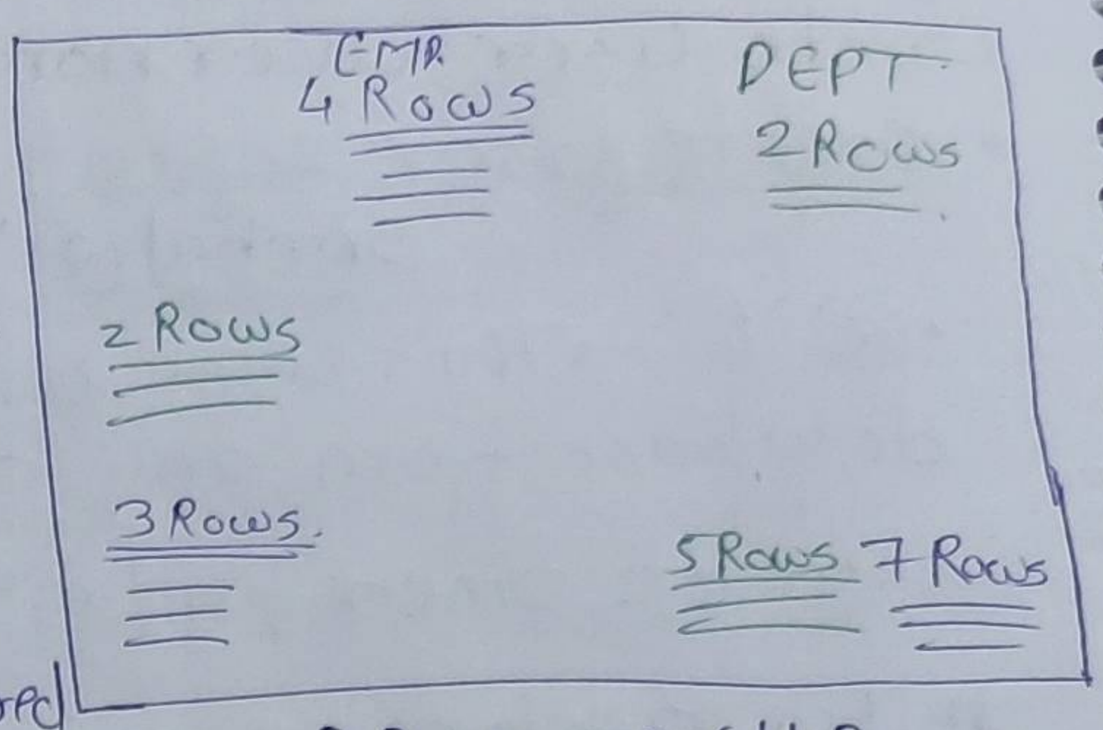
How Data store in DB server HD.

22/11/2021

- In DBMS, data is store in a file.
- within a file, the rows are stored sequentially
- In RDBMS, table is not a file; every row is a file.
- In RDBMS, rows of a table are not store sequentially, the rows of a table are scattered (fragmented) all over the DB server HD.



DB server HD.

- When you insert a new row in to a table, whereever it finds the free space in the DB server HD, it will store the row there.
- The reason why RDBMS does this is to speedup the insert statement (considering multi-user environment)
- In a multi-user environment, if multiple users are inserting rows simultenously into the same table, if the rows were to be stored sequentially, it would be very slow.
- When you select from a table, the order of rows in the output depends on the row address; it will always be in ascending order of row address.
- When you update a row, if the row length is increasing, then the row address may change (it is only in the case of varchar that the row length may increase or decrease.
- later when you select from that table, you will see the rows in some other order in the output.
- select deptno, job, ename, sal, hire date from emp;

To get in any specific order (sorted), ascending order

- select deptno, job, ename, sal, hire date from emp order by ename;

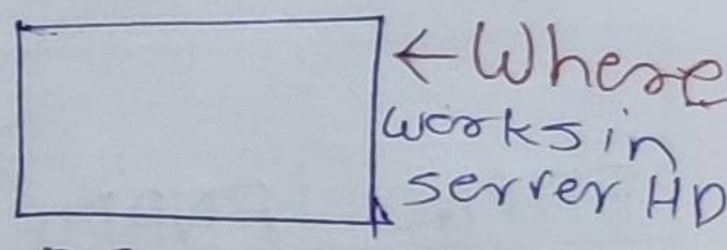
- used for sorting. sorting takes place on ASCII value.
- Sorting takes place in server RAM.
- select deptno, job, ename, sal, hiredate from emp order by ename desc;

  asc ← by default.

ORDER by clause :-

- to make the output more presentable.
- select deptno, job, ename, sal, hiredate from emp order by deptno;
- try this on different column.
- if we order by date, and some entries have same data then it depends on which row get found first.

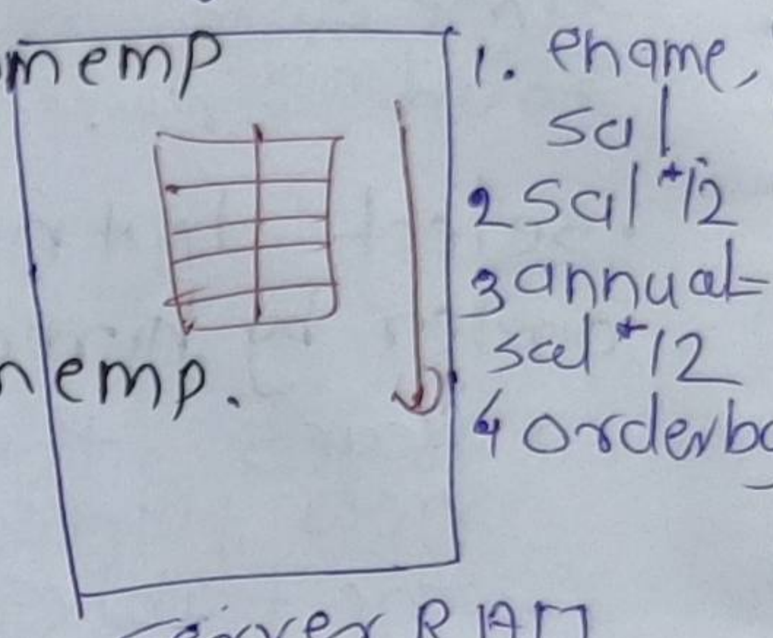
- select deptno, job, ename, sal, hiredate from emp. order by hiredate; ← Business intelligence.
  - means, the senior employee will come 1st and new joiners or last hired will come last in the table.
  - i.e. business intelligence
  - Order by plays imp role in business intelligence.



- select deptno, job, ename, sal, hire date from emp order by deptno, job; (default ascending)
- select deptno, job, ename, sal, hiredate from emp order by dept no desc, job; (in descending).
- select deptno, job, ename, sal, hiredate from emp order by dept desc, job desc;

- No upper limit on the number of columns in ORDER By clause.
- select ..... ~~as~~
  order by country, state, city;

- select deptno, job, ename, sal, hiredate from emp where deptno = 10 ← (takes place in Server HD) order by ename; ← (sorting takes place in server Ram)
- where clause is specified before the order by calue.
- The order by clause is the last clause in select statement.
- select ename, sal*12 from emp;
- select ename, sal*12 from emp (1)
  order by sal*12;
- select ename, sal*12 annual from emp (2)
  order by annual;
- select ename, sal*12 "Annual salary" from emp order by "Annual salary"
- select ename, sal*12 "Annual salary" from emp. order by 2;
- It is column ~~name~~ number in select statement.



New EMPTable.

		EMP		
EMPNO	ENAME	SAL	CITY	DeptNO.
1.	ADAMS	1000	Mumbai	10
2.	BLAKE	2000	Delhi	10
3.	ALLEN	3500	Mumbai	20
4.	KING	3000	Delhi	25
5.	FORD	4000	Kolhapur	30
	Varchar(20)			

- To get name starting from A,
- select * from emp.
  where ename > 'A' and ename < 'B';
  ADAMS > AbbbbB
  ADAMS < Bbbbb

- This is Blank-padded comparision semantics :-
- when you compare 2 strings of different lengths the shorter of the 2 strings is temporarily padded with blank spaces on RHS such that their lengths become equal; then it will start the comparision character by character, based on ASCII Value.

- select * from emp
where ename >= 'A' and ename < 'B'; (recommanded) if name is only 'A'.

- select * from emp
where ename like 'A%';
  - Like (special operator).
  - % implies any character and any number of character.

select * from emp
where ename like 'A%' or ename like 'a%'; oracle.
To make it case insensitive.

- Wildcards (use for pattern matching). %
  - like word gives meaning to %.
- select * from emp
where ename = 'A%'
- select * from emp
where ename like 'A%'. ← ENAME starts with A.
- select * from emp. ← ENAME ends with A.
where ename like '%A'
- select * from emp
where ename like '%A%'.
starting with A or ending with A or A somewhere with in the ENAME (names contains A).
- select * from emp
where ename like '__A%'; ← name which has A at 3rd position.
- select * from emp
where ename like '____'; ← all name with any 4 character.

• select * from emp ← name with 4 characters
where ename like '_I__'; I is second

• select * from emp ← name not starts with A.
where ename not like 'A%';

• select * from emp
where sal >= 2000 and sal <= 3000;
• select * from emp (faster than above).
where sal between 2000 and 3000;
Includes both the values 2000 & 3000
it is (mutully inclusive)

} same working.

• Between:
  • Between is special operator.
  • readymade method by the name between is already present in the database in the compiled format; the plan etc. is ready; it directly executes.

• select * from emp
where sal not between 2000 and 3000; ← Exclusive
works with all datatypes.

• select * from emp (faster)
where hiredate between '2020-01-01' and '2020-12-31';

• select * from emp
where hiredate >= '2020-01-01' and hiredate <= '2020-21-31';

• select * from emp (faster)
where ename between 'A' and 'F';

• select * from emp
where ename >= 'A' and ename <= 'F';
ford will not come.
FORD > AĆĆĆ
FORD > FĆĆĆ

↓
< = 'G';
then Ford will include.

(searching discrete value)

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

- select * from emp (faster), easy to write.
where deptno = any(20,30,10);

Any: • It is already present in compiler in compiled format
• It will perform logical OR.

- select * from emp (fastest)
where deptno in(10, 20, 30)

IN:
In ← logical and.

- In operator is faster than ANY operater.
but any operater is more powerflow than In operater (Any is overloaded, this slows it down.
- with IN operator, you can 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 the IN operater.
- If you want to check for >, >=, <, <= then use ANY operater.

- select * from emp
where city in('Mumbai', 'Delhi');
- select * from emp.
where city not in('Mumbai', 'Delhi');

- In operater is supported by MySQL and oracle
- Any operater is supported by oracle directly, but not supported by MySQL directly.

- select * from emp
where deptno = any(10,20); Not supported by MySQL
- Any operator works in MySQL provided it is used with sub-query
- In MySQL, you will have to use the IN operator.

- DDL → create
- DML → insert, update, delete.
- DQL → select *, select col1, col2, ..., where clause, Relational, Logical, Arithmetic operators, computed column, Alias, Distinct, how rows are scattered in the DB server HD, ORDER by clause, asc/desc, string comparision, special operater

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='Nashik'
  where empno = 1;
- update emp
  set sal = 10000
  where city = 'Mumbai';
- update emp
  set sal = 10000, city = 'Nashik'
  where city = 'Mumbai';
- you can update maltiple rows and maltiple columns simultaneously but only 1 table at a time.
- separate Update command would be needed for every table.
- update emp
  set sal = 10000
  where city = 'mumbai';
- updateemp ← It will update all the rows.
  set sal = 10000;

#DELETE:

- delete from emp
  where empno = 1;
- delete from emp
  where city = 'Mumbai';
- delete from emp; ← row will be deleted. (all).
  - table will remain.
- drop table emp; ← deletes tables also.
  - drop table emp, dept, customers;
  - you cannot specify WHERE clause with Drop table

TRANSACTION PROCESSING:

- Commit will save all the DML changes since the latst committed state.
- When the user issues a commit, it is known as End of transaction.
- commit will make the Transaction permanent.
- Commit work; or
- commit;
  - work ← optional in mysql or oracle.
  - work ← ANSI SQL:
- Total work Done = T1 + T2 + T3 + ... + Tn;
  - Transaction is a unit of work.
  - When the issue the commit, it depends on the logical scope of work. (to be decided by user).
- Rollback work;
- Roll back will undo all the DML changes since the last committed states.
- Roll back;
  work → ANSI SQL
  work → optional in mysql & oracle.
- only DML commands are affected by Rollback & commit.

- any DDL command, it automatically commits.
- when you exit from SQL*PLUS(Oracle client s/w) it will automatically commits.
- any kind of power failure, network failure, system failure, PC reboot, window close, end of task, etc. in all such cases your last uncommitted Transation, it is automatically rolled back in Mysql and oracle.
- update emp set sal = 10000;
- delete from emp;
- update and delete commands without where clauses will not be allowed in mysql workbench.
- To try above commands in MySQL WB:
- click on Edit (menu on the top) → Preferences → SQL Editor → safe updates (checkbox at the bottom) → uncheck it → click on ok.
- Click on Query (menue of the top) → Reconnect to server


```
# insert
  insert
  insert
  save point abc;
  update
  update
  save point PgR;
  Delete
  Deleter
  Rollback Work to PgR;
  Rollback work to ABC;
  Rollback Work;
default.
```

max 30 char for savepoint.

- you can rollback to savepoint
- a save point is a point within the work. (it's similar to a bookmark)
- savepoint is sub-unit of Transation.

rollback work to abc;
work → optional in Mysql and oracle.
work → ANSI SQL.
- roll back to abc;

- when you Rollback or commit, the intermediate savepoints will be cleared; if you want to use them again then you will have to reissue them in some new work.
- You cannot commit to a savepoint.
- commit will save all the DML changes since the last committed sate.
- you can only rollback sequentially (bottom to top)
- Savepoint have max 30 charactor

insert -
insert -
insert -
~~Savepoint ABC;~~ (no longer exist.
update -
update -
savepoint ABC;
delete
delete
Roll back to ABC;

- ~~you can only rollba~~
- within a transation, you can have 2 savepoint with same name. the latest savepoint overwrit the older savepoint; the older savepoint no longer exist.

SQL-ex-1, SQL-ex-2
SQL ass-1, SQL ass-4.

#To try out Rollback, commit & savepoint in MSQL workbench :—
- click on query (menu at the top) → ~~automat~~ auto commit transactions → uncheck it.

Snehal Sawant