CONSTRAINTS

- ✓ They are the data integrity rules/restrictions which allows only valid data into tables.
- ✓ Through constraints we can fulfil the business requirements
- ✓ We provide constraints on tables and even on views.

Constraint keys:

1. **Primary key**
2. **Unique**
3. **Not null**
4. **Check**
5. **Ref(log)**
6. **Default**
7. **Foreign key**

1. Primary key:

- ✓ It acts as both (unique +not null) which means it won't allow duplicate and null values
- ✓ Implicitly an unique index et defined on primary key columns
- ✓ These should be only one primary key for an entire table
- ✓ A P.K can hold maximum of 32 columns(i.e unique index limitation)
- ✓ Materialized view get defined only on tables, which are aving primary key's
- ✓ Generally we call primary key table as master table/parent table.

2. Unique key:

- ✓ It allows only unique (null) and values (won't allow duplicates)
- ✓ Not null values are allowed through unique constraint
- ✓ For unique key also an implicit unique index get defined
- ✓ An unique key can hold maximum of 32 columns

3) Not Null: It won't allow null values, but allows duplicate values.

Note:

- ✓ Not null constraint are allowed only in columns levels
- ✓ Views won't allow not null constraints

4) Check:

- ✓ To restrict/enforce other than standard integrity rules (Primary key (unique + not null) we use check constraints.

- ✓ Check constraint throws an error only when condition becomes false, won't throw for 'true and null'

- ✓ It won't allow sub queries as a condition
- ✓ It won't allow user defined functions

Note: Check constraint allows regular expressions. Views won't allow check constraints.

5) Default:

- ✓ It takes default values (if user won't provide any value to a columns then default provides default values to columns).
- ✓ It won't allow sub queries and user defined functions.

6) Foreign key:

- ✓ It's a reference integrity constraint (RIC), if ever you provide any value into the foreign key columns before begin inserted that value will be referred through F.K with primary/unique column
- ✓ F.K allows null values for flexibility and allows duplicates
- ✓ PK and FK columns names could be different but data types should be same/compatible, size should also be same.
  - System defined column level (CL)
  - System defined table level (TL)
  - User defined column level
  - User defined table level.

- **System defined column level:**

```
SQL>create table con(sno number(5) primary key, name varchar2(10)
unique,    bal number(5) check (bal>5000),id number(5) not null,
bank varchar2(10) default 'sbi');

SQL>create table con1 (sno number (5) references con(sno), loc
varchar2(10));
```

- **System defined table level:**

```
SQL>create table con2(sno number(5), name varchar2(10), bal
number(5), primary key(sno), unique (name), check(bal between
1000 and 5000));
```

```
SQL>create table con3(sno number(5), loc varchar2(10), foreign
key(sno) references con2(sno));
```

Schema: The logical collection of dat structures called 'schema'.

Schema objects and name spaces:-

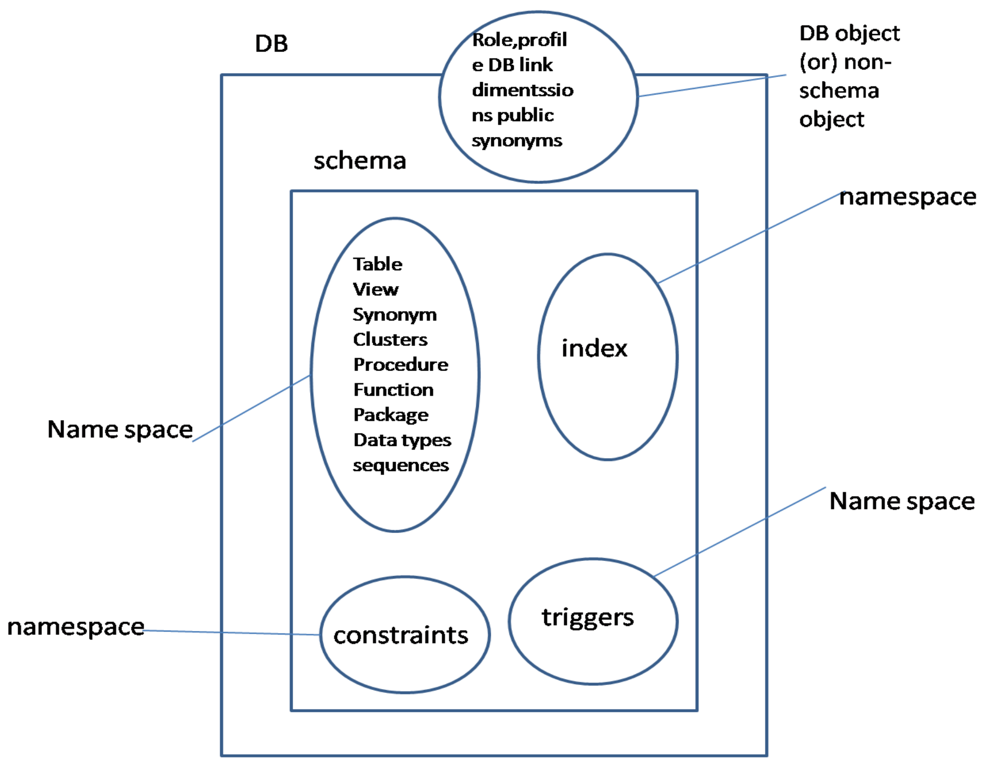

DB
Role,profil e DB link dimentssio ns public synonyms
DB object (or) non-schema object
schema
namespace
Table View Synonym Clusters Procedure Function Package Data types sequences
index
Name space
Name space
namespace
constraints
triggers

Table also DB object but additionally it acts as schema object
For eg in schema objects it table has emp name

- **User defined column level:**

  SQL>create table con5(sno number(5) constraint key1 primary key, name varchar2(10) constraint un1 unique, bal number(10) constraint chk1 check(bal>1000), id number(5) constraint nn1 not null);

  SQL>ceate table con6 (sno number (5) constraint keyfl references con5 (sno), loc varchar2(10));

- **User defined table level:**

  SQL>create table con7(sno number(5), name varcar2(10), id number(5),   bal number(5),constraint keyp2 primary key(sno),constraint un2 unique(name),constraint chk2 check(bal in(1000,2000,3000)));

(or)

```
SQL>create table con9 (constraint key3 primary key (sno), constraint
un3 unique (name),sno number(5),name varchar2(10));

SQL>create table con8 (sno number (5),loc varchar2(10), constraint
keyf2 foreign key(sno) references con7(sno));
```

- **Mixed method:**

```
SQL>create table con10 (sno number (5) primary key, loc varchar2
(10),name varchar2(10), constraint un5(loc));
```

Advantage of table level than column level constraints:

Column comparison is possible in table level which is not in column level

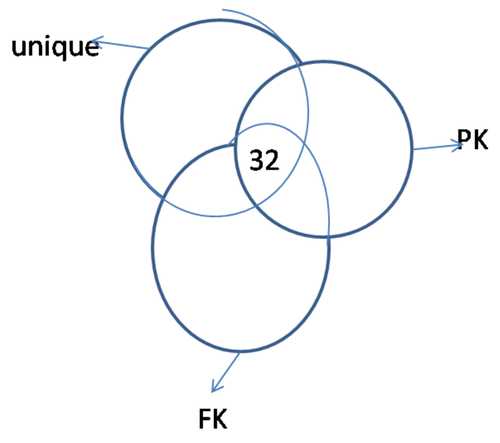
```
SQL>create table ctab(x number(5),y number(5), check (x>y));
SQL>create table ctab(x number(5),y number(5) check(x>y));
//invalid
```

- **Self-key:** If you refer a FK with in the same table PK that FK is called as 'self-key'.

```
SQL>create table selftab(sno number(5) primary key, id number(5)
references selftab(sno));
```

- **Composite key:**

It holds more than one column (PK +FK +unique). Whichever the number of columns PK is having FK also s to contain same number of columns.



Eg:-

```
SQL>create table bt(sno number(5), loc varchar2(10),name
varchar2(10), constraint comkey primary key(sno,loc));
SQL>create table ct(sno number(5),addr varcar2(10), constraint
comfkey foreignkey(sno, addr) references bt(sno,loc));
SQL>insert into bt(sno,loc) values(10,'x');
SQL>Insert into ct values (10,'x');
SQL>Insert into ct values (10,'y');      //invlid
SQL>Insert into ct values (10, null);
SQL>Insert into ct values (20, null);
SQL>create table utab(sno number(5),name
varcar2(10),unique(sno,name));
SQL>Insert into utab values (10,'x');
SQL>Insert into utab values (10,'y');
SQL>Insert into utab values (10,'x');          //invalid
SQL>Insert into utab values (10, null);
SQL>Insert into utab values (10,null);         //invald
```

C1	C2
10	X
10	X
10	Y
10	Null
10	Null

In the above table last two rows are unique or same values then null also treat as same at that time it treats as duplicate record, so won't allow.
In the above example null values become equal when all of the non-null values are same.

Defining constraints on existing table by using alter command:

By using alter command we can also provide constraints in two levels.

1. Columns level. (inline constraints)
2. Table level (dat_of_line constraints).

Generic sysntax: (not exact syntax)

```
SQL>alter table <table_name>
add|modify|disable|enable|validate|nonvalidate|rename|drop|enforce constraint
<constraint_name>;
```

Note: It is not possible to add Not Null constraint rather we modify it from Null to Not Null and Not Null to Null by using one alter we can use 'n' number of 'adds'.

Defining our own index name on a PK column:

```
SQL>create table ctab(sno number(5) primary key using index
(create index dex on ctab(sno));
```

if index is already exists

```
SQL>create table ctab1 sno number(5) primary key using index <index_nme>;
```

Add:

```
SQL>create table con15(sno number(5),nme varchar2(10),bal number(5));
    SQL>alter table con15 add constraint PK15 primary key(sno);
    SQL>alter table con15 add unique(name)
        add constaint c15 check(bal>1000)
        add check(loc in('hyd'));

    SQL>alter table con15 add foreign key(bal) references con15(sno);
```

Modify:

```
    SQL>create table con16(sno number(5) null);       //it is not a constraint
    SQL>alter table con16 modify sno constraint nn15 not null;
    SQL>alter table con16 modify sno null;
    SQL>alter tble con15 modify (sno not null, name unique);
```

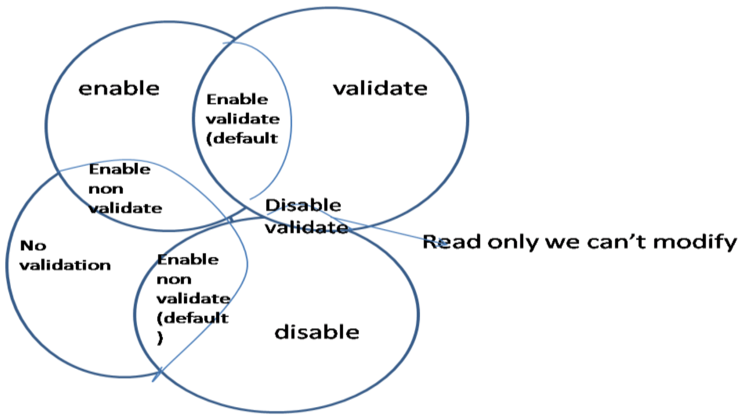
Rename:

Syn: Alter table <table_name> rename constraint oldname to newname;

SQL>create table ct(sno number(5) constraint kp10 primary key, name varchar2(10) constraint un10 unique);
SQL>alter table ct rename constraint kp10 to kp11;
Note: we cannot alter more than one column at a time.

Constraint states:

Enable/disable: works for futex data.
Validate/novalidate: past data.



- But by default enable means enable-validate and disable means disable no validate.
- While disabling and enabling constraints indexes get dropped and re-created but by using 'keep index' we can prevent the index by dropping.

Defining a disabled constraint:

SQL>create table lb(sno number(5) constraint kp12 primary key disable);
SQL>alter table lb enable constraint kp12;
SQL>alter table lb disable constraint kp12;

```
SQL>alter table lb enable primary key;
SQL>alter table lb disable primary key;
```

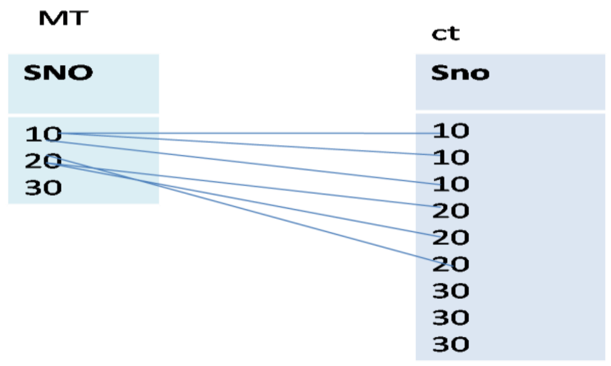
Note: Without knowing the name of a PK unique constraint we can enable and disable and drop the constraints of a table.

Eg for 'keep index':

```
SQL>alter table lb disable primary key keep index;
SQL>alter table lb enable constraint uk1 unique;
SQL>alter table lb disable constraint(sno,loc);
SQL>alter table lb drop constraint kp12;
SQL>alter table lb drop unique(sno,name);
```

Delete rules:

Event	Action
• Deleting the parent records • Dropping parent column and table • Dropping/disabling PK when it is relation with FK	• On deleting (while defining FK) • Cascade constraint • Cascade



- ✓ If you want to delete the PK record you can not delete because table is in relation with FK and you have child records so, you can not delete, first you need to delete child records, these is a chance to delete the records by using 'on delete cascade'.
- ✓ If tables are in rlation (PK with FK) you can't delete PK column and PK record and PK table and PK until unless deleting the CT, but we have the chance to drop and delete by using one table.

On deleting cascade: Generally it is not possible to delete the parent records/master records directly when they are having child records but by providing 'on delete cascade' at the time of FK definition it is possible.

On delete set null: It provides null values for department child records while deleting parent record.

Cascade constraint: Vary directly dropping master/parent table and PK column is not possible when they are in relation with FK , but by using cascade constraint it is possible.

Cascade: If you to drop/disable PK/unique key constraint when they are in relation with FK you have use cascade.

```
SQL>create table mtab(sno number(5)) constraint kp3 primary key, loc varchar2(10));
SQL>create table mtab1(sno number(5) constraint kp4 primary key, name varchar2(10));
SQL>create table ctab(sno number(5) conatrnt fk3 references matb(sno) on delete cascade);
SQL>create table ctab1(sno number(5) constraint fk4 references mtab(sno) on delete set null);
SQL>insert into mtab(sno) values(10);
SQL>insert into mtab(sno) values(11);
SQL>insert into ctab(sno) values(10);
SQL>insert into ctab(sno) values(10);
SQL>insert into ctab(sno) values(10);
SQL>delete from mtab where sno=10;
SQL>select * from ctab;
SQL>delete from mtab1 where sno=10;
SQL>select * from ctab1;
SQL>alter table matb drop column sno;       //error
SQL>alter table mtb column sno cascade constraint;
SQL>drop table mtab;    //valid
```

Note: Ere already we dropped PK column so that table is dropped in the below statement mtab1 is in relation so, table is not dropped it gives error.

```
SQL>drop table mtab1;  //invalid
SQL>drop table mtab1 cascade constraint;  //valid
SQL>alter table mtab1 drop constraint pk4;
//error: this unique/PK is referenced by some FK's
SQL>alter table mtab1 drop constraint PK4 cascade;       //valid
```

	CASCADE	CASCADE CONSTRAINT	ON DELETE CASCADE
To drop the master table which is having child table	N	Y	N
To drop primary key which is having foreign key	Y	N	N
To disable the primary key which is foreign key	Y	N	N
To delete the master record which is having foreign key	N	N (For foreign key)	Y (for foreign key)

<u>Constraint checking:</u>

Constraint checking is takes place in two ways:

1. Initially immediate(default)
2. Initially deferrable

- If constraint checking takes place at the individual statements i.e called 'initially immediate' which is default.
- But if constraint checking takes place at the time of transaction is called 'transaction specific' we do this with 'initially deferrable'.

Eg for initially immediate:

```
SQL>create table mtab4(sno number(4) primary key initially immediate);
SQL>insert into mtab4 values(10);
SQL>insert into mtab4 values(10);//invalid

Error: unique constraint (apps-sys-c00209..) violated.
SQL>set constraint all deferrable.
```

Eg for initially deferrable:

```
SQL>create table mtab5(sno number(5) constraint keyp primary key
deferrable);
SQL>set constraint keyp deferrable;
```

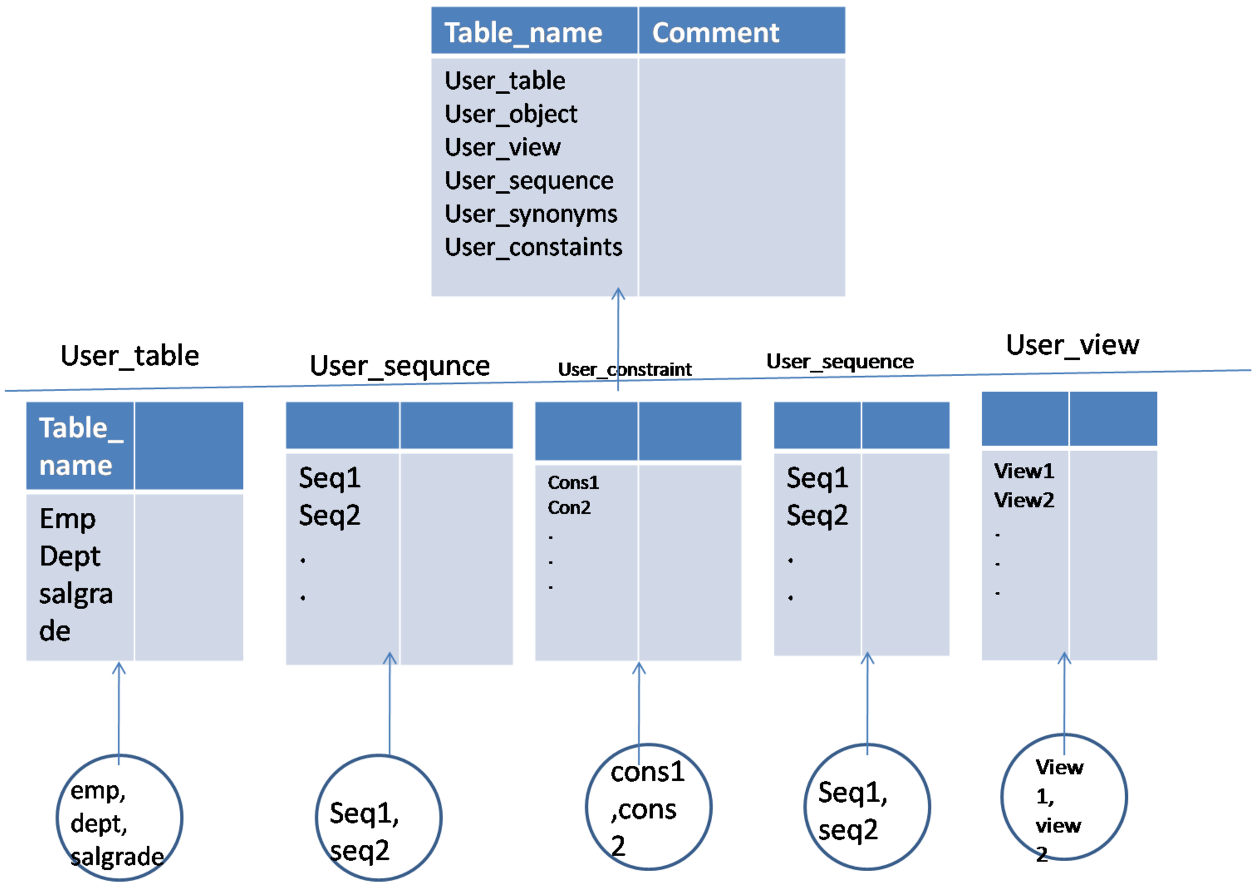
Note:here in the (create table) above eg we used deferrable, after that.

Disadvantages of constraints:

- ✓ Constraints can’t handle varying data, but by using trigger we can handle varying data.
- ✓ Constraint can check the old data but trigger’s can’t check the existing data.(it checks only incoming data)
- ✓ Constraints are more useful than trigger.
- ✓ Constraints can give guarantee for centralized data.

Tables regarding with constraints:

Data dictionary tables:



SQL>select table_name, comment from dict where table_name like '%constraint%','%view';

- ✓ In dict everything will be in uppercase
- ✓ Metadata: data to the data.
- ✓ Data dictionary tables will store in 'dict'
- ✓ To find out the column and total no of column in a given table we use following queries

SQL>select column-id,column-name,table-name from user-tab-columns where table-name='emp';

SQL>select count(column_id) from user_tab_columns where table_name='emp';
->To find out the constraint_name once we know the tablename;

SQL>select constraint_name, constraint_type, table_name, status,index_name from user_constraints where table_name='emp';

we use following query to find out constraint name and table name once we know the column name.

SQL>select column_name,table_name,constraint_name from user_cons_columns where column_name='sno';

Constraint types:

Primary key	→	P
Unique	→	U
Foreign key	→	R
Check	→	C
Not null	→	C

SQL>select object_name, created, timetamp, status from user_object where object_name='conid';

SQL>select object_type, count(*) from user_objects roup by object_type;

- ✓ In DICT we will get all table types information
- ✓ To know column identification, constraints, status, and everything we know by using data dictionary tables it's vary use full.