Unit VI Advanced Databases

Database Architectures : Centralized and Client - Server Architectures, 2 Tier and 3 Tier Architecture, Introduction to Parallel Databases, Key elements of Parallel Database Processing, Architecture of Parallel Databases, Introduction to Distributed Databases, Architecture of Distributed Databases, Distributed Database Design.

Emerging Database Technologies : Introduction, No SQL Databases - Internet Databases, Cloud databases, Mobile Databases, SQLite database, XML databases.

(Chapter - 6)

TABLE OF CONTENTS

Unit III

Unit VI Advanced Databases

Database Architectures : Centralized and Client - Server Architectures, 2 Tier and 3 Tier Architecture, Introduction to Parallel Databases, Key elements of Parallel Database Processing, Architecture of Parallel Databases, Introduction to Distributed Databases, Architecture of Distributed Databases, Distributed Database Design.

Emerging Database Technologies : Introduction, No SQL Databases - Internet Databases, Cloud databases, Mobile Databases, SQLite database, XML databases.

(Chapter - 6)

TABLE OF CONTENTS

Unit III

Unit IV

Unit V

Unit VI

Unit III

3 Introduction to SQL and PL/SQL

3.1 : Introduction to SQL

Q.1 What is SQL ? Enlist the characteristics of SQL.

Ans. : • SQL stands for Structured Query Language.

- It is the language of databases and almost all companies use databases to store their data.
- 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.

3.2 : SQL Data Types and Literals

Q.2 Discuss various MYSQL data types. ☞ **[SPPU : Nov.-17, Marks 5]**

Ans. : Various data types used in SQL are -

1) Numeric data types

- Integer numbers : INT, INTEGER, SMALLINT, BIGINT
- Floating-point (real) numbers : REAL, DOUBLE , FLOAT

- Fixed-point numbers : DECIMAL(n,m), DEC(n,m), NUMERIC(n,m), NUM(n,m)

2) Character-string data types

- Fixed length : CHAR(n), CHARACTER(n)
- Varying length : VARCHAR(n), CHAR VARYING(n), CHARACTER VARYING(n), LONG VARCHAR

3) Large object data types

- Characters : CLOB, CHAR LARGE OBJECT , CHARACTER LARGE OBJECT
- Bits : BLOB, BINARY LARGE OBJECT

4) Boolean data type

- Values of TRUE or FALSE or NULL

5) DATE data type

- Ten positions
- Components are YEAR, MONTH, and DAY in the form YYYY-MM-DD

6) Additional Data type

a) TIMESTAMP data type

It includes the DATE and TIME fields.

b) INTERVAL data type

It specifies a relative value that can be used to increment or decrement an absolute value of a date, time, or timestamp.

3.3 : DDL and DML

Q.3 Explain different DDL and DML commands with example.

[SPPU : Aug.-17, Marks 5]

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

Q.4 What is the difference between DDL and DML ?

Ans. :

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.

- Fixed-point numbers : DECIMAL(n,m), DEC(n,m), NUMERIC(n,m), NUM(n,m)

2) Character-string data types

- Fixed length : CHAR(n), CHARACTER(n)
- Varying length : VARCHAR(n), CHAR VARYING(n), CHARACTER VARYING(n), LONG VARCHAR

3) Large object data types

- Characters : CLOB, CHAR LARGE OBJECT , CHARACTER LARGE OBJECT
- Bits : BLOB, BINARY LARGE OBJECT

4) Boolean data type

- Values of TRUE or FALSE or NULL

5) DATE data type

- Ten positions
- Components are YEAR, MONTH, and DAY in the form YYYY-MM-DD

6) Additional Data type

a) TIMESTAMP data type

It includes the DATE and TIME fields.

b) INTERVAL data type

It specifies a relative value that can be used to increment or decrement an absolute value of a date, time, or timestamp.

3.3 : DDL and DML

Q.3 Explain different DDL and DML commands with example.

☞ **[SPPU : Aug.-17, Marks 5]**

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

Q.4 What is the difference between DDL and DML ?

Ans. :

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.

Changes made by DDL commands cannot be rolled back.	Changes by DML commands can be rolled back.
Example – CREATE, ALTER, DROP commands.	Example – SELECT, INSERT, UPDATE, and DELETE commands.

3.4 : SQL Operators

Q.5 Explain the use of percent and underscore operators in SQL.

Ans. : • Pattern matching can also be performed on strings using two types of special characters -

- Percent(%) : It matches zero, one or multiple characters.
- Underscore(_) : The _ character matches any single character.

- The percentage and underscore can be used in combinations.
- Patterns are case sensitive. That means upper case characters do not match lowercase characters or vice versa.
- **For instance :**
  - 'Data%' matches any string beginning with "Data", For instance it could be with "Database", "DataMining","DataStructure"
  - '___' matches any string of exactly three characters.
  - '___%'matches any string of at least length 3 characters.

Q.6 What is the use of between operator in SQL ?

Ans. : • The **between** operator can be used to simplify the where clause which is used to denote the value be less than or equal to some value and greater than or equal to some other value.

- For example – of we want the names of the students whose marks are between 80 and 90 then SQL statement will be

```
SELECT name
FROM Students
WHERE marks BETWEEN 80 and 90;
```


3.5 : Tables

Q.7 Write the syntax for following SQL commands : i) create table ii) alter table iii) drop table iv) insert v) delete vi) update.

☞ [SPPU : June-22, Marks 6]

Ans. :

i) Create Table : 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 TABLE person_details (
        AadharNo int,
        FirstName VARCHAR(20),
        MiddleName VARCHAR(20),
        LastName VARCHAR(20),
        Address VARCHAR(30),
        City VARCHAR(10)
        )
```

The blank table will be created with following structure

Person_details

AadharNo	FirstName	MiddleName	LastName	Address	City

ii) Alter Table :

Syntax for Adding columns

```
ALTER TABLE table_name
ADD column_name datatype;
```

Example

Consider following table

AadharNo	FirstName	MiddleName	LastName	Address	City
111	AAA	BBB	CCC	M.G. Road	Pune
222	DDD	EEE	FFF	Shivaji nagar	Pune
333	GGG	HHH	III	Chandani chowk	Delhi
444	JJJ	KKK	LLL	Viman nagar	Mumbai

If we execute following command

```
ALTER TABLE Customers
ADD Email varchar(30);
```

Then the result will be as follows -

AadharNo	FirstName	MiddleName	LastName	Address	City	Email
111	AAA	BBB	CCC	M.G. road	Pune	NULL
222	DDD	EEE	FFF	Shivaji nagar	Pune	NULL
333	GGG	HHH	III	Chandani chowk	Delhi	NULL
444	JJJ	KKK	LLL	Viman nagar	Mumbai	NULL

iii) Drop Table :

- 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.

iv) Insert :

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

Syntax

INSERT INTO table_name (col_1,col_2,...,col_n)
VALUES ($value_1$,$value_2$,...,$value_n$)

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

v) Delete :

- 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
```

The result will be -

AadharNo	FirstName	MiddleName	LastName	Address	City
111	AAA	BBB	CCC	M.G. road	Pune
222	DDD	EEE	FFF	Shivaji nagar	Pune
444	JJJ	KKK	LLL	Viman nagar	Mumbai

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;
```

vi) Update :

- For modifying the existing record of a table, update query is used.

Syntax

```
UPDATE table_name
SET col1=value1, col2=value2,...
WHERE condition;
```

The WHERE command is used to specify some condition. Based on this condition the data present in the table can be displayed or can be updated or deleted.

Example

Consider following table

Person_details table

AadharNo	FirstName	MiddleName	LastName	Address	City
111	AAA	BBB	CCC	M.G. Road	Pune
222	DDD	EEE	FFF	Shivaji nagar	Pune
333	GGG	HHH	III	Chandani Chowk	Delhi
444	JJJ	KKK	LLL	Viman Nagar	Mumbai

If we execute following query

```
UPDATE person_details
SET city='Chennai'
WHERE AadharNo=333
```

The result will be

AadharNo	FirstName	MiddleName	LastName	Address	City
111	AAA	BBB	CCC	M.G. Road	Pune
222	DDD	EEE	FFF	Shivaji nagar	Pune

333	GGG	HHH	III	Chandani chowk	Chennai
444	JJJ	KKK	LLL	Viman nagar	Mumbai

Q.8 Explain on delete cascade command with suitable example.

☞ **[SPPU : June-22, Marks 5]**

Ans. : • ON DELETE CASCADE clause is used to **automatically remove** the matching records from the child table when we delete the rows from the parent table. It is a kind of referential action related to the foreign key.

- For example - we have to create two tables Employee and Dept as follows -

Employee Table

EmpID	LastName	FirstName	Age
1	Khanna	Rajesh	30
2	Joshi	Sharman	23
3	Kapoor	Tushar	20

Dept Table

DeptID	DeptName	EmpID
1	Accounts	3
2	Production	3
3	Sales	2
4	Purchase	1

- While creating the child table **Dept** we can apply the ON DELETE CASCADE clause as follows -

```
CREATE TABLE DEPT (
    DeptID int
    DeptName VARCHAR(20),
    EmpID int,
    PRIMARY KEY(DeptID),
    FOREIGN KEY(EmpID) REFERENCES
    EMPLOYEE(EmpID) ON DELETE CASCADE
    );
```

- Let us delete data from the parent table Employee. To do this, execute the following statement :

```
DELETE FROM Employee WHERE EmpID = 1;
```

- The above statement will delete the employee records whose EmpID = 1 and referencing data into the child table **Dept**, Hence both the tables will be -

Employee Table

EmpID	LastName	FirstName	Age
2	Joshi	Sharman	23
3	Kapoor	Tushar	20

Dept Table

DeptID	DeptName	EmpID
1	Accounts	3
2	Production	3
3	Sales	2

Q.9 Describe DROP TABLE command of SQL with both the options CACADE 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 : Views

Q.10 Explain the concept of view along with its operations. ☞ **[SPPU : Nov.-19, Marks 5]**

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
FROM Employee E
WHERE E.Salary > 10000
```

The Output will be -

EmpID	EName
102	Madhura
103	Poonam
105	Monika

ii) **Creating a view from multiple table :** In this example we will create a view from two tables Employee and Department.

Employee

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

Department

EmpID	DName
101	Accounts

104	Sales
105	Sales

Now we need to create a view named Employee_dept_Details in which the names and salary of employees belonging to Sales department is displayed. The SQL statement will then be -

```
CREATE VIEW Employee_dept_Details(EName,Salary) AS
    SELECT E.EName,E.Salary
    FROM Employee E, Department D
    WHERE E.EmpID=D.EmpID AND D.DName='Sales'
```

The output will be

EName	Salary
Sharda	15000
Monika	7000

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;
```

Dropping View

For deleting a view, the DROP command is used.

Syntax

```
DROP VIEW view_name;
```

Example

```
DROP VIEW Employee_dept_Details;
```

Q.11 What is the difference between view and table ?

Ans. : • Table is a database object that contains the data in row and column form. View is also database object and it is virtual table which is built on the top of the other tables.

- The table contains the data while view does not hold data itself.
- A table is designed with a limited number of columns and an unlimited number of rows while a view is designed as a virtual table that is extracted from a database.

3.7 : SELECT Query and Clauses

Q.12 Explain (1) Order By (2) Group By (3) Having clause with suitable example.

Ans. : (1) Order By

- Many times we need the records in the table to be in sorted order.
- If the records are arranged in increasing order of some column then it is called **ascending order.**
- If the records are arranged in decreasing order of some column then it is called **descending order.**
- 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.

Example

Consider following table

AadharNo	FirstName	MiddleName	LastName	Address	City
111	AAA	BBB	CCC	M.G. road	Pune
222	DDD	EEE	FFF	Shivaji nagar	Pune
333	GGG	HHH	III	Chandani chowk	Delhi
444	JJJ	KKK	LLL	Viman nagar	Mumbai

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

The above query will result in

AadharNo	FirstName	MiddleName	LastName	Address	City
444	JJJ	KKK	LLL	Viman nagar	Mumbai
333	GGG	HHH	III	Chandani chowk	Delhi
222	DDD	EEE	FFF	Shivaji nagar	Pune
111	AAA	BBB	CCC	M.G. road	Pune

(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.
- Optionally it is used in conjunction with aggregate functions.

- The queries that contain the GROUP BY clause are called grouped queries.
- This query returns a single row for every grouped item.
- **Syntax :**

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

- **Example :** Consider the Student table as follows -

sid	sname	marks	city
1	AAA	60	Pune
2	BBB	70	Mumbai
3	CCC	90	Pune
4	DDD	55	Mumbai

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

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

The result will be as follows -

SUM(marks)	city
150	Pune
125	Mumbai

(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.

- WHERE and HAVING can be in the same query.
- **Syntax :**

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

- **Example :** Consider the Student table as follows -

sid	sname	marks	city
1	AAA	60	Pune
2	BBB	70	Mumbai
3	CCC	90	Pune
4	DDD	55	Mumbai
5	EEE	84	Chennai

Query : Find the total marks of each student in the city named 'Pune' and 'Mumbai' only

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

- The result will be as follows -

SUM(marks)	city
150	Pune
125	Mumbai

3.8 : Set Operations

Q.13 Explain various types of set operations in SQL with suitable example.

Ans. : Set is a collection of elements on which union, intersection and difference operations can be performed.

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 : Aggregate Functions

Q.14 Explain with suitable example SQL aggregate functions.

☞ **[SPPU : June-22, Dec.-22, Marks 6]**

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 :
- The aggregate functions that accept an expression parameter can be modified by the keywords DISTINCT or ALL. If neither is specified, the result is the same as if ALL were specified.

DISTINCT	Modifies the expression to include only distinct values that are not NULL
ALL	Includes all rows where expression is not NULL

- 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
SELECT COUNT(ALL id)
```

FROM Test

Output

3

- 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.15 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]**

Ans. :

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


3.10 : Join Operations

Q.16 Explain various types of outer join operations with example.

☞ **[SPPU : Nov.-17, Marks 5, Dec.-22, Marks 6]**

OR Explain different Join operations in Relational Algebra with suitable example.

☞ **[SPPU : May-19, Nov.-19, June-22, Marks 6]**

Ans. : Various types of join operations are -

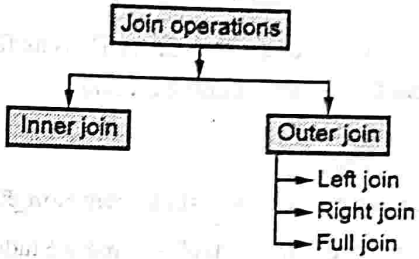


Fig. Q.16.1 Types of join operations

Example : Consider two tables for using the joins in SQL. Note that **cid** is common column in following tables.

Student

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

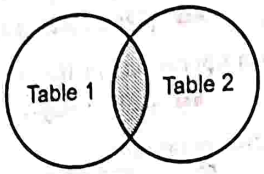
Reserve

cid	cname
101	Pune
102	Mumbai
103	Chennai

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
```

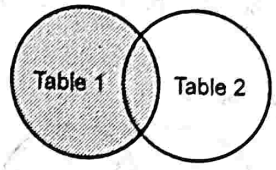
The result will be

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
```

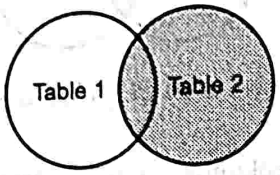
The result will be

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
```

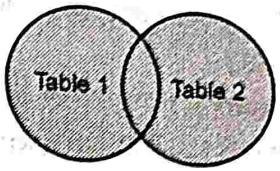
The result will be –

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 : Nested Queries

Q.17 Explain different types of nested queries along with example.

Ans. : There are two types of nested queries :

(i) Independent Query (ii) Corelated Query

i) Independent Query :

- In independent nested queries, query execution starts from innermost query to outermost queries.
- The execution of inner query is independent of outer query, but the result of inner query is used in execution of outer query.
- Various operators like IN, NOT IN, ANY, ALL etc are used in writing independent nested queries.
- For example - Consider three tables namely **Student, City and Student_City** as follows -

Student

sid	sname	phone
1	Ram	1111
2	Shyam	2222
3	Seeta	3333
4	Geeta	4444

City

cid	cname
101	Pune
102	Mumbai
103	Chennai

Student_City

sid	cid
1	101
1	103
2	101
3	102
4	102
4	103

- **Example 1 -** If we want to find out **sid** who live in city 'Pune' or 'Chennai'. We can then write independent nested query using IN operator. Here we can use the IN operator allows you to specify multiple values in a WHERE clause. The IN operator is a shorthand for multiple OR conditions.

Step 1 : Find cid for cname='Pune' or 'Chennai'. The query will be

```
SELECT cid
FROM City
WHERE cname='Pune' or 'Chennai'
```

Step 2 : Using cid obtained in step 1 we can find the sid. The query will be

```
SELECT sid
FROM Student_City
WHERE cid IN
   (
   SELECT cid
      FROM City
      WHERE cname='Pune' or cname='Chennai'
   )
```

The inner query will return a set with members 101 and 103 and outer query will return those **sid** for which **cid** is equal to any member of set (101 and 103 in this case). So, it will return 1, 2 and 4.

Example 2 : If we want to find out sname who live in city 'Pune' or 'Chennai'.

```
SELECT sname
FROM Student
WHERE sid IN
   (  SELECT sid
      FROM Student_City
      WHERE cid IN
      (SELECT cid
       FROM City
       WHERE cname='Pune' or cname='Chennai'
      )
   )
```

ii) Co-related Query :

In co-related nested queries, the output of inner query depends on the row which is being currently executed in outer query. For example

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
)
```

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.

3.12 : Database Modification using SQL Insert, Update, Delete Queries

Q.18 Write SQL statements for the following (any five)
Consider the following database
pilot (pid, pname)
flight (fid, ftype, capacity)
route (pid, fid, from_city, to_city)
i) List the details of flights having capacity more than 300.
ii) List the flights between 'Surat' and 'Mumbai'.
iii) List the names of the pilots who fly from 'Pune'.
iv) List the route on which, pilot named 'Mr Kapoor' flies
v) List the pilots whose names, starts with letter 'A' %' but does not end with letter 'A'.
vi) List the name of pilots who fly 'boing 737' type of flights.

Ans. : (i) List the details of flights having capacity more than 300.

```
SELECT * FROM flight
WHERE capacity>300
```

(ii) List the flights between 'Surat' and 'Mumbai'.

```
SELECT fid
FROM flight, route
WHERE
filight.fid=route.fid AND
route.from_city='Surat' AND route.to_city='Mumbai';
```

(iii) List the names of the pilots who fly from 'Pune'.

```
SELECT pname
FROM pilot,route
WHERE pilot.pid=route.pid
AND route.from_city='Pune';
```

(iv) List the route on which, pilot named 'Mr Kapoor' flies.

```
SELECT from_city, to_city
FROM route, pilot
WHERE pilot.pid=route.pid
AND pilot.pname='Mr.Kapoor';
```

(v) List the pilots whose names, starts with letter 'A' %' but does not end with letter 'A'.

```
SELECT pname
FROM pilot
WHERE pname LIKE 'A%' MINUS
SELECT pname FROM pilot
WHERE pname LIKE '%A';
```

(vi) List the name of pilots who fly 'boing 737' type of flights.

```
SELECT pname
FROM pilot, flight, route
WHERE flight.ftype = 'boing 737' AND
pilot.pid = route.pid AND
route.fid = flight.fid;
```

Q.19 Consider the relation Database.
Person (SSN, Name, city)
Car (License_no, year, Model, SSN)
Accident(drive_no, SSN, license_no, accidentyear, damage_Amt)
Query :
1. Find out total no of cars that had accident in 1988.
2. Find the Name of driver who did not have an accident in 'Delhi'.
3. Find the car, who don't have total damage of more than ₹ 1000.
4. Find the cars sold in 2006 and whose owner are from 'Vadodara'
5. How many different models of car are used by 'Mr.abc'
6. Find the lucky persons who have not met any accident yet.

Ans. : 1. Find out total no of cars that had accident in 1988.

```
SELECT count(License_no)
FROM Car, Accident
WHERE Acccident.accidentyear=1988
```

2. Find the Name of driver who did not have an accident in 'Delhi'.

```
SELECT Name
FROM Person,Accident
WHERE Person.SSN=Accident.SSN AND Peson.city<> 'Delhi'
```

3. Find the car, who don't have total damage of more than ` 1000.

```
SELECT License_no
FROM Car, Accident
WHERE Accident.damage_amt<1000 AND
Car.License_no=Accident.licence_no;
```

4. Find the cars sold in 2006 and whose owner are from 'Vadodara'

```
SELECT license_no
FROM Car, Person
WHERE Car.SSN = Person. SSN AND
Car.year = 2006 and Person.city = 'Vadodara';
```

5. How many different models of car are used by 'Mr.abc'

```
SELECT count(model)
FROM Car, Person
WHERE Car.SSN = Person. SSN AND
Person.Name = 'Mr.abc';
```


6. Find the lucky persons who have not met any accident yet.

```
SELECT Name
FROM Person
WHERE person.SSN NOT IN (SELECT SSN FROM Accident);
```

Q.20 We have following relations.
EMP (empno, ename, jobtitle, manager no, hiredate, sal, comm, dept no)
DEPT (dept no, dname, loc)
i) The employees who are getting salary greater than 3000 for those persons belongings to the department 20
ii) Employees who are not getting any commission.
iii) Find how many job titles are available in employee table.
iv) Display total salary spent for each job category.
v) Display number of employees working in each department and their department name.
vi) List ename whose manager is NULL.
vii) List all employee names and their salaries, whose salary lies between 1500/- and 3500/- both inclusive.

Ans. : i) The employees who are getting salary greater than 3000 for those persons belongings to the department 20

```
SELECT ename
FROM EMP
WHERE EMP.sal>3000 AND EMP.dept_no=20
```

ii) Employees who are not getting any commission.

```
SELECT ename
FROM EMP
WHERE EMP.comm IS  NULL;
```

iii) Find how many job titles are available in employee table.

```
SELECT count(jobtitle)
FROM EMP
```

iv) Display total salary spent for each job category.

```
SELECT ename,SUM(sal)
FROM EMP
GROUP BY jobtitle;
```

v) Display number of employees working in each department and their department name.

```
SELECT COUNT(EMP.eno), DEPT.dname
FROM EMP, DEPT
WHERE EMP.dept_no=DEPT.dept_no
GROUP BY DEPT.dept_no;
```

vi) List ename whose manager is NULL.

```
SELECT ename
FROM EMP
WHERE manager_no IS NULL;
```

vii) List all employee names and their salaries, whose salary lies between 1500/- and 3500/- both inclusive.

```
SELECT ename, sal
FROM EMP
WHERE sal>=1500 AND sal<=3500;
```

Q.21 We have following relations :
Supplier (S#, sname, status, city)
Parts (P#, pname, color, weight, city)
SP(S#, P#, quantity)
Answer the following queries in SQL,
i) Find name of parts whose color is 'red'.
ii) Find parts name whose weight less than 10 kg.
iii) Find all parts whose weight from 10 to 20 kg.
iv) Find average weight of all parts.
v) Find S# of supplier who supply part 'p2'.
vi) Find name of supplier who supply maximum parts.
vii) Sort the parts table by pname.

Ans. : i) Find name of parts whose color is 'red'.

```
SELECT pname
FROM Parts
WHERE Parts.color = 'red';
```

ii) Find parts name whose weight less than 10 kg.

```
SELECT pname
FROM Parts
WHERE Parts.weight < 10;
```


iii) Find all parts whose weight from 10 to 20 kg.

```
SELECT pname, weight
FROM Parts
WHERE Parts.weight BETWEEN 10 AND 20;
```

iv) Find average weight of all parts.

```
SELECT AVG (weight)
FROM Parts;
```

v) Find S# of supplier who supply part 'p2'.

```
SELECT S#
FROM SP
WHERE p# = 'p2';
```

vi) Find name of supplier who supply maximum parts.

```
SELECT sname, MAX(quantity)
FROM Supplier, Parts,SP
WHERE Supplier.S# = SP.S#
AND SP.P# = P.P#
GROUP BY sname;
```

vii) Sort the parts table by pname.

```
SELECT *
FROM Parts
ORDER BY pname;
```

Q.22 C Write queries for the following tables :
T1 (Empno, Ename, Salary, Designation)
T2 (Empno, Deptno.)
1) Display all the details of the employee whose salary is lesser than 10 K.
2) Display the Deptno in which employee Seeta is working.
3) Add a new column Deptname in table T2.
4) Change the designation of Geeta from 'Manager' to 'Senior Manager'.
5) Find the total salary of all the employees.
6) Display Empno, Ename, Deptno and Deptname
7) Drop the table T1.

Ans. : 1) Display all the details of the employee whose salary is lesser than 10 K.

```
SELECT *
FROM T1
WHERE Salary<10000
```

2) Display the Deptno in which employee Seeta is working.

```
SELECT Deptno
FROM T2, T1
WHERE T1.Empno=T2.Empno AND T1.Ename='Seeta';
```

3) Add a new column Deptname in table T2.

```
ALTER TABLE T2
ADD (Deptname VARCHAR(20));
```

4) Change the designation of Geeta from 'Manager' to 'Senior Manager'.

```
UPDATE T1
SET Designation ='Senior Manager'
WHERE Ename='Geeta';
```

5) Find the total salary of all the employees.

```
SELECT SUM(Salary)
FROM T1
```

6) Display Empno, Ename, Deptno and Deptname

```
SELECT T1.Empno,T1.Ename, T2.Deptno, T2.Deptname
FROM T1, T2
```

7) Drop the table T1.

```
DROP TABLE T1
```

Q.23 Consider following database : Student (Roll_no, Name, Address) Subject (Sub_code, Sub_name) Marks (Roll_no, Sub_code, marks) Write following queries in SQL :

i) Find average marks of each student, along with the Roll_no of student of subject code 'CE2412'.

ii) Find how many students have failed in the subject "DBMS".

iii) Construct suitable view on above schema.

☞ **[SPPU : June-22, Marks 6]**

Ans. : i) SELECT Student.Roll_no, AVG(Marks.marks) AS avg_marks

```
FROM Student
WHERE Student.Roll_no = Marks.Roll_no
```



```
AND Marks.Sub_code = Subject.Sub_code AND Subject.Sub_code =
'CE2412'
```

II) SELECT COUNT(*)

```
FROM Student
JOIN Marks ON Student.Roll_no = Marks.Roll_no
JOIN Subject ON Marks.Sub_code = Subject.Sub_code
WHERE Subject.Sub_name = 'DBMS' AND Marks.marks < 40
```

iii) CREATE VIEW student_avg_marks AS

```
SELECT Student.Roll_no, AVG(Marks.marks) AS avg_marks
FROM Student
WHERE Student.Roll_no = Marks.Roll_no
GROUP BY Student.Roll_no
```

Q.24 Consider the following book relation.
Book (Book_id, Title, Author, Publisher, Year, Price)
Write relational algebra expression for the following.
i) Display all book title with authors and price.
ii) Display the titles of book having price greater than 300.
iii) Display books publish in year 2000.
iv) Display all books published by 'PHP' with price greater than 300.

☞ **[SPPU : June-22, Marks 5]**

Ans. :

i) $\Pi_{\text{Author, Price}}$ (Book)

ii) Π_{title} ($\sigma_{\text{Price} > 300}$ (Book))

iii) Π_{title} ($\sigma_{\text{Year} = 2000}$ (Book))

iv) Π_{title} ($\sigma_{\text{Publisher} = \text{'PHP'} \text{ and Price} > 300}$ (Book))

3.13 : Stored Procedure

Q.25 Write and explain SQL function and procedures with sample example. ☞ **[SPPU : June-22, Marks 6]**

Ans. : Step 1 : Create a procedure as follows

```
CREATE OR REPLACE PROCEDURE MyMessage ← Creation of procedure
IS
```

```
BEGIN
  dbms_output.put_line('Welcome friend!!!');
END;
/

EXECUTE MyMessage;
```

← **This is for executing the procedure**

Step 2 : The output will be as follows -

Welcome friend!!!

Another way of execution of procedure

```
CREATE OR REPLACE PROCEDURE MyMessage
IS
BEGIN
  dbms_output.put_line('Welcome friend!!!');
END;
/
BEGIN
  MyMessage;
END;
/
```

← **Creation of procedure**

← **This is for calling the procedure**

3.14 : Cursor

Q.26 What is cursor ? Explain types of cursors with example.

☞ **[SPPU : Aug.-17, Marks 5, Dec.-22, Marks 6]**

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.
- There are two types of cursors -
  - Implicit cursor
  - Explicit cursor

(1) Implicit cursor

- Whenever Oracle executes an SQL statement such as SELECT INTO, INSERT, UPDATE, and DELETE, it creates an implicit cursor.
- The Implicit Cursor is the Default Cursor in PL/SQL block.
- Orcale provides some attributes known as Implicit cursor's attributes to check the status of DML operations. Some of them are as follows -
  - **%ROWCOUNT :** It Returns the number of rows influenced by an UPDATE, DELETE or an INSERT statement.
  - **%NOTFOUND :** It Returns TRUE if an UPDATE, DELETE or an INSERT statement influenced zero rows or a SELECT INTO statement returned no rows. Else, it Returns a FALSE.
  - **%FOUND :** It Returns TRUE if an UPDATE, DELETE or an INSERT statement influenced one or more rows or a SELECT INTO statement returned one or more rows. Else, it returns FALSE.
  - **%ISOPEN :** It Returns FALSE for Implicit cursors as Oracle closes the PL/SQL cursor by default after processing its associated PL/SQL statement.

Programming Example : Following is an example program that uses attributes of implicit cursor to demonstrate number of rows affected when SELECT query is executed for Student table

```
SET SERVEROUTPUT ON;
DECLARE
 s_name student.name%type;
BEGIN
  SELECT name INTO s_name
  FROM student
```

```
 WHERE marks>=70;
 DBMS_OUTPUT.PUT_LINE('Number of rows processed: '||sql%rowcount);
END;
/
```

Output

Number of rows processed:5

(2) Explicit cursor

- 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 cursor_name
IS
SELECT statement
```

For example

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

The explicit cursor works in four stages -

1) **Declaration of cursor :** The declaration of cursor is in declaration section of PL/SQL block. The name of the cursor requires to be defined along with the SELECT Statement.

Syntax

CURSOR cursor-name IS

SELECT statement;

2) **Open the cursor :** Opening the Cursor also means allocating the memory for the Cursor in the Context Area which thereby makes it sufficient to fetch and store records in it.

Syntax

OPEN cursor-name;

3) **Fetch the cursor :** Fetching the Cursor involves retrieval of data using the Fetch statement. It is used to help the Cursor process and access records or rows at a time.

Syntax

FETCH cursor_name INTO variable_list;

4) **Close the cursor :** Once the work associated for a Cursor to be completed is accomplished, it is necessary to release the Allocated Memory of the Cursor to let other tasks occupy memory.

Syntax

Close cursor_name;

- **Example :** The cursor is created to display the names of students who have scored more than 70 marks. The name of the cursor is **My_Cursor**. The SELECT query associated with it is

  SELECT name FROM STUDENT WHERE Marks >70;

```
DECLARE
 s_name VARCHAR2(30);
 --Declare Cursor
 CURSOR My_Cursor IS
 SELECT name FROM Student
 WHERE Marks>70;
BEGIN
 OPEN My_Cursor;
 Loop
  FETCH My_Cursor INTO s_name;
  DBMS_OUTPUT.PUT_LINE(s_name);
  EXIT WHEN My_Cursor %NOTFOUND;
 END Loop;
 CLOSE My_Cursor;
END;
```

Q.27 Why are cursors necessary in embedded SQL ?

☞ **[SPPU : Nov.-18, Marks 5]**

Ans. : Refer Q.26

3.15 : Triggers

Q.28 What is trigger ? Explain trigger with suitable example.

☞ **[SPPU : Aug.-17, Nov.-19, Marks 5]**

Ans. : • A trigger is a procedure that is automatically invoked by the DBMS in response to specified changes to the database.

- It is specified by DBA.
- A trigger description contains **three parts :**

  i) **Event :** A change to the database that activates the trigger.

  ii) **Condition :** A query or test that is run when the trigger is activated. It is a true false kind of statement.

  iii) **Action :** A procedure that is executed when the trigger is activated and its condition is true.
- Trigger is executed when the database is modified in a way that matches the event specification.
- An insert, delete, or update statement could activate a trigger.
- **Example :** The trigger called **init_Stud** initializes a counter variable before every execution of an INSERT statement that adds tuples to the Students relation. The trigger called **incr_Stud** increments the counter for each inserted tuple that satisfies the condition age < 18.

```
CREATE TRIGGER init_Stud ←——— Event
BEFORE INSERT ON Students
   DECLARE
         count INTEGER;
   BEGIN ←— Action
     count := 0;
   END
CREATE TRIGGER incr_Stud AFTER INSERT ON Students ←— Event
```



```
WHEN (new.age < 18) ←—— Condition is 'new' is just-inserted tuple
   FOR EACH ROW
   BEGIN ←—— Action; a procedure
     count := count + 1;
   END
```

3.16 : Programmatic SQL

Q.29 Explain embedded and dynamic SQL.

☞ **[SPPU : Nov.-17,19, Marks 5]**

Ans. : Embedded SQL :

- The programing module in which the SQL Statements are embedded is called Embedded SQL module.
- It is possible to embed SQL statements inside the programming language such as C, C++, PASCAL,Java and so on.
- It allows the application languages to communicate with DB and get requested result.
- The high level languages which supports embedding SQLs within it are also known as **host language.**
- An embedded SQL program must be processed by a special preprocessor prior to compilation. The preprocessor replaces embedded SQL requests with host-language declarations and procedure calls that allow runtime execution of the database accesses. Then, the resulting program is compiled by the host-language compiler. This is the main distinction between embedded SQL and JDBC or ODBC.

Embedding SQL Query

- Similarly we can embed the SQL query using SELECT, INSERT, DELETE, or UPDATE statement in the host language. Again the query must begin with the command EXEC SQL. For example –

```
EXEC SQL SELECT CustID, SalesPerson, Status
   FROM Orders
    WHERE OrderID = :OrderID
    INTO :CustID, :SalesPerson, :Status;
```

Example of Embedded SQL

Following program prompts the user for an order number, retrieves the customer number, salesperson, and status of the order, and displays the retrieved information on the screen.

```
int main() {
  EXEC SQL INCLUDE SQLCA;
  EXEC SQL BEGIN DECLARE SECTION;
    int OrderID;              /* Employee ID (from user)        */
    int CustID;               /* Retrieved customer ID          */
    char SalesPerson[10]      /* Retrieved salesperson name     */
    char Status[6]            /* Retrieved order status         */
  EXEC SQL END DECLARE SECTION;

  /* Set up error processing */
  EXEC SQL WHENEVER SQLERROR GOTO query_error;
  EXEC SQL WHENEVER NOT FOUND GOTO bad_number;

  /* Prompt the user for order number */
  printf ("Enter order number: ");
  scanf_s("%d", &OrderID);

  /* Execute the SQL query */
  EXEC SQL SELECT CustID, SalesPerson, Status
    FROM Orders
    WHERE OrderID = :OrderID
    INTO :CustID, :SalesPerson, :Status;
  /* Display the results */
  printf ("Customer number:  %d\n", CustID);
  printf ("Salesperson: %s\n", SalesPerson);
  printf ("Status: %s\n", Status);
  exit();

query_error:
  printf ("SQL error: %ld\n", sqlca->sqlcode);
  exit();
bad_number:
  printf ("Invalid order number.\n");
  exit();
}
```


Dynamic SQL :

- Dynamic SQL is a programming technique which allows to build the SQL statements dynamically at runtime.
- Dynamic SQL statements are not embedded in the source program but stored as strings of characters that are manipulated during a program's runtime.
- These SQL statements are either entered by a programmer or automatically generated by the program.
- Dynamic SQL statements also may change from one execution to the next without manual intervention.
- Dynamic SQL facilitates automatic generation and manipulation of program modules for efficient automated repeating task preparation and performance.
- Dynamic SQL facilitates the development of powerful applications with the ability to create database objects for manipulation according to user input.
- The simplest way to execute a dynamic SQL statement is with an EXECUTE IMMEDIATE statement. This statement passes the SQL statement to the DBMS for compilation and execution.

Q.30 Give the difference between embedded and dynamic SQL.

Ans. :

Sr. No.	Embedded SQL	Dynamic SQL
1.	SQL statements are compiled at coompile time.	SQL statements are compiled at run time.
2.	It is more efficient.	It is less efficient.
3.	It is less flexible.	It is more flexible.
4.	It is used in the situations where data is distributed uniformly	It is used in situations where data is distributed non uniformly.

END...

Unit IV

4 Database Design and Query Processing

Part I : Relational Databases Design

4.1 : Purpose of Normalization

Q.1 Explain the design guideline for relational schema.

Ans. : There are four informal measures for relational schema and those are -

(1) Semantics of the attributes

(2) Reducing the redundant value in tuple

(3) Reducing NULL values in tuple

(4) Disallowing generation of spurious tuples

4.2 : Data Redundancy and Update Anomalies

Q.2 Explain insertion, deletion and modification anomalies with example.

☞ **[SPPU : Nov.-19 (End Sem), Marks 5]**

Ans. : Redundancy is at **the root of several problems** associated with relational schemas.

Problems caused by redundancy : Following problems can be caused by redundancy -

i) **Redundant storage :** Some information is stored repeatedly.

ii) **Update anomalies :** If one copy of such repeated data is updated then inconsistency is created unless all other copies are similarly updated.

iii) **Insertion anomalies :** Due to insertion of new record repeated information get added to the relation schema.

iv) **Deletion anomalies :** Due to deletion of particular record some other important information associated with the deleted record get deleted and thus we may lose some other important information from the schema.

Example : Following example illustrates the above discussed anomalies or redundancy problems.

Consider following Schema in which all possible information about Employee is stored.

EmpID	EName	Salary	DeptID	DeptName	DeptLoc
1	AAA	10000	101	XYZ	Pune
2	BBB	20000	101	XYZ	Pune
3	CCC	30000	101	XYZ	Pune
4	DDD	40000	102	PQR	Mumbai

Redundancy!!!

1) **Redundant storage :** Note that the information about DeptID, DeptName and DeptLoc is repeated.

2) **Update anomalies :** In above table if we change **DeptLoc** of Pune to Chennai, then it will result **inconsistency** as for DeptID 101 the DeptLoc is Pune. Or otherwise, we need to **update multiple copies of DeptLoc** from Pune to Chennai. Hence this is an update anomaly.

3) **Insertion anomalies :** For above table if we want to add new tuple say (5, EEE,50000) for **DeptID 101** then it will cause repeated information of (101, XYZ,Pune) will occur.

4) **Deletion anomalies :** For above table, if we delete a record for **EmpID 4,** then automatically information about the **DeptID 102, DeptName PQR** and **DeptLoc Mumbai** will get deleted and one may not be aware about **DeptID 102.** This causes deletion anomaly.

4.3 : Functional Dependencies

Q.3 Explain the concept of functional dependencies.

Ans. : Definition : A functional dependency A->B in a relation holds if two tuples having same value of attribute A also have the same value for attribute

B. It is denoted by A->B where A is called determinant and B is called dependent.

For example - Consider Student table as follows -

Roll	Name	City
1	AAA	Mumbai
2	BBB	Pune
3	CCC	Gandhinagar

Here

Roll ->Name **hold**

But

Name->City **does not hold**

- In above table, student roll number is unique hence each student's name and city can be uniquely identified using his roll number.
- But using name we cannot uniquely identify his/her city because there can be same names of the students. Similarly using city name we can not identify the student uniquely. As in the same city may belong to multiple students.

Q.4 For the given below relation R(A,B,C,D,E) and its instance, check whether FDs given hold or not. Give reasons.
i) A->B ii) B->C iii) D->E iv) CD->E

A	B	C	D	E
a1	b1	c1	d1	e1
a1	b2	c1	d1	e1
a2	b2	c1	d2	e3
a2	b3	c3	d2	e2

Ans. : Association among attributes is known as Functional Dependencies (FD). AFD X->Y require that the value of X uniquely determines the value of Y where X and Y are set of attributes.

For example,

Roll_No -> Name : the value of Roll_No uniquely determines the Name.

Now from, the given relation and its instance -

i) The FD A->B does not hold because – a1 has two different values b1 and b2.

Similarly a2 has two different values and those are b2 and b3.

ii) The FD B->C holds true.

iii) D->E does not hold true because d2 gives two different values e3 and e2.

iv) CD->E hold true as (c1,d1) gives e1 , (c1,d2) gives e3 and (c3,d2) gives e2. All are uniquely identified.

Q.5 Explain the Armstrong's axioms with suitable example.

☞ **[SPPU : Dec.-22, Marks 6]**

Ans. : The closure set is a set of all functional dependencies implied by a given set F. It is denoted by F^+

The closure set of functional dependency can be computed using basic three rules which are also called as **Armstrong's Axioms.**

These are as follows -

i) **Reflexivity :** If $X \supseteq Y$, then $X \rightarrow Y$

ii) **Augmentation :** If $X \rightarrow Y$, then $XZ \rightarrow YZ$ for any Z

iii) **Transitivity :** If $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$

In addition to above axioms some additional rules for computing closure set of functional dependency are as follows -

- **Union :** If $X \rightarrow Y$ and $X \rightarrow Z$ then $X \rightarrow YZ$
- **Decomposition :** If $X \rightarrow YZ$, then $X \rightarrow Y$ and $X \rightarrow Z$

Q.6 Given FD's for relation R{A,B,C,D,E,F}, Find closure of FD set by applying Armstrong's Axioms.
A->B, A->C, CD->E, CD->F, B->E

Ans. :

Step 1 : A -> gives A attribute itself by reflexivity. It is called trivial production.

A -> B and A -> C, Hence by union rule A -> BC

A -> B and B -> E, Hence by transitivity rule A -> E

$\therefore (A)^+ = \{A,B,C,E\}$

Step 2 : B -> gives B itself

And B -> E

$\therefore (B)^+ = \{B,E\}$

Step 3 : CD -> gives CD itself. It is trivial.

CD -> E, CD -> F, Hence by union rule CD -> EF

$\therefore (CD)^+ = \{C,D,E,F\}$

So by omitting trivial productions, we get

$\therefore F^+ = \{A \rightarrow BC, A \rightarrow E, B \rightarrow E, CD \rightarrow EF\}$

Q.7 Compute the closure of the following set F of functional dependencies for relational schema R = (A,B,C,D,E) A -> BC, CD -> E, B -> D, E -> A

Ans. : The closure of F is denoted by F^+ and it can be computed in following steps

Step 1 : As A -> BC is given we get

A -> B and A -> C — By decomposition rule

Step 2 : As

A -> B — (Refer step 1)

B -> D — (given)

A -> D — (transitivity rule)

Step 3 :

A -> CD because A -> C and A -> D — (From step 1 and Step 2 applying union rule)

Step 4 :

CD -> E — (given)

A -> E — (transitive rule as A->CD, CD->E hence A->E)

Step 5 :

Since $A \rightarrow A$, — we have (reflexive)

$\mathbf{A \rightarrow ABCDE}$ — from the above steps (union)

Step 6 :

Since $E \rightarrow A$, $\mathbf{E \rightarrow ABCDE}$ — (transitive)

Step 7 :

Since $CD \rightarrow E$, $\mathbf{CD \rightarrow ABCDE}$ — (transitive)

Step 8 :

Since $B \rightarrow D$ and $BC \rightarrow CD$, $\mathbf{BC \rightarrow ABCDE}$ — (augmentative, transitive)

Step 9 :

Also, $C \rightarrow C$, $D \rightarrow D$, $BD \rightarrow D$

Thus any functional dependency with A, E, BC, or CD on the left hand side of the arrow is in F^+

Q.8 Give R = {A,B,C,G,H,I}. The following set F of functional dependencies holds
A -> B, A -> C, CG -> H, CG -> I , B -> H
Computer AG^+ . Is AG candidate key ?
Ans. :

Step 1 : A -> B, A -> C, Hence add A, B, C to the set of AG^+.

Step 2 : A -> B B -> H, hence add H to the set of AG^+

Step 3 : CG -> I, hence add I to the set of AG^+. Also add G to the set.

Thus $(AG)^+ = \{A,B,C,G,H,I\}$ = Relation R

Hence $(AG)^+$ is a candidate key.

Q.9 Consider schema R = (A,B,C,G,H,I) and the set F of functional dependencies
{A -> B, A -> C, CG -> H, CG -> I, B -> H}. Use $(F)^+$ Prove $(AG)^+$ -> I

Ans. :

Step 1 : As A -> B
B -> H
∴ A -> H (transitivity rule)

Step 2 : CG -> H
CG -> I
∴ **CG -> HI** (Union rule)

Step 3 : A -> C
CG -> I
AG -> I (Pseudo transitive rule)

Thus AG -> I is proved

$(F)^+$ = {A -> B, A -> C, CG -> H, CG -> I, B -> H,
A -> H, CG -> HI, AG -> I}

Q.10 Here are two sets of FDs for R(A,B,C,D,E). Are they equivalent ?

1) A -> B
AB -> C
D -> AC
D -> E

2) A -> BC
D -> AE

Ans. : Two sets of FDs are said to be equivalent if

Rule 1 : If FD2 ⊃ FD1. That means all FDs of FD1 can be derived from all the FDs of FD2.

Rule 2 : If FD1 ⊃ FD2. That means all FDs of FD2 can be derived from all the FDs of FD1.

Rule 3 : If both rule 1 and rule 2 are true then FD1 = FD2.

For given two sets

Step 1 : We will first check if all the FDs of FD1 are present in FD2

A -> B is present in FD1.	A -> BC is in FD2, that also means A -> B and A -> C by decomposition rule
Similarly AB -> C is in FD1 ∴ $(A)^+$ = {A,B,C}	Hence $(A)^+$ = {A,B,C}
∴ $(AB)^+$ = {A,B,C}	As D -> AE then by decomposition rule,
D -> AC	D -> A, D -> E
i.e D -> A, D -> C by decomposition rule.	As A -> B, then by transitivity rule D -> A, A -> B, D -> B
The D -> A, A -> B and C	∴ $(D)^+$ = {A,B,C,D,E}
D -> A, D -> B, D -> C by transitivity rule D -> E is given ∴ $(D)^+$ = {A,B,C,D,E}	

Step 2 : We will first check if all the FDs of FD2 are present in FD1

∴ $(A)^+$ = {A,B,C}	$(A)^+$ = {A,B,C}
∴ $(AC)^+$ = {A,B,C}	∴ $(D)^+$ = {A,B,C,D,E}
∴ $(D)^+$ = {A,B,C,D,E}	

Thus from Step 1 and Step 2, FD2 ⊃ FD1 and FD1 ⊃ FD2. Hence both the sets are equivalent.

Q.11 Consider two sets of functional dependency.
F = {A -> C, AC -> D, E -> AD, E -> H} and G = {A -> CD, E -> AH}.
Are they equivalent ?

Ans. : We will find the FDs of both F and G and then check for F and G ⊃ FD2 and G ⊃ F.

Step 1 : For F

Using G functional dependencies -

$(A)^+$ = {A,C,D} ← As A -> CD is in G

$(AC)^+$ = {A,C,D} ← As A -> CD is in G

$(E)^+$ = {A,C,D,E,H} ← As E -> AH, A -> CD is in G

Using F functional dependencies -

$(A)^+$ = {A,C,D} ← As A -> C and AC -> D is in F

$(AC)^+$ = {A,C,D} ← AC -> D is in F

$(E)^+$ = {A,C,D,E,H} ← As E -> AD, E -> H and A -> C is in F

Step 2 : For G

Using F functional dependencies -

$(A)^+$ = {A,C,D} ← As A -> C and AC -> D is in F

$(E)^+$ = {A,C,D,E,H} ← As E -> AD, E -> H is in F

Using G functional dependencies -

$(A)^+$ = {A,C,D} ← As A -> CD is in G

$(E)^+$ = {A,C,D,E,H} ← As E -> AH and A -> CD is in G

Step 3 : From both these steps

G⊇F and F⊇G

Hence F and G are equivalent.

Q.12 Consider the following functional dependencies over the attribute set R(ABCDE) for finding minimal cover
FD = {A -> C, AC -> D, B -> ADE}.
Ans. :

Step 1 : Split the FD such that R.H.S contain single attribute. Hence we get

A -> C

AC -> D

B -> A

B -> D

B -> E

Step 2 : Find the **redundant entries** and **delete** them. This can be done as follows -

- **For A -> C :** We find $(A)^+$ by assuming that we delete A -> C temporarily. We get $(A)^+$={A}. Thus from A it is not possible to obtain C by deleting A -> C. This means we can not delete A -> C.
- **For AC -> D :** We find $(AC)^+$ by assuming that we delete AC -> D temporarily. We get $(AC)^+$= {AC}. Thus by such deletion it is not possible to obtain D. This means we can not delete AC -> D.
- **For B -> A :** We find $(B)^+$ by assuming that we delete B -> A temporarily. We get $(B)^+$= {BDE}. Thus by such deletion it is not possible to obtain A. This means we can not delete B -> A.
- **For B -> D :** We find $(B)^+$ by assuming that we delete B -> D temporarily. We get $(B)^+$= {BEACD}. This shows clearly that even if we delete B -> D we can obtain D. This means we can delete B -> A. Thus it is redundant.
- **For B -> E :** We find $(B)^+$ by assuming that we delete B -> E temporarily. We get $(B)^+$= {BDAC}. Thus by such deletion it is not possible to obtain E. This means we can not delete B->E.

To summarize we get now

A -> C

AC -> D

B -> A

B -> E

Thus R.H.S gets simplified.

Step 3 : Now we will simplify L.H.S.

Consider AC -> D. Here we can split A and C. For that we find closure set of A and C.

$(A)^+ = (AC)$

$(C)^+ = (C)$

Thus C can be obtained from both A as well as C. That also means we need not have to have AC on L.H.S. Instead, only A can be allowed and C can be eliminated. Thus after simplification we get

A -> D

To summarize we get now

A -> C

A -> D

B -> A

B -> E

Thus L.H.S gets simplified.

Step 3 : The simplified L.H.S. and R.H.S can be combined together to form

A -> CD

B -> AE

This is a **minimal cover** or **canonical cover** of functional dependencies.

4.4 : Decomposition

Q.13 List the properties of decomposition. Explain lossless join with example. ☞ **[SPPU : May-18 (End Sem), June 22, Marks 6]**

Ans. : There are two properties associated with decomposition and those are -

1) **Loss-less join or non loss decomposition :** When all information found in the original database is preserved after decomposition, we call it as loss less or non loss decomposition.

2) **Dependency preservation :** This is a property in which the constraints on the original table can be maintained by simply enforcing some constraints on each of the smaller relations.

The lossless join can be defined using following three conditions :

i) **Union** of attributes of R1 and R2 must be equal to attribute of R. Each attribute of R must be either in R1 or in R2.

$Att(R1) \cup Att(R2) = Att(R)$

ii) **Intersection** of attributes of R1 and R2 must not be NULL.

$Att(R1) \cap Att(R2) \neq \Phi$

iii) **Common attribute** must be a key for at least one relation (R1 or R2)

$Att(R1) \cap Att(R2) \rightarrow Att(R1)$

or $Att(R1) \cap Att(R2) \rightarrow Att(R2)$

Q.14 Consider the following relation R(A, B, C, D)and FDs A->BC, is the decomposition of R into R1(A, B, C), R2(A, D). Check if the decomposition is lossless join or not.

Ans. :

Step 1 : Here $Att(R1) \cup Att(R2) = Att(R)$

i.e $R1(A,B,C) \cup R2(A,D) = (A,B,C,D)$ i.e R.

Thus first condition gets satisfied.

Step 2 : Here $R1 \cap R2 = \{A\}$. Thus $Att(R1) \cap Att(R2) \neq \phi$. Here the second condition gets satisfied.

Step 3 : $Att(R1) \cap Att(R2) \rightarrow \{A\}$. Now $(A)+ = \{A,B,C\}$ ϕ attributes of R1. Thus the third condition gets satisfied.

This shows that the given decomposition is a **lossless join.**

Q.15 Consider the following relation R(A, B, C, D, E, F) and FDs A->BC, C->A, D->E, F->A, E->D is the decomposition of R into R1(A, C, D), R2(B, C, D) and R3 (E, F, D). Check for lossless.

Ans. :

Step 1 : $R1 \cup R2 \cup R3 = R$. Here the first condition for checking lossless join is satisfied as $(A,C,D) \cup (B,C,D) \cup (E,F,D) = \{A,B,C,D,E,F\}$ which is nothing but R.

Step 2 : Consider R1∩ R2 = {CD} and R2∩R3 = {D}. Hence second condition of intersection not being Φ gets satisfied.

Step 3 : Now, consider R1(A, C, D) and R2(B, C, D).

We find R1∩R2 = {CD}

$(CD)^+$ = {ABCDE} ∈ attributes of R1 i.e.{A, C, D}. Hence condition 3 for checking lossless join for R1 and R2 gets satisfied.

Step 4 : Now, consider R2(B, C, D) and R3(E, F, D) . We find R2∩R3={D}.

$(D)^+$ = {D, E} which is neither complete set of attributes of R2 or R3.

[Note that F is missing for being attribute of R3].

Hence it is not **lossless join decomposition.** Or in other words we can say it is a **lossy decomposition.**

Q.16 Consider following relational table. Find nontrivial and trivial functional dependency.

A	B	C
a_1	b_1	c_1
a_1	b_1	c_2
a_2	b_1	c_1
a_2	b_1	c_3

☞ **[SPPU : June-22, Marks 5]**

Ans. : The non-trivial functional dependencies are :

1) A → B

2) C → B

3) AC → B

The trivial functional dependency is of the form –

α → β

where β ⊆ α

C does not functionally determine A

because

A	C
a_1	c_1
a_2	c_1

Here c_1 has different values of A i.e. a_1 and a_2.

Similarly B does not functionally determine C.

4.5 : Dependency Preservation

Q.17 What is dependency preservation ?

Ans. : • **Definition** : A decomposition D = {R1, R2, R3....Rn} of R is dependency preserving for a set F of functional dependency if - (F1 ∪ F2 ∪ ... ∪ Fm) = F.

- If decomposition is not dependency-preserving, some dependency is lost in the decomposition.

Q.18 Consider the relation R (A, B, C) for functional dependency set {A -> B and B → C} which is decomposed into two relations R1 = (A, C) and R2 = (B, C). Then check if this decomposition dependency preserving or not.

Ans. : This can be solved in following steps :

Step 1 : For checking whether the decomposition is dependency preserving or not we need to check following condition

$$F^+ = (F1 \cup F2)^+$$

Step 2 : We have with us the F^+ ={ A->B and B->C }

Step 3 : Let us find $(F1)^+$ for relation R1 and $(F2)^+$ for relation R2

R1(A,C)	R2(B,C)
A->A Trivial	B->B Trivial
C->C Trivial	C->C Trivial
A->C ∵ In (F)+A->B->C and it is Nontrivial	B->C ∵ In $(F)^+$ B->C and it is Non-Trivial
AC->AC Trivial	BC->BC Trivial
A->B but is not useful as B is not part of R1 set	We can not obtain C->B
We can not obtain C->A	

Step 4 : We will eliminate all the trivial relations and useless relations. Hence we can obtain R1 and R2 as

R1(A,C)	R2(B,C)
A->C Nontrivial	B->C Non-Trivial

$(F1 \cup F2)^+ = \{A->C, B->C\} \neq \{A->B, B->C\}$ i.e. $(F)^+$

Thus the condition specified in step 1 i.e. F+ = (F1 ∪ F2)+ **is not true.**

Hence it is **not dependency preserving decomposition.**

Q.19 Given relation r(X,Y,W,Z,P,Q) and the set F = {XY->W, XW->P, PQ->Z, XY->Q}. Consider the decomposition R1(Z,P,Q), 2(X,Y,Z,P,Q). Is this decomposition lossless or lossy ? Use lossless join algorithm.

Ans. :

Step 1 : Construct a table with six columns for given six attributes i.e. X, Y, W, Z, P, Q and two rows for given two relation i.e. R1 and R2.

	X	Y	W	Z	P	Q
R1						
R2						

Step 2 : Fill in the entries as follows -

R1 (Z, P, Q) and R2 (X, Y, Z, P, Q)

Consider R1 having attributes Z, P, Q so put 'α' in those row and put 'β' other remaining rows same as R2 having attributes X, Y, Z, P, Q so put 'α' in those row and put 'β' other remaining rows.

	X	Y	W	Z	P	Q
R1	β_{1X}	β_{1Y}	β_{1W}	α_Z	α_P	α_Q
R2	α_X	α_Y	β_{2W}	α_Z	α_P	α_Q

Step 3 :

i) Considering XY → W, we check if the two rows of the table have the same value under the columns XY that make up the determinant of the FD. Since the rows do not have identical values, the table will remain unchanged and we repeat step 3 by considering another FD.

ii) Considering XW → P, we check if the two rows of the table have the same value under the columns XW that make up the determinant of the FD. Since the rows do not have identical of values, the table will remain unchanged and we repeat step 3 by considering another FD.

iii) Considering PQ → Z, we check if the two rows of the table have the same value under the columns PQ that make up the determinant of the FD. Since the rows R1 and R2 have values of α_P and α_Q under the columns PQ. Therefore, we need to equate all the corresponding entries for the rows under column Z. The values in column Z are already equal no changes are necessary Repeat step 3 again.

iv) Considering XY → Q, we check if the two rows of the table have the value under the columns XY that make up determinant of the FD. Since rows do not have identical values, the table will remain unchanged. Now, there are no more FDs to consider.

Step 4 : Since there is no row in the table that has all ϕ in it's entries the decomposition is lossy. That is original table cannot be recovered from the join of relations R1 and R2.

4.6 : The process of Normalization : 1NF, 2NF, 3NF, BCNF.

Q.20 State the need of normalization? Explain 1NF, 2NF, and 3NF with example. **[SPPU : Aug.-17 (In Sem), Marks 5, Dec.-22, Marks 8]**

Ans. : Need of Normalization :

- Normalization is the process of reorganizing data in a database so that it meets **two basic requirements :**

1) There is **no redundancy** of data (all data is stored in only one place) and
2) **Data dependencies** are logical (all related data items are stored together)
   1) **It eliminates redundant data.**
   2) **It reduces chances of data error.**
   3) The normalization is important because it allows database to take up **less disk space.**
   4) It also help in increasing the **performance.**
   5) It improves the data integrity and consistency.

First Normal Form : The table is said to be in 1NF if it follows following rules -

i) It should only have single (atomic) valued attributes/columns.

ii) Values stored in a column should be of the same domain.

iii) All the columns in a table should have unique names.

iv) And the order in which data is stored, does not matter.

Consider following student table

Student

sid	sname	Phone
1	AAA	11111 22222
2	BBB	33333
3	CCC	44444 55555

As there are multiple values of phone number for sid 1 and 3, the above table is not in 1NF. We can make it in 1NF. The conversion is as follows -

sid	sname	Phone
1	AAA	11111
1	AAA	22222
2	BBB	33333
3	CCC	44444
3	CCC	55555

Second Normal Form :

For a table to be in the Second Normal Form, following conditions must be followed

i) It should be in the First Normal form.

ii) It should not have partial functional dependency.

For example : Consider following table in which every information about a the Student is maintained in a table such as student id(sid), student name(sname), course id(cid) and course name(cname).

Student_Course

sid	sname	cid	cname
1	AAA	101	C
2	BBB	102	C++
3	CCC	101	C
4	DDD	103	Java

This table is not in 2NF. For converting above table to 2NF we must follow the following steps -

Step 1 : The above table is in 1NF.

Step 2 : Here **sname** and **sid** are associated similarly **cid** and **cname** are associated with each other. Now if we delete a record with **sid = 2,** then automatically the course C++ will also get deleted. Thus.

sid->sname or **cid->cname** is a partial functional dependency, because **{sid,cid}** should be essentially a candidate key for above table. Hence to bring the above table to 2NF we must decompose it as follows :

Student

sid	sname	cid
1	AAA	101
2	BBB	102
3	CCC	101
4	DDD	103

Here candidate key is (sid, cid) and (sid, cid)->sname

Course

cid	cname
101	C
102	C++
101	C
103	Java

Here candidate key is cid here cid- > cname

Thus now table is in 2NF as there is no partial functional dependency.

Third Normal Form :

A table is said to be in the third normal form when,

i) It is in the second normal form.(i.e. it does not have partial functional dependency).

ii) It doesn't have transitive dependency.

Or in other words

In other words 3NF can be defined as : A table is in 3NF if it is in 2NF and for each functional dependency

X-> Y

at least one of the following conditions hold :

i) X is a super key of table.

ii) Y is a prime attribute of table.

For example : Consider following table **Student_details** as follows -

sid	sname	zipcode	cityname	state
1	AAA	11111	Pune	Maharashtra
2	BBB	22222	Surat	Gujarat
3	CCC	33333	Chennai	Tamilnadu
4	DDD	44444	Jaipur	Rajastan
5	EEE	55555	Mumbai	Maharashtra

Here

Super keys :

{sid},{sid,sname},{sid,sname,zipcode}, {sid,zipcode,cityname}... and so on.

Candidate keys : {sid}

Non-prime attributes : {sname,zipcode,cityname,state}

The dependencies can be denoted as

sid->sname
sid->zipcode
zipcode->cityname
cityname->state

The above denotes the transitive dependency. Hence above table is not in 3NF. We can convert it into 3NF as follows :

Student

sid	sname	zipcode
1	AAA	11111
2	BBB	22222
3	CCC	33333
4	DDD	44444
5	EEE	55555

Zip

zipcode	cityname	state
11111	Pune	Maharashtra
22222	Surat	Gujarat
33333	Chennai	Tamilnadu
44444	Jaipur	Rajasthan
55555	Mumbai	Maharashtra

Q.21 What is Normalization ? State and explain 2NF and 3NF.

[SPPU : May-19 (End Sem), Marks 4]

Ans. : Refer Q.20

Q.22 Consider the relation R = {A, B, C, D, E, F, G, H, I, J} and the set of functional dependencies F = {{A, B} → C, A→ {D, E}, B→ F, F→{G, H}, D→{I, J} }

1. What is the key for R ? Demonstrate it using the inference rules.

2. Decompose R into 2NF, then 3NF relations.

Ans. : Let,

$A \rightarrow DE$ (given)

$\therefore$ $A \rightarrow D$, $A \rightarrow E$ (decomposition rule)

As $D \rightarrow IJ$, $A \rightarrow IJ$

Using union rule we get

$A \rightarrow DEIJ$

As $A \rightarrow A$

we get $A \rightarrow ADEIJ$

Using augmentation rule we compute AB

$AB \rightarrow ABDEIJ$

But $AB \rightarrow C$ (given)

$\therefore$ $AB \rightarrow ABCDEIJ$

$B \rightarrow F$ (given) $F \rightarrow GH$ $\therefore$ $B \rightarrow GH$ (transitivity)

$\therefore$ $AB \rightarrow AGH$ is also true

Similarly $AB \rightarrow AF$ $\because$ $B \rightarrow F$ (given)

Thus now using union rule

$AB \rightarrow ABCDEFGHIJ$

$\therefore$ AB is a key

The table can be converted to 2NF as

$R_1 = (\underline{A}, \underline{B}, C)$

$R_2 = (\underline{A}, D, E, I, J)$

$R_3 = (\underline{B}, F, G, H)$

The above 2NF relations can be converted to 3NF as follows

$R_1 = (\underline{A}, \underline{B}, C)$

$R_2 = (\underline{A}, D, E)$

$R_3 = (\underline{D}, I, J)$

$R_4 = (\underline{B}, E)$

$R_5 = (E, G, H)$.

Q.23 Normalize the below relation upto 3NF.

Module	Dept	Lecturer	Text
M1	D1	L1	T1
M1	D1	L1	T2
M2	D1	L1	T1
M2	D1	L1	T3
M3	D1	L2	T4
M4	D2	L3	T1
M4	D2	3	T5
M5	D2	L4	T6

Ans. : The given relation is already in 1st normal form. But it has Insert, delete and update anomalies. Because -

1) Insert anomalies : We can not add a module(M) with no texts(T).

2) Delete anomalies : If we remove M3, we remove L2 as well.

3) Update anomalies : To change lecturer for M1, we have to change two rows.

Hence we will convert it to second normal form.

Step 1 : We can define the functional dependency FD as

{Module, Text}->{Lecturer, Dept}

But

{Module}->{Lecturer, Dept}

That means Lecturer and Dept are partially dependent on the primary key. Hence for conversion of first normal form to second normal form we will decompose the give table into two tables as

Table 2a

Module	Dept	Lecturer
M1	D1	L1

M2	D1	L1
M3	D1	L2
M4	D2	L3
M5	D2	L4

Table 2b

Module	Text
M1	T1
M1	T2
M2	T1
M2	T3
M3	T4
M4	T1
M4	T5
M5	T6

The relation is now in second normal form.

Step 2 : The table 2a has Insert, Delete and Update anomalies. Because -

1) **INSERT anomalies :** We can't add lecturers who teach no modules.

2) **UPDATE anomalies :** To change the department for L1 we must alter two rows.

3) **DELETE anomalies :** If we delete M3 we delete L2 as well.

Hence, to eliminate these anomalies, we decompose table 2a into two tables and convert it to third normal form.

Step 3 : Hence we get

Table 3a

Lecturer	Dept.
L1	D1
L2	D1
L3	D2
L4	D2

Table 3b

Module	Lecturer
M1	L1
M2	L1
M3	L2
M4	L3
M5	L4

Step 4 : Thus now the complete relation is decomposed into three tables and it is in third normal form. It is summarized as below

Table 3a

Lecturer	Dept.
L1	D1
L2	D1
L3	D2
L4	D2

Table 3b

Module	Lecturer
M1	L1
M2	L1
M3	L2
M4	L3
M5	L4

Table 2b

Module	Text
M1	T1
M1	T2
M2	T1
M2	T3
M3	T4
M4	T1
M4	T5
M5	T6

Q.24 Explain BCNF with suitable example.

Ans. : Boyce and Codd Normal Form is a **higher version** of the Third Normal form. This form deals with certain type of anomaly that is not handled by 3NF.

A **3NF** table which **does not have multiple overlapping** candidate keys is said to be in BCNF.

Or in other words,

For a table to be in BCNF, following conditions must be satisfied :

i) R must be in 3rd Normal Form

ii) For each functional dependency ($X \rightarrow Y$), X should be a super key. In simple words if Y is a prime attribute then X can not be non prime attribute.

For example - Consider following table that represents that a student enrollment for the course -

Enrollment table

sid	Course	Teacher
1	C	Ankita
1	Java	Poonam
2	C	Ankita
3	C++	Supriya
4	C	Archana

From above table following observations can be made :

- One student can enroll for multiple courses. For example student with sid = 1 can enroll for C as well as Java.

- For each course, a teacher is assigned to the student.
- There can be multiple teachers teaching one course for example course C can be taught by both the teachers namely - Ankita and Archana.
- The candidate key for above table can be (sid, course), because using these two columns we can find,
- The above table holds following dependencies
  - (sid, course)->Teacher
  - Teacher->course
- The above table is not in BCNF because of the dependency teacher ->course. Note that the teacher is not a superkey or in other words, teacher is a non prime attribute and course is a prime attribute and non-prime attribute derives the prime attribute.
- To convert the above table to BCNF we must decompose above table into student and course tables

Student

sid	Teacher
1	Ankita
1	Poonam
2	Ankita
3	Supriya
4	Archana

Course

Teacher	Course
Ankita	C
Poonam	Java
Ankita	C

Supriya	C++
Archana	C

Now the table is in BCNF

Q.25 Prove the statement "Every relation which is in BCNF is in 3NF but the converse is not true." [SPPU : Dec.-22, Marks 6]

Ans. : For a relations to be in 3NF

A table is said to be in the Third Normal Form when,

i) It is in the Second Normal form. (i.e. it does not have partial functional dependency)

ii) It doesn't have transitive dependency.

Or in other words

In other words 3NF can be defined as : A table is in 3NF if it is in 2NF and for each functional dependency X-> Y

at least one of the following conditions hold :

iii) X is a super key of table

iv) Y is a prime attribute of table

For a relation to be in BCNF

1) It should be in 3NF
2) **A 3NF** table which **does not have multiple overlapping** candidate keys is said to be in BCNF.

For proving that the table can be in 3NF but not in BCNF consider Following relation

R(Student, Subject, Teacher) . Consider following are FDs

(Subject, Student)-> Teacher

Because subject and student combination gives unique teacher.

Teacher -> Subject

Because each teacher teaches only Subject.

(Teacher, Student)->Subject

- So, this relation is in 3NF as every non-key attribute is non-transitively fully functional dependent on the primary key.
- But it is not in BCNF. Because this is a case of **overlapping of candidate keys** because there are two composite candidate keys :
  - (Subject, Student)
  - (Teacher, Student)

And student is a common attribute in both the candidate keys.

So we need to normalize the above table to BCNF. For that purpose we must set Teacher to be a candidate key

The decomposition of above takes place as follows

R1(Student, Teacher)

R2(Teacher, Subject)

Now table is in 3NF, as well as in BCNF.

This show that the relation Every relation which is in BCNF is in 3NF but the converse is not true.

Q.26 What is the difference between 3NF and BCNF ?

☞ **[SPPU : Dec.-22, Marks 6]**

Ans. :

Sr. No.	3NF	BCNF
1.	3NF stands for Third Normal Form.	BCNF stands for Boyce Codd Normal Form.
2.	The table is in 3NF if it is in 2NF and for each functional dependency X->Y at least following condition hold: (i) X is a superkey, (ii) Y is prime attribute of table.	The table is in BCNF if it is in 3rd normal form and for each relation X->Y X should be super key.
3.	3NF can be obtained without sacrificing all dependencies.	Dependencies may not be preserved in BCNF.
4.	Lossless decomposition can be achieved in 3NF.	Lossless decomposition is hard to obtain in BCNF.

5.	3NF can be achieved without loosing any information from the old table	For obtaining BCNF we may loose some information from old table.

Q.27 Consider following relation - Book(book_title, authorname, book_type, listprice, author_affiliation, publisher)
Suppose the following functional dependencies exist –
book_title->publisher,book_type
book_type->listprice
authorname->author_affiliation
i) What normal form is the relation in ? Explain your answer.
ii) Apply normalization until you can not decompose the relations further. State the reasons behind each decomposition.
Ans. :

(i) The key for this relation is (Book_title, Authorname). This relation is in 1NF and not in 2NF as no attributes are Fully Functionally Dependent on the key.

Step 1 : We will first decompose it in **2nd Normal Form** by following decomposition.

R1(book_title, authorname)
R2(book_title, publisher,book_type,listprice)
R3(authorname,authoraffiliation)

Reason : This decomposition eliminates partial functional dependencies.

Step 2 : Now we will decompose it further to bring the table in **3rd Normal form**

R1(book_title, authorname)
R2(book_title, publisher,book_type,listprice)
R21(book_title, publisher,book_type)
R22(book_type, listprice)
R3(authorname,authoraffiliation)

Reason : This decomposition eliminates transitive functional dependency of listprice.

Part II : Introduction to Query Processing

4.7 : Overview

Q.28 Define query processing . What are the steps involved in query processing ? [SPPU : May-18 (End Sem), June-22, Dec.-22, Marks 5]

Ans. : • Query processing is a collection of activities that are involved in extracting data from database.

- During query processing there is translation high level database language queries into the expressions that can be used at the physical level of filesystem.
- There are three basic steps involved in query processing and those are –

1. Parsing and Translation

- In this step the query is translated into its internal form and then into relational algebra.
- Parser checks syntax and verifies relations.
- For instance - If we submit the query as,

```
SELECT RollNo, name
FROM Student
HAVING RollNo=10
```

Then it will issue a syntactical error message as the correct query should be

```
SELECT RollNo, name
FROM Student
HAVING RollNo=10
```

Thus during this step the syntax of the query is checked so that only correct and verified query can be submitted for further processing.

2. Optimization :

- During this process the query evaluation plan is prepared from all the relational algebraic expressions.
- The query cost for all the evaluation plans is calculated.

- Amongst all equivalent evaluation plans the one with lowest cost is chosen.
- Cost is estimated using statistical information from the database catalog, such as the number of tuples in each relation, size of tuples, etc.

3. Evaluation

- The query-execution engine takes a query-evaluation plan,executes that plan, and returns the answers to the query.

The above describe steps are represented by following Fig. Q.28.1.

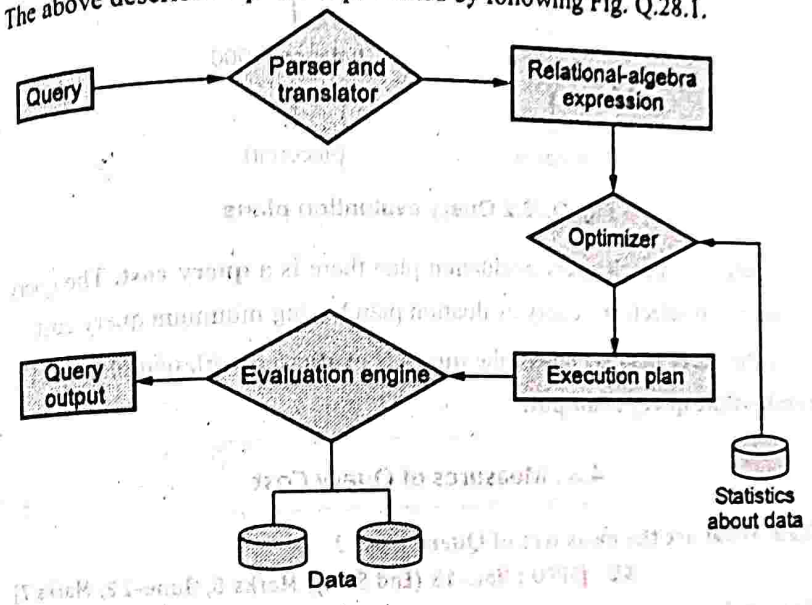


Fig. Q.28.1 Query processing

For example - If the SQL query is,

```
SELECT balance
FROM account
WHERE balance<1000
```

Step 1 : This query is first verified by the **parser and translator** unit for correct syntax. If so then the relational algebra expressions can be obtained. For the above given queries there are two possible relational algebra

(1) $\sigma_{balance<1000}(\pi_{balance}(account))$

(2) $\pi_{balance}$ ($\sigma_{balance<1000}$ (account))

Step 2 :

Query Evaluation Plan : To specify fully how to evaluate a query, we need not only to provide the relational-algebra expression, but also to annotate it with instructions specifying how to evaluate each operation. For that purpose, using the order of evaluation of queries, two query evaluation plans are prepared. These are as follows

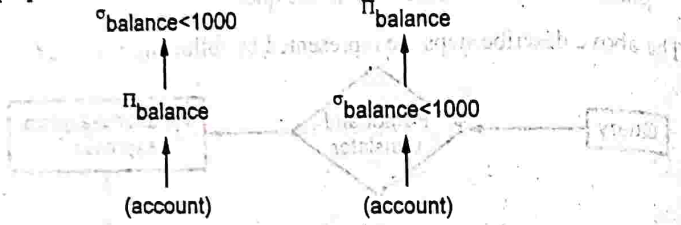


Fig. Q.28.2 Query evaluation plans

Associated with each query evaluation plan there is a **query cost.** The query optimization selects the query evaluation plan having minimum query cost.

Once the query plan is chosen, the **query is evaluated** with that plan and the **result of the query is output.**

4.8 : Measures of Query Cost

Q.29 What are the measures of Query Cost ?

☞ **[SPPU : Nov.-18 (End Sem), Marks 5, June-22, Marks 7]**

Ans. : • There are many factors that contribute to query cost are -

- Disk access
- CPU time to execute the query
- Cost of communication in distributed and parallel database system.

• The cost of access data from disk is an important cost. Normally disk access is relatively slow as compared to in-memory operations. Moreover, the CPU speed is much faster than the disk speed. Hence the time spent in disk is relatively dominant factor in query execution.

• Computing CPU access time is comparatively harder, hence we will not consider the CPU time to execute the query.

• Similarly the cost of communication does not matter for simple large databases present in the centralized database system.

• Typically disk access is the predominant cost and is also relatively easy to estimate, taking into account:

- Number of seeks × average-seek-cost
- Number of blocks read × average-block-read-cost
- Number of blocks written × average-block-write-cost

• Cost to write a block is greater than cost to read a block because data is read back after being written to ensure that the write was successful.

• We use **number of block transfers** from disk and **number of disk seeks** to estimate the Query cost.

• Let,

- b be the number to blocks
- S be the number of Seeks
- tT is average time required to transfer a block of data, in seconds
- tS is average block access time in seconds.

Then query cost can be computed using following formula $b*t_T+S*t_S$

4.9 : Selection and Join Operations

Q.30 Explain algorithm for nested loop join

Ans. : This algorithm is for computing r ⋈Θ s

Let, r is called the outer relation and s the inner relation of the join.

```
for each tuple t_r in r do begin
    for each tuple t_s in s do begin
    test pair (t_r,t_s) to see if they satisfy the join condition θ
    if θ is satisfied, then, add (t_r , t_s) to the result.
    end
end
```

- This algorithm requires no indices and can be used with any kind of join condition.
- It is expensive since it examines every pair of tuples in the two relations.
- In the **worst case**, if there is enough memory only to hold one block of each relation, the estimated cost is
- $n_r \times b_s + b_r$ block transfers, plus $n_r + b_r$ seeks
- If the smaller relation fits entirely in memory, use that as the inner relation
  - Reduces cost to $b_r + b_s$ block transfers and 2 seeks
- **For example** - Assume the query CUSTOMERS ⋈ ORDERS (with join attribute only being CName)

Number of records of customer : 10000 order : 5000

Number of blocks of customer : 400 order : 100

Formula Used :

(1) $n_r \times b_s + b_r$ block transfers,

(2) $n_r + b_r$ seeks

r is outer relation and s is inner relation.

With order as outer relation :

$n_r = 5000$ $b_s = 400$, $b_r = 100$

$5000 \times 400 + 100 = 2000100$ block transfers and

$5000 + 100 = 5100$ seeks

With customer as the outer relation :

$n_r = 10000$, $b_s = 100$, $b_r = 400$

$10000 \times 100 + 400 = 1000400$ block transfers and

$10000 + 400 = 10400$ seeks

If smaller relation (order) fits entirely in memory, the cost estimate will be :

$b_r + b_s = 500$ block transfers

Algorithm for Block Nested Loop Join

Variant of nested-loop join in which every block of inner relation is paired with every block of outer relation.

This algorithm is for computing $r \bowtie_\theta s$

Let, r is called the outer relation and s the inner relation of the join.

```
for each block B_r of r do
for each block B_s of s do
for each tuple t_r in B_r do begin
        for each tuple t_s in B_s do begin
          test pair (t_r, t_s) to see if they satisfy the join condition θ
          if θ is satisfied, then, add (t_r, t_s) to the result.
          end
      end
  end
end
```

4.10 : Evaluation of Expressions

Q.31 Explain with example materialized evaluation and pipelining.

☞ [SPPU : June-22, Marks 6]

Ans. : (1) Materialization :

- The result of relational algebra operator operation is saved in some temporary or intermediate files which can be used for processing by next subsequent operation. Such an approach is called materialization.
- This approach is called materialization approach because - When the results are temporarily stored in intermediate files then it is said that the **tuples are materialized.**
- During evaluation of query a query tree is built. The process of materialization starts at the **lowest level** operation of the query tree. That means the lowest level expression is evaluated, the result is stored in intermediate files, this result is then used by subsequent higher level of the

tree, this process is repeated and finally the operation at the root of the query tree is evaluated giving the final result of the expression.

- The cost of materialization approach includes the cost of all operations as well as cost of writing results of each intermediate operation to disk.

(2) Pipelining :

- The process of sending the result of one operation to another operation without storing it to intermediate files is called **pipelining.**
- This approach **improves the performance** of the query.
- The efficiency of query evaluation can be improved by reducing the number of temporary files that are produced for storing intermediate results. The pipelining approach helps to improve the query evaluation process because it does not make use of temporary files and results are directly send to the next subsequent operator.

4.11 : Introduction to Query Optimization

Q.32 Why is query optimization important for databases ?

☞ **[SPPU : May-19 (End Sem), Marks 5, Dec.-22, Marks 5]**

Ans. : Definition : Query optimization is a technique using which optimizer attempts to determine the most efficient way to execute a given query by considering the possible query plans.

Importance of Query Optimization : Following are some key reasons why query optimization is important in DBMS -

1) Query optimization helps in faster query processing.
2) It requires less cost per query.
3) It provides high performance of the system.
4) It help in better resource utilization by reducing the amount of system resources needed to execute them.
5) Query optimization is essential for scaling up a database system. Because without proper query optimization, the database may become overloaded and will not be able to handle the increased workload.

Q.33 Explain query optimization with respect to SQL databases.

☞ **[SPPU : Nov.-18 (End Sem), Marks 8]**

Ans. : Following are the two algorithms used for query optimization –

Heuristic Estimation

- Heuristic is a rule that leads to least cost in most of cases.
- Systems may use heuristics to reduce the number of choices that must be made in a cost-based fashion.
- Heuristic optimization transforms the query-tree by using a set of rules that typically t improve execution performance. These rules are
  1. Perform selection early (reduces the number of tuples).
  2. Perform projection early (reduces the number of attributes).
  3. Perform most restrictive selection and join operations before other similar operations (such as cartesian product).
- Some systems use only heuristics, others combine heuristics with partial cost-based optimization.

Steps in heuristic estimation

Step 1 : Scanner and parser generate initial query representation.

Step 2 : Representation is optimized according to heuristic rules.

Step 3 : Query execution plan is developed.

For example : Suppose there are two relational algebra -

(1) $\sigma_{city=\text{"Pune"}}$ (π_{cname} Branch)⋈ Account ⋈ Customer)

(2) π_{cname}($\sigma_{city=\text{"Pune"}}$(Branch⋈ Account ⋈ Customer))

The query evaluation plan can be drawn using the query trees as follows -

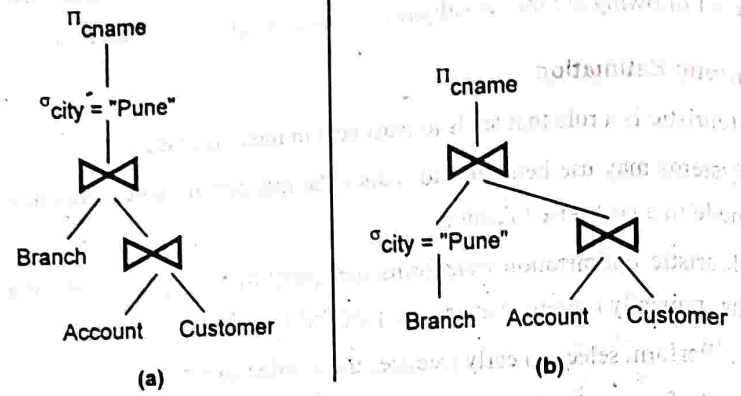


Fig. Q.33.1 Query evaluation plan

Out of the above given query evaluation plans, the Fig. Q.33.1 (b) is much faster than Fig. Q.33.1 (a) because - in Fig. Q.33.1 (a) the join operation is among branch, account and customer, whereas in Fig. Q.33.1 (b) the join of (account and customer) is made with the selected tuple for City ="Pune". Thus the output of entire table for join operation is much more than the join for some selected tuples. Thus we get choose the optimized query.

Cost Based Estimation

- A cost based optimizer will look at all of the possible ways or scenarios in which a query can be executed.
- Each scenario will be assigned a 'cost', which indicates how efficiently that query can be run.
- Then, the cost based optimizer will pick the scenario that has the least cost and execute the query using that scenario, because that is the most efficient way to run the query.
- Scope of query optimization is a query block. Global query optimization involves multiple query blocks.
- Cost components for query execution
  - Access cost to secondary storage

  - Disk storage cost
  - Computation cost
  - Memory usage cost
  - Communication cost
- Following information stored in DBMS catalog and used by optimizer
  - File size
  - Organization
  - Number of levels of each multilevel index
  - Number of distinct values of an attribute
  - Attribute selectivity
- RDBMS stores histograms for most important attributes.

4.12 : Transformation of Relational Expression

Q.34 State the rules for transformation of relational expressions

☞ [SPPU : Dec.-22, Marks 4]

Ans. :

(1) Conjunctive selection operations can be deconstructed into a sequence of individual selections.

$\sigma_{\theta 1 \wedge \theta 2}(E) = \sigma_{\theta 1}(\sigma_{\theta 2}(E))$

(2) Selection operations are commutative.

$\sigma_{\theta 1}(\sigma_{\theta 2}(E)) = \sigma_{\theta 2}(\sigma_{\theta 1}(E))$

(3) Only the last in a sequence of projection operations is needed, the others can be omitted.

$\Pi_{L1}(\Pi_{L2}(\ldots(\Pi_{Ln}(E))\ldots)) = \Pi_{L1}(E)$

(4) Selections can be combined with Cartesian products and theta joins.

(i) $\sigma_{\theta}(E1 \times E2) = E1 \bowtie_{\theta} E2$

(ii) $\sigma_{\theta 1}(E1 \bowtie_{\theta 2} E2) = E1 \bowtie_{\theta 1 \wedge \theta 2} E2$

(5) Theta-join operations (and natural joins) are commutative.

$E1 \bowtie_{\theta} E2 = E2 \bowtie_{\theta} E1$

(6)

(a) **Natural join operations are associative :**

$(E1 \bowtie E2) \bowtie E3 = E1 \bowtie (E2 \bowtie E3)$

(b) **Theta joins are associative :**

$(E1 \bowtie_{\theta 1} E2) \bowtie_{\theta 2 \wedge \theta 3} E3 = E1 \bowtie_{\theta 1 \wedge \theta 3} (E2 \bowtie_{\theta 2} E3)$

where $\theta 2$ involves attributes from only E2 and E3

(7) The selection operation distributes over the theta join operation under the following two conditions :

(a) When all the attributes in θ_0 involve only the attributes of one of the expressions (E1) being joined.

$\sigma_{\theta 0}(E1 \bowtie_{\theta} E2) = (\sigma_{\theta 0}(E1)) \bowtie_{\theta} E2$

(b) When θ_1 involves only the attributes of E1 and θ_2 involves only the attributes of E2

$\sigma_{\theta 1 \wedge \theta 2}(E1 \bowtie_{\theta} E2) = (\sigma_{\theta 1}(E1)) \bowtie_{\theta} (\sigma_{\theta 2}(E2))$

(8) The projection operation distributes over the theta join operation as follows :

(a) if θ involves only attributes from L1 ∪ L2 :

$\Pi_{L1 \cup L2}(E1 \bowtie_{\theta} E2) = (\Pi_{L1}(E1)) \bowtie_{\theta} (\Pi_{L2}(E2))$

(b) Consider a join E1 $\bowtie_{\theta}$ E2 :

- Let L1 and L2 be sets of attributes from E1 and E2 , respectively.
- Let L3 be attributes of E1 that are involved in join condition θ, but are not in L1 ∪ L2 and
- Let L4 be attributes of E2 that are involved in join condition θ, but are not in L1 ∪ L2 .

$\Pi_{L1 \cup L2}(E1 \bowtie_{\theta} E2) = (\Pi_{L1 \cup L2}((\Pi_{L1 \cup L3}(E1)) \bowtie_{\theta} (\Pi_{L2 \cup L4}(E2)))$

(9) The set operation union and intersection are commutative

$E1 \cup E2 = E2 \cup E1$

$E1 \cap E2 = E2 \cap E1$

(10) The set union and intersection are associative

$(E1 \cup E2) \cup E3 = E1 \cup (E2 \cup E3)$

$(E1 \cap E2) \cap E3 = E1 \cap (E2 \cap E3)$

(11) The selection operation distributes over union ∪,∩ and -

$\sigma_{\theta}(E1 - E2) = \sigma_{\theta}(E1) - \sigma_{\theta}(E2)$

and similarly for ∪ and ∩ in place of -

$\sigma_{\theta}(E1 - E2) = \sigma_{\theta}(E1) - E2$

and similarly for ∩ in place of –, but not for ∪

(12) The projection operation distributes over union

$\Pi_L(E1 \cup E2) = (\Pi_L(E1)) \cup (\Pi_L(E2))$

For example - Consider the following query

Find the names of all employees in accounts department, along with the department location in which the employee work for.

The schema is

Employee(empID, empname, deptname, salary)
Works_for(empID, deptID,working_hr)
Department(deptID, deptname,deptloc)

The relational algebra is -

$\Pi_{ename,\ deptloc}(\sigma_{deptname="Account"}(employee \bowtie (Works\_for \bowtie \Pi_{deptID,\ deptloc}(dept))))$

Now applying transformation using 7(a) rule we get

$\Pi_{ename,\ deptloc}((\sigma_{deptname="Account"}(employee)) \bowtie (Works\_for \bowtie \Pi_{deptID,\ deptloc}(dept)))$

Performing selection as early as possible reduces the size of the relation to be joined.

END...

Unit V

5 Transaction and Concurrency Control

Part I : Transaction Management

5.1 : Basic Concept of a Transaction

Q.1 Define the term - 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.**

Q.2 What are the steps followed in executing write(X) command in transaction ? ☞ **[SPPU : May-18, (End Sem), Marks 5]**

Ans. : • Basic operations on data item A are -

1. read_item(X)	2. write_item(X)

1. **read_item(X) :** This is a reading operation in which database item named X is read into a programming variable. We can name the program variable as X for simplification.
2. **write_item(X) :** This operation writes the value of program variable X into the database item which is also named as X.

- Basic unit of data transfer from the disk to the computer main memory is one block.
- **read_item(X) command includes the following steps :**

Step 1 : Find the address of the disk block that contains item X.

Step 2 : Copy that disk block into a buffer in main memory (if that disk block is not already in some main memory buffer).

Step 3 : Copy item X from the buffer to the program variable named X.

- **write_item(X) command includes the following steps :**

Step 1 : Find the address of the disk block that contains item X.

Step 2 : Copy that disk block into a buffer in main memory (if that disk block is not already in some main memory buffer).

Step 3 : Copy item X from the program variable named X into its correct location in the buffer.

Step 4 : Store the updated block from the buffer back to disk.

- **Example of sample transaction performing read and write operations**

```
read_item(X);
X=X+M;
Write_item(X);
```

- **Transaction Notations :**

The transaction notations focuses on read and write operations. For example -Following are two transactions denoted by T1 and T2

```
T1:b1;r1(X);w1(X);r1(Y);W1(Y);e1;
T2:b2;r2(X);r2(Y);e2
```

The r and w represents the read and write operations. The b1 and b2 represents the beginning and e1 and e2 represents ending of transaction.

5.2 : Properties of Transactions

Q.3 What is transaction ? Explain ACID properties of transaction. ☞ **[SPPU : Aug.-17, Marks 5, Dec.-22, Marks 6]**

Ans. : Transaction : Refer Q.1

ACID Properties of Transaction :

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'.**
- For example - In above mentioned withdrawal of money transaction all the five steps must be completed fully or none of the step is completed. Suppose if transaction gets failed after step 3, then the customer will get the money but the balance will not be updated accordingly. The state of database should be either at before ATM withdrawal (i.e customer without withdrawn money) or after ATM withdrawal (i.e. customer with money and account updated). This will make the system in consistent state.

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.
- **For example :** If a bank manager is checking the account balance of particular customer, then manager should see the balance either before withdrawing the money or after withdrawing the money. This will make sure that each individual transaction is completed and any other dependent transaction will get the consistent data out of it. Any failure to any transaction will not affect other transaction in this case. Hence it makes all the transactions consistent.

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.
- For example : In ATM withdrawal example, if the system failure happens after customer getting the money then the system should be strong enough to update Database with his new balance, after system recovers. For that purpose the system has to keep the **log of each transaction and its failure.** So when the system recovers, it should be able to know when a system has failed and if there is any pending transaction, then it should be updated to Database.

Q.4 What are the possible causes of transaction failure ? Explain the significance of ACID properties. ☞ **[SPPU : May-19, (End Sem), Marks 5]**

Ans. : Causes of Transaction Failure : Following are the causes of transaction failure –

1) **Transaction Failure :** Following are two types of errors due to which the transaction gets failed.

Logical Error :

i) This error is caused due to internal conditions such as bad input, data not found, overflow of resource limit and so on.

ii) Due to logical error the transaction can not be continued.

System Error :

i) When the system enters in an undesired state and then the transaction can not be continued then this type of error is called as system error.

2) **System Crash :** The situation in which there is a hardware malfunction, or a bug in the database software or the operating system, and because of which there is a loss of the content of volatile storage, and finally the transaction processing come to a halt is called system crash.

3) **Disk Failure :** A disk block loses its content as a result of either a head crash or failure during a data-transfer operation. The backup of data is maintained on the secondary disks or DVD to recover from such failure.

Significance of ACID Properties : Refer Q.3

5.3 : Database Architecture

Q.5 Explain various states of transaction.

Ans. : Each transaction has following five states :

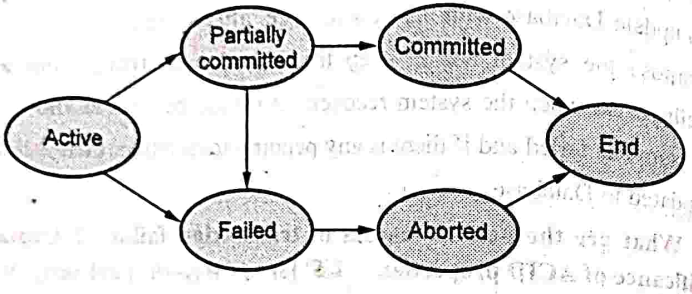


Fig. Q.5.1 Transaction states

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.

5.4 : Serializability

Q.6 What is serializable schedule ? Explain with suitable example the types of serializable schedules.

☞ [SPPU : Aug.-17, Marks 5, Dec.-22, Marks 6]

Ans. : Serializability Schedule :

- When multiple transactions run concurrently, then it may lead to inconsistency of data (i.e. change in the resultant value of data from different transactions).
- Serializability is a concept that helps to identify which non serial schedule and find the transaction equivalent to serial schedule.

For example :

T1	A	B	T2
Initial Value	100	100	
A=A – 10			
W(A)			
B=B+10			
W(B)			
	90	110	
			A=A–10
			W(A)
	80	110	

- In above transactions initially T1 will read the values from database as A=100, B=100 and modify the values of A and B. But transaction T2 will read the modified value i.e. 90 and will modify it to 80 and perform write operation. Thus at the end of transaction T1 value of A will be 90 but at

end of transaction T2 value of A will be 80. Thus conflicts or inconsistency occurs here. This sequence can be converted to a sequence which may give us consistent result. This process is called **serializability.**

There are two types of serializable schedules –

1) Conflict Serializability 2) View Serializability

Conflict Serializability : Refer Q.7

View Serializability : Refer Q.9

Q.7 Explain conflict serializability with suitable 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.

Testing for serializability

Following method is used for testing the serializability : To test the conflict serializability we can draw a graph G = (V, E) where V = vertices which represent the number of transactions.

E = Edges for conflicting pairs.

Step 1 : Create a node for each transaction.

Step 2 : Find the conflicting pairs (RW, WR, WW) on the same variable (or data item) by different transactions.

Step 3 : Draw edge for the given schedule. Consider following cases

1. T_i executes write(Q) before T_j executes read(Q), then draw edge from T_i to T_j.
2. T_i executes read(Q) before T_j executes write(Q) , then draw edge from T_i to T_j
3. T_i executes write(Q) before T_j executes write(Q), , then draw edge from T_i to T_j

Step 4 : Now, if precedence graph is cyclic then it is a non conflict

serializable schedule and if the precedence graph is acyclic then it is conflict serializable schedule.

Example - Refer Q.8

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)	

Ans. :

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

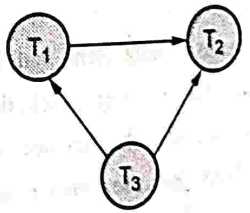


Fig. Q.8.1 Precedence graph

Step 2 : 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 3 : 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 4 : Find the vertex which has no incoming edge which is T_1. Finally find the vertex having no outgoing edge which is T_2. So in between them is T_3. Hence the order will be $T_1 - T_3 - T_2$.

Q.9 Explain view serializability with suitable example.

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.

Steps to check whether the given schedule is view serializable or not

Step 1 : If the schedule is conflict serializable then it is surely view serializable because conflict serializability is a restricted form of view serializability.

Step 2 : If it is not conflict serializable schedule then check whether there exist any blind write operation. The blind write operation is a write operation without reading a value. If there does not exist any blind write then that means the given schedule is not view serializable. In other words if a blind write exists then that means schedule may or may not be view conflict.

Step 3 : Find the view equivalence schedule

Example - Refer Q.10

Q.10 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)

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 -

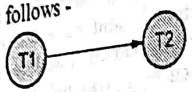


Fig. Q.10.1 Precedence graph

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	

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

write (B)	
	read (B) B: = B + temp
	write (B)

Schedule S

	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.

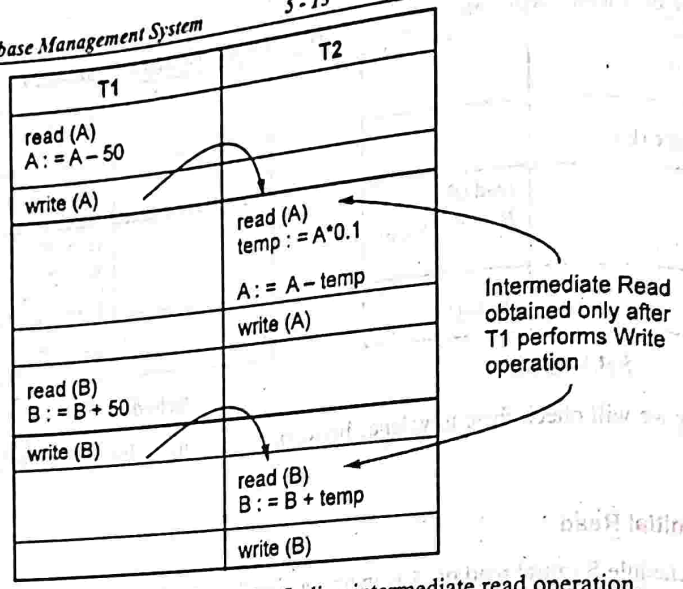
(2) Final Write

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

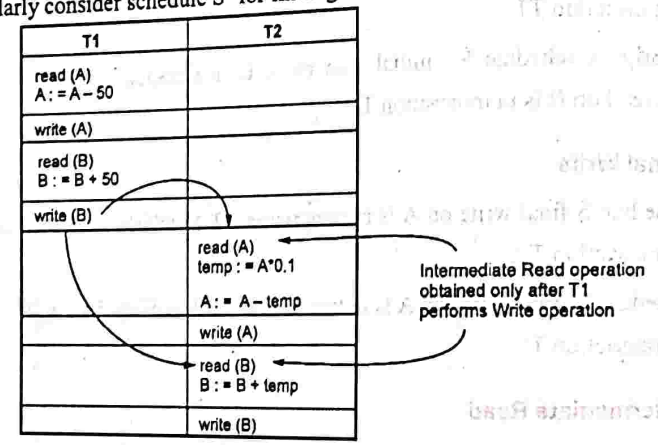
In schedule S' final write on A is in transaction T2. Similarly final write on B is in transaction T2

(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.11 When are two schedules said to be view equivalent ?

☞ **[SPPU : May-18, Nov.-18, (End Sem), Marks 6]**

Ans. : Refer Q.10

Q.12 Illustrate difference between conflict serializable schedule and view serializable schedule by an appropriate example.

☞ **[SPPU : June-22, Marks 6]**

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.
3.	For example - Refer Q.7	For example - Refer Q.9

5.5 : Transaction Isolation and Atomicity

Q.13 Explain the concept of recoverable schedule with example.

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. A transaction is said to be dependable if it contains a **dirty read.**
- 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

- 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.14 Explain the concept of cascadeless schedule with example.

Ans. : Definition : If in a schedule, a transaction is not allowed to read a data item until the last transaction that has written that data item is committed or aborted, then such a schedule is known as a cascadeless schedule.

The cascadeless schedule allows only **committed Read** operation. For example :

T_1	T_2	T_3
R(A)		
A=A+50		
W(A)		
Commit		
	R(A)	
	A=A–20	
	W(A)	
	Commit	
		R(A)
		W(A)

In above schedule at any point if the failure occurs due to commit operation before every Read operation of each transaction, the schedule becomes recoverable and atomicity can be maintained.

Part II : Concurrency Control

5.6 : Need for Concurrency Control

Q.15 What is concurrency control mechanism ?

[SPPU : Dec.-22, Marks 8]

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.**

- The concurrency control can be achieved with the help of various protocols such as - lock based protocol, deadlock handling, multiple granularity, timestamp based protocol, and validation based protocols.

Q.16 What is do you understand by the term - dirty read ? Explain it with suitable example.

Ans. : 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

	T_1	T_2	
Time	...	...	Salary = ₹ 1000
t_1		Update Salary = Salary + 200	Salary = ₹ 1200
t_2	Read		Salary = ₹ 1200
t_3		Rollback	Salary = ₹ 1000

Dirty read

1) At the time t1, 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.

5.7 : Locking Methods

Q.17 Explain two types of locks. Explain the role of locks in transaction management.

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.
- In a set of locks may consists of shared or exclusive locks. Following matrix represents the compatibility between modes of locks.

	S	X
S	T	F
X	F	F

Fig. Q.17.1 Compatibility matrix for locks

Here T stands for True and F stands for False. If the control manager get the compatibility mode as True then it grant the lock otherwise the lock will be denied.

- **For example :** If the transaction T_1 is holding a shared lock in data item A, then the control manager can grant the shared lock to transaction T_2 as compatibility is True. But it cannot grant the exclusive lock as the compatibility is false. In simple words if transaction T_1 is reading a data item A then same data item A can be read by another transaction T_2 but cannot be written by another transaction.
- Similarly if an exclusive lock (i.e. lock for read and write operations) is hold on the data item in some transaction then no other transaction can acquire shared or exclusive lock as the compatibility function denotes F. That means of some transaction is writing a data item A then another transaction can not read or write that data item A.

Q.18 Explain two phase locking protocol and its forms.

[SPPU : Nov.-19, (End Sem), Marks 6, Dec.-22, Marks 4]

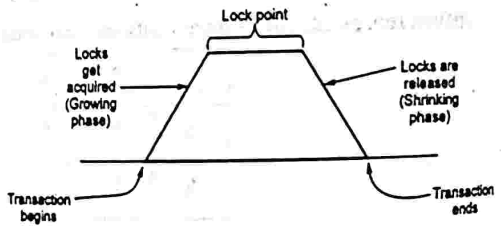
Ans. : • The two phase locking is a protocol in which there are two phases :

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 -

Lock(A)
Lock(B)
Lock(C)
....
...
... Lock Point
Unlock(A)
Unlock(B)
Unlock(C)



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.

Q.19 What are advantages and disadvantages of two phase locking ?

Ans. : 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.

5.8 : Dead Locks

Q.20 What is deadlock ? Explain how deadlock detection and prevention is done. **[SPPU : Nov.-17, 19, (End Sem), June-22, Marks 6, Dec.-22, Marks 8]**

Ans. : Definition : 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.

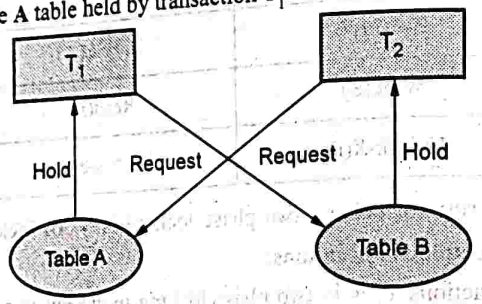


Fig. Q.20.1

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 can be handled using two techniques -

1. Deadlock prevention 2. Deadlock detection and deadlock recovery

1. 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 Tj 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.

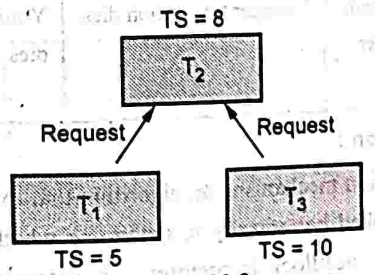


Fig. Q.20.2

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.

To summarize

	Wait-Die	Wound-wait
Older transaction needs a data item held by younger transaction	Older transaction waits	Younger transaction dies.
Younger transaction needs a data item held by older transaction	Younger transaction dies	Younger transaction dies.

2. 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_1 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

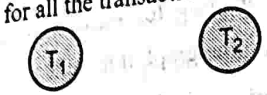


Fig. Q.20.3

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 -

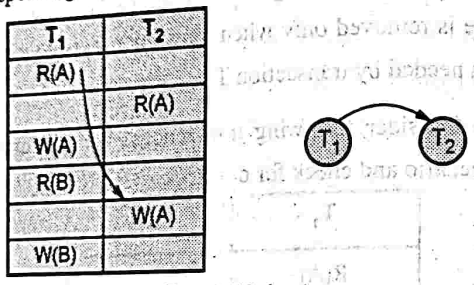


Fig. Q.20.4

Step 3 :

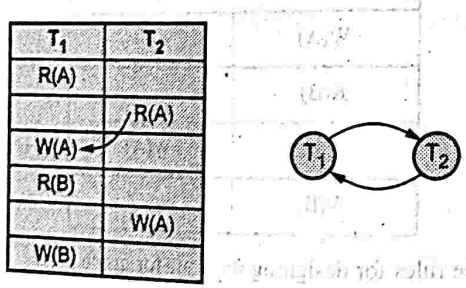


Fig. Q.20.5

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

Q.21 What is a deadlock ? Explain deadlock recovery techniques.

[SPPU : June-22, Marks 6]

Ans. : Refer Q.20

5.9 : Time Stamping Methods

Q.22 Explain time-stamp ordering protocol.

[SPPU : May-19, (End Sem), Dec.-22, 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.

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

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

Fig. Q.22.1

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 T_1 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.22.2

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

Q.23 What is concurrency control ? Explain time stamp based concurrency control. [SPPU : Nov.-17, (End Sem), Marks 6]

Ans. : Concurrency Control : Refer Q.15

Time-stamp Based Concurrency Control : Refer Q.22

5.10 : Optimistic Techniques and Multi-version Concurrency Control

Q.24 Explain optimistic concurrency control mechanism in detail.

Ans. : • The optimistic concurrency control algorithm is basically a validation based protocol.

- It works in three phases -
  - **Read Phase :**
    - In this phase, transaction T is read.
    - The values of various data items are read and stored in temporary variables.
    - All the operations are then performed on temporary variables without updating the actual database.
  - **Validation Phase :**
    - In this phase, the temporary variable value is validated against the actual data in the database and it checked whether the transaction T follows serializability or not.

  - **Write Phase :**
    - If the transaction T is validated then only the temporary results are written to database, otherwise the system rolls back.
- Each phase has following **different timestamps**
  - **Start (T_i) :**
    - It contains the timestamp when Ti starts the execution.
  - **Validation (Ti) :**
    - It contains the timestamp when transaction Ti finishes the read phase and starts its validation phase.
  - **Finish (Ti) :**
    - It contains the timestamp when transaction Ti finishes its write phase.
    - With the help of timestamp in validation phase, this protocol determines if the transaction will commit or rollback. Hence $TS(T_i)$ = validation (Ti).
    - The serializability is determined at the validation process, it can't be determined in advance.
    - While executing the transactions, this protocol gives **greater degree of concurrency when there are less number of conflicts**. That is because the serializability order is not pre-decided (validated and then executed) and relatively less transactions will have to be rolled back.

5.11 : Granularity of Data Items and Multiple Granularity

Q.25 What are the issues in determining lock granularity ? What is multiple granularity ? [SPPU : May-19, (End Sem), June-22, Marks 5]

Ans. : • In database management system there are **data at various levels** depending upon the **data sizes.**

- For example : A database contains a table and a table contains rows and each row contains some data value. Thus different levels of data can be database, tables, records.
- This can be represented in the form of tree. For example.

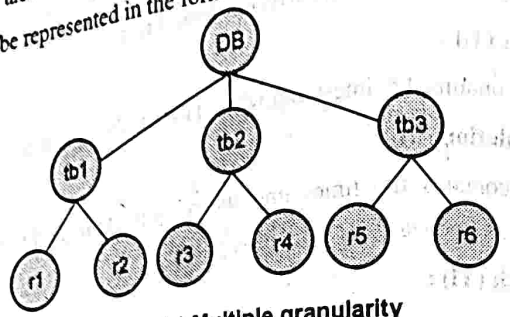


Fig. Q.25.1 Multiple granularity

- Each node **can be locked** individually. When a transaction locks a node, in either shared or exclusive mode, the transaction also **implicitly locks** all the descendants of that node in the same lock mode.
- For example - If transaction T1 gets an explicit lock on table tb1 in exclusive mode, then it has an implicit lock in exclusive mode on all the records belonging to that table. It does not need to lock the individual records r1 and r2.
- **A new mode of lock** is introduced along with exclusive and shared locks. This is called **intention mode lock.** Thus in addition to S and X lock modes, there are three **additional lock modes** with multiple granularity :
  - **Intention-Shared (IS) :** Explicit locking at a lower level of the tree but only with shared locks.
  - **Intention-Exclusive (IX) :** Explicit locking at a lower level with exclusive or shared locks.
  - **Shared and Intention-Exclusive (SIX) :** The sub-tree rooted by that node is locked explicitly in shared mode and explicit locking is being done at a lower level with exclusive mode locks.

- The **compatibility matrix** for these lock modes are described below :

	IS	IX	S	SIX	X
IS	True	True	True	True	False
IX	True	True	False	False	False
S	True	False	True	False	False
SIX	True	False	False	False	False
X	False	False	False	False	False

- With help of intention lock modes on the transactions the multiple-granularity locking protocol ensures serializability.
- The protocol follows **following rules -**
  1) Transaction Ti must follow the lock-compatibility matrix.
  2) Transaction Ti must lock the root of the tree first, and it can lock it in any mode.
  3) Transaction Ti can lock a node in S or IS mode only if Ti currently has the parent of the node locked in either IX or IS mode.
  4) Transaction Ti can lock a node in X, SIX, or IX mode only if Ti currently has the parent of the node locked in either IX or SIX mode.
  5) Transaction Ti can lock a node only if Ti has not previously unlocked any node. That means Ti is two phase.
  6) Transaction Ti can unlock a node only if Ti currently has none of the children of the node locked.

Q.26 How does the granularity of data items affect the performance of concurrency control ? What factors affect the selection of granularity size of data items ? **[SPPU : June-22, Marks 4]**

Ans. : Refer Q.25

5.12 : Different Crash Recovery Methods

Q.27 List down all the possible crash recovery methods. Explain shadow paging with proper example.

[SPPU : Nov.-17, (End Sem), Marks 8, Dec.-22, Marks 4]

Ans. : Following are commonly used crash recovery methods –

1) Shadow paging
2) Log-based recovery
   a. Deferred Update
   b. Immediate Update
3) Checkpointing

Shadow Paging :

- Shadow paging is a recovery scheme in which database is considered to be made up of number of fixed size disk pages.
- A directory or a page table is constructed with n number of pages where each i^{th} page points to the i^{th} database page on the disk. (Refer Fig. Q.27.1)

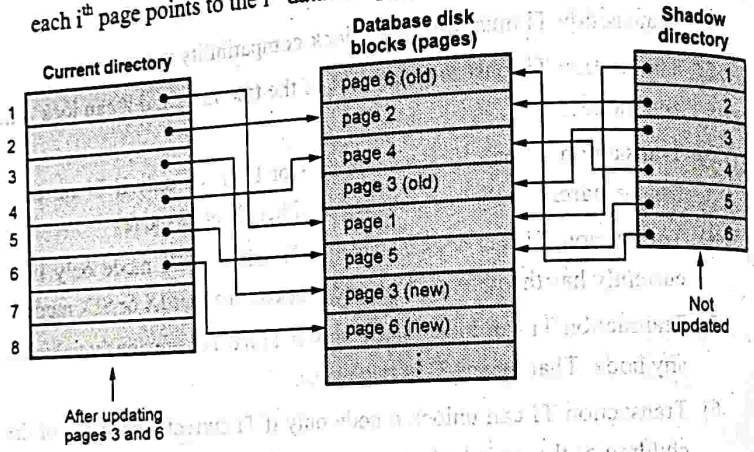


Fig. Q.27.1 Demonstration of Shadow Paging

- The directory can be kept in the main memory.
- When a transaction begins executing, the **current directory**-whose entries point to the most recent or current database pages on disk-is copied into a another directory called **shadow directory**.
- The shadow directory is then saved on **disk** while the current directory is used by the transaction.

- During the execution of transaction, the shadow directory is never modified.
- When a write operation is to be performed then the new copy of modified database page is created but the old copy of database page is never overwritten. This newly created database page is written somewhere else.
- The current directory will point to newly modified web page and the shadow page directory will point to the old web page entries of database disk.
- When the failure occurs then the modified database pages and current directory is discarded.
- The state of database before the failure occurs is now available through the shadow directory and this state can be recovered using shadow directory pages.
- This technique does not require any UNDO/REDO operation.

5.13 : Log-based Recovery

Q.28 Explain the concept of log and how it helps with database recovery with suitable diagram. ☞ **[SPPU : May-19, (End Sem), Marks 8]**

Ans. : Log is the most commonly used structure for recording the modifications that is to be made in the actual database. Hence during the recovery procedure a log file is maintained.

- A log record maintains four types of operations. Depending upon the type of operations there are four types of log records -
  1. **<Start>** Log record : It is represented as $<T_i$, Start>
  2. **<Update>** Log record
  3. **<Commit>** Log record : It is represented as $<T_i$, Commit>
  4. **<Abort>** Log record : It is represented as $<T_i$, Abort>
- The log contains various fields as shown in following Fig. Q.28.1. This structure is for <update> operation

Transaction ID(T_i)	Data Item Name	Old Value of Data Item	New Value of Data Item

Fig. Q.28.1

- For example : The sample log file is

<T_1, Start>
<T_1,a,10,20>
<T_1, Commit>

Here 10 represents the old value before commit operation and 20 is the new value that needs to be updated in the database after commit operation.

- The log must be maintained on the stable storage and the entries in the log file are maintained before actually updating the physical database.
- There are two approaches used for log based recovery technique . Deferred Database Modification and Immediate Database Modification.

Q.29 What is the basis of immediate updates recovery technique ? What does the deferred updates recovery technique involve ?

☞ **[SPPU : May-19, (End Sem), Marks 8]**

Ans. : Immediate Database Update :

In this technique, the database is updated during the execution of transaction even before it reaches to its commit point.

If the transaction gets failed before it reaches to its commit point, then the a **ROLLBACK Operation** needs to be done to bring the database to its earlier consistent state. That means the effect of operations need to be undone on the database. For that purpose both Redo and Undo operations are both required during the recovery. This technique is known as **UNDO/ REDO** technique.

For example : Consider two transaction T_1 and T_2 as follows :

T_1	T_2
Read(A,a)	Read(C, c)
a = a –10	c = c – 20
Write(A, a)	Write(C, c)
Read(B, b)	
b = b+10	
Write(B, b)	

Here T_1 and T_2 are executed serially. Initially A = 100, B = 200 and C = 300

If the crash occurs after

i) Just after Write(B, b) ii) Just after Write(C, c)

iii) Just after <T_2,Commit>

Then using the immediate Database modification approach the result of above three scenarios can be elaborated as follows :

The contents of **log** and **database** is as follows :

Log	Database
<T_1,Start>	
<T_1,A,100,90>	A = 90
<T_1,B,200,210>	B = 210
<T_1,Commit>	
<T_2,Start>	
<T_2,C,300,280>	C = 280
<T_2,Commit>	

The recovery scheme uses two recovery techniques -

i) **UNDO (T_i) :** The transaction T_i needs to be undone if the log contains <T_i,Start> but does not contain <T_i,Commit>. In this phase, it restores the values of all data items updated by T_i to the old values.

ii) **REDO (T_i) :** The transaction T_i needs to be redone if the log contains both <T_i,Start> and <T_i,Commit>. In this phase, the data item values are set to the new values as per the transaction. After a failure has occurred log record is consulted to determine which transaction need to be redone.

a) **Just after Write (B, b) :** When system comes back from this crash, it sees that there is <T_1, Start> but no <T_1, Commit>. Hence T_1 must be undone. That means old values of A and B are restored. Thus old values of A and B are taken from log and both the transaction T_1 and T_2 are re-executed.

b) **Just after Write (C, c) :** Here both the redo and undo operations will occur.

c) **Undo :** When system comes back from this crash, it sees that there is <T_2, Start> but no <T_2, Commit>. Hence T_2 must be undone. That means old values of C is restored.

Thus old value of C is taken from log and the transaction T_2 is re-executed.

d) **Redo :** The transaction T_1 must be done as log contains both the <T_1,Start> and <T_1,Commit>

So A = 90, B = 210 and C = 300

e) **Just after <T_2, Commit> :** When the system comes back from this crash, it sees that there are two transaction T_1 and T_2 with both start and commit points. That means T_1 and T_2 need to be redone. So A = 90, B = 210 and C = 280

Deferred Database Update :

- In this technique, the database is not updated immediately.
- Only log file is updated on each transaction.
- When the transaction reaches to its commit point, then only the database is physically updated from the log file.
- In this technique, if a transaction fails before reaching to its commit point, it will not have changed database anyway. Hence there is no need for the UNDO operation. The REDO operation is required to record the operations from log file to physical database. Hence deferred database modification technique is also called as **NO UNDO/REDO algorithm.**
- **For example :**

Consider two transactions T_1 and T_2 as follows :

T_1	T_2
Read (A, a)	Read (C, c)
a = a – 10	c = c – 20

Write (A, a)	Write (C, c)
Read (B, b)	
b = b + 10	
Write (B, b)	

If T_1 and T_2 are executed serially with initial values of A = 100, B = 200 and C = 300, then the state of log and database if crash occurs

a) Just after write (B, b)

b) Just after write (C, c)

c) Just after <T_2, commit>

The result of above 3 scenarios is as follows :

Initially the log and database will be

Log	Database
<T_1, Start>	
<T_1, A, 90>	
<T_1, B, 210>	
<T_1, Commit>	
	A = 90
	B = 210
<T_2, Start>	
<T_2, C, 280>	
<T_2, Commit>	
	C = 280

a) Just after write (B, b)

Just after write operation, no commit record appears in log. Hence no write operation is performed on database. So database retains only old values. Hence A = 100 and B = 200 respectively.

Thus the system comes back to original position and no redo operation take place.

The incomplete transaction of T_1 can be deleted from log.

b) Just after write (C, c)

The state of log records is as follows

Note that crash occurs before T_2 commits. At this point T_1 is completed successfully, so new values of A and B are written from log to database. But as T_2 is not committed, there is no redo (T_2) and the incomplete transaction T_2 can be deleted from log.

The redo (T_1) is done as < T_1, commit> gets executed. Therefore A = 90, B = 210 and

C = 300 are the values for database.

c) Just after < T_2, commit>

The log records are as follows :

< T_1, Start>

< T_1, A, 90>

< T_1, B, 210>

< T_1, Commit>

< T_2, Start>

< T_2, 6, 280>

< T_2, Commit>

← Crash occurs here

Clearly both T_1 and T_2 reached at commit point and then crash occurs. So both redo (T_1) and redo (T_2) are done and updated values will be A = 90, B = 210, C = 280.

Q.30 Explain log-based recovery technique.

[SPPU : Nov.-19, (End Sem), Marks 6]

Ans. : Refer Q.28

5.14 : Check Points

Q.31 What is a checkpoint ? Explain the operations performed by a system during checkpoint. Explain the recovery mechanism during system crash.

[SPPU : May-18, (End Sem), Marks 8, Nov.-18, (End Sem), Marks 10]

Ans. : • Checkpoint is a mechanism where all the previous logs are removed from the system and stored permanently in a storage disk.

- Checkpoint declares a point before which the DBMS was in consistent state, and all the transactions were committed.
- The recovery system reads the logs backwards from the end to the last checkpoint.
- Performing a checkpoint consists of the following operations :
  - Suspending executions of transactions temporarily;
  - Writing (force-writing) all modified database buffers of committed transactions out to disk;
  - Writing a checkpoint record to the log; and
  - Writing (force-writing) all log records in main memory out to disk.
- A checkpoint record usually contains additional information, including a list of transactions active at the time of the checkpoint.
- Many recovery methods (including the deferred and immediate update methods) need this information when a transaction is **rolled back**, as all transactions active at the time of the checkpoint and any subsequent ones may need to be redone.
- Since checkpoints cause some loss in performance while they are being taken, their frequency should be reduced if fast recovery is not critical.
- If we need fast recovery check-pointing frequency should be increased. If the amount of stable storage available is less, frequent check-pointing is unavoidable.

Recovery Mechanism During System Crash : Refer Q.29

END...

Unit VI

6 Advanced Databases

Part I : Database Architectures

6.1 : Centralized and Client-Server Architectures

Q.1 Explain centralized architecture.

☞ [SPPU : Dec.-22, Marks 6]

Ans. :

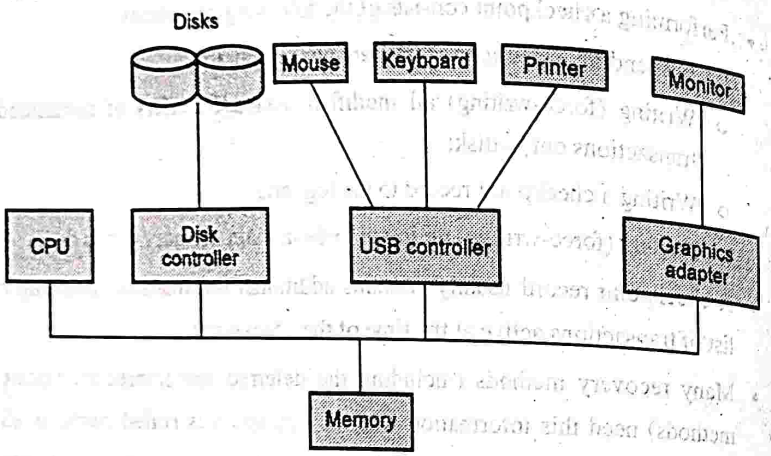


Fig. Q.1.1 Centralized systems

1) Exists on single system that **doesn't interact** with any other system
2) Single user system will have **one CPU** with one or two hard disks supported by single OS
3) Multiuser system have multiple CPUs, many disks and more memory where multiple users are connected to the system via terminals
4) A standalone general purpose system will have a single **shared memory** accessed through common bus.

Q.2 Explain client-server architecture.

Ans. :

☞ [SPPU : Dec.-22, Marks 6]

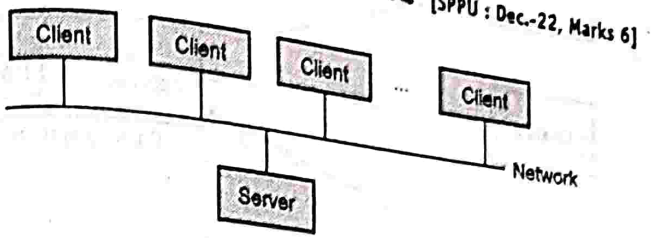


Fig. Q.2.1

A network of workstations are connected to the backend servers to achieve :

1) Improved functionality saving the cost
2) Scaling up of the resources is easily possible
3) User interfaces can be designed effectively
4) Easy to maintain and improves efficiency.

6.2 : The 2 Tier and 3 Tier Architecture

Q.3 Explain 2-tier and 3-tier architecture with diagram for online banking database system.

☞ [SPPU : June-22, Marks 6]

Ans. : Two tier architecture for online banking system

- In a two-tier architecture, the client is on the first tier.
- The database server and web application server reside on the same server machine, which is the second tier. This second tier serves the data and executes the business logic for the web application.
- Banking system that favor this architecture consolidate the application capabilities and database server capabilities on a single tier.

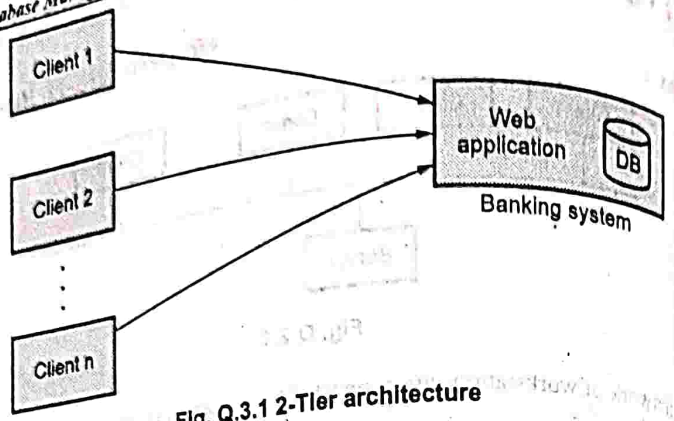


Fig. Q.3.1 2-Tier architecture

Three tier architecture for online banking system

- The banking application logic resides in the very middle-tier. It stays totally separated from the UI and the banking database.
- The User Interface (UI) lies at the client's web browser. The banking database server is separated from middle tier.
- The three tier architecture of online banking system is as shown below.

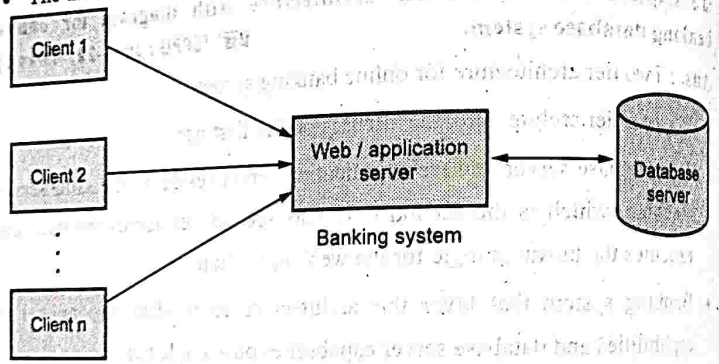


Fig. Q.3.2 3-Tier architecture

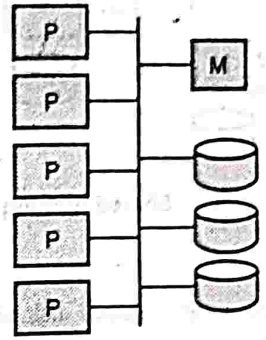
6.3 : Introduction to Parallel Databases

Q.4 Explain any two parallel database system architecture in detail.

☞ [SPPU : June-22, Dec.-22, Marks 6]

Ans. : Shared memory :

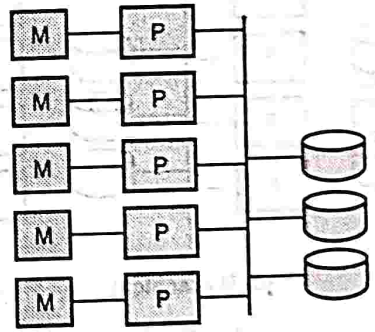
In this architecture, the processor and disks access common memory via interconnection network among multiple processors. The high speed connection is used in this architecture. Each processor do have large cache memory.



(a) Shared memory

Shared disk :

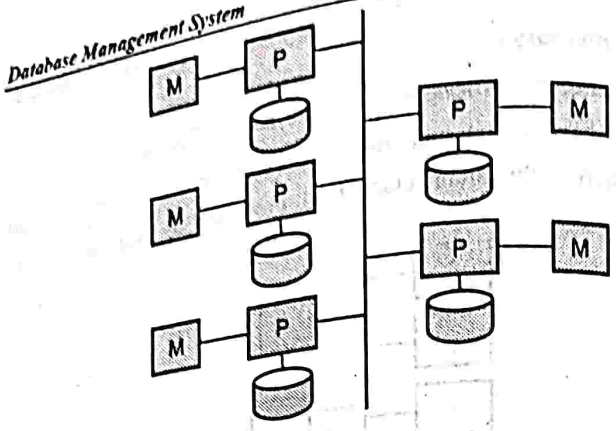
In this architecture, the multiple processors access common disk via interconnection network. Each processor have a private memory that helps in data sharing.



(b) Shared disk

Shared nothing :

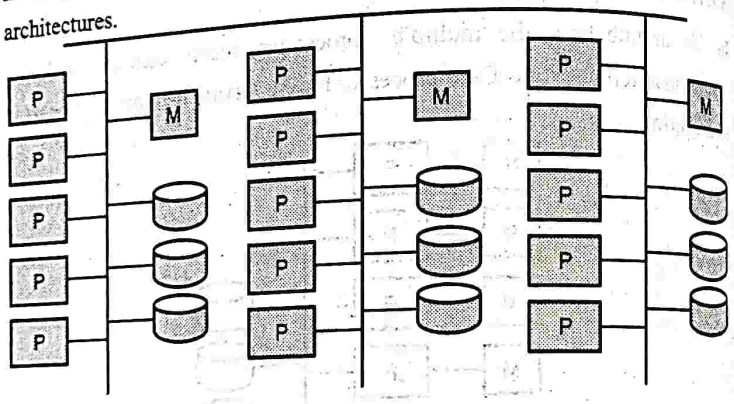
In this architecture, the processor,disks and memory units are separate for every system and uses interconnection network for parallel processing.



(c) Shared nothing

Hierarchical :

This architecture is a combination of shared disk, shared memory and shared nothing architectures. This architecture is scalable due to availability of more memory and many processors. But it is costly compared to other architectures.



(d) Hierachical

Fig. Q.4.1

Q.5 Explain need of partitioning techniques used in I/O parallelism. Explain techniques in detail. ☞ **[SPPU : June-22, Marks 5]**

Ans. : • I/O parallelism is a form of parallelism in which relations are partitioned across multiple disk.

- The main purpose of I/O parallelism is to reduce the retrieval time of relations from disks.
- In this technique, the inputted data is partitioned and processing is done in parallel with each partition. The results are merged after processing all the partitioned data.
- It is also called as **data partitioning.**
- There are four types of partitioning in I/O parallelism -
  - **Hash partitioning :** Hash function is a mathematical function. The hash function is applied on each row of the relation. If there are five disks disk1, disk2, disk3, disk4, disk5 and if, the hash function returns 4 then the row is placed on disk4.
  - **Range partitioning :** Continuous attribute values ranges to each disk. For example - if we have 3 disks then we can assign values 0 to 20 to disk1, 21-to 40 to disk2 and so on.
  - **Round robin partitioning :** In this schema, the i^{th} tuple is sent to the disk number (i % n). So, disks take turns receiving new rows of data. This technique ensures the even distribution of tuples across disks.
  - Schema partitioning : In this scheme, different tables from a database are placed on different disks. For example - Relation1(table) is placed on disk1, Relation2 is placed on disk2 and so on.

Q.6 Explain distributed database systems.

☞ **[SPPU : Dec.-22, Marks 5]**

Ans. : When multiple geographically distributed databases are logically interconnected and are under the control of a normal site is termed as Distributed Database System (DDS). Refer Fig. Q.6.1

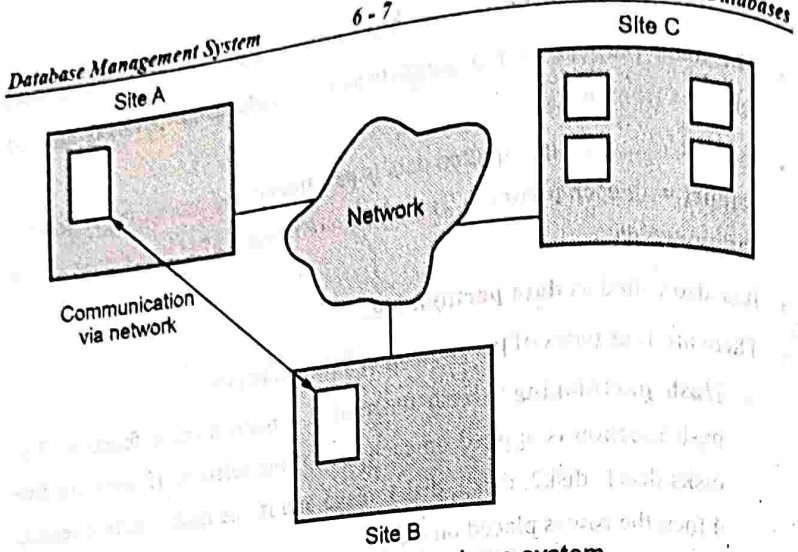


Fig. Q.6.1 Distributed database system

Q.7 Write a short note on centralized and distributed database systems.

[SPPU : June-22, Marks 6]

Ans. : Refer Q.1 and Q.6.

6.4 : Architecture of Distributed Databases

Q.8 Give the distributed database systems classification.

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.8.1.

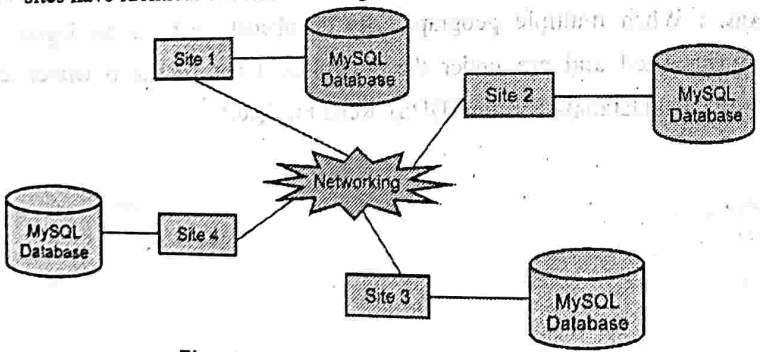


Fig. Q.8.1 Homogeneous databases

- 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.8.2.

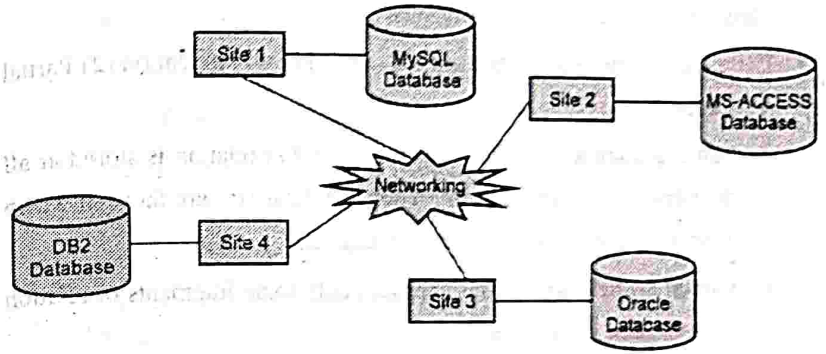


Fig. Q.8.2 Heterogeneous databases

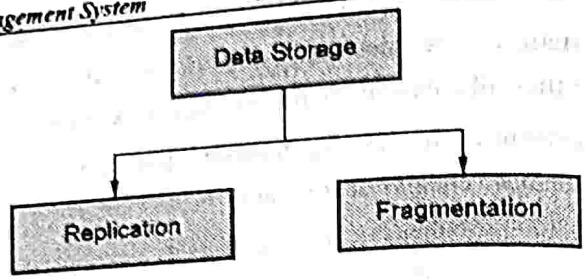
- The participating sites are not aware of other sites present in the system.
- These sites provide limited facilities for cooperation in transaction processing.

6.5 : Distributed Database Design

Q.9 What are the two approaches of storing relation r in distributed database system ?

Ans. : There are two approaches of storing relation r in distributed database -

(1) Replication : System maintains multiple copies of data, stored in different sites, for faster retrieval and fault tolerance.



(2) Fragmentation : Relation is partitioned into several fragments stored in distinct sites.

Q.10 Explain the 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.

Q.11 Enlist the advantages and disadvantages of replication.

☞ **[SPPU : June-22, Marks 5]**

Ans. : 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.

2) **Increased complexity in concurrency control :** It becomes complex to implement the concurrency control mechanism for all the sites containing replica.

Q.12 What are different data fragmentation techniques in distributed databases ?

☞ **[SPPU : June-22, Marks 6]**

Ans. : Concept : Data fragmentation is a division of relation r into fragments $r_1, r_2, r_3, \ldots, r_n$ 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 r_1 and r_2 fragments, then to bring these fragments back to R we must use union operation. That means $R = r_1 \cup r_2$
  - **Vertical fragmentation :** In this approach, the relation r is fragmented based on one or more columns. If relation R is fragmented into r_1 and r_2 fragments using vertical fragmentation then to bring these fragments back to original relation R we must use join operation. That means $R = r_1 \bowtie r_2$.
- 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

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.13 If we are to ensure atomicity, all the sites in which a transaction T executed must agree on the final outcome of the execution T must either commit at all sites, or it must abort at all sites. Describe the two phase commit protocol used to ensure this property in detail.

[SPPU : June-22, Marks 8]

Ans. : The two phase 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.

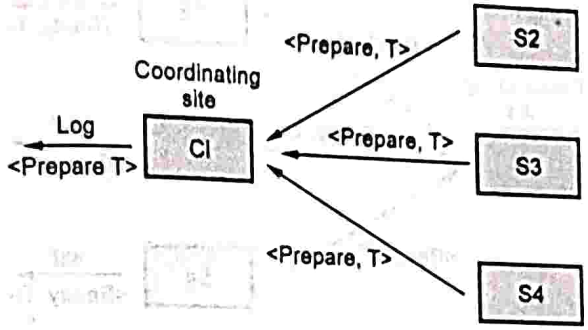


Fig. Q.13.1

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.

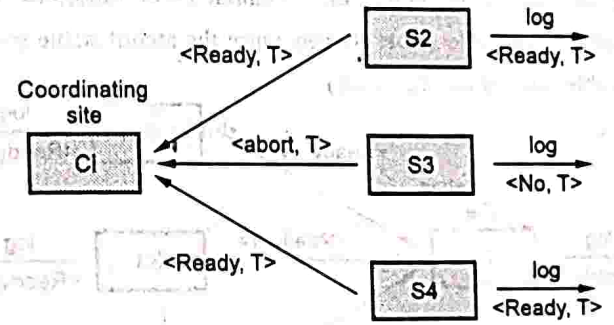


Fig. Q.13.2

- 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

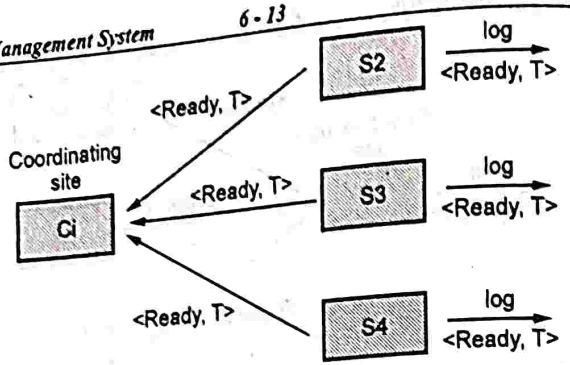


Fig. Q.13.3

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)

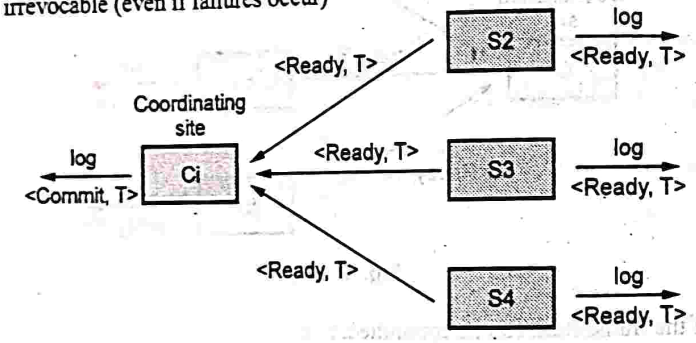


Fig. Q.13.4

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

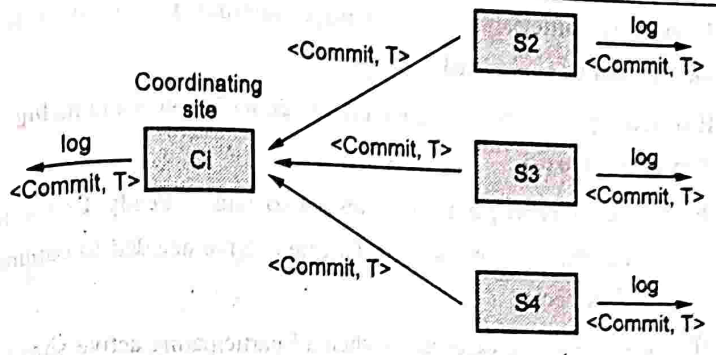


Fig. Q.13.5

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.

Part II : Emerging Database Technologies

6.6 : No SQL Databases - Internet Databases

Q.14 Explain the concept of WWW.

Ans. : • The World Wide Web, or "the Web" has been one of the great successes in the history of computing.

- The Web is the most popular and powerful-networked information system till date.
- The combinations of the Web technology and databases have resulted into many new opportunities for creating advanced database applications.
- The World Wide Web is a subset of the Internet.
- The WWW uses computers called **Web servers** to store multimedia files. These multimedia files are called Web pages that are stored at locations known as **Web sites.**
- The working of WWW is based on Hypertext Transmission Protocol (HTTP).
- The HTTP is a communication protocol for providing, organising and accessing a wide variety of resources such as text, video (images) and

audio (sounds) that are available via the Internet. Following Fig. Q.14.1 shows architecture of web database

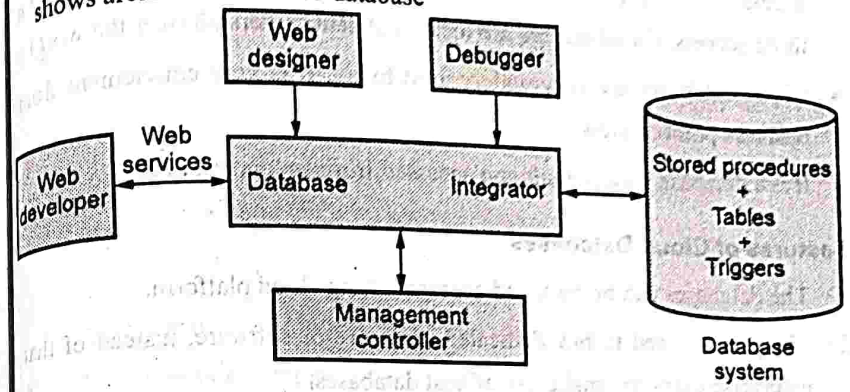


Fig. Q.14.1 Typical architecture of web database

Q.15 Write short note on - internet database.

Ans. : • Web database is a database system that store web application data and managed or accessed through the Internet.

- It contains data regarding E-commerce, E-business, email, and online web applications.
- Database information can be published on the Web in two different formats :

  1) Static Web publishing

  2) Dynamic Web publishing.

Static Web publishing simply involves creating a list or report, based on the information stored in a database and publishing it online. Static Web publishing is a good way to publish information that does not change often and does not need to be filtered or searched. Dynamic Web publishing involves creating pages "on the fly" based on information stored in the database.

6.7 : Cloud Databases

Q.16 Write short note on – Cloud database.

Ans. : • The meaning of the word cloud is collections of servers that are accessed over the Internet and the software and databases that run on those servers. Cloud servers are located in data centers all over the world.

- Cloud databases are very widely used today. It is very convienient from business point of view.
- It is a database service built and accessed from a cloud platform.

Features of Cloud Databases

1) The databases can be built and accessed using cloud platform.
2) There is no need to buy dedicated hardware or software, instead of that enterprise users can make use of host databases.
3) It can be accessed through web interfaces.
4) The cloud databases can be made available by distributing resources and data across different geographical zones.
5) It can support all types of data formats.

6.8 : Mobile Databases

Q.17 Write short note on - Mobile database.

Ans. : Mobile database is a kind of database that can be connected by mobile computing device over a mobile network.

Significance :

1) Many applications need to download information from an information repository and operate on this information even when out of range or disconnected. For example of - Consider your contacts and calendar on the phone. In this scenario, we the mobile user would require access to update information from files in the home directories on a server or customer records from a database. This type of access is possible due to mobile database.
2) Due to mobile database information on customer and market trends, competitors can be maintained.

3) The data in database can be accessed from anywhere using mobile database. Thus it provides wireless access. For instance : a sales person can update sales and customer data due to mobile database.
4) Physicians can store and retrieve information about the patients in mobile database.
5) Mobile database makes use of the data present on the mobile devices and using those data the customized and personalized mobile applications can be created.

Architecture of Mobile Databases

Using mobile databases, users have access to data on their laptop computers, Personal Digital Assistant (PDA), or other Internet access device that is required for applications at remote sites. Following Fig. Q.17.1 shows its architecture

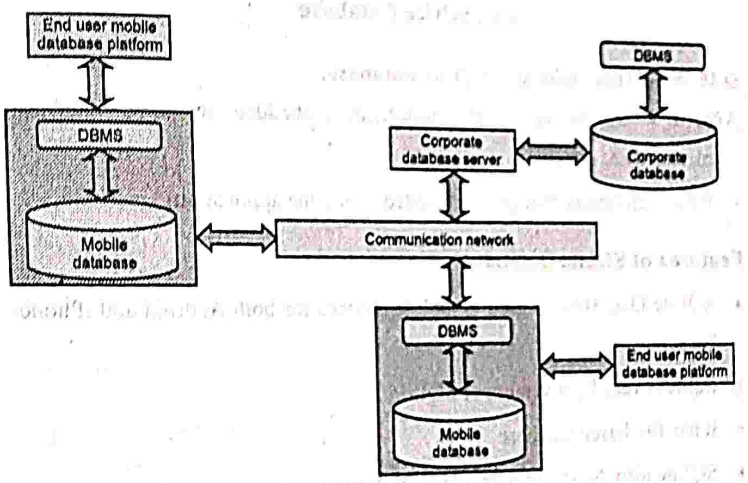


Fig. Q.17.1 General architecture of mobile database

- Mobile database architecture is a **distributed architecture.** The meaning of distributed architecture is that the architecture where several computers, generally referred to as database servers (or hosts) and those are interconnected through a high-speed communication network.

- Mobile database consists of the following **components :**
  - Corporate database server and DBMS to manage and store the corporate data and provide corporate applications.
  - Mobile (remote) database and DBMS at several locations to manage and store the mobile data and provide mobile applications.
  - End user mobile database platform consisting of laptop computer, PDA and other Internet access devices to manage and store client (end user) data and provide client applications.
  - Communication links between the corporate and mobile DBMS for data access. The communication between the corporate and mobile databases is usually established for short period of time at irregular intervals.

6.9 : SQLite Database

Q.18 Write short note on - SQLite database.

Ans. : • SQLite is a very powerful, embedded relational database management system.

- It has rich library that gets embedded inside the application.

Features of SQLite Database

- SQLite Data Base is used in mobile devices for both Android and iPhone based.
- Sqlite is very light version of SQL.
- It has file-based database
- SQLite also doesn't require a special database server or anything. It is just a direct filesystem engine that uses SQL syntax
- It can handle all sorts of data and can be processed by relational databases very easily.
- SQLite is fast, efficient and very powerful.

Advantages of SQLite

1. SQLite is a very **light weighted database** hence it can be easily embedded with devices like televisions,Mobile phone and so on.
2. Reading and writing **operations are very fast** for SQLite database.
3. There is no installation needed.
4. It is easy to learn.
5. Due to availability of continuous updates, it is **reliable.** SQLite queries are smaller hence chances of having bugs is less.
6. It is **portable.** Therefore it can be used with all programming languages without any compatibility issue.
7. It reduces application cost and complexity.

SQLite Disadvantages

1. SQLite is used to handle only low to medium traffic HTTP requests.
2. Database size is restricted.

Q.19 Write short note on any three :
a. SQLite database b. Mobile database
c. Internet database d. Cloud database

☞ **[SPPU : May-18, Marks 12, May-19, Marks 6, Dec.-19, Marks 8]**

Ans. :

a) SQLite database : Refer Q.18.

b) Mobile database : Refer Q.17.

c) Internet database : Refer Q.14.

d) Cloud database : Refer Q.16.

6.10 : XML Databases

Q.20 Explain XML model. List advantages of XML.

☞ **[SPPU : Dec.-18, Marks 8]**

Ans. : • XML stands for eXtensible Markup Language.

- This scripting language is similar to HTML. That means, this scripting language contains various tags. But these tags are not predefined tags, in-fact user can define his own tags.
- Thus HTML is designed for **representation of data** on the web page whereas the XML is designed for **transport or to store data.**

Uses of XML

1. XML is used to display the meta contents i.e. XML describes the content of the document.
2. XML is useful in exchanging data between the applications.
3. The data can be extracted from database and can be used in more than one application. Different applications can perform different tasks on this data.

Advantages of XML

1. XML document is human readable and we can edit any XML document in simple text editors.
2. The XML document is language neutral. That means a Java program can generate an XML document and this document can be parsed by Perl.
3. Every XML document has a tree structure. Hence complex data can be arranged systematically and can be understood in simple manner.
4. XML files are independent of an operating system.

For example –

```
<bank>
   <account>
      <account-number> S101 </account-number>
      <branch-name> Shivaji Nagar </branch-name>
      <balance> 5000 </balance>
   </account>
   <account>
      <account-number>C102 </account-number>
      <branch-name> Model Colony </branch-name>
      <balance> 4000 </balance>
```



```
   </account>
   <customer>
      <customer-name> Ram Kumar </customer-name>
      <customer-street> Fergusson Road</customer-street>
      <customer-city> Pune </customer-city>
   </customer>
   <customer>
      <customer-name> Shiv Prasad </customer-name>
      <customer-street> Main Road </customer-street>
      <customer-city> Nasik </customer-city>
   </customer>
   <depositor>
      <account-number> S101 </account-number>
      <customer-name> RamKumar </customer-name>
   </depositor>
   <depositor>
      <account-number> C102 </account-number>
      <customer-name> Shiv Prasad </customer-name>
   </depositor>
</bank>
```

Fig. Q.20.1 XML representation of bank information

Q.21 What are XML namespaces ? How to solve name conflict ?

[SPPU : May-19, Marks 6]

Ans. : • Sometimes we need to create two different elements by the same name. The xml document allows us to create **different elements** which are having the **common name.** This technique is known as **namespace.**

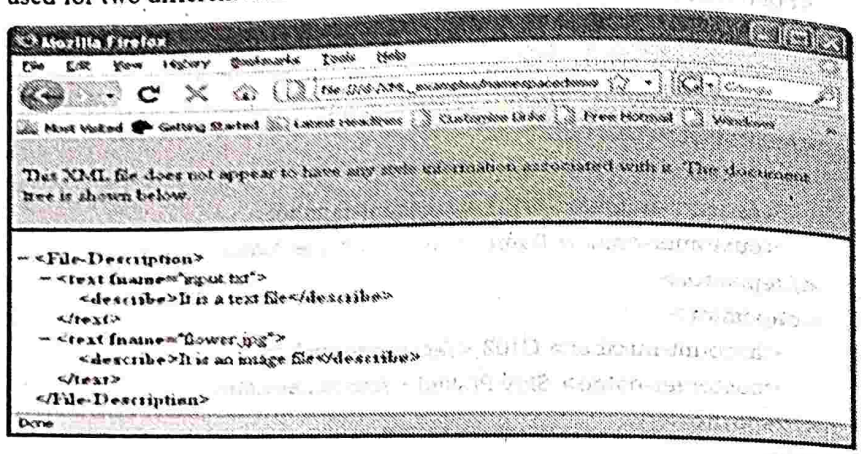
- In some web documents it becomes necessary to have the same name for two different elements. Here different elements mean the elements which are intended for different purposes. In such a case support for namespace technique is very much helpful.
- For example : Consider the following xml document -

namespacedemo.xml

```
<File-Description>
   <text fname="input.txt">
      <describe>It is a text file</describe>
   </text>
```

```
<text fname="flower.jpg">
    <describe>It is an image file</describe>
</text>
</File-Description>
```

The above document does not produce any error although the element **text** is used for two different attribute values. The output will be,



END...

June-2022 [END SEM] - [5869] - 288 | Solved Paper

Course 2019

Time : $2\frac{1}{2}$ Hours] **[Maximum Marks : 70**

Instructions to the candidates :

1) ***Answer Q.1 or Q.2, Q.3 or Q.4, Q.5 or Q.6, Q.7 or Q.8.***
2) ***Neat diagrams must be drawn wherever necessary.***
3) ***Figures to the right side indicate full marks.***
4) ***Assume suitable data, if necessary.***
5) ***Use of scientific calculator is permitted.***

Q.1 **a)** *Consider following database :*

Student (Roll_no, Name, Address)

Subject (Sub_code, Sub_name)

Marks (Roll_no, Sub_code, marks)

Write following queries in SQL :

i) Find average marks of each student, along with the Roll_no of student of subject code 'CE2412'.

ii) Find how many students have failed in the subject "DBMS".

iii) Construct suitable view on above schema.

(Refer Q.23 of Chapter - 3) **[6]**

b) *Explain on delete cascade command with suitable example.*

(Refer Q.8 of Chapter - 3) **[5]**

c) *What are different types of joins in SQL ? Explain with suitable example.* **(Refer Q.16 of Chapter - 3)** **[7]**

OR

Q.2 **a)** *Explain with suitable example SQL aggregate functions.* **(Refer Q.14 of Chapter - 3)** **[6]**

b) *Write the syntax for following SQL commands :*

i) create table ii) alter table iii) drop table iv) insert v) delete vi) update **(Refer Q.7 of Chapter - 3)** **[6]**

c) *Write and explain SQL function and procedures with sample example.* **(Refer Q.25 of Chapter - 3)** **[6]**

Q.3 **a)** *Explain with example materialized evaluation and pipelining.* **(Refer Q.30 of Chapter - 4)** **[6]**

b) *Consider following relational table. Find nontrivial and trivial functional dependency.* **[5]**

A	B	C
a_1	b_1	c_1
a_1	b_1	c_2
a_2	b_1	c_1
a_2	b_1	c_3

(Refer Q.16 of Chapter - 4)

c) *List the desirable properties of decomposition. Explain lossless join with example.* **(Refer Q.13 of Chapter - 4)** **[6]**

OR

Q.4 **a)** *Consider the following book relation.*

Book (Book_id, Title, Author, Publisher, Year, Price)

Write relational algebra expression for the following.

i) Display all book title with authors and price.

ii) Display the titles of book having price greater than 300.

iii) Display books publish in year 2000.

iv) Display all books published by 'PHP' with price greater than 300.

(Refer Q.24 of Chapter - 3) **[5]**

b) *What are the measure of query cost ?* **(Refer Q.28 of Chapter - 4)** **[7]**

c) *Define query processing. What are the steps involved in query processing ?* **(Refer Q.27 of Chapter - 4)** **[5]**

Q.5 **a)** *What is a deadlock ? Explain deadlock recovery techniques.* **(Refer Q.20 of Chapter-5)** **[6]**

b) *If we are to ensure atomicity, all the sites in which a transaction T executed must agree on the final outcome of the execution T*

must either commit at all sites, or it must abort at all sites. Describe the two phase commit protocol used to ensure this property in detail. **(Refer Q.13 of Chapter - 6)** **[8]**

c) *How does the granularity of data items affect the performance of concurrency control ? What factors affect the selection of granularity size of data items ?* **(Refer Q.25 of Chapter - 5)** **[4]**

OR

Q.6 **a)** *Explain deadlock prevention and recovery.* **(Refer Q.20 of Chapter - 5)** **[8]**

b) *Illustrate difference between conflict serializable schedule and view serializable schedule by an appropriate example.* **(Refer Q.12 of Chapter - 5)** **[6]**

c) *What are the types of errors that may cause a transaction to fail ?* **(Refer Q.4 of Chapter - 5)** **[4]**

Q.7 **a)** *Explain 2-tier and 3-tier architecture with diagram for online banking database system.* **(Refer Q.3 of Chapter - 6)** **[6]**

b) *Explain any two parallel database system architecture in detail.* **(Refer Q.4 of Chapter - 6)** **[6]**

c) *Enlist the advantages and disadvantages of replication.* **(Refer Q.11 of Chapter - 6)** **[5]**

OR

Q.8 **a)** *What are different data fragmentation techniques in distributed databases ?* **(Refer Q.12 of Chapter - 6)** **[6]**

b) *Write a short note on centralized and distributed database systems.* **(Refer Q.7 of Chapter - 6)** **[6]**

c) *Explain need of partitioning techniques used in I/O parallelism. Explain techniques in detail.* **(Refer Q.5 of Chapter - 6)** **[5]**

December - 2022 [END SEM] [5925] - 271 | **Solved Paper**

Course 2019

Time : $2\frac{1}{2}$ Hours] **[Maximum Marks : 70**

Q.1 a) *What are different types of joins in SQL? Explain with suitable example.* **(Refer Q.16 of Chapter - 3)** **[6]**

b) *Consider the following relations. It defines the schema of the database application for a bank. It manages the branches and customers of the bank. Customers take loans (borrow money) or open accounts (deposit money) at one or more branches.*

Branch (B_No, B_name, B_city, asset), Customer (C_No,C_Name, C_citystreet)

Loan (Loan_no, B_name, amount), Account (Acc_No, B_name, Balance)

Borrower (C_No, Loan_No), Depositor (C_No, Acc_No)

Answer the following queries in SQL :

1) Find the names and address of customers who have a loan.

2) Find the total amount of balance of all the accounts.

3) List all the customers who are borrowers.

4) Find all the accounts of "shivaji nagar" branch of Pune city. **[8]**

Ans. :

```
(1) SELECT C_Name, C_citystreet
FROM Customer
JOIN Borrower ON Customer.C_No = Borrower.C_no;
(2)SELECT SUM(Balance)
FROM Account;
(3) SELECT *
FROM Customer
JOIN Borrower ON Customer.C_No = Borrower.C_no;
(4) SELECT Acc_No
FROM Branch
JOIN Account ON Branch.B_name = Account.B_name
WHERE B_city = 'Pune' AND B_name = 'shivaji nagar';
```

c) *What is trigger ? State and explain two categories of triggers.* **[4]**

Ans. : Trigger : Refer Q.28 of chapter 3

Types of triggers

1) Row level trigger :

- Row level trigger is executed when each row of the table is inserted/ updated/deleted.
- If it is a row level trigger, then we have to explicitly specify it while creating the trigger.
- Also, we have to specify the WHEN (condition) in the trigger.

2) Statement level trigger :

- This trigger will be executed only once for DML statement.
- This DML statement may insert / delete/ update one row or multiple rows or whole table.
- Irrespective of number of rows, this trigger will be fired for the statement.
- If we have not specified the type of trigger while creating, by default it would be a statement level trigger.

OR

Q.2 a) *Explain with suitable example SQL aggregate functions.* **(Refer Q.14 of Chapter - 3)** **[6]**

b) *Consider the following database.*

Doctor (Doctor_no, Doctor_name, Address, City).

Hospital (Hospital_no. Name. Street, City).

Doc_Hosp (Doctor_no, Hospital_no, Date).

Construct the following queries in SQL.

1) Find out all doctors who have visited to hospital in same city in which they live.

2) Find to which hospital "Dr. Joshi" has visited.

3) Count no. of doctors visited to "Shree Clinic" on 1st March 2014. **[6]**

Ans. :

(1)

```
SELECT DISTINCT Doctor_no, Doctor_name
FROM Doctor
JOIN Doc_Hosp ON Doctor.Doctor_no = Doc_Hosp.Doctor_no
JOIN Hospital ON Doc_Hosp.Hospital_no = Hospital.Hospital_no
WHERE Doctor.City = Hospital.City;
```

(2)

```
SELECT Hospital.Name
FROM Doctor
JOIN Doc_Hosp ON Doctor.Doctor_no = Doc_Hosp.Doctor_no
JOIN Hospital ON Doc_Hosp.Hospital_no = Hospital.Hospital_no
WHERE Doctor.Doctor_name = "Dr. Joshi";
```

(3)

```
SELECT COUNT(DISTINCT Doctor.Docto_No)
FROM Doctor
JOIN Doc_Hosp ON Doctor.Doctor_no = Doc_Hosp.Doctor_no
JOIN Hospital ON Doc_Hosp.Hospital_no = Hospital.Hospital_no
WHERE Hospital.Name = 'Shree Clinic' AND Doc_Hosp.Date = '1st March 2014';
```

c) *What is cursor ? State and explain two categories of cursors and their syntax.* **(Refer Q.26 of Chapter - 3)** [6]

Q.3 a) *Define database normalization. Explain any two normal form with the suitable example.* **(Refer Q.20 of Chapter - 4)** [8]

b) *Why is query optimization important for databases ?* **(Refer Q.32 of Chapter - 4)** [5]

c) *Explain role of "Selection" operation in query processing.* **(Refer Q.34 of Chapter - 4)** [4]

OR

Q.4 a) *State and explain Armstrong's axioms and its properties.* **(Refer Q.5 of Chapter - 4)** [6]

b) *Define Boyce Codd normal form. How does it differ from 3NF ? Why is considered a stronger form of 3NF.* **(Refer Q.25 and Q.26 of Chapter - 4)** [6]

c) *What is query processing ? Explain query processing steps with neat sketch.* **(Refer Q.28 of Chapter - 4)** [5]

Q.5 a) *What is transaction ? Explain ACID properties of transaction.* **(Refer Q.3 of Chapter - 5)** [6]

b) *What is deadlock ? Explain how deadlock detection and prevention is done.* **(Refer Q.20 of Chapter - 5)** [8]

c) *What is the need of two phase locking protocol ? Explain.* **(Refer Q.18 of Chapter - 5)** [4]

OR

Q.6 a) *What is serializable schedule ? Explain with suitable example the types of serializable schedules.* **(Refer Q.6 of Chapter - 5)** [6]

b) *What is concurrency control ? Explain time stamp based concurrency control.* **(Refer Q.15 and Q.22 of Chapter - 5)** [8]

c) *Write short note on : Shadow paging.* **(Refer Q.27 of Chapter - 5)** [4]

Q.7 a) *Differentiate between centralized and client server architecture.* **(Refer Q.1 and Q.2 of Chapter - 6)** [6]

b) *State and explain key elements of parallel database.* **(Refer Q.4 of Chapter - 6)** [6]

c) *Explain distributed database architecture with neat sketch.* **(Refer Q.6 of Chapter - 6)** [5]

OR

Q.8 a) *Explain the concept of speed up and scale up in case of parallel databases.* [8]

Ans. : When a fixed size problem is given to a system which is N-times larger speed-up is measured by :

Speed-up = Small system elapsed time / Large system elapsed time

The speed-up is linear of equation equals N. (Refer Fig. 1 and Fig. 2)

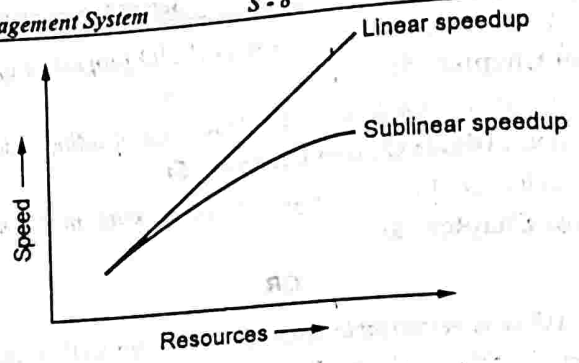


Fig. 1 Performace of parallel system

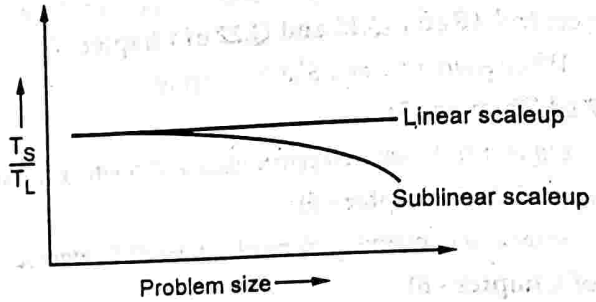


Fig. 2 Performance of parallel system

If we increase the size of both problem to be solved and the system used for solving the problem is N-times larger than the problem size then scale-up is measured by :

Scale-up = Small system small problem elapsed time / Large system large problem.

b) *Explain cloud database in detail. Also explain architecture along with components.* **[9]**

Ans. :

- The cloud computing architecture is divided into two parts - front end and back end.
- **Front end :** The front end consists of client part of the cloud computing architecture. It mainly consists of user interfaces and applications that are required to access the cloud platform.

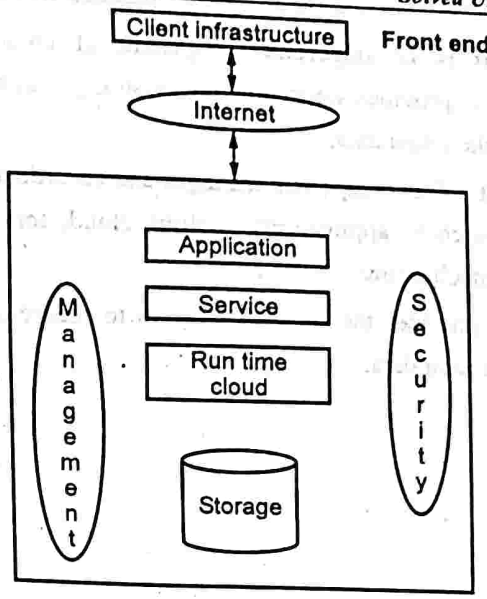


Fig. 3 Cloud computing architecture

- **Back End :** The back end consists of resources that are required for cloud computing services. It consists of data storage, security mechanism, applications, services and management mechanism.
- **Components of cloud computing architecture :**
  - **(1) Client infrastructure :** The client infrastructure is a part of front end. It consists of Graphical User Interface (GUI). This GUI is used by the client to interact with the cloud computing platform.
  - **(2) Application :** The applications are the software that is used when client accesses the cloud platform.
  - **(3) Services :** There are three cloud computing services :
    a. Software as a service(SaaS)
    b. Platform as a service(PaaS)
    c. Infrastructure as a service(IaaS)
  - **(4) Runtime cloud :** It offers the services for supporting the execution and runtime environment.

(5) **Storage :** It is an important component of cloud computing architecture. It provides large amount of storage capacity to store and manage the cloud data.

(6) **Management :** This component manages and co-ordinates the other components such as applications, runtime cloud, services, storage and security mechanisms.

(7) **Security :** It provides the security services to secure cloud system resources, files and data.

END...