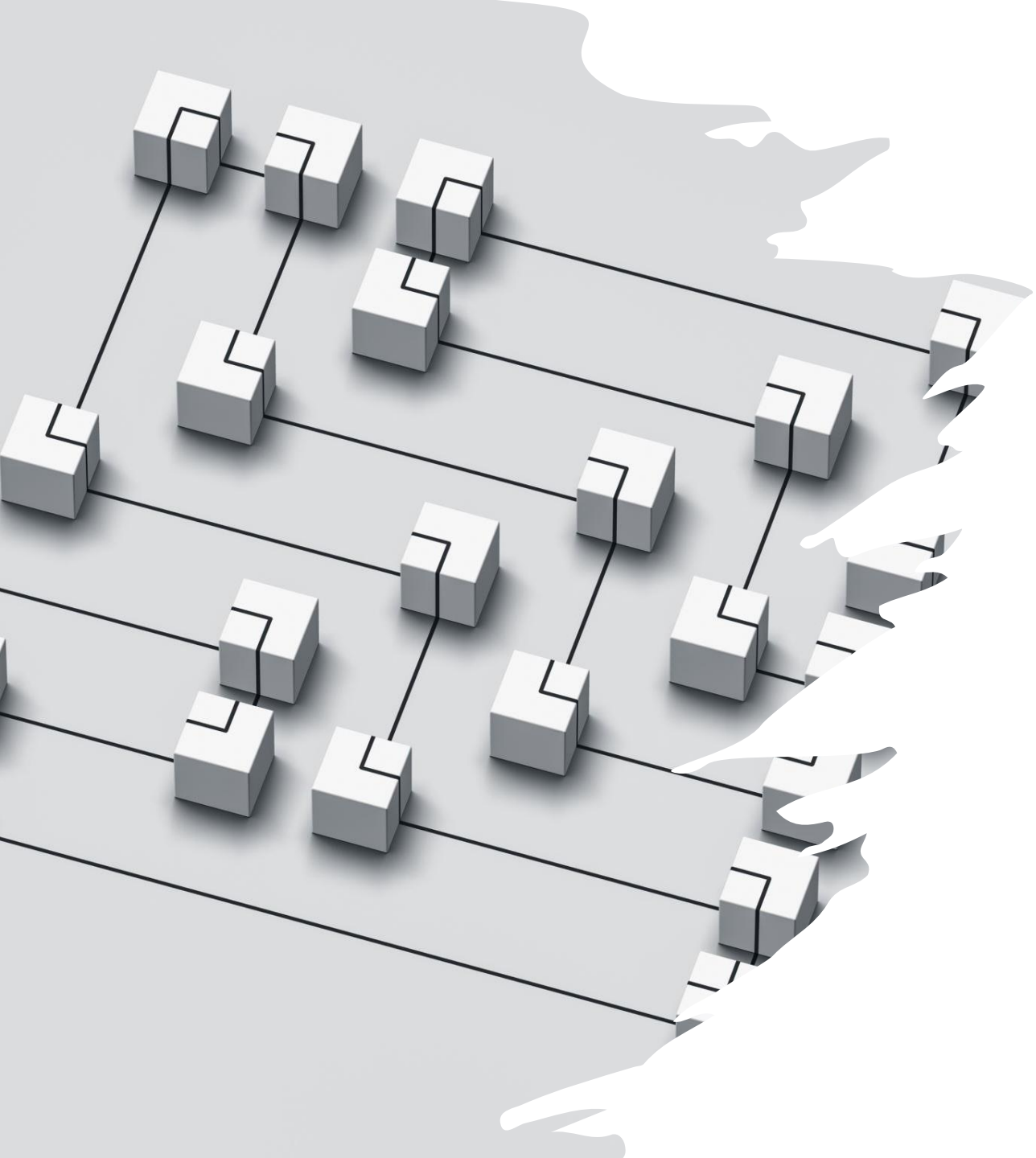
Structured Query Language

EXL



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.

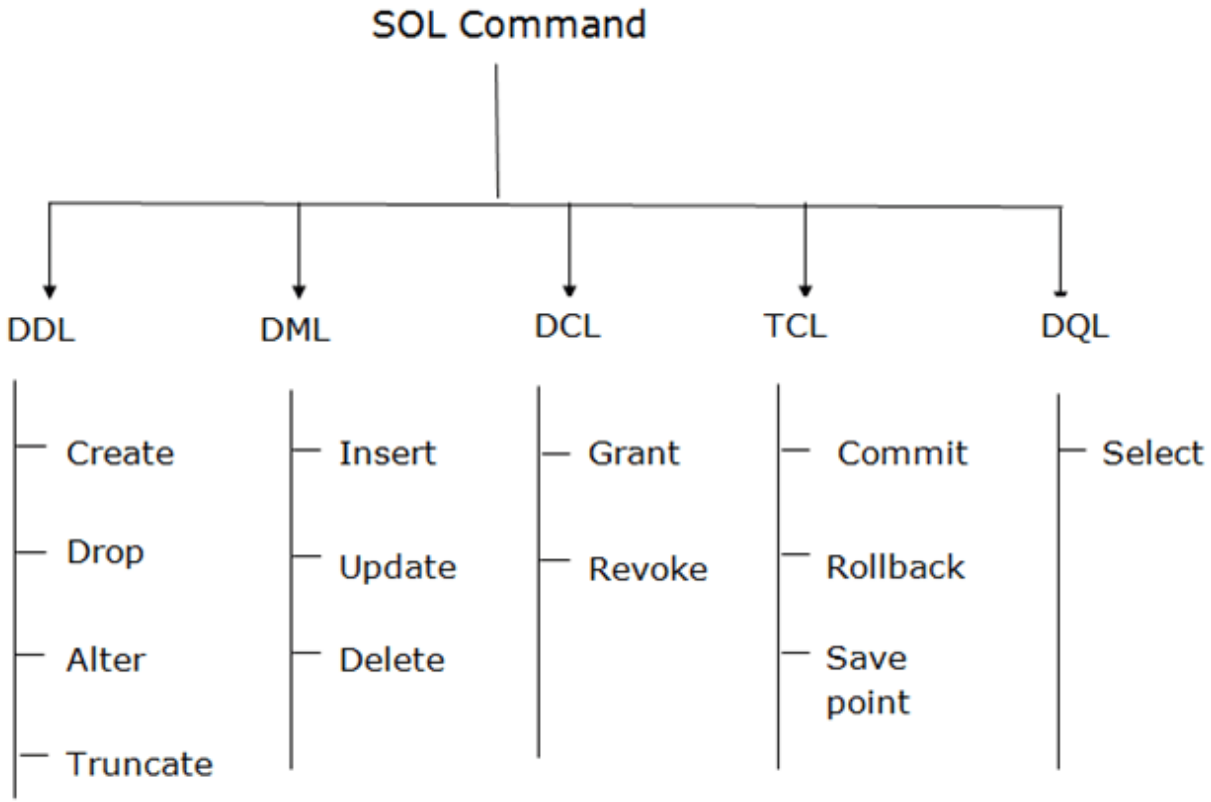
EXL

Why SQL?

- Nowadays, SQL is widely used in data science and analytics. Following are the reasons which explain why it is widely used:
- The basic use of SQL for data professionals and SQL users is to insert, update, and delete the data from the relational database.
- SQL allows the data professionals and users to retrieve the data from the relational database management systems.
- It also helps them to describe the structured data.
- It allows SQL users to create, drop, and manipulate the database and its tables.
- It also helps in creating the view, stored procedure, and functions in the relational database.
- It allows you to define the data and modify that stored data in the relational database.
- It also allows SQL users to set the permissions or constraints on table columns, views, and stored procedures.

EXL

Types of SQL Command



- There are five types of SQL commands: DDL, DML, DCL, TCL, and DQL.

EXL

Some SQL Commands

CREATE Command

- This command helps in creating the new database, new table, table view, and other objects of the database.

UPDATE Command

- This command helps in updating or changing the stored data in the database.

DELETE Command

- This command helps in removing or erasing the saved records from the database tables. It erases single or multiple tuples from the tables of the database.

SELECT Command

- This command helps in accessing the single or multiple rows from one or multiple tables of the database. We can also use this command with the WHERE clause.

DROP Command

- This command helps in deleting the entire table, table view, and other objects from the database.

INSERT Command

- This command helps in inserting the data or records into the database tables. We can easily insert the records in single as well as multiple rows of the table.

EXL

Advantages of SQL

SQL provides various advantages which make it more popular in the field of data science. It is a perfect query language which allows data professionals and users to communicate with the database. Following are the best advantages or benefits of Structured Query Language:

1. No programming needed

- SQL does not require a large number of coding lines for managing the database systems. We can easily access and maintain the database by using simple SQL syntactical rules. These simple rules make the SQL user-friendly.

2. High-Speed Query Processing

- A large amount of data is accessed quickly and efficiently from the database by using SQL queries. Insertion, deletion, and updation operations on data are also performed in less time.

3. Standardized Language

- SQL follows the long-established standards of ISO and ANSI, which offer a uniform platform across the globe to all its users.

4. Portability

- The structured query language can be easily used in desktop computers, laptops, tablets, and even smartphones. It can also be used with other applications according to the user's requirements.

5. Interactive language

- We can easily learn and understand the SQL language. We can also use this language for communicating with the database because it is a simple query language. This language is also used for receiving the answers to complex queries in a few seconds.

6. More than one Data View

- The SQL language also helps in making the multiple views of the database structure for the different database users.

EXL

Disadvantages of SQL

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

1. Cost

- The operation cost of some SQL versions is high. That's why some programmers cannot use the Structured Query Language.

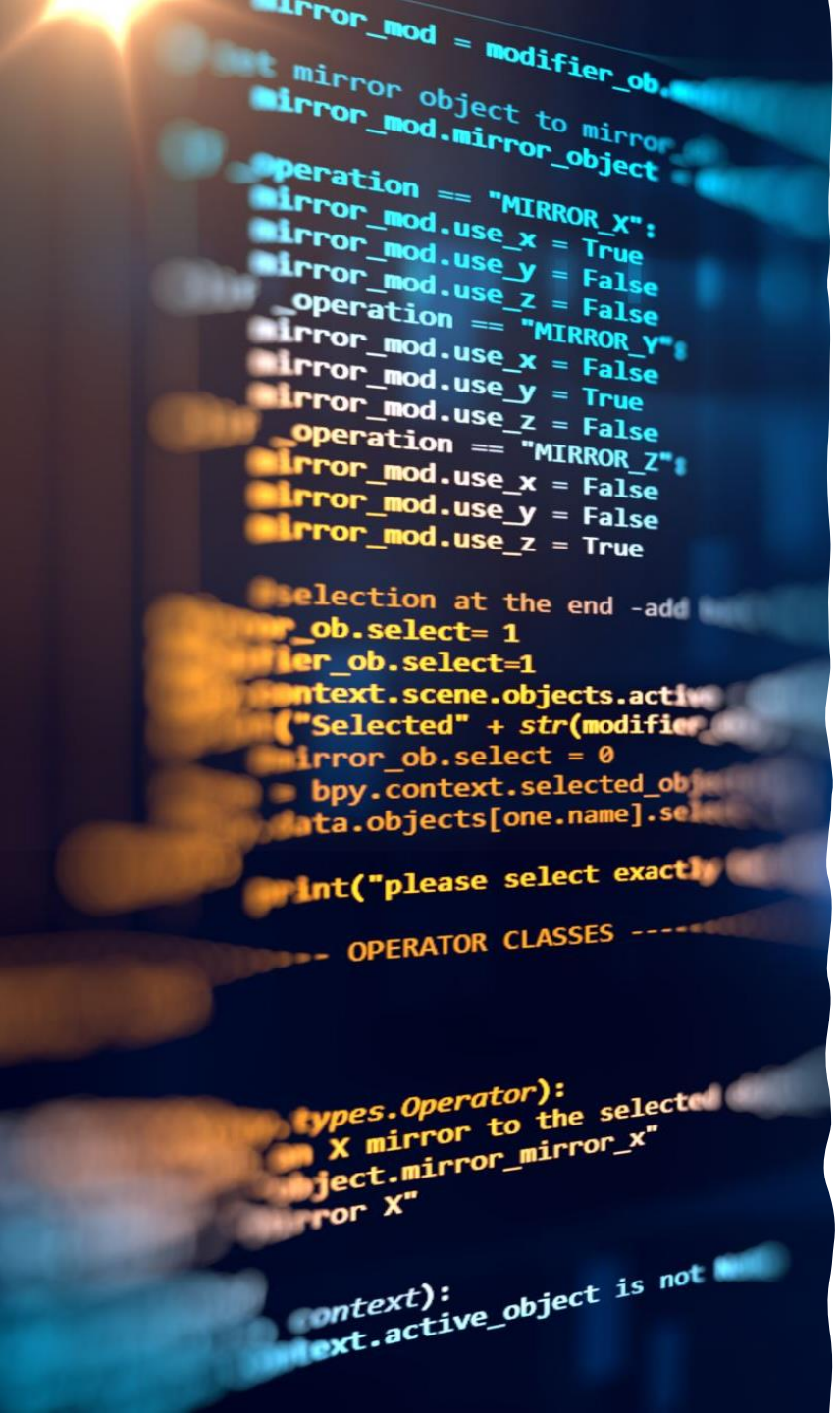
2. Interface is Complex

- Another big disadvantage is that the interface of Structured query language is difficult, which makes it difficult for SQL users to use and manage it.

3. Partial Database control

- The business rules are hidden. So, the data professionals and users who are using this query language cannot have full database control.

EXL



SQL Syntax

- When you want to do some operations on the data in the database, then you must have to write the query in the predefined syntax of SQL.
- The syntax of the structured query language is a unique set of rules and guidelines, which is not case-sensitive. Its Syntax is defined and maintained by the ISO and ANSI standards.
- Following are some most important points about the SQL syntax which are to remember:
- You can write the keywords of SQL in both uppercase and lowercase, but writing the SQL keywords in uppercase improves the readability of the SQL query.
- SQL statements or syntax are dependent on text lines. We can place a single SQL statement on one or multiple text lines.
- You can perform most of the action in a database with SQL statements.
- SQL syntax depends on relational algebra and tuple relational calculus.

EXL

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";

- Each SQL statement begins with any of the SQL keywords and ends with the semicolon (;). The semicolon is used in the SQL for separating the multiple Sql statements which are going to execute in the same call. In this SQL tutorial, we will use the semicolon (;) at the end of each SQL query or statement.

EXL

Most Important SQL Commands and Statements

Select Statement	Update Statement	Delete Statement	Create Table Statement	Alter Table Statement	Drop Table Statement
Create Database Statement	Drop Database Statement	Insert Into Statement	Truncate Table Statement	Describe Statement	Distinct Clause
Commit Statement	Rollback Statement	Create Index Statement	Drop Index Statement	Use Statement	

EXL

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

   [**FROM** table_name]

   [**WHERE** condition]

   [**ORDER BY** order_column_name1 [**ASC** | **DESC**], ...];

Example of SELECT Statement:

1. **SELECT** Emp_ID, First_Name, Last_Name, Salary, City

   FROM Employee_details

   WHERE Salary = 100000

   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**.

EXL

UPDATE Statement

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

- **Syntax of UPDATE Statement:**

1. **UPDATE** table_name

   SET column_name1 = new_value_1, column_name2 = new_value_2, ..., column_nameN = new _value_N

   [**WHERE** CONDITION];

Example of UPDATE Statement:

UPDATE Employee_details

SET Salary = 100000

WHERE Emp_ID = 10;

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

EXL

DELETE Statement

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

Syntax of DELETE Statement:

```
DELETE FROM table_name
 [ WHERE CONDITION ];
```

Example of DELETE Statement:

```
DELETE FROM Employee_details
 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.

EXL

CREATE DATABASE Statement

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

- **Syntax of CREATE DATABASE Statement:**

  CREATE DATABASE database_name;

- **Example of CREATE DATABASE Statement:**

  CREATE DATABASE Company;

USE: This command is used to select a specific database to work with. For example:

USE database_name;

EXL

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:**

  DROP DATABASE database_name;

- **Example of DROP DATABASE Statement:**

  DROP DATABASE Company;

- The above example deletes the company database from the system.

EXL

CREATE TABLE Statement

```
CREATE TABLE table_name
(
column_name1 data_type [column1 constraint(s)],
column_name2 data_type [column2 constraint(s)],
.....
.....,
column_nameN data_type [columnN constraint(s)],
PRIMARY KEY(one or more col)
);
```

```
CREATE TABLE Employee_details(
    Emp_Id NUMBER(4) NOT NULL,
    First_name VARCHAR(30),
    Last_name VARCHAR(30),
    Salary Money,
    City VARCHAR(30),
    PRIMARY KEY (Emp_Id)
 );
```

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

EXL

ALTER TABLE Statement

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

Syntax of ALTER TABLE Statement:

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.

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.

ALTER TABLE table_name **DROP COLUMN** column_name;

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

EXL

ALTER TABLE Statement

Example of ALTER TABLE Statement:

ALTER TABLE Employee_details

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.

EXL

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:**

  DROP TABLE [IF EXISTS]

  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:

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.

EXL

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:

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


EXL

Example of insert a single record

```
INSERT INTO Employee_details
(
Emp_ID,
First_name,
Last_name,
Salary,
City
)
VALUES
(101,
Akhil,
Sharma,
40000,
Bangalore
);
```

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**.

EXL

Syntax of inserting a multiple records in a single query

INSERT INTO table_name

(column_name1, column_name2, ..., column_nameN)

VALUES (value_1, value_2, ..., value_N), (value_1, value_2, ..., value_N),....;

Example of inserting multiple records in a single query:

INSERT INTO Employee_details

(Emp_ID, First_name, Last_name, Salary, City)

VALUES (101, Amit, Gupta, 50000, Mumbai), (101, John, Aggarwal, 45000, Calcutta), (101, Sidhu, Arora, 55000, Mumbai);

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

EXL

TRUNCATE TABLE Statement

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

- **Syntax of TRUNCATE TABLE Statement:**

TRUNCATE TABLE table_name;

- **Example of TRUNCATE TABLE Statement:**

TRUNCATE TABLE Employee_details;

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

EXL

DESCRIBE Statement

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

- **Syntax of DESCRIBE Statement:**

DESCRIBE table_name | view_name;

- **Example of DESCRIBE Statement:**

DESCRIBE Employee_details;

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

EXL

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.

EXL

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.

EXL

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:**

  1. **DELETE FROM** Employee_details
  2. **WHERE** City = Mumbai;
  3. **ROLLBACK**;

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

EXL

CREATE INDEX Statement

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

- **Syntax of CREATE INDEX Statement:**
  1. **CREATE INDEX** index_name
  2. **ON** table_name (column_name1, column_name2, ..., column_nameN);
- **Example of CREATE INDEX Statement:**
  1. **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.

EXL

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.

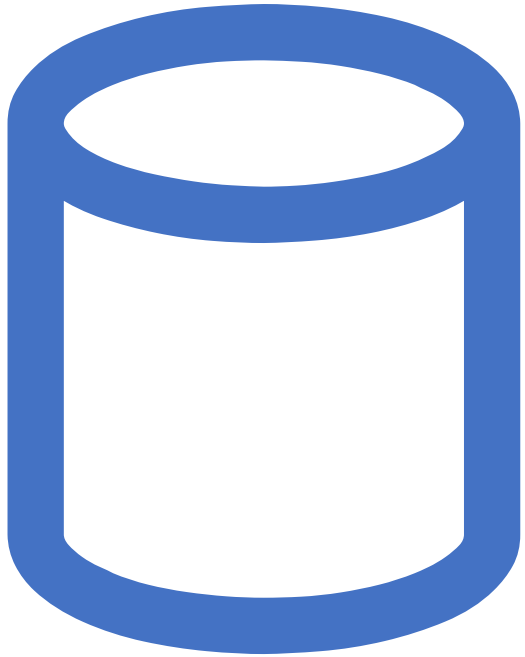
EXL

SQL Data Types

- Data types are used to represent the nature of the data that can be stored in the database table. For example, in a particular column of a table, if we want to store a string type of data then we will have to declare a string data type of this column.

Data types mainly classified into three categories for every database.

- String Data types
- Numeric Data types
- Date and time Data types



Data Types in MySQL

- A list of data types used in MySQL database. This is based on MySQL 8.0.

CHAR(Size)	It is used to specify a fixed length string that can contain numbers, letters, and special characters. Its size can be 0 to 255 characters. Default is 1.
VARCHAR(Size)	It is used to specify a variable length string that can contain numbers, letters, and special characters. Its size can be from 0 to 65535 characters.
BINARY(Size)	It is equal to CHAR() but stores binary byte strings. Its size parameter specifies the column length in the bytes. Default is 1.
VARBINARY(Size)	It is equal to VARCHAR() but stores binary byte strings. Its size parameter specifies the maximum column length in bytes.
TEXT(Size)	It holds a string that can contain a maximum length of 255 characters.
TINYTEXT	It holds a string with a maximum length of 255 characters.
MEDIUMTEXT	It holds a string with a maximum length of 16,777,215.
LONGTEXT	It holds a string with a maximum length of 4,294,967,295 characters.
ENUM(val1, val2, val3,...)	It is used when a string object having only one value, chosen from a list of possible values. It contains 65535 values in an ENUM list. If you insert a value that is not in the list, a blank value will be inserted.
SET(val1,val2,val3,....)	It is used to specify a string that can have 0 or more values, chosen from a list of possible values. You can list up to 64 values at one time in a SET list.
BLOB(size)	It is used for BLOBs (Binary Large Objects). It can hold up to 65,535 bytes.

EXL

MySQL Numeric Data Types

- A list of data types used in MySQL database. This is based on MySQL 8.0

BIT(Size)	It is used for a bit-value type. The number of bits per value is specified in size. Its size can be 1 to 64. The default value is 1.
INT(size)	It is used for the integer value. Its signed range varies from -2147483648 to 2147483647 and unsigned range varies from 0 to 4294967295. The size parameter specifies the max display width that is 255.
INTEGER(size)	It is equal to INT(size).
FLOAT(size, d)	It is used to specify a floating point number. Its size parameter specifies the total number of digits. The number of digits after the decimal point is specified by **d** parameter.
FLOAT(p)	It is used to specify a floating point number. MySQL used p parameter to determine whether to use FLOAT or DOUBLE. If p is between 0 to24, the data type becomes FLOAT (). If p is from 25 to 53, the data type becomes DOUBLE().
DOUBLE(size, d)	It is a normal size floating point number. Its size parameter specifies the total number of digits. The number of digits after the decimal is specified by d parameter.
DECIMAL(size, d)	It is used to specify a fixed point number. Its size parameter specifies the total number of digits. The number of digits after the decimal parameter is specified by **d** parameter. The maximum value for the size is 65, and the default value is 10. The maximum value for **d** is 30, and the default value is 0.
DEC(size, d)	It is equal to DECIMAL(size, d).
BOOL	It is used to specify Boolean values true and false. Zero is considered as false, and nonzero values are considered as true.

EXL

MySQL Date and Time Data Types

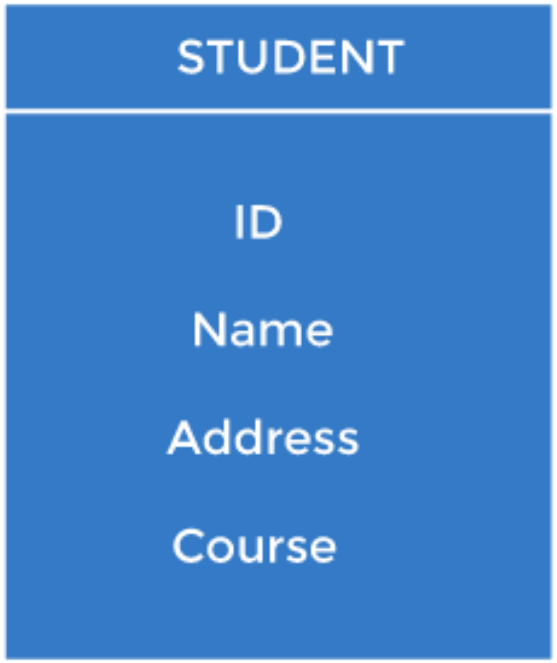
- A list of data types used in MySQL database.

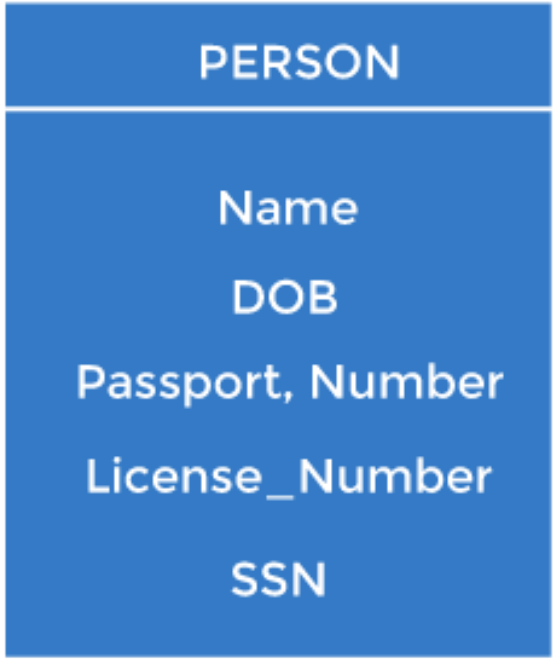
DATE	It is used to specify date format YYYY-MM-DD. Its supported range is from '1000-01-01' to '9999-12-31'.
DATETIME(fsp)	It is used to specify date and time combination. Its format is YYYY-MM-DD hh:mm:ss. Its supported range is from '1000-01-01 00:00:00' to 9999-12-31 23:59:59'.
TIMESTAMP(fsp)	It is used to specify the timestamp. Its value is stored as the number of seconds since the Unix epoch('1970-01-01 00:00:00' UTC). Its format is YYYY-MM-DD hh:mm:ss. Its supported range is from '1970-01-01 00:00:01' UTC to '2038-01-09 03:14:07' UTC.
TIME(fsp)	It is used to specify the time format. Its format is hh:mm:ss. Its supported range is from '-838:59:59' to '838:59:59'
YEAR	It is used to specify a year in four-digit format. Values allowed in four digit format from 1901 to 2155, and 0000.

EXL

SQL Keys

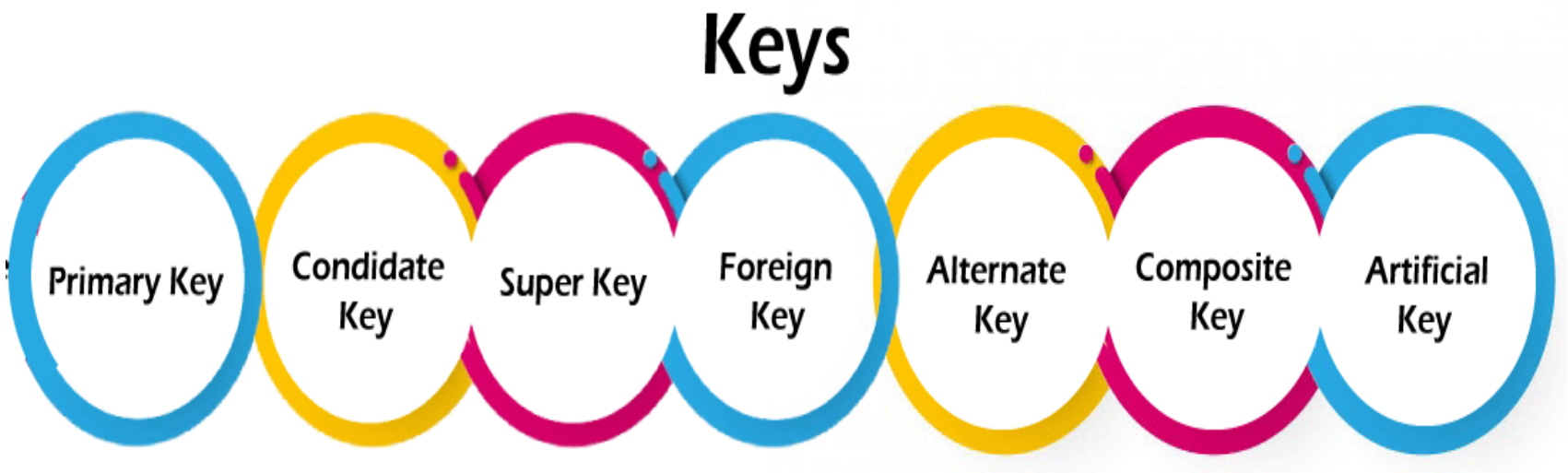
- Keys play an important role in the relational database.
- It is used to uniquely identify any record or row of data from the table. It is also used to establish and identify relationships between tables.
- **For example,** ID is used as a key in the Student table because it is unique for each student. In the PERSON table, passport_number, license_number, SSN are keys since they are unique for each person.





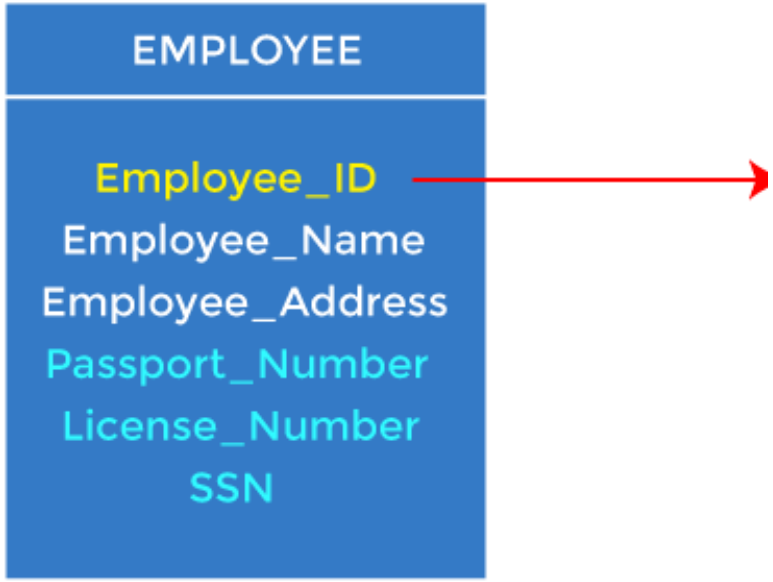
EXL

Types of Keys:



EXL

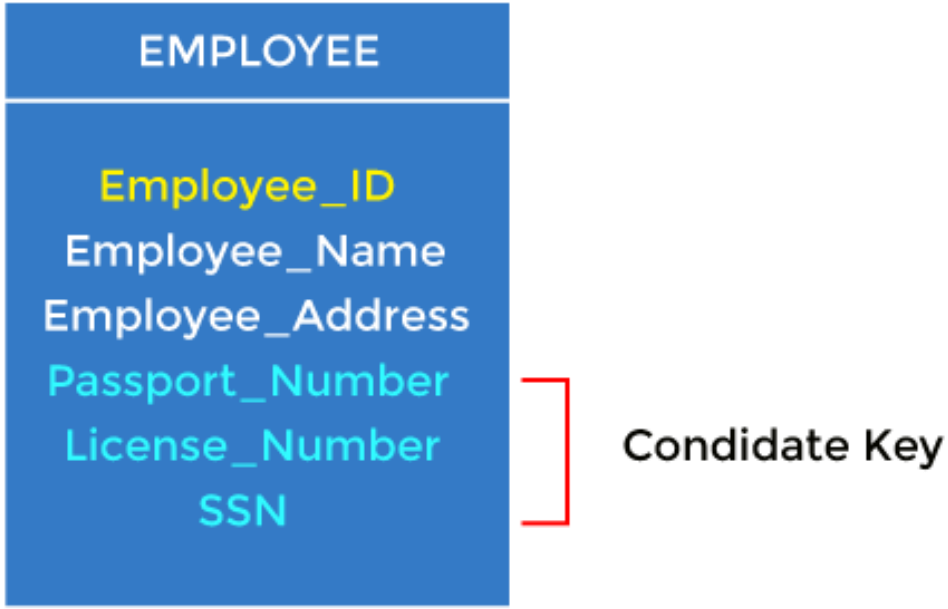
Primary Key



- 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.

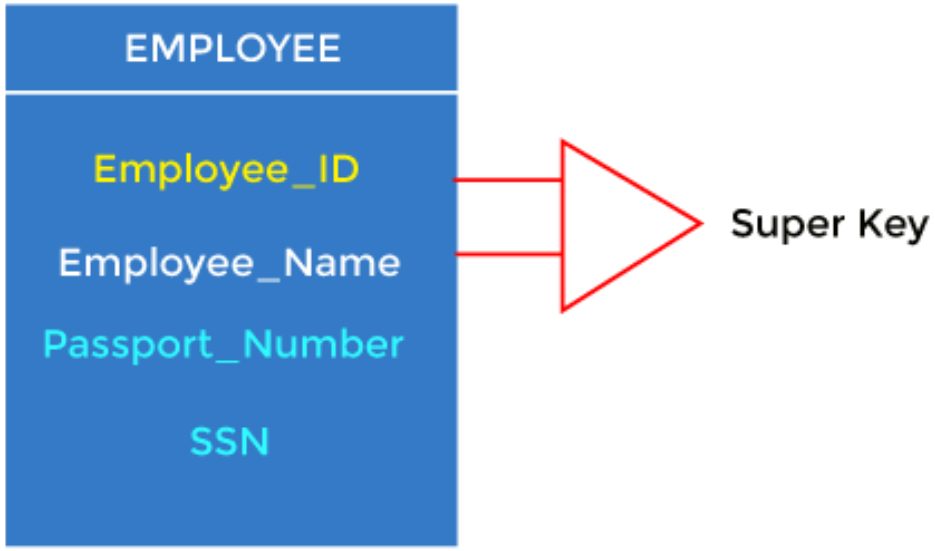
EXL

Candidate key



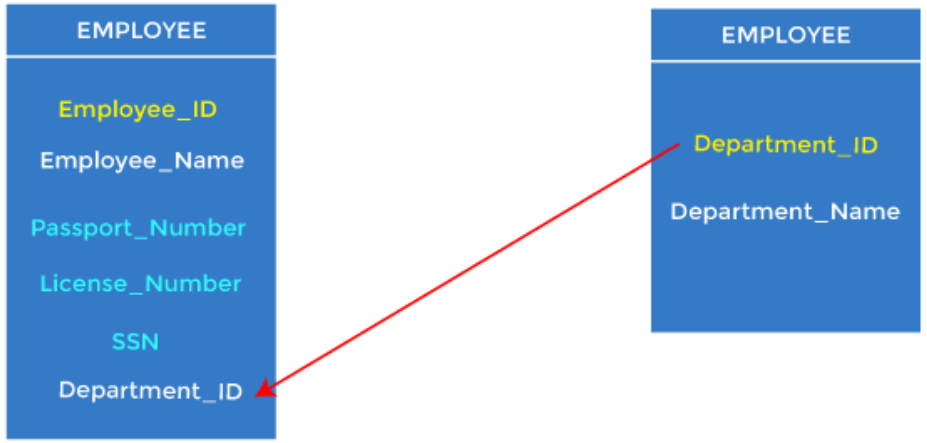
- A candidate key is an attribute or set of attributes that can uniquely identify a tuple.
- Except for the primary key, the remaining attributes are considered a candidate key. The candidate keys are as strong as the primary key.
- **For example:** In the EMPLOYEE table, id is best suited for the primary key. The rest of the attributes, like SSN, Passport_Number, License_Number, etc., are considered a candidate key.

Super Key



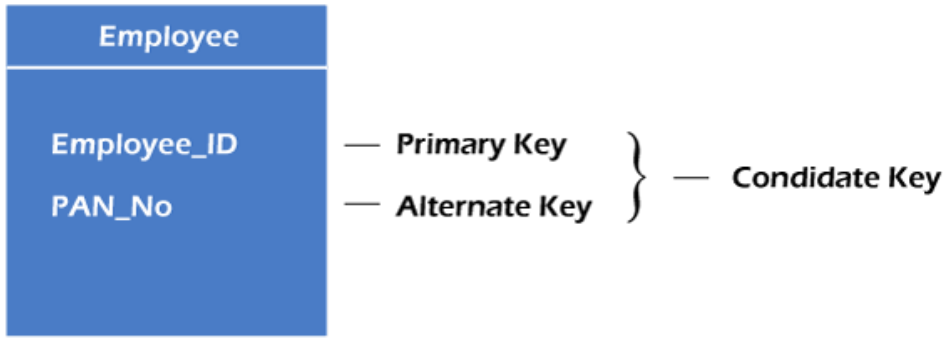
- Super key is an attribute set that can uniquely identify a tuple. A super key is a superset of a candidate key.
- **For example:** In the above EMPLOYEE table, for(EMPLOEE_ID, EMPLOYEE_NAME), the name of two employees can be the same, but their EMPLYEE_ID can't be the same. Hence, this combination can also be a key.
- The super key would be EMPLOYEE-ID (EMPLOYEE_ID, EMPLOYEE-NAME), etc.

Foreign key



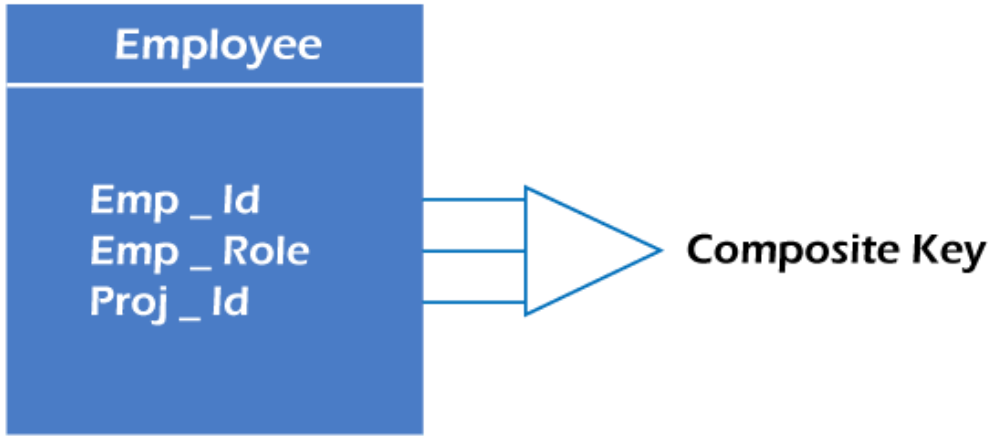
- Foreign keys are the column of the table used to point to the primary key of another table.
- Every employee works in a specific department in a company, and employee and department are two different entities. So we can't store the department's information in the employee table. That's why we link 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.

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.

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.