Scan or click here for more resources

UNIT-3

What is SQL?

SQL is a short-form of the structured query language, and it is pronounced as S-Q-L or sometimes as See-Quell.

This database language is mainly designed for maintaining the data in relational database management systems. It is a special tool used by data professionals for handling structured data (data which is stored in the form of tables). It is also designed for stream processing in RDSMS.

You can easily create and manipulate the database, access and modify the table rows and columns, etc. This query language became the standard of ANSI in the year of 1986 and ISO in the year of 1987.

If you want to get a job in the field of data science, then it is the most important query language to learn. Big enterprises like Facebook, Instagram, and LinkedIn, use SQL for storing the data in the back-end.

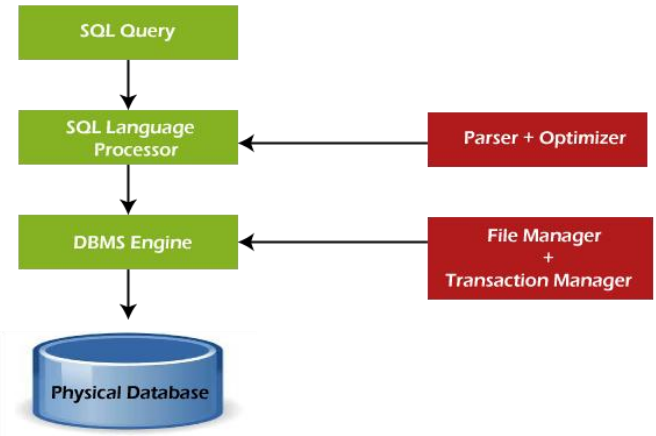
Process of SQL

When we are executing the command of SQL on any Relational database management system, then the system automatically finds the best routine to carry out our request, and the SQL engine determines how to interpret that particular command.

Structured Query Language contains the following four components in its process:

- Query Dispatcher
- Optimization Engines
- Classic Query Engine
- SQL Query Engine, etc.

A classic query engine allows data professionals and users to maintain non-SQL queries. The architecture of SQL is shown in the following diagram:



Advantages of SQL

1. No programming needed
2. High-Speed Query Processing.
3. Standardized Language.
4. Portability
5. Interactive language.
6. More than one Data View

Disadvantages of SQL

With the advantages of SQL, it also has some disadvantages, which are as follows:

1. Cost
2. Interface is Complex
3. Partial Database control

SQL Statement

SQL statements tell the database what operation you want to perform on the structured data and what information you would like to access from the database.

The statements of SQL are very simple and easy to use and understand. They are like plain English but with a particular syntax.

Simple Example of SQL statement:

```
1.  SELECT "column_name" FROM "table_name";
```

Let's discuss each statement in short one by one with syntax and one example:

1. SELECT Statement

This SQL statement reads the data from the SQL database and shows it as the output to the database user.

Syntax of SELECT Statement:

```
1.  SELECT column_name1, column_name2, ...., column_nameN
2.      [ FROM table_name ]
3.      [ WHERE condition ]
4.      [ ORDER BY order_column_name1 [ ASC | DESC ], .... ];
```

Example of SELECT Statement:

```
1.  SELECT Emp_ID, First_Name, Last_Name, Salary, City
2.  FROM Employee_details
3.  WHERE Salary = 100000
```

```
4.      ORDER BY Last_Name
```

This example shows the **Emp_ID, First_Name, Last_Name, Salary, and City** of those employees from the **Employee_details** table whose **Salary** is **100000**. The output shows all the specified details according to the ascending alphabetical order of **Last_Name**.

3. UPDATE Statement

This SQL statement changes or modifies the stored data in the SQL database.

Syntax of UPDATE Statement:

```
1.      UPDATE table_name
2.      SET column_name1 = new_value_1, column_name2 = new_value_2, ...., column_nameN = new_value_N
3.      [ WHERE  CONDITION ];
```

Example of UPDATE Statement:

```
1.      UPDATE Employee_details
2.      SET Salary = 100000
3.      WHERE Emp_ID = 10;
```

This example changes the **Salary** of those employees of the **Employee_details** table whose **Emp_ID** is **10** in the table.

3. DELETE Statement

This SQL statement deletes the stored data from the SQL database.

Syntax of DELETE Statement:

```
1.      DELETE FROM table_name
2.      [ WHERE CONDITION ];
```

Example of DELETE Statement:

```
1.      DELETE FROM Employee_details
2.      WHERE First_Name = 'Sumit';
```

This example deletes the record of those employees from the **Employee_details** table whose **First_Name** is **Sumit** in the table.

4. CREATE TABLE Statement

This SQL statement creates the new table in the SQL database.

Syntax of CREATE TABLE Statement:

```
1.      CREATE TABLE table_name
2.      (
3.      column_name1 data_type [column1 constraint(s)],
4.      column_name2 data_type [column2 constraint(s)],
5.      .....
6.      .....,
7.      column_nameN data_type [columnN constraint(s)],
```

```
8.      PRIMARY KEY(one or more col)
9.      );
```

Example of CREATE TABLE Statement:

```
1.      CREATE TABLE Employee_details(
2.          Emp_Id NUMBER(4) NOT NULL,
3.          First_name VARCHAR(30),
4.          Last_name VARCHAR(30),
5.          Salary Money,
6.          City VARCHAR(30),
7.          PRIMARY KEY (Emp_Id)
8.      );
```

This example creates the table **Employee_details** with five columns or fields in the SQL database. The fields in the table are **Emp_Id, First_Name, Last_Name, Salary,** and **City**. The **Emp_Id** column in the table acts as a **primary key**, which means that the Emp_Id column cannot contain duplicate values and null values.

5. ALTER TABLE Statement

This SQL statement adds, deletes, and modifies the columns of the table in the SQL database.

Syntax of ALTER TABLE Statement:

```
1.      ALTER TABLE table_name ADD column_name datatype[(size)];
```

The above SQL alter statement adds the column with its datatype in the existing database table.

```
1.      ALTER TABLE table_name MODIFY column_name column_datatype[(size)];
```

The above 'SQL alter statement' renames the old column name to the new column name of the existing database table.

```
1.      ALTER TABLE table_name DROP COLUMN column_name;
```

The above SQL alter statement deletes the column of the existing database table.

Example of ALTER TABLE Statement:

```
1.      ALTER TABLE Employee_details
2.      ADD Designation VARCHAR(18);
```

This example adds the new field whose name is **Designation** with size **18** in the **Employee_details** table of the SQL database.

6. DROP TABLE Statement

This SQL statement deletes or removes the table and the structure, views, permissions, and triggers associated with that table.

Syntax of DROP TABLE Statement:

```
1.      DROP TABLE [ IF EXISTS ]
2.      table_name1, table_name2, ....., table_nameN;
```

The above syntax of the drop statement deletes specified tables completely if they exist in the database.

Example of DROP TABLE Statement:

```
1.      DROP TABLE Employee_details;
```

This example drops the **Employee_details** table if it exists in the SQL database. This removes the complete information if available in the table.

7. CREATE DATABASE Statement

This SQL statement creates the new database in the database management system.

Syntax of CREATE DATABASE Statement:

```
1.  CREATE DATABASE database_name;
```

Example of CREATE DATABASE Statement:

```
1.  CREATE DATABASE Company;
```

The above example creates the company database in the system.

8. DROP DATABASE Statement

This SQL statement deletes the existing database with all the data tables and views from the database management system.

Syntax of DROP DATABASE Statement:

```
1.  DROP DATABASE database_name;
```

Example of DROP DATABASE Statement:

```
1.  DROP DATABASE Company;
```

The above example deletes the company database from the system.

9. INSERT INTO Statement

This SQL statement inserts the data or records in the existing table of the SQL database. This statement can easily insert single and multiple records in a single query statement.

Syntax of insert a single record:

```
1.  INSERT INTO table_name
2.  (
3.  column_name1,
4.  column_name2, ....,
5.  column_nameN
6.  )
7.  VALUES
8.  (value_1,
9.  value_2, .....,
10. value_N
11. );
```

Example of insert a single record:

```
1.  INSERT INTO Employee_details
2.  (
3.  Emp_ID,
4.  First_name,
```

```
5.      Last_name,
6.      Salary,
7.      City
8.      )
9.      VALUES
10.     (101,
11.     Akhil,
12.     Sharma,
13.     40000,
14.     Bangalore
15.     );
```

This example inserts **101** in the first column, **Akhil** in the second column, **Sharma** in the third column, **40000** in the fourth column, and **Bangalore** in the last column of the table **Employee_details**.

Syntax of inserting a multiple records in a single query:

```
1.      INSERT INTO table_name
2.      ( column_name1, column_name2, ...., column_nameN)
3.      VALUES (value_1, value_2, ....., value_N), (value_1, value_2, ....., value_N),....;
```

Example of inserting multiple records in a single query:

```
1.      INSERT INTO Employee_details
2.      ( Emp_ID, First_name, Last_name, Salary, City )
3.      VALUES (101, Amit, Gupta, 50000, Mumbai), (101,  John, Aggarwal, 45000, Calcutta), (101, Si
dhu, Arora, 55000, Mumbai);
```

This example inserts the records of three employees in the **Employee_details** table in the single query statement.

10. TRUNCATE TABLE Statement

This SQL statement deletes all the stored records from the table of the SQL database.

Syntax of TRUNCATE TABLE Statement:

```
1.      TRUNCATE TABLE table_name;
```

Example of TRUNCATE TABLE Statement:

```
1.      TRUNCATE TABLE Employee_details;
```

This example deletes the record of all employees from the Employee_details table of the database.

11. DESCRIBE Statement

This SQL statement tells something about the specified table or view in the query.

Syntax of DESCRIBE Statement:

```
1.      DESCRIBE table_name | view_name;
```

Example of DESCRIBE Statement:

```
1.      DESCRIBE Employee_details;
```

This example explains the structure and other details about the **Employee_details** table.

12. DISTINCT Clause

This SQL statement shows the distinct values from the specified columns of the database table. This statement is used with the **SELECT** keyword.

Syntax of DISTINCT Clause:

```
SELECT DISTINCT column_name1, column_name2, ...
FROM table_name;
```

Example of DISTINCT Clause:

```
SELECT DISTINCT City, Salary
FROM Employee_details;
```

This example shows the distinct values of the **City** and **Salary** column from the **Employee_details** table.

13. COMMIT Statement

This SQL statement saves the changes permanently, which are done in the transaction of the SQL database.

Syntax of COMMIT Statement:

```
COMMIT
```

Example of COMMIT Statement:

```
DELETE FROM Employee_details
WHERE salary = 30000;
COMMIT;
```

This example deletes the records of those employees whose **Salary** is **30000** and then saves the changes permanently in the database.

14. ROLLBACK Statement

This SQL statement undo the transactions and operations which are not yet saved to the SQL database.

Syntax of ROLLBACK Statement:

```
ROLLBACK
```

Example of ROLLBACK Statement:

```
DELETE FROM Employee_details
WHERE City = Mumbai;
ROLLBACK;
```

This example deletes the records of those employees whose **City** is **Mumbai** and then undo the changes in the database.

15. CREATE INDEX Statement

This SQL statement creates the new index in the SQL database table.

Syntax of CREATE INDEX Statement:

```
CREATE INDEX index_name
ON table_name ( column_name1, column_name2, ..., column_nameN );
```

Example of CREATE INDEX Statement:

```
CREATE INDEX idx_First_Name
```

```
2.      ON employee_details (First_Name);
```

This example creates an index **idx_First_Name** on the **First_Name** column of the **Employee_details** table.

16. DROP INDEX Statement

This SQL statement deletes the existing index of the SQL database table.

Syntax of DROP INDEX Statement:

```
1.      DROP INDEX index_name;
```

Example of DROP INDEX Statement:

```
1.      DROP INDEX idx_First_Name;
```

This example deletes the index **idx_First_Name** from the SQL database.

17. USE Statement

This SQL statement selects the existing SQL database. Before performing the operations on the database table, you have to select the database from the multiple existing databases.

Syntax of USE Statement:

```
1.      USE database_name;
```

Example of USE DATABASE Statement:

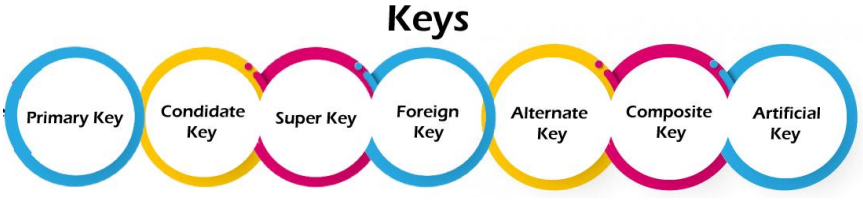
```
1.      USE Company;
```

Keys

- Keys play an important role in the relational database.
- It is used to uniquely identify any record or row of data from the table. It is also used to establish and identify relationships between tables.

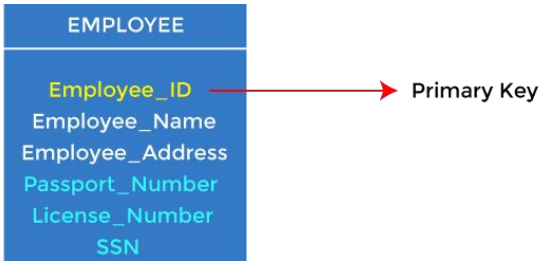
For example, ID is used as a key in the Student table because it is unique for each student. In the PERSON table, passport_number, license_number, SSN are keys since they are unique for each person.

Types of keys:



1. Primary key

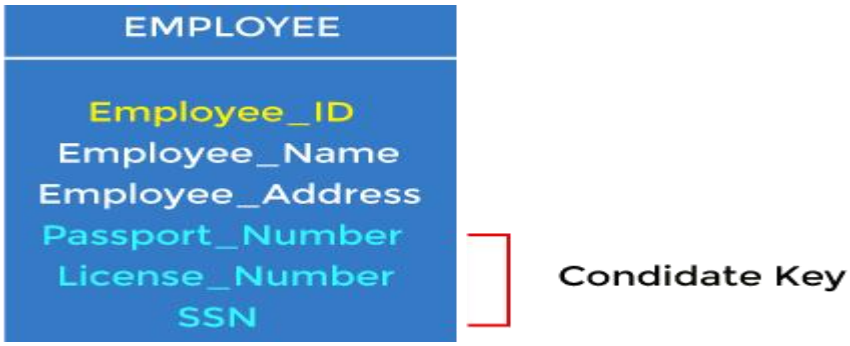
- It is the first key used to identify one and only one instance of an entity uniquely. An entity can contain multiple keys, as we saw in the PERSON table. The key which is most suitable from those lists becomes a primary key.
- In the EMPLOYEE table, ID can be the primary key since it is unique for each employee. In the EMPLOYEE table, we can even select License_Number and Passport_Number as primary keys since they are also unique.
- For each entity, the primary key selection is based on requirements and developers.



2. Candidate key

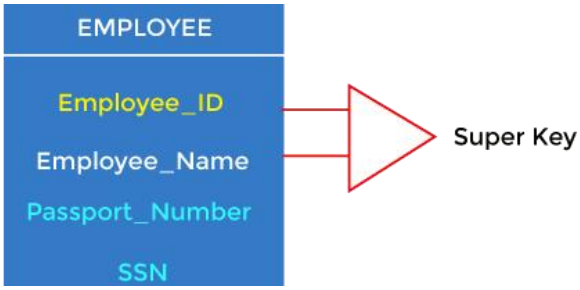
- A candidate key is an attribute or set of attributes that can uniquely identify a tuple.
- Except for the primary key, the remaining attributes are considered a candidate key. The candidate keys are as strong as the primary key.

For example: In the EMPLOYEE table, id is best suited for the primary key. The rest of the attributes, like SSN, Passport_Number, License_Number, etc., are considered a candidate key.



3. Super Key

Super key is an attribute set that can uniquely identify a tuple. A super key is a superset of a candidate key.



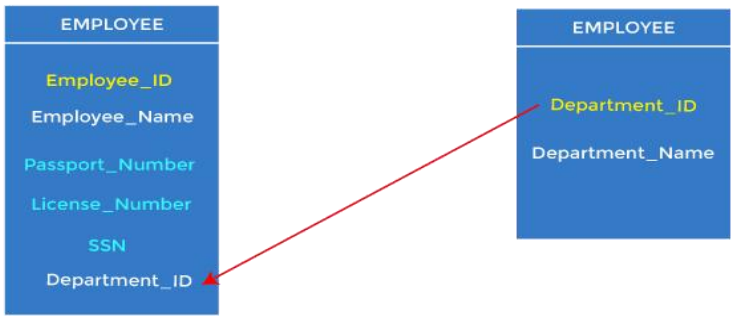
For example: In the above EMPLOYEE table, for(EMPLOEE_ID, EMPLOYEE_NAME), the name of two employees can be the same, but their EMPLYEE_ID can't be the same. Hence, this combination can also be a key.

The super key would be EMPLOYEE-ID (EMPLOYEE_ID, EMPLOYEE-NAME), etc.

4. Foreign key

- Foreign keys are the column of the table used to point to the primary key of another table.

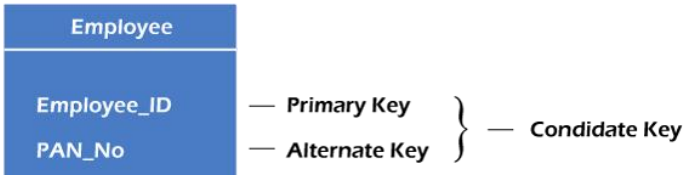
- Every employee works in a specific department in a company, and employee and department are two different entities. So we can't store the department's information in the employee table. That's why we link these two tables through the primary key of one table.
- We add the primary key of the DEPARTMENT table, Department_Id, as a new attribute in the EMPLOYEE table.
- In the EMPLOYEE table, Department_Id is the foreign key, and both the tables are related.



5. Alternate key

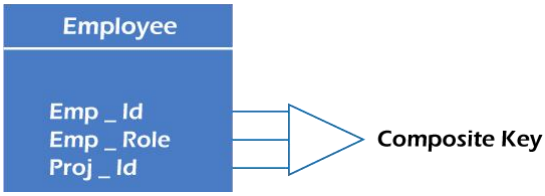
There may be one or more attributes or a combination of attributes that uniquely identify each tuple in a relation. These attributes or combinations of the attributes are called the candidate keys. One key is chosen as the primary key from these candidate keys, and the remaining candidate key, if it exists, is termed the alternate key. **In other words,** the total number of the alternate keys is the total number of candidate keys minus the primary key. The alternate key may or may not exist. If there is only one candidate key in a relation, it does not have an alternate key.

For example, employee relation has two attributes, Employee_Id and PAN_No, that act as candidate keys. In this relation, Employee_Id is chosen as the primary key, so the other candidate key, PAN_No, acts as the Alternate key.

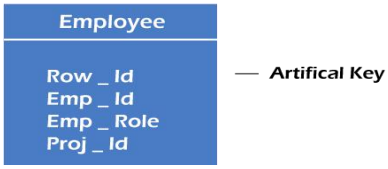


6. Composite key

Whenever a primary key consists of more than one attribute, it is known as a composite key. This key is also known as Concatenated Key.



For example, in employee relations, we assume that an employee may be assigned multiple roles, and an employee may work on multiple projects simultaneously. So the primary key will be composed of all three attributes, namely Emp_ID, Emp_role, and Proj_ID in combination. So these attributes act as a composite key since the primary key comprises more than one attribute.



7. Artificial key

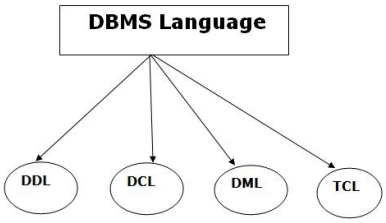
The key created using arbitrarily assigned data are known as artificial keys. These keys are created when a primary key is large and complex and has no relationship with many other relations. The data values of the artificial keys are usually numbered in a serial order.

For example, the primary key, which is composed of Emp_ID, Emp_role, and Proj_ID, is large in employee relations. So it would be better to add a new virtual attribute to identify each tuple in the relation uniquely.

Database Languages in DBMS

- A DBMS has appropriate languages and interfaces to express database queries and updates.
- Database languages can be used to read, store and update the data in the database.

Types of Database Languages



1. Data Definition Language (DDL)

- **DDL** stands for **D**ata **D**efinition **L**anguage. It is used to define database structure or pattern.
- It is used to create schema, tables, indexes, constraints, etc. in the database.
- Using the DDL statements, you can create the skeleton of the database.
- Data definition language is used to store the information of metadata like the number of tables and schemas, their names, indexes, columns in each table, constraints, etc.

Here are some tasks that come under DDL:

- **Create:** It is used to create objects in the database.
- **Alter:** It is used to alter the structure of the database.
- **Drop:** It is used to delete objects from the database.
- **Truncate:** It is used to remove all records from a table.
- **Rename:** It is used to rename an object.
- **Comment:** It is used to comment on the data dictionary.

These commands are used to update the database schema that's why they come under Data definition language.

2. Data Manipulation Language (DML)

DML stands for **D**ata **M**anipulation **L**anguage. It is used for accessing and manipulating data in a database. It handles user requests.

Here are some tasks that come under DML:

- **Select:** It is used to retrieve data from a database.
- **Insert:** It is used to insert data into a table.

- **Update:** It is used to update existing data within a table.
- **Delete:** It is used to delete all records from a table.
- **Merge:** It performs UPSERT operation, i.e., insert or update operations.
- **Call:** It is used to call a structured query language or a Java subprogram.
- **Explain Plan:** It has the parameter of explaining data.
- **Lock Table:** It controls concurrency.

3. Data Control Language (DCL)

- **DCL** stands for **D**ata **C**ontrol **L**anguage. It is used to retrieve the stored or saved data.
- The DCL execution is transactional. It also has rollback parameters.

  (But in Oracle database, the execution of data control language does not have the feature of rolling back.)

Here are some tasks that come under DCL:

- **Grant:** It is used to give user access privileges to a database.
- **Revoke:** It is used to take back permissions from the user.

There are the following operations which have the authorization of Revoke:
CONNECT, INSERT, USAGE, EXECUTE, DELETE, UPDATE and SELECT.

4. Transaction Control Language (TCL)

TCL is used to run the changes made by the DML statement. TCL can be grouped into a logical transaction.
Here are some tasks that come under TCL:

- **Commit:** It is used to save the transaction on the database.
- **Rollback:** It is used to restore the database to original since the last Commit.

SQL | Triggers

Trigger is a statement that a system executes automatically when there is any modification to the database. In a trigger, we first specify when the trigger is to be executed and then the action to be performed when the trigger executes. Triggers are used to specify certain integrity constraints and referential constraints that cannot be specified using the constraint mechanism of SQL.

Example –
Suppose, we are adding a tuple to the 'Donors' table that is some person has donated blood. So, we can design a trigger that will automatically add the value of donated blood to the 'Blood_record' table.

Types of Triggers –
We can define 6 types of triggers for each table:

1. **AFTER INSERT** activated after data is inserted into the table.
2. **AFTER UPDATE:** activated after data in the table is modified.
3. **AFTER DELETE:** activated after data is deleted/removed from the table.
4. **BEFORE INSERT:** activated before data is inserted into the table.

5. **BEFORE UPDATE:** activated before data in the table is modified.
6. **BEFORE DELETE:** activated before data is deleted/removed from the table.

End……