SQL Server

1. What is RDBMS?

Relational Data Base Management Systems (RDBMS) are database management systems that maintain data records and indices in tables. Relationships may be created and maintained across and among the data and tables. In a relational database, relationships between data items are expressed by means of tables. Interdependencies among these tables are expressed by data values rather than by pointers. This allows a high degree of data independence. An RDBMS has the capability to recombine the data items from different files, providing powerful tools for data usage.

2. What are the properties of the Relational tables?

Relational tables have six properties:

1. Values are atomic.
2. Column values are of the same kind.
3. Each row is unique.
4. The sequence of columns is insignificant.
5. The sequence of rows is insignificant.
6. Each column must have a unique name.

3. What is Normalization?

Database normalization is a data design and organization process applied to data structures based on rules that help building relational databases. In relational database design, the process of organizing data to minimize redundancy is called normalization. Normalization usually involves dividing a database into two or more tables and defining relationships between the tables. The objective is to isolate data so that additions, deletions, and modifications of a field can be made in just one table and then propagated through the rest of the database via the defined relationships.

4. What is De-normalization?

De-normalization is the process of attempting to optimize the performance of a database by adding redundant data. It is sometimes necessary because current DBMSs implement the relational model poorly. A true relational DBMS would allow for a fully normalized database at the logical level, while providing physical storage of data that is tuned for high performance. De-normalization is a technique to move from higher to lower normal forms of database modeling in order to speed up database access.

5. What are different normalization forms?

1. **1NF: Eliminate Repeating Groups** Make a separate table for each set of related attributes, and give each table a primary key. Each field contains at most one value from its attribute domain.
2. **2NF: Eliminate Redundant Data** If an attribute depends on only part of a multi-valued key, remove it to a separate table.
3. **3NF: Eliminate Columns Not Dependent On Key** If attributes do not contribute to a description of the key, remove them to a separate table. All attributes must be directly dependent on the primary key.
4. **BCNF: Boyce-Codd Normal Form** If there are non-trivial dependencies between candidate key attributes, separate them out into distinct tables.
5. **4NF: Isolate Independent Multiple Relationships** No table may contain two or more 1:n or n:m relationships that are not directly related.

6. **5NF: Isolate Semantically Related Multiple Relationships** There may be practical constrains on information that justify separating logically related many-to-many relationships.
7. **ONF: Optimal Normal Form** A model limited to only simple (elemental) facts, as expressed in Object Role Model notation.
8. **DKNF: Domain-Key Normal Form** A model free from all modification anomalies is said to be in DKNF.

Remember, these normalization guidelines are cumulative. For a database to be in 3NF, it must first fulfill all the criteria of a 2NF and 1NF database.

6. What is Stored Procedure?

A stored procedure is a named group of SQL statements that have been previously created and stored in the server database. Stored procedures accept input parameters so that a single procedure can be used over the network by several clients using different input data. And when the procedure is modified, all clients automatically get the new version. Stored procedures reduce network traffic and improve performance. Stored procedures can be used to help ensure the integrity of the database.

7. What is Trigger?

A trigger is a SQL procedure that initiates an action when an event (INSERT, DELETE or UPDATE) occurs. Triggers are stored in and managed by the DBMS. Triggers are used to maintain the referential integrity of data by changing the data in a systematic fashion. A trigger cannot be called or executed; DBMS automatically fires the trigger as a result of a data modification to the associated table. Triggers can be viewed as similar to stored procedures in that both consist of procedural logic that is stored at the database level. Stored procedures, however, are not event-drive and are not attached to a specific table as triggers are. Stored procedures are explicitly executed by invoking a CALL to the procedure while triggers are implicitly executed. In addition, triggers can also execute stored procedures.

8. What is Nested Trigger?

A trigger can also contain INSERT, UPDATE and DELETE logic within itself, so when the trigger is fired because of data modification it can also cause another data modification, thereby firing another trigger. A trigger that contains data modification logic within itself is called a nested trigger.

9. What is View?

A simple view can be thought of as a subset of a table. It can be used for retrieving data, as well as updating or deleting rows. Rows updated or deleted in the view are updated or deleted in the table the view was created with. It should also be noted that as data in the original table changes, so does data in the view, as views are the way to look at part of the original table. The results of using a view are not permanently stored in the database. The data accessed through a view is actually constructed using standard T-SQL select command and can come from one to many different base tables or even other views.

10. What is Index?

An index is a physical structure containing pointers to the data. Indices are created in an existing table to locate rows more quickly and efficiently. It is possible to create an index on one or more columns of a table, and each index is given a name. The users cannot see the indexes; they are just used to speed up queries. Effective indexes are one of the best ways to improve performance in a database application. A table scan happens when there is no index available to help a query. In a table scan SQL Server

examines every row in the table to satisfy the query results. Table scans are sometimes unavoidable, but on large tables, scans have a terrific impact on performance.

11. What is a Linked Server?

Linked Servers is a concept in SQL Server by which we can add other SQL Server to a Group and query both the SQL Server dbs using T-SQL Statements. With a linked server, you can create very clean, easy to follow, SQL statements that allow remote data to be retrieved, joined and combined with local data. Stored Procedure sp_addlinkedserver, sp_addlinkedsrvlogin will be used add new Linked Server.

12. What is Cursor?

Cursor is a database object used by applications to manipulate data in a set on a row-by- row basis, instead of the typical SQL commands that operate on all the rows in the set at one time.

In order to work with a cursor we need to perform some steps in the following order:

1. Declare cursor
2. Open cursor
3. Fetch row from the cursor
4. Process fetched row
5. Close cursor
6. Deallocate cursor

13. What is Collation?

Collation refers to a set of rules that determine how data is sorted and compared. Character data is sorted using rules that define the correct character sequence, with options for specifying case sensitivity, accent marks, kana character types and character width.

14. What is Difference between Function and Stored Procedure?

UDF can be used in the SQL statements anywhere in the WHERE/HAVING/SELECT section where as Stored procedures cannot be. UDFs that return tables can be treated as another rowset. This can be used in JOINs with other tables. Inline UDF's can be thought of as views that take parameters and can be used in JOINs and other Rowset operations.

15. What is sub-query? Explain properties of sub-query?

Sub-queries are often referred to as sub-selects, as they allow a SELECT statement to be executed arbitrarily within the body of another SQL statement. A sub-query is executed by enclosing it in a set of parentheses. Sub-queries are generally used to return a single row as an atomic value, though they may be used to compare values against multiple rows with the IN keyword.

A subquery is a SELECT statement that is nested within another T-SQL statement. A subquery SELECT statement if executed independently of the T-SQL statement, in which it is nested, will return a resultset. Meaning a subquery SELECT statement can standalone and is not depended on the statement in which it is nested. A subquery SELECT statement can return any number of values, and can be found in, the column list of a SELECT statement, a FROM, GROUP BY, HAVING, and/or ORDER BY clauses of a T-SQL statement. A Subquery can also be used as a parameter to a function call. Basically a subquery can be used anywhere an expression can be used.

16. What are different Types of Join?

1. **Cross Join** A cross join that does not have a WHERE clause produces the Cartesian product of the tables involved in the join. The size of a Cartesian product result set is the number of rows in the first table multiplied by the number of rows in the second table. The common example is when company wants to combine each product with a pricing table to analyze each product at each price.
2. **Inner Join** A join that displays only the rows that have a match in both joined tables is known as inner Join. This is the default type of join in the Query and View Designer.
3. **Outer Join** A join that includes rows even if they do not have related rows in the joined table is an Outer Join. You can create three different outer join to specify the unmatched rows to be included:
    1. **Left Outer Join:** In Left Outer Join all rows in the first-named table i.e. "left" table, which appears leftmost in the JOIN clause are included. Unmatched rows in the right table do not appear.
    2. **Right Outer Join:** In Right Outer Join all rows in the second-named table i.e. "right" table, which appears rightmost in the JOIN clause are included. Unmatched rows in the left table are not included.
    3. **Full Outer Join:** In Full Outer Join all rows in all joined tables are included, whether they are matched or not.
4. **Self Join** This is a particular case when one table joins to itself, with one or two aliases to avoid confusion. A self join can be of any type, as long as the joined tables are the same. A self join is rather unique in that it involves a relationship with only one table. The common example is when company has a hierarchal reporting structure whereby one member of staff reports to another. Self Join can be Outer Join or Inner Join.

17. What are primary keys and foreign keys?

Primary keys are the unique identifiers for each row. They must contain unique values and cannot be null. Due to their importance in relational databases, Primary keys are the most fundamental of all keys and constraints. A table can have only one Primary key. Foreign keys are both a method of ensuring data integrity and a manifestation of the relationship between tables.

18. What is User Defined Functions? What kind of User-Defined Functions can be created?

User-Defined Functions allow defining its own T-SQL functions that can accept 0 or more parameters and return a single scalar data value or a table data type.
Different Kinds of User-Defined Functions created are:

1. **Scalar User-Defined Function** A Scalar user-defined function returns one of the scalar data types. Text, ntext, image and timestamp data types are not supported. These are the type of user-defined functions that most developers are used to in other programming languages. You pass in 0 to many parameters and you get a return value.
2. **Inline Table-Value User-Defined Function** An Inline Table-Value user-defined function returns a table data type and is an exceptional alternative to a view as the user-defined function can pass parameters into a T-SQL select command and in essence provide us with a parameterized, non-updateable view of the underlying tables.
3. **Multi-statement Table-Value User-Defined Function** A Multi-Statement Table-Value user-defined function returns a table and is also an exceptional alternative to a view as the function can support multiple T-SQL statements to build the final result where the view is limited to a single SELECT statement. Also, the ability to pass parameters into a TSQL select command or a group of them gives us the capability to in essence create a parameterized, non-updateable view of the data in the underlying tables. Within the create function command you must define the table structure that is being returned. After creating this type of user-defined function, It can be used in the FROM clause of a T-SQL command unlike the behavior found when using a stored procedure which can also return record sets.

19. What is Identity?

Identity (or AutoNumber) is a column that automatically generates numeric values. A start and increment value can be set, but most DBA leave these at 1. A GUID column also generates numbers; the value of this cannot be controlled. Identity/GUID columns do not need to be indexed.

20. What is DataWarehousing?

1. Subject-oriented, meaning that the data in the database is organized so that all the data elements relating to the same real-world event or object are linked together;
2. Time-variant, meaning that the changes to the data in the database are tracked and recorded so that reports can be produced showing changes over time;
3. Non-volatile, meaning that data in the database is never over-written or deleted, once committed, the data is static, read-only, but retained for future reporting.
4. Integrated, meaning that the database contains data from most or all of an organization's operational applications, and that this data is made consistent.