**SQL   
JDBC   
Interview   
Preparations   
Questions**

### **Do you know SQL?**

* + Yes, I am very comfortable with writing SQL Queries and DML commands and exposure to DDL.
  + Currently working with Oracle database that is running in AMAZON CLOUD SERVER.
  + DDL (Data definition language) : CREATE , ALTER, DROP, TRUNCATE
  + DML(Data manipulation language): SELECT, DELETE, INSERT, UPDATE

**2**.**Why do you use the SQL language ?**

Sql is the language that we use to retrieve some information from a database. I use it to get some info from db and compare it against to UI

**3. Can you use oracle sql developer to connect different types of database ?**

Yes, the only thing that we need to do is add the needed driver to SQL developer.

Preferences -> Database - > Third party JDBC drivers.

**4. What database did you use in your project?**

In my project I used mysql database, but prior to that I have worked with oracle database as well I have done manual testing as well as automated testing on database.

**5. Different database types that you know ?**

Oracle,mysql,postgresql...

noSQL database example => mongoDB, no experience

**6. How do you connect to a database using selenium/java in general?**

In java there is a JDBC api that handles connections to the database. So this is what I used in my project to connect to db and automate.

First, based on the type of the database I need to add the driver to the project. Since I used maven I added to the pom file. for mysql i used the mysql-connector-java driver, for oracle ojdbc driver.

Next I need to create a CONNECTION for this I use the connection class in Java/JDBC.

connection class requires the url, username and password which is connection string

After a connection is created, I use the STATEMENT class to execute queries and RESULT SET class to store the query result and interact with it.

**7. How do you implement database testing ?**

I execute the test manually, first I go and check the test case, implement the UI part then based on the information that I need, I write my SQL query.

I use SQL developer to connect the database and run the queries.

For automating any test case, the first thing I need is getting the connection string which consists of the username,password, and url of the database.

I add that information to my configuration.properties file to manage in one central location.

Then based on the database type I added needed dependency to my maven project.(mysql-connector)

Then I have my DBUtils class which has some reusable methods to create connection,close connection and get the data in java data structure.

If the test data that I am getting from the database is only one row, I use **Map** to store that information as a key and value format. The key will be Column names, and values will be each cell value that belongs to that column.

If I am getting more than one row, table format then I have a ready method that stores information in **List of map**, which is each row represented as a one map they stored inside the list based on the row order.

**8.What did you use to create List of map methods dynamically(follow up question for previous question) ?**

**How do you store ResultSet information in Java data Structure(independent question) ?**

I use resultset object and also resultsetmetadata. in order to get column names and number of columns I use methods from rsmd(resultsetmetadata) . to get cell values and iterate through each row I use ready methods from resultSelt object.

which is .next method with a while loop and getObject method which I can store any type of information.

After I get the result in Java data structure ( map or list of map) I compare those results with that I got from UI.

Assert.assertEquals(UiResult,DbResult);

**9.In Database testing which part is going to be expected, which one actual result ?**

It depends on the scenario/test case.

For example, if we are inserting some data using UI, lets say creating new user, new product,new books etc. then we assume these information stored in the database properly. so we assume these are EXPECTED results because we provide them though UI, and we try to get those information from Database and compare. In this case the database will be the actual result.

Another scenario could be we might have some existing information in the database, and we might wanna check if the database info is correctly mapped to the UI of the application. in this case our expected data will be database data, and actual data will be UI.

**10. What is your current framework structure and how do you implement database testing with it ? (cucumber +db)**

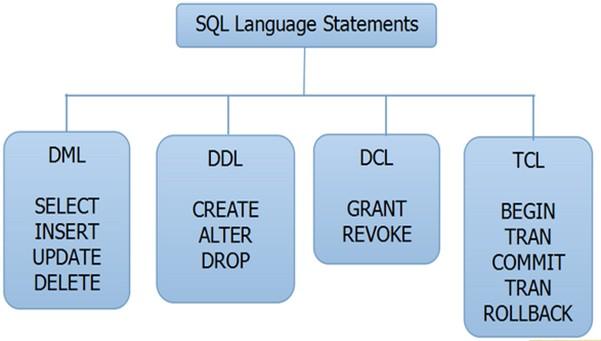
I have a Junit-Cucumber framework that I use for automating UI. I added needed dependency ,for example mysql connector, in pom.xml . and all connection information to configuration.properties

I manage creating and closing database connection though with custom hooks. so I have my custom @Before and @After hooks where I create connection and close the connection, I label my scenarios that includes database testing with the same tag(@db) that I use in my custom hooks.so I can make sure even if the test case fails, I close connection with the database.

### **10. SQL?**

* + Structured Query Language. Used for managing and manipulating data in db.
  + Provide statements for a variety of tasks
    - Querying data
    - Inserting, updating ,deleting rows in a table
    - Creating, replacing, altering, and dropping objects
    - Controlling access to the database and its objects
    - Database consistency and integrity

### **What are the categories of SQL statements?**



**DML (Data Manipulation Language)**

DML statements affect records in a table. These are basic operations we perform on data such as selecting a few records from a table, inserting new records, deleting unnecessary records, and updating/modifying existing records.

**DDL (Data Definition Language)**

DDL statements are used to alter/modify a database or table structure and schema. These statements handle the design and storage of database objects.

**DCL (Data Control Language)**

DCL statements control the level of access that users have on database objects.

**TCL (Transaction Control Language)**

TCL statements allow you to control and manage transactions to maintain the integrity of data within SQL statements.

### **Versions**

* + Java 8 → 2014 present Java 7 → 2011 - 2014 Java 6 → 2006 - 2011

Selenium 3.5.3

### **Database Schema?**

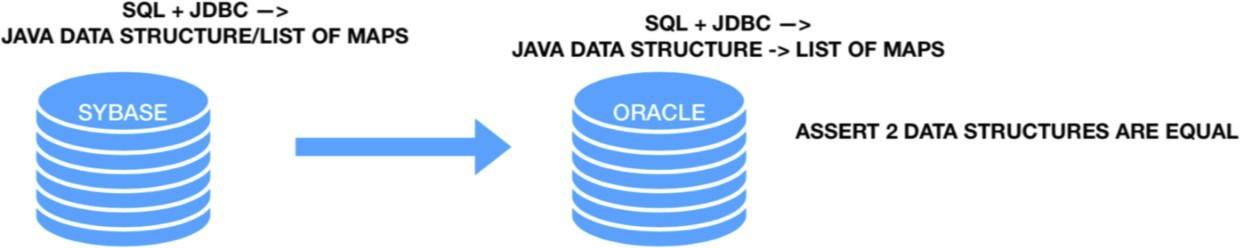
* + It is like a diagram with all tables and column names, data types and PK, FK and how tables are related to each other

### **SQL clause?**

* + SELECT and FROM

### **What kind of Database testing are you doing?**

* + I am mostly doing Database validations.
  + I make changes or insert data (create loan) in the front end and validate in the database. Data in front end matches the DB
  + I also make changes using RESTapi and verify that changes are successful in Database as well.
  + I also support DB migration process. My code connects to Sybase (legacy database) using JDBC then Connects to Oracle (NEW DB) then compare records to make sure data was migrated



### **RDBMS**

* + Relational Database Management System
  + Data is organized into tables that are related to each other
    - How are they related?
      * Primary Key (unique and not NULL) and Foreign Key (
    - What type of database system you have expertise with?
      * RDBMS, such as SQL and Oracle

### **What are constraints?**

* + Properties that table column must comply with.
  + Columns have constraints that defined how data can be stored.
    - Primary Key: unique and NOT NULL
    - Foreign Key: duplicate and NULL and cannot add data which is not in PK
    - Unique Key: only unique value
    - Null: can have null
    - Not null: cannot have null

### **Data types in SQL?**

* + Number
  + Integers
  + char → char(20): 20 years spaces are taken from memory
  + varchar → varchar(30): 5 spaces from memory varchar2
  + boolean
  + date
  + currency

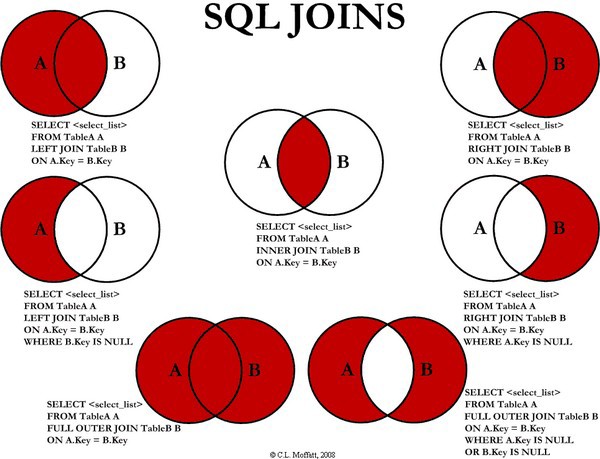
### **Capabilities for SQL select statements**

* + Projection → Select the columns in a table that are returned by a query
  + Selection → Selects the rows in a table that are returned by a query
  + Join → Brings together data that is stored in different tables by specifying the link between them

### **DML (Data Manipulation Language) vs DDL (Data Definition Language)**

|  |
| --- |
| DML command actions can be restored. |
| **Commands**:   * **SELECT** from tablename; (read) * **INSERT** into tablename values (...); (add) * **UPDATE** tablename SET value WHERE location; * **DELETE** from tablename WHERE location; (rows) * **MERGE** |

### **SQL JOIN JOINS**



.

**JOIN (INNER) JOIN**

is used when retrieving data from multiple tables and will return only matching data

**LEFT (OUTER) JOIN**

is used when retrieving data from multiple tables and will return left table and any matching right table records.

**RIGHT (OUTER) JOIN**

is used when retrieving data from multiple tables and will return right table and any matching left table records.

**FULL (OUTER) JOIN**

is used when retrieving data from multiple tables and will return both table records, matching and non-matching

1. **UNION**
   * Union combines the resultSets of two queries

select from column\_names from table\_name {UNION} select column\_name from table\_name)

### **Find duplicate names in employees**

SELECT first\_name, COUNT (first\_name) FROM employees GROUP BY first\_name

HAVING (COUNT(first\_name>1);

1. **How to find employees whose salaries are below the average?**

SELECT first\_name, salary FROM employees

WHERE salary<= (SELECT AVG(salary) FROM employees);

1. **How to find maximum salaries in each department?**

SELECT first\_name, MAX(salary)

FROM department d LEFT OUTER JOIN employee e ON (d.department\_id = e.department\_id) GROUP BY department\_id;

1. **How to find lowest salaries?**

SELECT first\_name, last\_name, salary, job\_id FROM employees WHERE salary = (SELECT MIN(salary) From employees);

1. **How to find second highest salary of employees?**

SELECT MAX(salary) FROM employees

WHERE salary NOT IN (SELECT MAX(salary) FROM employees);

1. **SQL Developer**
   * Development environment (manual testing the database using the queries)
     + Release 2.1 -2009  3.0 - 2011
     + Release 4.0 - 2013(latest)
   * Has a Migration release(1.2) → provides users with a single point to browse data in third-party DB and to migrate from these DB to Oracle
   * Supports Window, Linux and Mac OS x

### **Writing SQL Statements**

* + Keywords are uppercased while columns and table names are lowercase
  + Statements are not case sensitive
  + Clauses are usually placed on separate lines
  + Keywords cannot be abbreviated or split across lines

### **Arithmetic Expressions**

* + You use the operators in any clause (except the From clause)
  + With Date and Timestamp - can only use addition and subtraction
  + Add (+), Subtract (-), Multiply (\*), Divide (/)

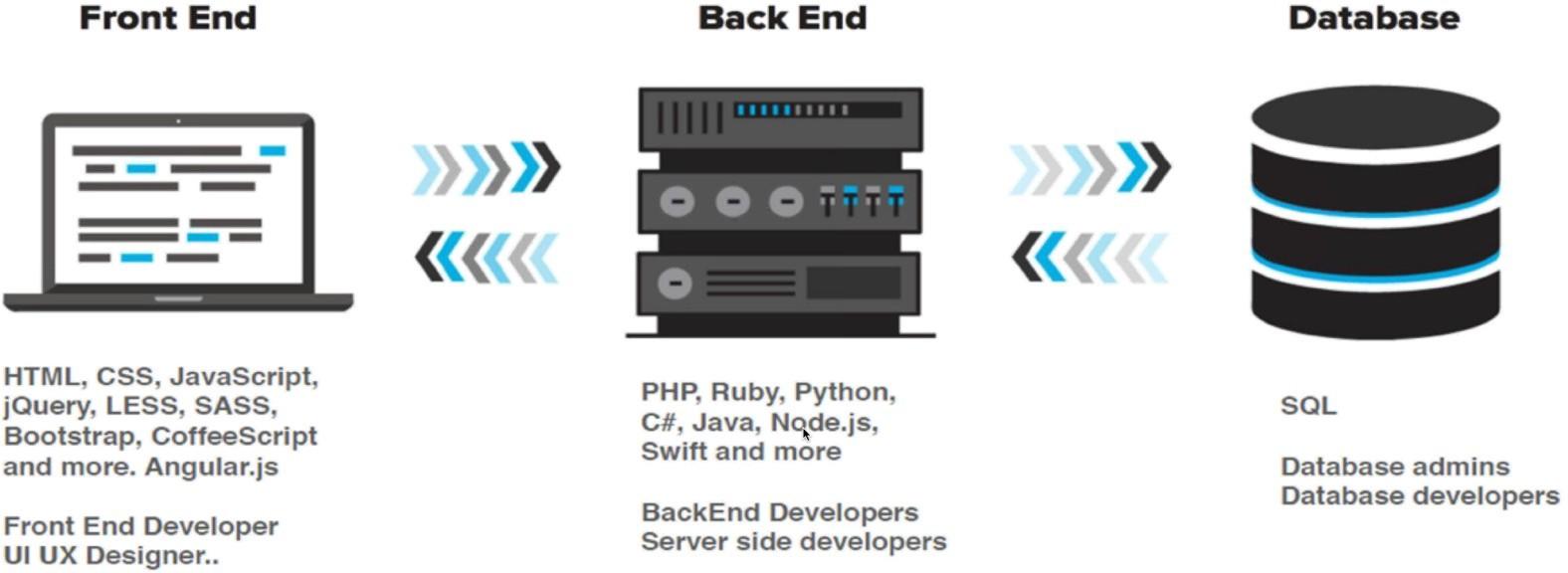
### **Do you have experience with SQL?**

* Yes, I have worked with relational databases and i am very comfortable with and DML commands

### **Back-end test framework**

* + Using RESTASSURED in my framework in order for the process of Deserialization and serialization to occur that way you store Json response into a java collection data structure and assert the data with the expected value (also stored in java data structure)
  + I use postman for manual testing first BEFORE I INVOKE MY FRAMEWORK

### **Back-end Testing**



1. **Have you done any backend/database testing?**
   * Yes, I have lots of experience with working with databases.
   * And I am very comfortable with writing SQL queries.
   * I have experience with working on Relational Databases like Oracle, MySQL, SQL Server
2. **Have you worked with non-relational databases?**
   * I don’t have hands on experience, but I know that it is like JSON format
   * Database and I have good experience with working with JSON files.
   * And I am a quick learner

### **Database testing framework**

* + For manual testing I use SQL developer for producing SQL queries
  + FOR AUTOMATION; I use JDBC library to integrate java by getting a CONNECTION from oracle database then creating STATEMENTS using SQL queries and then storing the data into a RESULTSET object.

### **What is metadata?**

* + MetaData Data about Data

ResultSetMetaData rsmd = rs.getMetaData(); int columNum=rsmd.getColumnCount();

### **Data Structures and Why we need it?**

* + Data structures are way of organizing data for efficient
  + manipulation: Insertion , searching, reading , deletion of data.
  + I always use java data structures for reading data and storing data from our application, database, or API.

### **How can we connect DB?**

Connection connection = DriverManager.getConnection(URL, user, password); Statement statement = connection.createStatement();

ResultSet resultset = statement.executeQuery(“query”);

resultset.close(); statement.close(); connection.close();

// After connection;

DatabaseMetaData db = connection.getMetaData();

// After resultSet;

ResultSetMetaData rs = resultSet.getMetaData();

* + If failure to connect will throw an exception:
* SQLException (bad URL or credentials)
* ClassNotFoundException (JDB driver not in classpath)

### **Dependency for JDBC**

<dependency>

<groupId>oracle</groupId>

<artifactId>ojdbc6</artifactId>

<version>11.2.0.3</version>

</dependency>

### **What is SQL Constraints?**

* + SQL Constraints are rules used to limit the type of data that can go into a table, to maintain the accuracy and integrity of the data inside table.
  + Constraints can be divided into the following two types,
* **Column level constraints**: Limits only column data.
* **Table level constraints**: Limits whole table data.
  + Constraints are used to make sure that the integrity of data is maintained in the database. Following are the most used constraints that can be applied to a table.
* **NOT NULL**  constraint restricts a column from having a **NULL** value

CREATE TABLE Student(s\_id int NOT NULL, Name varchar(60), Age int);

* **UNIQUE**  constraint ensures that a field or column will only have **unique values.**
  + Constraint when creating a table (Table level)

CREATE TABLE Student(s\_id int NOT NULL UNIQUE, Name varchar(60), Age int);

* + Constraint after creating a table (Column level)

ALTER TABLE Student ADD UNIQUE(s\_id);

* **PRIMARY KEY**  constraint uniquely identifies each record in a database. Unique and not null
  + Constraint at Table level

CREATE TABLE Student(s\_id int PRIMARY KEY, Name varchar(60) NOT NULL, Age int);

* + Constraint at Column level

ALTER TABLE Student ADD PRIMARY KEY (s\_id);

* **FOREIGN KEY**  is used to relate two tables. **FOREIGN KEY** constraint is also used to restrict actions that would destroy links between tables. Let's see its use, with help of the below tables:

|  |  |  |
| --- | --- | --- |
| **c\_id** | **Customer\_Name** | **address** |
| 101 | Adam | Noida |
| 102 | Alex | Delhi |
| 103 | Stuart | Rohtak |

|  |  |  |
| --- | --- | --- |
| **Order\_id** | **Order\_Name** | **c\_id** |
| 10 | Order1 | 101 |
| 11 | Order2 | 103 |
| 12 | Order3 | 102 |

### **Customer\_Detail Table Order\_Detail Table**

*In Customer\_Detail table,* ***c\_id*** *is the primary key which is set as foreign key in Order\_Detail table.*

*If you try to insert any incorrect data, DBMS will return error and will not allow you to insert the data*.

o Using FOREIGN KEY constraint at Table Level

CREATE table Order\_Detail( order\_id int PRIMARY KEY,

order\_name varchar(60) NOT NULL,

c\_id int FOREIGN KEY REFERENCES Customer\_Detail(c\_id)

);

*In this query, c\_id in table Order\_Detail is made as foreign key, which is a reference of c\_id column in*

*Customer\_Detail* table.

o Using FOREIGN KEY constraint at Column Level

ALTER table Order\_Detail ADD FOREIGN KEY (c\_id) REFERENCES Customer\_Detail(c\_id);

* **CHECK**: constraint is used to restrict the value of a column between a range. It performs check on the values, before storing them into the database. It’s like condition checking before saving data into a column.
  + Using **CHECK** constraint at Table Level

CREATE table Student(

s\_id int NOT NULL CHECK(s\_id > 0), Name varchar(60) NOT NULL,

Age int

);

*The above query will restrict the s\_id value to be greater than zero.*

* + Using **CHECK** constraint at Column Level

ALTER table Student ADD CHECK(s\_id > 0);

* **DEFAULT**