* Constraints

Constraints are the conditions that are assigned to a particular column to validate the data.

Types of constraints:

1. UNIQUE
2. NOT NULL
3. CHECK
4. PRIMARY KEY
5. FOREIGN KEY

Null: It is keyword which represents Empty cell
→ Any operations performed with NULL, the results will be null

1. UNIQUE:

Unique is a constraint which is assigned to a particular column which cannot accept repeated or duplicate values

2. NOT NULL:

Not Null is a constraint which is assigned to a particular column which cannot be null or which are mandatory.

3. CHECK:

Check is a constraint which is assigned to a particular column for extra validations.

→ Check constraint is assigned with a condition, if the condition is true the value gets accepted, else rejected.

Ex: Check [length(phno)=10]
Check (sal>0)

19/1/21

4. Primary key:

Primary key is a constraint which is used to assign to a column to identify a record uniquely from the table.

Characteristics of Primary key:

1. We can have only one primary key in a table
2. It cannot accept repeated or duplicate values
3. It cannot accept null
4. It is a combination of Unique and not null
5. primary key is not mandatory but recommended to have one in table.

5. Foreign key:

It is contraint which is used to establish the connection b/w two tables.

Characteristics of foreign key:

1. We can have n' no. of foreign keys in a table
2. It can accept repeated or Duplicate values
3. It can accept Null
4. It is not a combination of Unique and not null
5. It is present in child table but actually belongs to parent table
6. It is also referred as "Referential Intigrity constraint".

16/2/21

/* JOINS */

This statement is used to retrieve the data from multiple Tables simultaneously.

Types of Joins:

1. Cartesian Join or Cross Join
2. Inner Join or Equi Join
3. Outer Join
   a. Left outer Join
   b. Right outer Join
   c. full outer Join
4. Self Join
5. Natural Join

(1) CARTESIAN JOIN:

In cartesian Join, A record from Table 1 will be merged with all the records of Table 2

→ No. of columns in result table will be sumation of columns present in Table 1 & Table 2

→ No. of Records in Result table will be the product of Records present in Table 1 & Table 2.

1. Display BName, GName

```
Select BName, GName
from Boys, Girls;
```

Boys

BID	BName	GID
1.	RANBIR	11
2.	VIRAT	22
3.	RAJ	33

Girls

GID	GName
11	ALIA
22	ANUSHKA
33	SIMRAN

o/p of FROM

BID	BNAME	GID	GID	GName
1	RANBIR	11	11	ALIA
1	RANBIR	11	22	ANUSHKA
1	RANBIR	11	33	SIMRAN
2	VIRAT	22	11	ALIA
2	VIRAT	22	22	ANUSHKA
2	VIRAT	22	33	SIMRAN
3	RAJ	33	11	ALIA
3	RAJ	33	22	ANUSHKA
3	RAJ	33	33	SIMRAN

o/p of select

BName	GName
RANBIR	ALIA
RANBIR	ANUSHKA
RANBIR	SIMRAN
VIRAT	ALIA
VIRAT	ANUSHKA
VIRAT	SIMRAN
RAJ	ALIA
RAJ	ANUSHKA
RAJ	SIMRAN

(14) Self Join :

Self Join is used to join the same two tables (or) the table itself

* Why we use self join ?

The data to be selected and the condition to be executed is present in same table but in different record we use self.

Syntax:

1. ANSI:

```
select column_name
from Table_name T1 JOIN Table_name T2
ON <JOIN_CONDITION>;
```

Ex:
```
select *
from empE1 JOIN Emp E2
ON E1.MGR = E2.Empno;
```

2. ORACLE:

```
select column_name
from Table_name T1, Table_name T2
where <Join_condition>;
```

Ex:
```
select *
from emp E1, emp E2
where E1.MGR = E2.Empno;
```

1. WAQTD Ename & Managers name

```
select E1.Ename, E2.Ename
From Emp.E1, Emp.E2
where E1.MGR = E2.Empno
```

17/2/21

(2) Inner Join (or) Equi Join :

We use inner join to obtain only the matched Records or the Records which has pair

***→ We use join condition to obtain the matched Records.

Join condition :

It is a condition on which we merge two tables to get only the matched Records.

syntax for join condition :

```
Table_name1.col_name = Tablename2.col-name;
```

18/2/21

(3) Outer Join:

In outer join we get unmatched Records along with matched Records.

(i) Left Outer Join:

In left outer join we get unmatched Records of left Table along with Matched Records.

Syntax:

1. ANSI:

```
Select column_name
from Table_Name1 LEFT [OUTER] JOIN Table_name2
ON <JOIN_CONDITION>;
```

Ex:
```
select *
from emp E left outer join DeptD
ON E.Deptno = D.Deptno;
```

(ii) Right Outer Join:

An Right outer join we get unmatched Records of Right table along with matched Records.

Syntax:

1. ANSI:

```
select column_name
from Table_name1 RIGHT [OUTER] JOIN Table_name 2
ON <JOIN-CONDITION>;
```

Ex:
```
select *
from empE RIGHT OUTER JOIN DeptD
ON E.Deptno = D.Deptno;
```

2. ORACLE:

```
select column_name
from Table_name1, Table_name2
WHERE Table_name1.col_name(+) = Table_name2.col_name;
```

Ex:
```
select *
from empE, DeptD
where E.Deptno(+) = D.Deptno;
```

1. WAQTD all the details of employees & department along with unmatched Records of dept table.

```
select *
from empE, DeptD
where E.Deptno(+) = D.Deptno;
```

Ename	Dno	Dname	Dno	
A	30	D3	30	Matched Records
C	20	D2	20	
E	10	D1	10	unmatched Records from Right table
NULL	NULL	D4	40	

1. WAQTD Department name in which they are no Employees working

```
select Dname
from empE, DeptD
where E.deptno(+) = D.Deptno AND Ename is Null;
```

(iii) Full outer Join:

To obtain unmatched records of both the tables along with matched Records

Syntax:

1. ANSI:

```
select column_name
from Table_name 1 full [outer] Join Table_name 2
ON <Join_condition>;
```

Ex:

```
select *
from empE full outer Join DeptD
ON E.Deptno = D.Deptno;
```

1. WAQTD unmatched records along with matched Records from both tables.

Emp

Ename	D.NO
A	10
B	Null
C	30
D	Null
E	20

Dept

Dname	DNO
D1	10
D2	20
D3	30
D4	40

Ename	DNO	Dname	DNO
A	10	D1	10
C	30	D3	30
E	20	D2	20
B	Null	Null	Null
D	Null	Null	Null
Null	Null	D4	40

Matched Records

unmatched records of both tables

22/2/21

(5) Natural Join:

→ In Natural Join we can't be writing any join condition.

→ If the table contains similar columns we get the o/p of inner join.

→ If the table is not having similar columns we will get the o/p of cartesian Join.

Why (or) When we use Natural Join?

→ Whenever there is no table structure we use "natural Join".

→ columns are present in the table structure

Syntax for Natural Join:

ANSI:

```
select column_name
from Table_name1 NATURAL JOIN Table_name2;
```

Ex:

```
select *
from emp NATURAL JOIN DEPT;
```

(or)

```
select *
from emp Natural Join SALGRADE;
```

Cardinality Ratio: (Relationship Ratio)

We have 4 cardinal Relationships

> 1:1 [one to one]
> 1:N [one to many]
> N:1 [Many to one]
> N:N [Many to Many]

4/3/21

1. If the ratio's are:

   1:N
   N:1
   1:1, then we don't have to use a reparate table to store the Relationship.

*** The primary key of an Entity whose cardinality no. is N is choosen to be a foreign key in the Entity whose cardinality is 1.

2. If the ratio is N:N then we need a separate table to store the relationship.

* The primary keys of both the table are introduced as foreign key in the new table.

* VIEW:

Views are the virtual table which can be created and Re-used when ever we are dealing with a part of A table

Syntax:

```
CREATE VIEW view_name

AS
select stmt;
```

Normalization

It is a process of Reducing the larger table into smaller table in order to Remove Redundency and Anomaly by identifying their functional Dependency.

(or)

It is a process of Decomposing a large table into smaller table to Remove Redundency and Anomaly.

(or)

It is a process of reducing the table to its Normal form.

Normal form:

A table without Redundency and Anomaly is set to be in Normal form.

Levels of Normal form:

1. First Normal form (1NF)
2. Second Normal form (2NF)
3. Third Normal form (3NF)
4. Boyce-CODD Normal form (BCNF)

*** Note:

A table is said to be Normalized if we reduce the table to 3rd Normal form.

1. 1NF :

A table is said to be in 1st Normal form if it satisfies the following conditions

→ A table should not consist of Multi-valued data

→ A table should not have duplicate or Repeated values.

2. 2NF :

A table is said to be in 2nd Normal form if it satisfied the following conditions

→ The table should be in 1st Normal form

→ The table should not have partial Functional Dependency

Note: If the table consists of partial functional Dependency then the attributes responsible are Removed from the table

3. 3NF :

A table is said to be in 3rd Normal form if it satisfies the following conditions

→ A table should be in 2nd Normal form

→ A table should not have Transitive Functional Dependency

Note: If the table consists of Transitive Functional Dependency then the attributes responsible are removed from the table

Difference B/w Truncate, Delete & Drop

Truncate	Delete	Drop
1. It is Data Definition Language.	1. It is Data Manipulation Language	1. It is Data Definition Language
2. It is used to delete all the records from table permanently	2. Used to delete a particular record from table	2. Used to delete tables from database along with structure
3. syntax: Truncate Table Table_name;	3. syntax: DELETE From Table_name [where <filter_condition>];	3. syntax: Drop Table table_name;
4. Auto-commit statement	4. Non-Auto commit statement	4. Auto-commit statement

Difference B/w where & Having

WHERE	HAVING
1. It is not ~~used~~ Multi Row function	1. This clause is used in Multi Row functions
2. It executes Row-by-Row	2. It executes Group-By-Group
3. where clause is used to filter the Records	3. HAVING clause is used to filter the Groups
4. It executes after the FROM clause	4. It executes After the Group By clause
5. we can't pass multi-row function as an argument in WHERE clause.	5. we can pass multi-row function as an argument in HAVING clause

order of Execution :

1- from

2- WHERE

3- select

2. RENAME:

This statement is used to Rename the current Table name to new name.

Syntax:

Rename current_Table_name TO New_table_name;

Ex: Rename customer TO cust;

5. To Rename the column :

```
Alter Table table_name
Rename column current_name To new_name;
```

15/1/21

Tables:

→ Table is a collection of data or it is a logical organization of data which consists of rows and columns.

columns:

→ Column is also referred as Attributes or fields

→ A column is used to represent one property of all the entities.

Rows:

→ Rows is also referred as Records or Tupples

→ A Row is used to represent all the properties of single entity.

What is a Stored Procedure?

A stored procedure is a prepared SQL code that you can save, so the code can be reused over and over again.

So if you have an SQL query that you write over and over again, save it as a stored procedure, and then just call it to execute it.

You can also pass parameters to a stored procedure, so that the stored procedure can act based on the parameter value(s) that is passed.

Stored Procedure Syntax

```
CREATE PROCEDURE procedure_name
AS
sql_statement
GO;
```

Characteristics	OLTP (Transactional system)	OLAP (decision support system)
Application	Ordinary management, production	Analysis / Decision-suppor
Users	Information system experts	Decision-makers
Data schema	Entity / Relationship	Star / Snowflake / Constellation
Normalization	Frequent	Scare
Data	Up to date / Raw	Archived / Aggregated
Up dating	Immediate / Real time	Delay or postpone
Queries	Simple / Regular / Predefine / Predictable	Complex / Irregular / Non-Predictable / Ad-hoc
Query language	SQL, QBE, QUEL	MDX, XQuery, XMLA
Analysis axis	Uni- or bi-directional	Multidimensional or multi-axes
Operations	Modification / Up to date / Cancelling / Insertion	Lecture / Cross analysis / Refreshment
Data size	Mega or Gigabytes	Tera, Peta or Zetabytes

27/2/21

Data Definition Language

This statement is used to create, Rename, Alter or Delete an object from Database.

→ There are 5 statements:

1. CREATE
2. RENAME
3. ALTER
4. TRUNCATE
5. DROP

1. CREATE:

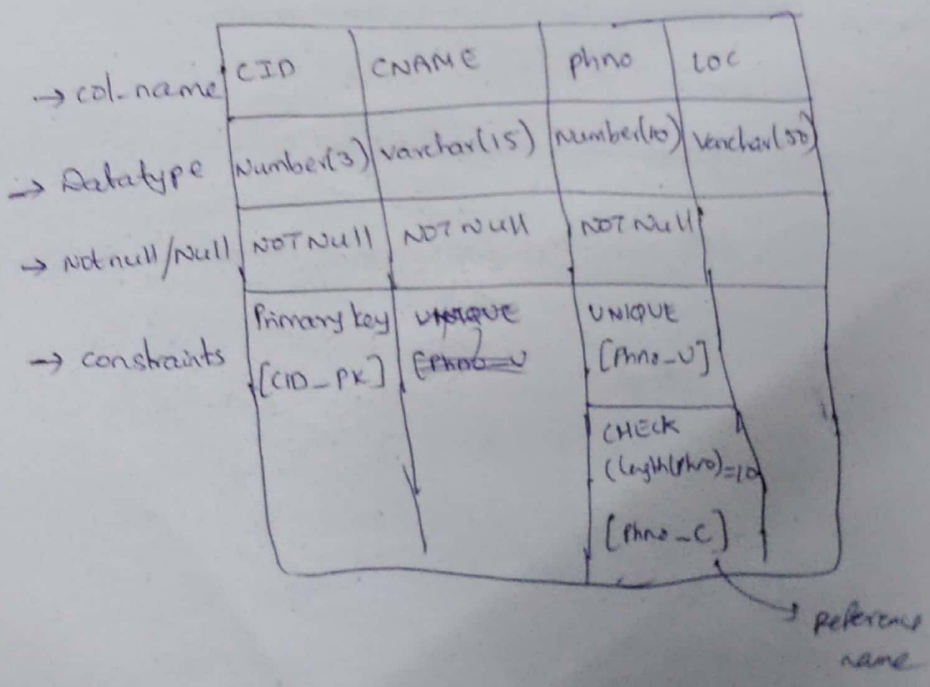
This statement is used to create an object in Database.

Ex: Table ; View

BluePrint:

Table_name : customer

No. of column: 4



Syntax:

```
Create Table table_name
(
 column_name_1 Datatype Notnull / [Null],
 column_name_2 Datatype Notnull / [Null],
  .
  .
  .
 column_name_n Datatype Notnull / [Null],
 CONSTRAINT constraint_ref_name UNIQUE (COLUMN_NAME),
 CONSTRAINT constraint_ref_name CHECK (CONDITION),
 CONSTRAINT constraint_ref_name PRIMARYKEY (COLUMN_NAME),
 CONSTRAINT constraint_ref_name FOREIGNKEY (COLUMN_NAME)
 REFERENCES parent_table_name (COLUMN_NAME)
 );
```

Ex:

```
Create Table customer
(
 CID NUMBER(3) NOT NULL,
 CNAME VARCHAR(15) NOTNULL,
 PHNO NUMBER(10) NOT NULL,
 LOC VARCHAR(50) NULL,
CONSTRAINT CID_PK PRIMARY KEY (CID),
CONSTRAINT PHNO_U UNIQUE (PHNO),
CONSTRAINT PHNO_C CHECK (length (phno) = 10)
 );
```

SQL > Table Created.

1. Create a table 'product' with columns PID, Pname, Price, Discount

Blue Print:

Table_name : Product

No. of column : 4

→ col_name	PID	Pname	Price	Discount
→ Datatype	Number(3)	Varchar(20)	Number(5)	Number(2)
→ Ndonull / Null	NOTNULL	NOTNULL	NOT NULL	
→ constraints	Primary key [PID_PK]		UNIQUE [Price_U]	

2.

```
Create Table Product
(
PID Number(3) NOTNULL,
Pname varchar(20) NOT NULL,
Price Number(5) NOT NULL,
Discount number(2) NULL,
Constraint PID_PK PRIMARY KEY (PID),
Constraint Price_U UNIQUE (Price)
);
```

2. RENAME:

This statement is used to Rename the current Table name to new name.

Syntax:

```
Rename current_Table_name TO New_table_name;
```

Ex: Rename customer To cust;

3. ALTER:

This statement is used to modify the objects in Database.

Syntax:

1. To Add a col:

```
ALTER Table Table_name
ADD column_name Datatype [null/ Not null];
```

2. To Drop a col: // delete

```
Alter Table Table_name
Drop column column_name;
```

3. To change the Datatype:

```
Alter Table table_name
Modify column_name new_datatype;
```

4. To change the Not Null Constraint:

```
Alter Table table_name
Modify column_name existing_datatype NULL/NOTNULL;
```

5. To Rename the column:

Alter Table table_name

Rename column current_name To new_name;

6. To Modify Constraints:

a) Alter Table table_name

ADD constraint constraint_ref_name UNIQUE (col_name);

b) Alter Table table_name

ADD constraint constraint_ref_name CHEQUE (condition);

c) Alter Table table_name

ADD constraint constraint_ref_name PRIMARY KEY (col_name);

d) Alter Table table_name

ADD constraint constraint_ref_name FOREIGN KEY

(column_name) REFERENCES parent_table_name

(column_name);

Ex:

1. Alter Table cust

Add mailID varchar(50) Null;

2. Alter Table cust

Drop column LOC;

3. Alter Table cust

Modify CID Number(4);

4. Alter Table cust

Modify mailID Varchar(50) NOT Null;

5. Alter Table cust

Rename column phno To Mobno;

6a. Alter Table cust
ADD constraint M_U UNIQUE (MailID);

```
select *
from user_constraints;
```
To check the constraints

4. TRUNCATE:

It is used to delete all the Records from the table permanently.

Syntax:

Truncate Table table_name;

Ex: Truncate Table cust;

5. DROP:

This statement is used to Delete the object i.e, Tables from the database along with table structure.

Syntax:

Drop Table table_name;

Ex: Drop Table cust;

* To Recover the table: (only in oracle)

Syntax:

Flashback Table table_name
To Before Drop
[Rename To New_name];

* To Drop the table from Recycle Bin

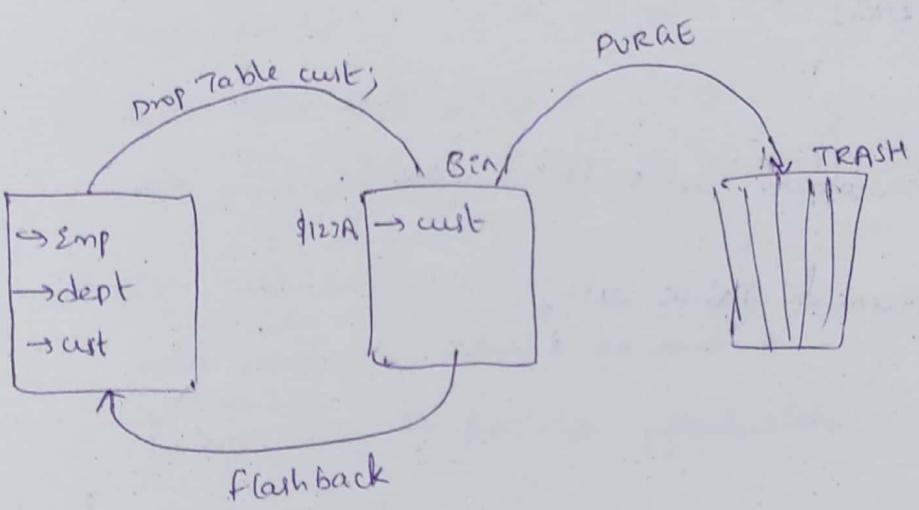
Syntax:

Purge Table table_name;

Ex: flashback Table cust

To Before Drop

Rename to customer;



Data Manipulation Language

This statement is used to insert, update or delete the Records from the table.

→ There are three statements:

1. INSERT
2. UPDATE
3. DELETE

1. INSERT:

This statement is used to insert the records into the table.

Syntax:

1. INSERT INTO table_name VALUES ($v_1, v_2, \ldots v_n$);
2. INSERT INTO table_name (COL1, COL2, ... COLn)
   VALUES ($v_1, v_2, \ldots v_n$); // If we don't know the order of columns

or

SMPL: INSERT INTO table_name (COL1, COL2 ... COLn)
VALUES (&COL1, &COL2, ... &COLn);

Ex:

INSERT INTO CUST VALUES (1, 'MR.SAGAR', 9876543210,
'SAGAR@GMAIL.COM');

// Row gets created.

COMMIT;

// commit complete i.e., the row that we created will be stored permanently in Database

Ex: INSERT INTO CUST (CName, CID, MobNo, MailID)
values ('Dinga', 2, 9876543210, 'ABC');

Ex: 3

INSERT INTO CUST (CID, Cname, Mobno, Mail ID)
values (&CID, &Cname, &Mobno, &Mail ID);

enter value for CID: 3
enter value for CName: 'BALU'
enter value for mobno: 9876543210
enter value for mailID: 'ASDF'

⇒ / (forward slash) → Enter.

we can enter another Row Details.

enter value for CID:
enter value for CName:
enter value for Mobno:
enter value for MailID:

COMMIT;

2. UPDATE:

This statement is used to update the Records in the table

Syntax:

UPDATE table_name
SET col1 = v1, col2 = v2, ---, coln = vn
[WHERE <filter-condition>];

Ex: UPDATE CUST
SET MOBNO = 9080797866
where CID = 1;

COMMIT;

3. DELETE:

This statement is used to delete a particular Record from the table

syntax:

```
DELETE
From table_name
[where <filter_condition>];
```

→ If we don't mention where condition, all the Records will get deleted.

Ex: DELETE
From cust
where CID = 4;

COMMIT;

Ex: DELETE
From cust
where CID IN (2,3,4);

Q: Diffrence B/w Truncate, Delete & Drop

Diffrence B/w DDL & DML

DDL	DML
1. It is Data definition Language	1. It is Data Manipulation Language
2. These are used to define database structure	2. It is used to Manipulate the existing database
3. COMMands are: CREATE, RENAME, ALTER, TRUNCATE, DROP	3 COMMANDS are: INSERT, UPDATE, DELETE

4. It works on whole table	4. It works on one or more rows
5. Changes done by DDL commands can't be rolled back	5. Changes can be rolled back.
6. Auto-COMMIT statements	6. Non-Auto COMMIT statements

Data control Language

This statement is used to give the permission or take back permission from another user.

→ There are 2 statements:

1. GRANT:

This statement is used to give the permission to another user.

Syntax:

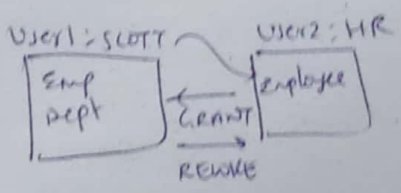
```
GRANT sql_statement ON table_name
TO user_name;
```

2. REVOKE:

This statement is used to take back the permission from another user.

Syntax:

```
REVOKE sql_statement ON table_name
FROM user_name;
```



1. Ex: CONNECT HR

Enter password : tiger

→ Connected.

```
select * from Tab;
```

```
GRANT select ON Employees
TO SCOTT;
```

→ Grant succeeded.

Transaction control language

→ There are 3 statements:

1. COMMIT
2. SAVEPOINT
3. ROLLBACK

1. COMMIT:

This statement is used to save the Transaction on Database

syntax:

```
COMMIT;
```

2. SAVEPOINT:

This statement is used to mark the position on Database

syntax:

```
SAVEPOINT savepoint_name;
```

3. ROLLBACK:

syntax:

```
ROLLBACK;
```

ROLLBACK TO SAVEPOINT

syntax:

```
ROLLBACK TO savepoint_name;
```

This statement is used to go back or undo to the previous savepoint.

Ex:

```
SQL> INSERT INTO COST values(   );
     savepoint R1;
     INSERT - - -
     savepoint R2;
     INSERT - - -
     savepoint R3;
     INSERT - - -
     RollBack TO R2;
```

deleted

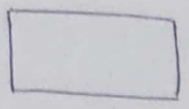
ER DIAGRAMS

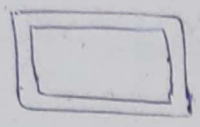
Entity - Relationship:

An Entity Relationship models describes the structure of database with the help of Diagram which is known as ER Diagram.

→ ER Diagram has 3 main components

1. Entity → Table
2. Attribute → column
3. Relationship

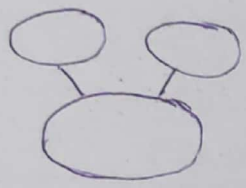
①  → strong Entity

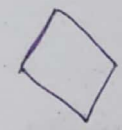
②  → weak Entity

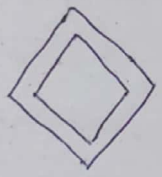
③  → Attributes

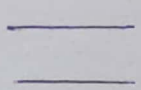
④  → prime key attribute

⑤  → Derived Attributes

⑥  → composite key attribute

⑦  → strong Relationship

⑧  → weak Relationship

⑨  Total participation

⑩  partial participation

1. Strong Entity:

An Entity which has a primary key is known as strong Entity

2. Weak Entity:

An Entity which does not have a primary key is known as weak Entity

3. Strong Relationship:

The relationship that exists B/w 2 strong entities is known as strong Relationship

4. Weak Relationship:

The relationship that exists B/w the strong and the weak Entities is known as weak Relationship

5. Total participation:

It is used to represent the total participation of an Entity in the Relationship

6. Partial participation:

It is used to represent the partial participation of an Entity in the Relationship.

Maximum Participation:

It is the maximum no. of times an Entity that can take part in a Relation

Minimum Participation:

It is the minimum no. of times an entity that can take part in a Relation.

Cardinality Number:

The maximum participation is considered as Cardinality Number

Cardinality Ratio: (Relationship Ratio)

We have 4 cardinal Relationships

> 1:1 [one to one]
> 1:N [one to many]
> N:1 [Many to one]
> N:N [Many to Many]

4/3/21

1. If the ratio's are:

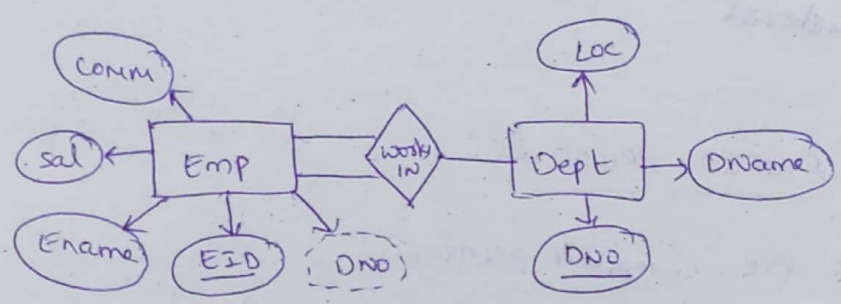
   1:N
   N:1
   1:1, then we don't have to use a separate table to store the Relationship.

*** The primary key of an Entity whose cardinality no. is N is choosen to be a foreign key in the Entity whose cardinality is 1.

2. If the ratio is N:N then we need a separate table to store the relationship.

*** The primary keys of both the table are introduced as foreign key in the new table.

ER Diagram



* VIEW:

Views are the virtual table which can be created and Re-used when ever we are dealing with a part of A table

Syntax:

```
CREATE VIEW view_name
AS
select stmt;
```

Q//

* > Diff. b/w Table & View

* > Advantages of Views

INDEX:

When we have index, it Enhance (increases) the speed of searching in the Database.

i.e; There are 1000 tables in Database, so for easy of access, we use index.

Index are of 2 types:

1. clustered
2. Non-clustered

PROCEDURE / STORED PROCEDURE:

These are Pre-compiled programs.
(Already compiled and which will be stored in Executable form).

CURSOR:

Cursor Means Memory (Internal storage)

→ while executing these things should be stored somewhere i.e; cursor.