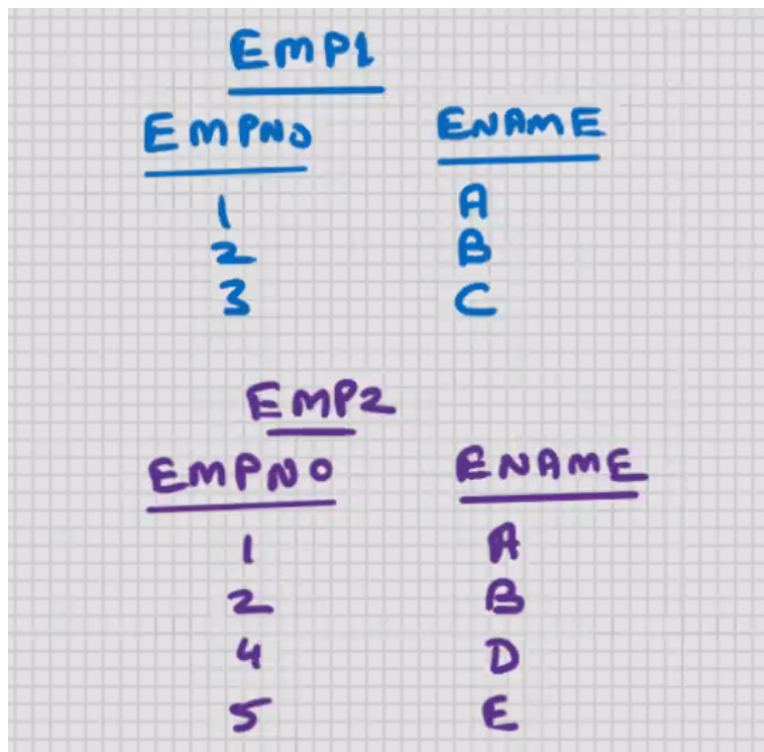


Day_7

MySQL - SQL- Set Operators

- foundation of RDBMS based on Set theory
- founder of RDBMS → Dr. E. F. Codd(1968)



<u>EMPNO</u>	<u>ENAME</u>
1	A
2	B
3	C

<u>EMPNO</u>	<u>ENAME</u>
1	A
2	B
4	D
5	E

select empno, ename from emp1

union

select empno, ename from emp2;

EMPNO	ENAME
-----	-----
1	A
2	B
3	C
4	D
5	E

- union → will combine the output of both the SELECT statements and it will suppress the duplicates
- structure of both the SELECT statements has to be the same
- number of columns in both and the corresponding datatype has to match
- the column names maybe different

select empno, ename from emp1

union all

select empno, ename from emp2;

EMPNO1	ENAME
-----	-----
1	A
1	A
2	B
2	B
3	C
4	D
5	E

- union all will combine the output of both the SELECT statements and the duplicates are not suppressed

select empno, ename from emp1

intersect

select empno, ename from emp2;

EMPNO1	ENAME
-----	-----
1	A
2	B

- intersect will return what is common in both the SELECT statements and the duplicates are suppressed

select empno, ename from emp1

~~intersect~~ except

select empno, ename from emp2;

EMPNO1	ENAME
-----	-----
3	C

- except will return what is present in first select statement and not present in the second SELECT statements and the duplicates are suppressed
- union, union all, intersect, except

```

select .....
      union
select .....
      intersect
select .....
      union
select .....
      union all
select .....
      except
select .....
      order by x;|

```

- max upto 255 SELECT statements, this limit of SQL can be exceeded with the help of Views
- the order by clause will come after all the select statement

select job from emp where deptno = 10

intersect

select job from emp where deptno=20;

```

JOB
-----
MANAGER
CLERK

```

select job from emp where deptno = 10

except

select job from emp where deptno=20;

JOB

PRESIDENT

MySQL- Pseudocolumns

- fake columns(virtual columns)
- its not a column of the table, but you can use it in SELECT statement
- e.g. a. Computed column(ANNUAL=sal*12)
b. Expressions(NET_Earnings=sal+comm-tax)
c. Function-based columns (AVG-SAL=avg(sal), R_SAL=round(sal, -3))

RDBMS supplied Pseudocolumns:

ROWID

- rowid is stands for row identifier
- Rowid is not a column of the table , but can use it is SELECT statement
- Rowid is the row address
- Rowid is the actual physical memory location in the DB Server HD where that row is located
- 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 Rowid
- it is encrypted string, it is fixed length only 18 characters
- when you insert a row, the address(id) is constant for the life of the row
- when you update a row, if the row length is decreasing, then the address(id) it will not change
- when you update a row, if the row length is increasing, and if free space is not available. then the address(id) it will change
- No two rows of any table in the entire DB Server HD can have the same Rowid
- Rowid works as an unique identifier for every row in the database.

- Rowid is used by MySQL to distinguish between two rows in the DB Server HD
- In MySQL, feature of Rowid is available; but you cannot view it
- In Oracle, feature of Rowid is available; you can view it

select rowid, ename, sal from emp;

ROWID	ENAME	SAL
-----	-----	-----
AAARmEAAAAAACfxAAA	KING	5000
AAARmEAAAAAACfxAAB	BLAKE	2850
AAARmEAAAAAACfxAAC	CLARK	2450
AAARmEAAAAAACfxAAD	JONES	2975
AAARmEAAAAAACfxAAE	MARTIN	1250
AAARmEAAAAAACfxAAF	ALLEN	1600
AAARmEAAAAAACfxAAG	TURNER	1500
AAARmEAAAAAACfxAAH	JAMES	950
AAARmEAAAAAACfxAAI	WARD	1250
AAARmEAAAAAACfxAAJ	FORD	3000
AAARmEAAAAAACfxAAK	SMITH	800
AAARmEAAAAAACfxAAL	SCOTT	3000

- you can use Rowid to update or delete the duplicate rows
- Rowid is used internally by MySQL:
 - Row Locking
 - to manage the indexes
 - to manage the cursors
 - to distinguish between two rows
 - row management

MySQL - INDEXES

	<u>EMP</u>			
<u>ROWID</u>	<u>EMPNO</u>	<u>ENAME</u>	<u>SAL</u>	<u>DEPTNO</u>
X001	5	A	5000	1
X002	4	A	6000	1
X003	1	C	7000	1
X004	2	D	9000	2
X005	3	E	8000	2

- Indexes are present in all RDBMS, all DBMS and some of the programming languages also
- used to speed up for searching operations(for faster access)
- to speed up the SELECT statement with a WHERE clause
- In MySQL, the indexes are automatically invoked by the system as and when required
- In MySQL, the indexes are automatically updates by the system for all the DML operations
- duplicate values are stored in an index
- Execution Plan→
 - plan created by MySQL as to how it is going to execute the SELECT statement.
- no upper limit on the number of indexes per table
- larger the number of indexes, the slower would be the DML operations
- you cannot index TEXT and BLOB columns
- null values are not stored in an Index
- if you have 2 or more INDEPENDENT columns in the WHERE clause, then create separate indexes for each column; MySQL will use the necessary indexes as when it required

Composite index:

- combine 2 or more INTER-DEPENDENT columns together in a single index

- In MySQL, you can combine upto 32 columns in a Composite index

INDEX KEY

- column or set of columns on whose basis the index has been created

In other RDBMS:

use index ind_empno;← awake the index

insert/update/delete.....;

REINDEX;

MySQL - INDEXES

Conditions when an index should be created:-

- * to speed up the SELECT statement with a WHERE clause
- * if SELECT retrieves < 25% of table data

```
select * from emp where empno = 1;
select * from emp where empno = 5;
select * from emp where empno < 2;
select * from emp where empno > 1;          <- NOT RECOMMENDED
```

* Primary key should always be indexed

```
select * from emp where ename = 'A';
```

```
select * from emp where empno = 1;
```


EMP				
ROWID	EMPNO	ENAME	SAL	DEPTNO
X001	1	A	5000	1
X002	2	A	6000	1
X003	3	C	7000	1
X004	4	D	9000	2
X005	5	E	8000	2

DEPT			
ROWID	DEPTNO	DNAME	LOC
Y011	1	TRN	Bby
Y012	2	EXP	Dlh
Y013	3	MKTG	Cal

- Common columns in join operations should always be indexed

Create index indexname on table(column); ← command to create index

create index i_emp_empno on emp(empno);

create index i_emp_ename on emp(ename);

create index i_emp_sal on emp(sal);

select * from emp where empno = 1;

- to see which all indexes are created for specific table:

show indexes from emp;

use information_schema;

select * from statistics;

- to drop the index:

drop index i_emp_empno on emp;

- create a index in descending:

create index i_emp_empno on emp(empno desc);

- for composite index:

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

Unique index:

```
create unique index i_emp_empno on emp(empno);
```

- perform on extra function, it wont allow to INSERT/UPDATE duplicate value in Empno
- at the time creating the unique index, if you already have duplicate values in empno, then MySQL will not allow you to create the unique index

Dropping

- if you drop the table/column, then associated indexes are dropped automatically

Types of Indexes:

- Normal Index
- Normal Composite Index
- Unique Index
- Unique Composite Index
- Clustered Index
- Non-Clustered Index
- Covering Index
- Full-Text index
- Filtered Index
- Spatial Index
- XML Index
- Hash Index
- Bitmap Index
- Index-Organized Table

- Table and Index Partitioning
- Global and Local Indexes
- Index on Abstract columns etc.