Assignment - 1

Subject
Sub code:
Name: Chandrajit K
VTU: 08548

What is database?

A database is an Organized collection of data that can be easily accessed, managed and updated. It stores information in a structured form so that it can be efficiently retrieved and manipulated.

example:-

Bank Database- stores customer details, accounts, transactions.

Student Database- stores names, roll numbers, marks, attendance.

What is Database Architecture?

Database Architecture defines how the database system is structured and how different components interact with each other. It shows the relationship between users, applications and the database management system (DBMS).

The main goal:

- Provide efficient data storage
- Enable fast access
- Ensure security & integrity

Types of Database Architecture

There are mainly three levels of architecture as per ANSI/SPARC model:

1. Internal Level (Physical level)
   - Lowest level of abstraction.
   - Describe how data is physically stored in memory (hard disk, indexes & data blocks).
   - Deals with storage, file structure, compression, and encryption.

   example:- Database schema → Student (Name, Roll_No, class, Marks).

2. Conceptual Level (Logical level)
   - Middle level of abstraction.
   - Describes what data is stored and relationships among them.
   - Independent of hardware / software.

example :- Database schema → Student (Roll_No, Class, Marks).

3. External level (View level)

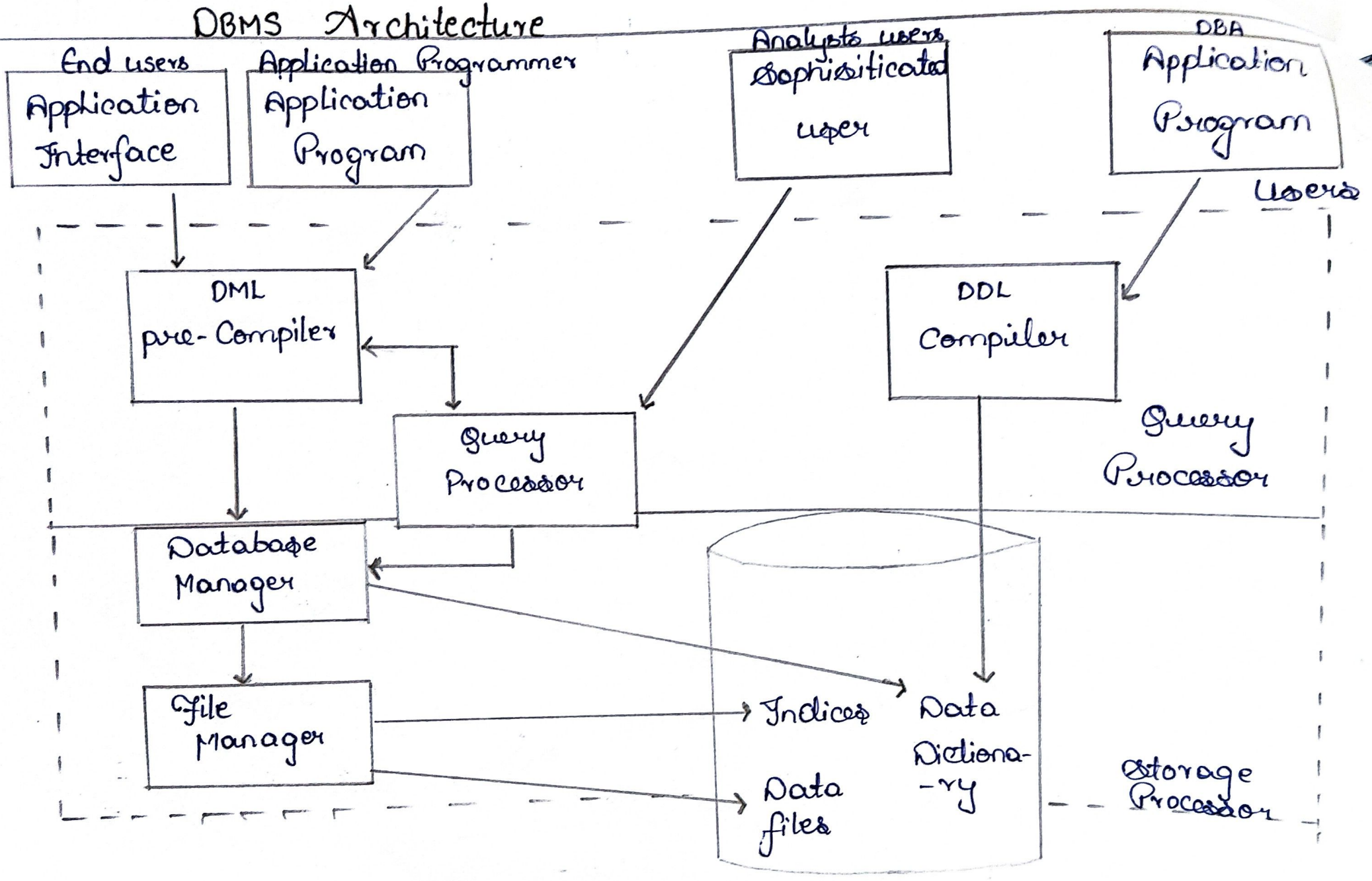
- Highest level of abstraction.
- Describe how data is viewed by individual users.
- Different users may have different views depending on their needs.

example :-

- Teacher sees → Student name, Rollno, Marks.
- Librarian sees → Student name, Rollno, Books issued.

D

DBMS Architecture



Summary

Database = Organized data storage.

Architecture = How DBMS organizes data at 3 levels.

3 levels = Internal (storage), Conceptual (schema), External (views).

This seperation provides Data Abstraction and Data Independence.

Assignment - 2

Subject
Sub code:
Name Chandrajit.K
VTU no: 08548

What is Nested Query?

A nested query, also called a subquery, is a SQL query inside another SQL query. The inner query is executed first, and its result is used by the outer query.

Types of Nested Queries

1. Single-row subquery
2. multi-row subquery
3. Correleated subquery

examples:

ID	Name	Salary	Dept_ID
1	Alice	50000	10
2	Bob	60000	20
3	Charlie	55000	10
4	Diana	70000	30

1 Single-row Subquery

find employees who earn more then the avg salary.

```
SELECT * FROM Employees
WHERE Salary > ( SELECT AVG(Salary) FROM
                 Employees);
```

The subquery: SELECT AVG(Salary) FROM EMPLOYEES returns a single value (the avg salary).

Ex-2 Multi-row Subquery

find names of employees who work in departments with ID 10 or 20.

```
SELECT Name FROM Employees
WHERE Dept_ID IN ( SELECT Dept_ID FROM
 Departments WHERE Dept_ID IN (10,20));
```

The inner query returns multiple values (10 and 20), and the outer query matches Dept_IDs.

Correlated Subquery

find employees who earn more than the avg salary of their department.

```
SELECT E1.Name, e1.Salary, E1.Dept_ID
FROM Employees E1
WHERE e1.Salary > (
    SELECT AVG(E2.Salary)
    FROM Employees e2
    WHERE E2.Dept_ID = E1.Dept_ID
);
```

- The subquery refers to the outer query's row (E1.Dept_ID).

JOINS

What is Join?

A JOIN combines row from two or more tables based on a related column between them.

Types of Joins

1. Inner Join
2. LEFT (Outer) Join
3. RIGHT (Outer) Join
4. Full outer Join
5. Self Join
6. Cross Join

Example 1: INNER JOIN

List all employees with their department names.

```
SELECT E.Name , E. Salary, D. Dept_Name
FROM Employees E
INNER JOIN Departments D
ON E. Dept_ID = D. Dept_ID;
```

. Inner Join returns only matching records in both tables.

Example 2: LEFT JOIN

List all employees and their departments (even if the department is not found).

```
SELECT E.Name, D. Dept_Name
FROM Employees E
LEFT JOIN Departments D
ON E. Dept_ID = D. Dept_ID;
```

. Left JOIN returns all records from the left table (employees) and matching ones from the right (Departments).

. If there is No match Dept_Name will be NULL.

Example 3: Full outer join (not supported in MySQL but possible in PostgreSQL / SQL Server)

List all employees and departments, showing all even if no match exists.

```
SELECT E.Name, D.Dept_Name
FROM Employees E
FULL OUTER JOIN Departments D
ON E.Dept_ID = D.Dept_ID;
```

Summary

Feature	Nested Query	JOIN
Executes	Inner query runs first	Tables are joined in one logical query.
Use-Case	filtering based on aggregated results	Combining related data from multiple tables.
Performance	Can be slower	often optimized by database engine.
Example	WHERE Salary > (SELECT AVG(...))	FROM A JOIN B ON A.id = B.id

Assignment

Name - Chandrajit K

Subcode - 10211CA207

VTU - 08548

Normalization

Normalization is a systematic process in database design that organizes data to reduce redundancy and improve data integrity by dividing large tables into smaller, related tables according to rules called normal forms.

Types of Normalization

Normalization involves structuring a relational database so that its table follow specific rules to ensure logical data grouping minimal duplication, and easier maintenance.

1NF - Eliminates repeating groups and ensures that each field contains atomic values.

2NF - Removes partial dependency of data on a composite primary key.

3NF - Removes transitive dependency from non-prime attributes to candidate keys.

BCNF - Handles anomalies not addressed by 3NF, applying stricter candidate key rules.

4NF- Eliminates multi-valued dependencies

5NF- Deals with Join dependency anomalies

Requirements

1NF - Every attribute value is atomic; each column has a unique name.

2NF - Removes partial dependency. Table must be in 1NF and all non-key attributes fully depend on the entire primary key.

3NF - Table must be in 2NF, no transitive dependency.

BCNF - Table is in 3NF, and every determinant is a candidate key.

4NF - Table is in BCNF and no multi-valued dependency exists.

5NF - Table is in 4NF and every join dependency is implied by candidate keys.

Benefits of Normalization

- Minimizes redundancy and data anomalies.
- Maintain data consistency and integrity
- Simplifies database maintenance and updates.

Practical usage in Databases

Applying normalization is crucial in relational databases like hospital management system or any structured tabular schema to optimize queries, prevent update/delete issues, and streamline logical design.

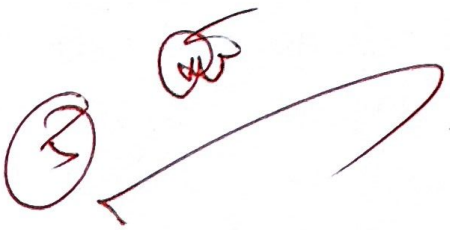
Normalization is typically performed in stages, increasing from 1NF to higher forms as needed by system requirements.

Database Management System.

Name : S.Bhaskar.

VTU NO : VTU 28066.

Branch : CSE(AI&ML)



Assignment

Name- Chandrajit.t

Subcode- 10211CA207

VTU- 28548

Deadblock and its handling

A deadblock in database system in a situation where 2 or more transaction are stuck, each waiting for the other to release resources, leading to all of them being unable to proceed.

Deadblock are a critical concern in multi-user enviroments, especially when transaction compete for locks on shared resources such as data, memory or devices.

Necessary Conditions for Deadlock

All of these conditions must be present for a deadlock to occur:-

- Mutual Exclusion - Resources cannot be shared; only one transaction at a time can use a resource.
- Hold and wall- A transaction holding at least one resources is waiting to acquire additional resources held by other transactions.

- RAID 2 (Bit - Level Striping):

Rarely used, implements bit-level striping with Hamming code for error correction.

Requires three or more disks and is complex / costly.

RAID 3 (Byte- Level striping with Dedicated Parity):

Stripes data at the byte level with a dedicated parity drive for fault tolerance.

Suitable for high - throughput sequential data, but less efficient for random access.

Raid 4 (Block - Level Striping with Dedicated Parity):

- Stripes data at the block level, storing parity on a seperate disk.
  Allows parallel read operations, tolerates one disk failure.

Raid 5 (Block - Level Striping with Distributed Parity):

- Distributes both data and parity blocks across all disks.
- Can withstand one disk failure with good read performance.

Eliminate mutual Exclusion: make resources shareable if feasible.

Common prevention schemes include:-

- Wait-Die - Older transactions wait for younger ones, while younger transaction requesting resources held by older ones are aborted and restarted.

Woundwait - Older transactions preempt younger ones, forcing younger transaction to abort if they block older ones.

Dead lock Detection & Recovery

- Detection algorithm, such as wait for graph are used to identify cycle.
- Once detected, recovery may involve aborting or rolling back one or more involved transaction to break the cycle.

Assignment

(5)

(6th)

Name - Chandrajit L

Subcode - 10211CA207

VTU - 28548

- RAID 6 (Block-level striping with Double Parity):

  Similar to RAID 5 but with two parity blocks, enabling up to 2 disk failures.

  Requires at least 4 disks.

- RAID 10 (Hybrid 1+0):

  Combines mirroring and striping for high speed and redundancy.

  Needs at least 4 disks, offering improved fault tolerance compared to RAID 1.

Raid storage and its types

RAID (Redundant Array of Independent Disks) is a storage technology used to combine multiple physical disk drives into a single logical unit to improve data redundancy, performance or both.

Types of RAID storage

The major RAID levels include.

- RAID 0 (striping):
  Data is split across 2 or more disks for high speed access.
  Provides no redundancy → data loss occurs if any disk fails.
  Minimum 2 disk required.

- RAID 1 (Mirroring):
  Data is duplicated on two or more disks, offering redundancy.
  Can sustain one disk failure without data loss.
  Reduce usable storage to 50%.

- No Preemption : Resources cannot be forcibly taken from a transaction - they must be released voluntarily.
- Circular Wait: A closed chain strategies transaction exists, with each transaction waiting for a resources held by the next in the chain.

Deadlock handling strategies

Deadlock can be entirely avoided by ensuring at least one of the four necessary conditions cannot occur:-

Eliminate hold and wait → Require transaction to request all needed resource at once, blocking them until all are available.

Eliminate Circular Wait → Impose an order for requesting resources and require all transaction to request resources in this order.

Eliminate No preemption : If a transaction holding resources requests another that cannot be immediately allocated, force it to release all held resources.