CONSTRAINTS :-

Rules which are enforced on data being entered, and prevents the user from entering invalid data into tables are called CONSTRAINTS.

ORACLE allows programmers to define constraints at:

↳ Column level

↳ Table level

COLUMN LEVEL :→

If data Constraints are defined along with the column definition when creating or altering a table structure, they are column level constraints.

Note1:→ Column level constraints are applied to the current column.

Note 2:→ A column level constraints cannot be applied if the data constraint spans across multiple columns in a table.

TABLE LEVEL :→

If data Constraints are defined after defining all the table columns when creating or altering a table structure, it is a table level Constraint.

Note1:→ Table level Const^n must be applied if the data Const^n spans across multiple columns in a table.

① PRIMARY KEY:-

↳ Column with primary key constraints imposed on it cannot be left blank i.e the NOT NULL attribute is active.

↳ Data held across column must be unique.

↳ Primarykey can be on a single column or on a combination of Several columns (at most 16 columns in oracle).

eg:→ Column level

a) eg) Create table emp(

```
        empno   number(5) PRIMARY KEY,
        ename   varchar(20),
          :
          :
      );
```

eg:→ Table level

b) eg) Create table emp(

```
      empno   number(5),
      ename   varchar(20),
        :
        :
      comm    number(5,2) NOT NULL,
      PRIMARY KEY (empno, ename));
```

(2) UNIQUE KEY:→

- ↳ Data held across column should be unique
- ↳ Column can also take NULL values.

eg:→ Column level

a)

Sql)
```
create table client1
  (
      client_no varchar2(6) UNIQUE,
      name varchar2(20),
      add1 varchar2(20),
      add2 varchar2(20),
      bal_due number(10,2));
```

eg:→ Tablelevel

b):

Sql)
```
create table client2
   ( clientno varchar2(6),
     name varchar2(20),
     add1 varchar2(20),
     add2 varchar2(20),
      bal-due number (10,2),
   UNIQUE (client-no));
```

③ foREIGN KEY:-

foreign keys represents relationships bet^n tables. A foreign key is a column(s) whose values are derived from the primary key or unique key of some other table.

↳ Table in which the foreign key is defined is called a foreign Table or Detail Table. The table that defines the primary key or unique key and is referenced by the foreign key is called Primary Tabl or Master Table.

↳ The master table can be referenced in the foreign key definition by using the adverb REFERENCES. Even if the column is not specified, by default, oracle references the PRIMARY KEY in the master table.

REIGN KEY:-

3) column level :→

a)(i)

```
1) Create table emp
   empno number(5) Primary key,
(depono number(5) referenices dept(depno)
      =
      =
  );
```

a)(ii)

```
sql> Create table emp
        empno number(5) Primary key,
   ( depno number (5) references dept,
        =
        =
    );
```

Note1:→ If applied at column level then "FOREIGN KEY" phrase can be omitted

Note2:→ If applied at column level then we can omit column ~~name~~ of master table as shown in eg 3 (a) (ii).

eg:-
b). Table level:→

```
sql). create table emp
  (empno number(5) PRIMARY KEY,
   depno  number(5),
   ename  varchar2(20),
     =
     =
   comm   number(5,2) NOT NULL,
   FOREIGN KEY (depno) REFERENCES dept(
                                depno));
```

NULL VALUE :→

A null value is different from a blank or a zero.

- * Setting NULL value is appropriate when the actual value is unknown, or when a value would not be meaningful.
- * NULL multiplied by 10 is NULL
- * " added/sub by 10 is NULL
- * NULL value can be inserted into columns of any data type.
- * NOT NULL constraint can only be applied at column level.
- * NOT NULL can be applied as a CHECK constraints, however Oracle reccommends that this be not done.
- * Applying NOT NULL on a columns makes it necessary to provide valid value for that specific column.

NVL func:→

SQL):-

```
Select ename, sal * NVL(comm,0) G-sal
from emp;
```

Note:→ NVL (comm, 0) will convert NULL value(s) (if any) in comm column to zero and then add it to sal column & calculate Gross salary G-sal.

mportant facts regarding FOREIGN KEY(s):-

1) We cannot delete records from the master table if corresponding records exists in the detail table.

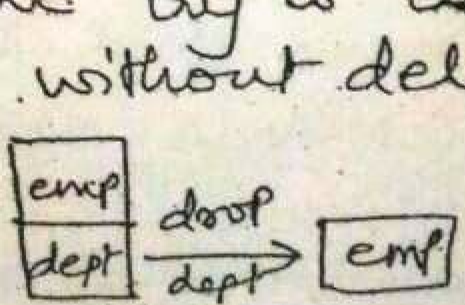
But this default behavior of the foreign key can be changed by using the ON DELETE CASCADE option. Now if user deletes a record in the master table, all corresponding records in the detail table along with the record in the master table will be deleted.

In Table emp:

eg:- deptno number(5) references dept(deptno) ON DELETE CASCADE

2) We ~~cannot~~ insert record in detail table if a corresponding value does not currently exist in the master table.

Note 1:→ If we try to ~~delete~~ drop dept table without deleting emp table



issue

SQL> drop ~~dept~~ table dept Cascade Constraints;

THE CHECK CONSTRAINT :-

It is used to apply Business Rule validations. CHECK constraints must be specified as a logical expression that evaluate either to True or False.

NOTE:→ A CHECK constraint takes longer to execute as compared to NOT NULL, PRIMARY KEY, FOREIGN KEY or UNIQUE. Thus it should be avoided.

Column level

```
create table emp
 (   empno  number(5) Primary key,
     deptno  number(5) references dept,
     ename varchar2(20) check (ename=upper(ename)),
     Hiredate  date  default sysdate,
      =
      =
 );
SQL> create table emp(
     empno   number(5) Primarykey,
      =
     check (ename=upper(ename)),
     check (sal>1500));
```

Restrictions on check constraint:↓

(1) A CHECK integrity constraints requires that a condition be true or unknown for the row to be processed.

eg:- Check (comm>0)

if we put NULL for comm column for a particular row then check will not be violated because NULL is an unknown value and not a zero or negative number.

(2) The condition cannot contain a subquerry or sequences.

(3) The Condition cannot include the SYSDATE, UID, USER --- functions.

MODIFYING TABLE STRUCTURE

① Adding New columns :↴

Eg)
```
Alter table emp
      add (empskills varchar2(20), empexp varchar2
           (10), ---);
```

② Modify existing columns :↴

Eg)
```
Alter table emp
      Modify (empskills varchar2 (25));
```

Restrictions on the alter table :↴

Using alter table clause you cannot perform the following tasks :↴

- ↳ change the name of the table
- ↳ change the name of the column.
- ↳ Drop a column.
- ↳ Decrease the size of a column if table data exists.

Naming Constraints

We should name all constraints applied so as to facilitate their enabling/disabling and other operations at some later stage.

eg:-

```
SQL) create tableemp
     ( empno  number(5) constraint cons
         PRIMARY KEY,
       ename varchar2(20) constraint cons_ck
          ,

       deptno number(5) constraint cons_fk_dept
        references dept(deptno));
```

Defining Integrity Constraints in the ALTER TABLE:

1.) Add Primary key on column supplier_no in table supplier-master;

```
SQL> alter table supplier-master
        add Primary key (supplier-no);
```

2.) Add foreign key constraint on column s_orderno in table sales_order_details referencing taple salesorder's column named sorderno.

```
SQL> alter table sales_order_details
       add constraint cons-fk-s_orderno
         FOREIGN KEY (s_orderno) references
         sales_order (s_orderno);
```

Dropping a constraint:→

```
SQL> alter table emp
       drop primary key
SQL> alter table sales_order_details
       drop constraint cons-fk-s_order_no;
```

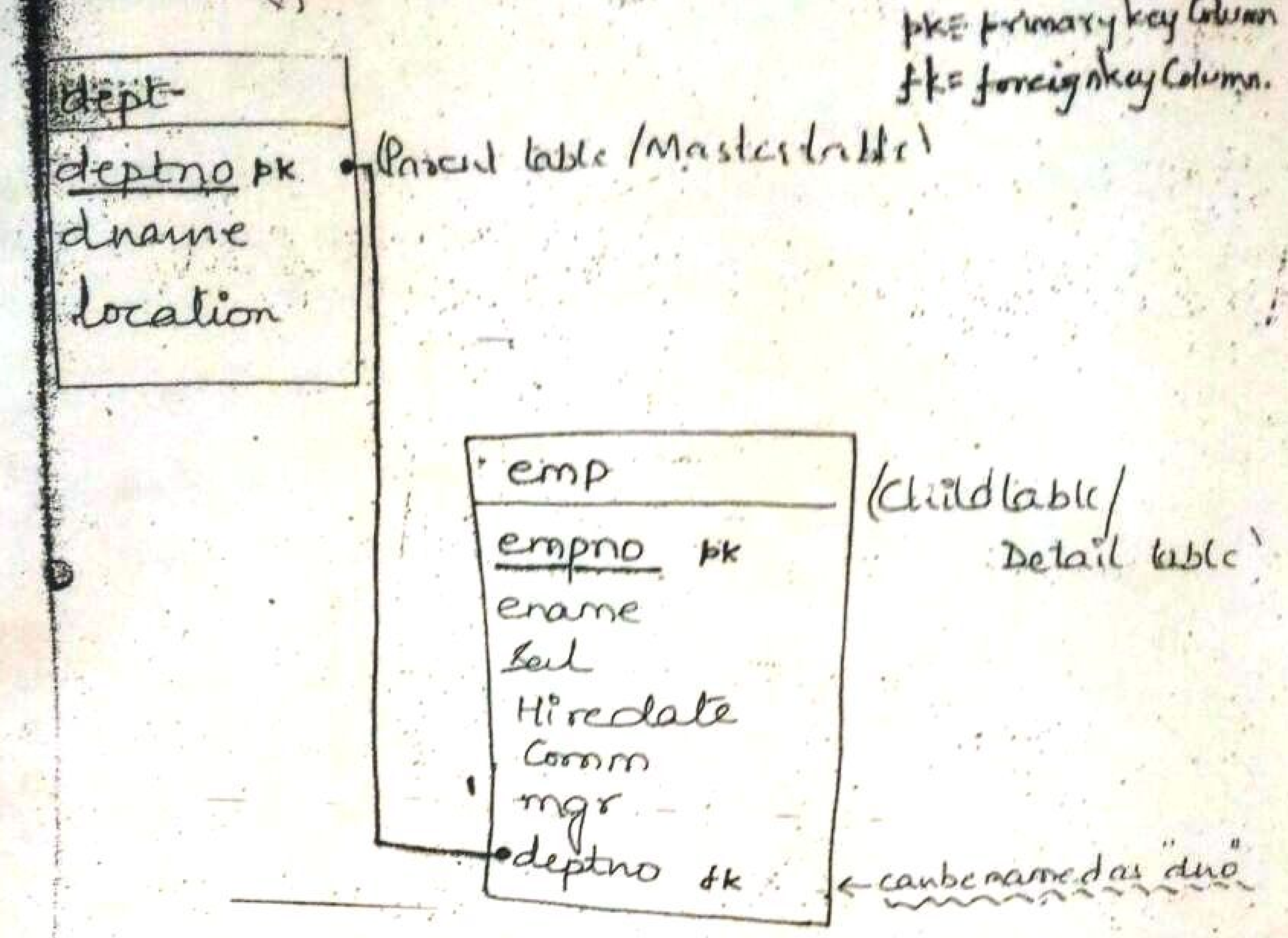
Note :→ Same way we can ENABLE/DISAB the Constraints.

Note: → If we don't specify (name) name a constraint, oracle assigns it a default name of the type

SYS_Cn

where n is an integer that makes the name unique in the database.

Creating tables in SQL



1.) first create "dept" table, i.e. parent table should exist before creating a dependent (child table "emp") table on it.

2.) Secondly create the child table with a foreign key (deptno).

3.) Primary key of Parent table (dept) i.e. deptno in the child table (emp). behaves as foreign key i.e column with name deptno defined in "emp" table is the foreignkey column. It is also not necessary to name this column as deptno in emp i.e. table we can change name of the column say "dno" query. to create dept & emp : √

```
create table dept(
    deptno   number(5) primarykey,
    dname  varchar2(10) check(dname=UPPER(dname)),
    loc    varchar2(10) check(loc = UPPER(loc))); ↵

1 Table(s) created

) create table emp(
    empno    number(5) Constraint Cons_Pk_emp
                  primary key,
    ename  varchar2(20) Constraint Cons_ck_emp
                  check(ename = UPPER (ename)),
    Sal    number(7,2) NOT NULL,
    Hiredate   date  default  SYSDATE,
    comm       number(7,2),
    deptno   number(4) Constraint Cons_fk_emp
            references dept(deptno)); ↵

    1 Table(s) created.
```

WAYS TO INSERT ROW IN A TABLE:

emp

empno	ename	sal
100	ALLEN	10000
101	SMITH	20000
102	AJAY	15000

empno → number(4)
ename → varchar2
sal → number(7,2)

```
SQL> insert into emp values(100,'ALLEN',10000);

SQL> insert into emp(ename,empno,sal) values
        ('SMITH',101,20000);

SQL> insert into emp values(&empno,'&ename',&sal);
o/p
  Enter value for empno: 102
  Enter value for ename: AJAY
  Enter value for sal: 15000
  1 row(s) inserted.
```

SYNONYM :→

Synonym is the different name for a table.

General Syntax :↓

```
Create synonym syn-name for table name;
```

eg→

```
Create synonym myemp for emp;
```

Note1:→ If we rename our original table emp to emp1, then synonym myemp will no longer work.

Note2:→ Utility:↓

We can grant synonym to any user and later if user drops the synonym than in such a case original table remains unaltered.

Note3:→ To see all synonyms created for a particuler table we can refer datadictionary table user-synonyms as follows:↓

```
sql> select synon-name from user-synonym
     where table-name = 'EMP';
```

Dropping Indexes:- ↓

```
sql) Drop Index emp_indx;
```

Note:→ If a user drops the primary key unique key constraint or the table ; oracle automatically drops the Indexes on the primary key column, unique key or the table itself.

INDEXES :→

An index, is an ordered list of contents of a columns or groups of columns in a table. An index created on the single column of the table is called SIMPLE INDEX. An index created on multiple Columns of the table is called Composite Index.

Generalized syntax :→

1.) Simple Index

```
CREATE INDEX indexfilename
    ON TABLENAME (columnname);
```

eg:-

```
create index emp_ndx
    on emp(empno);
```

2.) Composite Index.

```
CREATE INDEX indexfilename
    ON tablename(column1, column2, ---);
```

eg:-

```
create index salesorder_details_ndx
    on sales_order_details(s_orderno,
                           product_no);
```

NOTE :→ Indexes in the above examples donotenforce uniqueness i.e. the column included in the index can have duplicate values. To create UNIQUE Index do use UNIQUE keyword as

eg:→

```
create UNIQUE index emp_ndx
    on emp(empno);
```

stances when the Oracle Engine uses an index for data extraction:

↳ A SELECT statement with WHERE clause specified on the column on which an index exists

↳ A SELECT statement with ORDER BY clause specified on the column on which an index exists.

nstances when the Oracle engine does not use an index for data extraction:

↳ A SELECT statement without search criteria & order by clause.

↳ A SELECT statement with WHERE clause specified on the column on which an index is not defined.

↳ A SELECT statement with ORDERBY clause specified on the column on which an index is not defined.