MySQL:

DR. E. F Codd is father of RDBMS (Relational algebra and relational calculus). He has framed 12 rules, which is popularly known as Codd rules.

Rule 3: It states that Systematic treatment of null values.

To extract null records use **IS** keyword. null represents **unknown value**.

* Null != 0 * Null != nothing * Null != null * **Null is null**

Rule 8: Physical data independence

Rule 9: Logical data independence

Rule 10: Integrity data independence.

- ## What is database and example?

A database is a systematic collection of data. They support electronic storage and manipulation of data. Databases make data management easy.

example: An online telephone directory uses a database to store data of people, phone numbers, and other contact details.

- ## What is RDBMS and example?

A relational database is a type of database that stores data in tables made up of rows and columns. And there exists a relation between each table via foreign key constraints.

All modern database management systems like SQL, MS SQL Server, IBM DB2, ORACLE, My-SQL, and Microsoft Access are based on RDBMS. It is called Relational Database Management System (RDBMS) because it is based on the relational model introduced by E.F. Codd.

The operators are used to manipulate the relations. And it is classified to 2 types,

Set Oriented Operators - UNION, INTERSECT, MINUS, PRODUCT

Relational Oriented Operators - PROJECTION, RESTRICTION, JOIN.

UNION: It combine rows with same dimension and the datatypes and constraints of every field must be compatible from 1 table to another table, Union operator helps us to combine two or more select statements in same order of fields and of same datatypes.

Each SELECT statement within the UNION operator must have the same number of fields in the result sets with similar data types.

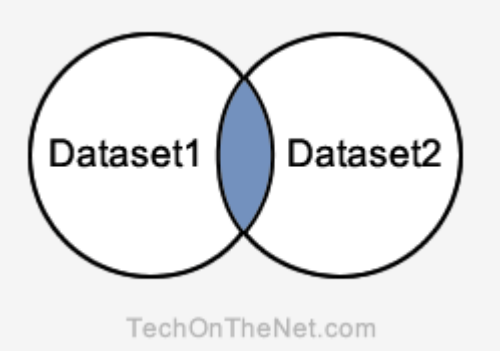
Syntax

The syntax for the UNION operator in MySQL is:

```
SELECT expression1, expression2, ... expression_n
FROM tables
[WHERE conditions]
UNION [DISTINCT]
SELECT expression1, expression2, ... expression_n
FROM tables
[WHERE conditions];
```

INTERSECT: An INTERSECT query returns the intersection of 2 or more datasets. If a record exists in both data sets, it will be included in the INTERSECT results. However, if a record exists in one data set and not in the other, it will be omitted from the INTERSECT results.

Intersect Query



The syntax for the INTERSECT operator in MySQL is:

```
SELECT expression1, expression2, ... expression_n
FROM tables
[WHERE conditions]
INTERSECT
SELECT expression1, expression2, ... expression_n
FROM tables
[WHERE conditions];
```

Relational Oriented Operators:

PROJECTION – specific columns are selected using select clause.

RESTRICTION – specific rows are selected using where clause.

JOINS - Combining two or more tables based on the common column.

MySQL is classified into 5 types:

DDL - DATA DEFINITION LANGUAGE - CREATE, ALTER, DROP, TRUNCATE

DML - DATA MANIPULATION LANGUAGE - INSERT, UPDATE, DELETE

DQL - DATA QUERY LANGUAGE - SELECT

DCL - DATA CONTROL LANGUAGE - GRANT, REVOKE

TCL - TRANSACTIONAL CONTROL LANGUAGE - COMMIT, ROLLBACK

Note:

BOTH THE COMMAND QUERIES AND DATA ARE NOT CASE SENSITIVE.

QUERIES:

1)

```
SELECT * FROM EMP WHERE SAL > 1295;
```

2)

```
SELECT * FROM EMP WHERE SAL <= 1250;
```

3)

```
SELECT * FROM EMP WHERE JOB <> "CLERK";
SELECT * FROM EMP WHERE JOB not in ( "CLERK");
SELECT * FROM EMP WHERE NOT JOB = "CLERK";
```

4)

```
select * from emp where deptno = 30 or deptno = 10;
select * from emp where deptno in (10,30);
```

5)

```
select ename, deptno, job from emp where deptno = 40 or job = "clerk";
```

6)

```
select * from emp where ename = "smith";
select * from emp where ename like "smith";
```

7) NOT OPERATOR

```
SELECT * FROM EMP WHERE SAL > 3000;
SELECT * FROM EMP WHERE NOT SAL < 3000;
```

8) BETWEEN AND

```
select * from emp where sal >= 950 and sal <= 2975;

select * from emp where sal between 950 and 2975;
```

9) BETWEEN OPERATOR IN STRING USING ASCII:

```
SELECT ENAME FROM EMP WHERE ENAME BETWEEN "JAMES" AND "TURNER";

-- J AND T WILL BE CONSIDERED AS ASCII VALUES.
```

10) LIKE OPERATIONS:

```
SELECT ENAME FROM EMP WHERE ENAME LIKE "J%";

SELECT ENAME FROM EMP WHERE ENAME LIKE "%D";

SELECT ENAME FROM EMP WHERE ENAME LIKE "%T%";

SELECT ENAME FROM EMP WHERE ENAME LIKE "__R%"; 3RD CHARACTER IS R.

SELECT ENAME FROM EMP WHERE ENAME LIKE "____";  EXACTLY 4 CHARACTERS ENAME.
```

11) LIKE OPERATIONS: (%, EXTRACTION)

```
CREATE TABLE EMP1 AS SELECT * FROM EMP; COPYING A TABLE.

UPDATE EMP1 SET JOB = "sales_rep" WHERE JOB  = "salesman";

UPDATE EMP1 SET JOB = "HR_CLERK" WHERE JOB = "CLERK";

SELECT JOB FROM EMP1 WHERE JOB LIKE "%\  %"; USING \ TO EXTRACT % OR  INSIDE A
STRING.

UPDATE EMP1 SET JOB = "HR%CLERK" WHERE JOB = "HR_CLERK";

SELECT JOB FROM EMP1 WHERE JOB LIKE "%\%%";
```

12) DATE OPERATIONS:

```
SELECT * FROM EMP WHERE YEAR(HIREDATE) = 1981;
```

INSERT FUNCTION:

1) NORMAL INSERTION:

```
INSERT INTO `company_new` VALUES(1,'WIPRO');
```

2) CHANGE ORDER:

```
INSERT INTO `company_new` (`compname`,`compid`) VALUES ("TCS",2);
```

3) MULTIPLE ROWS:

```
INSERT INTO `company_new` VALUES (3,"ORACLE"),(4,"INFOSYS"),(5,"SG"),(6,"CTS");
```

4) SET VARIABLES:

```
SET @V1 = 7, @V2 = "EMC2";

INSERT INTO `company_new` VALUES(@V1,@V2);
```

5) INSERTING NULL (OMIT THE NON PRIMARY KEY COLUMN OR EXPLICITLY USE NULL WITHOUT QUOTES):

METHOD1 : INSERT INTO `company_new` VALUES(8,NULL); (Explicitly null is used)

METHOD2 : INSERT INTO `company_new`(`compid`) VALUES(9); OMIT THE COLUMN.

6) COPYING THE ROWS FROM OTHER TABLE: (The datatypes must be compatible)

SYNTAX : INSERT INTO <TABLE> SELECT <COL1>, <COL2> FROM <OTHER TABLE>;

```
INSERT INTO `company_new` SELECT `deptno`, `dname` FROM `dept`;
```

7) DEFAULT INSERTION:

EXAMPLE:

TABLE CREATION:

```
CREATE TABLE `default_tab`(`c1` INT,

`c2` TIMESTAMP DEFAULT NOW());

ALTER TABLE `default_tab` ADD CONSTRAINT `PK_c1` PRIMARY KEY(`c1`);
```

RECORDS INSERTIONS:

```
INSERT INTO `default_tab`(`c1`) VALUES(101);

INSERT INTO `default_tab` VALUES(201, DEFAULT); EXPLICITLY DEFAULT INSERTION
```

UPDATE OPERATIONS: (MODIFY THE EXISTING ROWS):

SYNTAX :

UPDATE <TABLENAME>

SET <COL1> = VALUE1, <COL2> = VALUE2

RESTRICTION(WHERE CLAUSE);

EXAMPLE:

```
UPDATE `company_new`
SET `compname` = "EY"
WHERE `compid` = 7;
```