|  |  |
| --- | --- |
| **## SQL Study Guide** | |
| **### SQL** | |
| - What is SQL | **Structured Query Language** - standard Database language which is used to create, maintain and retrieve the relational database. |
| - What are the SQL sublanguages;  - What are the Key Statements for the Sub Languages | **DDL – Data Definition Language -** defines the database schema. Create and modify the structure of database objects in the database.  CREATE, ALTER, DROP and TRUNCATE  **DML – Data Manipulation Language -** deals with the manipulation of data present in the database.  INSERT, (SELECT\*), UPDATE and DELETE  **DQL\* - Data Query Language -** used for performing queries on the data within schema objects. Records retrieved from a Select statement are known as the ‘Result Set’.  SELECT  **DCL – Data Control Language -** deals with the rights, permissions and other controls of the database system.  GRANT and REVOKE  **TCL – Transaction Control Language -** deals with the transaction within the database.  COMMIT, ROLLBACK, SAVEPOINT, RELEASE, SET  TRANSACTION. |
| - What is multiplicity | **Multiplicity** establishes the bounds for data. |
| - What is cardinality | **Cardinality** .A description of the numerical relationship between two tables.   1. One to One relationship 2. One to Many (Many to One) relationship 3. Many to Many relationship |
| - What is a Primary Key | A **primary key** is a column of table which uniquely identifies each tuple (row) in that table. Primary key enforces integrity constraints to the table. Only one primary key is allowed to use in a table. The primary key does not accept the any duplicate and NULL values. The primary key value in a table changes very rarely so it is chosen with care where the changes can occur in a seldom manner. A primary key of one table can be referenced by foreign key of another table. |
| - What is a Foreign Key | A **foreign key** is a column or group of columns in a relational database table that provides a link between data in two tables. It is a column (or columns) that references a column (most often the primary key) of another table. |
| - What is referential integrity | is a relational database concept, which states that table relationships must always be consistent. The consistency and accuracy of data within a table as it relates to other information within an RDBMS.  Referential integrity is maintained using constraints, particularly foreign keys which reference primary keys. |
| - What are the different constraints | Integrity constraints are mainly enforced to maintain the data consistency in database as they restrict the data that can go into the table. These integrity constraints are categorized into two categories **column level** and **table level** constraints. Mainly used constraints are:  **not null constraint**: Ensures that a column cannot have NULL value.  **Default**: Provides a default value for a column when none is specified.  **Unique**: Ensures that all values in a column are different.  **Primary Key**: Uniquely identified each rows/records in a database table.  **Foreign Key**: Uniquely identified a rows/records in any another database table(fetch the value from the column of another table that is the primary key in that table).  **Check constraint**: The CHECK constraint ensures that all values in a column satisfy certain conditions that are mentioned in the check condition. |
| - What do you call a record with a foreign key relationship that doesn’t exist | Orphaned records |
| - What are the differences between WHERE vs HAVING | The **WHERE clause** works on row’s data, not on aggregated data.    The **HAVING clause** works on aggregated data. |
| - what are the differences between GROUP BY and ORDER BY | **GROUP BY statement** is used to group the rows that have the same value. It is often used with aggregate functions.  **ORDER BY keyword** sort the result-set either in ascending or in descending order. |
| - What is the difference between an aggregate function and a scalar function? | **Aggregate function** - Functions that are used to do operations from the values of the column and a single value is returned.  **Scalar Function** - Functions that are based on user input that return a single value each time it’s invoked.  **Window Function** - Window functions applies aggregate and ranking functions over a particular window (set of rows). |
| - Name some scalar functions and their usage | UCASE() - It converts the value of a field to uppercase  LCASE() - It converts the value of a field to uppercase  MID() - extracts texts from the text field.  LEN() - returns the length of the value in a text field.  ROUND() - used to round a numeric field to the number of decimals specified.  NOW() - the current system date and time  FORMAT() - used to format how a field is to be displayed. |
| - Name some aggregate functions and their usage | AVG() - returns average value after calculating from values in a numeric column  COUNT() - used to count the number of rows returned in a SELECT statement.  FIRST() - returns the first value of the selected column  LAST() - returns the last value of the selected column.  MAX() - returns the maximum value of the selected column.  MIN() - returns the minimum value of the selected column  SUM() - returns the sum of all the values of the selected column. |
| - What does LIKE do? | It is useful when you want to search rows to match a specific pattern, or when you do not know the entire value. Used in Where clauses. |
| - How do I use sub queries? | Query within another query. In other words we can say that a Subquery is a query that is embedded in WHERE clause of another SQL query. |
| - How does BETWEEN work? | The **BETWEEN** and **AND** operators are used in WHERE clauses to compare data for a range of values. |
| - What is the order of operations in an SQL statement? | 1. FROM 2. WHERE 3. GROUP BY 4. HAVING 5. SELECT 6. ORDER BY |
| Equi Joins vs Non-Equi Joins | **Equi Joins -** sql join condition which uses the equal sign as the comparison operator. Two types of equi joins are SQL Outer join and SQL Inner join.  **Non-Equi Joins -** It is a sql join condition which makes use of some comparison operator other than the equal sign like >, <, >=, <= |
| - What are the different joins in SQL? | **INNER JOIN - (Natural Join) -** All the rows returned by the sql query satisfy the sql join condition specified. Selects all rows from both the tables as long as the condition satisfies.  **OUTER JOIN** - This sql join returns all rows from both tables which satisfy the join condition along with rows which do not satisfy the join condition from one of the tables. Full, Left and Right joins are all considered types of outer joins  **LEFT JOIN -**This join returns all the rows of the table on the left side of the join and matching rows for the table on the right side of join.  **RIGHT JOIN -** This join returns all the rows of the table on the right side of the join and matching rows for the table on the left side of join.  **FULL JOIN -** creates the result-set by combining result of both LEFT JOIN and RIGHT JOIN. The result-set will contain all the rows from both the tables.  **SELF JOIN** - a table is joined to itself. That is, each row of the table is joined with itself and all other rows depending on some conditions. In other words we can say that it is a join between two copies of the same table.  **CROSS JOIN/CARTESIAN JOIN** - a join for each row of one table to every row of another table. This usually happens when the matching column or WHERE condition is not specified. |
| - What are the different set operations in SQL? | **Set Operations** eliminate duplicate tuples and can be applied only to the relations which are union compatible. Set Operations available in SQL are :  **Set Union -** This operation includes all the tuples which are present in either of the relations.  **Set Intersection -** This operation includes the tuples which are present in both of the relations.  -- con’t --  **Set Difference -** This operation includes the tuples which are present one relation but should not be present in other relation. |
| - Difference between union and union all | **UNION:**  1) It removes duplicate values in the data  2) As it removes duplicates so it is usually slower than UNION ALL  3) It performs less than UNION ALL  4) It is sorted.  **UNION ALL:**  1) It does not remove duplicate values from the data  2) As it does not removes duplicates so it is usually faster.  3) It is more efficient than UNION.  4) It is unsorted |
| - What is the difference between joins and set operations? | **Joins -**  **Set Operations** eliminate duplicate tuples and can be applied only to the relations which are union compatible. |
| - How can I create a alias in SQL | **as** = alias\_name |
| - What does the AS keyword do in a query? | Aliases are the temporary names given to table or column for the purpose of a particular SQL query. It is used when name of column or table is used other than their original names, but the modified name is only temporary. |
| - What are the properties of a transaction? | **Atomicity Consistency Isolation Durability**  **Atomicity** - The entire transaction takes place at once or doesn’t happen at all. The transaction will be aborted and the change reverted.  **Consistency** - The database must be consistent before and after the transaction. No transaction should have any adverse effect on the data residing in the database/table.  **Isolation -** Multiple transactions occur independently without interference.  **Durability** - The changes of a successful transaction occurs even if the system failure occurs. |
| - What are the transaction isolation levels and what do they prevent? | **Read Uncommitted** - Read Uncommitted is the lowest isolation level. In this level, one transaction may read not yet committed changes made by other transaction, thereby allowing dirty reads. In this level, transactions are not isolated from each other.  **Read Committed** - This isolation level guarantees that any data read is committed at the moment it is read. Thus it does not allows dirty read. The transaction holds a read or write lock on the current row, and thus prevent other transactions from reading, updating or deleting it.  **Repeatable Read** - This is the most restrictive isolation level. The transaction holds read locks on all rows it references and writes locks on all rows it inserts, updates, or deletes. Since other transaction cannot read, update or delete these rows, consequently it avoids non-repeatable read.  **Serializable** - Highest Isolation Level. An execution of operations in which concurrently executing transactions appears to be serially or sequentially executing. |
| - What are dirty reads, non repeatable reads, and phantom reads? | **Dirty Read** - when a transaction reads data that has been added by a different transaction that has not yet been committed.  **Non Repeatable Read** - occurs when a transaction reads the same row twice, and get a different value each time. Transactions re-read data that it has previously read, and finds another committed transaction as modified or deleted data.  **Phantom Read** - occurs when two same queries are executed, but the rows retrieved by the two, are different. Transaction returns a query to find that the number of records that satisfies a certain condition has changed. |
| - What is normalization | Normalization is the method used to eliminate or reduce redundancy in database tables. |
| - What are the requirements for the different normalization levels | **0NF** - Information is just put into a DB with no meaning or planning. No normalization.  **1NF** - Every attribute in that relation is singled valued attribute. All data must be atomic (smallest possible data possible while maintaining all the information). Each record should be unique. All tables should have a primary key  **2NF** - relation must be in first normal form and relation must not contain any partial dependency (If the proper subset of candidate key determines non-prime attribute, it is called partial dependency.)  **3NF** - if there is no transitive dependency for non-prime attributes as well as it is in second normal form. No columns are dependent on a column that is not the primary key. |
| - What is pl/pgsql | Procedural Language/SQL - PL/SQL is a block structured language that enables developers to combine the power of SQL with procedural statements.All the statements of a block are passed to oracle engine all at once which increases processing speed and decreases the traffic.  **Disadvantages of SQL**:  SQL doesn’t provide the programmers with a technique of condition checking, looping and branching.  SQL statements are passed to Oracle engine one at a time which increases traffic and decreases speed.  SQL has no facility of error checking during manipulation of data.  **Features of PL/SQL**:  PL/SQL is basically a procedural language, which provides the functionality of decision making, iteration and many more features of procedural programming languages.  PL/SQL can execute a number of queries in one block using single command.  One can create a PL/SQL unit such as procedures, functions, packages, triggers, and types, which are stored in the database for reuse by applications.  PL/SQL provides a feature to handle the exception which occurs in PL/SQL block known as exception handling block.  Applications written in PL/SQL are portable to computer hardware or operating system where Oracle is operational.  PL/SQL Offers extensive error checking. |
| - What are triggers | Are stored procedure in database which automatically invokes whenever a special event in the database occurs. |
| **### MAVEN** | |
| - What is Maven? | Yiddish word meaning “Accumulation of knowledge”. Maven is a tool that can be used for building and managing Java projects.  Maven automates the process of importing files into your build path/project structure. Uses a large online library of projects that can downloaded and used. |
| - What are the Maven Lifecycles? | There are 3 Maven lifecycles:  **Cleaning** – Used to remove excess files & reduce clutter  **Default** - Usually when we talk about the Maven lifecycles, we are usually talking about Default.   1. **Validate:** This step validates if the project structure is correct. For example – It checks if all the dependencies have been downloaded and are available in the local repository. 2. **Compile**: It compiles the source code, converts the .java files to .class and stores the classes in target/classes folder. 3. **Test**: It runs unit tests for the project. 4. **Package**: This step packages the compiled code in distributable format like JAR or WAR. 5. **Integration test**: It runs the integration tests for the project. 6. **Verify**: This step runs checks to verify that the project is valid and meets the quality standards. 7. **Install:** This step installs the packaged code to the local Maven repository. 8. **Deploy:** It copies the packaged code to the remote repository for sharing it with other developers.   **Site** - Documentation of project. |
| - What is the purpose of the POM.xml? | **Project Object Model** - a singleton configuration file that contains most of the information that is required to build a project in just the way you want. |
| **### Design Patterns** | |
| - What is the difference between Singleton and Factory? | **Singleton** - software design pattern that ensures there will be one single instance of that class created throughout the lifetime of our application. Creation of a singleton object can occur either:  **Lazily** – only when it is first needed  **Eagerly** – available when the class is loaded into memory.  **Factory** - allows for the creation of objects without exposing the actual creation logic to the client/users and allows for the created objects to be referenced using a common interface. |
| **### JDBC** | |
| - What is JDBC | Java Database Connectivity - API used to allow Java applications to interact with SQL databases. It acts as a middle layer interface between java applications and database. Located in the java.sql package. |
| - What are the interfaces and classes of JDBC | **DriverManager [class]** – Provides instance of specific drivers (uses a factory design pattern)  **Driver <interface>** - Interfacing entity that speak to SQL  **Connection <interface>** - Connection between the application in java and the database  **Statement <interface>** - Responsible for executing actions on the database  **ResultSet <interface>** - Responsible for transporting query results from the database  **SQLExcception [class]** – Provides information regarding mistakes accessing the database. |
| - What are the different types of statements | **Simple Statement** - Literal interpretation of SQL strings which are compiled and executed on the database side.  **Prepared Statement** - when you want to use SQL statements many times. The PreparedStatement interface accepts input parameters at runtime.  **Callable Statement** - It is used when you want to use the database stored procedures. CallableStatement can accept runtime input parameters. |
| - What is SQL Injection | The inclusion of unintended SQL statements sent to a database through input streams. SQL injection is a technique used to exploit user data through web page inputs by injecting SQL commands as statements. Basically, these statements can be used to manipulate the application’s web server by malicious users.  SQL injection is a code injection technique that might destroy your database.  SQL injection is one of the most common web hacking techniques.  SQL injection is the placement of malicious code in SQL statements, via web page input. |
| - What is a DAO | **Data Access Object Pattern** - A Design pattern by which we use java objects in order to access tables in SQL databases. |
| What is an ERD? | **Entity Relationship Diagram** - used to visually describe the relationships between tables within a database, as well as the structure of the tables. |
|  |  |