Database

Table of contents

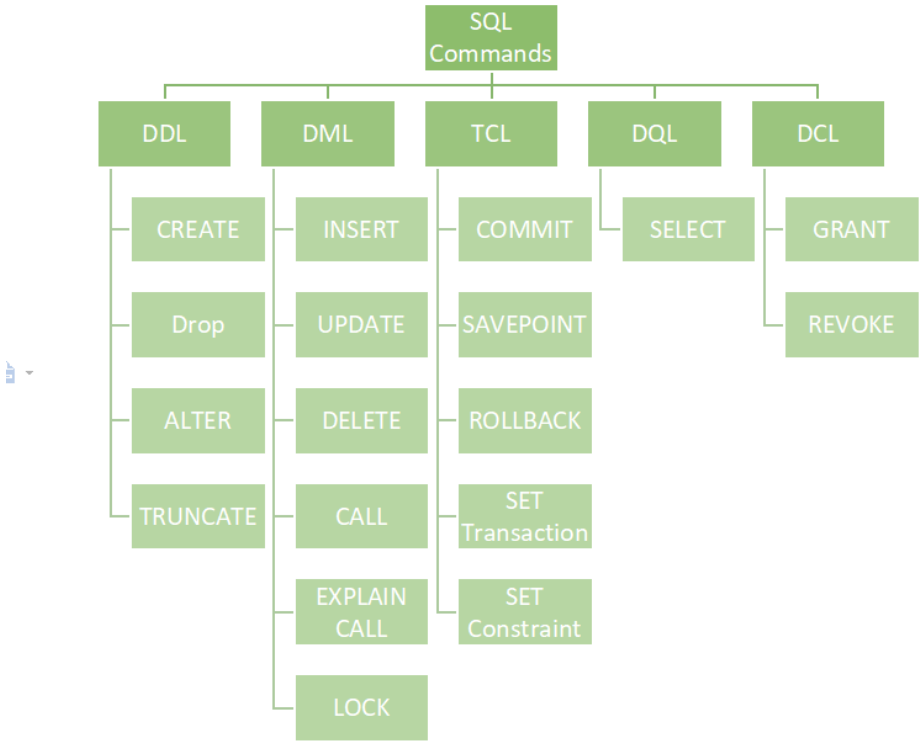
Database

▼ Installation

- sudo apt-get install mysql-server
- Install workbench from website using deb

- sudo mysql -uroot -p
    - ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY 'Mind@123';

▼ SQL DML, DDL, DCL, and TCL Commands



What are SQL DML, DDL, DCL, and TCL Commands, and their abbreviations are the most common question you might face in SQL interviews.

The Structured Query Language or SQL is all about executing certain commands against the database or creating it. In SQL Server, all these commands are categorically organized into four categories, and they are DML, DDL, DCL, and TCL Commands.

DDL → command table/database name;

DML → command name; //no need to mention **table as it is manipulating table**

SQL DDL Commands

In SQL, DDL means **Data Definition Language**. The Sql Server DDL commands are used to create and modify the structure of a database and database objects.

Examples of Sql Server DDL commands are

The following are the examples of the DDL commands

1. CREATE – Create an object. I mean, create a database, table, triggers, index, functions, stored procedures, etc.
2. DROP – This SQL DDL command helps to delete objects. For example, delete tables, delete a database, etc.
3. ALTER – Used to alter the existing database or its object structures.
4. TRUNCATE – This SQL DDL command removes records from tables
5. RENAME – Renaming the database objects

SQL DML Commands

DML means **Data Manipulation Language** in Sql Server. As its name suggests, these Sql Server DML commands will perform data manipulation manipulate data presented in the server).

Examples of DML commands in SQL Server are

1. SELECT – This SQL DML command select records or data from a table
2. INSERT – Insert data into a database table.
3. UPDATE – This SQL DML command will update existing records within a table
4. DELETE – Delete unwanted records from a table

SQL DCL Commands

The Sql Server DCL means **Data Control Language**. These DCL commands in SQL Server will control the Data access permission.

Sql Server DCL commands Examples are

- GRANT – It permits users to access the database.
- REVOKE – This SQL DCL command withdraws the permission given by GRANT to access the database.

SQL TCL Commands

TCL means **Transaction Control Language**. These Sql Server TCL commands will control the Transactions. Please refer SQL interviews.

Examples of TCL commands in SQL Server are

- COMMIT – This SQL TCL command will commit the running transaction
- ROLLBACK – Rollback the current transaction
- SAVEPOINT – You can set a save point so that, next time it will start from here
- SET TRANSACTION – Specify the characteristics of the transactions

▼ MySQL Data Types

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

MySQL String Data Types

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.

MySQL Numeric Data Types

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.

MySQL Date and Time Data Types

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.

▼ char vs varchar

For `VARCHAR` columns, trailing spaces in excess of the column length are truncated prior to insertion and a warning is generated, regardless of the SQL mode in use. For `CHAR` columns, truncation of excess trailing spaces from inserted values is performed silently regardless of the SQL mode.

`VARCHAR` values are not padded when they are stored. Trailing spaces are retained when values are stored and retrieved, in conformance with standard SQL.

The following table illustrates the differences between `CHAR` and `VARCHAR` by showing the result of storing various string values into `CHAR(4)` and `VARCHAR(4)` columns (assuming that the column uses a single-byte character set such as `latin1`).

Value	CHAR(4)	Storage Required 1	VARCHAR(4)	Storage Required
`''`	`'  '`	4 bytes	`''`	1 byte
`'ab'`	`'ab '`	4 bytes	`'ab'`	3 bytes
`'abcd'`	`'abcd'`	4 bytes	`'abcd'`	5 bytes
`'abcdefgh'`	`'abcd'`	4 bytes	`'abcd'`	5 bytes

Using char over varchar

The general rule is to **pick CHAR if all rows will have close to the same length**. Pick VARCHAR (or NVARCHAR) when the length varies significantly. CHAR may also be a bit faster because all the rows are of the same length.

▼ Operator

SQL Operator Symbols	Operators
**	Exponentiation operator
+, -	Identity operator, Negation operator
*, /	Multiplication operator, Division operator
+, -, ||	Addition (plus) operator, subtraction (minus) operator, String Concatenation operator
=, !=, <, >, <=, >=, IS NULL, LIKE, BETWEEN, IN	Comparison Operators
NOT	Logical negation operator
&& or AND	Conjunction operator
OR	Inclusion operator

Types of Operator

SQL operators are categorized in the following categories:

1. SQL Arithmetic Operators [+, -, *, /, %]
2. SQL Comparison Operators [=, ! = , >, > =, <, < =]
3. SQL Logical Operators [ALL, AND, OR, BETWEEN, IN, NOT, ANY, LIKE]
4. SQL Set Operators [UNION, UNION ALL, INTERSECT, MINUS]
5. SQL Bit-wise Operators [AND(&), OR(|)]
6. SQL Unary Operators [unary positive, unary negative, unary bitwise]

▼ Copy database

Now, open a DOS or terminal window to access the MySQL server on the command line. For example, if we have installed the MySQL in the **C folder**, copy the following folder and paste it in our DOS command. Then, press the **Enter** key.

1. C:\Users\javatpoint> CD C:\Program Files\MySQL\MySQL Server 8.0\bin

In the next step, we need to use the mysqldump tool to copy the database objects and data into the SQL file. Suppose we want to dump (copy) the database objects and data of the testdb into an SQL file located at **D:\Database_backup folder.** To do this, execute the below statement:

1. mysqldump -u root -p testdb > D:\Database_backup\testdb.sql
2. Enter **password**: **********

▼ Creating table and not null

```
1. mysql> CREATE TABLE employee_table(
2.     id int NOT NULL AUTO_INCREMENT,
3.     name varchar(45) NOT NULL,
4.     occupation varchar(35) NOT NULL,
5.     age int NOT NULL,
6.     PRIMARY KEY (id)
7. );
```

NOTE:1. Here, NOT NULL is a field attribute, and it is used because we don't want this field to be NULL. If we try to create a record with a NULL value, then MySQL will raise an error.2. The field attribute AUTO_INCREMENT specifies MySQL to go ahead and add the next available number to the id field. PRIMARY KEY is used to define a column's uniqueness. We can use multiple columns separated by a comma to define a primary key.

▼ SHOW FULL TABLES

We can also use the **FULL modifier** with the SHOW TABLES query to get the type of table (Base or View) that appears in a second output column.

mysql> SHOW **FULL** TABLES;

▼ Rename table

NOTE: If we use the RENAME TABLE statement, it is required to have ALTER and DROP TABLE privileges to the existing table. Also, this statement cannot change the name of a temporary table.

▼ Transaction

Transaction represents **a single unit of work**.

The ACID properties describes the transaction management well. ACID stands for Atomicity, Consistency, isolation and durability.

Atomicity means either all successful or none.

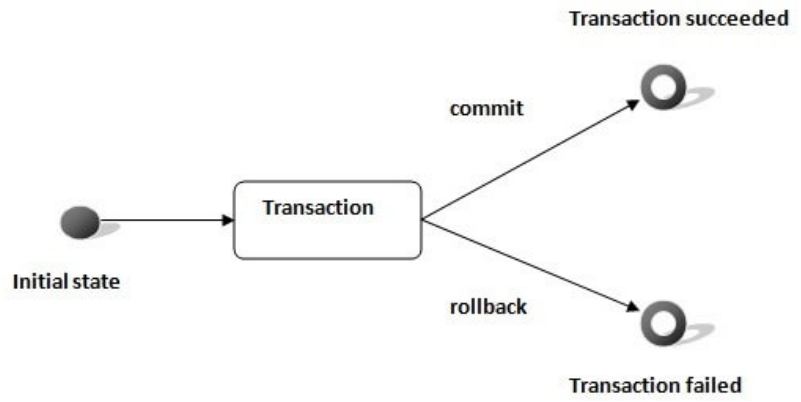
Consistency ensures bringing the database from one consistent state to another consistent state.

Isolation ensures that transaction is isolated from other transaction.

Durability means once a transaction has been committed, it will remain so, even in the event of errors, power loss etc.

Advantage of Transaction Mangement

fast performance It makes the performance fast because database is hit at the time of commit.



In JDBC, **Connection interface** provides methods to manage transaction.

Method	Description
void setAutoCommit(boolean status)	It is true bydefault means each transaction is committed bydefault.
void commit()	commits the transaction.
void rollback()	cancels the transaction.

▼ Rollback

Some statements cannot be rolled back. In general, these include data definition language (DDL) statements, such as those that create or drop databases, those that create, drop, or alter tables or stored routines.

We can rollback a delete query but not so for truncate and drop.

DROP: database, table, column

TRUNCATE: remove all entries

DELETE: remove given row or rows

If you have not committed INSERT INTO statements or UPDATE statement, then you have a chance to roll data back to the original state.

In work bench

COMMIT command in SQL is used to save all the transaction-related changes permanently to the disk. Whenever DDL commands such as INSERT, UPDATE and DELETE are used, the **changes made by these commands are permanent only after closing the current session**. So before closing the session, one can easily roll back the changes made by the DDL commands. Hence, if we want the changes to be saved permanently to the disk without closing the session, we will use the commit command.

▼ Truncate vs delete

The TRUNCATE command works the same as a DELETE command without using a WHERE clause that deletes complete rows from a table. However, the TRUNCATE command is more efficient as compared to the DELETE command because it removes and recreates the table instead of deleting single records one at a time. Since this command internally drops the table and recreates it, the number of rows affected by the truncate statement is zero, unlike the delete statement that returns the number of deleted rows.

> The Truncate command **resets AUTO_INCREMENT** counters on the table.

> MySQL truncates the table by dropping and creating the table. Thus, the DELETE triggers for the table do not fire during the truncation.

This command does not maintain the transaction log during the execution. It deallocates the data **pages instead of rows** and makes an entry for the deallocating pages instead of rows in transaction logs. This command also locks the pages instead of rows; thus, it requires fewer locks and resources.

The following points must be considered while using the TRUNCATE command:

- We cannot use the **WHERE** clause with this command so that filtering of records is not possible.
- We **cannot rollback the deleted data** after executing this command because the log is not maintained while performing this operation.
- We cannot use the truncate statement when a table is referenced by a **foreign key** or participates in an **indexed view**.

- The TRUNCATE command doesn't fire DELETE **triggers** associated with the table that is being truncated because it does not operate on individual rows.

▼ How to Truncate Table with Foreign key?

If we perform the TRUNCATE operation for the table that uses a foreign key constraint, we will get the following error:

1. ERROR 1217 (23000): Cannot **delete** or **update** a parent row: a **foreign key constraint** fails

In that case, we need to log into the MySQL server and **disable foreign key** checks before executing the TRUNCATE statement as below:

1. **SET** FOREIGN_KEY_CHECKS=0;

Now, we are able to truncate tables. After execution, **re-enable foreign key** checks as given below:

1. **SET** FOREIGN_KEY_CHECKS=1;

▼ on delete cascade on update cascade

A very good thread on this subject is to be found here and also here. The definitive guide for MySQL is, of course, the documentation, to be found here.

In the SQL 2003 standard there are 5 different referential actions:

1. CASCADE
2. RESTRICT
3. NO ACTION
4. SET NULL
5. SET DEFAULT

To answer the question:

1. **CASCADE**
   - `ON DELETE CASCADE` means that if the parent record is deleted, any child records are also deleted. This is **not** a good idea in my opinion. You should keep track of all data that's ever been in a database, although this can be done using `TRIGGER`s. (However, see caveat in comments below).
   - `ON UPDATE CASCADE` means that if the parent primary key is changed, the child value will also change to reflect that. Again in my opinion, not a great idea. If you're changing `PRIMARY KEY`s with any regularity (or even at all!), there is something wrong with your design. Again, see comments.

- `ON UPDATE CASCADE ON DELETE CASCADE` means that if you `UPDATE` **OR** `DELETE` the parent, the change is cascaded to the child. This is the equivalent of `AND`ing the outcomes of first two statements.

2. **RESTRICT**

   - `RESTRICT` means that any attempt to delete and/or update the parent will fail throwing an error. This is the default behaviour in the event that a referential action is not explicitly specified.

     > For an ON DELETE or ON UPDATE that is not specified, the default action is always RESTRICT`.

3. **NO ACTION**

   - `NO ACTION`: From the manual. A keyword from standard SQL. In MySQL, equivalent to `RESTRICT`. The MySQL Server rejects the delete or update operation for the parent table if there is a related foreign key value in the referenced table. Some database systems have deferred checks, and `NO ACTION` is a deferred check. In MySQL, foreign key constraints are checked immediately, so `NO ACTION` is the same as `RESTRICT`.

4. **SET NULL**

   - `SET NULL` - again from the manual. Delete or update the row from the parent table, and set the foreign key column or columns in the child table to `NULL`. This is not the best of ideas IMHO, primarily because there is no way of "time-travelling" - i.e. looking back into the child tables and associating records with `NULL`s with the relevant parent record - either `CASCADE` or use `TRIGGER`s to populate logging tables to track changes (but, see comments).

5. **SET DEFAULT**

   - `SET DEFAULT`. Yet another (potentially very useful) part of the SQL standard that MySQL hasn't bothered implementing! Allows the developer to specify a value to which to set the foreign key column(s) on an UPDATE or a DELETE. InnoDB and NDB will reject table definitions with a `SET DEFAULT` clause.

As mentioned above, you should spend some time looking at the documentation, here.

▼ Duplicating table

1. **CREATE TABLE** IF NOT EXISTS new_table_name
2. **SELECT** column1, column2, column3
3. **FROM** existing_table_name
4. **WHERE** condition;

It is to be noted that this statement only copies the table and its data. It doesn't copy all dependent objects of the table, such as indexes, triggers, primary key constraints, foreign key constraints, etc.

So the **command of copying data along with its dependent objects** from an existing to the new table can be written as the following statements:

1. **CREATE TABLE** IF NOT EXISTS new_table_name LIKE existing_table_name;
2.
3. **INSERT** new_table_name **SELECT** * **FROM** existing_table_name;

▼ Key points before deleting a column

MySQL works with relational databases where the schema of one table can depend on the columns of another table. So when we remove a column from one table, it will effects all dependent tables also. Consider the below points while removing column:

- When we remove columns from a table, it will affect all associated objects such as triggers, stored procedures, and views. Suppose we delete a column that is referencing in the trigger. After removing the column, the trigger becomes invalid.
- The dropped column depends on other applications code, must also be changed, which takes time and effort.
- When we remove a column from the large table, it will affect the database's performance during removal time.

▼ Select

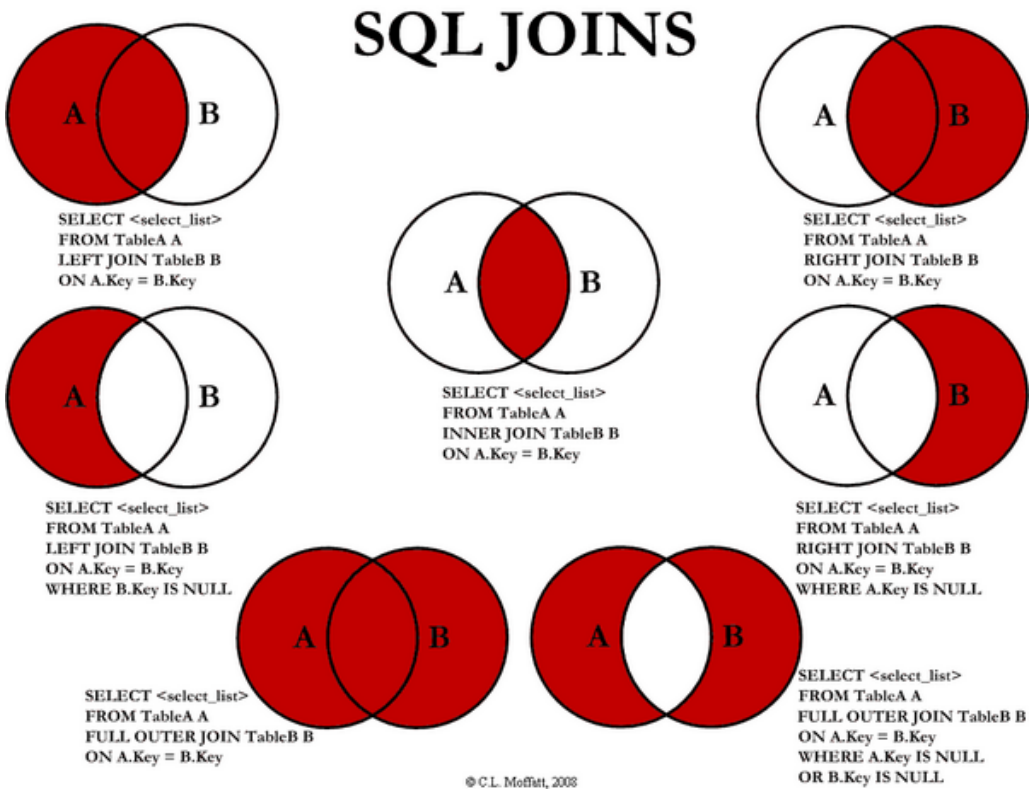
```
SELECT field_name1, field_name2,... field_nameN
FROM table_name1, table_name2...
[WHERE condition]
[GROUP BY field_name(s)]
[HAVING condition]
[ORDER BY field_name(s)]
[OFFSET M ][LIMIT N];
```

The SELECT statement uses the following parameters:

Parameter Name	Descriptions
field_name(s) or *	It is used to specify one or more columns to returns in the result set. The asterisk (*) returns all fields of a table.
table_name(s)	It is the name of tables from which we want to fetch data.
WHERE	It is an optional clause. It specifies the condition that returned the matched records in the result set.
GROUP BY	It is optional. It collects data from multiple records and grouped them by one or more columns.

Parameter Name	Descriptions
HAVING	It is optional. It works with the GROUP BY clause and returns only those rows whose condition is TRUE.
ORDER BY	It is optional. It is used for sorting the records in the result set.
OFFSET	It is optional. It specifies to which row returns first. By default, It starts with zero.
LIMIT	It is optional. It is used to limit the number of returned records in the result set.

▼ Join



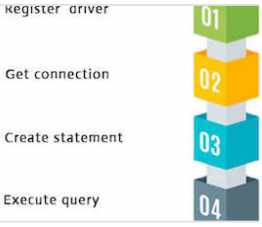
▼ JDBC

▼ 5 steps

How to connect to database in Java | Java Database Connectivity - javatpoint

There are 5 steps to connect any java application with the database using JDBC. These steps are as follows: Register the Driver class Create connection Create statement Execute queries Close connection The forName() method of Class class is

https://www.javatpoint.com/steps-to-connect-to-the-database-in-java



▼ Program

Java Database Connectivity with MySQL - javatpoint

To connect Java application with the MySQL database, we need to follow 5 following steps. In this example we are using MySql as the database. So we need to know following informations for the mysql database: Driver class: The driver class for the mysql database is com.mysql.jdbc.Driver.

https://www.javatpoint.com/example-to-connect-to-the-mysql-database

▼ JDBC methods

The execute() method: This method is used to execute SQL **DDL statements**, it returns a boolean value specifying weather the ResultSet object can be retrieved.

executeUpdate(): This method is used to execute statements such as **insert, update, delete**. It returns an integer value representing the number of rows affected.

executeQuery(): This method is used to execute statements that **returns tabular data** (example select). It returns an object of the class ResultSet.

▼ getConnection

public static Connection getConnection(String url,String userName,String password) throws SQLException: is used to establish the connection with the specified url, username, and password. The SQLException is thrown when the corresponding Driver class of the given database is not registered with the DriverManager.

▼ Batch processing

Instead of executing a single query, we can execute a batch (group) of queries. It makes the performance fast. It is because when one sends multiple statements of SQL at once to the database, the communication overhead is reduced significantly, as one is not communicating with the database frequently, which in turn results to fast performance.

Batch Processing in Java JDBC - javatpoint

Instead of executing a single query, we can execute a batch (group) of queries. It makes the performance fast. It is because when one sends multiple statements of SQL at once to the database, the communication overhead is reduced significantly, as one is not communicating with the database frequently, which in turn results to fast performance.

https://www.javatpoint.com/batch-processing-in-jdbc

▼ Callable statement usage

CallableStatement interface is used to call the **stored procedures and functions**.

We can have business logic on the database by the use of stored procedures and functions that will make the performance better because these are precompiled.

Suppose you need the get the age of the employee based on the date of birth, you may create a function that receives date as the input and returns age of the employee as the output.

▼ Union

MySQL Union is an operator that allows us to combine two or more results from multiple SELECT queries into a single result set. It comes with a default feature that removes the **duplicate** rows from the result set. MySQL always uses the name of the column in the first SELECT statement will be the column names of the result set(output).

MySQL Union must follow these basic rules:

- The **number and order of the columns should be the same** in all tables that you are going to use.
- The data type must be compatible with the corresponding positions of each select query.
- The column name selected in the different SELECT queries must be in the same order.

```
SELECT column_list FROM table1
UNION
SELECT column_list FROM table2;
```

We can understand the Union operator with the following visual representation:

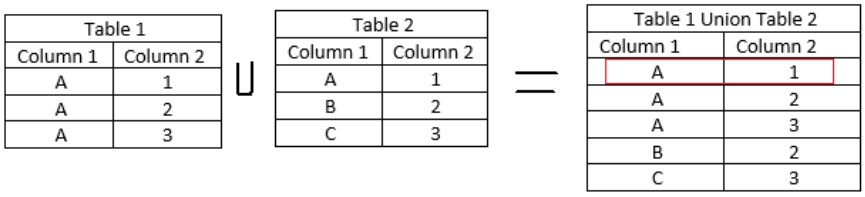


Table 1	
Column 1	Column 2
A	1
A	2
A	3

∪

Table 2	
Column 1	Column 2
A	1
B	2
C	3

=

Table 1 Union Table 2	
Column 1	Column 2
A	1
A	2
A	3
B	2
C	3

UNION VS UNION ALL

This operator returns all rows by combining two or more results from multiple SELECT queries into a single result set. It does not remove the duplicate rows from the result set.

We can understand it with the following pictorial representation:

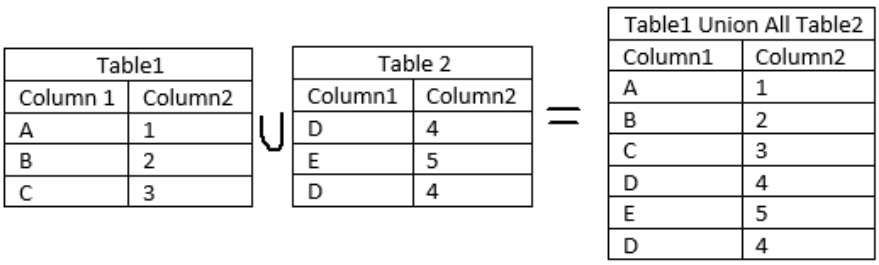


Table1	
Column 1	Column2
A	1
B	2
C	3

∪

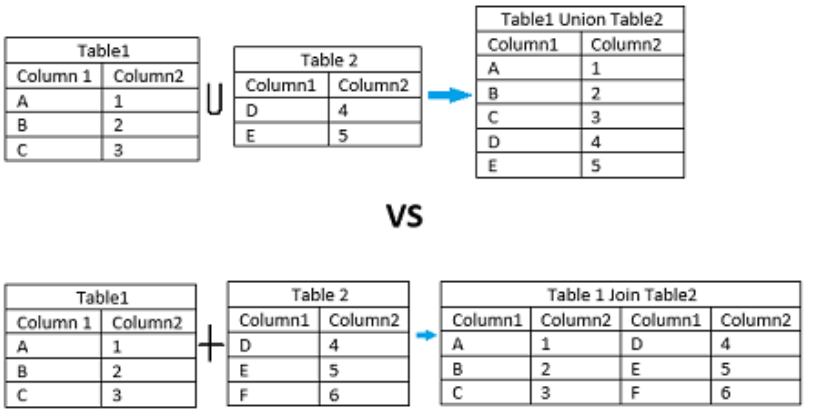
Table 2	
Column1	Column2
D	4
E	5
D	4

=

Table1 Union All Table2	
Column1	Column2
A	1
B	2
C	3
D	4
E	5
D	4

UNION VS JOIN

The Union and Join clause are different because a union always combines the result set **vertically** while the join appends the output **horizontally**. We can understand it with the following visual representation:



UNION with ORDER BY

If you want to sort the result returned from the query using the union operator, you need to use an ORDER BY clause in the last SELECT statement. We can put each SQL SELECT query in the parenthesis and then use the ORDER BY clause in the last SELECT statement as shown in the following example:

1. (**SELECT** stud_name, subject, marks **FROM** students)
2. **UNION**
3. (**SELECT** stud_name, subject, marks **FROM** student2)
4. **ORDER BY** marks;

▼ MySQL boolean and if()

MySQL Boolean - javatpoint

A Boolean is the simplest data type that always returns two possible values, either true or false. It can always use to get a confirmation in the form of YES or No value. MySQL does not contain built-in Boolean or Bool data type. They

https://www.javatpoint.com/mysql-boolean

udentid	name	complete
	Peter	true
	John	false
	Miller	true

▼ Like

```
expression LIKE pattern [ ESCAPE 'escape_character' ]
```

Parameters

expression: It specifies a column or field.

pattern: It is a character expression that contains pattern matching.

escape_character: It is optional. It allows you to test for literal instances of a wildcard character such as % or _. If you do not provide the escape_character, MySQL assumes that "\" is the escape_character.

- % : indicate any number of character

  SELECT officer_name **FROM** officers **WHERE** address LIKE 'Bh%';

  Eg. Bharuch, Bharatpur

- _ : indicate only one character

  SELECT officer_name **FROM** officers **WHERE** address LIKE 'Luc_now';

  Eg. Lucknow

- SELECT officer_name FROM officers WHERE address NOT LIKE 'Luck%';

▼ IN

```
SELECT * FROM officers WHERE officer_name IN ('Ajeet', 'Vimal', 'Deepika');
```

similar to

```
SELECT *
FROM officers
WHERE officer_name = 'Ajeet'
OR officer_name = 'Vimal'
OR officer_name = 'Deepika';
```

It also produces the same result. So IN condition is preferred over OR condition because it has minimum number of codes.

▼ ANY

```
SELECT colm1 FROM table1 WHERE colm1 > ANY (SELECT colm1 FROM table2);
```

Suppose **table1** has a row that contains a **number (10)**. In such a case, the above expression returns **true** if **table2** contains (20, 15, and 6). It is because there is a value 6 in table2, which is

less than 10. This expression returns **false** if table2 contains (15, 20), or if table2 is **empty**. If all the table fields contain (NULL, NULL, NULL), this expression is **unknown**.

The word **SOME** in MySQL can be an alias for ANY. Therefore, these two SQL statements are equivalent:

```
SELECT colm1 FROM table1 WHERE colm1 <> ANY (SELECT colm1 FROM table2);
SELECT colm1 FROM table1 WHERE colm1 <> SOME (SELECT colm1 FROM table2);
```

Advantages of ANY operator in MySQL

- ANY is a logical operator that returns the Boolean value. It allows us to select any or some rows of the SELECT statement.
- Since comparison operators precede this operator, it always returns TRUE if any subqueries satisfy the specified condition.
- It provides the result, which is a unique column value from a table that matches any record in the second table.
- We can perform several comparisons using ANY operator with the SELECT and WHERE keywords.

▼ EXISTS

If a subquery returns any rows at all, `EXISTS` ***`subquery`*** is `TRUE`, and `NOT EXISTS` ***`subquery`*** is `FALSE`. For example:

```
SELECT column1 FROM t1 WHERE EXISTS (SELECT * FROM t2);
```

Traditionally, an EXISTS subquery starts with SELECT *, but it could begin with SELECT 5 or SELECT column1 or anything at all. MySQL ignores the SELECT list in such a subquery, so it makes no difference.

> MySQL :: MySQL 8.0 Reference Manual :: 13.2.11.6 Subqueries with EXISTS or NOT EXISTS
>
> If a subquery returns any rows at all, is TRUE, and is FALSE. For example: SELECT column1 FROM t1 WHERE EXISTS (SELECT * FROM t2); Traditionally, an EXISTS subquery starts with SELECT *, but it could begin with SELECT 5 or SELECT column1 or anything at all.
>
> https://dev.mysql.com/doc/refman/8.0/en/exists-and-not-exists-subqueries.html

▼ DISTINCT

When you specify multiple columns in the `DISTINCT` clause, the `DISTINCT` clause will use the combination of values in these columns to determine the uniqueness of the row in the result set.

▼ DONE QUERIES WEBSITE

50 SQL Query Questions You Should Practice for Interview

If you want to improve SQL skills, then install a SQL package like MySQL and start practicing with it. To get you started, we've outlined a few SQL query questions in this post. Solving practice questions is the fastest way to learn any

https://www.techbeamers.com/sql-query-questions-answers-for-practice/

Part 1 to Part 9

https://stackhowto.com/mysql-practice-exercises-with-solutions-part-1/

▼ MySql summary

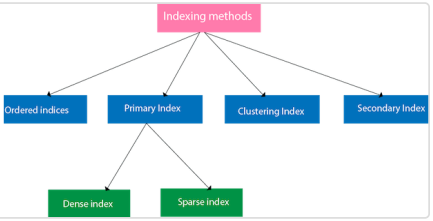
```
create database trial;
use trial;

-- ddl
create table table1(id int primary key, name varchar(45));
drop table table1;
truncate table table1;
alter table table1 add column address varchar(100);
alter table table1 drop column address;         -- change & rename

-- dml
insert into table1(id, name) values (1,"abc"), (2, "xyz");
update table1 set id = 3 where name = "xyz";
delete from table1 where id = 1;

select * from table1;
describe table1;
```

▼ Concept of indexing

DBMS Indexing in DBMS - javatpoint

DBMS Indexing in DBMS with DBMS Overview, DBMS vs Files System, DBMS Architecture, Three schema Architecture, DBMS Language, DBMS Keys, DBMS Generalization, DBMS Specialization, Relational Model concept, SQL

https://www.javatpoint.com/indexing-in-dbms



- [x] ~~DDL operations~~
  - [x] ~~Create, show~~
  - [x] ~~Drop~~
  - [x] ~~Alter~~
  - [x] ~~Truncate~~
- [x] ~~DML operations~~
  - [x] ~~Select~~
  - [x] ~~Insert~~
  - [x] ~~Update~~
  - [x] ~~Delete~~
- [x] ~~TCL~~
- [ ] DCL
- [x] ~~Understand prepareStatement and executeQuery and execute~~
- [x] ~~Query including joins~~

Doubt:

- [x] ~~char vs varchar~~