SQL *(Structured Query Language)*

- SQL is a standard language based on the English language.
- SQL is the core of the relational database which is used for accessing and managing database.

SUBSETS of SQL:

- ***DDL** (Data Definition Language)* – It allows you to perform various operations on the database such as CREATE, ALTER and DELETE objects.
- ***DML** (Data Manipulation Language)* – It allows you to access and manipulate data. It helps you to insert, update, delete and retrieve data from the database.
- ***DCL** (Data Control Language)* – It allows you to control access to the database. Example – Grant, Revoke access permissions.

MySQL *(DBMS)*

MySQL is a **Relational database management system,** such as **SQL Server, Informix etc.**

Database Management System (DBMS) – a software application that interacts with the user, applications and the database itself to capture and analyze data. The data stored in the database can be modified, retrieved and deleted and can be of any type like strings, numbers, images etc. There are two types of DBMS:

- ***Relational Database Management System***: The data is stored in relations (tables). Example – **MySQL**.
- ***Non-Relational Database Management System***: There is no concept of relations, tuples and attributes. Example – Mongo

SQL General

Data Integrity – defines the accuracy as well as the consistency of the data stored in a database. It also defines integrity constraints to enforce business rules on the data when it is entered into an application or a database.

Datawarehouse – a central repository of data where the data is assembled from multiple sources of information. Those data are consolidated, transformed and made available for the mining as well as online processing. Warehouse data also have a subset of data called Data Marts.

TABLE – *to a collection of data in an organised manner in form of rows and columns.*
FIELD – *the number of columns in a table.*

***ENTITIES* –** A person, place, or thing in the real world about which data can be stored in a database. Tables store data that represents one type of entity. For example – A bank database has a customer table to store customer information. Customer table stores this information as a set of attributes (columns within the table) for each customer.

***RELATIONSHIPS* –** Relation or links between entities that have something to do with each other. E.g., the customer name is related to the customer account number and contact information, which might be in the same table. There can also be relationships between separate tables (for example, customer to accounts). There are various relationships, namely:

- **One to One** Relationship.
- **One to Many** Relationship.
- **Many to One** Relationship.
- **Self-Referencing** Relationship.

ACID *(Atomicity, Consistency, Isolation, Durability)*

Used to ensure that the data transactions are processed reliably in a database system.

- **Atomicity –** refers to the transactions that are completely done or failed where transaction refers to a single logical operation of a data. It means if one part of any transaction fails, the entire transaction fails and the database state is left unchanged.
- **Consistency –** ensures that the data must meet all the validation rules. In simple words, you can say that your transaction never leaves the database without completing its state.
- **Isolation –** the main goal of isolation in concurrency control.
- **Durability –** means that if a transaction has been commited, it will occur whatever may come in between (e.g.: power loss, crash, any sort of error).

Keys

Primary key – a column (or collection of columns) or a set of columns that uniquely identifies each row in the table.

- Uniquely identifies a single row in the table.
- Null values not allowed.

Unique key

- Uniquely identifies a single row in the table.
- Multiple values allowed per table.
- Null values allowed.

Foreign key

- Maintains referential integrity by enforcing a link between the data in two tables.
- The foreign key in the child table references the primary key in the parent table.
- The foreign key constraint prevents actions that would destroy links between the child and parent tables.

Auto Increment keyword allows the user to create a unique number to get generated whenever a new record is inserted into the table. This keyword is usually required whenever PRIMARY KEY is used.

- AUTO INCREMENT keyword can be used in Oracle and IDENTITY keyword can be used in SQL SERVER.

SQL Index

Refers to a performance tuning method of allowing faster retrieval of records from the table. An index creates an entry for each value and hence it will be faster to retrieve data. There are 3 types of indices:

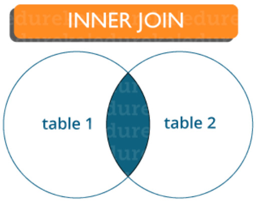
- ***Unique*** *index* – does not allow the field to have duplicate values if the column is unique indexed. If a primary key is defined, a unique index can be applied automatically.
- ***Clustered*** *index* - reorders the physical order of the table and searches based on the basis of key values. Each table can only have one clustered index. It is used for an easy retrieval of the data from the database and its faster as opposed to a non-clustered index. It alters the way records are stored in a database by sorting out rows by the column which is set to be clustered index.
- ***Non-clustered*** *index* - does not alter the physical order of the table and maintains a logical order of the data. Each table can have many nonclustered indexes. It does not alter the way it was stored but it creates a separate object within a table which points back to the original table rows after searching.

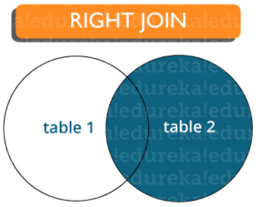
SQL Joins

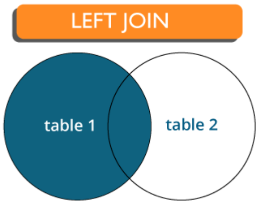
Clauses used to combine rows from two or more tables, based on a related column between them. It is used to merge two tables or retrieve data from there. There are 4 joins in SQL namely:

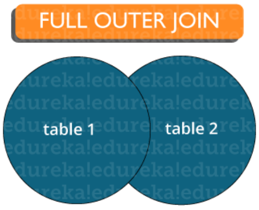
Cross vs. **Natural**
The cross join produces the cross product or Cartesian product of two tables, whereas the natural join is based on all the columns having the same name and data types in both of the tables.

- **Inner** – the most common type of the joins, used to return all the rows from multiple tables where the join condition is satisfied.
- **Right** – used to return all the rows from the right table but only the matching rows from the left table where the join condition is fulfilled.
- **Left** – used to return all the rows from the left table but only the matching rows from the right table where the join condition is fulfilled.
- **Full** – returns all the records when there is a match in any of the tables. Therefore, it returns all the rows from the left-hand side table and all the rows from the right-hand side table.









Normalization

The process of organizing data to avoid duplication and redundancy. There many successive levels of normalization, called normal forms. Each consecutive normal form depends on the previous one. The first three are usually adequate:

First Normal Form (1NF) – No repeating groups within rows

Second Normal Form (2NF) – Every non-key (supporting) column value is dependent on the whole primary key.

Third Normal Form (3NF) – Dependent solely on the primary key and no other non-key (supporting) column value.

Advantages:

- Better database organization, more tables with smaller rows.
- Efficient data access (quickly find the information) and greater flexibility in queries.
- Easier to implement security measures, and allows for an easier modification in general.
- Reduction of redundant and duplicate data, a more compact database.
- Ensure consistent data after modification.

Denormalization – a technique which is used to access data from higher to lower forms of a database. It helps the database managers to increase the performance of the entire infrastructure as it introduces redundancy into a table. It adds the redundant data into a table by incorporating database queries that combine data from various tables into a single table.

Constraints

- NOT NULL
- CHECK
- DEFAULT
- UNIQUE
- PRIMARY KEY
- *FOREIGN KEY*

Used specify the limit on the data type of the table. It can be specified while creating or altering the table statement. Enforces data entity and consistency. There are **table-level** and **column-level** constraints.

Variables

Local variables – can be used or exist sonly inside the function, are not referred by any other function.

Global variables – can be accessed throughout the program, but cannot be created whenever that function is called.

CHAR vs. VARCHAR2 **–** both are used for characters datatype, but varchar2 allows to store character strings of variable length. E.g., char(10) can only store 10 characters whereas varchar2(10) can store any length i.e 6,8,2 in this variable.

Operators

- **Arithmetic**
- **Logical**
- **Comparison**
- **Set operators –** Union, Intersect or Minus operators.

Condition Operators:

- **BETWEEN** is used to display rows based on a range of values in a row.
  - *SELECT * FROM Students where ROLL_NO BETWEEN 10 AND 50;*
- **IN** is used to check for values contained in a specific set of values.
  - *SELECT * FROM students where ROLL_NO IN (8,15,25);*

Operators for pattern matching:

- **LIKE** operator is used for pattern matching, and it can be used as -.
- **%** – It matches zero or more characters – *select * from students where studentname like ‘a%’*
- **_ *(Underscore)* –** it matches exactly one character – select * from student where studentname like ‘abc_’

Clauses & Functions

SQL clauses help to limit the result set by providing a condition to the query, filtering the rows from the entire set of records.

- **HAVING** clause can be used only with **SELECT** statement and in a **GROUP BY** clause (whenever group by is missing, it behaves just like the WHERE clause).
- **WHERE** clause is applied to each row before they are a part of the GROUP BY function in a query.

FUNCTIONS

- To perform some calculations on the data.
- To modify individual data items.
- To manipulate the output.
- To format dates and numbers.
- To convert the data types

Types of user-defined functions:

- Scalar Functions
- Inline Table-valued functions
- Multi-statement valued functions

Scalar returns the unit, variant defined the return clause. Other two types of defined functions return tables.

- ***Aggregate*** functions are used to evaluate mathematical calculation and returns a single value. These calculations are done from the columns in a table. E.g.: ***max()***, ***count()*** are calculated with respect to numeric.
- ***Scalar*** functions return a single value based on the input value. For example – ***UCASE()***, ***NOW()*** are calculated with respect to string.
- ***Group functions* –** work on the set of rows and returns one result per group. Some of the commonly used group functions are: ***AVG***, ***COUNT***, ***MAX***, ***MIN***, ***SUM***, ***VARIANCE***.

***MERGE* –** allows conditional updates or insertion of data into the table. It performs an UPDATE if a row exists, or an INSERT otherwise.

ALIAS name can be given to any table or a column. This alias name can be referred in **WHERE** clause to identify a particular table or a column. E.g., in the example below, *emp* refers to alias name for employee table and *dept* refers to alias name for department table.

- *Select emp.empID, dept.Result from employee emp, department as dept where emp.empID=dept.empID*

Collation – a set of rules that determine how data can be sorted as well as compared. Character data is sorted using the rules that define the correct character sequence along with options for specifying case-sensitivity, character width etc. Different types of collation sensitivity:

- **Case** Sensitivity – A and a and B and b.
- **Kana** Sensitivity – Japanese Kana characters.
- **Width** Sensitivity – Single byte character and double-byte character.
- **Accent** Sensitivity.

Case manipulation functions:

- **LOWER('string')** - returns the string in lowercase. It takes a string as an argument and returns it by converting it into lower case.
- **UPPER('string') -** returns the string in uppercase. It takes a string as an argument and returns it by converting it into uppercase.
- **INITCAP('string')** - returns the string with the first letter in uppercase and rest of the letters in lowercase.

REPLACE – Used to replace the existing characters of all the occurrences:

- *REPLACE (string_expression, search_string, replacement_string)*

Here, every *search_string* in the *string_expression* will be replaced with the *replacement_string*.

STUFF – overwrite existing character or inserts a string into another string.

STUFF(string_expression, start, length, replacement_characters)

- *string_expression* – string that will have characters substituted.
- *start* – the starting position
- *length* – the N of characters in the string which are substituted.
- *replacement_string* – new characters which are injected in the string.

DELETE

- Delete command is used to delete a row in a table.
- You can rollback data after using delete statement.
- It is a DML command.
- It is slower than truncate statement.

TRUNCATE

- Truncate is used to delete all the rows from a table.
- You cannot rollback data.
- It is a DDL command.
- It is faster.

DROP

- Drop is used to remova a table.
- Cannot be rolled back from the database.

Null

Inserting NULL values in a column while inserting the data?

- Implicitly by omitting column from column list.
- Explicitly by specifying NULL keyword in the VALUES clause

NULL values the same as zero or a blank space?

A NULL value is not at all same as that of zero or a blank space. NULL value represents a value which is unavailable, unknown, assigned or not applicable whereas a zero is a number and blank space is a character.

Additional

SQL vs. PL/SQL

- SQL is a query language that allows you to issue a single query or execute a single insert/update/delete.
- PL/SQL is Oracle’s “Procedural Language” SQL, which allows you to write a full program (loops, variables, etc.) to accomplish multiple operations such as selects/inserts/updates/deletes.

Dynamic SQL can be executed by:

- Writing a query with some parameters.
- Using EXEC.
- Using sp_executesql.

VIEW – a virtual table (a snapshot) which consists of a subset of data contained in a table. Since views are not present, it takes less space to store. View can have data of one or more tables combined and it depends on the relationship. It is used for:

- Restricting access to data.
- Making complex queries simple.
- Ensuring data independence.
- **Providing different views of same data.**

Stored Procedure – a function that consists of many SQL statements to access the database system. Several SQL statements are consolidated into a stored procedure and executed whenever required. This saves time and avoids repetition in writing the code, supports fast executios. It also reduces network traffic and provides better security to the data. However, these procedures can be executed only within the databases and utilize more memory in the database server.

- **Recursive stored procedure –** calls by itself until it reaches some boundary condition. This allows programmers to use the same set of code N of times.

Trigger in SQL are a special type of stored procedures that are defined to execute automatically in place or after data modifications. It allows you to execute a batch of code when an insert, update or any other query is executed against a specific table.

Example Queries

Correlated subquery: These are queries which select the data from a table referenced in the outer query. It is not considered as an independent query as it refers to another table and refers the column in a table.

Non-Correlated subquery: This query is an independent query where the output of subquery is substituted in the main

SUBQUERY – query inside another query where a query is defined to retrieve data or information back from the database.

- In a subquery, the outer query is called as the main query whereas the inner query is called subquery.
- Subqueries are always executed first and the result of the subquery is passed on to the main query.
- It can be nested inside a SELECT, UPDATE or any other query.
- A subquery can also use any comparison operators such as >,< or =.

Display the current date In SQL, there is a built-in function called ***GetDate()*** which helps to return the current timestamp/date.	**Count the number of records in a table:** • *SELECT * FROM table1* • *SELECT COUNT(*) FROM table1* • *SELECT rows FROM sysindexes WHERE id = OBJECT_ID(table1) AND indid < 2* ***Get unique records from a table:*** • *Select DISTINCT studentID from Student*

N of employees that begin with 'A':
● *SELECT * FROM Table_name WHERE EmpName like 'A%'*
	Get the third highest salary of an employee from employee_table:
	SELECT TOP 1 salary
	FROM(
	SELECT TOP 3 salary
	FROM employee_table
	ORDER BY salary DESC) AS emp
	ORDER BY salary ASC;

Fetch common records from two tables: • *Select studentID from student. <strong>INTERSECT </strong> Select StudentID from Exam*	**Fetching alternate records from a table** You can fetch alternate records, i.e. both odd and even row numbers. • Select studentId from (Select rowno, studentId from student) where mod(rowno,2)=0 • Select studentId from (Select rowno, studentId from student) where mod(rowno,2)=1
Fetch first 5 characters of the string? ● ***Select SUBSTRING(StudentName,1,5) as studentname from student***	