**DBMS FUNCTIONS**

* **Query mechanisms**

**Query Processing**

Query Processing is the activity performed in extracting data from the database. In query processing, it takes various steps for fetching the data from the database. The steps involved are:

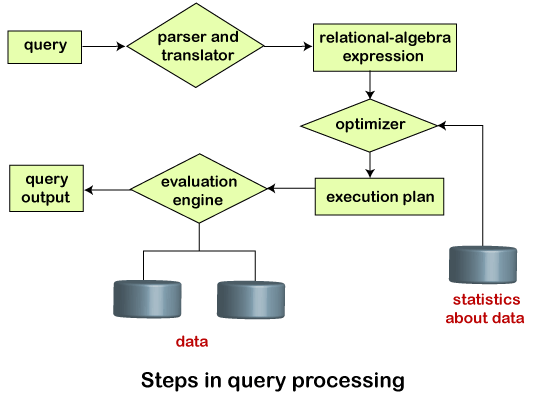
1. Parsing and translation
2. Optimization
3. Evaluation

## **Parsing and Translation**

A query processing includes certain activities for data retrieval.

* Initially, the given user queries get translated in high-level database languages such as SQL. It gets translated into expressions that can be further used at the physical level of the file system.
* After this, the actual evaluation of the queries and a variety of query -optimizing transformations and takes place.

Thus, before processing a query, a computer system needs to translate the query into a human-readable and understandable language. Consequently, SQL or Structured Query Language is the best suitable choice for humans. But it is not perfectly suitable for the internal representation of the query to the system. Relational algebra is well suited for the internal representation of a query. The translation process in query processing is similar to the parser of a query. When a user executes any query, for generating the internal form of the query, the parser in the system checks the syntax of the query, verifies the name of the relation in the database, the tuple, and finally the required attribute value. The parser creates a tree of the query, known as 'parse-tree.' Further, translate it into the form of relational algebra. With this, it evenly replaces all the use of the views when used in the query.



Suppose a user executes a query. As we have learned that there are various methods of extracting the data from the database. In SQL, a user wants to fetch the records of the employees whose salary is greater than or equal to 10000. For doing this, the following query is undertaken:

**select emp\_name from Employee where salary>10000;**

Thus, to make the system understand the user query, it needs to be translated in the form of relational algebra. We can bring this query in the relational algebra form as:

* **σsalary>10000 (πsalary (Employee))**
* **πsalary (σsalary>10000 (Employee))**

After translating the given query, we can execute each relational algebra operation by using different algorithms. So, in this way, a query processing begins its working.

## **Evaluation**

For this, with addition to the relational algebra translation, it is required to annotate the translated relational algebra expression with the instructions used for specifying and evaluating each operation. Thus, after translating the user query, the system executes a query evaluation plan.

### **Query Evaluation Plan**

* In order to fully evaluate a query, the system needs to construct a query evaluation plan.
* The annotations in the evaluation plan may refer to the algorithms to be used for the particular index or the specific operations.
* Such relational algebra with annotations is referred to as **Evaluation Primitives**. The evaluation primitives carry the instructions needed for the evaluation of the operation.
* Thus, a query evaluation plan defines a sequence of primitive operations used for evaluating a query. The query evaluation plan is also referred to as **the query execution plan**.
* A **query execution engine** is responsible for generating the output of the given query. It takes the query execution plan, executes it, and finally makes the output for the user query.

## **Optimization**

* The cost of the query evaluation can vary for different types of queries. Although the system is responsible for constructing the evaluation plan, the user does need not to write their query efficiently.
* Usually, a database system generates an efficient query evaluation plan, which minimizes its cost. This type of task performed by the database system and is known as Query Optimization.
* For optimizing a query, the query optimizer should have an estimated cost analysis of each operation. It is because the overall operation cost depends on the memory allocations to several operations, execution costs, and so on.

Finally, after selecting an evaluation plan, the system evaluates the query and produces the output of the query.

* **Transaction management**
* The transaction is a set of logically related operation. It contains a group of tasks.
* A transaction is an action or series of actions. It is performed by a single user to perform operations for accessing the contents of the database.

**Example:** Suppose an employee of bank transfers Ksh 800 from X's account to Y's account. This small transaction contains several low-level tasks:

**X's Account**

1. Open\_Account(X)
2. Old\_Balance = X.balance
3. New\_Balance = Old\_Balance - 800
4. X.balance = New\_Balance
5. Close\_Account(X)

**Y's Account**

1. Open\_Account(Y)
2. Old\_Balance = Y.balance
3. New\_Balance = Old\_Balance + 800
4. Y.balance = New\_Balance
5. Close\_Account(Y)

## **Operations of Transaction:**

Following are the main operations of transaction:

**Read(X):** Read operation is used to read the value of X from the database and stores it in a buffer in main memory.

**Write(X):** Write operation is used to write the value back to the database from the buffer.

Let's take an example to debit transaction from an account which consists of following operations:

1. R(X);
2. X = X - 500;
3. W(X);

Let's assume the value of X before starting of the transaction is 4000.

* The first operation reads X's value from database and stores it in a buffer.
* The second operation will decrease the value of X by 500. So buffer will contain 3500.
* The third operation will write the buffer's value to the database. So X's final value will be 3500.

But it may be possible that because of the failure of hardware, software or power, etc. that transaction may fail before finished all the operations in the set.

**For example:** If in the above transaction, the debit transaction fails after executing operation 2 then X's value will remain 4000 in the database which is not acceptable by the bank.

To solve this problem, we have two important operations:

**Commit:** It is used to save the work done permanently.

**Rollback:** It is used to undo the work done.

# Transaction property

The transaction has the four properties. These are used to maintain consistency in a database, before and after the transaction.

## **Property of Transaction**

1. Atomicity
2. Consistency
3. Isolation
4. Durability

# States of Transaction

### **Active state**

* The active state is the first state of every transaction. In this state, the transaction is being executed.
* For example: Insertion or deletion or updating a record is done here. But all the records are still not saved to the database.

### **Partially committed**

* In the partially committed state, a transaction executes its final operation, but the data is still not saved to the database.
* In the total mark calculation example, a final display of the total marks step is executed in this state.

### **Committed**

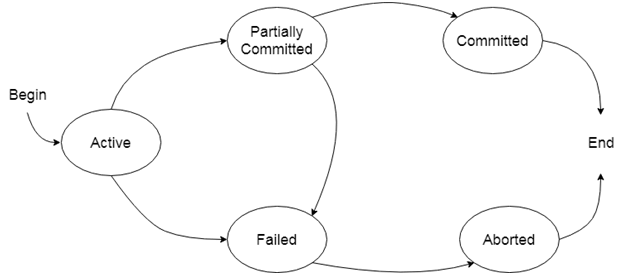
A transaction is said to be in a committed state if it executes all its operations successfully. In this state, all the effects are now permanently saved on the database system.

### **Failed state**

* If any of the checks made by the database recovery system fails, then the transaction is said to be in the failed state.
* In the example of total mark calculation, if the database is not able to fire a query to fetch the marks, then the transaction will fail to execute.

### **Aborted**

* If any of the checks fail and the transaction has reached a failed state then the database recovery system will make sure that the database is in its previous consistent state. If not then it will abort or roll back the transaction to bring the database into a consistent state.
* If the transaction fails in the middle of the transaction then before executing the transaction, all the executed transactions are rolled back to its consistent state.
* After aborting the transaction, the database recovery module will select one of the two operations:
  1. Re-start the transaction
  2. Kill the transaction



* Database Buffer

The goal of a database system is that a minimum number of transfers should take place between the disk and memory. To do so, it can reduce the number of disk accesses by keeping as many blocks in main memory. So, when the user wants to store the data, it can directly search in the main memory, and there will be no requirement of accessing the disk. However, it is difficult to keep so many blocks in main memory; we need to manage the allocation of the space available in the main memory for the storage of blocks.

A database buffer is a temporary storage area in the main memory. It allows storing the data temporarily when moving from one place to another. A database buffer stores a copy of disk blocks. But, the version of block copies on the disk may be older than the version in the buffer.

What is Buffer Manager

* A Buffer Manager is responsible for allocating space to the buffer in order to store data into the buffer.
* If a user requests a particular block and the block is available in the buffer, the buffer manager provides the block address in the main memory.
* If the block is not available in the buffer, the buffer manager allocates the block in the buffer.
* If free space is not available, it throws out some existing blocks from the buffer to allocate the required space for the new block.
* The blocks which are thrown are written back to the disk only if they are recently modified when writing on the disk.
* If the user requests such thrown-out blocks, the buffer manager reads the requested block from the disk to the buffer and then passes the address of the requested block to the user in the main memory.
* However, the internal actions of the buffer manager are not visible to the programs that may create any problem in disk-block requests. The buffer manager is just like a virtual machine.

For serving the database system in the best possible way, the buffer manager uses the following methods:

1. **Buffer Replacement Strategy:** If no space is left in the buffer, it is required to remove an existing block from the buffer before allocating the new one. The various operating system uses the LRU (least recently used) scheme. In LRU, the block that was least recently used is removed from the buffer and written back to the disk. Such type of replacement strategy is known as Buffer Replacement Strategy.
2. **Pinned Blocks:** If the user wants to recover any database system from the crashes, it is essential to restrict the time when a block is written back to the disk. In fact, most recovery systems do not allow the blocks to be written on the disk if the block updation being in progress. Such types of blocks that are not allowed to be written on the disk are known as **pinned blocks**. Luckily, many operating systems do not support the pinned blocks.

* **Forced Output of Blocks:** In some cases, it becomes necessary to write the block back to the disk even though the space occupied by the block in the buffer is not required. When such type of write is required, it is known as the **forced output of a block**. It is because sometimes the data stored on the buffer may get lost in some system crashes, but the data stored on the disk usually does not get affected due to any disk crash.
* **Access methods**