DATAINTEGRITY:

TO MAINTAIN ACCURATE & CONSISTENCY DATA IN DB TABLES.THIS DATAINTEGRITY AGAIN CLASSIFIED INTO TWO WAYS THOSE ARE

1. DECLARATIVE / PRE -DEFINE DATAINTEGRITY (USING CONSTRAINTS)

2. PROCEDURAL / USER - DEFINE DATAINTEGRITY (USING TRIGGERS)

DECLARATIVE / PRE -DEFINE DATAINTEGRITY:

- THIS DATAINTEGRITY CAN BE IMPLEMENTED WITH HELP OF "CONSTRAINTS". THESE ARE AGAIN THREE TYPES,

- ENTITY INTEGRITY

- REFERENCIAL INTEGRITY

- DOMAIN INTEGRITY

ENTITY INTEGRITY:

- IT ENSURE THAT EACH ROW UNIQULY IDENTIFY IN A TABLE.TO IMPLEMENT THIS MECHANISM WE USE PRIMARY KEY OR UNIQUE CONSTRAINT.

REFERENCIAL INTEGRITY:

- IT ENSURE THAT TO CREATE RELATIONSHIP BETWEEN TABLES.TO IMPLEMENT THIS MECHANISM THEN WE USE FOREIGN KEY (REFERENCIAL KEY).

DOMAIN INTEGRITY:

- DOMAIN IS NOTHING BUT COLUMN.IT ENSURE THAT TO CHECK VALUES WITH USER DEFINE CONDITION BEFORE ACCEPTING VALUES INTO A COLUMN.TO PERFORM THIS MECHANISM WE USE CHECK, DEFAULT, NOT NULL CONSTRAINTS.

CONSTRAINTS:

- CONTSRAINTS ARE USED TO RESTRICTED UNWANTED(INVALID) DATA INTO TABLE.ALL DATABASES ARE SUPPORTING THE FOLLOWING CONSTRAINT TYPES ARE

- UNIQUE

- NOT NULL

- CHECK

- PRIMARY KEY

- FOREIGN KEY (REFERENCES KEY)

- DEFAULT

- ALL DATABASES ARE SUPPORTING THE FOLLOWING TWO TYPES OF METHODS TO DEFINE CONSTRAINTS.THOSE ARE

I) COLUMN LEVEL:

- IN THIS METHOD WE ARE DEFINING CONSTRAINTS ON INDIVIDUAL COLUMNS.

SYNTAX:

CREATE TABLE <TN> (<COLUMN NAME1><DATATYPE>[SIZE] <CONSTRAINT TYPE>, ……);

II) TABLE LEVEL:

- IN THIS METHOD WE ARE DEFINING CONSTRAINTS AFTER ALL COLUMNS ARE DECLARED. (END OF THE TABLE DEFINITION)

SYNTAX:

CREATE TABLE <TN> (<COLUMN NAME1><DATATYPE>[SIZE], <COLUMN NAME2><DATATYPE>[SIZE], ……, <CONSTRAINT TYPE> (<COLUMN NAME1>, <COLUMN NAME2>, ……));

UNIQUE:

- TO RESTRICTED DUPLICATE VALUES BUT ACCEPTING NULLS INTO A COLUMN.

I) COLUMN LEVEL:

EX:

SQL> CREATE TABLE TEST1(SNO INT UNIQUE, NAME VARCHAR2(10) UNIQUE);

TESTING:

SQL> INSERT INTO TEST1 VALUES(1,'A');---ALLOWED

SQL> INSERT INTO TEST1 VALUES(1,'A');---NOT ALLOWED

SQL> INSERT INTO TEST1 VALUES(NULL,NULL);----ALLOWED

II) TABLE LEVEL:

EX:

SQL> CREATE TABLE TEST2(SNO INT, NAME VARCHAR2(10), UNIQUE (SNO, NAME));

TESTING:

SQL> INSERT INTO TEST2 VALUES(1,'A');---ALLOWED

SQL> INSERT INTO TEST2 VALUES(1,'A');---NOT ALLOWED

SQL> INSERT INTO TEST2 VALUES(2,'A');---ALLOWED

NOTE: WHEN WE APPLY UNIQUE CONSTRAINT ON GROUP OF COLUMNS THEN WE CALLED AS "COMPOSITE UNIQUE" CONSTRAINT.IN THIS MECHANISM INDIVIDUAL COLUMNS ARE ACCEPTING DUPLICATE VALUES BUT DUPLICATE COMBINATION OF COLUMNS DATA IS NOT ALLOWED.

NOT NULL:

- TO RESTRICTED NULLS BUT ACCEPTING DUPLICATE VALUES INTO A COLUMN.

- NOT NULL CONSTRAINT NOT SUPPORTS "TABLE LEVEL".

COLUMN LEVEL:

EX:

SQL> CREATE TABLE TEST3(STID INT NOT NULL, SNAME VARCHAR2(10) NOT NULL);

TESTING:

SQL> INSERT INTO TEST3 VALUES(101,'A');----ALLOW

SQL> INSERT INTO TEST3 VALUES(101,'A');---ALLOW

SQL> INSERT INTO TEST3 VALUES(NULL,NULL);----NOT ALLOW

CHECK:

- TO CHECK VALUES WITH USER DEFINED CONDITION BEFORE ACCEPTING VALUES INTO A COLUMN.

I) COLUMN LEVEL:

EX:

SQL> CREATE TABLE TEST4(EID INT, SAL NUMBER(10) CHECK(SAL>=10000));

TESTING:

SQL> INSERT INTO TEST4 VALUES (1021,9500);---NOT ALLOW

SQL> INSERT INTO TEST4 VALUES (1021,10000);---ALLOW

II) TABLE LEVEL:

EX:

SQL> CREATE TABLE TEST5(ENAME VARCHAR2(10),SAL NUMBER(10),

CHECK(ENAME=LOWER(ENAME) AND SAL>8000));

TESTING:

SQL> INSERT INTO TEST5 VALUES('SAI',7500);---NOT ALLOW

SQL> INSERT INTO TEST5 VALUES('SAI',9500);---ALLOW

PRIMARY KEY:

==========

- TO RESTRICTED DUPLICATES & NULLS INTO A COLUMN.

- A TABLE SHOULD HAVE ONLY "ONE PRIMARY KEY".

I) COLUMN LEVEL:

EX:

SQL> CREATE TABLE TEST6(PCODE INT PRIMARY KEY, PNAME VARCHAR2(10)

PRIMARY KEY);

ERROR AT LINE 1:

ORA-02260: TABLE CAN HAVE ONLY ONE PRIMARY KEY.

SOL:

SQL> CREATE TABLE TEST6(PCODE INT PRIMARY KEY, PNAME VARCHAR2(10));

TESTING:

SQL> INSERT INTO TEST6 VALUES(10021,'C');-----ALLOW

SQL> INSERT INTO TEST6 VALUES(10021,'C++');----NOT ALLOW

SQL> INSERT INTO TEST6 VALUES(NULL,'C++');----NOT ALLOW

II) TABLE LEVEL:

EX:

SQL> CREATE TABLE TEST7(BCODE INT, BNAME VARCHAR2(10),

LOC VARCHAR2(10), PRIMARY KEY (BCODE, BNAME));

TESTING:

SQL> INSERT INTO TEST7 VALUES (1021,'SBI','SRNAGAR');---ALLOW

SQL> INSERT INTO TEST7 VALUES (1021,'SBI','MADHAPUR');---NOT ALLOW

SQL> INSERT INTO TEST7 VALUES (1022,'SBI','MADHAPUR');---ALLOW

SQL> INSERT INTO TEST7 VALUES (1021,'ICICI','SRNAGAR');----ALLOW

NOTE: WHEN WE APPLY PRIMARY KEY CONSTRAINT ON GROUP OF COLUMNS THEN WE CALLED AS "COMPOSITE PRIMARY KEY" CONSTRAINT.IN THIS MECHANISM INDIVIDUAL COLUMNS ARE ACCEPTING DUPLICATE VALUES BUT DUPLICATE COMBINATION OF COLUMNS DATA IS NOT ALLOWED.

FOREIGN KEY (REFERENCES KEY):

- FOREIGN KEY IS USED TO ESTABLISH RELATIONSHIP BETWEEN TABLES.

BASIC THINGS:

1. WE HAVE A COMMON COLUMN NAME(OPTIONAL) BUT RECOMMENDED.

2. COMMON COLUMN DATATYPE MUST MATCH.

3. ONE TABLE FOREIGN KEY MUST BELONGS TO ANOTHER TABLE PRIMARY KEY.

AND HERE PRIMARY KEY & FOREIGN KEY COLUMN MUST BE COMMON COLUMN.

4. PRIMARY KEY TABLE IS CALLED AS "PARENT TABLE" AND FOREIGN KEY TABLE IS CALLED AS "CHID TABLE"(I.E PARENT & CHILD RELATIONSHIP).

5. FOREIGN KEY COLUMN VALUES SHOULD BE MATCH WITH PRIMARY KEY COLUMN

VALUES ONLY.

6. GENERALLY PRIMARY KEY IS NOT ALLOWED DUPLICATE AND NULL VALUES WHERE AS FOREIGN KEY IS ALLOWED DUPLICATE & NULL VALUES.

I) COLUMN LEVEL:

SYNTAX:

<COMMON COLUMN NAME OF CHILD> <DT>[SIZE] REFERENCES

<PARENT TABLE NAME> (<COMMON COLUMN NAME OF PARENT>)

EX:

STEP1:

```
SQL> CREATE TABLE DEPT1(DEPTNO INT PRIMARY KEY, DNAME
VARCHAR2(10));
```

STEP2:

```
SQL> INSERT INTO DEPT1 VALUES (10,'ORACLE');
SQL> INSERT INTO DEPT1 VALUES (20,'JAVA');
```

STEP3:

```
SQL> CREATE TABLE EMP1(EID INT PRIMARY KEY, ENAME VARCHAR2(10),
        DEPTNO INT REFERENCES DEPT1(DEPTNO));
```

STEP4:

```
SQL>INSERT INTO EMP1 VALUES (1021,'SAI',10);
SQL>INSERT INTO EMP1 VALUES (1022,'JONES',10);
```

SQL>INSERT INTO EMP1 VALUES (1023,'MILLER',20);

- ONCE WE ESTABLISH RELATIONSHIP BETWEEN TABLES THERE ARE TWO RULES ARE COME INTO PICTURE.THOSE ARE

1) INSERTION RULE:

- WE CANNOT INSERT VALUES INTO FOREIGN KEY(REFERENCES KEY) COLUMN THOSE VALUES ARE NOT EXISTING UNDER PRIMARY KEY COLUMN OF PARENT TABLE.

EX:

SQL> INSERT INTO EMP1 VALUES (1026,'SCOTT',30);

ERROR AT LINE 1:

ORA-02291: INTEGRITY CONSTRAINT (SCOTT.SYS_C005468) VIOLATED - PARENT KEY NOT FOUND.

2) DELETION RULE:

- WHEN WE TRY TO DELETE A RECORD FROM PARENT TABLE AND THOSE ASSOCIATED RECORDS ARE AVAILABLE IN CHILD TABLE THEN ORACLE RETURNS

AN ERROR IS,

EX:

SQL> DELETE FROM DEPT1 WHERE DEPTNO=20;

ERROR AT LINE 1:

ORA-02292: INTEGRITY CONSTRAINT (SCOTT.SYS_C005468) VIOLATED - CHILD RECORD FOUND.

NOTE:

IF WE WANT TO DELETE A RECORD FROM PARENT TABLE WHEN THEY HAVE CORRESPONDING CHILD RECORDS IN CHILD TABLE THEN WE PROVIDE SOME SET OF RULES TO PERFORM DELETE OPERATIONS ON PARENT TABLE.THOSE RULES ARE CALLED AS "CASCADE RULES".

I) ON DELETE CASCADE

II) ON DELETE SET NULL

I) ON DELETE CASCADE:

- WHENEVER WE ARE DELETING A RECORD FROM PARENT TABLE THEN THAT ASSOCIATED CHILD RECORDS ARE DELETED FROM CHILD TABLE AUTOMATICALLY.

EX:

STEP1:

```
SQL> CREATE TABLE DEPT2(DEPTNO INT PRIMARY KEY,DNAME
VARCHAR2(10));
```

STEP2:

```
SQL> INSERT INTO DEPT2 VALUES (10,'ORACLE');
SQL> INSERT INTO DEPT2 VALUES (20,'JAVA');
```

STEP3:

```
SQL> CREATE TABLE EMP2(EID INT PRIMARY KEY, ENAME VARCHAR2(10),
      DEPTNO INT REFERENCES DEPT2(DEPTNO) ON DELETE CASCADE);
```

STEP4:

```
SQL>INSERT INTO EMP2 VALUES (1021,'SAI',10);
SQL>INSERT INTO EMP2 VALUES (1022,'JONES',10);
SQL>INSERT INTO EMP2 VALUES (1023,'MILLER',20);
```

TESTING:

```
SQL> DELETE FROM DEPT2 WHERE DEPTNO=20; ----ALLOWED
```

II) ON DELETE SET NULL:

WHENEVER WE ARE DELETING A RECORD FROM PARENT TABLE THEN THAT ASSOCIATED CHILD RECORDS ARE SET TO NULL IN CHILD TABLE AUTOMATICALLY.

EX:

STEP1:

```
SQL> CREATE TABLE DEPT3(DEPTNO INT PRIMARY KEY,DNAME
VARCHAR2(10));
```

STEP2:

SQL> INSERT INTO DEPT3 VALUES (10,'ORACLE');

SQL> INSERT INTO DEPT3 VALUES (20,'JAVA');

STEP3:

SQL> CREATE TABLE EMP3(EID INT PRIMARY KEY, ENAME VARCHAR2(10),

DEPTNO INT REFERENCES DEPT3(DEPTNO) ON DELETE SET NULL);

STEP4:

SQL>INSERT INTO EMP3 VALUES (1021,'SAI',10);

SQL>INSERT INTO EMP3 VALUES (1022,'JONES',10);

SQL>INSERT INTO EMP3 VALUES (1023,'MILLER',20);

TESTING:

SQL> DELETE FROM DEPT3 WHERE DEPTNO=10; ----ALLOWED

SYNTAX FOR TABLE LEVEL:

CREATE TABLE <TN>(<COL1><DT>[SIZE], <COL2><DT>[SIZE], ……., FOREIGN KEY(<COL1>, <COL2>, ……..) REFERENCES <PARENT TABLE NAME>(<COL1>, <COL2>, ……);

PRE-DEFINE CONSTRAINT NAME:

- WHENEVER WE ARE APPLYING CONSTRAINT ON A PARTICULAR COLUMN THEN DB SERVER(SYSTEM) INTERNALLY GENERATE AN UNIQUE ID NUMBER (OR) AN UNIQUE CONSTRAINT KEY NAME AUTOMATICALLY FOR IDENTIFYING A CONSTRANT.

EX:

SQL> CREATE TABLE TEST8(SNO INT PRIMARY KEY, NAME VARCHAR2(10));

TESTING:

SQL> INSERT INTO TEST8 VALUES(1,'A');---ALLOWED

SQL> INSERT INTO TEST8 VALUES(1,'B');---NOT ALLOWED

ERROR:

ORA-00001: UNIQUE CONSTRAINT (SCOTT.SYS_C005475) VIOLATED

USER DEFINE CONSTRAINT NAME:

- IN PLACE OF PRE-DEFINE CONSTRAINT NAME WE CAN ALSO CREATE A USER DEFINED CONSTRAINT KEY NAME (OR) CONSTRAINT ID FOR IDENTIFYING A CONSTRAINT.

SYNTAX:

<COLUMN NAME> <DT>[SIZE] CONSTRAINT <USER DEFINED CONSTRAINT NAME> <CONSTRAINT TYPE>

EX:

```
SQL> CREATE TABLE TEST10(SNO INT CONSTRAINT PK_SNO PRIMARY KEY,
NAME VARCHAR2(10) CONSTRAINT UQ_NAME UNIQUE);
```

TESTING:

```
SQL> INSERT INTO TEST10 VALUES(1,'A');


SQL> INSERT INTO TEST10 VALUES(1,'B');

ERROR AT LINE 1:

ORA-00001: UNIQUE CONSTRAINT (SCOTT.PK_SNO) VIOLATED


SQL> INSERT INTO TEST10 VALUES(2,'A');

ERROR AT LINE 1:

ORA-00001: UNIQUE CONSTRAINT (SCOTT.UQ_NAME) VIOLATE
```

DATA DICTIONARIES (OR) READ ONLY TABLES:

- WHENEVER WE ARE INSTALLING ORACLE S/W INTERNALLY ORACLE SERVER IS CREATING SOME PRE-DEFINE TABLES ARE CALLED AS "DATA DICTIONARIES". THESE DATA DICTIONARIES ARE USED TO STORE THE INFORMATION ABOUT DB OBJECTS SUCH AS TABLES,INDEXES,VIEWS,SYNONYMS,.....ETC.

- THESE DATA DICTIONARIES ARE SUPPORTING "SELECT" AND "DESC" COMMANDS ONLY.SO THAT DATA DICTIONARIES ARE ALSO CALLED AS "READ ONLY TABLES" IN ORACLE DB.

- IF WE WANT TO VIEW ALL DATA DICTIONARIES IN ORACLE DB THEN WE FOLLOW THE FOLLOWING SYNTAX IS,

SYNTAX:

```
SQL> SELECT * FROM DICT;
```

NOTE1:

IF WE WANT TO VIEW ALL CONSTRAINTS INFROMATION OF A PARTICULAR TABLE THEN WE USE "USER_CONSTRAINTS" DATA DICTIONARY.

EX:

```
SQL> DESC USER_CONSTRAINTS;

SQL> SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE FROM
USER_CONSTRAINTS WHERE TABLE_NAME='TEST10';


CONSTRAINT_NAME                CONSTRAINT_TYPE
-------------------------------    ------------------------------
PK_SNO                         P
UQ_NAME                        U
```

NOTE2:

- IF WE WANT TO VIEW CONSTRAINT NAME ALONG WITH COLUMN NAME OF A PARTICULAR TABLE THEN WE USE " USER_CONS_COLUMNS " DATADICTIONARY.

EX:

```
SQL> DESC USER_CONS_COLUMNS;

SQL> SELECT CONSTRAINT_NAME, COLUMN_NAME FROM
USER_CONS_COLUMNS WHERE TABLE_NAME='TEST10';


CONSTRAINT_NAME                COLUMN_NAME
-------------------------------    ---------------------------
UQ_NAME                        NAME
PK_SNO                         SNO
```

NOTE3:

TO VIEW A LOGICAL CONDITION OF CHECK CONSTRAINT THEN WE NEED TO CALL "SEARCH_CONDITION" COLUMN FROM "USER_CONSTRAINTS" DATA DICTIONARY.

EX:

```
SQL> CREATE TABLE TEST11(SNO INT, SAL NUMBER(10) CHECK(SAL>5000));
```

EX:

```
SQL> DESC USER_CONSTRAINTS;

SQL> SELECT SEARCH_CONDITION FROM USER_CONSTRAINTS
       WHERE TABLE_NAME='TEST11';
```

SEARCH_CONDITION

SAL>5000

NOTE4:

TO VIEW ALL COLUMNS INFORMATION OF A PARTICULAR TABLE THEN WE USE "USER_TAB_COLUMNS" DATADICTIONARY.

EX:

```
SQL> DESC USER_TAB_COLUMNS;

SQL> SELECT COLUMN_NAME FROM USER_TAB_COLUMNS
       WHERE TABLE_NAME='EMP';
```

HOW TO FIND NO. OF ROWS IN A TABLE:

```
SQL> SELECT COUNT (*) FROM EMP;
```

COUNT (*)

14

HOW TO FIND NO. OF COLUMNS IN A TABLE:

```
SQL> SELECT COUNT (*) FROM USER_TAB_COLUMNS WHERE
TABLE_NAME='EMP';
```

COUNT (*)

8

HOW TO ADD CONSTRAINTS TO AN EXISTING TABLE:

SYNTAX:

=======

ALTER TABLE <TN> ADD CONSTRAINT <CONSTRAINT KEY NAME> <CONSTRAINT TYPE> (<COLUMN NAME>);

EX:

SQL> CREATE TABLE TEST12(EID INT, ENAME VARCHAR2(10),SAL NUMBER(10));

I) ADDING PRIMARY KEY:

SQL> ALTER TABLE TEST12 ADD CONSTRAINT PK_EID PRIMARY KEY(EID);

II) ADDING UNIQUE, CHECK CONSTRAINT:

SQL> ALTER TABLE TEST12 ADD CONSTRAINT UQ_ENAME UNIQUE(ENAME);

SQL> ALTER TABLE TEST12 ADD CONSTRAINT CHK_SAL CHECK(SAL=10000);

III) ADDING "NOT NULL" CONSTRAINT:

SYNTAX:

ALTER TABLE <TN> MODIFY <COLUMN NAME> CONSTRAINT <CONSTRAINT KEY NAME>

NOT NULL;

EX:

SQL> ALTER TABLE TEST12 MODIFY ENAME CONSTRAINT NN_ENAME NOT NULL;

IV) ADDING FOREIGN KEY CONSTRAINT:

SYNTAX:

ALTER TABLE <TN> ADD CONSTRAINT <CONSTRAINT KEY NAME>FOREIGN KEY (<COMMON COLUMN OF CHILD TABLE>) REFERENCES<PARENT TABLE> (<COMMON COLUMN OF PARENT TABLE>) ON DELETE CASCADE / ON DELETE SET NULL;

EX:

SQL> CREATE TABLE TEST13(DNAME VARCHAR2(10), EID INT);

TABLE CREATED.

EX:

SQL> ALTER TABLE TEST13 ADD CONSTRAINT FK_EID FOREIGN KEY(EID)

REFERENCES TEST12(EID) ON DELETE CASCADE;

HOW TO DROP CONSTRAINT FROM AN EXISTING TABLE:

SYNTAX:

ALTER TABLE <TN> DROP CONSTRAINT <CONSTRAINT KEY NAME>;

I) DROPPING PRIMARY KEY:

METHOD1:

SQL> ALTER TABLE TEST13 DROP CONSTRAINT FK_EID; -------FIRST

SQL> ALTER TABLE TEST12 DROP CONSTRAINT PK_EID; --------LATER

METHOD2:

- WHEN WE DROP PRIMARY KEY ALONG WITH FOREIGN KEY CONSTRAINT FROM PARENT AND CHILD TBALES THEN WE USE "CASCADE" STATEMENT.

EX:

SQL> ALTER TABLE TEST12 DROP CONSTRAINT PK_EID CASCADE;

II) DROPPING UNIQUE, CHECK, NOT NULL CONSTRANT:

SQL> ALTER TABLE TEST12 DROP CONSTRAINT UQ_ENAME;

SQL> ALTER TABLE TEST12 DROP CONSTRAINT CHK_SAL;

SQL> ALTER TABLE TEST12 DROP CONSTRAINT NN_ENAME;

HOW TO RENAME CONSTRAINT NAME:

SYNTAX:

ALTER TABLE <TN> RENAME CONSTRAINT < OLD CONSTRAINT NAME> TO <NEW CONSTRAINT NAME>;

EX:

SQL> CREATE TABLE TEST14(SNO INT PRIMARY KEY);

SQL> SELECT CONSTRAINT_NAME FROM USER_CONS_COLUMNS

WHERE TABLE_NAME='TEST14';

CONSTRAINT_NAME

SYS_C005489

SQL> ALTER TABLE TEST14 RENAME CONSTRAINT SYS_C005489 TO SNO_PK;

SQL> SELECT CONSTRAINT_NAME FROM USER_CONS_COLUMNS

WHERE TABLE_NAME='TEST14';

CONSTRAINT_NAME

SNO_PK

HOW DISABLE / ENABLE CONSTRAINT:

- BY DEFAULT, CONSTRAINTS ARE ENABLE MODE.IF WE WANT TO DISABLE CONSTRAINT TEMP. THEN WE USE "DISABLE" KEYWORD.IT MEANS THAT CONSTRAINT IS EXISTING IN DB BUT NOT WORK TILL IT MAKE AS "ENABLE".

- WHENEVER WE WANT TO COPY HUGE AMOUNT OF DATA FROM ONE TABLE TO ANOTHER TABLE THERE WE USE "DISABLE" KEYWORD.

SYNTAX:

ALTER TABLE <TN> DISABLE / ENABLE CONSTRAINT <CONSTRAINT KEY NAME>;

EX:

SQL> CREATE TABLE TEST15(ENAME VARCHAR2(10), SAL NUMBER (10)

CHECK(SAL=5000));

SQL> INSERT INTO TEST15 VALUES('SAI',5000); ----ALLOWED

SQL> INSERT INTO TEST15 VALUES('JONES',3000); -----NOT ALLOWED

ERROR AT LINE 1:

ORA-02290: CHECK CONSTRAINT (SCOTT.SYS_C005492) VIOLATED

SQL> ALTER TABLE TEST15 DISABLE CONSTRAINT SYS_C005492;

SQL> INSERT INTO TEST15 VALUES('JONES',3000); ----ALLOWED

EX:

SQL> ALTER TABLE TEST15 ENABLE CONSTRAINT SYS_C005492;

ERROR AT LINE 1:

ORA-02293: CANNOT VALIDATE (SCOTT.SYS_C005492) - CHECK CONSTRAINT VIOLATED

- TO OVERCOME THE ABOVE PROBLEM THEN WE USE "NOVALIDATE" KEYWORD AT THE TIME OF ENABLE CONSTRAINT.ONCE WE USE "NOVALIDATE" KEYWORDTHEN CONSTRAINT IS ENABLE WITH "NOVALIDATE" AND ORACLE SERVER WILL NOT CHECK EXISTING DATA IN TABLE BUT CHECKING NEW DATA WHILE INSERTING TIME.

EX:

SQL> ALTER TABLE TEST15 ENABLE NOVALIDATE CONSTRAINT SYS_C005492;

TABLE ALTERED.

TESTING:

SQL> INSERT INTO TEST15 VALUES('SCOTT',6000); ---NOT ALLOWED

ERROR AT LINE 1:

ORA-02290: CHECK CONSTRAINT (SCOTT.SYS_C005492) VIOLATED

SQL> INSERT INTO TEST15 VALUES('SCOTT',5000); ------ALLOWED

DEFAULT CONSTRAINT:

- IT A SPECIAL TYPE OF CONSTRAINT WHICH IS USED TO ASSIGN A USER DEFINE DEFAULT VALUE TO A COLUMN.

SYNTAX:

<COLUMN NAME> <DATATYPE>[SIZE] DEFAULT <VALUE / EXPRESSION>

EX:

SQL> CREATE TABLE TEST17(SNO INT, SAL NUMBER (10) DEFAULT 5000);

TABLE CREATED.

TESTING:

SQL> INSERT INTO TEST17 VALUES (1,8500);

SQL> INSERT INTO TEST17(SNO)VALUES (2);

OUTPUT:

SNO	SAL
1	8500
2	5000

HOW TO ADD DEFAULT VALUE TO AN EXISTING TABLE:

SYNTAX:

ALTER TABLE <TN> MODIFY <COLUMN NAME> DEFAULT <VALUE / EXPRESSION>;

EX:

SQL> CREATE TABLE TEST18(EID INT, SAL NUMBER (10));

SQL> ALTER TABLE TEST18 MODIFY SAL DEFAULT 8000;

TESTING:

SQL> INSERT INTO TEST18(EID)VALUES (1021);

NOTE:

- IF WE WANT TO VIEW DEFAULT VALUE OF A COLUMN THEN WE USE "USER_TAB_COLUMNS" DATADICTIONARY.

EX:

SQL> DESC USER_TAB_COLUMNS;

SQL> SELECT COLUMN_NAME, DATA_DEFAULT FROM USER_TAB_COLUMNS

WHERE TABLE_NAME='TEST18';

COLUMN_NAME	DATA_DEFAULT
SAL	8000

HOW TO REMOVE DEFAULT VALUE OF A COLUMN:

EX:

ALTER TABLE TEST18 MODIFY SAL DEFAULT NULL;

COLUMN_NAME	DATA_DEFAULT
SAL	NULL