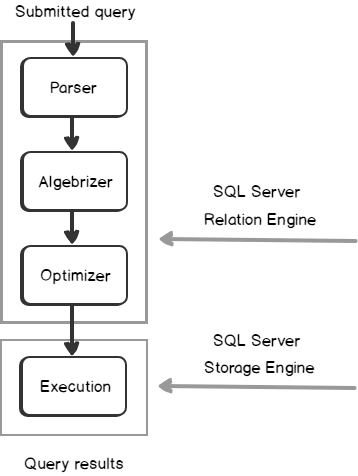
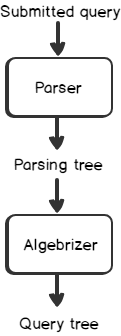
When you submit a T-SQL query, you tell the SQL Server Engine what you want, but without specifying how to do it. Between submitting the T-SQL query to the SQL Server Database Engine and returning the query result to the end user, the SQL Server Engine will perform four internal query processing operations, to convert the query into a format that can be used by the SQL Server Storage Engine easily use to retrieve the requested data.

The first three processes; Parsing, Algebrizing and Optimizing will be performed by the SQL Server Relational Engine. On the other hand, the Execution step is performed by the SQL Server Storage Engine. If the submitted query is not a Data Manipulation Language (DML) statement, such as CREATE TABLE or ALTER TABLE, there will be no need to optimize that query, as there is only one straight way for the SQL Server Engine to perform that action. The four processing steps that are performed by both the SQL Server Relational Engine and SQL Server Storage Engine are summarized as below:



Once the SQL query is submitted, it will be broken into individual parts and checked by the SQL Server Relational Engine to make sure that it is written in a correct way, with no syntax errors for example, under a process that is called **Query Parsing**. The result of the query parsing process is the parsing tree. The parsing tree is an internal representation of the query that includes all the steps, also known as preemptive operations, that are followed to execute the provided query. After creating the parsing tree for the DML queries, the **Algebrizer** will take the generated tree and resolve the names of the different database objects referenced by the submitted query against the system catalog, to make sure that these objects exist in the database and that the user has permission to execute the query. The algebrizer generates a query tree that will be used in the next step, as shown below:



The query processor tree result, from the algebrizer processing, is the input for the **Query Optimizer** for SQL Server Execution Plan. The Query Optimizer generates many execution plan and then selet the most efficient plan to execute the provided query, with the lowest SQL Server resources consumption; this is called the Cost-based Execution Plan. The query optimizer tries different execution methods, reading all table rows, using different indexes, joins and orders and compromising between different resources consumption scenarios, until it ends up with the most optimal execution plan with the least possible cost. The total cost of each candidate execution plan is calculated by assigning a weight for each operation, then summing the cost of all these operations together.

SQL Server **Execution Plan** is a binary representation of the steps that is followed by the SQL Server Engine to execute the query. In other words,. The plan is created based on the query processor tree, resulting from the binding step, and the database tables and indexes statistics, that describes the distribution and uniqueness of the data within the database objects, making sure that the optimization level setting is configured as Full. These statistics help the SQL Server Query Optimizer to compare the number of records returned from scanning all the table rows and the records returned from using different indexes, with the cost of each operation. It is very important to keep the statistics of the database tables and indexes up to date, in order to be able to create the most optimal execution plan.

After choosing the most efficient SQL Serve Execution Plan for the newly submitted query, it will be stored in the **Plan Cache** , which is a part of SQL Server buffer where query plans are stored, for future reuse. This is due to the fact that, generating the most optimal execution plan is an expensive process. When a new query is submitted, the SQL Server Query Optimizer searches in the plan cache for an existing SQL Server Execution Plan for that query to be reused. If it finds no plan that can be reused, the Query creates a new plan, taking more time to execute that query. Reusing execution plans is very beneficial in the case of stored procedures that are executed frequently with different parameters but are still using the same cached execution plan. Having a large number of ad hoc queries running on the database will prevent the execution plans from being reused and require continuous plans generations.

In specific situations, the SQL Server Query Optimizer prefers to create a basic plan, also known as a **Trivial** plan, to execute the queries that have no aggregations or complex calculations, rather than spending time and consuming resources in generating the most efficient plan for that query. After that, the execution plan will be scheduled and used by the SQL Server Storage Engine in the **Execution**stage of the query according to the selected plan, and perform the requested data retrieval, insertion or modification process, as shown below:

