**ASSIGNMENT 4:**

**INTRODUCTION TO DBMS**

**1.INTRODUCTION TO SQL:**

1. **What is SQL, and why is it essential in database management?**

* SQL (Structured Query Language) is the standard language used to interact with relational databases.
* **It is essential because it allows users to:**
* Query data (SELECT)
* Insert, update, or delete data (INSERT, UPDATE, DELETE)
* Define database structures (CREATE, ALTER)
* Control access (GRANT, REVOKE)
* Manage transactions to ensure data integrity
* SQL is crucial for efficiently storing, managing, and retrieving structured data.

1. **Explain the difference between DBMS and RDBMS.**

|  |  |  |
| --- | --- | --- |
| **Feature** | **DBMS** | **RDBMS** |
| **Full form** | Database management system | Relational database management system |
| **Data Storage** | Stores data as files or in hierarchical or network models | Stores data in tables (rows and columns) |
| **Relationships** | Does not support relationships between data | Supports **relationships** using keys (primary/foreign) |
| **Data integrity** | No built-in integrity constraints | Enforces **integrity constraints** (e.g., entity, referential) |
| **Example** | File systems, XML databases | MySQL, PostgreSQL, Oracle, SQL Server |

1. **Describe the role of SQL in managing relational databases.**

* **SQL** plays a central role in managing **relational databases** by providing the language to:
* **Create** and define database structures (CREATE, ALTER)
* **Insert**, **update**, and **delete** data (INSERT, UPDATE, DELETE)
* **Query** data to retrieve specific information (SELECT)
* **Control access** and permissions (GRANT, REVOKE)
* **Manage transactions** for data consistency (BEGIN, COMMIT, ROLLBACK)

1. **What are the key features of SQL?**

* Key Features of SQL:

1. Data Querying – Retrieve data using SELECT statements.
2. Data Manipulation – Insert, update, and delete data (INSERT, UPDATE, DELETE).
3. Data Definition – Create and modify database structures (CREATE, ALTER, DROP).
4. Data Control – Manage user access and permissions (GRANT, REVOKE).
5. Transaction Control – Ensure data integrity with transactions (COMMIT, ROLLBACK).
6. Built-in Functions – Perform operations like counting, averaging, and summing (COUNT(), AVG(), SUM()).
7. Joins – Combine data from multiple tables using relationships.

**2.SQL Syntax:**

1. **What are the basic components of SQL syntax?**

Basic Components of SQL Syntax:

1. Statements – Instructions like SELECT, INSERT, UPDATE, DELETE, CREATE.
2. Clauses – Parts of statements, such as WHERE, FROM, GROUP BY, ORDER BY.
3. Expressions – Combinations of columns, values, and operators (e.g., salary > 5000).
4. Predicates – Conditions used in clauses (e.g., WHERE age > 30).
5. Identifiers – Names of tables, columns, or databases (e.g., employees, id).
6. Keywords – Reserved words with special meaning (e.g., SELECT, FROM, WHERE).
7. Operators – Symbols for comparisons or calculations (e.g., =, >, <, AND, OR).
8. **Write the general structure of an SQL SELECT statement.**

**PROGRAM:**

SELECT column1, column2, ...

FROM table\_name

WHERE condition

GROUP BY column

HAVING condition

ORDER BY column [ASC|DESC];

1. **Explain the role of clauses in SQL statements.**

* **Clauses** in SQL are components of SQL statements that define specific parts of a query. Each clause serves a unique purpose to filter, group, or sort data.
* **Key Clauses and Their Roles:**

1. **SELECT** – Specifies which columns to retrieve.
2. **FROM** – Indicates the table(s) to query.
3. **WHERE** – Filters rows based on conditions.
4. **GROUP BY** – Groups rows that have the same values in specified columns.
5. **HAVING** – Filters groups (used after GROUP BY).
6. **ORDER BY** – Sorts the result set in ascending or descending order.
7. **JOIN** – Combines rows from two or more tables based on related columns.

**3. SQL Constraints:**

1. **What are constraints in SQL? List and explain the different types of constraints.**

* **Constraints** in SQL are rules that enforce data integrity in tables.
* **Types (in brief):**

1. **PRIMARY KEY** – Uniquely identifies each row; no NULL.
2. **FOREIGN KEY** – Links tables; enforces relationships.
3. **UNIQUE** – Ensures all values in a column are different.
4. **NOT NULL** – Requires a value (no NULL allowed).
5. **CHECK** – Validates data against a condition.
6. **DEFAULT** – Sets a default value if none is given.

* These constraints help maintain **accurate and consistent data**.

1. **How do PRIMARY KEY and FOREIGN KEY constraints differ?**

| **Feature** | **PRIMARY KEY** | **FOREIGN KEY** |
| --- | --- | --- |
| **Purpose** | Uniquely identifies a record | Links to a primary key in another table |
| **Uniqueness** | Must be unique | Can be duplicated |
| **Null Allowed** | Not allowed | Allowed |
| **Defined In** | Parent table | Child table |

**10)What is the role of NOT NULL and UNIQUE constraints?**

 **NOT NULL** = Field must have a value.

 **UNIQUE** = Field must have unique values (no duplicates).

**4. Main SQL Commands and Sub-commands (DDL):**

**11)Define the SQL Data Definition Language (DDL).**

* SQL Data Definition Language (DDL) is a part of SQL used to define and manage the structure of database objects like tables and schemas.
* The main DDL commands are:
* CREATE – to create new tables or other database objects
* ALTER – to modify existing objects
* DROP – to delete objects
* TRUNCATE – to remove all data from a table quickly
* RENAME – to rename database objects

**12)Explain the CREATE command and its syntax.**

* The **CREATE** command is used to **create new database objects**, such as tables, views, indexes, or databases.
* SYNTAX:

CREATE TABLE table\_name (

column1 datatype [constraint],

column2 datatype [constraint],

...

);

**13)What is the purpose of specifying data types and constraints during table creation?**

* **Data Types:**
* Define the **kind of data** each column will hold (e.g., INT, VARCHAR, DATE).
* Ensure **data consistency** and **storage efficiency**.
* Prevent invalid data entry (e.g., text in a numeric field).
* **Constraints:**
* Enforce **rules** on data (e.g., NOT NULL, UNIQUE, PRIMARY KEY).
* Ensure **data accuracy**, **integrity**, and **reliability**.
* Control relationships between tables (FOREIGN KEY).

**5. ALTER Command:**

**14)What is the use of the ALTER command in SQL?**

* The **ALTER** command in SQL is used to **modify the structure of an existing database object**, usually a table. It lets you add, delete, or change columns and constraints without deleting the table or losing data.
* **Common uses of ALTER:**
* Add a new column
* Modify an existing column’s data type or size
* Drop (remove) a column
* Add or drop constraints (like PRIMARY KEY, UNIQUE)

**15) How can you add, modify, and drop columns from a table using ALTER?**

* the ALTER command to **add**, **modify**, and **drop** columns in a table
* Use ADD to create a new column.
* Use MODIFY (or ALTER COLUMN) to change a column’s definition.
* Use DROP COLUMN to remove a column from the table.

**6.DROP Command:**

**16) What is the function of the DROP command in SQL?**

* The DROP command in SQL is used to delete database objects like tables, views, or databases entirely.
* When you drop a table, all the data, structure, and related indexes are permanently removed.

**17) What are the implications of dropping a table from a database?**

* Dropping a table from a database **permanently deletes**:
* The table’s **structure** (its definition and columns)
* All the **data** stored in the table
* Any **indexes**, **triggers**, or **constraints** related to that table
* Relationships linked via **foreign keys** (which may cause errors if not handled)
* **Implications:**
* Data loss — cannot be recovered unless backed up
* Application queries depending on the table will fail
* Referential integrity issues if related tables depend on it

1. **Data Manipulation Language (DML):**

**18) Define the INSERT, UPDATE, and DELETE commands in SQL.**

* In SQL, the **INSERT**, **UPDATE**, and **DELETE** commands are Data Manipulation Language (DML) statements used to manage data within tables of a relational database.

**1. INSERT**

* The INSERT command is used to add new records (rows) to a table.

**SYNTAX:**

INSERT INTO table\_name (column1, column2, ...)

VALUES (value1, value2, ...);

**2. UPDATE**

* The UPDATE command is used to modify existing records in a table.

**SYNTAX:**

UPDATE table\_name

SET column1 = value1, column2 = value2, ...

WHERE condition**;**

**3. DELETE**

* The DELETE command is used to remove existing records from a table.

**SYNTAX:**

DELETE FROM table\_name

WHERE condition;

**19) What is the importance of the WHERE clause in UPDATE and DELETE operations?**

* The WHERE clause is crucial in UPDATE and DELETE operations because it specifies which rows should be updated or deleted.
* **Importance:**
* Prevents unintended changes: Without a WHERE clause, all rows in the table will be affected.
* Targets specific records: Ensures only the intended rows are modified or removed.

1. **Data Query Language (DQL):**

**20) What is the SELECT statement, and how is it used to query data?**

* SELECT is an SQL command used to retrieve data from a database.
* **How it works:**
* You specify the columns you want to see.
* You specify the table(s) to get the data from.
* You can filter, sort, and limit the results.
* **Key points:**
* Use \* to select all columns (e.g., SELECT \* FROM employees;).
* Use WHERE to filter rows.
* Use ORDER BY to sort results.
* Use LIMIT to restrict the number of rows returned.

**20) Explain the use of the ORDER BY and WHERE clauses in SQL queries.**

* **WHERE Clause:**
* **Purpose**: Filters rows to return only those that meet a specified condition.
* **Use case**: Select records based on criteria.
* EXAMPLE:

SELECT \* FROM employees

WHERE department = 'Sales';

* **ORDER BY Clause**
* **Purpose:** Sorts the result set by one or more columns.
* **Default order:** Ascending (ASC)
* **You can specify:** Descending (DESC)
* EXAMPLE:

SELECT \* FROM employees

ORDER BY salary DESC;

1. **Data Control Language (DCL):**

**21) What is the purpose of GRANT and REVOKE in SQL?**

* **GRANT**: Gives specific privileges to a user or role, such as SELECT, INSERT, UPDATE, or DELETE on a table.
* EXAMPLE:

GRANT SELECT ON employees TO user1;

* **REVOKE**: Removes previously granted privileges from a user or role.
* EXAMPLE:

REVOKE SELECT ON employees FROM user1;

* **Purpose**: To control access to data and ensure database security.

**22) How do you manage privileges using these commands?**

* GRANT to give users access to perform actions (e.g., SELECT, INSERT) on database objects.
* REVOKE to remove those access rights.

1. **Transaction Control Language (TCL):**

**23) What is the purpose of the COMMIT and ROLLBACK commands in SQL?**

* **COMMIT**: Saves all changes made in the current transaction permanently to the database.
* **ROLLBACK**: Cancels all changes made in the current transaction and restores the database to its previous state.
* They are used to manage transactions and ensure data integrity.

**24) Explain how transactions are managed in SQL databases.**

* Transaction management in SQL ensures data integrity using the ACID principles (Atomicity, Consistency, Isolation, Durability).
* **BEGIN:** Starts a transaction.
* **COMMIT**: Saves all changes permanently.
* **ROLLBACK**: Undoes all changes if an error occurs.
* It treats multiple SQL operations as a single unit to ensure reliable and consistent database updates.

**11. SQL Joins**

**25) Explain the concept of JOIN in SQL. What is the difference between INNER JOIN, LEFT JOIN, RIGHT JOIN, and FULL OUTER JOIN?**

* **JOIN in SQL**
* Used to combine rows from two or more tables based on a related column.

| **JOIN Type** | **Description** | **Result Includes** |
| --- | --- | --- |
| **INNER JOIN** | Returns only the **matching rows** from both tables | Matches in both tables only |
| **LEFT JOIN** | Returns all rows from the **left table**, and matched rows from the right | All from left + matched from right |
| **RIGHT JOIN** | Returns all rows from the **right table**, and matched rows from the left | All from right + matched from left |

**26) How are joins used to combine data from multiple tables?**

* **Joins** combine data from multiple tables by matching rows based on a related column (usually a key).

**EXAMPLE:**

SELECT \*

FROM orders

JOIN customers ON orders.customer\_id = customers.customer\_id;

* this returns combined rows from both tables where customer\_id matches.

**12.SQL Group By:**

**27) What is the GROUP BY clause in SQL? How is it used with aggregate functions?**

* **GROUP BY in SQL:**
* The **GROUP BY** clause groups rows that have the same values in specified columns, allowing you to perform **aggregate functions** on each group.
* Example:

SELECT department, COUNT(\*)

FROM employees

GROUP BY department;

**28) Explain the difference between GROUP BY and ORDER BY.**

| **Feature** | GROUP BY | ORDER BY |
| --- | --- | --- |
| **Purpose** | Groups rows based on column values | Sorts rows based on column values |
| **Used With** | Aggregate functions (SUM(), COUNT(), etc.) | Sorting results (ascending/descending) |
| **Affects Result** | Reduces rows by grouping | Rearranges rows without reducing |

**13.SQL Stored Procedure:**

**29) What is a stored procedure in SQL, and how does it differ from a standard SQL query?**

A stored procedure is a precompiled collection of one or more SQL statements stored in the database. It can include logic (like conditions and loops) and be executed with a single call.

**Key Features:**

* Reusable: Can be called multiple times.
* Accepts parameters (optional).
* Improves performance (precompiled).
* Enhances security (users can execute without seeing the code).

**Example:**

CREATE PROCEDURE GetEmployeeByDept(@DeptID INT)

AS

BEGIN

SELECT \* FROM employees WHERE department\_id = @DeptID;

END;

| **Feature** | **Stored Procedure** | **Standard SQL Query** |
| --- | --- | --- |
| **Definition** | Named and stored in the database | Written and executed on the fly |
| **Reusability** | Reusable and can accept parameters | One-time use unless manually reused |
| **Logic** | Can include control structures (IF, WHILE) | Only a single set of instructions |
| **Security** | Access control via procedure permissions | Users see and execute raw queries |

**30) Explain the advantages of using stored procedures.**

* Faster Execution – Precompiled and stored in the database.
* Reusable – Can be called multiple times with different inputs.
* Secure – Users can run them without direct table access.
* Maintainable – Centralized logic that's easy to update.
* Less Network Load – Only the procedure call and result travel over the network.
* Consistent – Ensures uniform data handling and validation.

**14.SQL View:**

**31) What is a view in SQL, and how is it different from a table?**

* A **view** is a **virtual table** based on the result of a SQL query. It does not store data itself but displays data from one or more tables.
* Example:

CREATE VIEW ActiveEmployees AS

SELECT name, department

FROM employees

WHERE status = 'Active';

* Difference Between a View and a Table:

| **Feature** | **View** | **Table** |
| --- | --- | --- |
| **Storage** | Does **not** store data (virtual) | Stores **actual** data |
| **Definition** | Based on a SELECT query | Created with schema and data |
| **Updatable** | Sometimes (if simple) | Always updatable |

**32) Explain the advantages of using views in SQL databases.**

* Simplify queries – Hide complex SQL logic.
* Enhance security – Restrict access to specific data.
* Improve maintainability – Centralize and reuse logic.
* Provide abstraction – Users don't need to know table structure.
* Ensure consistency – Uniform data output across applications.

**15. SQL Triggers**

**33) What is a trigger in SQL? Describe its types and when they are used.**

A **trigger** is a procedure that runs automatically on INSERT, UPDATE, or DELETE.

**Types:**

* **BEFORE**: Runs before the event (for validation).
* **AFTER**: Runs after the event (for logging).
* **INSTEAD OF**: Replaces the event, mainly for views.

**Use:** Validate data, audit changes, enforce rules, automate tasks.

**34) Explain the difference between INSERT, UPDATE, and DELETE triggers.**

* **INSERT triggers** run when rows are added.
* **UPDATE triggers** run when rows are changed.
* **DELETE triggers** run when rows are removed

| **Trigger Type** | **When It Fires** | **Purpose/Use Case** |
| --- | --- | --- |
| **INSERT** | After or before a new row is added | Validate or modify data before/after insertion, log new entries |
| **UPDATE** | After or before existing row is changed | Check or modify data during updates, track changes |

**16. Introduction to PL/SQL**

**35)What is PL/SQL, and how does it extend SQL's capabilities?**

* PL/SQL (Procedural Language/SQL) is Oracle’s procedural extension to SQL. It adds programming features to SQL, allowing you to write complex scripts and applications inside the database.
* **PL/SQL Extends SQL:**
* **Procedural Logic:** Supports variables, loops, conditions (IF, WHILE, FOR).
* **Control Structures:** Enables flow control and error handling (EXCEPTION blocks).
* **Modularity:** Allows creating stored procedures, functions, and packages.
* **Improved Performance:** Runs inside the database, reducing network traffic.
* **Integration:** Can combine SQL statements with procedural constructs.

**36) List and explain the benefits of using PL/SQL.**

* **SQL Integration**: Seamlessly combines SQL with procedural logic.
* **Block Structure**: Organized code that's easy to read and maintain.
* **Performance**: Reduces network traffic by sending code blocks to the server.
* **Procedural Features**: Supports loops, conditions, and variables for complex logic.
* **Error Handling**: Robust exception handling for runtime errors.
* **Reusability**: Allows creation of reusable procedures and functions.
* **Triggers**: Supports automatic execution of code on database events.

**17. PL/SQL Control Structures**

**37) What are control structures in PL/SQL? Explain the IF-THEN and LOOP control structures.**

* Control structures in PL/SQL are used to control the flow of execution of statements. They help implement logic and repetition in a PL/SQL program.
* There are three main types:

1. **Conditional control** – e.g., IF-THEN, IF-THEN-ELSE, CASE
2. **Iterative control (Loops)** – e.g., LOOP, WHILE, FOR
3. **Sequential control** – e.g., GOTO, EXIT, CONTINUE

**1. IF-THEN Control Structure**

* Used to execute a block of code only if a specified condition is true.

**Syntax:**

IF condition THEN

-- statements

END IF;

**2. LOOP Control Structure**

* Used to repeatedly execute a block of code until explicitly exited using an EXIT or EXIT WHEN statement.

**Syntax:**

LOOP

-- statements

EXIT WHEN condition;

END LOOP;

**38) How do control structures in PL/SQL help in writing complex queries?**

* Control structures in PL/SQL allow you to add **logic and decision-making** to SQL operations, enabling more **dynamic, flexible, and powerful programs**. Here's how they help:

1. **Decision Making (IF-THEN)**
   * Allows conditional execution based on data values.
   * Useful for applying different logic depending on business rules.
2. **Repetition (LOOP, WHILE, FOR)**
   * Automates repetitive tasks like processing rows one by one.
   * Useful for batch operations or custom data processing.
3. **Error Handling (EXCEPTION block)**
   * Helps manage and recover from runtime errors without crashing.
4. **Modularity and Clarity**
   * Makes complex queries easier to organize and maintain by breaking logic into manageable parts.

**18. SQL Cursors:**

**39) What is a cursor in PL/SQL? Explain the difference between implicit and explicit cursors.**

* A cursor is a pointer to the result set of a SQL query. It allows row-by-row processing in PL/SQL.
* **Types of Cursors:**

**1. Implicit Cursor**

* Created automatically by Oracle for single-row queries (e.g., SELECT INTO, INSERT, UPDATE, DELETE).
* No need to declare.
* Accessed using SQL%ROWCOUNT, SQL%FOUND, etc.

**2. Explicit Cursor**

* Manually declared for multi-row queries.
* Requires OPEN, FETCH, and CLOSE.
* **Key Difference:**

| **Implicit Cursor** | **Explicit Cursor** |
| --- | --- |
| Auto-created by Oracle | Manually defined |
| For single-row queries | For multi-row queries |

**40) When would you use an explicit cursor over an implicit one?**

**Use an explicit cursor when:**

1. **The query returns multiple rows**
   * Implicit cursors work only for single-row SELECT INTO queries.
   * Explicit cursors let you fetch and process each row individually.
2. **You need fine control over row processing**
   * With OPEN, FETCH, CLOSE, you control the timing and handling of each row.
3. **You want to loop through query results**
   * Ideal for complex logic where each row needs specific handling.
4. **You need cursor attributes**
   * Like %FOUND, %NOTFOUND, %ROWCOUNT for the cursor, not just the last SQL statement.

**19. Rollback and Commit Savepoint:**

**41) Explain the concept of SAVEPOINT in transaction management. How do ROLLBACK and COMMIT interact with savepoints?**

* A SAVEPOINT marks a point within a transaction to which you can partially roll back without affecting the entire transaction.

**Purpose:**

* Allows partial undo of changes.
* Useful for error handling within complex transactions.

**COMMIT and SAVEPOINT:**

* COMMIT makes all changes permanent, including those before and after savepoints.
* After a COMMIT, all savepoints are lost.

**ROLLBACK and SAVEPOINT:**

* ROLLBACK TO savepoint\_name undoes changes after the savepoint, but keeps changes before it.
* ROLLBACK without a savepoint undoes all changes in the transaction.

**42) When is it useful to use savepoints in a database transaction?**

* **Savepoints are useful when:**
* You want to **partially undo** certain steps in a transaction without rolling back the entire transaction.
* Managing **complex transactions** with multiple steps, allowing recovery from errors at specific points.
* You need **fine-grained control** to handle errors or exceptions within long transactions.
* When you want to **improve efficiency** by avoiding restarting the entire transaction after a minor error.