) table
Note : for every row to be deleted, action should be performed.

☞ [SPPU : Dec.-17, End Sem, Marks 5]

Ans. :

Step 1 : Create table **stud_table** using following SQL statement

```
CREATE TABLE stud_table(
  roll_no NUMBER(5),
  sname VARCHAR2(30),
  sclass VARCHAR2(20)
);
```

Step 2 : Insert data into table using following commands

```
INSERT INTO stud_table VALUES(111,'AAA','TECOMP');
INSERT INTO stud_table VALUES(222,'BBB','SEMECH');
INSERT INTO stud_table VALUES(333,'CCC','BECIVIL');
INSERT INTO stud_table VALUES(444,'DDD','TEETC');
```

Step 3 : Create table **cancel_admission** using following SQL statement

```
CREATE TABLE cancel_admission(
  roll_no NUMBER(5),
  sname VARCHAR2(30)
);
```

Step 4 : Following is a PL/SQL program that deletes data from **stud_table** and insert the deleted data into **cancel_admission** table

```
TriggerDemo1.sql
SET SERVEROUTPUT ON;
CREATE OR REPLACE TRIGGER Stud_AD
AFTER DELETE  ON stud_table
FOR EACH ROW
ENABLE
```



```
BEGIN
 INSERT INTO cancel_admission
 (roll_no,
  sname)
 VALUES
 ( :old.roll_no,
  :old.sname);
END;
/

DELETE FROM stud_table WHERE roll_no=222
```

Output

On compiling above trigger we get the output as follows -

```
TRIGGER Stud_AD Compiled.
1 rows deleted
```

The execution of this trigger can be verified by observing the data from student_table and cancel_admission table.

Stud_table Table

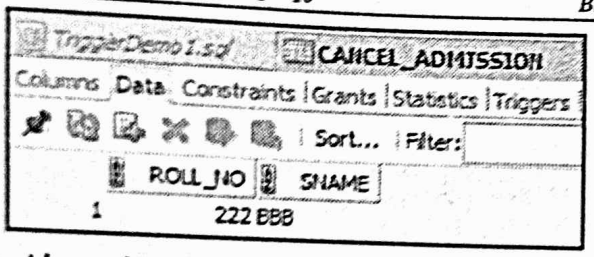
TriggerDemo1.sql | STUD_TABLE

Columns | Data | Constraints | Grants | Statistics | Tr

Sort... | Filter:

	ROLL_NO	SNAME	SCLASS
1	111	AAA	TECOMP
2	333	CCC	BECIVIL
3	444	DDD	TEETC

Cancel_admission Table



Q.38 Consider the schema : student_fees_detail (name, total_fees_deposited, till_date)
Answer the following :
i) Write a SQL query to display the total fees deposited by students whose minimum 3 character name starts with aj.
ii) Write Database Trigger to preserve the old values of student fees details before updating in table.

[SPPU : Dec.-18,19, End Sem, Marks 5]

Ans. : i) SQL Query is as follows -

```
SELECT name,total_fees_deposited
FROM student_fees_detail
WHERE LEN(name)>2 AND name LIKE 'aj%'
```

ii)

Step 1 : First of all we will create the table **stud_fees_detail** using following SQL statement.

```
CREATE TABLE stud_fees_detail
 (
  name VARCHAR2(30),
  fees_deposited NUMBER(5),
  till_date DATE
);
```

Step 2 : Insert values the values into this table using following SQL statement -

```
INSERT INTO stud_fees_detail VALUES('AAA',2000,'27-JUNE-19');
INSERT INTO stud_fees_detail VALUES('BBB',5000,'16-May-20');
INSERT INTO stud_fees_detail VALUES('CCC',1000,'01-August-19');
INSERT INTO stud_fees_detail VALUES('DDD',3000,'25-March-20');
INSERT INTO stud_fees_detail VALUES('EEE',2500,'12-November-20');
```


Step 3 : Simply create a table **backup_fees** using following SQL statement -

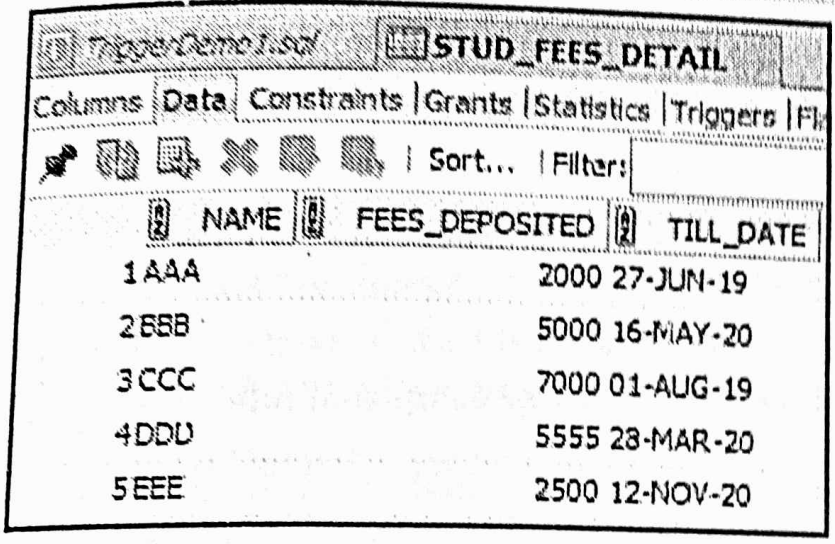
```
CREATE TABLE backup_fees
 (
  name VARCHAR2(30),
  fees_deposited NUMBER(5),
  till_date DATE
);
```

Step 4 : The PL/SQL script for the updating values is as shown below

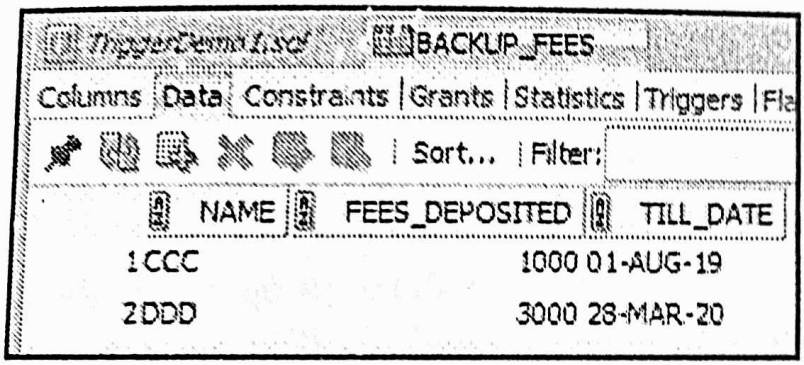
```
SET SERVEROUTPUT ON;
CREATE OR REPLACE TRIGGER Stud_fees_AU
AFTER UPDATE  ON stud_fees_detail
FOR EACH ROW
ENABLE

BEGIN
 INSERT INTO backup_fees
 (name,
 fees_deposited,
 till_date
 )
 VALUES
 (:old.name,
  :old.fees_deposited,
  :old.till_date
  );
END;
/
UPDATE stud_fees_detail
SET fees_deposited=7000
WHERE name='CCC'

UPDATE stud_fees_detail
SET fees_deposited=5555
WHERE name='DDD'
```

Output

Stud_fees_detail Table(This table is updated as trigger executes)



Backup_fees Table



END ...

Unit IV

4 Database Transactions Management

4.1 Basic Concepts of a Transaction

Q.1 Define transaction.

Ans. :

- **Definition of transaction :** A transaction can be defined as a **group of tasks** that form a single logical unit. For example - Suppose we want to withdraw ₹ 100 from an account then we will follow following operations :
  1) Check account balance
  2) If sufficient balance is present request for withdrawal.
  3) Get the money
  4) Calculate Balance = Balance – 100
  5) Update account with new balance.

The above mentioned **five steps** denote **one transaction**.

4.2 Transaction Management

Q.2 Transaction during its execution should be in one of the different states at any point of time, explain the different states of transactions during its execution.

☞ [SPPU : Dec.-17, End Sem, Marks 8]

Ans. : Each transaction has following five states :

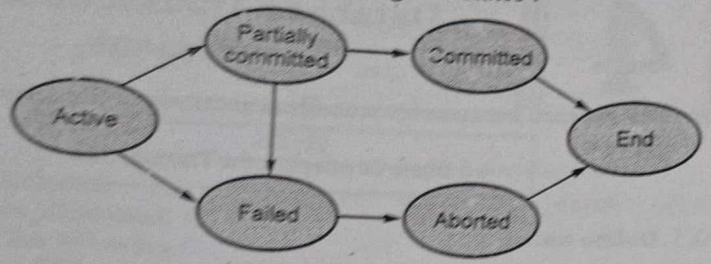


Fig. Q.2.1 Transaction states

1) **Active :** This is the first state of transaction. For example : Insertion, deletion or updation of record is done here. But data is not saved to database.

2) **Partially committed :** When a transaction executes its final operation, it is said to be in a partially committed state.

3) **Failed :** A transaction is said to be in a failed state if any of the checks made by the database recovery system fails. A failed transaction can no longer proceed further.

4) **Aborted :** If a transaction is failed to execute, then the database recovery system will make sure that the database is in its previous consistent state. If not, it brings the database to consistent state by aborting or rolling back the transaction.

5) **Committed :** If a transaction executes all its operations successfully, it is said to be committed. This is the last step of a transaction, if it executes without fail.

4.3 Properties of Transactions

Q.3 What are the ACID properties ? Explain.

Ans. : 1) Atomicity :

- This property states that each transaction must be considered as a **single unit and must be completed fully or not completed** at all.
- No transaction in the database is left half completed.
- Database should be in a state either before the transaction execution or after the transaction execution. It **should not** be in a **state 'executing'.**

2) Consistency :

- The database must remain in consistent state after performing any transaction.
- **For example :** In ATM withdrawal operation, the balance must be updated appropriately after performing transaction. Thus the database can be in consistent state.

3) Isolation :

- In a database system where **more than one transaction** are being executed simultaneously and in parallel, the property of isolation states that all the transactions will be carried out and executed as if it is the only transaction in the system.
- No transaction will affect the existence of any other transaction.

4) Durability :

- The database should be **strong enough** to handle any system **failure.**
- If there is any set of insert /update, then it should be able to handle and commit to the database.
- If there is any **failure,** the database should be able to **recover** it to the consistent state.

Q.4 State and explain in brief the ACID properties. During execution of transaction, a transaction passes through several states, until it finally commits or aborts. List all possible sequences of states through which a transaction may pass. Explain why each state transition occurs.

☞ [SPPU : May-18,19, Dec.-18,19, End Sem, Marks 8]

Ans. : Refer Q.3 and Q.2.

4.4 Concept of Schedule

Q.5 What is schedule ? Also explain the terms - serial schedule and non serial schedule.

Ans. : Schedule is an order of multiple transactions executing in concurrent environment.

Serial schedule : The schedule in which the transactions execute one after the other is called **serial schedule**. It is consistent in nature. For example : Consider following two transactions T1 and T2.

T1	T2
R(A)	
W(A)	
R(B)	
W(B)	
	R(A)
	W(A)
	R(B)
	W(B)

All the operations of transaction T1 on data items A and then B executes and then in transaction T2 all the operations on data items A and B execute. The **R stands for read** operation and **W stands for write** operation.

Non serial schedule : The schedule in which operations present within the transaction are intermixed. This may lead to conflicts in the result or inconsistency in the resultant data.

For example -

Consider following two transactions,

T1	T2
R(A)	
W(A)	
	R(A)
	W(B)
R(A)	
W(B)	
	R(B)
	W(B)

The above transaction is said to be non serial which result in inconsistency or conflicts in the data.

4.5 Serializability : Conflict and View

Q.6 Explain the concept of conflict serializability with example.

Ans. : • **Definition :** Suppose T_1 and T_2 are two transactions and I_1 and I_2 are the instructions in T_1 and T_2 respectively. Then these two transactions are said to be conflict serializable, if both the instruction access the data item d, and at least one of the instruction is write operation.

- In the definition **three conditions** are specified for a conflict in conflict serializability -

  1) There should be **different transactions**

2) The **operations** must be performed on **same data** items

3) **One of the operation** must be the **Write (W)** operation.

Example : Refer Q.7.

Q.7 Consider the following two transactions and schedule (time goes from top to bottom). Is this schedule conflict-serializable ? Explain why or why not.

T_1	T_2
R(A)	
W(A)	
	R(A)
	R(B)
R(B)	
W(B)	

Ans. : Step 1 : To check whether the schedule is conflict serializable or not we will check from top to bottom. Thus we will start reading from top to bottom as,

T_1 : R(A) -> T_1 : W(A) ->T_2 : R(A) -> T_2 : R(B) ->T_1 : R(B) -> T_1 : W(B)

Step 2 : We will find **conflicting operations**. Two operations are called as **conflicting operations** if all the following conditions hold true for them -

i) Both the **operations belong to different transactions.**

ii) Both the operations are on **same data item.**

iii) **At least one** of the two operations is a **write operation.**

From above given example in the top to bottom scanning we find the conflict as,

T_1 : W(A) ->T_2 : R(A).

i) Here note that there are two different transactions T_1 and T_2,

ii) Both work on same data item i.e. A and

iii) One of the operation is write operation.

Step 3 : We will build a precedence graph by drawing one node from each transaction. In above given scenario as there are two transactions, there will be two nodes namely T_1 and T_2.

Fig. Q.7.1

Step 4 : Draw the edge between conflicting transactions. For example in above given scenario, the conflict occurs while moving from T_1 : W(A) to T_2 : R(A). Hence edge must be from T_1 and T_2.

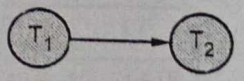


Fig. Q.7.2

Step 5 : Repeat the step 4 while reading from top to bottom. Finally the precedence graph will be as follows

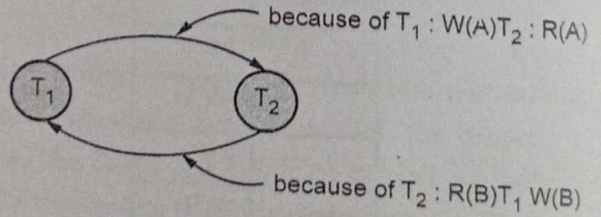


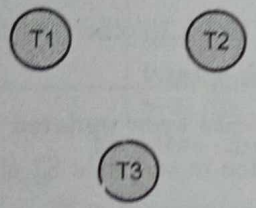
Fig. Q.7.3 : Precedence graph

Step 6 : Check if **any cycle** exists in the graph. Cycle is a path using which we can start from one node and reach to the same node. If the is cycle found then schedule is not conflict serializable. In the step 5 we get **a graph with cycle**, that means given schedule is **not conflict serializable.**

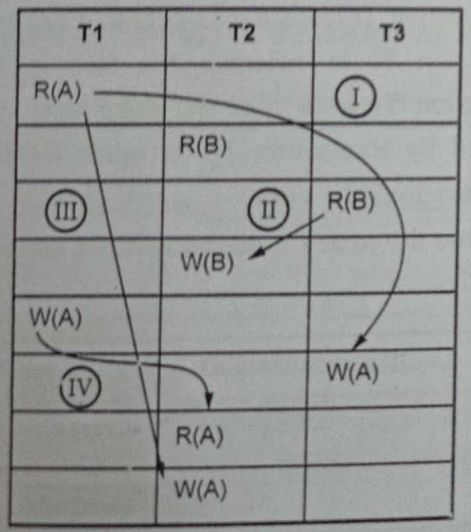
Q.8 Check whether following schedule is conflict serializable or not. If it is not conflict serializable then find the serializability order.

T1	T2	T3
R(A)		
	R(B)	
		R(B)
	W(B)	
W(A)		
		W(A)
	R(A)	
	W(A)	

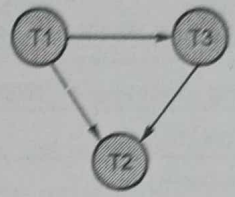
Step 1 : We will read from top to bottom, and build a precedence graph for conflicting entries. We will build a precedence graph by drawing one node from each transaction. In above given scenario as there are three transactions, there will be two nodes namely T1 T2, and T3



Step 2 : The conflicts are found as follows -



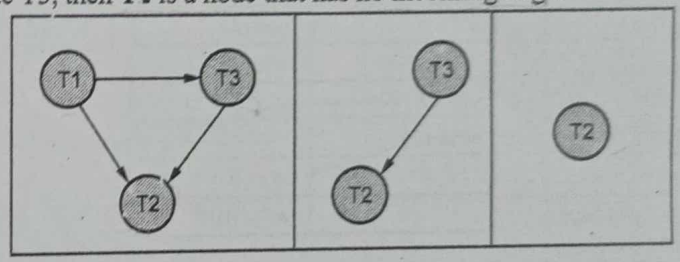
Step 3 : The precedence graph will be as follows -



Step 4 : As there is **no cycle** in the precedence graph, the given sequence is **conflict serializable**. Hence we can convert this non serial schedule to serial schedule. For that purpose we will follow these steps to find the serializable order.

Step 5 : A **serializability order** of the transactions can be obtained by finding a linear order consistent with the partial order of the precedence graph. This process is called **topological sorting**.

Step 6 : Find the vertex which has no incoming edge which is **T1**. If we delete T1 node then **T3** is a node that has no incoming edge. If we delete T3, then **T2** is a node that has no incoming edge.



Thus the nodes can be deleted in a order T1, T3 and T2. Hence the order will be T1-T3-T2.

Q.9 Explain the concept of conflict serializability. Decide whether following schedule is conflict serializable or not. Justify your answer.

T1	T2
read (A)	
write (A)	
	read (A)

	write (A)
read (B)	
write (B)	
	read (B)
	write (B)

[SPPU : May-18, End Sem, Marks 9]

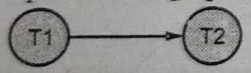
Ans. :

Step 1 : We will read from top to bottom, and build a precedence graph for conflicting entries.

The conflicting entries are as follows -

T1	T2
read (A)	
write (A)	
	read (A)
	write (A)
read (B)	
write (B)	
	read (B)
	write (B)

Step 2 : Now we will build precedence graph as follows -



Step 3 : There is **no cycle** in the precedence graph. That means this schedule is conflict serializable. Hence we can convert this non serial schedule to serial schedule. For that purpose we will follow the following steps to find the serializable order.

1) Find the vertex which has no incoming edge which is T1.
2) Then find the vertex having no outgoing edge which is T2. In between them there is no other transaction.
3) Hence the order will be T1-T2.

Q.10 What is view serializability ?

Ans. :

- If a given schedule is found to be view equivalent to some serial schedule, then it is called as **a view serializable schedule**.
- **View equivalent schedule :** Consider two schedules S1 and S2 consisting of transactions T1 and T2 respectively, then schedules S1 and S2 are said to be view equivalent schedule if it satisfies following **three conditions** :
  - If transaction T1 reads a data item A from the database initially in schedule S2, then in schedule S2 also, T1 must perform the **initial read** of the data item X from the database. This is same for all the data items. In other words - The initial reads must be same for all data items.
  - If data item A has been **updated** at last by transaction Ti in schedule S1, then in schedule S2 also, the data item A must be updated at last by transaction Ti.
  - If transaction Ti reads a data item that has been updated by the transaction Tj in schedule S1, then in schedule S2 also transaction Ti must read the same data item that has been updated by transaction Tj. In other words the Write-Read **sequence must be same**.

Q.11 Give the difference between conflict serializability and view serializability.

Ans. :

Sr. No.	Conflict serializability	View serializability
1.	Every conflict serializable is view serializable.	Every view serializable schedule is not necessarily conflict serializable.
2.	It is easy to test conflict serializability.	It is complex to test view serializability.

Q.12 Check whether following schedule is view serializable or not. Justify your answer. (Note : T1 and T2 are transactions). Also explain the concept of view equivalent schedules and conflict equivalent schedule considering the example schedule given below :

T1	T2
read (A) A: = A - 50	
write (A)	
	read (A) temp: = A*0.1 A: = A - temp
	write (A)
read (B) B: = B + 50	
write (B)	
	read (B) B: = B + temp
	write (B)

☞ **[SPPU : Dec.-18,19, End Sem, Marks 9]**

Ans. :

Step 1 : We will first find if the given schedule is conflict serializable or not. For that purpose, we will find the conflicting operations. These are as shown below -

T1	T2
read (A) A : = A – 50	
write (A)	
	read (A) temp : = A*0.1 A : = A – temp
	write (A)
read (B) B : = B + 50	
write (B)	
	read (B) B : = B + temp
	write (B)

The precedence graph is as follows -

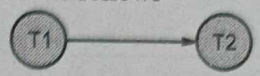


Fig. Q.12.1 Precedence graph

As there exists no cycle, the schedule is conflict serializable. The possible serializability order can be T1-T2

Now we check it for view serializability. As we get the serializability order as T1 – T2, we will find the view equivalence with the given schedule as serializable schedule.

Let S be the given schedule as given in the problem statement. Let the serializable schedule is S'={T1,T2}. These two schedules are represented as follows :

T1	T2
read (A) A: = A - 50	
write (A)	
	read (A) temp: = A*0.1 A: = A - temp
	write (A)
read (B) B: = B + 50	
write (B)	
	read (B) B: = B + temp
	write (B)

Schedule S

T1	T2
read (A) A: = A - 50	
write (A)	
read (B) B: = B + 50	
write (B)	
	read (A) temp: = A*0.1 A: = A - temp
	write (A)
	read (B) B: = B + temp
	write (B)

Schedule S'

Now we will check the equivalence between them using following conditions -

(1) Initial Read

In schedule S initial read on A is in transaction T1. Similarly initial read on B is in transaction T1.

Similarly in schedule S', initial read on A is in transaction T1. Similarly initial read on B is in transaction T1.

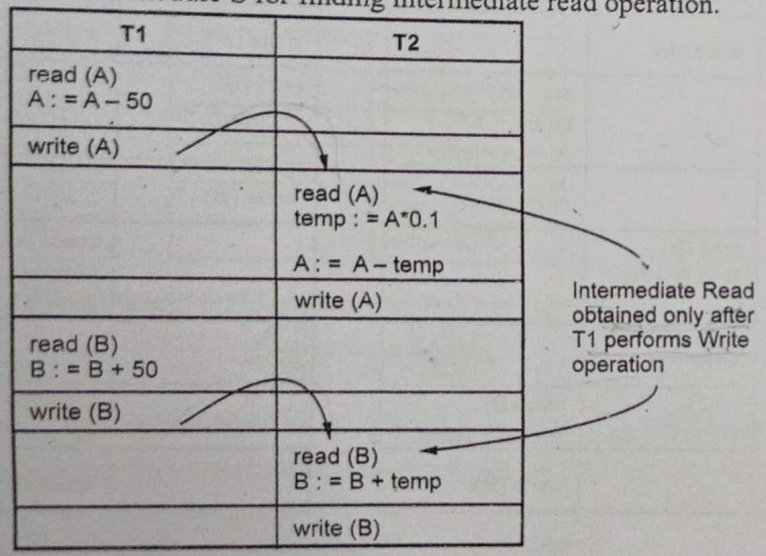
(2) Final Write

In schedule S final write on A is in transaction T2. Similarly final write on B is in transaction T2.

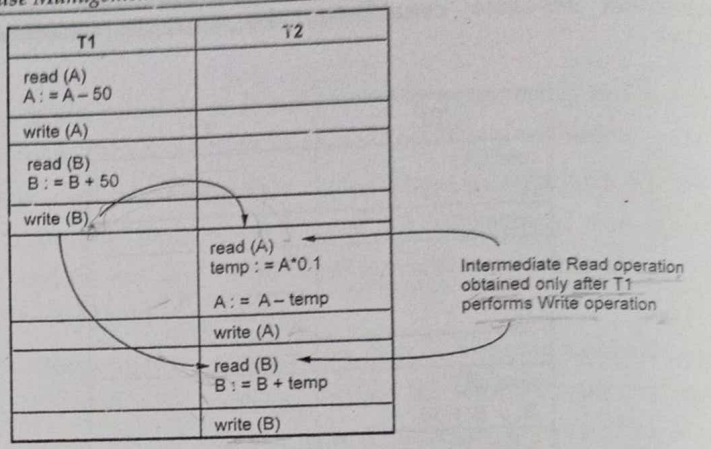
In schedule S' final write on A is in transaction T2. Similarly final write on B is in transaction T2

(3) Intermediate Read

Consider schedule S for finding intermediate read operation.



Similarly consider schedule S' for finding intermediate read operation.



In both the schedules S and S', the intermediate read operation is performed by T2 only after T1 performs write operation.

Thus all the above three conditions get satisfied. Hence given schedule is **view serializable**.

Q.13 Explain the concept of conflict serializability with example. Since every conflict-serializable schedule is view serializable, why do we emphasize conflict serializability rather than view serializability ? ☞ [SPPU : Dec.-18,19, End Sem, Marks 8]

Ans. : Refer Q.9.

4.6 Cascaded Aborts

Q.14 Write a short note on - Cascaded aborts.

Ans. : Cascaded Abort is a situation in which the abort of one transaction forces the abort of another transaction to prevent the second transaction from reading invalid. It is also called as cascading Rollback or Cascading Schedule

For example - Consider the following schedule

T1	T2	T3
Read(A)		
Write(A)		
	Read(A)	

	Write(A)	
		Read(A)
		Write(A)
Failure		

In above schedule, transaction T2 depends upon T1, transaction T3 depends upon T2.

Now failure of transaction T1, causes the transaction T1 to rollback, the rollback of transaction T2 causes T3 to rollback. Such rollbacks is called **cascading rollback or cascading abort.**

4.7 Recoverable and Non-Recoverable Schedules

Q.15 Explain recoverable schedule.

Ans. : Definition : A recoverable schedule is one where, for each pair of transactions T_i and T_j such that T_j reads a data item previously written by T_i, the commit operation of T_i appears before the commit operation of T_j.

For example : Consider following schedule, consider A = 100

T_1	T_2
R(A)	
A=A+50	
W(A)	
	R(A)
	A=A–20
	W(A)
	Commit
Some transaction...	
Commit	

Failure

- The above schedule is inconsistent if failure occurs after the commit of T_2.

- It is because T_2 is **dependable transaction** on T_1.
- Now if the dependable transaction i.e. T_2 is committed first and then failure occurs then if the transaction T_1 makes any changes then those changes will not be known to the T_2. This leads to non recoverable state of the schedule.
- To make the schedule recoverable we will apply the rule that - commit the independent transaction before any dependable transaction.
- In above example independent transaction is T_1, hence we must commit it before the dependable transaction i.e. T_2.
- The recoverable schedule will then be -

T_1	T_2
R(A)	
A=A+50	
W(A)	
	R(A)
	A=A–20
	W(A)
Commit	
	Commit

Q.16 Explain non-recoverable schedule with example.

Ans. : If in a schedule, when a transaction performs a dirty read operation from an uncommitted transaction and commits before the transaction from which it has read the value then such a schedule is known as an **irrecoverable schedule.**

For example : Consider following schedule, consider A = 100

T_1	T_2
R(A)	
A=A+50	
W(A)	

	R(A)
	A=A–20
	W(A)
	Commit
R(B)	
Fail	

Dirty Read

, Now in this operation as soon as failure occurs at T_1 , the transaction is rolled back. At this point again the value of A becomes 100. The value that T_2 read now stands to be incorrect. The T_2 can not recover since it has already committed and the transaction T_2 gets lost completely and it can not be recovered. Hence it is called **non recoverable schedule**.

4.8 Concurrency Control

Q.17 What is concurrency control ?

Ans. : • **Definition of concurrency control :** A mechanism which ensures that simultaneous execution of more than one transactions does not lead to any database inconsistencies is called **concurrency control mechanism.**

4.9 Need for Concurrency Control

Q.18 Explain the need for concurrency control mechanism.

Ans. : **1) Lost Update Problem :** This problem occurs when two transactions that access the same database items have their operations interleaved in a way that makes the value of some database item incorrect.

For example - Consider following transactions

1) Salary of employee is read during transaction T_1.
2) Salary of employee is read by another transaction T_2.

3) During transaction T_1, the salary is incremented by ₹ 200
4) During transaction T_2, the salary is incremented by ₹ 500

T_1	T_2	
Read		Salary = ₹ 1000
	Read	Salary = ₹ 1000
Update Increment salary by ₹ 200		Salary = ₹ 1200
	Update Increment salary by ₹ 500	Salary = ₹ 1500

This update is lost

Time

Only this update is successful

The result of the above sequence is that the update made by transaction T1 is completely lost. Therefor this problem is called as **lost update problem**.

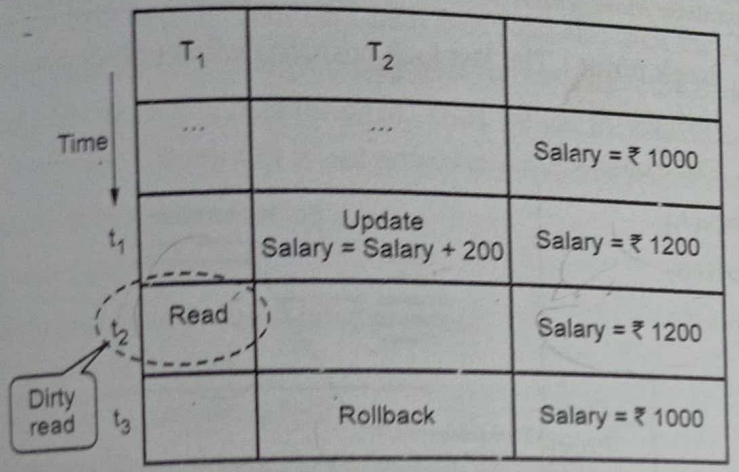
2) Dirty Read or Uncommited Read Problem : The dirty read is a situation in which one transaction reads the data immediately after the write operation of previous transaction

T_1	T_2
R(A)	
A=A+50	
W(A)	
	R(A)
	A=A–20
	W(A)
	Commit
Commit	

Dirty read

For example - Consider following transactions -

Assume initially salary is = ₹ 1000

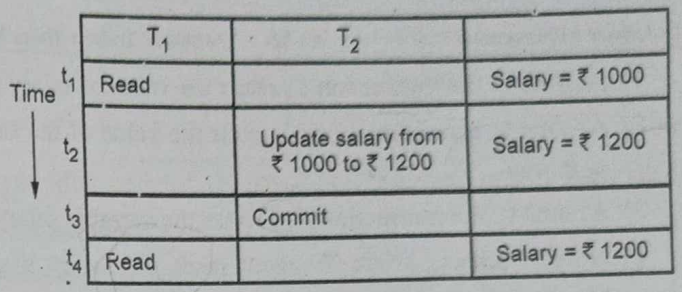


1) At the time t_1, the transaction T_2 updates the salary to ₹ 1200
2) This salary is read at time t_2 by transaction T_1. Obviously it is ₹ 1200
3) But at the time t_3, the transaction T_2 performs Rollback by undoing the changes made by T_1 and T_2 at time t1 and t_2.
4) Thus the salary again becomes = ₹ 1000. This situation leads to **Dirty Read or Uncommited Read** because here the read made at time t_2 **(immediately after update of another transaction)** becomes a dirty read.

3) Non-repeatable read problem

This problem is also known as **inconsistent analysis problem.** This problem occurs when a particular transaction sees two different values for the same row within its lifetime.

For example -



1) At time t_1, the transaction T_1 reads the salary as ₹ 1000
2) At time t_2 the transaction T_2 reads the same salary as ₹ 1000 and updates it to ₹1200
3) Then at time t_3, the transaction T_2 gets committed.
4) Now when the transaction T_1 reads the same salary at time t_4, it gets different value than what it had read at time t_1. Now, transaction T_1 cannot repeat its reading operation. Thus inconsistent values are obtained.

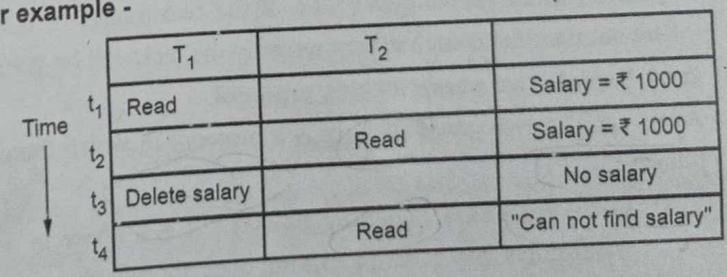
Hence the name of this problem is non-repeatable read or inconsistent analysis problem.

4) Phantom read problem

The phantom read problem is a special case of non repeatable read problem.

This is a problem in which one of the transaction makes the changes in the database system and due to these changes another transaction can not read the data item which it has read just recently.

For example -



1) At time t_1, the transaction T_1 reads the value of salary as ₹ 1000
2) At time t_2, the transaction T_2 reads the value of the same salary as ₹ 1000
3) At time t_3, the transaction T_1 deletes the variable salary.
4) Now at time t_4, when T_2 again reads the salary it gets error. Now transaction T_2 can not identify the reason why it is not getting the salary value which is read just few time back.

This problem occurs due to changes in the database and is called **phantom read problem**.

4.10 Locking Methods

Q.19 Explain the terms - Shared lock and exclusive lock.

Ans. :

i) **Shared lock :** The shared lock is used for reading data items only. It is denoted by Lock-S. This is also called as **read lock**.

ii) **Exclusive lock :** The exclusive lock is used for both read and write operations. It is denoted as Lock-X. This is also called as **write lock**.

- The **compatibility matrix** is used while working on set of locks. The **concurrency control manager** checks the compatibility matrix before granting the lock. If the two modes of transactions are compatible to each other then only the lock will be granted.

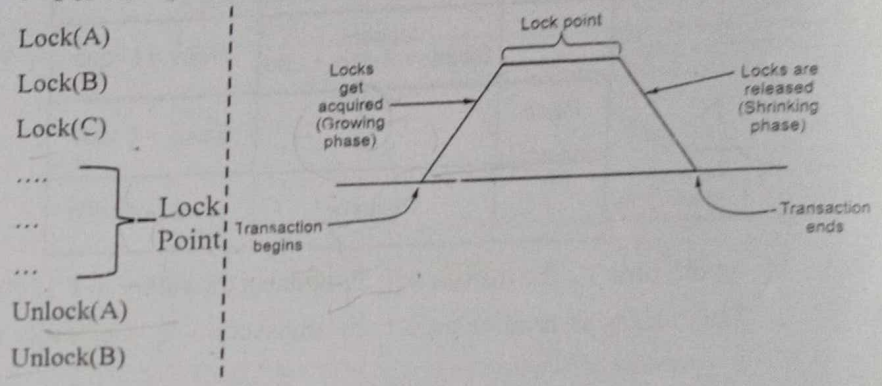
Q.20 Explain two phase locking protocol.

Ans. : • The two phase locking is a protocol in which there are two phases :

i) **Growing phase (Locking phase) :** It is a phase in which the transaction may obtain locks but does not release any lock.

ii) **Shrinking phase (Unlocking phase) :** It is a phase in which the transaction may release the locks but does not obtain any new lock.

- **Lock point :** The last lock position or first unlock position is called lock point.
- For example -



Consider following transactions,

T_1	T_2
Lock-X(A)	Lock-S(B)
Read(A)	Read(B)
A=A–50	Unlock-S(B)
Write(A)	
Lock-X(B)	
Unlock-X(A)	
B=B+100	Lock-S(A)
Write(B)	Read(A)
Unlock-X(B)	Unlock-S(A)

The important rule for being a two phase locking is - **All lock operations precede all the unlock operations.**

In above transactions T_1 is in two phase locking mode but transaction T_2 is not in two phase locking. Because in T_2, the shared lock is acquired by data item B, then data item B is read and then the lock is released. Again the lock is acquired by data item A, then the data item A is read and the lock is then released. Thus we get lock-unlock-lock-unlock sequence. Clearly this is not possible in two phase locking.

Advantages of two phase locking

(1) It ensures serializability.

Disadvantages of two phase locking protocol

(1) It leads to dealocks.

(2) It leads to cascading rollback.

Q.21 Explain types of two phase locking protocol.

Ans. : Types of two phase locking

1) **Strict two phase locking :** The strict 2PL protocol is a basic two phase protocol **but all the exclusive mode locks be held until the transaction commits**. That means in other words all the **exclusive locks** are **unlocked** only after the **transaction is committed**. That also means that if T_1 has exclusive lock, then T_1 will release the exclusive lock only after commit operation, then only other transaction is allowed to read or write. For example - Consider two transactions

T_1	T_2
W(A)	
	R(A)

If we apply the locks then

T_1	T_2
Lock-X(A)	
W(A)	
Commit	

Unlock(A)	
	Lock-S(A)
	R(A)
	Unlock-S(A)

Thus only after commit operation in T_1, we can unlock the exclusive lock. This ensures the strict serializability.

Thus compared to basic two phase locking protocol, the advantage of strict 2PL protocol is it ensures strict serializability.

2) **Rigorous two phase locking :** This is stricter two phase locking protocol. Here all locks are to be held until the transaction commits. The transactions can be seriealized in the order in which they commit.

Example - Consider transactions

T_1
R(A)
R(B)
W(B)

If we apply the locks then

T_1
Lock-S(A)
R(A)
Lock-X(B)
R(B)
W(B)
Commit
Unlock(A)
Unlock(B)

Thus the above transaction uses rigorous two phase locking mechanism.

Q.22 Explain the two phase lock protocol and show how it ensures conflict serializability. Two Phase lock protocol does not ensure freedom from deadlock explain with necessary example. Also explain its two versions : Strict two phase lock protocol and rigorous two phase lock protocol.

☞ [SPPU : Dec.-18,19, End Sem, Marks 9]

OR What benefit does rigorous two-phase locking provide ? How does it compare with other forms of two - phase locking ?

☞ [SPPU : May-19, End Sem, Marks 9]

Ans. : Two phase locking mechanism

Problem of deadlock in two phase locking : The deadlock problem can not be solved by two phase locking. Deadlock is a situation in which when two or more transactions have got a lock and waiting for another locks currently held by one of the other transactions.

For example

T_1	T_2
Lock-X(A)	Lock-X(B)
Read(A)	Read(B)
A=A–50	B=B+100
Write(A)	Write(B)
Delayed, wait for T_2 to release Lock on B	Delayed, wait for T_1 to release Lock on A

Strict and rigorous two phase locking protocol : Refer Q.21.

4.11 Deadlock Handling

Q.23 What is deadlock ? Explain how deadlock detection and prevention is done. ☞ [SPPU : Nov.-17, 19, End Sem, Marks 6]

Ans. : Deadlock : Deadlock can be formally defined as - " A system is in deadlock state if there exists a set of transactions such that every transaction in the set is waiting for another transaction in the set. "

For example - Consider that transaction T_1 holds a lock on some rows of table **A** and needs to update some rows in the **B** table. Simultaneously, transaction T_2 holds locks on some rows in the **B** table and needs to update the rows in the **A** table held by transaction T_1.

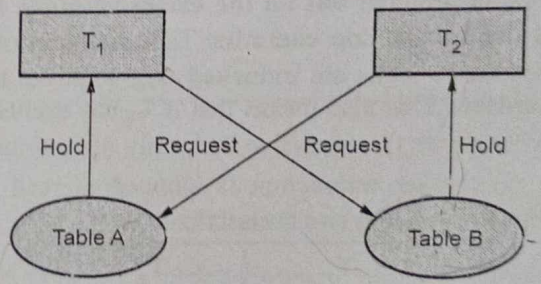


Fig. Q.23.1

Now, the main problem arises. Now transaction T_1 is waiting for T_2 to release its lock and similarly, transaction T_2 is waiting for T_1 to release its lock. All activities come to a halt state and remain at a standstill. This situation is called **deadlock** in DBMS.

Deadlock Detection :

- In deadlock detection mechanism, an algorithm that examines the state of the system is invoked periodically to determine whether a deadlock has occurred or not. If deadlock is occurrence is detected, then the system must try to recover from it.
- Deadlock detection is done using wait for graph method.

Wait for graph

- In this method, a graph is created based on the transaction and their lock. If the created **graph has a cycle or closed loop, then there is a deadlock.**
- The wait for the graph is maintained by the system for every transaction which is waiting for some data held by the others. The system keeps checking the graph if there is any cycle in the graph.
- This graph consists of a pair G = (V, E), where V is a set of vertices and E is a set of edges.
- The set of vertices consists of all the transactions in the system.
- When transaction T_i requests a data item currently being held by transaction T_j, then the edge $T_i \rightarrow T_j$ is inserted in the wait-for graph. This edge is removed only when transaction T_j is no longer holding a data item needed by transaction T_i.

For example - Consider following transactions, We will draw a wait for graph for this scenario and check for deadlock.

T_1	T_2
R(A)	
	R(A)
W(A)	
R(B)	
	W(A)
W(B)	

We will use three rules for designing the wait-for graph -

Rule 1 : If T_i has **Read** operation and then T_2 has **Write** operation then draw an edge $T_1 \rightarrow T_2$.

Rule 2 : If T_1 has **Write** operation and then T_2 has **Read** operation then draw an edge $T_1 \rightarrow T_2$.

Rule 3 : If T_1 has **Write** operation and then T_2 has **Write** operation then draw an edge $T_1 \rightarrow T_2$.

Let us draw wait-for graph

Step 1 : Draw vertices for all the transactions

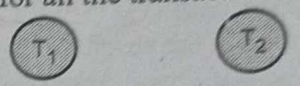


Fig. Q.23.2

Step 2 : We find the Read-Write pair from two different transactions reading from top to bottom. If such as pair is found then we will add the edges between corresponding directions.

For instance -

T_1	T_2
R(A)	
	R(A)
W(A)	
R(B)	
	W(A)
W(B)	

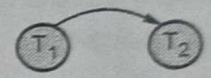


Fig. Q.23.3

Step 3 :

T_1	T_2
R(A)	
	R(A)
W(A)	
R(B)	
	W(A)
W(B)	

Fig. Q.23.4

As **cycle is detected** in the wait-for graph there is no need to further process. The **deadlock** is present in this transaction scenario.

Deadlock Prevention :

For large database, deadlock prevention method is suitable. A deadlock can be prevented if the resources are allocated in such a way that deadlock never occur. The DBMS analyzes the operations whether they can create deadlock situation or not, if they do, that transaction is never allowed to be executed.

There are two techniques used for deadlock prevention -

i) Wait-Die :

- In this scheme, if a transaction requests for a resource which is already held with a conflicting lock by another transaction then the DBMS simply checks the timestamp of both transactions. It **allows the older transaction to wait** until the resource is available for execution.
- Suppose there are two transactions T_i and T_j and let TS(T) is a timestamp of any transaction T. If T_2 holds a lock by some other transaction and T_1 is requesting for resources held by T_2 then the following actions are performed by DBMS :
  - Check if $TS(T_i) < TS(T_j)$ - If T_i is the older transaction and T_j has held some resource, then T_i is allowed to wait until the data-item is available for execution. That means if the older transaction is waiting for a resource which is locked by the younger transaction, then the older transaction is allowed to wait for resource until it is available.
  - Check if $TS(T_i) < TS(T_j)$ - If T_i is older transaction and has held some resource and if T_j is waiting for it, then T_j is killed and restarted later with the random delay but with the same timestamp.

Timestamp is a way of assigning priorities to each transaction when it starts. If timestamp is lower then that transaction has higher priority. **That means oldest transaction has highest priority.**

For example -

Let T_1 is a transaction which requests the data item acquired by transaction T_2. Similarly T_3 is a transaction which requests the data item acquired by transaction T_2.

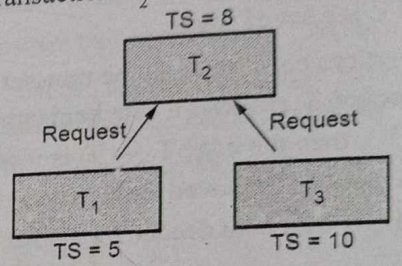


Fig. Q.23.5

Here $TS(T_1)$ i.e. Time stamp of T_1 is less than $TS(T_3)$. In other words T_1 is older than T_3.

Hence T_1 is made to **wait** while T_3 is **rolledback**.

ii) Wound- wait :

- In wound wait scheme, if the older transaction requests for a resource which is held by the younger transaction, then older transaction forces younger one to kill the transaction and release the resource. After some delay, the younger transaction is restarted but with the same timestamp.
- If the older transaction has held a resource which is requested by the younger transaction, then the younger transaction is asked to wait until older releases it.

Suppose T_1 needs a resource held by T_2 and T_3 also needs the resource held by T_2, with $TS(T_1) = 5$, $TS(T_2) = 8$ and $TS(T_3) = 10$, then T_1 being older waits and T_3 being younger dies. After the some delay, the younger transaction is restarted but with the same timestamp.

This ultimately prevents a deadlock to occur.

4.12 Time-stamp based Protocols

Q.24 Explain time-stamp ordering protocol.

☞ [SPPU : May-19, End Sem, Marks 8]

Ans. :

Case 1 (Read) : Transaction T issues **a read(X) operation**

i) If TS(T) < WTS(X), then read(X) is rejected. T has to abort and be rejected.

ii) If WTS(X) ≤ TS(T), then execute read(X) of T and update RTS(X).

Case 2 (Write) : Transaction T issues **a write(X) operation**

i) If TS(T) < RTS(X) or if TS(T) < WTS(X), then write is rejected

ii) If RTS(X) ≤ TS(T) or WTS(X) ≤ TS(T), then execute write(X) of T and update WTS(X).

Example for Case 1 (Read operation)

(i) Suppose we have two transactions T_1 and T_2 with timestamps 10 sec. and 20 sec. respectively.

10 Sec T_1	20 Sec T_2
R(X)	
	W(X)
R(X)	

RTS(X) and WTS(X) is initially = 0

Then RTS(X) = 10, when transaction T_1 executes

After that WTS(X) = 20 when transaction T_2 executes

Now if read operation R(X) occurs on transaction T_1 at $TS(T_1)$ = 10 then

$TS(T_1)$ i.e. 10 <WTS(X) i.e. 20, hence we have to reject second read operation on T_1 i.e.

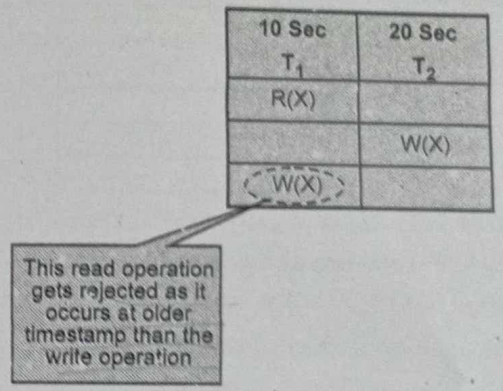


Fig. Q.24.1

ii) Suppose we have two transactions T_1 and T_2 with timestamps 10 sec. and 20 sec. respectively.

10 Sec T_1	20 Sec T_2
W(X)	
	R(X)

RTS(X) and WTS(X) is initially = 0

Then WTS(X) =10 as transaction T_1 executes.

Now if Read operation R(X) occurs on transaction T_2 at $TS(T_2)$ = 20 then

$TS(T_2)$ i.e. 20 >WTS(X) which is 10, hence we accept read operation on T_2. The transaction T_2 will perform read operation and now RTS will be updated as,

RTS(X) = 20

Example for Case 2 (Write Operation)

i) Suppose we have two transactions T_1 and T_2 with timestamps 10 sec. and 20 sec. respectively.

10 Sec. T_1	20 Sec. T_2
R(X)	
	W(X)
W(X)	

RTS(X) and WTS(X) is initially = 0

Then RTS(X) = 10, when transaction T1 executes

After that WTS(X) = 20 when transaction T_2 executes

Now if write operation W(X) occurs on transaction T_1 at $TS(T_1) = 10$ then

$TS(T_1)$ i.e. 10 <WTS(X), hence we have to reject second write operation on T_1 i.e.

10 Sec T_1	20 Sec T_2
R(X)	
	W(X)
R(X)	

This write operation gets rejected as it occurs at older timestamp than the write operation at transaction 2

Fig. Q.24.2

ii) Suppose we have two transactions T_1 and T_2 with timestamps 10 sec. and 20 sec. respectively.

10 Sec T_1	20 Sec T_2
W(X)	
	W(X)

RTS(X) and WTS(X) is initially = 0

Then WTS(X) = 10 as transaction T_1 executes.

Now if write operation W(X) occurs on transaction T_2 at $TS(T_2) = 20$ then

$TS(T_2)$ i.e. 20 >WTS(X) which is 10, hence we accept write operation on T_2. The transaction T_2 will perform write operation and now WTS will be updated as,

WTS(X) = 20

END...

Unit V

5 Parallel Databases

5.1 Introduction to Database Architectures

Q.1 Explain client-server architecture.

Ans. : Client-server architecture

- The client server architecture is also called as **two tier architecture.**
- It consists of **Client computer** and **Server computer** which can communicate through well - defined protocol.

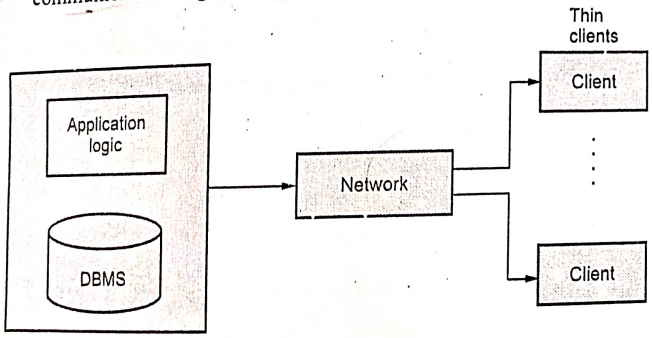


Fig. Q.1.1 Two server architecture

- In traditional client server architecture, the client computer implements simply the graphical user interface part. The Server computer implements application logic and data management part. In such architecture, the client is called as **thin client.**

- On the other hand there is some client server architecture, in which client implements graphical interface as well as business (application) logic part. Only data management part is taken care by server. Such client is called as **thick client.**
- The thick client model has several disadvantages as compared to thin client. Those are -
  - There is no central place to update and maintain the business logic as application program runs on the client computer only.
  - Client should run the business logic with reliability without affecting any security aspects.
  - The thick client architecture can handle only limited number of users.

Q.2 Explain the three tier architecture.

Ans. : The three tier architecture is made up of three tiers as -

1) **Presentation tier :** The presentation tier is comprised of graphical user interface. The user always expects the GUI which is easy to input. He/She expects the results in some organized format. The use web-based interfaces are getting popular.
2) **Middle tier :** This is a tier in which the application or business logic executes. The complex business processes get executed at this tier. The business logic can be implemented in some suitable programming language like C++ or Java.
3) **Data Management tier :** This tier takes care of data management activities. The database management (DBMS) systems are located in this architecture.

The typical three tier architecture is represented by Fig. Q.2.1.

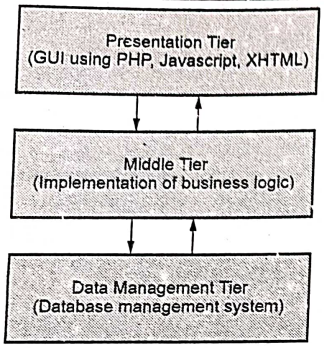


Fig. Q.2.1 Three tier architecture

Q.3 Write short note on - Oracle architecture.

Ans. : • Oracle database is one of the most commonly used database management system. It is popularly used as a database management system because of its features such as high performance, security and scalability.

- The oracle database architecture consists of three main components - Refer Fig. Q.3.1.
  1) Memory structure(Instances)
  2) Database system
  3) Processes -

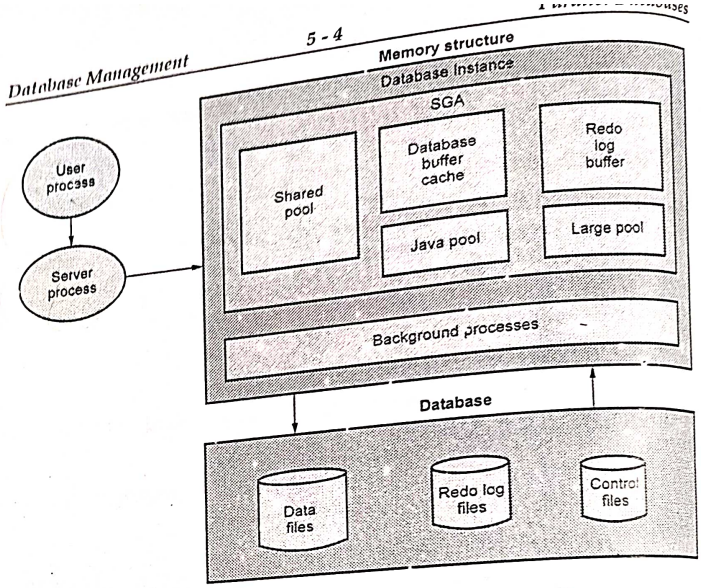


Fig. Q.3.1 Oracle architecture

- **Oracle Instance**
  - The instance is a collection of two things - SGA and **Background Processes**

1) SGA :

- It stands for System Global Area. It is a shared memory area.
- Whenever a database instance starts, some memory gets allocated and that memory is termed as SGA. Along with memory allocation, one or more background processes will also be kicked off to serve the same.
- SGA is used to store data as well as control information about one database instance through its various subcomponents, where each component is dedicated for a specific purpose.
- Various components of SGA are -
  - Database buffer cache

- Redo log buffer
- Java pool
- Large pool
- Shared pool.

2) Background processes :

- Oracle has collection of processes that are called background processes. These processes are responsible for managing memory, performing I/O operations and other maintenance activities.
- Following are some important background processes that are required -
  - **System Monitor Process (SMON) :** These processes are responsible for performing system level recovery and maintenance activities.
  - **Process Monitor Process (PMON) :** The task of these processes is to monitor other background processes.
  - **Database Writer Process (DBWR) :** This process performs the task of writing data blocks from **Database** buffer Cache (present in SGA) to physical data files(Present in Database system).
  - **Log Writer Process (LGWR) :** This process writes the Redo blocks from Redo Log Buffer (present in SGA) to Redo Log Files(present in Database system).
  - **CheckPoint (CKPT) :** This process maintains data files and control files with the most recent checkpoint information.

- **Database System**

a. The database system is situated in the storage of a computer. The database system is simply the collection of files.

b. There are three categories of files that are situated in database system and those are
   - i. **Data files :** These files hold the actual data within a database.
   - ii. **Redo log files :** These files are used to hold all the changes made to the database. Redo log files can be utilized during the database recovery process to retrieve the original information.
   - iii. **Control files :** It is a binary file that holds database status-related information like Database Name, Data File, and Redo Log file Names, File Locations, and Log Sequence Number.

c. There are other few more categories of files that contribute to database management. These are -
   - i. **Parameter file :** This file contains the parameters which defines the way the database is expected to start up.
   - ii. **Password file :** This file holds the user passwords and thus maintains the security of databases.

3) Processes

a. There are two types of processes -
   - i. **User process :** It is also known as client process. The user actually connects to the **instance** with the help of user processes. The user process is stablished when the user sends a connection request to Oracle Server.
   - ii. **Server process :** The server process connects the user to the database and performs the activities on client's behalf as - executing SQL statements or retrieving data from the database.

5.2 Parallel Databases

Q.4 What are benefits of parallel databases ?

Ans. :

1) **High performance :** With more CPUs available to the application the higher speedup can be obtained.
2) **High availability :** The nodes are isolated from each other, hence failure of one node does not cause the entire system to shut down. One of the survived node takes the responsibility of failure node and system continues to provide data access to the users.
3) **Flexibility :** The environment of parallel database execution is extremely flexible. One can allocate or deallocate the instances as necessary.
4) **Scalability :** In parallel systems, we can overcome the memory limits or processor limits by scaling up the required resources. This enables single system to serve thousands of users.

5.3 Types of Parallel Database Architecture

Q.5 Describe various types of parallel database architecture.

Ans. : There are three main architectures that have been proposed for building parallel DBMS.

1) Shared memory architecture :

- In this architecture, multiple CPUs are attached to an interconnection network.
- These CPUs can also access the common region of main memory. Refer Fig. Q.5.1.

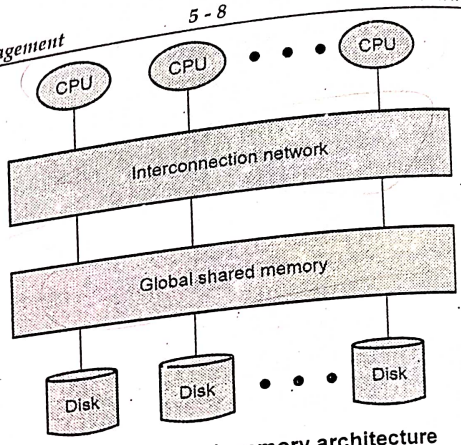


Fig. Q.5.1 Shared memory architecture

2) Shared nothing system :

- In this architecture, each CPU has local main memory and disk space.
- No two CPU can access the same storage area, all the communication between CPUs is through a network connection. Refer Fig. Q.5.2.

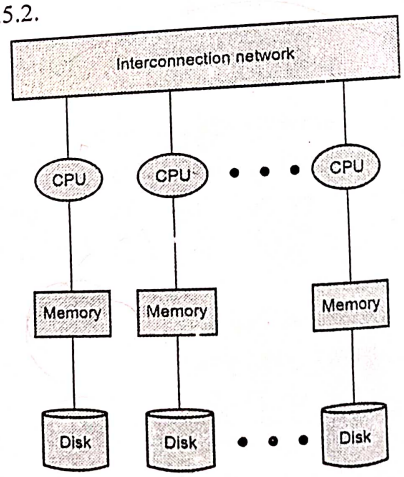


Fig. Q.5.2 Shared nothing system

3) Shared disk architecture :

- In this architecture, each CPU has private memory and direct access to all disks through interconnection network. Refer Fig. Q.5.3.

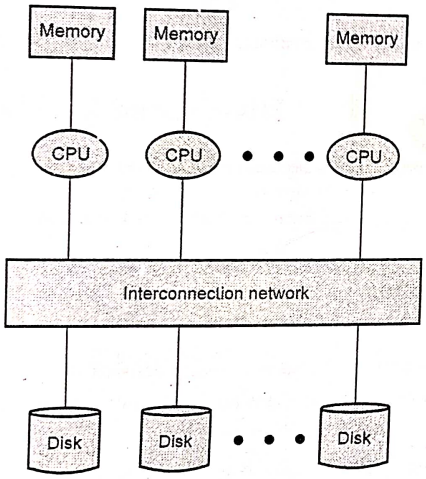


Fig. Q.5.3 Shared disk architecture

If more number of CPUs are added to **shared disk** and **shared memory** architecture, the performance of overall system gets degraded. Thus by addition of more processor the overall system becomes **extremely slow.**

On the other hand, in the **shared nothing** architecture the overall performance of the system does not become slow on addition of CPUs. Hence shared nothing architecture is supposed to be the best architecture for parallel database systems.

5.4 Evaluating Parallel Query in Parallel Databases

Q.6 Explain the parallel query in parallel databases.

Ans. :

- In Parallel execution of multiple queries, identifying the concurrently executing queries become difficult. So the emphasis has been on parallel execution of single query.
- A relational query execution plan is prepared during the evaluation of the parallel query. This graph consists of relational algebra operators.
- If an operator consumes the output of a second operator, the pipelined parallelism can be applied, if not then these operators can proceed independently.
- An operator is said to be blocked if the operator produces no result until it consumes all the inputs.
- Pipelined parallelism is limited by the presence of operators in that block.
- In addition to evaluating different operators in parallel, we can evaluate each individual operator in a query plan in parallel fashion.
- We can partition the input data and then can work on each partition in parallel and combine the results. This approach is called data-partitioned parallel evaluation.
- The data partitioned parallel evaluation approach is successfully used in **shared nothing systems.**

5.5 Virtualization on Multicore Processors

Q.7 What is virtualization ?

Ans. : Definition of virtualization : Virtualization is the process of abstracting computing resources such that multiple operating system

and application images that can share a single physical server, bringing significant cost-of-ownership and manageability benefits.

Q.8 What are the advantages of virtualization ?

Ans. : Following are some advantages of virtualization -

1. Use of virtualization **saves the cost** of **different hardware** required for the application development.
2. From the administration point of view, **managing** both the virtual and physical environment becomes **streamlined**.
3. Virtualization promotes **higher reliability** and **performance**.
4. One can provide **security** to the applications that are running in the virtual environment by protecting from the external threats.
5. As server **workloads** vary, virtualization provides the ability for virtual machines that are over **utilizing the resources** of a server to be moved to underutilized servers. This dynamic **load balancing** creates efficient utilization of server resources.
6. It is possible to have **transitions between different operating systems** on a single machine reducing desktop footprint and hardware expenditure.
7. With virtualization it is possible for **testing of software concurrently** with production use of a host.

END...

Unit VI

6 Distributed Databases

6.1 Basics of Distributed Database Management System

Q.1 What is distributed database management system?

Ans. :

- A distributed database system consists of loosely coupled sites (computer) that share no physical components and each site is associated a database system.
- The software that maintains and manages the working of distributed databases is called distributed database management system.
- The database system that runs on each site is independent of each other. Refer Fig. Q.1.1.

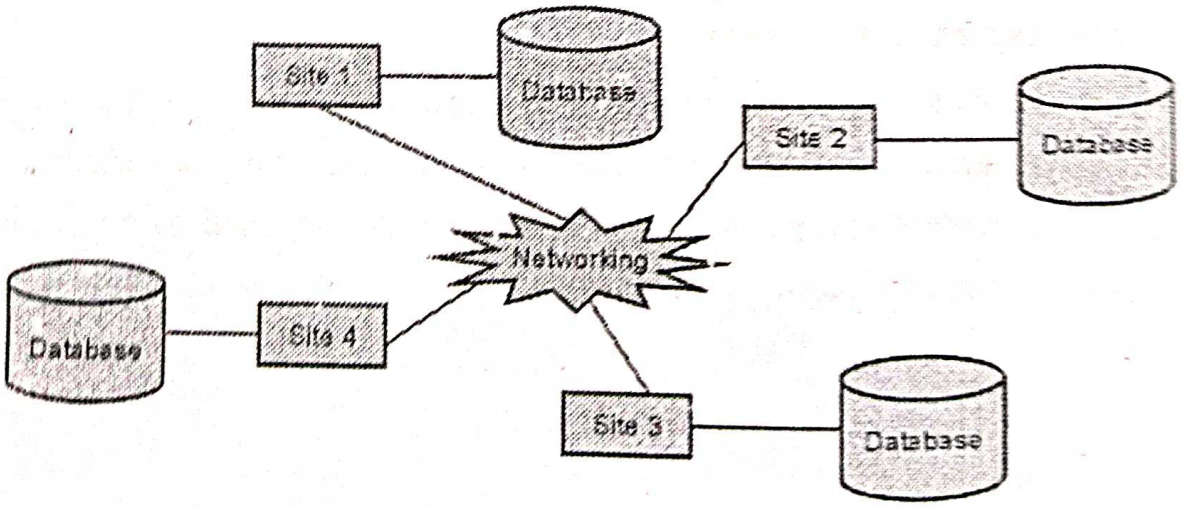


Fig. Q.1.1

Q.2 Enlist advantages and disadvantages of DBMS.

Ans. :

Advantages of distributed database system

(1) There is **fast data processing** as several sites participate in request processing.

(2) **Reliability and availability** of this system is high.

(3) It possess **reduced operating cost.**

(4) It is **easier to expand** the system by adding more sites.

(5) It has improved **sharing ability and local autonomy.**

Disadvantages of distributed database system

(1) The system becomes **complex** to manage and control.

(2) The **security issues** must be carefully managed.

(3) The system require **deadlock handling** during the transaction processing otherwise the entire system may be in inconsistent state.

(4) There is need of some **standardization** for processing of distributed database system.

Q.3 What is difference between distributed DBMS and centralized DBMS.

Ans. :

Sr. No.	Distributed DBMS	Centralized DBMS
1.	The database files are stored at geographically different locations across the network.	The database is stored at centralized location.
2.	As data is distributed over the network, it requires time to synchronize data and thus difficult to maintain.	A centralized database is easier to maintain and keep updated since all the data are stored in a single location.

3.	If one database fails, user can have access to other database files.	If the centralized database fails, then there is no access to a database.
4.	It can have data replication as database is distributed. Hence there can be some data inconsistency.	It have single database system, hence there is no data replication. Therefore there is no data inconsistency.

6.2 Types of Distributed Databases

Q.4 Explain types of distributed databases

Ans. : There are two types of distributed databases -

(1) Homogeneous databases

- The homogeneous databases are kind of database systems in which all sites have identical software running on them. Refer Fig. Q.4.1.

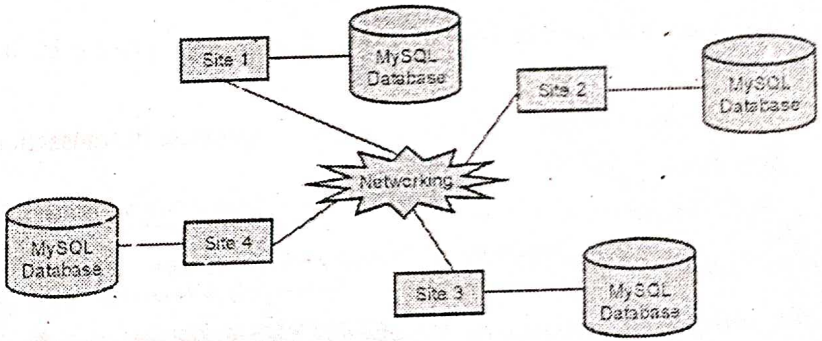


Fig. Q.4.1 Homogeneous databases

- In this system, all the sites are aware of the other sites present in the system and they all cooperate in processing user's request.
- Each site present in the system, surrenders part of its autonomy in terms of right to change schemas or software.
- The homogeneous database system appears as a single system to the user.

(2) Heterogeneous databases

- The heterogeneous databases are kind of database systems in which different sites have different schema or software. Refer Fig. Q.4.2.

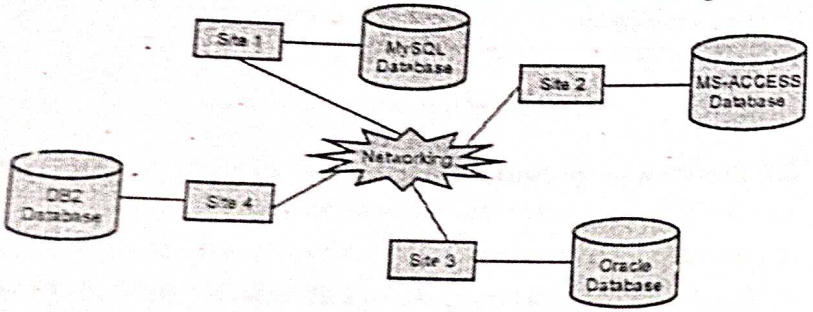


Fig. Q.4.2 Heterogeneous databases

- The participating sites are not aware of other sites present in the system.
- These sites provide limited facilities for cooperation in transaction processing.

6.3 Architecture of Distributed Databases

Q.5 Write short note on – Architecture of distributed databases

Ans. : • Following is an architecture of distributed databases. In this architecture the local database is maintained by each site.

- Each site is interconnected by communication network.

(See Fig.Q.5.1on next page)

- When user makes a request for particular data at site Si then it is first searched at the local database. If the data is not present in the local database then the request for that data is passed to all the other sites via communication network. Each site then searches for that data at its local database. When data is found at particular site say Sj then it is transmitted to site Si via communication network.

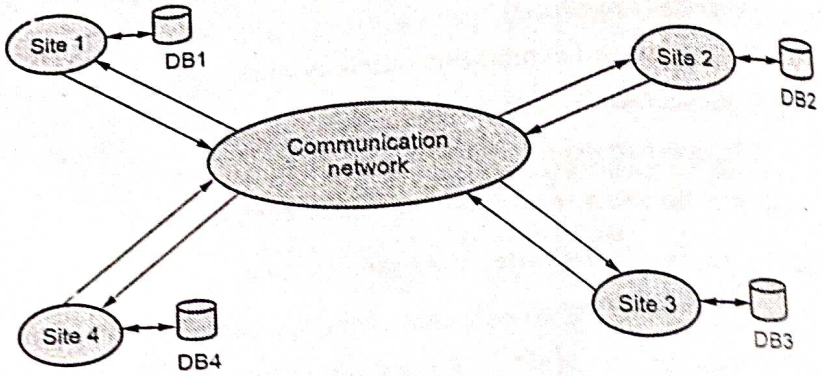


Fig. Q.5.1 Distributed database architecture

6.4 Distributed Database Design

Q.6 What are the techniques used in distributed database design.

Ans. :

- Distributed database design uses two important techniques namely fragmentation and data replication.
- Fragmentation is a technique used to break up the database into logical units. These units are assigned for storage at various sites.
- The data replication is a technique which permits certain data to be stored in more than one site.
- The information concerning data fragmentation, allocation and replication is stored in **global directory** which is accessible by Distributed Database systems as needed.

6.5 Distributed Data Storage

Q.7 What are data fragmentations ? Explain various approaches for fragmenting a relation with example.

Ans. :

- ***Concept :*** *Data fragmentation is a division of relation r into fragments r1,r2, r3,…,rn which contain sufficient information to reconstruct relation r.*
- There are two approaches of data fragmentation - (1) Horizontal fragmentation and (2) Vertical fragmentation.
  - **Horizontal fragmentation :** In this approach, each tuple of r is assigned to one or more fragments. If relation R is fragmented in r1 and r2 fragments, then to bring these fragments back to R we must use union operation. That means R = r1∪r2
  - **Vertical fragmentation :** In this approach, the relation r is fragmented based on one or more columns. If relation R is fragmented into r1 and r2 fragments using vertical fragmentation then to bring these fragments back to original relation R we must use join operation. That means R = r1⋈r2
- For example - Consider following relation r

  Student(RollNo, Marks,City)

The values in this schema are inserted as

RollNo	Marks	City
R101	55	Pune
R102	66	Chennai
R103	77	Mumbai

Fig. Q.7.1 Student table

Horizontal Fragmentation 1 :

SELECT * FROM Student WHERE Marks >50 AND City='Pune'

We will get

R101	55	Pune

Horizontal Fragmentation 2 :

SELECT * FROM Student WHERE Marks >50 AND City='Mumbai'

We will get

R103	77	Mumbai

Vertical Fragmentation 1 :

SELECT * FROM RollNo

R101
R102
R103

Vertical Fragmentation 2 :

SELECT * FROM city

Pune
Chennai
Mumbai

Q.8 Explain data replication in detail.

Ans. :

- **Concept :** Data replication means storing a copy or replica of a relation fragments in two or more sites.
- There are two methods of data replication – (1) Full replication (2) Partial replication.
  - **Full replication :** In this approach the **entire relation** is stored at **all the sites**. In this approach full redundant databases are those in which every site contains a copy of entire database.

- **Partial replication :** In this approach only some fragments of relation are replicated on the sites.

Advantages :

(1) **Availability :** Data replication facilitates increased availability of data.

(2) **Parallelism :** Queries can be processed by several sites in parallel.

(3) **Faster accessing :** The relation r is locally available at each site, hence data accessing becomes faster.

Disadvantages :

(1) **Increased cost of update :** The major disadvantage of data replication is increased cost of updated. That means each replica of relation r must be updated from all the sites if user makes a request for some updates in relation.

6.6 Distributed Transaction

Q.9 Explain two phase commit protocol

Ans. : This protocol works in two phases - i) Voting phase and ii) Decision phase.

Phase 1 : Obtaining decision or voting phase

Step 1 : Coordinator site Ci asks all participants to **prepare** to commit transaction Ti.

- Ci adds the records **<prepare T>** to the log and writes the log to stable storage.
- It then sends **prepare T** messages to all participating sites at which T will get executed.

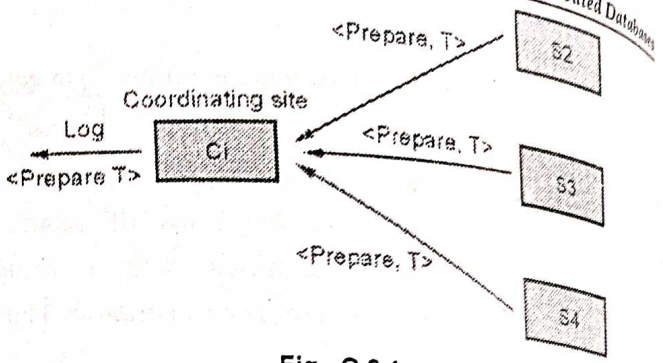


Fig. Q.9.1

Step 2 : Upon receiving message, transaction manager at participating site determines if it can commit the transaction

- If not, add a record **<no T>** to the log and send abort T message to coordinating site Ci.

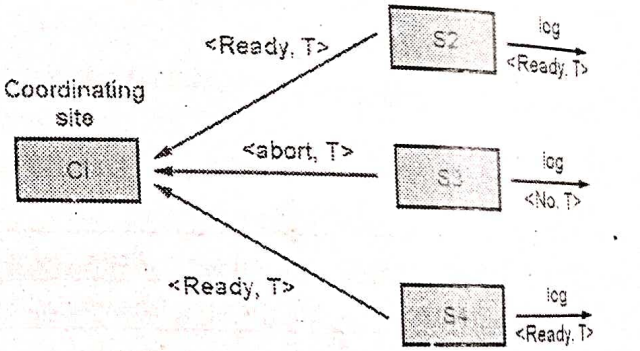


Fig. Q.9.2

- If the transaction can be committed, then :
  - Add the record **<ready T>** to the log
  - Force all records for T to stable storage
  - Send **ready T** message to Ci

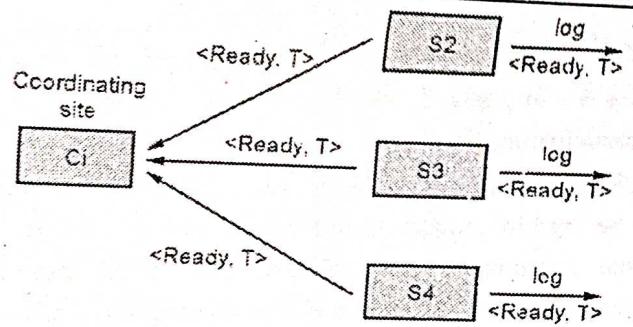


Fig. Q.9.3

Phase 2 : Recoding decision phase

- T can be committed of Ci received a ready T message from all the participating sites : otherwise T must be aborted.
- Coordinator adds a decision record, <commit T> or <abort T>, to the log and forces record onto stable storage. Once the record stable storage it is irrevocable (even if failures occur)

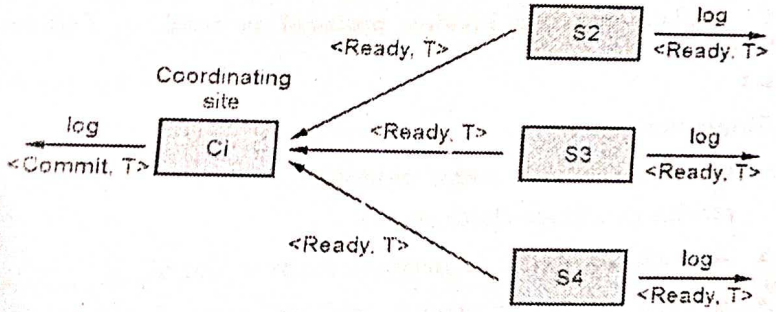
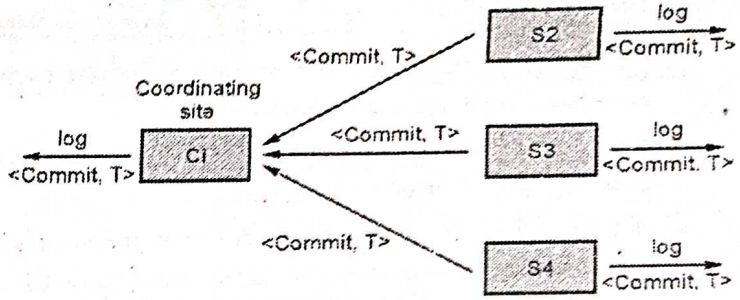


Fig. Q.9.4

- Coordinator sends a message to each participant informing it of the decision (commit or abort)
- Participants take appropriate action locally.



Failure of site

There are various cases at which failure may occur,

(1) Failure of participating sites

- If any of the participating sites gets failed then when participating site Si recovers, it examines the log entry made by it to take the decision about executing transaction.
  - If the log contains **<commit T>** record : participating site executes **redo** (T)
  - If the log contains **<abort T>** record : participating site executes **undo** (T)
  - If the log contains **<ready T>** record : participating site must consult *Coordinating site* to take decision about execution of transaction T.
    - If *T* committed, **redo** (T)
    - If *T* aborted, **undo** (T)
- If the log of participating site contains no record then that means Si gets failed before responding to **Prepare T** message from coordinating site. In this case it must **abort T**

(2) Failure of coordinator

- If coordinator fails while the commit protocol for T is executing then participating sites must take decision about execution of transaction T :

i) If an active participating site contains a <commit T> record in its log, then **T must be committed.**

ii) If an active participating site contains an <abort T> record in its log, then **T must be aborted.**

iii) If some active participating site does not contain a <ready T> record in its log, then the failed coordinator Ci cannot have decided to commit T. Can therefore abort T.

iv) If none of the above cases holds, then all participating active sites must have a <ready T> record in their logs, but no additional control records (such as <abort T> of <commit T>). In this case active sites **must wait** for coordinator site Ci to recover, to find decision.

Q.10 Explain three phase commit protocol.

Ans . :

- The three phase locking is an extension of two phase locking protocol in which eliminates the blocking problem.
- Various assumptions that are made for three phase commit protocol are –
  - No network partitioning.
  - At any point at least one site must be up.
  - At the most k sites(participating as well as coordinating) can fail.
- **Phase 1 :** This phase is similar to phase 1 of two phase protocol. That means Coordinator site Ci asks all participants to **prepare** to commit transaction Ti. The coordinator then makes the decision about commit or abort based on the response from all the participating sites.
- **Phase 2 :** In phase 2 coordinator makes a decision as in 2Phase Commit which is called the **pre-commit** decision **<Pre-commit, T>** , and records it in multiple (at least K) participating sites.

- **Phase 3 :** In phase 3, coordinator sends commit/abort message to all participating sites.
- Under three phase protocol, the knowledge of pre-commit decision can be used to commit despite coordinator site failure. That means if the coordinating site in case gets failed then one of the participating site becomes the coordinating site and consults other participating sites to know the **Pre-commit** message which they possess. Thus using this pre-commit message the decision about commit/abort is taken by this new coordinating site.
- This protocol avoids blocking problem as long as less than k sites fail.

6.7 Concurrency Control in Distributed Database

Q.11 Explain any two locking protocol is used in distributed systems.

Ans. :

(1) Single lock manager :

- In this approach, system maintains a single lock manager that resides in a single chose site Si
- All lock and unlock requests are made to site Si
- If some transaction needs to lock particular data item, then the lock request is made to site Si
- The lock manager determines whether the lock can be granted immediately. 1) If yes then the lock manger sends a message to the site which initiated the request. 2) If no, request is delayed until the lock can gets granted.

Advantages :

(1) It is simple to implement.

(2) As locks and unlocks requests are made at single site. The deadlock handling is simple.

Disadvantages :

(1) As the request for the lock is made to a single site, there occurs an overload on that site.

(2) If the site that contains the lock manager gets failed then severe vulnerability occurs in the system.

(2) Distributed lock manager

- In this approach each site maintains a local lock manger which is responsible to manage the lock and unlock requests for those data items that are stored in that site.
- When a transaction wants to lock particular data item which is not replicated and is located at site Si, then a message is sent to the lock manager at site Si for requesting the lock.

Advantages :

(1) It is easy to implement.

(2) As the locks are distributed across several sites, there is no overhead on a single site for the lock request.

Disadvantage :

(1) Deadlock detection is complicated.

Q.12 What is timestamp technique ? How it is used in distributed systems ?

Ans. :

- Timestamping is one of the technique used in distributed concurrency control mechanism.
- Each transaction must be associated with **unique timestamp**.
- In distributed systems the timestamps are generated with some **specialized technique**. It is as follows -
  - Each site generates a **unique local timestamp**, either by using logical counter or by local clock.
  - **Global unique timestamp** is obtained by concatenating unique local timestamp with **unique site identifier**.

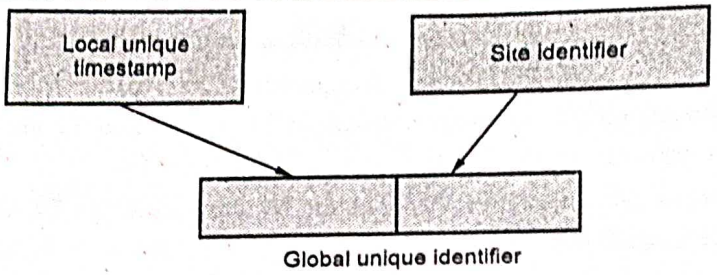


Fig. Q.12.1 Generation of unique timestamp

- The **main problem** in this approach is that - if one site generates its local timestamp at a rate faster than that of other sites, then the timestamp of this fast site will always be larger than all other sites.
- **To fix this problem -** a logical clock (LC_i) is used by each site to generate the local timestamp. This logical clock is implemented as a counter. In this approach, site S_i advances its logical clock whenever the request is received from transaction T_i with timestamp <x,y> and x is greater than current value of logical clock LC_i. In this case, the site S_i advances its logical clock to the value x + 1.

Q.13 Explain deadlock handling in distributed systems.

Ans. :

- In deadlock detection technique, the main problem in distributed system is deciding how to maintain **wait-for-graph**. The common solution to this problem is to maintain local wait for graphs for each site.
- In **centralized deadlock detection approach** the system constructs and maintains a **global wait for graph** which is union of all the local graphs in a single site : The deadlock detection coordinator.
- If the coordinator finds a cycle, it selects a victim and notifies all the sites. The sites rollback the victim transaction.

- For example - Consider the scenario in which, at site S1, transaction T2 waits for transaction T1, transaction T2 also waits for transaction T3. The transaction T5 waits for T1.
- At site S2, transaction T3 waits for T4, the transaction T4 waits for transaction T2.

Step 1 : We will construct the **local wait for graph** for both the sites S1 and S2

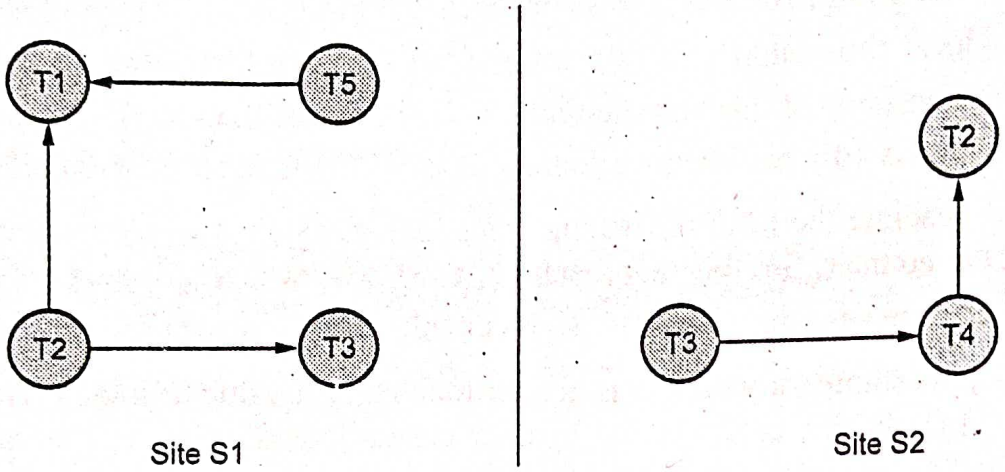


Fig. Q.13.1 Local wait for graphs for site S1 and S2

Step 2 : We will construct **global wait for graph** by combining the local wait for graphs of site S1 and S2.

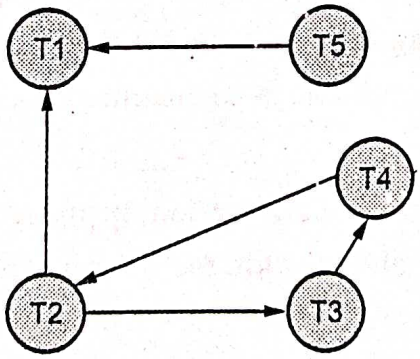


Fig. Q.13.2 Global wait for graph

Step 3 : Note that T2-T3-T4 forms a cycle in above graph. Thus these transactions in distributed system detects the dead-lock.

END...