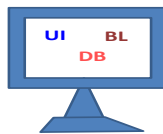


# INTRODUCTION

Software is being classified into System side and Application software. OS will be a part of system software. Our area of focussing is towards application. Industries implies application consists of UI(USER INTERFACE), BL (BUSINESS LOGIC ), DB(DATABASE).

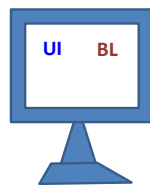
## DBMS

### WHAT IS AN APPLICATION?

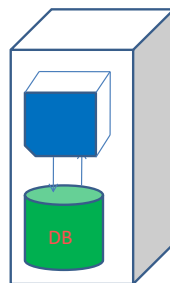


Initially all the 3 components were available in single location. Technical advancements has lead to the separation of DB from UI,BL. This has lead to a set up of CLIENT and SERVER.

## DBMS



CLIENT

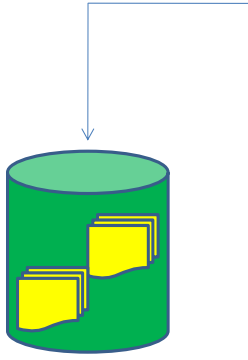


SERVER

Client Server enabled many clients can connect to the same DB server which demanded a robust Network.

## WHAT IS DATABASE?

Database allows to store data.



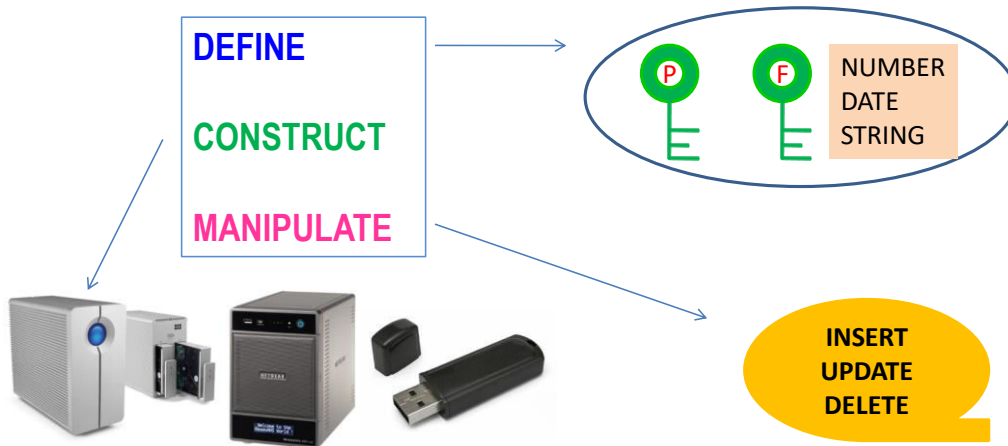
## WHAT IS DBMS?

***DBMS stands for Database Management System. DBMS facilitates not only storing of data but also provides a means to access the stored data.***

THERE IS A BROADER DEFINITION OF DBMS

- **Database Management System:**

Set of programs that



***DBMS is a set of programs that defines constraints, datatypes . Construct dealing with storage options and manipulate dealing with insert, update, delete.***

## RELATIONAL MODEL

It has a solid foundation from Relational Algebra and Relational Calculus. Data is stored in the form of tables in Relational model. It uses operators to manipulate relations which are divided into 2 categories.

## OPERATORS

SET ORIENTED	RELATIONAL ORIENTED
UNION	PROJECTION
INTERSECTION	SELECTION(RESTRICTION)
MINUS	JOIN
PRODUCT	

There are many RDBMS such as Oracle, MySQL, Microsoft SQL Server, PostgreSQL, etc. that allow us to create a database consisting of relations and to link one or more relations for efficient querying to store, retrieve and manipulate data on that database.

## MYSQL

MYSQL is an open source RDBMS.

## FOUNDERS



*DAVID AXMARK*

*ALLAN LARSON*

*MICHAEL MONTY*



"my Ess Que El" (not my sequel).

3

## INSTALLING MYSQL

## INSTALLING MYSQL 8 ON WINDOWS 10

Search Results

Web results

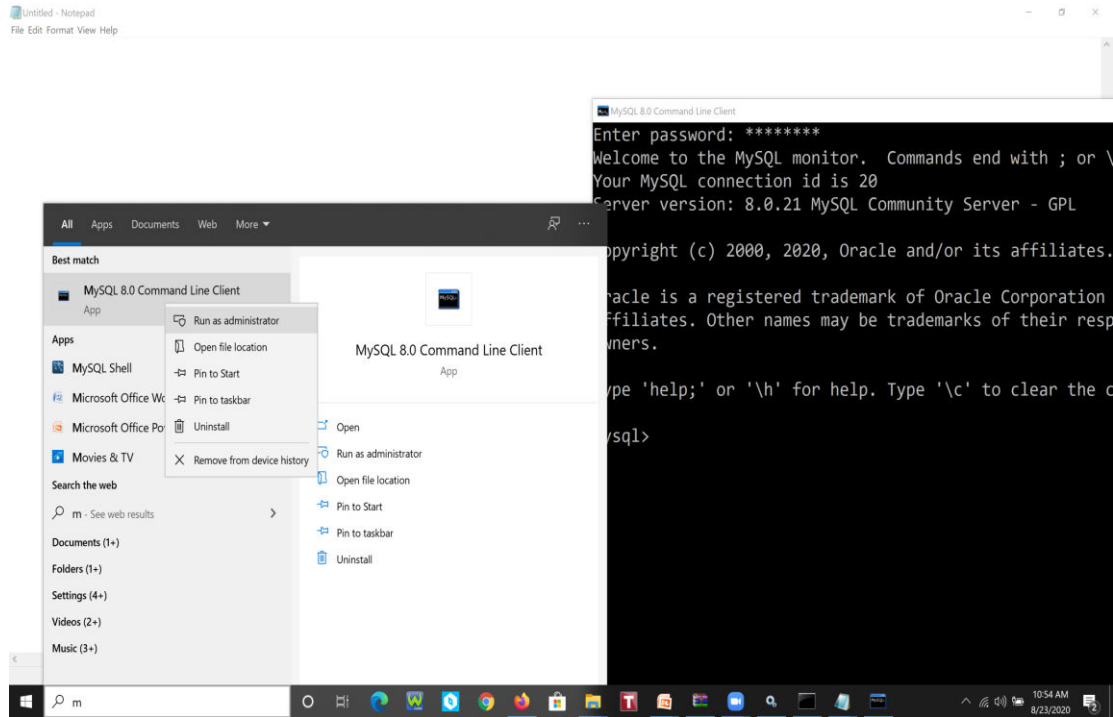
[Download & Install MySQL 8.0.11 on Windows 10 Operating ...](#)

<https://www.youtube.com › watch>

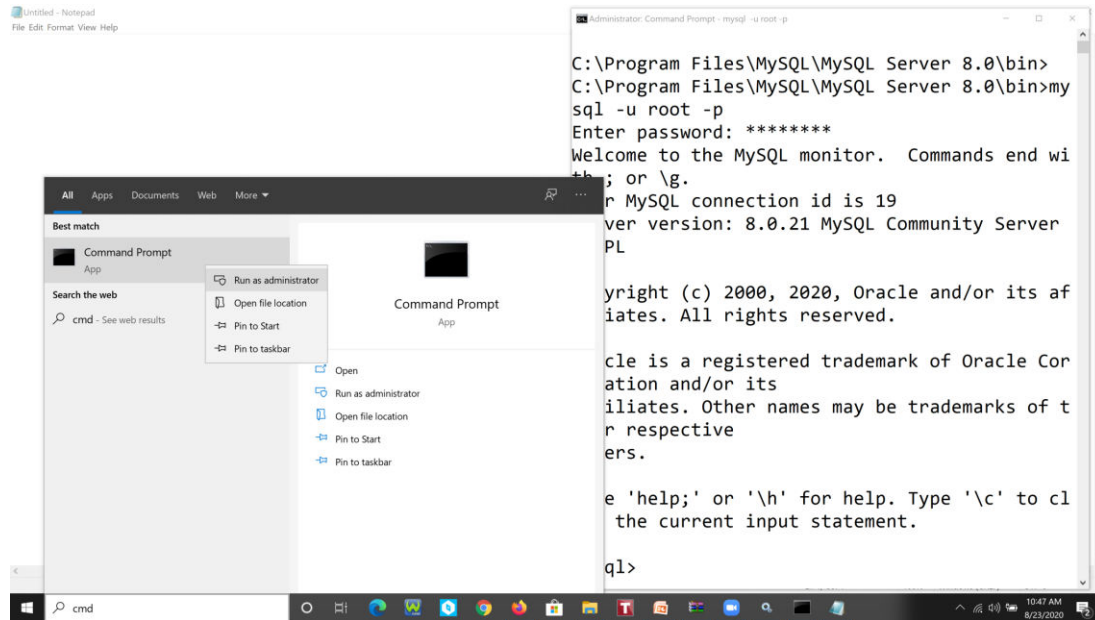
**USER NAME=> ROOT**  
**PASSWORD => Sysdba\_1**

## CONNECTING TO MYSQL

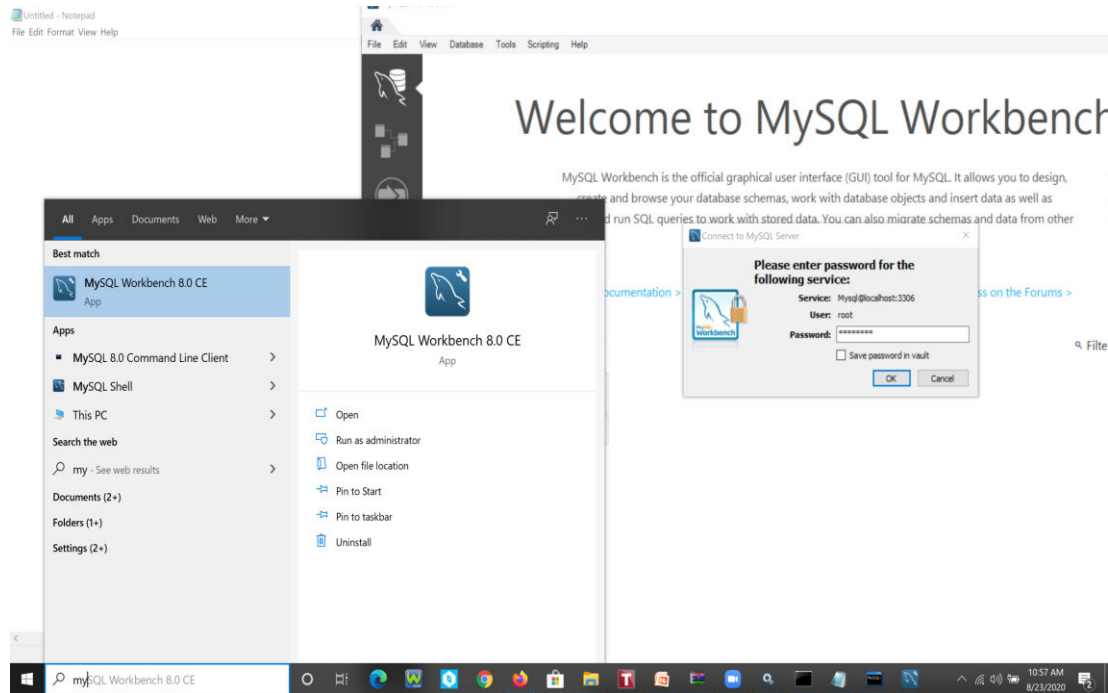
# MYSQL-CONNECT



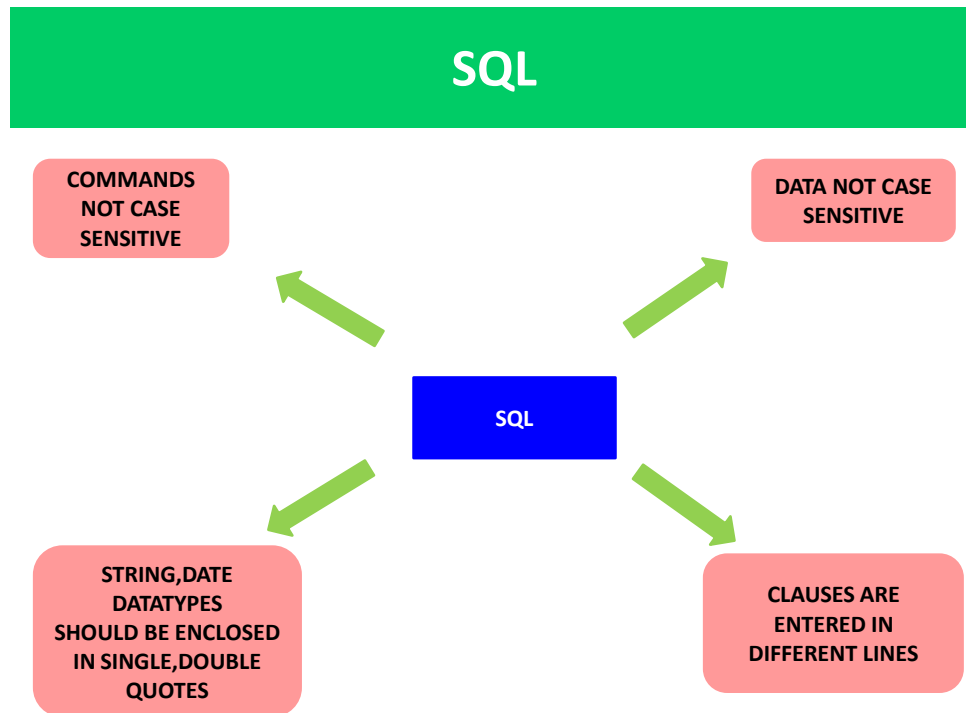
# MYSQL-CONNECT COMMAND PROMPT



# MYSQL-CONNECT WORKBENCH







## GETTING STARTED

After getting Mysql prompt, to begin with check the databases available using the command

```
mysql> show databases;
+-----+
| Database          |
+-----+
| information_schema |
| mysql              |
+-----+
```

## CREATE DATABASE

To create a database, we use the CREATE DATABASE statement as shown in the following syntax:

**CREATE DATABASE** databasename;

eg: Create Database Test;

```
mysql> show databases;
+-----+
| Database          |
+-----+
```

```

| information_schema |
| mysql              |
| test               |
+-----+
3 rows in set (0.01 sec)

```

### HOW TO USE CREATED DATABASE

*A DBMS can manage multiple databases on one computer. Therefore, we need to select the database that we want to use. Once the database is selected, we can proceed with creating tables or querying data. Write the following SQL statement for using the database:*

```

mysql> use Test;
mysql> select database();

```

```

+-----+
| database() |
+-----+
| test      |
+-----+
1 row in set (0.00 sec)

```

### HOW TO SEE TABLES AVAILABLE

Having selected the database to use, next we need to check tables available. Command used is SHOW TABLES

```

mysql> show tables;

```

Empty set (0.02 sec)

Later we learn how to create tables. Initially a script is given from which you can do that activity. After running the script run show tables again to see:

```

mysql> show tables;

```

```

+-----+
| Tables_in_test |
+-----+
| dept           |
| emp            |
| salgrade       |
+-----+
3 rows in set (0.01 sec)

```

### SAMPLE TABLES USED

dept
Deptno(pk)
Dname
Loc

emp
Empno(pk)
Ename
Job
Sal
Hiredate
Comm
Deptno(fk)

salgrade
Grade
Losal
Hisal

**In RDBMS, data is stored in the form of tables.**

### **WHAT IS A TABLE?**

Table is a two dimensional matrix consisting of rows and columns. In order to work with tables one should be knowing the structure of the table. Command that can be used is

**mysql> show columns from dept;**

Field	Type	Null	Key	Default	Extra
DEPTNO	int	NO	PRI	NULL	
DNAME	varchar(14)	YES		NULL	
LOC	varchar(13)	YES		NULL	

3 rows in set (0.01 sec)

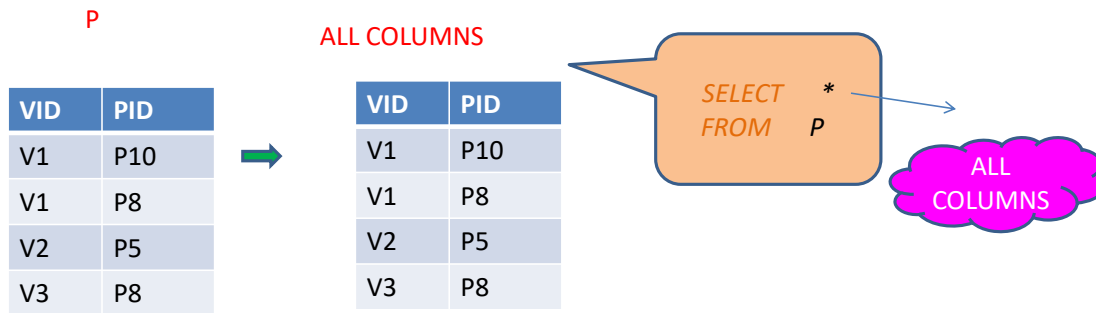
**mysql> desc dept;**

Field	Type	Null	Key	Default	Extra
DEPTNO	int	NO	PRI	NULL	
DNAME	varchar(14)	YES		NULL	
LOC	varchar(13)	YES		NULL	

3 rows in set (0.00 sec)

**From the table we can select all columns or specific columns.**

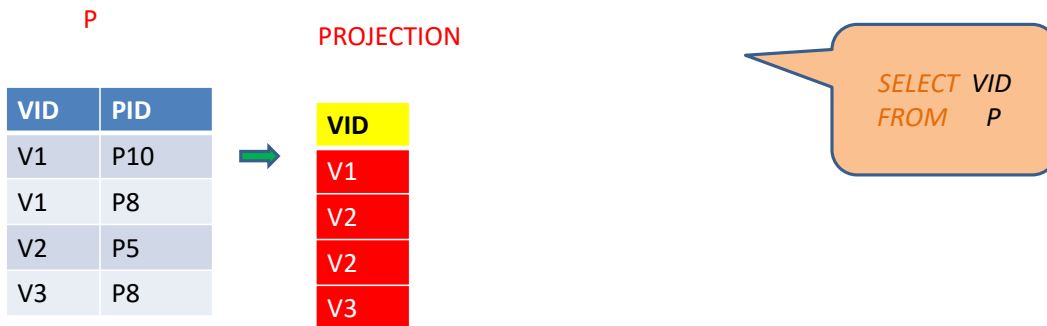
# RELATIONAL ALGEBRA



---

In order to select specific columns we need to explicitly column names.

# RELATIONAL ALGEBRA



```
mysql> select * from dept;
```

```
mysql> select * from dept;
```

	DEPTNO		DNAME
			LOC
	10		ACCOUNTING
	20		RESEARCH
	30		SALES
	40		OPERATIONS

4 rows in set (0.10 sec)

```
mysql> select dname from dept;
```

	dname	
	ACCOUNTING	
	RESEARCH	
	SALES	
	OPERATIONS	

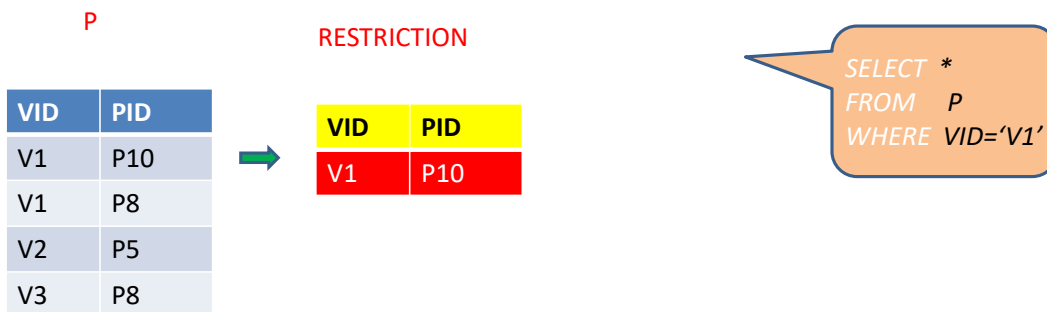
We can select all rows or particular rows

**mysql> select \* from dept where deptno=10;**

```
+-----+-----+-----+
| DEPTNO | DNAME      | LOC      |
+-----+-----+-----+
|      10 | ACCOUNTING | NEW YORK |
+-----+-----+-----+
```

1 row in set (0.00 sec)

## RELATIONAL ALGEBRA



### OPERATORS USED WITH WHERE CLAUSE

Having learnt that restriction needs where clause, let us enter into different types of operators that can be used.

# OPERATORS USED WITH WHERE

OPERATOR TYPE	OPERATOR
COMPARISON	=,<=,>=,<>
LOGICAL	AND,OR,NOT
SPECIAL	IN,NOT IN ,BETWEEN
PATTERN MATCHING	LIKE,NOT LIKE
NULL	IS NULL,IS NOT NULL

## COMPARISON OPERATORS

Earlier itself = operator example was shown, but one thing should be known is = operator takes one value only. Also how to use = operator with string, date datatypes.

It is required to find employees who joined on 3rd dec 81. Query corresponding to that is below:

```
mysql> select ename,hiredate from emp where hiredate='81-12-03';
```

```
+-----+-----+
| ename | hiredate |
+-----+-----+
| JAMES | 1981-12-03 |
| FORD  | 1981-12-03 |
+-----+-----+
```

One more query we will look into:

```
--FIND EMPLOYEES WHO ARE WORKING AS CLERK
```

```
mysql> select ename,job from emp where job='CLERK';
```

```
+-----+-----+
```

ename	job
SMITH	CLERK
ADAMS	CLERK
JAMES	CLERK
MILLER	CLERK

**Both of the above examples shows that hiredate,job value should be enclosed in single quotes or double quotes.**

Having learnt how to use = operator, it is time for us to go for other comparison operators.

Following queries gives employees details taking salary more than or equal to 2975 and less than or equal to 1250 respectively.

**mysql> select ename,sal from emp where sal>=2975;**

ename	sal
JONES	2975
SCOTT	3000
KING	5000
FORD	3000

**4 rows in set (0.00 sec)**

**mysql> select ename,sal from emp where sal<=1250;**

ename	sal
SMITH	800
WARD	1250
MARTIN	1250
ADAMS	1100
JAMES	950

**5 rows in set (0.00 sec)**

Following query selects records of all the employees except who are not working as MANAGER.

**mysql> select ename,job from emp where job<>'manager';**

ename	job
SMITH	CLERK
ALLEN	SALESMAN
WARD	SALESMAN
MARTIN	SALESMAN
SCOTT	ANALYST
KING	PRESIDENT
TURNER	SALESMAN
ADAMS	CLERK
JAMES	CLERK



```

| FORD      | ANALYST |
| MILLER    | CLERK   |
+-----+-----+
11 rows in set (0.00 sec)

```

## **LOGICAL OPERATORS**

Multiple conditions are required to satisfy certain class of queries. Take for example, employees working in deptno 20.

**mysql> select ename,deptno,job from emp where deptno=20;**

```

+-----+-----+-----+
| ename | deptno | job   |
+-----+-----+-----+
| SMITH |      20 | CLERK |
| JONES |      20 | MANAGER |
| SCOTT |      20 | ANALYST |
| ADAMS |      20 | CLERK |
| FORD  |      20 | ANALYST |
+-----+-----+-----+

```

In the above query if it is required who are CLERK, it demands additional condition.

**mysql> select ename,deptno,job from emp where deptno=20 *AND* job='CLERK';**

```

+-----+-----+-----+
| ename | deptno | job   |
+-----+-----+-----+
| SMITH |      20 | CLERK |
| ADAMS |      20 | CLERK |
+-----+-----+-----+

```

Let us know the difference between and,or operator

In the above query, if I use **OR** operator the following results will be obtained.

**mysql> select ename,deptno,job from emp where deptno=20 *OR* job='CLERK';**

```

+-----+-----+-----+
| ename | deptno | job   |
+-----+-----+-----+
| SMITH |      20 | CLERK |
| JONES |      20 | MANAGER |
| SCOTT |      20 | ANALYST |
| ADAMS |      20 | CLERK |
| JAMES |      30 | CLERK |
| FORD  |      20 | ANALYST |
| MILLER |      10 | CLERK |
+-----+-----+-----+

```

**The NOT operator displays a record if the condition(s) is NOT TRUE.**

**NOT Syntax**

```
SELECT column1, column2, ...
FROM table_name
WHERE NOT condition;
```

**mysql> select ename,job,deptno from emp where not deptno=30;**

ename	job	deptno
CLARK	MANAGER	10
KING	PRESIDENT	10
MILLER	CLERK	10
SMITH	CLERK	20
JONES	MANAGER	20
SCOTT	ANALYST	20
ADAMS	CLERK	20
FORD	ANALYST	20

8 rows in set (0.00 sec)

## **SPECIAL OPERATORS**

The IN operator compares a value with a set of values and returns true if the value belongs to that set. Following query selects details of all the employees who have salary of 1250 or 3000.

**mysql> SELECT ENAME,JOB,SAL FROM EMP WHERE SAL IN(1250,3000);**

ENAME	JOB	SAL
WARD	SALESMAN	1250
MARTIN	SALESMAN	1250
SCOTT	ANALYST	3000
FORD	ANALYST	3000

4 rows in set (0.00 sec)

The following query selects details of all the employees except who have salary of 1250 or 3000.

**mysql> SELECT ENAME,JOB,SAL FROM EMP WHERE SAL NOT IN (1250,3000);**

ENAME	JOB	SAL
SMITH	CLERK	800
ALLEN	SALESMAN	1600
JONES	MANAGER	2975
BLAKE	MANAGER	2850
CLARK	MANAGER	2450
KING	PRESIDENT	5000
TURNER	SALESMAN	1500
ADAMS	CLERK	1100

```

| JAMES   | CLERK       | 950 |
| MILLER  | CLERK       | 1300 |
+-----+-----+-----+
10 rows in set (0.00 sec)

```

The BETWEEN operator defines the range of values inclusive of boundary values. BETWEEN operator defines the range of values in which the column value must fall into, to make the condition true.

**mysql> SELECT ENAME,SAL FROM EMP WHERE SAL BETWEEN 1300 AND 2850;**

```

+-----+-----+
| ENAME  | SAL   |
+-----+-----+
| ALLEN  | 1600  |
| BLAKE  | 2850  |
| CLARK  | 2450  |
| TURNER | 1500  |
| MILLER | 1300  |
+-----+-----+
5 rows in set (0.00 sec)

```

## **PATTERN MATCHING OPERATORS**

Sometimes instead of looking for a particular text or exact match, we might be interested in a single character or few characters. Like operator comes to your rescue.

The LIKE operator makes use of the following two wild card characters:

- % (per cent)- used to represent zero, one, or multiple characters
- \_ (underscore)- used to represent exactly a single character

The following query selects details of all those employees whose name starts with 'J'.

**mysql> SELECT ENAME FROM EMP WHERE ENAME LIKE 'J%';**

```

+-----+
| ENAME |
+-----+
| JONES |
| JAMES |
+-----+

```

The following query selects details of all those employees whose name ends with 'D'.

**mysql> SELECT ENAME FROM EMP WHERE ENAME LIKE '%D';**

```

+-----+
| ENAME |
+-----+
| WARD  |
| FORD  |
+-----+
2 rows in set (0.00 sec)

```

Following query gives where 'E' is the 4th character

**mysql> SELECT ENAME FROM EMP WHERE ENAME LIKE '\_\_\_E%';**

```

+-----+
| ENAME |
+-----+

```

```
+-----+
| ALLEN |
| JONES |
| JAMES |
+-----+
3 rows in set (0.00 sec)
```

Let us go for using NOT LIKE operator.

**mysql> #find out ename which are not containing 'S'**  
**mysql> select ename from emp where ename not like '%S%';**

```
+-----+
| ename |
+-----+
| ALLEN |
| WARD  |
| MARTIN|
| BLAKE |
| CLARK |
| KING  |
| TURNER|
| FORD  |
| MILLER|
+-----+
```

## **NULL OPERATORS**

Missing or unknown value is represented as NULL. Following query selects details of all those employees who do not have MGR.

**mysql> SELECT ENAME,MGR FROM EMP WHERE MGR IS NULL;**

```
+-----+-----+
| ENAME | MGR |
+-----+-----+
| KING  | NULL|
+-----+-----+
```

Following query selects names of all employees who have been having a MGR,

**mysql> SELECT ENAME,MGR FROM EMP WHERE MGR IS not NULL;**

```
+-----+-----+
| ENAME | MGR |
+-----+-----+
| SMITH  | 7902 |
| ALLEN  | 7698 |
| WARD   | 7698 |
| JONES  | 7839 |
| MARTIN | 7698 |
| BLAKE  | 7839 |
| CLARK  | 7839 |
| SCOTT  | 7566 |
| TURNER | 7698 |
| ADAMS  | 7788 |
| JAMES  | 7698 |
| FORD   | 7566 |
| MILLER | 7782 |
```

+-----+-----+

## **CREATING TABLE FROM EXISTING TABLE**

We can take structure only, take structure plus data.

```
sql> create table hm1 as select * from dept;
```

Query OK, 4 rows affected (0.05 sec)

Records: 4 Duplicates: 0 Warnings: 0

```
mysql> select * from hm1;
```

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

4 rows in set (0.00 sec)

```
mysql> create table hm2 as select * from dept where 1=2;
```

Query OK, 0 rows affected (0.07 sec)

Records: 0 Duplicates: 0 Warnings: 0

```
mysql> select * from hm2;
```

Empty set (0.00 sec)

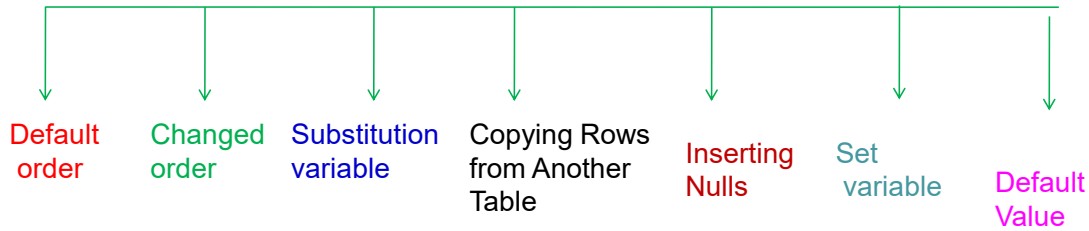
#creating our own table

```
mysql> create table hm3(sln int,name varchar(20));
```

Query OK, 0 rows affected (0.06 sec)

We need to populate the table using insert.

# INSERT Statement



**mysql> # inserting in default order**

**mysql> insert into hm3 values(1,'rahul');**

**mysql> #inserting in changed order**

**mysql> insert into hm3(name,slno) values('rishab',2);**

mysql> #inserting nulls

mysql> insert into hm3 values(3,null);

mysql> insert into hm3(slno) values(4);

mysql> select \* from hm3;

```
+-----+-----+
| slno | name  |
+-----+-----+
| 1    | rahul |
| 2    | rishab|
| 3    | NULL  |
| 4    | NULL  |
```

```
+-----+-----+
```

#using set variable

```
mysql> set @v1=5;
```

```
mysql> set @v2='giri';
```

```
mysql> insert into hm3 values(@v1,@v2);
```

Query OK, 1 row affected (0.01 sec)

```
mysql> select * from hm3;
```

```
+-----+-----+
```

```
| slno | name |
```

```
+-----+-----+
```

```
| 1 | rahul |
```

```
| 2 | rishab |
```

```
| 3 | NULL |
```

```
| 4 | NULL |
```

```
| 5 | giri |
```

```
+-----+-----+
```

```
mysql> # inserting multiple rows
```

```
mysql> insert into hm3 values(6,'hari'),(7,'nikhil'),(8,'nakul');
```

```
mysql> select * from hm3;
```

```
+-----+-----+
```

```
| slno | name |
```

```
+-----+-----+
```

```
| 1 | rahul |
```

```
| 2 | rishab |
```

```
| 3 | NULL |
```

```
| 4 | NULL |
```

```
| 5 | giri |
```

```
| 6 | hari |
```

```
| 7 | nikhil |
```

```
| 8 | nakul |
```

```
+-----+-----+
```

```
mysql> # copying rows from other table
```

```
mysql> insert into hm3 select empno,ename from emp where deptno=10;
```

```
mysql> select * from hm3;
```

```
+-----+-----+
```

```
| slno | name |
```

```
+-----+-----+
```

```
| 1 | rahul |
```

```
| 2 | rishab |
```

```
| 3 | NULL |
```

```
| 4 | NULL |
```

```
| 5 | giri |
```

```
| 6 | hari |
```

```

|      7 | nikhil |
|      8 | nakul  |
| 7782  | CLARK  |
| 7839  | KING   |
| 7934  | MILLER |
+-----+-----+

```

## UPDATE Statement

- Specific row or rows are modified if you specify the `WHERE` clause.

ENAME	SAL
SMITH	800

```

UPDATE emp
SET     sal=1.1*sal
WHERE   empno=7566;
1 row updated.

```

ENAME	SAL
SMITH	880

**mysql> # modifying rows**

**mysql> update hm3 set name='atul' where slno=3;**

**mysql> update hm3 set name='cherry' where slno=4;**

**mysql> select \* from hm3;**

```

+-----+-----+
| slno | name  |
+-----+-----+
| 1    | rahul |
| 2    | rishab |
| 3    | atul  |
| 4    | cherry |
| 5    | giri  |
| 6    | hari  |
| 7    | nikhil |
| 8    | nakul  |
| 7782 | CLARK  |

```



```
| 7839 | KING   |
| 7934 | MILLER |
+-----+-----+
11 rows in set (0.00 sec)
```

## DELETE Statement

- Specific rows are deleted if you specify the WHERE clause.

```
DELETE emp
WHERE  ename='SMITH';
1 row deleted.
```

- All rows in the table are deleted if you omit the WHERE clause.

```
DELETE emp;
14 rows deleted.
```

**mysql> # removing rows**

**mysql> delete from hm3 where slno=7782;**

**mysql> select \* from hm3;**

```
+-----+-----+
| slno | name  |
+-----+-----+
| 1    | rahul |
| 2    | rishab |
| 3    | atul  |
| 4    | cherry |
| 5    | giri  |
| 6    | hari  |
| 7    | nikhil |
| 8    | nakul |
| 7839 | KING  |
| 7934 | MILLER |
+-----+-----+
```

### CONSTRAINTS

Constraint indicates the restrictions imposed on the values of an attribute. Constraints can be specified when the table is created with the CREATE TABLE statement, or after the table is created with the ALTER TABLE statement. SQL constraints are used to specify rules for the data in a table. Constraints are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the table. If there is any violation between the constraint and the data action, the action is aborted.

### NOT NULL

Any column we don't want null values, NOT NULL can be very handy.

## The NOT NULL constraint

Ensures values to be entered compulsorily

```
create table not_tab(i1 int not null,  
                    i2 timestamp);
```

Table created.

```
Mysql> insert into  
      not_tab values(1,current_timestamp);  
1 row created.  
Mysql> select * from not_tab;
```

I1	I2
1	2019-08-28 10:32:53

Not null  
violation

```
insert into not_tab values(null,current_timestamp);  
ERROR 1048 (23000): Column 'i1' cannot be null
```

### UNIQUE CONSTRAINT

This Constraint does not allow to insert a **duplicate value** in a column. The UNIQUE constraint maintains the uniqueness of a column in a table. More than one UNIQUE column can be used in a table.

# The Unique key constraint

Ensures entered values to be unique

```
create table uni_tab(c1 int unique,  
                    c2 varchar(20));
```

Table created.

```
MariaDB [test]> insert into  
uni_tab values(100,'hp');
```

1 row created.

```
MariaDB [test]> insert into  
-> uni_tab values(100,'wipro');
```

Unique  
violation

**ERROR 1062 (23000): Duplicate entry '100' for key 'c1'**

## PRIMARY KEY

A PRIMARY KEY constraint for a table enforces the table to accept unique data for a specific column.

# The Primary key constraint

Ensures entered values to be unique

```
create table pri_tab(c1 int primary key,  
                    c2 date);
```

Table created.

```
Mysql> insert into  
      pri_tab values(500,current_date);
```

1 row created.

Unique  
violation



```
Mysql>insert into pri_tab values(500,'2010-01-05');  
ERROR 1062 (23000): Duplicate entry '500' for key 'PRIMARY'
```

# The Primary key constraint

Does not allow **NULLS**

```
Mysql>> insert into  
      pri_tab  
      values (null, '2010-01-01');  
ERROR 1048 (23000): Column 'c1' cannot be null
```

## FOREIGN KEY

A FOREIGN KEY in MySQL creates a link between two tables by one specific column of both tables. The specified column in one table must be a PRIMARY KEY and referred by the column of another table known as FOREIGN KEY. It allows duplicate and null values.

# The Foreign key constraint

Ensures entered value depends on *parent table*

```
create table for_tab(c1 int references pri_tab,  
                    c5 char(1));
```

Table created.

```
Mysql>> select * from pri_tab;
```

C1	C2
500	2016-10-16

```
Mysql>> insert into  
2 for_tab values  
3 (501,'m');  
insert into
```

Integrity  
constraint  
violation

**ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint fails (`test`.`for\_tab`, CONSTRAINT `fk\_pri` FOREIGN KEY (`c1`) REFERENCES `pri\_tab` (`c1`))**

## CHECK

A CHECK constraint controls the values in the associated column. The CHECK constraint determines whether the value is valid or not. Check constraints allow users to restrict possible attribute values for a column to admissible ones.

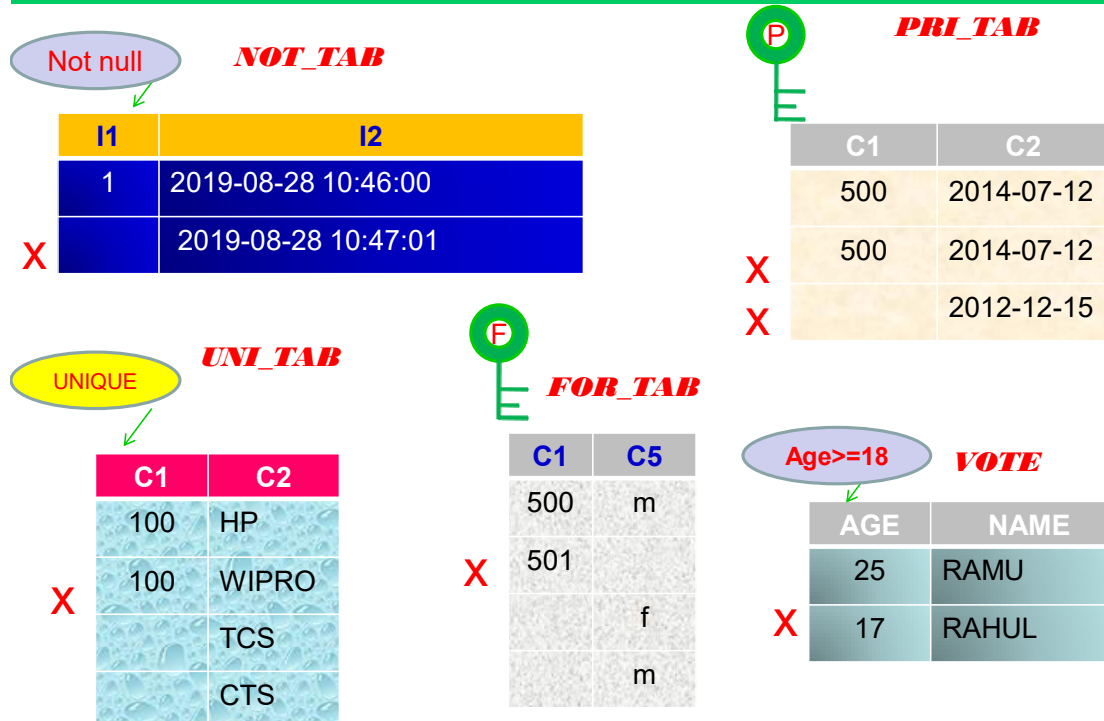
# Check Constraints

```
Mysql>> create table
vote(age int check (age>=18),
name varchar(10));
```

Table created.

```
Mysql>> insert into
vote values(17,'rahul');
ERROR 4025 (23000): CONSTRAINT
`vote.AGE` failed for `test`.`vote`
```

## CONSTRAINTS



### **DEFAULT**

While inserting data into a table, if no value is supplied to a column, then the column gets the value set as DEFAULT.

```
mysql> CREATE TABLE DEFA_TAB (C1 INT PRIMARY KEY,C2 TIMESTAMP DEFAULT NOW());
```

```
mysql> INSERT INTO DEFA_TAB(C1) VALUES(100);
```

```
mysql> SELECT * FROM DEFA_TAB;
```

C1	C2
100	2022-08-09 07:26:18

```
mysql> INSERT INTO DEFA_TAB VALUES(200,DEFAULT);
```

```
mysql> SELECT * FROM DEFA_TAB;
```

C1	C2
100	2022-08-09 07:26:18
200	2022-08-09 07:28:26

### **THREE LEVEL ARCHITECTURE**

A database management system three-level architecture . Basic idea behind this is to provide DATA ABSTRACTION,DATA INDEPENDENCE.

# EXTERNAL

## CONCEPTUAL

### INTERNAL

#### External Level :

The external level is at the highest level of database abstraction . At this level, there will be many views define for different users requirement. A view will describe only a subset of the database. Any number of user views may exist for a given global schema(coneptual schema).

One person may be interested in

other person might be in

EMPNO
ENAME

EMPNO
ENAME
SAL
HIREDATE

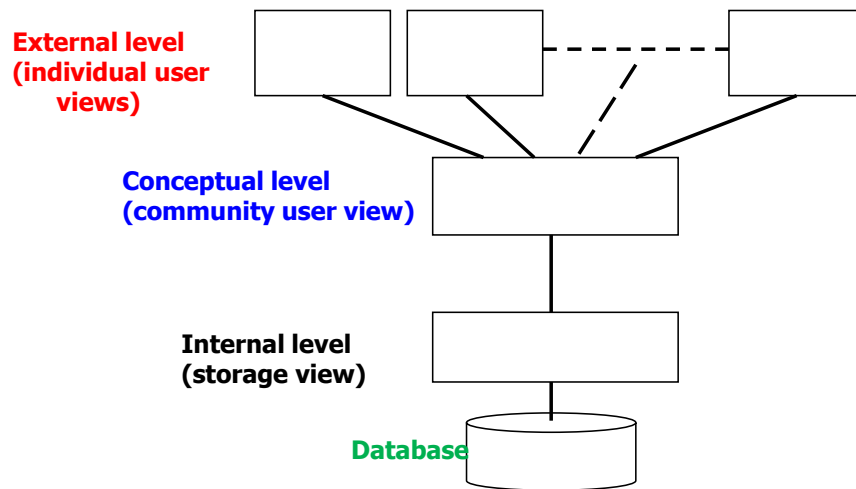
#### Conceptual Level :

At this level of database abstraction all the database entities and the relationships among them are included. One conceptual view represents the entire database. This conceptual view is defined by the conceptual schema.

The conceptual schema hides the details of physical storage structures and concentrate on describing entities, data types, relationships, user operations and constraints. It describes all the records and relationships included in the conceptual view. There is only one conceptual schema per database. It includes feature that specify the checks to relation data consistency and integrity.



# Database Architecture



31

## **Internal level :**

It is the lowest level of abstraction closest to the physical storage method used. It indicates how the data will be stored and describes the data structures and access methods to be used by the database. The internal view is expressed by internal schema.

## **DATA INDEPENDENCE**

Property by virtue of which Higher level of abstraction not affected by changes in lower level of abstraction.

## **TYPES**

### **Logical Data Independence:**

It is the ability to modify the conceptual scheme without making it necessary to rewrite application programs

### **Physical Data Independence:**

It is the ability to modify the physical scheme without making it necessary to rewrite application programs

### **Database Design:**

Database Design focuses on detailed description of data, the Relationship among them, and any Rules defined.

Suppose We want to develop a database application for college. How do we go about this?

4 steps


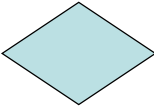
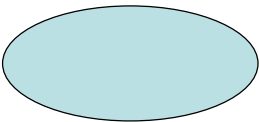
-----

1. Collecting data
2. Creating E-R diagram using collected data
3. Converting E-R diagrams to tables
4. Normalizing the tables

### **Data Model:**

The data model describes the structure of a database. It is a collection of conceptual tools for describing data, data relationships and consistency constraints.

An entity-relationship (ER) diagram is a specialized graphic that illustrates the interrelationships between entities in a database.

Terminology	Representation
Entity	
Relationship	
Attribute	
Identifier	<u>id</u>

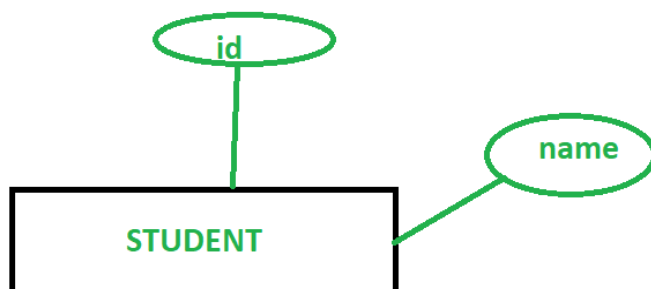
*Entities* are the principal data object about which information is to be collected.

The information about an entity is attribute

Relationship is association among entities or entity sets.

Key(Identifier) identifies uniquely a tuple.

Below example shows STUDENT is An Entity. id,name are attributes. We are collecting the data about student against 2 attributes. When E-R diagram is converted, Entity will become table name, attributes are columns.



## STUDENT

Id	Name
101	Anusha
102	Monisha

### TYPES OF ATTRIBUTES

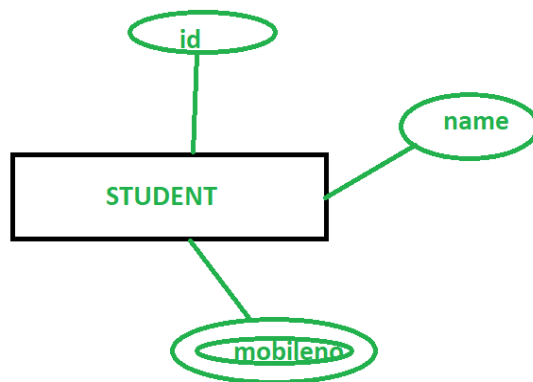
SIMPLE  
MULTIVALUED  
DERIVED  
COMPOSITE

### SIMPLE ATTRIBUTE

An attribute with a single value.

### MULTIVALUED ATTRIBUTE

An attribute with a more than 1 value. We can see mobileno is a multivalued attribute. Multivalued attribute is represented as double ellipse. When we convert multivalued attributes into tables, primary key and foreign key combination based.



## STUDENTPHONE

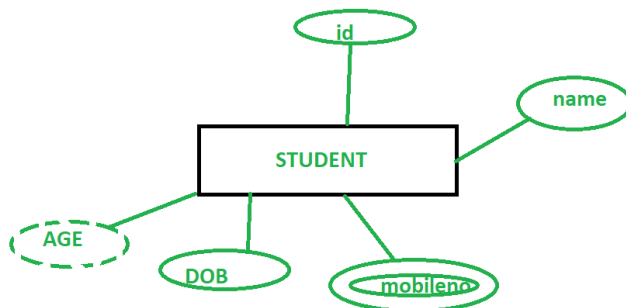
Id	MOBILENO
101	9343564642
101	7777444222
102	9343564652
102	7777444455

## STUDENT

Id	Name
101	Anusha
102	Monisha

### DERIVED ATTRIBUTE

Attribute is represented as dotted ellipse. Derived attributes are derived from existing attribute. In the example shown below DOB is the attribute from which age is derived.



Modelling derived attributes we can have a view as below

## STUDENT

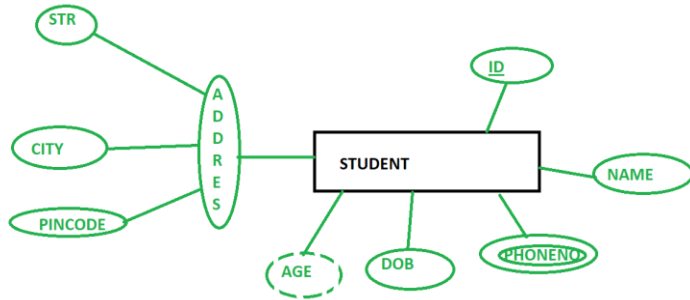
Id	Name	DOB
101	Anusha	22-JUN-99
102	Monisha	25-AUG-99

## STUDENTAGE\_VIEW

Id	AGE
101	23
102	22

### COMPOSITE ATTRIBUTE

Attribute is made up of more than 1 attribute. Address attribute consists of ,street,city,pincode attributes.



Modelling composite attribute, each attribute will become column of a table. In fact we concatenate and show it. We can create a view wrt composite attribute.

STUDENT

Id	Name	Street	City	Pincode
101	Anusha	1 <sup>st</sup> Main,NTI Layout	Bangalore	560075
102	Monisha	5 <sup>th</sup> Main,Rajajinagar	Bangalore	560010

STUDENT\_ADDRESS\_VIEW

Id	Name	Address
101	Anusha	1 <sup>st</sup> Main,NTI Layout,Bangalore 560075
102	Monisha	5 <sup>th</sup> Main,Rajajinagar,Bangalore-560010

## NORMALIZATION

It is a decompositin process of data into small tables to reduce database anomalies and reduce redundancy.

Following is a scenerio of Unnormalized form.

INVOICE												
INVNO	CUSTID	CUSTNAME	CUSLOC	ITEM1	QTY1	TOTAL1	ITEM2	QTY2	TOTAL2	ITEM3	QTY3	TOTAL3
1001	11	LMN	PUNE	TFT	5	35000	KB	15	12000	MB	8	48000
1002	22	ABC	HYD	MOUSE	25	18000	HDD	8	56000			
1003	11	LMN	PUNE	ROUTER	12	72000						

Looking at above table, Repeating columns are available in the form of item1, qty1, total1,item2,qty2.total2,item3,qty3,total3 which can be better converted as below.

*Reduce entities to first normal form (1NF) by removing repeating or multivalued attributes to another*

INVNO	CUSTID	CUSTNAME	CUSLOC	ITEM	QTY	TOTAL
1001	11	LMN	PUNE	TFT	5	35000
				KB	15	12000
				MB	8	48000
1002	22	ABC	HYD	MOUSE	25	18000
				HDD	8	56000
1003	11	LMN	PUNE	ROUTER	12	72000

### 1NF

INVNO	CUSTID	CUSTNAME	CUSLOC	ITEM	QTY	TOTAL
1001	11	LMN	PUNE	TFT	5	35000
1001	11	LMN	PUNE	KB	15	12000
1001	11	LMN	PUNE	MB	8	48000
1002	22	ABC	HYD	MOUSE	25	18000
1002	22	ABC	HYD	HDD	8	56000
1003	11	LMN	PUNE	ROUTER	12	72000

Reformed table above, there is no key item through which we can identify tuple uniquely. Database allows to add columns to establish a composite key item.

### INVOICE

INVNO	LINENO	CUSTID	CUSTNAME	CUSLOC	ITEM	QTY	TOTAL
1001	1	11	LMN	PUNE	TFT	5	35000
1001	2	11	LMN	PUNE	KB	15	12000
1001	3	11	LMN	PUNE	MB	8	48000
1002	1	22	ABC	HYD	MOUSE	25	18000
1002	2	22	ABC	HYD	HDD	8	56000
1003	1	11	LMN	PUNE	ROUTER	12	72000

It is seen that customer details depend on INVNO and not on Lineno which is called PARTIAL Dependency. In order to remove Partial dependency we can go for 2nd Normal Form.

### 2NF

INV_PART				
INVNO	LINENO	ITEM	QTY	TOTAL
1001	1	TFT	5	35000
1001	2	KB	15	12000
1001	3	MB	8	48000
1002	1	MOUSE	25	18000
1002	2	HDD	8	56000
1003	1	ROUTER	12	72000

INV_CUST			
INVNO	CUSTID	CUSTNAME	CUSLOC
1001	11	LMN	PUNE
1002	22	ABC	HYD
1003	11	LMN	PUNE

We can see here CUSTID depends on INVNO. CUSTNAME,CUSLOC DEPENDS ON CUSTID. This is called Transitive dependency. We can use 3NF to remove transitive dependency.

3NF

INV_CUST	
INVNO	CUSTID
1001	11
1002	22
1003	11

CUSTOMER		
CUSTID	CUSTNAME	CUSLOC
11	LMN	PUNE
22	ABC	HYD

### **FINAL TABLES**

INV_PART				
INVNO	LINENO	ITEM	QTY	TOTAL
1001	1	TFT	5	35000
1001	2	KB	15	12000
1001	3	MB	8	48000
1002	1	MOUSE	25	18000
1002	2	HDD	8	56000
1003	1	ROUTER	12	72000

INV_CUST	
INVNO	CUSTID
1001	11
1002	22
1003	11

CUSTOMER		
CUSTID	CUSTNAME	CUSLOC
11	LMN	PUNE
22	ABC	HYD