CONTRAINTS

→ Constraint is a mechanisms which is used to stop or restrict invalid data entered by end user to implement business rules.

→ We can apply the constraints in two situations :-

(i) During the creation of table
(ii) After creation of table

constraints has two name :-
1) System defined constraint name
2) user defined constraint name.

1) System defined constraints Name :-

→ Constraint name defined by the system along with the software is known as system defined constraint name.

2) User defined constraints name :-

→ constraints name defined by user manually is known as user define constraint name.

→ We can apply the constraint at two cases

(i) column level
(ii) Table level

(i) column level :-

→ Applying the constraint after defining the column immediately than those constraints can be called as column level constraints.

(ii) Table level :-

→ Applying the constraints after defining all columns in the table (or) at the end of table then constraints are called as table level constraints.

constraints are of following types :-

1. UNIQUE
2. NOT NULL
3. CHECK
4. primary key
5. Foreign key / Referential Integrity
6. Default.

1. UNIQUE :-

→ Allows unique value in the column
→ We can apply on more than one column
→ It allows us to enter Null values.

Create table title
(
Title Id Number(4) Unique
Tname Varchar2(20) unique
Price Number(5)
Discount Number(2)
);

② NOT-NULL :-
→ when specified for a column it prevents NULL values into the column.
→ we can apply not null on more than one column

```
create table title {
    :
    Price Number(5) not null
    :
    };
```

③ PRIMARY KEY :-
→ Same as unique but allows only one column per table.
→ It does not allow to enter NULL values.

Eg:-
```
create table title {
    Title Id Number(4) primary key,
    Tname varchar2(20) not null unique,
    Price Number(5) Not Null,
    Discount Number(2),
    };
```

④ CHECK :-

→ Restrict the value of column being inserted.

→ it allows the user to check particular condition on column before it is accepted in table.

```
create table Title
        {
        Title id Number(4) primary key,
        name Varchar2(20) not Null,
        price Number(5) check (price>0),
        Discount Number(2),
        };
```

⑤ REFAULT :- Specifies a default value for the column if no values is assigned to the column at the time of inserting it.

```
create Table Title {
        ...
        ...
        Discount number(2) default 10,
        ...
        ...
        };
```

⑥ Column level Referential Integrity constraints :-

→ (Foreign Key)

→ It is a column in a table matches with column of primary key of another table. These columns are called foreign keys.

→ Referential Integrity constraint means data inserted into the foreign key column must have matching data in another table's column.

→ Foreign key constraint must be specified on child but not on parent.

→ parent record can be deleted provided no child record exists.

→ parent table / master table cannot be updated if child record exists.

→ Records cannot be inserted into the detail / child table if corresponding records in master table do not exist.

→ Records of master table cannot be deleted if corresponding child records exist in detail table / child table.

Eg :-

Child table

```
Creat table Author_Title
  {
  A# number(4),
  Title Id varchar2(5) Reference Titles
                       (Title Id),
  copies sold number(5)
  );
```

table name of parent

→ Parent table

```
create Table Title
    {
    Title Id Number(5) primary key,
    TName varchar2(20),
    Price Number(5),
    Discount Number(2),
    };
```

on-delete cascade :-

→ Generally, we cannot delete the parent record if any child records exists but by using on delete cascade we can forcefully delete parent records from parent table even though child records exists.

→ once we delete parent records ~~child~~ corresponding child records also gets deleted.

→ Supposing you don't want child row to exist whenever parent row is deleted, then ON DELETE CASCADE option can be used along with foreign key.

Eg:- create Table Author_title {

```
    A# Number(4),
    Title Id varchar2(5) Refrence Title (TitleId)
                                On delete cascade,
    Copies sold Number(5)
    };
```

Syntax to see the list of all constraints →

Select * from user_constraints;

OR

Desc user_constraints

Syntax to see list of constraints on a specific table :-

Select owner, constraint_name, constraint_type, status, search_condition from user_constraints where table_name = 'child';

Syntax to drop, disable or enable constraint →

alter table <Table Name> disable / enable / drop constraint <constraint name>

Eg :- alter table Author_title drop constraint unique;

Syntax to change the constraint Name →

Alter table <Table Name> rename constraint <old constraint Name> to <New constraint name>;

JOINS :-

→ It is the mechanism, which is used to combine or add one or more than one table at the time.

→ Sometime we may not get complete information from single table, if we want to get complete information about particular point we need to retrieve data from more than one table by performing join operations.

Types :-

1. Equi Join / Inner Join / Simple Join
2. Natural Join
3. Non Equi Join
4. Outer Join (left outer, Right outer, Full outer)
5. Self Join
6. Cross Join / Cross product / Cartesian product

→ We can perform join operations in two ways :-

i) Horizontal join
ii) Vertical join.

→ We can perform join operations in two styles :-

1) Theta / Non ANSI Style
2) ANSI Style.

EQUI – JOIN :- In this join whenever the rows are matching or values are matching only those records will be displayed as output.

→ In this joint we need to use "=" operator in where clause

→ Atleast we require one common column between the tables

→ Here, whenever rows are not matching or values are not matching those records are not displayed as o/p.

EMP1

ED	Ename	SAL	Dept NO
101	Raj	9K	10
102	Ram	10K	20
103	Ravi	11K	30
104	Raghu	12K	20
105	Sai	13K	10

DEPT1

Dept NO	Dname	Loc
10	Sales	Hyd
20	marketing	mumbai
30	HR	Delhi
40	finance	Kolkata

Syntax :-

Select * from <Table1> <Table2> ... <TableN> where <condition> ;

Eg:-

select * from emp1, dept no1 where emp1. Dept NO = Dept1. Dept NO.

ANSI Style of equi join :-

Syntax :-

Select * from emp 1 inner join dept 1 on emp 1 . dept no = dept 1 . dept no

using USING clause :-

select * from emp 1 inner join dept 1 using (dept no)

Joining two tables :- (both table is not having same column name)

Select * from emp 1 inner join dept 1 on dept no = dno;

→ But if we are using USING clause then both table have same column name. (atleast one column).

NATURAL JOIN :-

→ using clause and natural join both are exactly same as far as output is concerned.

→ If you want to perform natural join then also we need same column name.

Eg :-

Select * from emp1 natural join dept1;

NON - EQUI JOIN :-

→ In this join user needs to use any relational operator except equal to operator in the where clause of the select statement.

EID	Ename	Sal	DeptNo
102	Ram	8655	20
103	Vikay	8642	30
104	ajay	8636	20
105	vijay	8633	10
106	tom Hanks	12492	10
107	vam dam	2492	20
101	Raj	9655	10

emp1

Grade	SS	ES
1	7000	5000
2	5001	10,000
3	10,001	15000

grade

Select emp1.*, gno from emp1, grade where Sal > SS and Sal <= ES;

output

EID	EName	Sal	DeptNo	GNo
107	Vamdamre	2492	20	1
102	Ram	8653	20	2
103	Vikas	8642	30	2
104	ajay	8636	20	2
105	vijay	8633	10	2
101	Raj	9653	10	2
106	Tomhanks	12492	10	3

→ The advantage of Non-equi join is even though there is no common column between the tables we can perform join operations

OUTER-JOINS :-

→ In equi-join there might be a chance of loosing some information, to recover that lost information we need to use outer joins.

→ outer joins are broadly divided into three types :-

1) left outer join
2) Right outer join
3) Full outer join

left outer join :-

→ This join is a combination of equi-join operation plus lossed information from the left hand side table.

Eg :-

```
Select * from emp1 left outer join
dept1 on dept1.deptno =
                    emp1.deptno;
```

→ here left outer join is the outer point & hence emp1 is the left table and dept1 is the right table.

→ This is ANSI Style of left outer join.

Now,

Non-ANSI Style of left outer join :-

```
Select * from emp1, dept1
where emp1.deptno = dept1.
                    deptno(+);
```

→ In the above example (+) indicates deficiency ie loss of information.

→ If you placed the (+) symbol on the ~~left~~ right hand side

of equal to operator then it will display all the values from left hand side table.

Right outer join :-

→ This join is the combination of equi join operation plus lossed information from right hand side of the table.

Eg :-

```
Select * from emp 1 right outer join
dept 1 on emp 1. deptno = dept1.
                          deptno;
```

NON ANSI style of right outer join :-

```
Select * from emp 1, dept 1 where
emp 1. dept no (+) = dept 1. dept no;
```

→ In the above example if we placed the (+) symbol on left hand side of the equal to operator, then it will display all the values from right hand side of the table.

Full outer join :-

→ This join is the combination of equi join operation plus lost information from the left hand side table and also lost information from the right hand side table.

Select * from emp 1 full outer join dept 1 on emp 1 . deptno = dept 1 . deptno ;

Non-ANSI style of full outer join →

Select * from emp 1 , dept 1 where emp 1 . deptno = dept 1 . deptno(+)
union
Select * from emp 1, dept 1 where emp 1 . deptno(+) = dept 1 . deptno