DBMS UNIT I

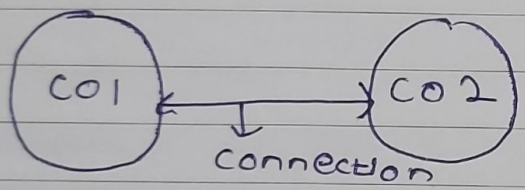
Questions:-

1. level of Abstraction
2. Draw ER Diagram
3. Keys
4. Architecture of DBMS
   1. Storage Manager
   2. Query Processor
5. Characteristics of database approach.

(*) Database Management System:-

- DBMS is collection of interrelated data of enterprises and set of Program to acess these data is called Database Management System.

*



* Database:-

- collection of data of Particular enterprise is called Database.

* Characteristics of DBMS:-

1. Data Integrity:-

- Changes made by authorized user does not result in loss of data.
- Correctness and completeness of data.

② Data security :-

- Protection of database against virus, misuse accidental and intentional loss.

③ Data independence :-

- Capacity to change data at one location without affecting data stored in other location

④ Transaction control - Rollback :-

- Changes made to database can be reverted back by using rollback command.
- In order to save changes we use commit command.

⑤

⑤ Concurrency control :-

- The data stored in database can be acessed by multiple user at same time. eg- Railway Reservation

⑥ Data recovery :-

- Database recovery is Process to restore ~~to~~ of data to original state after Failure of database.

⑦ Atomicity :-

- Ability to do certain operation.

* ~~First~~

* Database Differ from Filesystem in Following ways:-

DBMS	Filesystem
① DBMS is collection of data in which user is not required to write Procedures	① Filesystem is collection of data in which user is required to write procedure.
② Due to centralized approach sharing of data is easy	② Due to distributed approach sharing of data is not easy.
③ DBMS Provide abstraction to data some data ↑	③ Detailed view of data is shown
④ Data redundancy is not Problem loss of data ↑	④ Data redundancy is Problem
⑤ Data inconsistency does not exist	⑤ Data inconsistency exists
⑥ Structure to design is complex	⑥ Structure is easy

DBMS	File system
(7) Accessing of data is easier	(7) Accessing of data is difficult
(8) Data independence exists	(8) Data independence does not exists.
(9) ~~Data~~ 3 types of data model :- ① Hierchical ② Network ③ Relational	(9) No data model.

* levels of Abstraction :-

• Abstraction :-

- Data Abstraction means retrieving only required amount of data or information about the system and hiding background details.

Three level are there :-

① Physical level :-

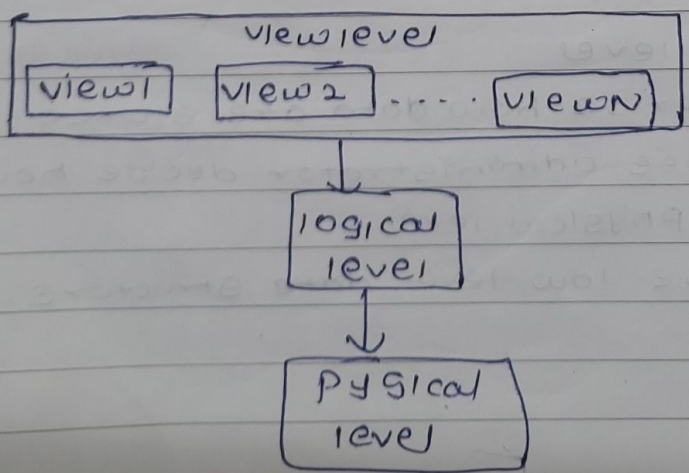
- lowest level
- It describe how data are stored
- Database administrator decide how to store data at physical level
- Describe low level data structure.

② Logical level :-

- This is next higher level, which describe what data are stored in database?
- This level also describe relationship between data.
- Database administractor use logical level of abstraction for deciding what information to keep in database.

③ View level :-

- This is highest level of Abstraction that describes only part of entire database
- This level show specific part of database
- This level is used to have interaction with system.
- It provide multiple view of some system.
- For eg :- A clerk in reservation system can see only part of database and acess passenger's required information.



DMS

(1) Architecture of DBMS
   (1) Storage Manager
   (2) Query Processor

(2) Draw a neat diagram of Database system structure and explain its component in detail

(3) ER diagram Practice

(4) Keys

(*) Architecture of DBMS :-

(1) ~~Storage Manager~~ :-

(1) Query Processor :-
- Query Processor help the database system to simplify and Facilate access to the data stored in database.
- Query Processor consist of DDL interpreter DML compiler and query evaluation engine.
- with this 3 component following Function ality is provided :-

↳ Translator :- DDL Statement → Data dictionaries

① DDL Interpreter :-

- DDL interpreter is basically a translator which interpretes the DDL statement into data dictionaries.

↳ evaluation plan

② DML Compiler :-

- DML compiler translate the ~~DDL~~ DML statement into a evaluation plan which is understandable by query evaluation engine.

↳ execute low level instruction

③ Query evaluation engine :-

- It execute low level instruction generated by DML compiler.

- When a user issue a query the parsed query is presented to query optimizer which uses information about how data is stored and produce a evaluation plan which is evaluated by query evaluation engine.

② Storage Manager :-

- The storage manager is component of database which provide interface between low level data stored in database and application program and query submitted to system.

- Storage Manager is responsible for storing, updating and retieving data from database

The Storage Manager has following component:-

↳ security

① Authorization and integrity Manager:-

- It check or validate the user who want to acess the data so that authorized user can acess the data and also test for integrity constraint.

↳ safety of data

② Transaction Manager:-

- It ensure that the data in database remain safe despite of system failure and concurrent transaction execution should be there.

③ File Manager:-

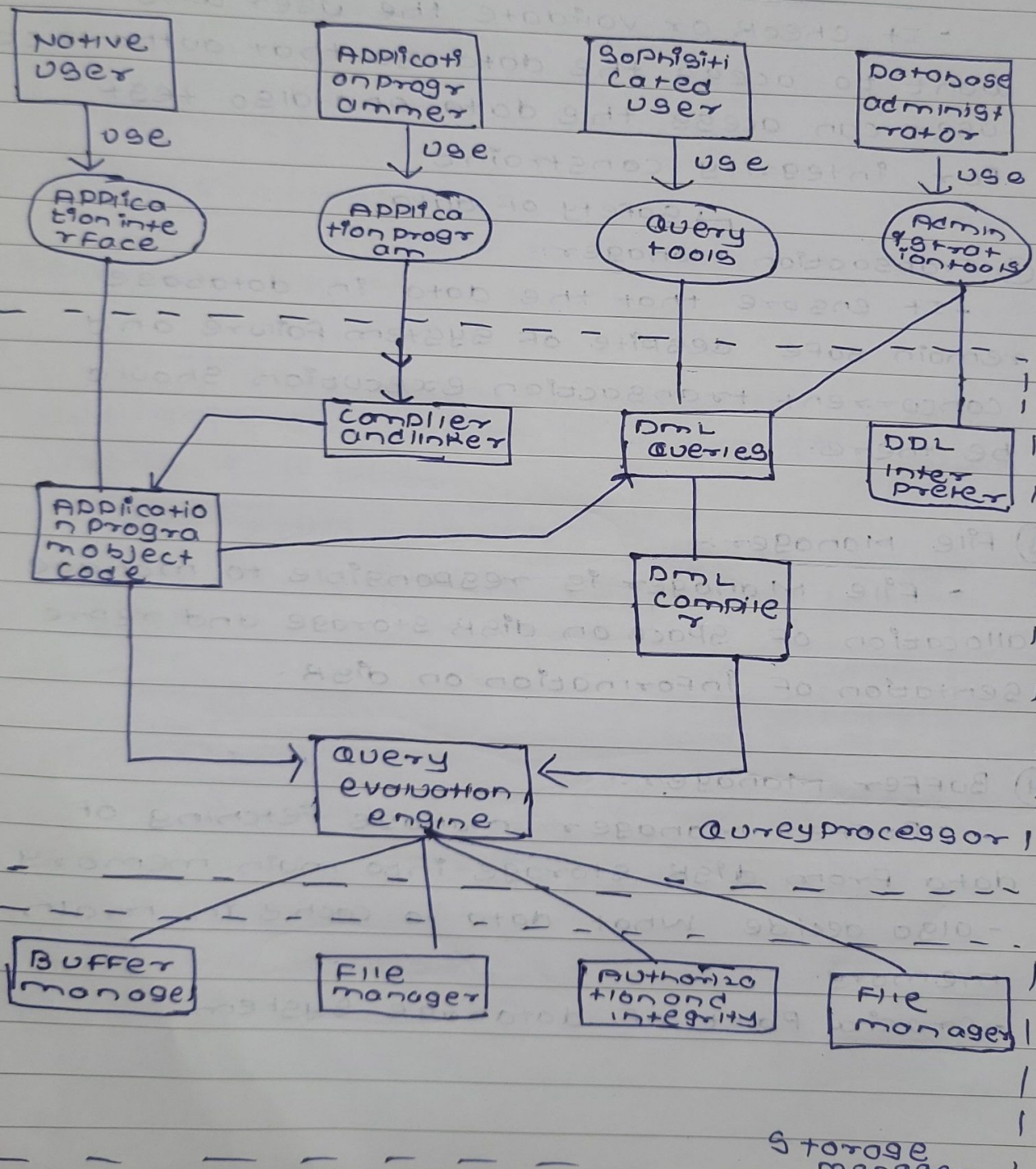
- File Manager is responsible to manage allocation of space on disk storage and repre sentation of information on disk.

④ Buffer Manager:-

- Buffer Manager manage fetching of data from disk storage into main memory
- also decide what data to cache in main memory.
- Crucial part of database system.

* Storage Manager implement several data structure such as :-

① Files - used for storing database itself
② Dictionaries - storing schema of database
③ Indices - used to have faster access to database.

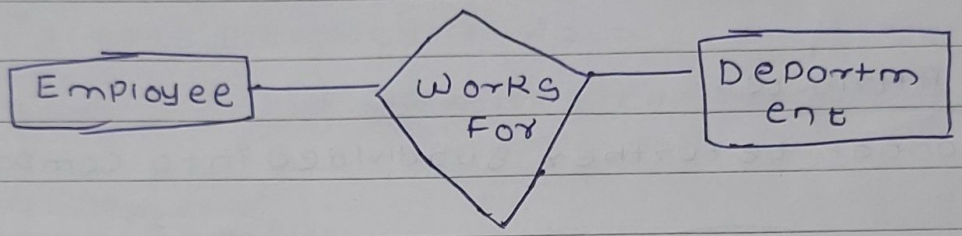


* ER Model :-

- Entity :- Weak Entity
  Strong Entity

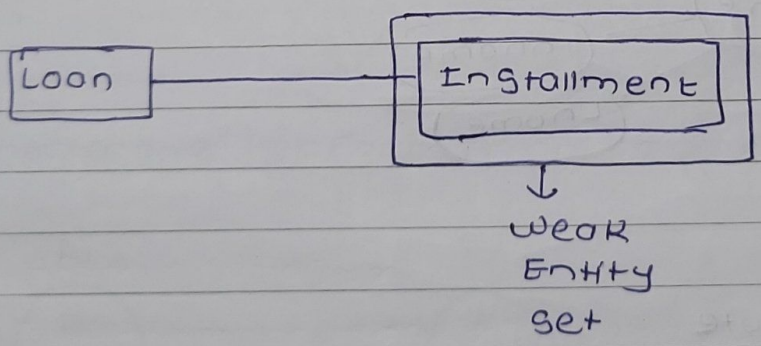
↓

id, name,
↓ ↓
34 Gondhorv → Entity
35 Mohit
↓
Entityset



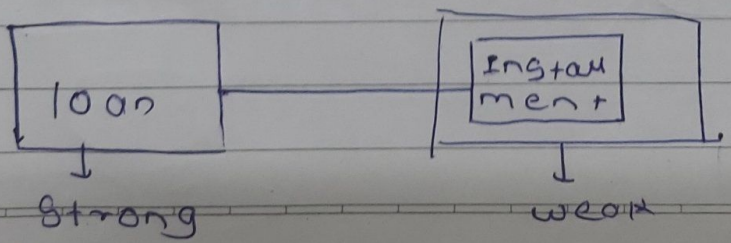
① Weak Entity Set :-

- Not have Primary Key
- Dependedent entity
- does not Key attribute of own
- Represent by double rectongle



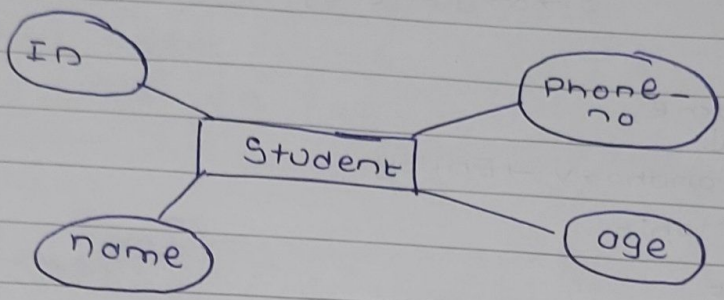
② Strong Entity Set :-

- Which has Primary Key



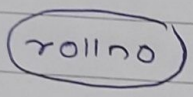
* Attribute:-

- Entity Property describe
- 1 Entity set can have different Attribute
- esclipse



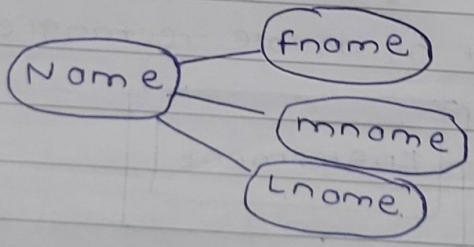
a] Simple Attribute:-

- cannot be further subdivided into component



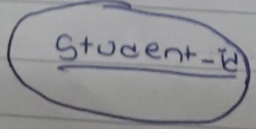
b] Composite:-

- Attribute that can split into component is called Composite



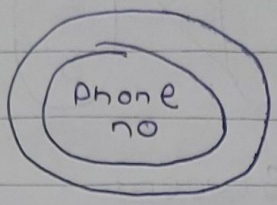
c] Key Attribute:-

- Unique attribute
- Represent Primary Key



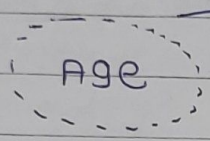
④ Multivalued Attribute:-

- An attribute which have more than one value called Multivalued attribute



⑤ Derived Attribute:-

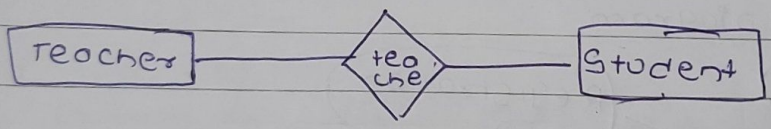
- Attribute derived from another attribute is called derived Attribute.



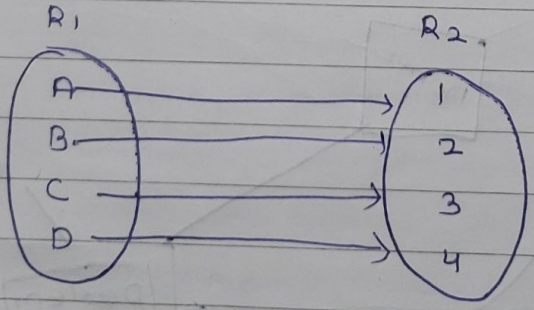
→ ~~From~~ age can derived From DOB.

* Relationship/Mapping constraint:-

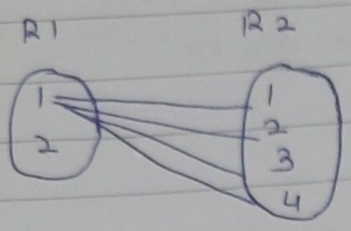
- Mapping cardinity express no of entity to which another entity can be associated via relationship set.



① One to one Relationship:-

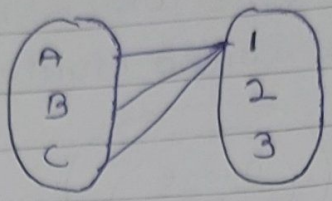


② one to many



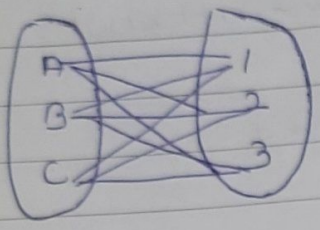
Scientist Invention

③ Many to one



Student Course

④ Many to many

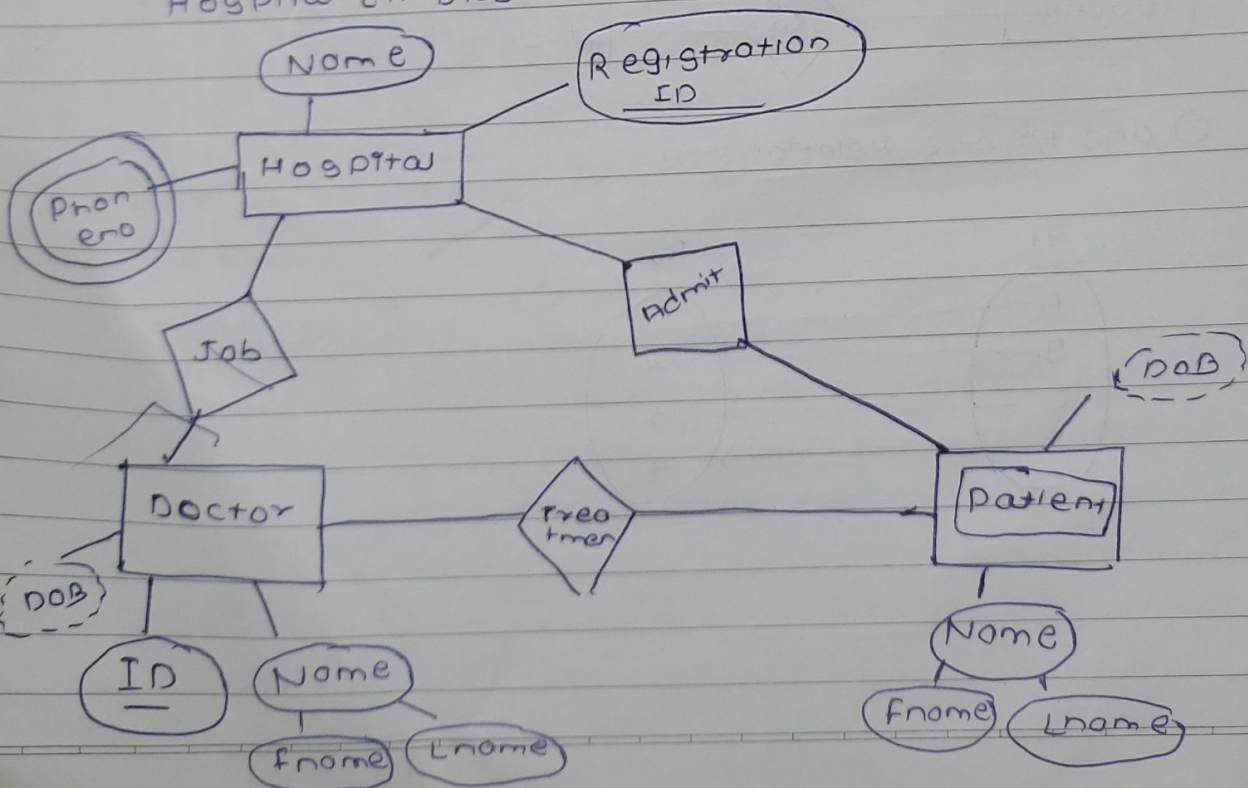


Employee Project

* Notation :-
- To define cardinality Notation is used.

* ER Diagram:-

Hospital ER Diagram



* Keys in DBMS:

- Value that can have identify the table.

① Super Key:-

- set of one or more attribute that can be taken collectively to uniquely identify a entity in entity set.

For eg:- In student table

(S_Rollno, S_name, S_Branch, S_year)

SuperKey => S1 → S_Rollno, S_Name

S2 → S_Rollno, S_Branch

S3 → S_Rollno, S_year

S4 → S_Rollno, S_Name, S_Branch

② Candidate Key:-

- minimal no of super key that identify value.

C1 → S_Rollno

C2 → S_Rollno, S_name

③ Primary Key:-

- Subset of candidate key.
- It unique key
- minimum no of candidate key

PK → Rollno

④ Composite Key :-

Primary Key consist of different attribute

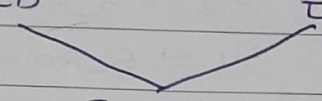
CS_Rollno, S_ID, S_name, S_Branch)

Composite Key → S_Rollno, S_ID

⑤ Foreign Key :-

- Combine two table
- If one table Attribute is Primary Key in other table then it is called Foreign Key.

eg:- Table 1 (Rollno, Name, ID) Table 2 (ID, Branch, address)



Foreign Key

What is view and how to create it ? Can you update view ? if yes, how ? if not, why not ?

→

- A view is a virtual table that represents the result of a database query. Unlike a physical table, a view does not store data itself, it dynamically generates data when queried based on the underlying tables.

- We can create a view by selecting fields from one or more tables present in the database. A view can either have all rows of a table or specific rows based on certain condition.

- Creating a View :-

syntax

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

eg.

```
CREATE VIEW high_salary_employees AS
SELECT name, salary
FROM employees
WHERE salary > 50000;
```

- Yes, we can update view but with certain limitations. An updatable view is one which allows performing a UPDATE command on itself without affecting any other table.

- ~~Only simple views~~ If the view is based on a single table and does not include complex operation like 'JOIN', 'GROUP BY', or aggregate functions (eg. 'SUM', 'COUNT') you can update it

eg.

```
UPDATE high_salary_employees
SET salary = 60000
WHERE name = 'John';
```

Define stored procedure. Explain the creating & calling stored procedure with example.

- Stored procedure is a type of subprogram in PL/SQL block. It is a group of statements that can be called by its name.

- This is a subprogram that doesn't return a value directly.

DATE / /

- creating a stored procedure:

A procedure is created with the CREATE OR REPLACE PROCEDURE statement.

Syntax:

```
CREATE PROCEDURE procedure_name(parameters)
BEGIN
    SQL statements;
END;
```

example:

```
CREATE PROCEDURE add_employee ( IN emp_nam
VARCHAR(50), IN emp_sal DECIMAL(10,2),
IN emp_dept_id INT)
BEGIN
    INSERT INTO employees (name, salary, id)
    VALUES (emp_name , emp_sal , emp_dept_id);
END;
```

- Calling a stored Procedure:

We can use the 'CALL' statement to execute a stored procedure.

Syntax:

```
CALL procedure_name(arguments);
```

eg.

```
CALL add_employee ('JohnDoe', 55000, 3);
```

Input and Output parameters:

eg.

```
CREATE PROCEDURE get_sal (IN emp_id INT,
OUT emp_sal DECIMAL(10,2))
BEGIN
    SELECT salary INTO emp_sal
    FROM employees
    WHERE id = emp_id;
END;

CALL get_sal (1, @salary);
SELECT @salary;
```

What is a trigger? How to create it? Discuss various types of trigger?

- Triggers are stored procedure, which are automatically executed or fired when some event occurs. It is stored in database and invoked repeatedly, when specific condition match. such as inserting, updating or deleting records in a table.

- bigger is created using the 'CREATE TRIGGER' statement. The bigger is associated with a tablu and is activated by specific actions (eg. 'INSERT', 'UPDATE', 'DELETE')

Syntax:

```
CREATE TRIGGER trigger_name
{BEFORE| AFTER} {INSERT| UPDATE| DELETE}
ON table_name
FOR EACH ROW
BEGIN
    SQL statements;
END;
```

eg.

```
CREATE TRIGGER log_insert
AFTER INSERT ON employees
FOR EACH ROW
BEGIN
    INSERT INTO audit(emp_id, action, time)
    VALUES (New.id, 'INSERT', NOW());
END;
```

Q.1: Consider the following schema
student details (roll no, name, deposit, date)
Write a trigger to preserve old values of student fee details before updating in table.

Creation of table:

```
CREATE TABLE std. fee_his (
    roll no INT,
    name VARCHAR(50),
    fee deposited DECIMAL(10,2),
    date DATE,
    updated on TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Trigger:

```
CREATE TRIGGER preserve old_values
BEFORE UPDATE ON student_fee_details
FOR EACH ROW
BEGIN
    INSERT INTO student_fee_his VALUES
    (OLD.rollno, OLD.name, OLD.fee_deposited,
    OLD.date, CURRENT_TIMESTAMP);
END;
```

PAGE No.
DATE / /

Que Write a PL/SQL block of code which accepts the rollno from user. The attendance of rollno entered by user will be checked in student_attendance (ROLLNO, Attendance) table and display on the screen.

```
DECLARE
    v_rollno student_attendance.rollno%TYPE;
    v_attendance student_attendance.attendance%TYPE;
BEGIN
    dbms_output.put_line('Enter Roll no :');
    v_rollno := :rollno

    SELECT attendance
    INTO v_attendance
    FROM student_attendance
    WHERE rollno = v_rollno;

    dbms_output.put_line('Attendance for roll no '||
                    v_rollno||' is : '||v_attendance
                    ||'%');

  EXCEPTION
    WHEN NO_DATA_FOUND THEN
        dbms_output.put_line('Roll number '||
        v_rollno||'not found');

END;
/
```

Write a PL/SQL code block for displaying the area of circle with radius values from 3 to 8. Create table areas (r number(2), area number (14,2));

```
CREATE TABLE areas(
    r number(2),
    area number(14,2));

BEGIN
    for radius IN 3..8 LOOP
        DECLARE
        INSERT INTO areas(r, area)
        VALUES(radius, (3.14 * radius * radius));

    END LOOP;
    COMMIT;
END;
/
```

* what are the aggregate fuction ? And list aggregate fuctions supported by SQL.

→ *Aggregate fuction :-
- Aggregate fuctions in SQL perform a calculation on a set of values and return a single value.
- They are often used with the "Group By" clause to group rows that share a property so that an aggregate value can be computed for each group

* Aggregate fuction supported by SQL :-
1. COUNT() :-
- Return the no of rows that match the specified criteria.
- Ex : select COUNT(*)
FROM employees.

2. SUM() :-
- Return sum of a numeric column.
- EX : SELECT SUM (salary) FROM employees;

3. AVG() :-
- Return the average value of a numeric column
- EX : SELECT AVG (Salary) FROM employees;

4. MIN() :-
- Return the minimum value in a set of values.
- EX : SELECT MIN (Salary) FROM employees;

5. MAX() :-
- Return the maximum value in a set of values
- EX : SELECT MAX(Salary) FROM employees;

Page No.
Date

⑥ what are different types of ~~joins~~ joins in database & Explain with example.

→ - In database the concept of "joins" is used to combine data from two or more tables based on a related column between them.
- there are several types of join each serving a different purpose.

1. INNER JOIN
- An "INNER JOIN" return records that have matching values in both tables.
- Example :
Suppose we have two tables "Employee" and Department's

- Employees Table.

EmployeeID	Name	DepartmentID
1	Alice	101
2	Bob	102
3	charlie	103
4	David	~~104~~ 101

- Department Table.

DepartmentID	Department Name
101	HR
102	IT
104	Marketing.

- Query :-

```
SELECT Employees.Name, Departments.DepartmentName
FROM Employees
INNER JOIN Departments
ON Employees.DepartmentID = Departments.
                            DepartmentsID
```

Result :-

Name	DepartmentName
Alice	HR
Bob	IT
David	HR.

- only the employees who have a matching "DepartmentID" in both table are returned.

2. LEFT JOIN (or LEFT OUTER JOIN):
- A "LEFT JOIN" return all record's from the left table and the matched records from the right table.
- If no match is found, "NULL" value are returned for columns from the right table.

- Example :-

```
SELECT Employee.Name, Department's.DepartmentName
FROM Employees.
LEFT JOIN Departments
ON Employees.DepartmentID = Departments.
                            DepartmentID.
```

- Result :-

Name	DepartmentName
Alice	HR
Bob	IT
charlie	NULL
David	HR.

Name

3. RIGHT JOIN (or RIGHT OUTER JOIN)

- A 'RIGHT JOIN' is the opposite of a 'LEFT JOIN' It returns all records from the right table and the matched records from the left table.
- If no match is found, "NULL" value are returned for colums from the left table.

- Example :-

```
SELECT Employees.Name, Departments.
DepartmentName FROM Employees
RIGHT JOIN Departments
ON Employees.DepartmentID = Departments.
                            DepartmentID
```

- Result :-

Name	Department Name
Alice	HR
Bob	IT
David	HR
NULL	Marketing.

4. FULL OUTER JOIN

- A 'FULL OUTER JOIN' returns all records when there is a match in either left or right table
- If there is no match, 'NULL' values are returned from columns where no match is found.

- Example

```
SELECT Employees.Name, Departments.
DepartmentName FROM Employees.
FULL OUTER JOIN Departments
ON Employees.DepartmentID = Department.
                            DepartmentID
```

Page No.
Date

Result :-

Name	Department Name
Alice	HR
Bob	IT
Charlie	NULL
David	AR
NULL	Marketting.

Q.1 What is index? what are advantages and disadvantages of using index on table?

→ An index in a database management system (DBMS) is a special data structure that provides fast way to look up data in a table.

Indexes are created on columns in a table, and they allow the database to locate data efficiently without scanning the entire table.

* Common types of indexes:-

① Primary index :- Automatically created on primary key column.

② Unique index :- The indexed columns do not have duplicate value.

③ Clustered Index :- Sorts the actual data rows in the table based on the indexed columns.

④ Non-clustered Index :-
- creates a seprate structure from the data rows with pointers back to the actual data.

* Advantages :-

① Faster data retrieval :- Indexes drastically improve query performance by reducing the number of rows the database need to scan.

2. Improved sorting and searching :-
queries with "order by", "group by", and "where" clauses are executed faster when relevant columns are indexed.

3. Enhanced performance :-
When joining table, indexes help locate matching rows quickly, making complex queries run faster.

4. Uniqueness Enforcement :-
Indexes can enforce uniqueness in a column, preventing duplicate value.

5. Efficient full-text search
6. Reduced I/O operations

Disadvantages :

1. Increased storage space :- Indexes consume additional ~~storage~~ disk space

2. Slower data modification (Insert, Update, Delete) : Indexes need to be updated whenever data in the indexed columns changes, slowing down write operations.

3. Complex Index Management :-
Having too many indexes can lead to complex decision-making by the query optimizer.

4. Performance Trade offs :- while indexes speed up read operations, they slow down operations.

2 SQL Queries

SQL (Structured Query Language) :- is a standardized language used to manage and manipulate relational databases.

→ you can write all small or all uppercase.

1 Select Query

Purpose :-
To retrieve data from one or more tables.

example :-

```
Select column1, column2
From table_name
where condition ;
```

2] Insert query :-

Purpose :- To insert new rows into a table.

example :-

```
INSERT INTO table_name (col1, col2)
values (value1, value2);
```

3] Update query

Purpose :- To modify existing data in a table.

example :-

```
UPDATE table_name
SET column1 = value1, column2 = value2
WHERE condition ;
```

4] Delete query :-

Purpose :- To remove data from a table

e.g.

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

⑤ CREATE TABLE query

Purpose:- To create a new table in the database.

example:-

CREATE TABLE table_name (column1 datatype column2 datatype-);

⑥ ALTER TABLE query :-

Purpose:- To modify an existing table structure (e.g add, delete, or modify column).

example: * add new column *

ALTER TABLE table_name
ADD column_name datatype;

* Drop a column *

e.g ALTER TABLE table_name
DROP COLUMN column_name;

* Modify a column

e.g = ALTER TABLE table_name
MODIFY column_name new_datatype;

⑦ DROP TABLE Query :-

Purpose:- To delete an entire table from the database.

e.g

DROP TABLE table_name;

4] What is con importance of creating constraints on table? Explain with example any 4 constraints that can be specified when a database table is created.

→

- Creating table in a database is fundamental because it allows for the organization, storage and management of data in the structural format.

- Tables are essentially the backbone of a relational database, where data is stored in rows & columns.

- Each table represents a specific entity or concept.

- The column defines the attributes or concept characteristics of that entity.

- Here are some reasons why creating tables is important.

i] Structured Data Storage -
ii] Data Integrity
iii] Data Retrival
iv] Relationship Manggement.

- When creating a table, you can specify various constraints to enforce rules & ensure data integrity.

- Here are 4 common constraints & their examples

i] Primary Key Constraints.

- Primary key Constraint uniquely identifies each record in the table.
- It ensures that no two rows have the same value in the primary key column and that the column cannot contains a null values.

Eg- CREATE TABLE Employee (

EmployeeId INT PRIMARY KEY,
Firstname varchar(50),
Lastname varchar(50),
);

In these example, 'EmployeeId' is the primary key, which means each 'EmployeeId' must be unique and cannot be 'NULL'.

ii] Foreign key Constraints:

- Importance is that the foreign key constraint ensures the value in one table matches a value in another table.
- This maintains referential integrity between related table & prevents orphaned records.

Eg-
```
CREATE TABLE Orders (
    OrderID INT PRIMARY KEY,
    OrderDate DATE,
    CustomerID INT,
    FOREIGN KEY (CustomerKeyID)
        References Customer (CustomerID)
);
```

iii] Unique Key Constraints.

- The 'UNIQUE' constraints ensures that all values in a column or a combination of column are distinct accross the table.

- These prevents duplicate entries & helps maintaining data accuracy.

-Eg-
```
CREATE TABLE Users (
    UserID INT PRIMARY KEY;
    Username VARCHAR (50),
    Email VARCHAR (100) UNIQUE
);
```

- In a 'Users' table, you might want to ensure th each email address is unique. You would use the 'UNIQUE' constraint on email column.

iv] Example Cheak Constraint:

- The 'CHEAK' constraint allows you to specify a condition that must be met for a row to be inserted or updated.

- This is useful for enforcing domain-specific rules & constraints.

- Eg-

In a 'Products' table, you might want to ensure that the 'Price' column always has a value greater than zero. You can use the 'CHEAK' constraints to enforce this rule.

```
CREATE TABLE Products (
    ProductID INT PRIMARY KEY,
    ProductName VARCHAR(100),
    Price DECIMAL(10,2),
    CHEAK (Price > 0)
);
```

By applying these constraints, you help ensure that the data in your database remains accurate.

* ~~What is importance~~

* Discuss in detail the operator SELECT 'UNION'.

→

- UNION operator is used to combine result-set of two or more SELECT statements.

- Each SELECT statement must have:
i) Same number of columns.
ii) Same data type
iii) Have them in same order.

- the 'UNION' operator, including its syntax, behavior, & variations.

- Syntax:-

```
SELECT column-name(s) FROM table1;
  UNION
SELECT column-name(s) FROM table2;
```

- key points:-

1. Column compatibility:-

• The 'SELECT' statement combined using 'UNION' must have the same number of columns.

2. Column Names:
• The column names in the result set are taken from the first 'SELECT' statement.

3. Duplicate Removal:
• By default, 'UNION' removes duplicate rows from the result set.

4. Sorting the result set:
• You can apply an 'ORDER BY' clause to the final result set to sort the combined rows.

- example:-
Assume you have two tables, 'Employees_2023' & 'Employees-2024' & you want to get a combined list of employees from both years.

```
SELECT EmployeeID, EmployeeName,
  Department
FROM Employees_2023
UNION
SELECT EmployeeID, EmployeeName,
  Department
FROM Employees_2024;
```

- this query will combine the result from 'Employees_2023' & 'Employees_2024', eliminating any duplicate rows based on the 'EmployeeID', 'EmployeeName', & 'Department'.

- The 'UNION' operator is a powerful tool in SQL for combining the result of multiple queries into a single result set.