- execution is top to bottom, left to right

```
SELECT deptno, job, ename, sal, hiredate from emp;
```

- When you insert a row into the table, wherever MySQL finds the free space, it will store the row there
- in all RDBMS, the rows inside the table are not stored sequentially
- in all RDBMS, the rows inside the table are scattered (fragmented) all over the DB server HD
- When you SELECT from a table, the order of rows in the output depends on the row address
- When you SELECT from a table, the order of rows in the output will always be in ascending order of row address
- in future if you UPDATE the row, if the row length is increasing, the row address MAY change
- if the free space is not available, then the row address MAY change
- Hence there's no way of finding out the first 'N' rows inside a table or te last 'N' rows inside the table

---

ORDER

- used for sorting
    - asc → ascending (by default)
    - desc → descending

```
SELECT deptno, job, ename, sal, hiredate from emp
order by ename asc;

SELECT deptno, job, ename, sal, hiredate from emp
order by ename desc;

SELECT deptno, job, ename, sal, hiredate from emp
order by deptno asc;

SELECT deptno, job, ename, sal, hiredate from emp
order by deptno desc;

SELECT deptno, job, ename, sal, hiredate from emp
order by hiredate;

SELECT deptno, job, ename, sal, hiredate from emp
where deptno = 10
order by ename;
```

```
SELECT deptno, job, ename, sal, hiredate from emp
order by ename
where deptno = 10;                                    <- ERROR
```

- WHERE clause is specified BEFORE the ORDER BY clause
- WHERE clause is used for searching
- searching takes place in DB server HD
- WHERE clause is used to restrict the rows
- WHERE clause is used to retrieve the rows from DB Server HD to server RAM
- ORDER BY sorting takes place AFTERWARDS in Server RAM

Sorting by multiple columns -

```
SELECT deptno, job, ename, sal, hiredate from emp
ORDER BY deptno, job;

SELECT deptno, job, ename, sal, hiredate from emp
ORDER BY deptno desc, job;

SELECT deptno, job, ename, sal, hiredate from emp
ORDER BY deptno desc, job desc;
```

- No upper limit on the number of columns in ORDER BY clause
- if you have large number of rows in the table, and if you have large number of columns in order by clause, then your select statement will be slow; because that much sorting has to take place in server RAM
- Sorting is one operation which always slows down the SELECT statement

```
select ename, sal*12 from emp;

select ename, sal*12 from emp
order by sal*12;

select ename, sal*12 as "ANNUAL" from emp
order by sal*12;

select ename, sal*12 as annual from emp
order by annual;
```

- alias can be used in ORDER BY clause but not WHERE clause
- ORDER BY clause is LAST statement in SELECT statement

ORDER BY column number

```
select ename, sal*12 as annual from emp
order by 2;
```

- order by 2; → order by 2nd column

blank-padded comparison semantics :
When you compare 2 string of different lengths, the shorter of the two strings is temporirily padded with blank spaces on RHS such that there lengths become equal; then
it will start the comparison, character by character, based on ASCII value

```
SELECT * from emp
WHERE ename >= 'A' and ename < 'B';
```

```
SELECT * from emp
WHERE ename like 'A%';
```

```
                -> Starting with A
```

WILDCARD
- % → any character and any number of characters
- _ → any 1 character

To make is case-insensitive, Solution for Oracle :
Wildcard %

```
select * from emp
where ename like 'A%' or ename like 'a%';

SELECT * from emp
WHERE ename like '%A';                  -> Ending with A

SELECT * from emp
WHERE ename like '%A%';                 -> Contains with A
```

Wildcard _

```
SELECT * from emp
WHERE ename like '__A%';                -> 3rd character should be A

SELECT * from emp
WHERE ename like '____';                -> any ename with 4 characters

SELECT * from emp
WHERE ename like '_I__';                -> ename with 4 characters where 2nd character is
I
```

```
SELECT * from emp
WHERE ename = 'A%';                     -> will check for ename = 'A%'

SELECT * from emp
```

```
WHERE ename not like 'A%';              -> ename not starting with A

SELECT * from emp
WHERE sal >= 2000 and sal <= 3000;      -> sal >= 2000 and sal <= 3000

SELECT * from emp
WHERE sal between 2000 and 3000;        -> inclusive
        ** above 2 will give same output
        ** 2nd one will run faster

SELECT * from emp
WHERE sal not between 2000 and 3000;    -> exclusive

SELECT * from emp
WHERE hiredate between '2021-01-01' and '2021-12-31';

SELECT * from emp
WHERE ename between 'A' and 'F';
```

ANY and IN Operator

```
select * from emp
where deptno = 10 or deptno = 20 or deptno = 30;
```

above query can be written as using ANY :

```
select * from emp
where deptno = any(10, 20, 30);
        - easier to write
        - works faster
select * from emp
where deptno != any(10, 20, 30);

select * from emp
where deptno >= any(10, 20, 30);

select * from emp
where deptno <= any(10, 20, 30);
```

above query can be written as using IN :

```
select * from emp
where deptno in(10, 20, 30);
        - fastest
select * from emp
where deptno not in(10, 20, 30);
```

ANY - perform Logical OR
IN - perform Logical OR
- IN operator is faster than ANY operator
- ANY operator is more powerfull than IN operator
- with IN operator, you can only check for IN and NOT IN
- with ANY operator , you can check for =ANY, !=ANY, >=ANY, <=ANY, <ANY, >ANY
- If you want to check for equality or inequality, then use IN operator
- If you want to check for >, <, >=, <= then use ANY operator

---

UPDATE

```
update emp
set sal = 10000
where empno = 1;

update emp
set sal = sal + sal*0.4
where empno = 1;

update emp
set sal = 10000, city = 'Jalgaon'
where city = 'Mumbai';
```

- you can update multiple rows and multiple columns simultaneously but only one table at a time
- if you want to update two or more tables, then separate UPODATE command is required for each table

```
update emp
set sal = 10000;        - all the rows will be updated where sal will be 10000
```

DELETE

```
delete from emp
where empno = 1;

delete from emp
where city = 'Mumbai';

delete from emp;        - all rows will be delated but table will be present in db
```

DROP

```
drop table emp;
drop table emp, dept;
```

- you cannot use WHERE clause with DROP table, because DROP table is a DDL command
    - UPDATE and DELETEcommands without WHERE clause will not be allowed in MySQL workbench
    - to UPDATE and DELETE without a WHERE clause in MySQL workbench :
      → click on Edit (menu at the top)
      → Preferences
      → SQL Editor
      → "Safe Updates" (Checkbox at the bottom)
      → Uncheck it
      → Click on Ok
    - This requires a reconnection to the server
      → Click on Query (menu at the top)
      → Reconnect to server
      → Click on it

Transaction Processing
- commit will save all the DML changes since the last committed statement
- when the user issues a commit, it is known as End of Transaction
- commit will make the Transaction permanent
- Rollback will undo all the DML changes since the last committed state
- what is committed, that cannot be rolled back
- Only the DML commands are affected by rollback and commit
- Any DDL command, it automatically commits
- In Oraclke, when you exit from SQL*PLUS, it automatically commits
-Any kind of power failures, network failure, system failure, PC reboot, window close, improper exit, etc.; your last uncommitted Transaction is automatically Rolled back.

To try out Rollback, Commit and Savepoint in MySQL Workbench:-
Click on Query (Menu at the top) → Auto-commit transactions → Uncheck it

```
Commit work;
Commit;

Rollback work;
rollback;
```

work → ANSI SQL
work → optional in MySQL and Oracle

Total work done = T1 + T2 + T3 + ... + Tn;