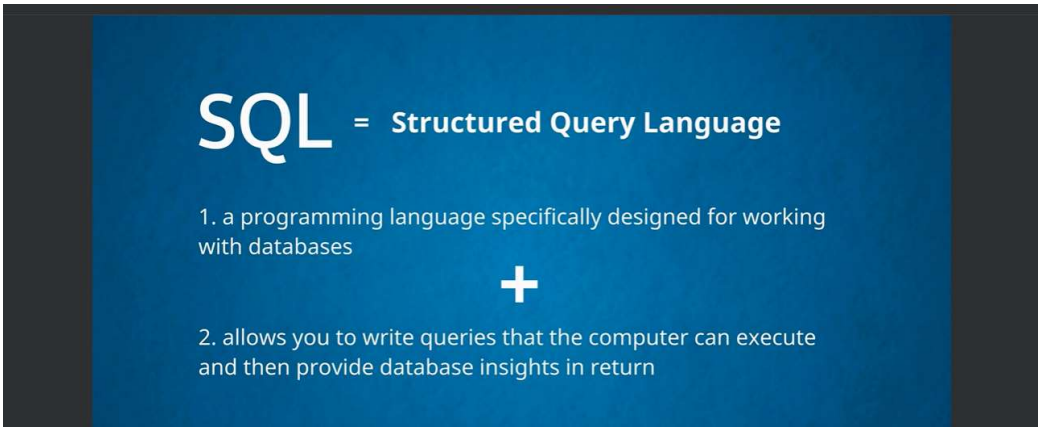
11 December 2023 11:58

SQL :-



The SQL optimizer will separate your task into smaller steps and do the magic to give you the desired output?

RDS ---MY SQL , Ora , PSQL --- table -- Column, Rows
NRDS-- Mongo DB , Neo4J -- Table -- attributes , values

tables -- Fact ,, Dimne

DDL -- DATA definition Language
- a Syntax
- A set of statements that allow the user to define or modify data structures and objects, such as tables
-- The Create statement
used for creating entire database and database objects as table

CREATE -- Used to create the table
ALTER-- used when altering existing onbjects ,, Add cloumns , Rename colum , Remove the colunm
DROP -- where we can delete the comple table
RENAME -- where we can rename the column
TRUNCATE -- Instead of deleting an entire table through drop , we can also remove its data and contains to have the table as an object in the data base

Keywords == reserved word
They cannot be used when naming objects

The ALTER statement
used when altering existing objects
Add
remove
Rename

Data Manipulation Language (DML)
- SELECT ... FROM ...
- INSERT INTO ... VALUES
-UPDATE .. SET .. WHERE
-DELETE .. FROM .. WHERE

DCL : Data control language

GRANT && REVOKE
GRANT: gives (or grant) certain permissions to users
one can grant a specific type of permission , like complete or parial access
GRANT type of permission on database_name.tablename to 'username@localhost';
These rights will be assigned to a person who has a user name registered at the local server (local host : IP 129.0.0.1)

Revoke is quite opposite to Grant

DDL -- creation of data
DML -- manipulation of data
DCL-- assignment and removal of permissions to use this data

TCL -- saving and restoring changes to a database

Operators :

1. IN --- The IN operator allows SQL to return the names written in parentheses , if they exist in our table .

2. LIKE - UNLIKE --- Pattern matching -- %

Ex:- Like ('Mar%') -- Name contains first three letters with Mar
ex:- Like('%Mar') -- ending names with Mar
Ex:- like ('%Mar%') -- it returns Mar in anywhere in that word

3 . _ == helps you match a single characters
Ex:- Like('mar_') -- returns first latter's with Mar along with next letter

Wildcard characters:-
% , _ , *
% a substitute for a sequence of characters
_ helps you match a single character
* will deliver a list of all columns of a table

Between… And…

- Not used only for data values.
- Cloud also be applied to strings and numbers .

GROUP BY must be placed immediately after the WHERE Condition , If any , and just before the ORDER BY clause

Where >>> Group BY >>> Order By

In Most cases , when you need an aggregate function, You must add a GROUP BY Clause in your query , too.

HAVING:

1. Refines the output from records that do not satisfy a certain conditions
2. frequently implemented with Group by
3. WHERE >> GROUP BY >> HAVING >>> ORDER BY
4. After HAVING, you can have a condition with an aggregate function, while WHERE cannot use aggregate function within its condition.
5. Aggregate functions – GROUP BY and HAVING
6. General conditions - WHERE

TCL :- TCL commands can only use with DML commands like INSERT , DELETE , and UPDATE only.

Here are some commands that comes under TCL

1. COMMIT
2. ROLLBACK
3. SAVEPOINT

A. COMMIT:- Commit command is used to save the transactions to the database
-- Syntex :- COMMIT;

B. ROLLBACK:- Rollback command is used to undo transactions that have not already been saved to the database .
-- Syntex:- ROLLBACK

C. SAVEPOINT :- It is used to roll the transaction back to a certain point without rolling back the entire transaction
--- Syntax :- SAVEPOINT SAVEPOINT_NAME;

DCL COMMANDS:- Data Control Languages

DCL Commands are used to grand and take authority from any database user
Here are some commands that come under DCL

- GRANT
- Revoke

A. GRANT:- It is used to give user access privileges (Select , update) to a database.
-- Syntex :- GRANT privileges on TABLE_NAME to USER1 , USER2;

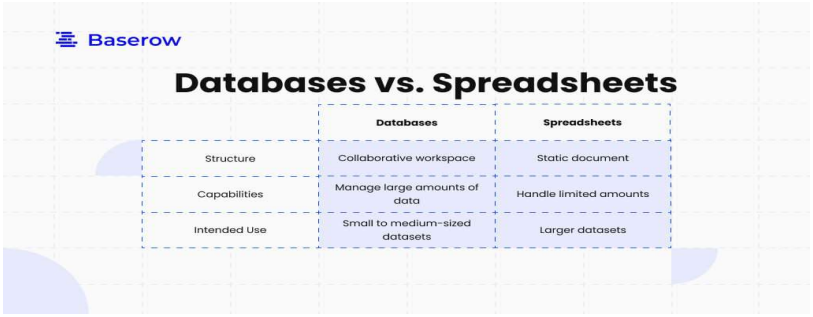
1. REVOKE:- it is used to take back permissions from the user.
-- Syntex :- REVOKE privileges on TABLE_NAME FROM USER1, USER2;

NOTE :-- privileges (Select , update) -- we can give anyone on of the privileges to the users .

Aggregate functions:-

1. Count :- here Count(*) will works and returns with includes of null values -- it will works for both numeric and non-numeric
2. SUM:- here Sum(*) will not works -- but this will works only for numeric

---- UPDATE TABLE _NAME SET WHERE
---- DELETE from TABLE _Name
----INSERT INTO TABLENAME () VALUES()



Main goal :- Organize(Compact , well- structured) huge amounts of data that can be quickly retrieved

Relationships:
Relationships tell you how much of the data from a foreign key field can be seen in the primary key column of the table the data is related to and viceversa

Unique key :
used whenever you would like to specify that you don't want to see duplicate data in a given filed

wildcard characters

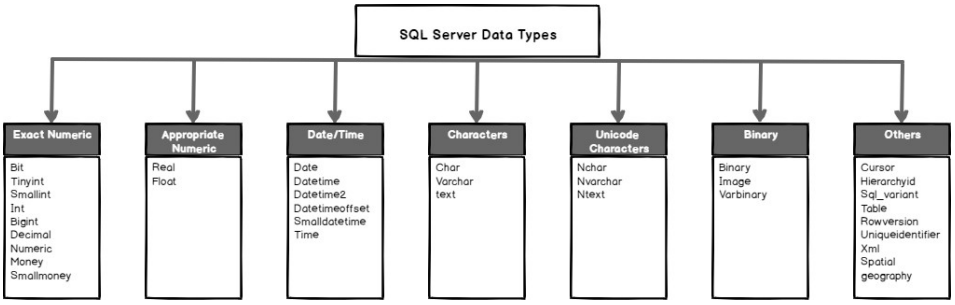
% - a substitute for a sequence of characters
_ helps you match a single charater
* will deliver a list of all column in a table

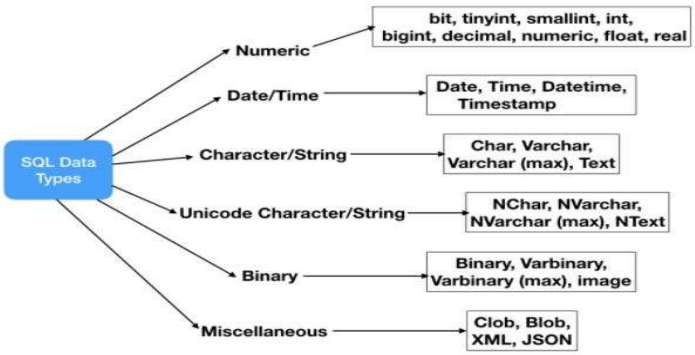
Drop:- you won't be able to roll back to its initial state, or to the last COMMIT statement

use DROP TABLE only when you are sure you aren't going to use the table in question anymore
TRUNCATE when truncating, auto-increment values will be reset

Delete removes records row by row
auto-increment values are not reset with DELETE

Data Types :-





Digits , Symbols , or blanks spaces can also use in the String format

CONSTRAINTS IN SQL :

Constraints: Constraints are used to specify conditions or rules or restrictions for data in a table . In case of any violation to these constraints , the SQL statements will fail

Below are the constraints n SQL

1. UNIQUE
2. NOT NULL
3. DEFAULT
4. CHECK
5. PRIMARY KEY
6. FOREIGN KEY

1. UNIQUE :- Unique constraints can be applied at column level, this constraints ensures that all values in that column are different and it won't allow the duplicate values . But It allows null values

2. NULL VALUE : Not null can be applied at column level, this constraint won't allow nulls into the column ., If you try to load null values the SQL statement gives error .

3. DEFAULT :- Default can be applied to column level, where we have a specify a default value for that column , for any record if we do not insert / load a value to that column then default value will be stored in that column.

4. CHECK :- Check constraints can be applied at column level , where we have to specify a condition . This allows only the value that are satisfying the condition . If the condition is false then SQL statement give error .

5. NOTE :- we can specify all above constraints at the time of table creation or can add constraints later by ALTER table statement .

6. PRIMARY KEY :-
- Primary key helps us to identify each row uniquely in a table.
- It is combination of UNIQUE and NOT NULL constraints .
- Primary key can be applied on a single column or combination of multiple columns.
- This is a table level constraints and can be added at the time of table creation or later by using ALTER table statement

7. FOREIGN KEY:
- Foreign key is used to specify referential integrity between tables.
- That means same column present in two tables, in the first table that column is a primary key and in other table it can be used as foreign key to refer the record from that other table
- Foreign key is basically used to provide the links between database tables
- We use REFERENCES key words to specify foreign key relationship

INTERVIEW Question :-
How many unique constraints and primary key constraints a table have >

Ans: A table can have any number of unique keys but only one primary key .That means we can defined multiple unique keys on different columns , But can define single primary key at table level.

ON DELETE CASCADE Constraint:- ON DELETE CASCADE constraint is used in MySQL to delete the rows from the child table automatically, when the rows from the parent table are deleted.

From <https://www.geeksforgeeks.org/mysql-on-delete-cascade-constraint/>

SQL JOIN
SQL JOIN clause is used to query and access data from multiple tables by establishing logical relationships between them. It can access data from multiple tables simultaneously using common key values shared across different tables.

We can use SQL JOIN with multiple tables. It can also be paired with other clauses, the most popular use will be using JOIN with **WHERE clause** to filter data retrieval.

SQL JOIN Example
Consider the two tables below as follows:

Student:

ROLL_NO	NAME	ADDRESS	PHONE	Age
1	HARSH	DELHI	xxxxxxxxxx	18
2	PRATIK	BIHAR	xxxxxxxxxx	19
3	RIYANKA	SILIGURI	xxxxxxxxxx	20
4	DEEP	RAMNAGAR	xxxxxxxxxx	18
5	SAPTARHI	KOLKATA	xxxxxxxxxx	19
6	DHANRAJ	BARABAJAR	xxxxxxxxxx	20
7	ROHIT	BALURGHAT	xxxxxxxxxx	18
8	NIRAJ	ALIPUR	xxxxxxxxxx	19

StudentCourse :

COURSE_ID	ROLL_NO
1	1
2	2
2	3
3	4
1	5
4	9
5	10
4	11

Both these tables are connected by one common key (column) i.e ROLL_NO.

We can perform a JOIN operation using the given SQL query:

```
SELECT s.roll_no, s.name, s.address, s.phone, s.age, sc.course_id
FROM Student s

JOIN StudentCourse sc ON s.roll_no = sc.roll_no;
```

Output:

ROLL_NO	NAME	ADDRESS	PHONE	AGE	COURSE_ID
1	HARSH	DELHI	XXXXXXXXXX	18	1
2	PRATIK	BIHAR	XXXXXXXXXX	19	2
3	RIYANKA	SILGURI	XXXXXXXXXX	20	2
4	DEEP	RAMNAGAR	XXXXXXXXXX	18	3
5	SAPTARHI	KOLKATA	XXXXXXXXXX	19	1

Types of JOIN in SQL

There are many types of Joins in SQL. Depending on the use case, you can use different type of SQL JOIN clause. Here are the frequently used SQL JOIN types:

- INNER JOIN
- LEFT JOIN
- RIGHT JOIN
- FULL JOIN
- NATURAL JOIN

SQL INNER JOIN

The **INNER JOIN** keyword selects all rows from both the tables as long as the condition is satisfied. This keyword will create the result-set by combining all rows from both the tables where the condition satisfies i.e value of the common field will be the same.

Syntax:

The syntax for SQL INNER JOIN is:

```
SELECT table1.column1,table1.column2,table2.column1,....
FROM table1

INNER JOIN table2

ON  table1.matching_column = table2.matching_column;
```

Here,

- **table1**: First table.
- **table2**: Second table
- **matching_column**: Column common to both the tables.

***Note**: We can also write JOIN instead of INNER JOIN. JOIN is same as INNER JOIN.*

INNER JOIN Example

Let's look at the example of INNER JOIN clause, and understand it's working.

This query will show the names and age of students enrolled in different courses.

```
SELECT StudentCourse.COURSE_ID, Student.NAME, Student.AGE FROM Student
INNER JOIN StudentCourse

ON Student.ROLL_NO = StudentCourse.ROLL_NO;
```

Output:

COURSE_ID	NAME	Age
1	HARSH	18
2	PRATIK	19
2	RIYANKA	20
3	DEEP	18
1	SAPTARHI	19

SQL LEFT JOIN

LEFT JOIN returns all the rows of the table on the left side of the join and matches rows for the table on the right side of the join. For the rows for which there is no matching row on the right side, the result-set will contain *null*. LEFT JOIN is also known as LEFT OUTER JOIN.

Syntax

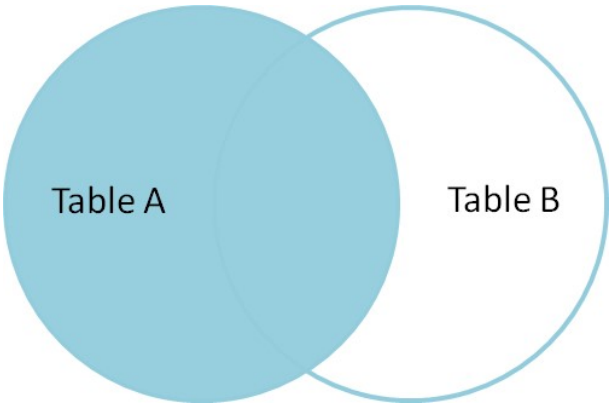
The syntax of LEFT JOIN in SQL is**:**

```
SELECT table1.column1,table1.column2,table2.column1,....
FROM table1
LEFT JOIN table2
ON table1.matching_column = table2.matching_column;
```

Here,

- **table1:** First table.
- **table2**: Second table
- **matching_column**: Column common to both the tables.

***Note**: We can also use LEFT OUTER JOIN instead of LEFT JOIN, both are the same.*



LEFT JOIN Example

Let's look at the example of LEFT JOIN clause, and understand it's working

```
SELECT Student.NAME,StudentCourse.COURSE_ID
FROM Student
LEFT JOIN StudentCourse
ON StudentCourse.ROLL_NO = Student.ROLL_NO;
```

Output:

NAME	COURSE_ID
HARSH	1
PRATIK	2
RIYANKA	2
DEEP	3
SAPTARHI	1
DHANRAJ	*NULL*
ROHIT	*NULL*
NIRAJ	*NULL*

SQL RIGHT JOIN

RIGHT JOIN returns all the rows of the table on the right side of the join and matching rows for the table on the left side of the join.It is very similar to LEFT JOIN For the rows for which there is no matching row on the left side, the result-set will contain *null*. RIGHT JOIN is also known as RIGHT OUTER JOIN.

Syntax:

The syntax of RIGHT JOIN in SQL is:

```
SELECT table1.column1,table1.column2,table2.column1,....
FROM table1
RIGHT JOIN table2
ON table1.matching_column = table2.matching_column;
```

Here,

- **table1**: First table.
- **table2**: Second table
- **matching_column**: Column common to both the tables.

Note: We can also use **RIGHT OUTER JOIN** instead of RIGHT JOIN, both are the same.

RIGHT JOIN Example:

Let's look at the example of RIGHT JOIN clause, and understand it's working

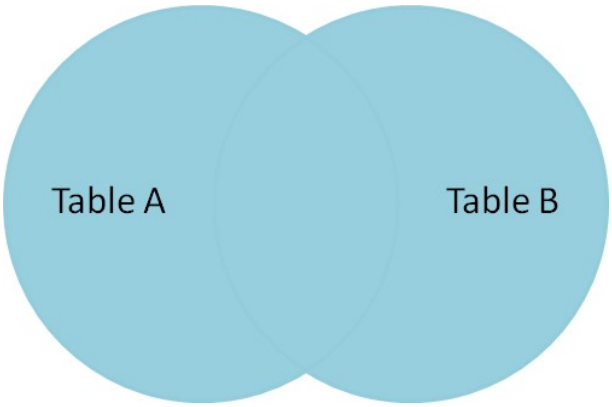
```
SELECT Student.NAME,StudentCourse.COURSE_ID
FROM Student
RIGHT JOIN StudentCourse
ON StudentCourse.ROLL_NO = Student.ROLL_NO;
```

Output:

NAME	COURSE_ID
HARSH	1
PRATIK	2
RIYANKA	2
DEEP	3
SAPTARHI	1
NULL	4
NULL	5
NULL	4

SQL FULL JOIN

FULL JOIN creates the result-set by combining results of both LEFT JOIN and RIGHT JOIN. The result-set will contain all the rows from both tables. For the rows for which there is no matching, the result-set will contain *NULL* values.



Syntax

The syntax of SQL FULL JOIN is:

```
SELECT table1.column1,table1.column2,table2.column1,....
FROM table1
FULL JOIN table2
ON table1.matching_column = table2.matching_column;
```

Here,

- **table1**: First table.
- **table2**: Second table
- **matching_column**: Column common to both the tables.

FULL JOIN Example

Let's look at the example of FULL JOIN clause, and understand it's working

```
SELECT Student.NAME,StudentCourse.COURSE_ID
FROM Student
FULL JOIN StudentCourse
ON StudentCourse.ROLL_NO = Student.ROLL_NO;
```

Output:

NAME	COURSE_ID
HARSH	1
PRATIK	2
RIYANKA	2
DEEP	3

SAPTARHI	1
DHANRAJ	NULL
ROHIT	NULL
NIRAJ	NULL
NULL	4
NULL	5
NULL	4

SQL Natural join (?)

Natural join can join tables based on the common columns in the tables being joined. A natural join returns all rows by matching values in common columns having same name and data type of columns and that column should be present in both tables.

Both table must have at least one common column with same column name and same data type.

The two table are joined using **Cross join**.

DBMS will look for a common column with same name and data type Tuples having exactly same values in common columns are kept in result.

Natural join Example:

Look at the two tables below- Employee and Department

Employee		
Emp_id	Emp_name	Dept_id
1	Ram	10
2	Jon	30
3	Bob	50

Department	
Dept_id	**Dept_name**
10	IT
30	HR
40	TIS

Problem: Find all Employees and their respective departments.

Solution Query: (Employee) ? (Department)

Emp_id	Emp_name	Dept_id	Dept_id	Dept_name
1	Ram	10	10	IT
2	Jon	30	30	HR
Employee data			Department data	

From <https://www.geeksforgeeks.org/sql-join-set-1-inner-left-right-and-full-joins/>

UNION VS UNION ALL

- UNION displays only distinct values in the output ---- for better results
- UNIONALL retrieves the duplicates as well --- Optimize performance.

VIEW:- View is a database object , it can created over an SQL Query
-- View does not store any data
-- View is like a virtual table
-- it is used to make secure access of the data
-- simple way to access the data

Why we use the VIEW :-

1. Security -- By hiding the query used to generate the view
2. To simplify complex SQL queries

1 for the security -- we have to give login and password and as well as permission .

Feature	**Views**	**Stored Procedures**	**Functions**
Definition	A virtual table based on the result-set of a SQL query.	A set of SQL statements that perform a specific task.	A set of SQL statements that return a value or a table.
Use Case	Simplify complex queries, provide data abstraction.	Encapsulate business logic, perform operations, manage transactions.	Perform calculations, return single value or table, reuse code.
Execution	Select statement on the view.	Called explicitly using EXEC or similar command.	Called as part of an expression, often used in SELECT or WHERE.
Return Type	Always returns a result set (table).	Can return zero or more result sets, and output parameters.	Returns a scalar value, table, or composite data type.
Parameters	Cannot accept parameters (in standard SQL).	Can accept input, output, and input-output parameters.	Can accept input parameters, limited in some SQL dialects.
Compilation	Not precompiled, executed as part of the query.	Precompiled and stored in the database.	Precompiled, may offer slight performance advantage over raw SQL.
Transactions	Cannot manage transactions.	Can begin, commit, or rollback transactions.	Cannot begin or commit transactions directly.
DML	Generally read-only, but can be updatable in	Can perform any DML operations (INSERT,	Generally used for calculations and data retrieval, some

Operations	some cases.	UPDATE, DELETE).	DML possible in certain SQL dialects.
Side Effects	No side effects, just a select query.	Can have side effects, such as modifying database state.	Typically no side effects, used for computations and returning results.
Reusability	Used to simplify complex queries and promote reusability.	Highly reusable, can be called from applications and other procedures.	Reusable, often used to encapsulate common logic in queries.
Security	Can provide a level of abstraction and restrict access to underlying data.	Can include security logic and enforce access controls.	Can enforce business rules and validation logic.
Maintenance	Easier to maintain views that encapsulate complex joins and logic.	Can centralize and maintain business logic separately from application code.	Easier to maintain logic for calculations and data manipulations.