**Index**

Indexes are used to retrieve data from the database very fast. The users cannot see the indexes, they are just used to speed up searches/queries.

* index in SQL is a tool used to quickly identify rows with specific column values. If there were no indexes, the SQL server would have to start with the first row and then go through the entire table until it discovers the relevant rows. This method is known as a full-table scan and can be highly inefficient for large tables.

**Why Indexes in SQL are Used?**

* **Improved Query Performance**: The primary reason for using indexes is to accelerate query processing. Indexes can drastically reduce the amount of data the server needs to examine.
* **Efficient Data Access**: Indexes provide a quick way to access row data for SELECT statements. This is particularly beneficial for tables with a large number of rows.
* **Sorting and Grouping Speed**: Indexes improve the speed of data retrieval operations by providing a sorted version of the data, which is faster to process for ORDER BY and GROUP BY operations.
* **Unique Constraints**: Indexes can be used to enforce uniqueness for columns to ensure that no two rows of a table have duplicate values in a particular column or a combination of columns.
* **Optimized Join Operations**: In databases with multiple tables, indexes improve the speed of join operations by quickly locating the joining rows in each table.

CREATE INDEX idx\_pname ON Persons (LastName, FirstName);

**SQL:-** DROP INDEX table\_name.index\_name;

**MY SQL** : - ALTER TABLE table\_nameDROP INDEX index\_name;

**Primary Key Index**

A primary key is a field or a combination of fields in a database table that uniquely identifies each record (row) in that table. A primary key index is an automatically generated index associated with the primary key column(s) to enhance data retrieval and enforce data uniqueness.

**Unique Key Index**

A unique index in a relational database is a data structure that enforces the uniqueness constraint on one or more columns within a table. Its primary purpose is to ensure that values stored in the indexed column(s) are unique across all records in the table.

**Clustered Index**

A clustered index sorts and stores the rows of a table based on the values in one or more specified columns. Each table can have only one clustered index, and the choice of the clustering column(s) significantly impacts how data is stored and retrieved.

**Non-Clustered Index**

non-clustered indexes create separate data structures to allow fast access to specific data subsets.

non-clustered indexes do not rearrange the physical organization of data, but rather create a separate structure to facilitate quicker access to the data.

**Views and types**

A view is a virtual table. A view consists of rows and columns just like a table. The difference between a view and a table is that views are definitions built on top of other tables (or views), and do not hold data themselves. If data is changing in the underlying table, the same change is reflected in the view. A view can be built on top of a single table or multiple tables. It can also be built on top of another view

**CREATE VIEW V\_Customer AS SELECT First\_Name, Last\_Name, Country  
FROM Customer;**

**Materialised view -** a table on a disk that contains the result set of a query

**Non-materiased view** - a query that pulls data from the underlying table

**UPDATE Query :**

**UPDATE "table\_name" SET "column\_1" = [new value] WHERE "condition";**

**Substring -** The **Substring** function in SQL is used to return a portion of the string

**SELECT SUBSTR (Store\_Name, 3) FROM GeographyWHERE Store\_Name = 'Los Angeles';**

**INSTR Function -** The **INSTR** function in SQL is used to find the starting location of a pattern in a string.

**SELECT INSTR (Store\_Name, 'o')FROM Geography WHERE Store\_Name = 'Los Angeles';** Result -2

**Minus-**

The **MINUS** command operates on two SQL statements. It takes all the results from the first SQL statement, and then subtract out the ones that are present in the second SQL statement to get the final result set. If the second SQL statement includes results not present in the first SQL statement, such results are ignored.

A **subquery** is a SQL statement that has another SQL query embedded in the [**WHERE**](https://www.1keydata.com/sql/sqlwhere.html) or the [**HAVING**](https://www.1keydata.com/sql/having-clause.html) clause.

**SELECT SUM (Sales) FROM Store\_Information WHERE Store\_Name IN  
(SELECT Store\_Name FROM Geography WHERE Region\_Name = 'West');**

**NVL Function**The **NVL( )** function is available in Oracle, and not in MySQL or SQL Server. This function is used to replace NULL value with another value.

**SELECT SUM(NVL(Sales,100)) FROM Sales\_Data;**

**COALESCE** function in SQL returns the first non-NULL expression among its arguments.

**SELECT Name, COALESCE (Business\_Phone, Cell\_Phone, Home\_Phone) Contact\_Phone  
FROM Contact\_Info;**

[**SQL**](https://www.1keydata.com/sql/sql.html) > **Constraint**

* [**NOT NULL Constraint**](https://www.1keydata.com/sql/not-null-constraint.html): Ensures that a column cannot have NULL value.

-If you do not want to allow **NULL** value in a column, you will want to place the **NOT NULL** constraint on this column.

* [**DEFAULT Constraint**](https://www.1keydata.com/sql/default-constraint.html): Provides a default value for a column when none is specified.

-The **DEFAULT** constraint provides a default value to a column when the [**INSERT INTO**](https://www.1keydata.com/sql/sqlinsert.html) statement does not provide a specific value

**CREATE TABLE Student  
(Student\_ID integer Unique,First\_Name varchar (30),Integer DEFAULT 80);**

* [**UNIQUE Constraint**](https://www.1keydata.com/sql/unique-constraint.html): Ensures that all values in a column are different.

The **UNIQUE** constraint ensures that all values in a column are distinct.

**CREATE TABLE Customer(SID integer UNIQUE,First\_Name varchar(30));**

* [**CHECK Constraint**](https://www.1keydata.com/sql/check-constraint.html): Makes sure that all values in a column satisfy certain criteria.

The **CHECK** constraint ensures that all values in a column satisfy certain conditions. Once defined, the database will only insert a new row or update an existing row if the new value satisfies the **CHECK** constraint. The **CHECK** constraint is used to ensure data quality.

**CREATE TABLE Customer (SID integer CHECK (SID > 0),First\_Name varchar(30));**

* [**Primary Key Constraint**](https://www.1keydata.com/sql/sql-primary-key.html): Used to uniquely identify a row in the table.

A primary key is used to uniquely identify each row in a table. A primary key can consist of one or more columns on a table. When multiple columns are used as a primary key, they are called a composite key.

* [**Foreign Key Constraint**](https://www.1keydata.com/sql/sql-foreign-key.html): Used to ensure referential integrity of the data.

A foreign key is a column (or columns) that references a column (most often the primary key) of another table. The purpose of the foreign key is to ensure referential integrity of the data. In other words, only values that are supposed to appear in the database are permitted.