```
============================================================
 MySQL INDEXED
============================================================
CREATE table emp(empno int(4),ename varchar(20),sal int(10),deptno int(4));
insert into emp values(5,'A',5000,1);
insert into emp values(4,'A',6000,1);
insert into emp values(1,'C',7000,1);
insert into emp values(2,'D',9000,2);
insert into emp values(3,'E',8000,2);
mysql> select * from emp;
+-------+-------+------+--------+
| empno | ename | sal  | deptno |
+-------+-------+------+--------+
|     5 | A     | 5000 |      1 |
|     4 | A     | 6000 |      1 |
|     1 | C     | 7000 |      1 |
|     2 | D     | 9000 |      2 |
|     3 | E     | 8000 |      2 |
+-------+-------+------+--------+


Select * from empno where empno=1;
1. READ (MySQL will Read Select statement)
2. COMPILE (permission,check EMPNO Column ,check index if available)
3. PLAN (how to execute select statment flowchart i.e. execution plan)
4. EXECUTE


* Execution PLAN: plan created MySQL as to how its going to execute your select statement.
* In oracle we can check the Execution plan and also modify the plan not in MySQL.



In other RDBMS:-
* create index.....?(manually)
* Use index ind_empno;
  Select * from empno where empno=1;(manually)
* Insert/delete/UPDATE
  REINDEX; (reindex cmd manually write)



In MySQL-SQL
Select * from empno where empno=1;
* whenever user write select statement,then MySQL finds first index. if index is avalaible
then
  it will goes to index table automacally and gives result. Advantage is for programmer no
  need
  to worry about index invoking in MySQL.


MySQL-SQL-INDEXES
*  Present in all RDBMS,all DBMS and Some programming languages also
*  B tree indexes in JAVA
*  to speed up the searching operations(for faster access)
*  TO speed up the select statement with a WHERE clause.
*  Data is stored randomly in dbms.mixed all table.
*  index occupies the space
*  rather searching whole table it will un the table
*  MySQl will all table data in RAM and full table scan.to make
   faster we need index.
*  there has row id MYSQl and ORACLE DBMS..we can see in oracle ...not in
*  indexs are automatically invoked by MySQL and when required
*  index are automatically updated by MySQL for all your DML Operations.
*  no upper limit on the number of indexes per table
*  if you have 2 or more INDEPENDANT columns in the WHERE caluse,THEN
   create separate indexes for each colums; MySQL will use both the INDEXES
   as when required.


Disavantage:
```

* larger the number of indexes, the slower would be the DML operations.(insert delete update)

* When Need index?--when searching need is faster
* when not need index?--When insert update delete need faster than searching.
* you cannot index text and blob
* duplicate values are not stored in an index
* Null values are not stored in an index

```
select * from emp where empno=1;(use index for column IND_ENAME)
select * from emp where empno='D';(use index for column IND_EMPNO)
select * from emp where sal>7000;(use index for columnIND_SAL)
```

Problem

```
select * from emp where empno is null; <-FULL TABLE SCAN(VERY SLOW)
```

Solution 1:

dont store instead null store zero in table

```
select * from emp where empno=0; <- works faster
```

Solution 2:

Store blank space

```
select * from emp where empno=' ';
```

---

it will Use the single index:

```
select * from emp where empno=2;(use index for column IND_EMPNO asending order)
select * from emp where Sal>5000;(use index for columnIND_SAL asending order)
```

it will both index:

```
select * from emp where empno=2 and sal>5000(it will use both index)
```

i.e. if you have 2 or more INDEPENDANT columns in the WHERE caluse,THEN
create separate indexes for each colums; MySQL will use both the INDEXES
as when required.

UPDATE EMP ACCORDING TO FOLLWING :

```
CREATE table emp(empno int(4),ename varchar(20),sal int(10),deptno int(4));
insert into emp values(1,'A',5000,1);
insert into emp values(2,'A',6000,1);
insert into emp values(3,'C',7000,1);
insert into emp values(1,'D',9000,2);
insert into emp values(2,'E',8000,2);
```

```
mysql> select * from emp;
+-------+-------+------+--------+
| empno | ename | sal  | deptno |
+-------+-------+------+--------+
|     1 | A     | 5000 |      1 |
|     2 | A     | 6000 |      1 |
|     3 | C     | 7000 |      1 |
|     1 | D     | 9000 |      2 |
|     2 | E     | 8000 |      2 |
+-------+-------+------+--------+
```

COMPOSITE INDEX:->

* combine 2 or more INTER-DEPENDENT colums in a single index
* you can combine upto 32 colums in composite index(oracle 16 limit)

INDEX Key-> column or set of colums on whose basis the index has been created
in above table ename depends on deptno(inter dependant)

```
SQL> select * from emp where deptno=1 and empno=1;
```

```
 * we need to create two sperate index DEPTNO EMPNO

ROWIN  DEPTNO EMPNO
 x0001   1     1
 x0001   1     2
 x0001   1     3
 x0001   2     1
 x0001   2     2

 ROWID-->Primary index key
 DEPTNO-->index key
 EMPNO-->Secondary index key

 Conditon when an index should be created:
* if select statement has a WHERE clause.(because where use searching)
* if select statement retrrieves<25% of the data.
* Primary key and unique columns  should always be indexed
* common columns in join operations should always be indexed


select * from emp where empno=1; <-Faster use index
select * from emp where empno=5; <-Faster use index
select * from emp where empno<2; <-Faster use index


-----------------------------------------------------------------------
Problem
select * from emp where empno>1;<-Slower ...because it will return all ROWS
i.e. if select returen more 25% table data then dont crate INDEX
-----------------------------------------------------------------------

Problem

+--------+-------+------+
| deptno | dname | loc  |Row ID
+--------+-------+------+
|      1 | TRN   | Bby  |z0001
|      2 | EXP   | Dlh  |z0002
|      3 | MKTG  | Cal  |z0002
+--------+-------+------+

select dename,ename from emp,dept
where dept.deptno=emp.deptno;<--slower before creating two index

need to create two INDEX I1 and I2

   I1
---------------
rowid deptno
x001    1
x002    1
x003    1
x004    2
x005    2

   I2
--------------
rowid deptno
z001    1
z002    2
z003    3


now we have driving,drivan index...
create index is perment cmd..DDL cmd


-------------------------------------------------------
Syatnx for creating index
-------------------------------------------------------
```

Syntax:
```
create index indexname on tablename(column);  (common for all RDBMS)

e.g.
create index i_emp_empno on emp(empno);
create index i_emp_ename on emp(ename);
create index i_emp_esal on emp(sal);

select * from emp where empno=1;
select * from emp where ename='A';
select * from emp where sal=5000;
```

---
to see which all indexex are created for particular table;

Syantax:
```
show indexes from emp;
```

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
emp	1	i_emp_empno	1	empno	A		3	NULL	NULL	YES	BTREE			YES	NULL
emp	1	i_emp_ename	1	ename	A		4	NULL	NULL	YES	BTREE			YES	NULL
emp	1	i_emp_esal	1	sal	A		5	NULL	NULL	YES	BTREE			YES	NULL

---
to see all the indexes on all the table in the databse;

Syantax:
```
use information_schema;
select * from statistics;

* statistics is a system table.
* by default all indexes are in ascending order (except postgresql  ...right left read)
  (common for all RDBMS)
```

Decending order Index:

```
create index i_emp_empno on emp(empno desc);
create index i_emp_empno on emp(onum desc);
```

Composite INDEX

```
create index i_emp_empno on emp(deptno, empno);
create index i_emp_empno on emp(deptno desc, empno);
create index i_emp_empno on emp(deptno desc, empno desc);
```

---
to drop the index:

```
drop index i_emp_empno on emp;
```

```
   create Unique index
-------------------------------------------------------
Create unique index i_emP_empno on emp(empno);
* performs extra function; it wont allow the user to INSERT
  duplicate values for empno
* we cannot create index of index
* drop and create index


Types of indexes:
1. Normal index
2. unique index
3. clustered index
4. etc


================================================================================
  SQL Privileges Grant and Revoke(DCL)
================================================================================
Grant(grant permission) and REVOKE(remove permission)

Grant:
--------------
Grant select on emp to scott;
Grant insert on emp to scott;
Grant update on emp to scott;
Grant delete on emp to scott;

Grant select,insert on emp to scott;
Grant all on emp to scott;(insert,update,delete,select)

Grant select on emp to scott,Dnyanu,Nanya,Nanu;
Grant select,insert on emp to scott,Dnyanu,Nanya,Nanu;

Grant select on emp to public;(public means all user)


Revoke:
Revoke select on emp to scott;
--------------------------------------------------------------------------------
------

To see the granted and received permission:
select * from information_schema.table_Privileges;
mysql> select * from information_schema.table_Privileges;
+-----------------------------+---------------+-------------+------------+----------------+--
------------+
| GRANTEE                     | TABLE_CATALOG | TABLE_SCHEMA | TABLE_NAME | PRIVILEGE_TYPE |
IS_GRANTABLE |
+-----------------------------+---------------+-------------+------------+----------------+--
------------+
| 'mysql.session'@'localhost' | def           | mysql        | user       | SELECT         |
NO           |
| 'mysql.sys'@'localhost'     | def           | sys          | sys_config | SELECT         |
NO           |
+-----------------------------+---------------+-------------+------------+----------------+--
------------+

* schema is a synonum for database;

--------------------------------------------------------------------------------
-----------
```