# ASSIGNMENT - 1

NAME : T. Bindu

VTU NO : 30569

COURSE CODE : 10211CA207

COURSE NAME : Database Management System.

xplain in detail about DBatabase Architecture with neat diagram.

A: Database Architecture:

The Database system architecture can be divided into three main parts: Users, Query Processor, and storage Manager, with the disk storage at bottom.

1. Users:

Different types of users interact with the database:

- Naive Users (tellers, agents, web users) → use application interfaces.
- Application Programmers → write application programs.
- Sophisticated Users (analysts) → use query tools.
- Database Administrators (DBAs) → use administration tools.

2. Query Processor.

This is responsible for interpreting and executing queries.

- Application Program Object Code: Generated by compiler and linker.
- DML Queries: Data Manipulation Language (e.g., SELECT, INSERT, UPDATE, DELETE).
- DDL Interpreter: Interprets schema definitions (tables, Constraints).
- DML Compiler and Organizer: checks syntax and optimizes queries.

Query Evaluation Engine: Executes optimized queries.

3. Storage Manager.

The storage manager controls how data is stored and retrieved.

- Buffer Manager: Minimizes disk I/O by storing frequently accessed data in memory.
- File Manager: Manages allocation of space and file structures.
- Authorization and Integrity Manager: Ensures data security and integrity constraints.
- Transaction Manager: Ensures consistency, concurrency ~~and~~ control, and recovery.

4. Disk storage.

This is the physical layer where actual data resides.

- Data: Tables and records.
- Indices: Used for fast searching.
- Data Dictionary: Stores metadata. (information about tables, schemas, users).
- Statistical Data: Used for query optimization.


naive users (tellers, agents, web users)
application programmers
sophisticated users (analysts)
database administrators
use
write
use
use
application interfaces
application programs
query tools
administration tools
compiler and linker
DML queries
DDL interpreter
application program object code
DML compiler and organizer
query evaluation engine
query processor
buffer manager
file manager
authorization and integrity manager
transaction manager
storage manager
disk storage
Indices
data dictionary
data
statistical data

Explain in detail about nested queries and joins with suitable example.

- A nested query (or subquery) is a query inside another query.
- The inner query runs first and its result is used by the outer query.
- They are usually written in the WHERE, HAVING, or FROM clause.

Example Database:

| student ID | Name | Age | Dept ID | Dept Name | phone no |
|---|---|---|---|---|---|
| 1. | Ravi | 20 | 101 | CSE | 9876543210 |
| 2. | Sita | 21 | 102 | ECE | 9876765454 |
| 3. | Arjun | 22 | 103 | Mechanical | 7676768291 |
| 4. | Meena | 23 | 101 | civil. | 9182767676. |

Example 1: Simple Nested Query.

find the students who belong to CSE department.

```
SELECT Name
FROM Students
WHERE Dept ID = (
        SELECT Dept ID
        FROM Departments.
        WHERE Deptname = 'CSE'
    );
```

planation:

- Inner query finds Dept ID of CSE → 101.
- Outer query selects all students with Dept ID 101.

Results: Ravi, Arjun.

Example: 2 Nested query with IN.

find students who belong to CSE or ECE.

```
SELECT Name.
FROM Students
WHERE DeptID IN (
    SELECT Dept ID
    FROM Departments
    WHERE DeptName IN ('CSE', 'ECE')
);
```

## Joins:

- Join operations take two relations and return as a result another relation.
- A join is used to combine data from two or more tables based on a related column.

Types of JOINS:

a) INNER JOIN

- Returns only the matching rows from both tables

```
SELECT Students.Name,
Departments. Dept Name,
FROM Students
```

INNER JOIN Departments

ON students. Dept ID = Departments. Dept ID;

Result :

| | |
|---|---|
| Ravi | CSE |
| Sita | ECE |
| Arjun | CSE |
| Meena | Mechanical |

b) LEFT JOIN (OUTER JOIN)

- Returns all rows from the left table (students) and matched rows from Departments,
- if no match → NULL.

```
SELECT students. Name,
Departments. Dept Name.
FROM students.
LEFT JOIN Departments
ON students. Dept ID = Departments. Dept. ID;
```

c) Right JOIN

- opposite of LEFT JOIN.
- Returns all rows from Departments, and matched students
- If no match, NULLS are shown for left table columns.

SELECT Students. Name,
Departments. Dept Name,
FROM Students.
RIGHT JOIN Departments.
ON Students. Dept. ID = Departments. Dept ID;

If a department has no students, it still appears with NULL for Name.

d) FULL OUTER JOIN

- Returns all rows from both tables, matching rows where available, and filling NULLs where there's no match.

Basic syntax:

SELECT columns
FROM table 1
FULL OUTER JOIN table 2.
ON table 1. common_column = table 2. common_column;

- Example: Display all instructors and all courses, even if there's no mataching entry in either table.

Left join:
SELECT instructor. ID, name, course_id
FROM instructor
LEFT JOIN (teaches ON instructor_ID = teaches. ID
UNION:
SELECT instructor. ID, name, course_id
FROM instructor.
RIGHT JOIN teaches ON instructor. ID = teaches. ID;

| | name | course_id. |
|---|---|---|
| 10101 | Srinivasan | CS_101 |
| 12121 | Wu | FIN-201 |
| 15151 | Mozart | NULL |
| 76766 | NULL | BIO-101. |

## e) EQui JOIN :

> A type of INNER JOIN that uses an equality (=) operator to match rows.

Basic syntax :

```
SELECT Columns
FROM table 1, table 2
WHERE table 1. Common_column = table 2. Common_column;
```

Ex :- Find instructor-course associations using equality condition.

```
SELECT instructor.ID, name, course_id
FROM instructor, teaches.
WHERE instructor.ID = teaches.ID;
```

| ID | name | Course_id. |
|---|---|---|
| 10101 | Srinivasan. | CS-101. |
| 12121 | Wu | FIN-201 |

## CROSS JOIN.

Returns the Cartesian product of two tables - every row from the first tables joined with every row from the second table.

Basic syntax:

```
SELECT columns.
FROM table1
CROSS JOIN table2;
```

## 9) NATURAL JOIN.

A type of join that automatically joins tables using columns with the same name and Compatible data types

Basic syntax:

```
SELECT *
FROM table1
NATURAL JOIN table2;
```

# ASSIGNMENT - III

NAME : T. Bindu.

VTUNO : 30569

COURSE NAME : Database Management System

COURSE CODE : 10211CA207.

5

## Normalization and its various types of Normalization.

Normalization is a process of organizing the data in database to avoid data redundancy and improve data integrity.

It divides large tables into smaller related tables and links them using foreign keys.

Objectives:

1. To eliminate data redundacy
2. To avoid update, insert, and delete anomalies.
3. To ensure data consistency and integrity.
4. To make database structure simple and efficient.

### Types of Normalization

1. First Normal Form (1NF):

Rule:

- Each column should contain atomic (indivisible) values.
- No repeating groups or arrays are allowed.

Ex: (Before 1NF).

| StudentID | Name | Subjects. |
|---|---|---|
| 1 | Ravi | Math, Science. |

(After 1NF):

| StudentID | Name. | Subject. |
|---|---|---|
| 1 | Ravi | Math |
| 1 | Ravi | Science. |

## Second Normal Form (2NF):

Rule:

- Table must be in 1NF.
- No partial dependency – i.e., a non-key attribute should not depend on part of a composite key.

Ex: (Before 2NF):

| studenID | CourseID | StudentName | CourseName. |
|---|---|---|---|
| 1 | C1 | Ravi | DBMS. |

Here,

- StudentID + CourseID = Primary Key.
- StudentName depends only on studentID (partial dependency)

After (2NF): student Table:

| StudentID | StudentName. |
|---|---|
| 1 | Ravi. |

course Table:

| Course ID | CourseName. |
|---|---|
| C1 | DBMS. |

Enrollment Table:

| StudentID | CourseID. |
|---|---|
| 1 | C1. |

### 3) Third Normal Form (3NF):

Rule:

- Table must be in 2NF
- No transitive dependency (non-key attibut should not depent on another non-key attribute).

ample (Befone 3NF):

| StudentID | StudentName. | DeptID | DeptName |
|---|---|---|---|
| 1 | Ravi | D1 | Maths. |

Here, DeptName depends on DeptID, not directly on studentID.

After 3NF:

Student Table.

| StudentID | StudentName. | DeptID |
|---|---|---|
| 1 | Ravi | D1 |

Department Table:

| DeptID | DeptName. |
|---|---|
| D1 | Maths. |

4) Boyce-codd Normal Format (BCNF):

Rule:

- Table must be in 3NF.
- For every functional dependency ($X \to Y$). X should be a super key

Example:

| Course. | Instructor | Room |
|---|---|---|
| DBMS | Raj | R1. |
| DBMS | Ravi | R1 |

5) Fourth Normal Form (4NF):

Rule:

- Tabb must be in BCNF.
- There should be no multi-valued dependencies.

Ex:

| Student | Hobby | Language |
|---|---|---|
| Ravi | Cricket | English |
| Ravi | Music. | Hindi. |

Here, "Hobby" and "Language" are independent multi-valued facts.

4NF:

udent_Hobby:

| Student | Hobby. |
|---|---|
| Ravi | Cricket. |
| Ravi | Music. |

Student_language:

| Student | Language. |
|---|---|
| Ravi | English |
| Ravi | Hindi |

6) Fifth Normal Format (5NF):

Rule:

- Table must be in 4NF.
- Should not have join dependency.

Ex:

Relation P = {subject, lecture, Semester}

| Subject | lecturer | Semester. |
|---|---|---|
| computer | Anshika | Semester 1 |
| Computer | John | Semester 1 |
| Math. | John | Semester 1 |
| Math. | John | Semester 1. |
| chemistry | Praveen | Semester 1. |

P1 = {semester, Subject}

P2 = {subject, lecturer}

P3 = {semester, lecturer}

All three relations are now is 5NF

# ASSIGNMENT - IV

NAME : T. Bindu

UTU NO : 30569

COURSE NAME : Database Management System

COURSE CODE : 10211CA207.

3

Explain about deadlock and its handling in dbms.

A deadlock is a condition in a multiprogramming system where two or more processes are waiting indefinitely for resources held by each other, causing all of them to remain blocked forever.

Ex:

- Process P1 holds Resource R1 and waits for R2.
- Process P2 holds Resource R2 and waits for R1

→ Both process wait forever - deadlock occurs.

Necessary conditions for deadlock:

A deadlock can occur only if all these four conditions hold simultaneously.

1. Mutual Exclusion: only one process can use a resource at a time.
2. Hold and wait: A process holding a resource is waiting for others
3. No preemption: Resources cannot be forcibly taken from a process.
4. Circular wait: A circular chain processes exists, each waiting for a resource held by the next process

Deadlock Handling Methods:

1. Deadlock prevention: Design the system in such a way that at least one of the four necessary conditions never occurs.

Ex: Don't allow hold-and-wait.

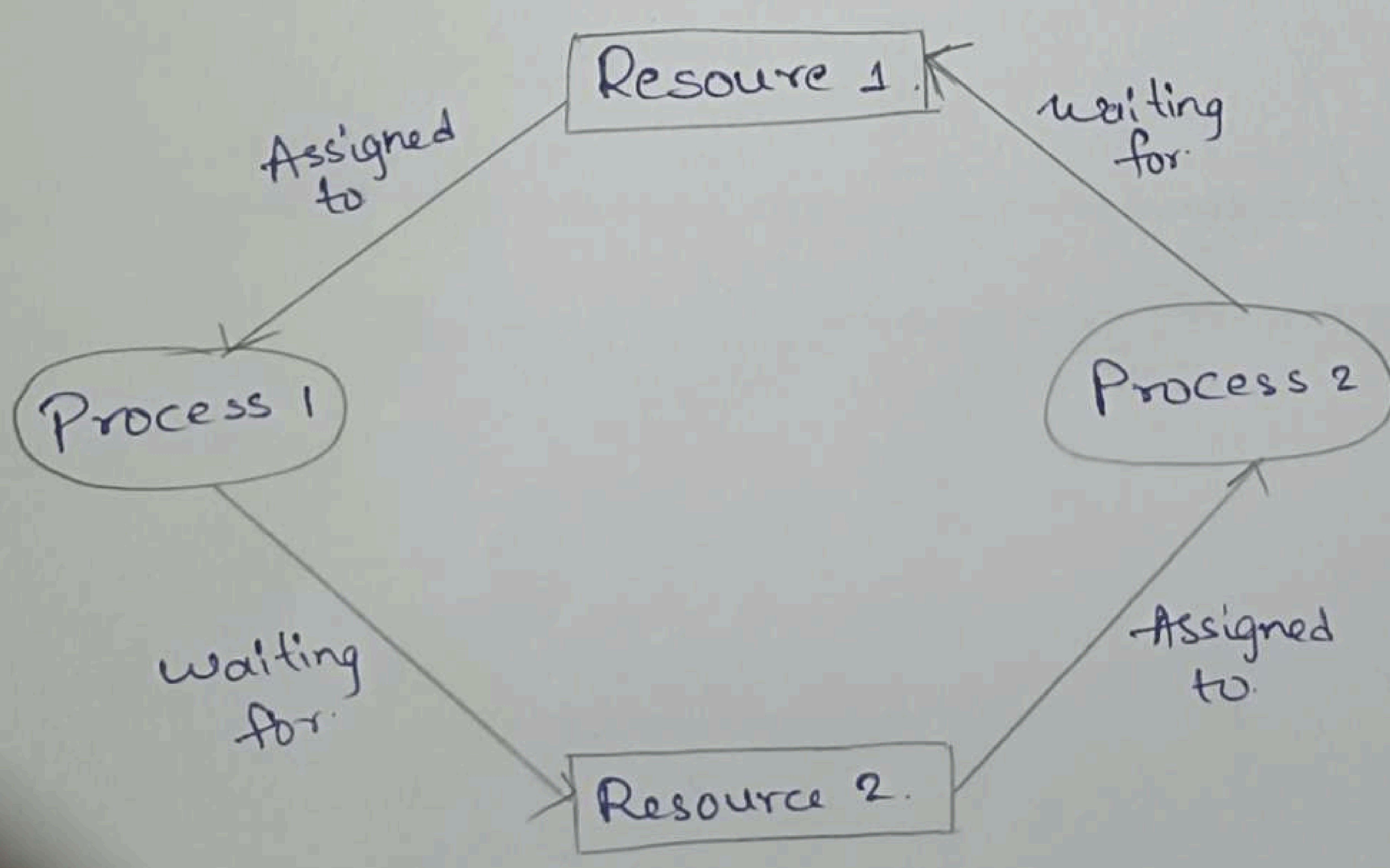
2. Deadlock Avoidance: Dynamically check resource requests to ensure the system never enters on unsafe state.

Ex:- Banker's Algorithm.

3. Deadlock Detection and Recovery: Allow deadlock to occur, detect it using an algorithm, and the recover. (by terminating or rolling back processes)

4. Deadlock Ignorance: The system assumes that deadlock never occurs. used in most OS (like windows, linux) because deadlocks are rare.

Deadlock handling is essential in operating system to ensure smooth execution of processes and efficient resource utilization. Different techniques are used depending on system requirements and complexity.



# ASSIGNMENT - V

NAME : T. Bindu

UTU NO : 30569

COURSE NAME : Database Management System

COURSE CODE : 10211CA207.

5

RAID storage and its types.

RAID stands for Redundant Array of Independent Disks. It is a data storage technology that combines multiple physical hard drives into a single logical unit to improve performance, fault tolerance, and data reliability.

RAID is mainly used in servers and high-performance systems where data audio availability and speed are important.

Objectives of RAID:

- To increase storage capacity
- To improve read/write performance.
- To provide data redundancy (backup in case of disk failure).
- To enhance system reliability.

Working principle:-

RAID uses a technique called data striping (dividing data into blocks and a storing them accross multiple disks) and parity (extra information used for recovery if one disk fails).

Types of RAID levels:

RAID 0 (striping): Data is divided into blocks and stored across multiple disks.

RAID 1 (Mirroring): same data is copied (mirrored) on two or more disks.

RAID 2 (Bit-level striping): Data is striped at bit level. with error Correction Code (ECC).

RAID 3 (Byte-level stripping with parity): Data is striped a byte level and a separate disk stores parity bits.

RAID 4 (Block-level striping with ~~Distributed~~ parity):- Parity Information is stored on a dedicated disk.

RAID 5 (Block-level striping with Distributed parity): Parity is distributed accross all disks.

RAID 6 (Double Distributed parity): similar to RAID 5 but uses two parity blocks for extra protection.

RAID 10 (Combination of RAID 1 + RAID 0): Combines mirroring and striping.

RAID storage technology provides a balance between performance and reliability.

The selection of RAID level depends on system needs, whether speed, cost or data safety in the priority.

RAID·1.