**Interview Questions of Different Software Companies**

=> Joins and its types as well

**JOIN is an SQL clause used to query and access data from multiple tables, based on logical relationships between those tables.**

In other words, JOINS indicate how SQL Server should use data from one table to select the rows from another table.

Different types of JOINS in SQL Server

* INNER JOIN
* OUTER JOIN
  + [LEFT OUTER JOIN](https://www.devart.com/dbforge/sql/sqlcomplete/sql-join-statements.html#sql-left-outer-join)
  + [RIGHT OUTER JOIN](https://www.devart.com/dbforge/sql/sqlcomplete/sql-join-statements.html#sql-right-outer-join)
* [SELF JOIN](https://www.devart.com/dbforge/sql/sqlcomplete/sql-join-statements.html#sql-self-join)
* [CROSS JOIN](https://www.devart.com/dbforge/sql/sqlcomplete/sql-join-statements.html#sql-cross-join)
* **SQL INNER JOIN** creates a result table by combining rows that have matching values in two or more tables.
* **SQL LEFT OUTER JOIN** includes in a result table unmatched rows from the table that is specified before the LEFT OUTER JOIN clause.
* **SQL RIGHT OUTER JOIN** creates a result table and includes into it all the records from the right table and only matching rows from the left table.
* **SQL SELF JOIN** joins the table to itself and allows comparing rows within the same table.
* **SQL CROSS JOIN** creates a result table containing paired combination of each row of the first table with each row of the second table.

------------------------------------------------------------------------------------------------------------------------------------------

=> index’s their advantages and disadvantages

An index is a schema object that contains keys from various columns of our data table.

**B-Trees** are used to store the index keys. It helps us find the rows or the values to be searched faster.

They have various advantages like

* increased performance in searching for records,
* sorting records,
* grouping records
* maintaining a unique column

Some of the disadvantages include

* increased disk space
* slower data modification
* updating records in the clustered index

=>WHY INDEXING IS USED? benfit of indexing

An index is a schema object that contains an entry for each value that appears in the indexed column(s) of the table or cluster and provides direct, fast access to rows.  
Indexes allow the database application to find data fast; without reading the whole table.

The users cannot see the indexes; they are just used to speed up searches/queries.

**Syntax:**  
CREATE INDEX index\_name

ON table\_name (column\_name)  
  
Note: The syntax for creating indexes varies amongst different databases. Therefore: Check the syntax for creating indexes in your database.

**Example:**  
CREATE INDEX IXuser\_lilink\_item\_id

ON user\_line\_item\_link (line\_item\_id);  
  
The above example is to create an Index on user\_line\_item\_link table to tune the performance of the database when user is accessing large number of Items in a RFx.

=>at which column index should be applied?

In general, you should create an index on a column in any of the following situations:

* The column is queried frequently.
* A referential integrity constraint exists on the column.
* A UNIQUE key integrity constraint exists on the column

-----------------------------------------------------------------------------------------------------------------------------------------

=> diff b/w trigger and stored procedure

**Procedures:** A procedure is a combination of SQL statements written to perform specified tasks.

**Triggers:** A trigger is a special kind of procedure that executes only when some triggering event such as INSERT, UPDATE, or DELETE operations occur in a table.

| S. No. | Parameters | Triggers | Procedures |
| --- | --- | --- | --- |
| 1. | Basics | A Trigger is implicitly invoked whenever any event such as INSERT, DELETE, or UPDATE occurs in a TABLE. | A Procedure is explicitly called by the user/application using statements or commands such as exec, EXECUTE, or simply procedure name |
| 2. | Action | When an event occurs, a trigger helps to execute an action automatically. | A procedure helps to perform a specified task when it is invoked. |
| 3. | Define/ call | Only nesting of triggers can be achieved in a table. We cannot define/call a trigger inside another trigger. | We can define/call procedures inside another procedure. |
| 4. | Syntax | In a database, the syntax to define a trigger:  CREATE TRIGGER TRIGGER\_NAME | In a database, the syntax to define a procedure:  CREATE PROCEDURE PROCEDURE\_NAME |
| 5. | Transaction statements | Transaction statements such as COMMIT, ROLLBACK, and SAVEPOINT are not allowed in triggers. | All transaction statements such as COMMIT and ROLLBACK are allowed in procedures. |
| 6. | Usage | Triggers are used to maintain referential integrity by keeping a record of activities performed on the table. | Procedures are used to perform tasks defined or specified by the users. |
| 7. | Return value | We cannot return values in a trigger. Also, as an input, we cannot pass values as a parameter. | We can return 0 to n values. However, we can pass values as parameters. |

=>Why Normalization is necessary?

Normalization is necessary **to ensure that the table only contains data directly related to the primary key**, each data field contains only one data element, and to remove redundant (duplicated and unnecessary) data.

*The process of refining the structure of a database to minimize redundancy and improve integrity of database is known as Normalization. When a database has been normalized, it is said to be in normal form.*

=>Can there exist a situation where denormalization is required instead of normalization

NO

=>Can we store a binary tree in db

Yes

=>How many tables we require to store a binary tree in table

Two Tables

For 2 subsets

=>How we can regenerate a binary tree from that table

Get all the data from your db in the form of Parent-Child relation and then store the result set in Binary Tree

Parent Child

=====================

NULL 17265

17265 17270

17265 17394

17270 17796

17270 17797

...

...

...

...

Now using this it becomes easy to plot your Binary Tree (infact any Tree)  
So your function prototype to add nodes will look something like this

public void add(int parent, int child);

=>