***Assignment 3***

1. **What is RDBMS?**

In software testing, an RDBMS (Relational Database Management System) is a system used to manage and interact with relational databases, which store data in tables with rows and columns.

RDBMS is crucial for testing as it provides the foundation for verifying data integrity, consistency, and functionality within an application.

It allows testers to create, update, and query data, ensuring the application behaves as expected with its database interactions.

**Relational Database**

A database that organizes data into tables with rows (records) and columns (attributes). These tables are related to each other through shared keys.

**Management System**

The RDBMS software provides the functionalities to interact with the database.

**Data Definition:** Creating, modifying, and deleting database structures (tables, relationships, etc)

**Data Manipulation:**  Adding, updating, and deleting data within the tables.

**Data Retrieval:** Querying the database to retrieve specific information using languages like SQL.

**Data Control:**  Managing access permissions and security to protect the data.

**Examples**

Popular RDBMS include Oracle Database, MySQL, Microsoft SQL Server, and PostgreSQL.

In essence, an RDBMS allows users to interact with a relational database in an organized and controlled manner, ensuring data integrity and efficient access.

1. **What is SQL?**

Structured query language (SQL) is a programming language for storing and processing information in a relational database.

Structured Query Language (SQL) is used to store, manage, and get data from a database.

Suppose there is a table named Employees. In that table, there are 3 columns. One is Employee ID, the Second Column is Employee Name, and the Third Column is Employee Phone Number.

1. **Write SQL Commands?**

SQL (Structured Query Language) is the standard language used to interact with and manage relational databases.

SQL commands are essentially instructions you give to the database to perform various operations, like creating tables, inserting data, querying information, and controlling access and security.

These commands are categorized into five main types, each serving a distinct purpose.

Data Definition Language (DDL): Used for defining and modifying the structure or

schema of the database.

Data Manipulation Language (DML): Used for manipulating (adding, changing,

deleting) data within the database.

**Data Query Language (DQL):** Used for retrieving data from the database.

Data Control Language (DCL): Used for controlling access to data within the

database.

Transaction Control Language (TCL): Used for managing transactions (groups of

operations that should succeed or fail as a single unit).

DDL (Data Definition Language) commands

These commands define the database structure.

* CREATE TABLE: Used to create a new table in the database.

sql

CREATE TABLE Employees (

EmployeeID INT PRIMARY KEY,

FirstName VARCHAR(50),

LastName VARCHAR(50),

Department VARCHAR(50)

);

Use code with caution.

This creates a new table called Employees with columns for EmployeeID, FirstName, LastName, and Department.

* ALTER TABLE: Used to modify the structure of an existing table.
  + Add a column:

sql

ALTER TABLE Employees ADD Email VARCHAR(100);

Use code with caution.

This adds a new column named 'Email' to the Employees table.

* + Modify a column's data type:

sql

ALTER TABLE Employees MODIFY COLUMN LastName VARCHAR(150);

Use code with caution.

This changes the data type of the 'LastName' column.

* + Drop a column:

sql

ALTER TABLE Employees DROP COLUMN PhoneNumber;

Use code with caution.

This removes the 'PhoneNumber' column from the Employees table.

* DROP TABLE: Used to delete an entire table from the database.

sql

DROP TABLE Employees;

Use code with caution.

This deletes the Employees table and all its data.

* TRUNCATE TABLE: Used to delete all records from a table without deleting the table structure itself.

sql

TRUNCATE TABLE Employees;

Use code with caution.

This removes all rows from the Employees table but keeps its structure.

* CREATE DATABASE: Used to create a new database.

sql

CREATE DATABASE CompanyDB;

Use code with caution.

This creates a new database named CompanyDB.

* DROP DATABASE: Used to delete a database.

sql

DROP DATABASE CompanyDB;

Use code with caution.

This deletes the CompanyDB database.

DML (Data Manipulation Language) commands

These commands are used to manipulate data within the database tables.

* INSERT INTO: Used to add new rows or records into a table.

sql

INSERT INTO Employees (EmployeeID, FirstName, LastName, Department)

VALUES (101, 'John', 'Doe', 'HR');

Use code with caution.

This adds a new employee record to the Employees table.

* UPDATE: Used to modify existing records in a table.

sql

UPDATE Employees SET Salary = 50000 WHERE EmployeeID = 101;

Use code with caution.

This updates the salary of the employee with EmployeeID 101.

* DELETE FROM: Used to delete existing records (rows) from a table.

sql

DELETE FROM Employees WHERE EmployeeID = 101;

Use code with caution.

This deletes the employee record with EmployeeID 101.

* SELECT: Used to retrieve data from one or more tables in a database.

sql

SELECT FirstName, LastName, Department FROM Employees;

Use code with caution.

This retrieves the FirstName, LastName, and Department of all employees.

* SELECT \*: Retrieves all columns from a table.

sql

SELECT \* FROM Employees;

Use code with caution.

This retrieves all data for all employees.

* WHERE: Used with SELECT, UPDATE, or DELETE to filter records based on a specified condition.

sql

SELECT \* FROM Employees WHERE Department = 'Sales';

Use code with caution.

This retrieves all data for employees in the 'Sales' department.

* ORDER BY: Used to sort the result set of a SELECT statement.

sql

SELECT FirstName, LastName FROM Employees ORDER BY LastName ASC;

Use code with caution.

This sorts the results by LastName in ascending order.

* GROUP BY: Used with aggregate functions to group rows that have the same values in specified columns.

sql

SELECT Department, COUNT(\*) AS NumberOfEmployees FROM Employees GROUP BY Department;

Use code with caution.

This groups employees by department and counts the number of employees in each department.

* HAVING: Used with the GROUP BY clause to filter groups based on a specified condition.

sql

SELECT Department, COUNT(\*) FROM Employees GROUP BY Department HAVING COUNT(\*) > 5;

Use code with caution.

This filters groups to show only departments with more than 5 employees.

* JOIN: Combines rows from two or more tables based on a related column.

sql

SELECT Employees.FirstName, Employees.LastName, Departments.DepartmentName

FROM Employees

INNER JOIN Departments ON Employees.DepartmentID = Departments.DepartmentID;

Use code with caution.

This joins the Employees and Departments tables to retrieve employee names and their corresponding department names.

DCL (Data Control Language) commands

These commands are used to control access and permissions within a database.

* GRANT: Used to assign new privileges to a user account or role, allowing them to access specific database objects or functions.

sql

GRANT SELECT ON Employees TO HR\_Manager;

Use code with caution.

This grants the 'HR\_Manager' role the privilege to select data from the 'Employees' table.

* REVOKE: Used to remove previously granted privileges from a user account or role.

sql

REVOKE SELECT ON Employees FROM HR\_Manager;

Use code with caution.

This removes the SELECT privilege on the 'Employees' table from the 'HR\_Manager' role.

TCL (Transaction Control Language) commands

These commands manage transactions within a database.

* BEGIN TRANSACTION: Starts a new transaction, marking the beginning of a sequence of operations that should be treated as a single unit.
* COMMIT: Saves all changes made during a transaction permanently to the database.

sql

BEGIN TRANSACTION;

UPDATE Employees SET Salary = Salary \* 1.1 WHERE Department = 'Sales';

COMMIT;

Use code with caution.

This increases the salary of all employees in the 'Sales' department by 10% and then commits the changes to the database.

* ROLLBACK: Undoes all changes made during a transaction, restoring the database to its state before the transaction began.

sql

BEGIN TRANSACTION;

DELETE FROM Employees WHERE EmployeeID = 101;

ROLLBACK;

Use code with caution.

This starts a transaction, deletes an employee record, but then rolls back the changes, effectively undoing the deletion.

* SAVEPOINT: Creates a point within a transaction to which you can later roll back if needed.

sql

BEGIN TRANSACTION;

UPDATE Employees SET Salary = Salary \* 1.1 WHERE Department = 'Sales';

SAVEPOINT before\_another\_update;

UPDATE Employees SET Salary = Salary \* 1.05 WHERE Department = 'Marketing';

ROLLBACK TO SAVEPOINT before\_another\_update;

COMMIT;

Use code with caution.

This example increases the salary of employees in 'Sales', sets a savepoint, increases the salary of employees in 'Marketing', then rolls back to the savepoint before the 'Marketing' update, and finally commits the transaction.

1. **What is join?**

SQL JOIN is a database operation that combines rows from two or more tables based on a related column between them.

It allows you to create a single result set from multiple tables, providing a way to retrieve data that is spread across various tables and establish relationships between them.

1. **Write type of joins.**

4 types of joins

1. **(INNER) JOIN:** Returns records that have matching values in both tables.
2. LEFT (OUTER) JOIN: Returns all records from the left table, and the matched records from the right table.



1. RIGHT (OUTER) JOIN: Returns all records from the right table, and the matched records from the left table.



1. FULL (OUTER) JOIN: Returns all records when there is a match in either left or right table



1. **How Many constraint and describes it self.**

In the context of databases, constraints are rules that define the valid data within a table.

 They ensure data integrity and consistency by enforcing restrictions on the data that can be inserted, updated, or deleted.

There are several types of constraints, each serving a specific purpose:

Types of Constraints and their Descriptions:

1. **NOT NULL:** Prevents a column from having a NULL value.
2. **UNIQUE:** Ensures that all values in a column or a set of columns are unique.
3. **PRIMARY KEY:** Uniquely identifies each row in a table. It combines the characteristics of NOT NULL and UNIQUE.
4. **FOREIGN KEY:** Establishes a link between two tables. It ensures that the values in one table (the referencing table) match the values in a column of another table (the referenced table), maintaining referential integrity.
5. **CHECK:** Enforces domain integrity by specifying a condition that each value in a column must satisfy.
6. **DEFAULT:** Specifies a default value for a column if no value is explicitly provided during insertion.
7. **INDEX:** Creates an index on a column or a set of columns to improve query performance. While not strictly an integrity constraint, it can help enforce uniqueness and other rules indirectly.
8. **Difference between RDBMS vs DBMS.**

DBMS (Database Management System) is a broader category of software that manages databases, while RDBMS (Relational Database Management System) is a specific type of DBMS that uses the relational model to organize data into tables with predefined relationships.

 In essence, all RDBMS systems are DBMS, but not all DBMS are RDBMS.

DBMS (Database Management System):

* **Data Storage:** Can store data in various formats like hierarchical, network, or object-oriented, not just tables.
* **Relationships:** May or may not have defined relationships between data elements.
* **Query Language:** Can use different query languages, not just SQL.
* **Examples:** Includes file systems, XML databases, and older hierarchical/network databases.

RDBMS (Relational Database Management System):

* **Data Storage:** Stores data in tables with rows and columns, establishing relationships between tables using keys.
* **Relationships:** Emphasizes relationships between data elements, using primary and foreign keys to link tables.
* **Query Language:** Primarily uses SQL (Structured Query Language) for data manipulation.
* **Examples:** Includes MySQL, PostgreSQL, Oracle, SQL Server, and Microsoft Access.

Key Differences Summarized:

|  |  |  |
| --- | --- | --- |
| **Feature** | **DBMS** | **RDBMS** |
| Data Model | Various (hierarchical, network, etc.) | Relational (tables, rows, columns) |
| Relationships | May or may not be defined | Defined using keys |
| Query Language | Varies | Primarily SQL |
| Data Redundancy | Can be high | Reduced through normalization |
| Scalability | Varies | Highly scalable |
| Complexity | Generally simpler | More complex |
| Examples | File systems, XML databases | MySQL, PostgreSQL, Oracle, etc. |
| Data Integrity | Varies | Enforces integrity through constraints |
| User Support | May be single-user | Supports multiple users |
| Distributed DB | Not always supported | Supports distributed databases |

1. **What is API Testing.**

API testing is a type of software testing that focuses on verifying the functionality, reliability, performance, and security of Application Programming Interfaces (APIs).

APIs are the building blocks that allow different software systems to communicate and exchange data, and API testing ensures these interactions are working correctly.

It involves sending requests to the API, validating the responses, and checking for errors, performance issues, or security vulnerabilities.

1. **Types of API Testing.**

API testing involves various methods to ensure an API functions correctly, securely, and efficiently. Key types include functional, security, performance, integration, and load testing, each with specific goals.

**1. Functional Testing**:

* **Purpose:**

Verifies the API's core functionality by testing if it meets specified requirements.

* **Focus:**

Checks if the API performs as expected, handling various inputs and producing the correct outputs.

* **Example:**

Testing an e-commerce API to ensure that adding items to a cart, calculating the total, and processing payment all work correctly.

**2. Security Testing:**

* **Purpose:**

Identifies vulnerabilities and ensures the API is protected against potential threats.

* **Focus:**

Authentication, authorization, data encryption, and input validation are crucial aspects.

* **Example:**

Testing for SQL injection vulnerabilities, cross-site scripting (XSS) attacks, or unauthorized access to sensitive data.

**3. Performance Testing:**

* **Purpose:** Evaluates the API's speed, responsiveness, and scalability under different load conditions.
* **Focus:** Measuring response times, throughput, and resource utilization.
* **Example:** Simulating multiple users accessing the API simultaneously to assess its performance under high traffic.

**4. Integration Testing:**

* **Purpose:**

Ensures that different modules or services within the application interact correctly through the API.

* **Focus:**

Testing how the API integrates with other components, databases, and external systems.

* **Example:**

Testing an API's integration with a payment gateway to ensure accurate transaction processing.

**5. Load Testing:**

* **Purpose:** Determines how the API behaves under expected and peak load conditions.
* **Focus:** Simulating various load levels to identify bottlenecks and performance issues.
* **Example:** Testing the API's ability to handle a large number of concurrent users without performance degradation.

**6. Unit Testing:**

* **Purpose:** Tests individual components or functions of the API in isolation.
* **Focus:** Validating the correctness of each unit of code, such as functions, methods, or classes.
* **Example:** Testing the "login" function of an API to ensure user credentials are authenticated correctly.

**7. Fuzz Testing:**

* **Purpose:**

Finds vulnerabilities by sending invalid, unexpected, or random data to the API.

* **Focus:**

Identifying potential crashes or unexpected behaviour caused by malformed inputs.

* **Example:**

Sending a large number of random characters to an API endpoint to see if it crashes or produces an error.

**8. Regression Testing:**

* **Purpose:**

Ensures that new code changes or updates to the API don't break existing functionality.

* **Focus:**

Re-running previously executed tests to verify that the API still behaves as expected.

* **Example:**

Running regression tests after fixing a bug to ensure that the fix doesn't introduce new issues.

**9. Validation Testing:**

* **Purpose:**

Ensures that the API meets all the specified requirements and functions as intended.

* **Focus:**

Verifying the API's functionality, performance, and security against the defined requirements.

* **Example:**

Comparing the actual results of API calls with the expected results to validate the API's behaviour.

1. **What is Responsive Testing?**

Responsive testing in software testing ensures a website or application displays and functions correctly across various devices, screen sizes, and browsers.

 It's crucial for delivering a seamless user experience regardless of whether a user is on a desktop, tablet, or smartphone.

This testing process involves verifying layout adjustments, navigation, content rendering, and overall functionality on different platforms.

1. **Which types of tools are available for Responsive Testing.**

Responsive testing tools help ensure websites and web applications display correctly across various devices and screen sizes.

These tools can be broadly categorized into emulators/simulators, cross-browser testing platforms, and dedicated responsive design checkers.

**1. Emulators/Simulators:**

* **Google Chrome DevTools Device Mode:**

This built-in browser tool allows developers to simulate various devices and screen sizes within the Chrome browser, enabling them to test how their website renders on different devices without needing the physical devices.

* **LT Browser:**

A dedicated browser built by Lambda test for responsive testing, offering multiple device viewports, dedicated Chrome DevTools, and features like hot reloading and Lighthouse reports for performance analysis.

**2. Cross-Browser Testing Platforms:**

* **Browser Stack:**

A cloud-based platform that provides access to a wide range of real browsers, devices, and operating systems for testing.

* **Lambda Test:**

Another cloud-based platform offering responsive testing on a vast number of real browsers, devices, and OS combinations, with support for automation frameworks like Selenium and Cypress.

* **CrossBrowser Testing:**

Provides interactive testing on different browsers and devices, including features like screenshot comparison and responsive design testing.

* **Comparium:**

An online service that captures screenshots of your website on various platforms for comparison.

**3. Responsive Design Checkers:**

* **Responsinator:**

A free online tool that displays how a website renders across common screen sizes and allows interaction with the site.

* **Screenfly:**

A free online tool that allows developers to test websites on different devices and screen sizes.

* **Responsive Design Checker:**

Tests the responsiveness of websites across various devices and resolutions.

* **Viewport Resizer:**

A browser extension that allows you to test your website on different screen sizes.

**4. Other Tools and Techniques:**

* **Selenium:**

An automation framework that can be used for responsive testing by simulating user interactions across different devices and browsers.

* **Cypress:**

Another popular automation framework known for its speed and reliability in testing, including responsive design aspects.

* **Appium:**

An open-source tool for automating mobile applications, enabling responsive testing on Android and iOS.

* **Visual Testing:**

Tools like Percy (integrated with Browser Stack) can be used to visually compare different versions of your website to identify visual discrepancies caused by responsive design changes.

1. **What is the full form of .ipa, .apk.**

The full forms of . ipa and .apk are iOS App Store Package and Android Application Package, respectively. An .ipa file is used for distributing and installing applications on iOS devices (iPhones, iPads), while an .apk file is used for Android devices.

Here's a more detailed breakdown:

* **.ipa (iOS App Store Package):**

This file format is specifically designed for distributing and installing applications on Apple's iOS operating system. Think of it as the iOS equivalent of an .apk file for Android.

* **.apk (Android Application Package):**

This file format is used by the Android operating system for distributing and installing applications. It contains all the necessary elements for an Android app to be installed and run on a device.

1. **How to create step for to open the developer option mode ON?**

To enable Developer options on an Android device, navigate to Settings > About phone > Build number and tap it seven times.

This will unlock the Developer options menu, which can be found under Settings > System > Developer options (or similar, depending on your device).

Detailed steps:

1. **Open Settings:** Locate and open the Settings app on your Android device.
2. **Find "About phone":** Scroll down and tap on "About phone" or "About device".
3. **Locate "Build number":** Within the "About phone" section, find and tap on the "Build number" option multiple times (usually 7 times).
4. **Confirmation:** You may see a message indicating that you are now a developer or that Developer options have been enabled. You might also be prompted to enter your PIN or password for confirmation.
5. **Access Developer options:** Go back to the main Settings menu. You should now see a "Developer options" or "System" menu item (depending on your device). Tap on it to access the settings.