**1. Some tips to improve the performance of SQL queries**

1. Instead of writing long queries use either views or stored procedures for minimizing network load.
2. It’s better to introduce constraints instead of triggers. They are more efficient than triggers and can increase performance.
3. Make use of table-level variables instead of temporary tables.
4. The UNION ALL clause responds faster than UNION. It doesn’t look for duplicate rows whereas the UNION statement does that regardless of whether they exist or not.
5. Prevent the usage of DISTINCT and HAVING clauses.
6. Avoid excessive use of SQL cursors.
7. Make use of SET NOCOUNT ON clause while building stored procedures. It represents the rows affected by a T-SQL statement. It would lead to reduced network traffic.
8. It’s a good practice to return the required column instead of all the columns of a table.
9. Create indexes for tables

**2. Bottlenecks that affect the performance of a database**

1. First of all need to identify system performance using CPU utilization (Processors).
2. Low memory is the next most common bottleneck. You can resolve it by expanding the physical RAM, but it won’t solve memory leaks if there is any. In such a case, you need to profile the application to identify the potential leaks within its code.
3. Disk Storage bottleneck –upgrading faster drives--Its impact gets visible while writing large data to the disk. If output operations are very slow, then it is a clear indication an issue becoming the bottleneck.

**3. Steps to Improving the SQL Performances**

1. **Discover –** First of all, find out the areas of improvement. Explore tools like Profiler, Query execution plans, SQL tuning advisor, dynamic views, and custom stored procedures.
2. **Review –** Brainstorm the data available to isolate the main issues.
3. **Propose –** Here is a standard approach one can adapt to boost the performance. However, you can customize it further to maximize the benefits.

1. Identify fields and create indexes.  
2. Modify large queries to make use of indexes created.  
3. Refresh table and views and update statistics.  
4. Reset existing indexes and remove unused ones.  
5. Look for dead blocks and remove them.

1. **Validate –** Test the SQL performance tuning approach. Monitor the progress at a regular interval. Also, track if there is any adverse impact on other parts of the application.
2. **Publish –** Now, it’s time to share the working solution with everyone in the team. Let them know all the best practices so that they can use it with ease.

**4. Difference between Heap table and clustered table? How can we identify if the table is a heap?**

A Heap table is table in which, the data rows are not stored in any particular order within each data pages. In addition, there is no particular order to control the data pages sequence. That is not linked in a linked list. This is due to the fact that the heap table contains no clustered index.

A clustered table is a table that has a predefined clustered index on a column or multiple columns of the table that defines the sorting order of the rows within the data pages and the order of the data pages within the table on the clustered index key.

The heap table can be identified by querying the [**sys.partitions**](https://docs.microsoft.com/en-us/sql/relational-databases/system-catalog-views/sys-partitions-transact-sql%22%20/t%20%22_blank) system object that has one row per each partition with index\_id value equal to 0. You can also query the **sys.indexes** system object also to show the heap table index details, which shows, the id of that index is 0 and the type of it is HEAP.

**5. Index Allocation Map (IAM):**

Sql Server Engine uses an Index Allocation Map (IAM) to keep an entry for each page to track the allocation of these available pages. The IAM is considered as the only logical connection between the data pages that the SQL Server Engine will use to move through the heap.

**6. Forwarding Pointers issue:-**

When a data modification operation is performed on heap table data pages. **Forwarding Pointers** will be inserted into the heap to point to the new location of the moved data. These forwarding pointes will cause the performance issues over time due to visiting the old/original location vs the new location specified by the forwarding pointers to get a specific value.

Starting from SQL Server 2008, a new method was introduced to overcome the forwarding pointers performance issues, by using the ALTER TABLE REBUILD command that will rebuild the heap table.

**7. SQL Server Index:-**

A SQL Server Index is a one of the most important factor in the SQL Server Performance tuning process. Indexes are created to speed up the data retrieval and the querying process operations from a database tables or views, by providing swift access to the database rows, without no need to scan all the table’s data in order to retrieve the requested data.

For example, if we need to search any particular information instead of searching page by page, it’s an easy to identify the index page.