LET US INCREASE THE INTEGRITY AND QUALITY OF DATABASE USING CONSTRAINTS

Data Integrity in Data Bases:

- It Is a State in Which All the Data Values Stored in the Data Base Are Correct.
- Enforcing Data Integrity Ensures the Quality of the Data In the Data Base.

Categories of Data Integrity:

i. Entity Integrity.
ii. Domain Integrity
iii. Referential Integrity
iv. User Defined Integrity

i. Entity Integrity:

- It Defines a Row as a UNIQUE Entity for a Particular Table.
- Entity Integrity Enforces the Integrity of the Identifier Columns, or the PRIMARY KEY of a Table.

Illustration: 1

STUDENT NAME	DOB	DOA	COURSE NAME	COURSE FEE
Sampath	02-Jan-76	15-Jun-00	MCA	25000.00
Sampath	02-Jan-76	15-Jun-00	MBA	25000.00
Srinivas	02-Jan-76	15-Jun-00	MBA	25000.00
Srinivas	02-Jan-76	25-Aug-02	MBA	25000.00
Sampath	10-Dec-80	25-Aug-02	MCA	25000.00
Srinivas	02-Jan-76	25-Aug-02	M.SC	15000.00

Illustration : 2

STUDENT NAME	DOB	DOA	COURSE NAME	COURSE FEE	EMAIL ID
Sampath	02-Jan-76	15-Jun-00	MCA	25000.00	sampath@gmail.com
Sampath	02-Jan-76	15-Jun-00	MBA	25000.00	sampath@yahoo.com
Srinivas	02-Jan-76	15-Jun-00	MBA	25000.00	srinivas@gmail.com
Srinivas	02-Jan-76	25-Aug-02	MBA	25000.00	srinivas@gmail.com
Sampath	10-Dec-80	25-Aug-02	MCA	25000.00	sampath@rediff.com
Srinivas	02-Jan-76	25-Aug-02	M.SC	15000.00	srinivas@rediff.com

Illustration: 3

STUDENT ID	STUDENT NAME	DOB	DOA	COURSE NAME	COURSE FEE	EMAIL ID
1000	Sampath	02-Jan-76	15-Jun-00	MCA	25000.00	sampath@gmail.com
1001	Sampath	02-Jan-76	15-Jun-00	MBA	25000.00	sampath@yahoo.com
1002	Srinivas	02-Jan-76	15-Jun-00	MBA	25000.00	srinivas@gmail.com
1003	Srinivas	02-Jan-76	25-Aug-02	MBA	25000.00	srinivas@gmail.com
1004	Sampath	10-Dec-80	25-Aug-02	MCA	25000.00	sampath@rediff.com
1005	Srinivas	02-Jan-76	25-Aug-02	M.SC	15000.00	srinivas@rediff.com

ii. Domain Integrity:

- Domain Integrity Validates the Entries for a Given Column.
- Restricting Type (Data Type)
- by format (CHECK Constraint)
- by Range of Possible Values Using
    1. Foreign key Constraint
    2. Check Constraint
    3. DEFAULT Key Word
    4. NOT NULL Constraint.
- Domain Integrity Major Fact of Concentration is on the Data That is Being Collected in That Column.

Illustration: 4

STUDENT ID	STUDENT NAME	DOB	DOA	COURSE NAME	COURSE FEE	EMAIL ID
1000	Sampath	02-Jan-76	15-Jun-00	MCA	25000.00	sampath@gmail.com
1001	Sampath		15-Jun-00	MBA	25000.00	
1002	Srinivas	02-Jan-76	15-Jun-00	MBA	25000.00	srinivas@gmail.com
1003	Srinivas	02-Jan-78	04-Aug-07	MBA	25000.00	srinivas@gmail.com
1004	Sampath		25-Aug-02	MCA	25000.00	
1005	Srinivas	02-Jan-76		M.SC	15000.00	srinivas@rediff.com

iii. Referential Integrity:

- It Preserves the Defined Relationship When Records Are Entered Or Deleted.
- It Ensures the Key Values are Consistent Across the Tables.
- When Referential Integrity is Enforced, It Prevents from…
    - Adding Records to a Related Table if there is no associated Record in the Primary Table.
    - Changing Value in a Primary Table That Result in Orphaned Records in a Related Table.
    - Deleting Records from a Primary Table if there are Matching Related Records.

Illustration: 5

Course Information:

COURSE ID	COURSE NAME	COURSE FEES
COO1	MBA	25000
COO2	MCA	25000
COO3	M.Sc	15000

iv. User Defined Integrity

STUDENT ID	STUDENT NAME	DOB	DOA	COURSE ID	EMAIL ID
1000	Sampath	02-Jan-76	15-Jun-00	C002	sampath@gmail.com
1001	Sampath	02-Jan-76	15-Jun-00	C001	sampath@yahoo.com
1002	Srinivas	02-Jan-76	15-Jun-00	C001	srinivas@gmail.com
1003	Srinivas	02-Jan-76	25-Aug-02	C001	srinivas@gmail.com
1004	Sampath	10-Dec-80	25-Aug-02	C002	sampath@rediff.com
1005	Srinivas	02-Jan-76	25-Aug-02	C003	srinivas@rediff.com

- It Allows to Define Specific Business Rules That Do Not Fall into any one of the Other Integrity Categories.
- these are Business Rules Which Can be Handled at Run Time, Usually Designed Using Database Triggers in PLSQL.
- these are Rules Generally Specific to the Organizational Business Process.
- Can be any Situation That Looks Abnormal to the Current Systems Process.

Constraints in Oracle

Constraints in Data Bases Are Used to Define an Integrity Constraint, as a Rule That Restricts the Values in a Data Base.

As Per Oracle there are Six Types of Constraints……

i. NOT NULL Constraint.
ii. UNIQUE Constraint.
iii. PRIMARY KEY Constraint.
iv. FOREIGN KEY Constraint.
v. CHECK Constraint.
vi. REF Constraint.

Declaration Style:

- Column Level (OR) IN LINE Style.
- Table Level (OR) OUT OF LINE Style.

Column Level:

- they Are Declared as Part of the Definition of an Individual Columns or Attribute.
- Usually Applied When the Constraint is Specific to That Column only.

Table Level:

- they Are Declared as Part of the Table Definition.
- Definitely Applied When the Constraint is Applied on Combination of Columns together.

.

> Note: NOT NULL Constraint is the only Constraint Which Should Be Declared as INLINE only.

- Every Constraint is Managed by Oracle with a Constraint Name in the Meta Data.
- Hence When We Declare a Constraint if We Do Not Provide a Constraint Name Oracle associates the Constraint with Name.
- within a Single User No Two Constraints Can Have the Same Name.
- Rather Than Depending on the Oracle Supplied Constraint Name, it is Better to Define our Own Name for all Constraints.
- When Constraints are Named, We Should Use 'CONSTRAINT' Clause.

The CONSTRAINT Clause Can Appear in

- CREATE and ALTER Table Statement.
- CREATE and ALTER View Statement.

- Oracle Does Not Support Constraints on Columns or Attributes Whose Data Type is
  - USER_DEFINED OBJECTS.
  - NESTED TABLES and VARRAY.
  - REF and LOB.

Exceptions:

- NOT NULL Constraint is Supported for an Attribute Whose Data Type is USER_DEFINED Object, VARRAY, REF, LOB.
- NOT NULL, FOREIGN KEY, and REF Constraints Are Supported on a Column of TYPE REF.

NOT NULL Constraint:

- A NOT NULL Constraint Prohibits a Column from Containing NULL Values.
- NOT NULL Should Be Defined only At COLUMN Level.
- the Default Constraint if Not Specified is NULL Constraint.
- to Satisfy the Rule, Every Row in the Table Must Contain a Value for the Column.

Restrictions:

- NULL or NOT NULL Cannot Be Specified as View Constraints.
- NULL or NOT NULL Cannot Be Specified for an Attribute of an Object.

Syntax:

```
SQL> CREATE  TABLE  <Table_Name>
     (
      Column_Name 1 <Data  Type >(Width)  NOT  NULL,
      Column_Name 2<Data Type>(Width)
      CONSTRAINT  ConsName  NOT  NULL,
      Column_Name N  <Data Type>(Width)
     );
```

Illustration:

```
SQL> CREATE  TABLE  Students
          ( StudNo     NUMBER(6)  CONSTRAINT  StudnoNN  NOT  NULL,
```

```
StudName  VARCHAR2(25)  CONSTRAINT  StudNameNN  NOT  NULL,
CourseName   VARCHAR2(25)  CONSTRAINT  CourseNameNN  NOT  NULL,
JoinDate   DATE  NOT  NULL
 );
```

UNIQUE Constraint:

- the UNIQUE Constraint Designates a Column as a UNIQUE Key.
- A Composite UNIQUE Key Designates a Combination of Columns as the UNIQUE Key.
- A Composite UNIQUE Key is Always Declared at the Table Level.
- To Satisfy a UNIQUE Constraint, No Two Rows in the Table Can Have the Same Value for the UNIQUE Key.
- UNIQUE Key made Up of a Single Column Can Contain NULL Values.
- Oracle Creates an Index Implicitly on the UNIQUE Key Column.

Restrictions:

- A Table or View Can Have only one UNIQUE Key Column.
- UNIQUE Key Cannot Be Implemented on Columns Having
  - LOB, LONG, LONG RAW, VARRAY, NESTED TABLE, OBJECT, BFILE, REF, TIMESTAMP WITH TIME ZONE.
- A Composite UNIQUE Key Cannot Have More Than 32 Columns.
- Same Column or Combination of Columns Cannot Be Designated as Both PRIMARY KEY and UNIQUE KEY.
- We Cannot Specify a UNIQUE Key When Creating a Sub Table or Sub View in an Inheritance Hierarchy.
- the UNIQUE Key Can Be Specified only for the top Level (Root) Table or View.

Syntax:

```
SQL> CREATE Table <Table_Name>
(
Column_Name1 <Data Type>(Width) UNIQUE,
Column_Name 2<Data Type>(Width) CONSTRAINT ConsName UNIQUE,
Column_Name N <Data Type>(Width)
);
```

Illustration: 1

Column Level Syntax:

```
SQL> CREATE Table Promotions
(Promo_ID NUMBER(6) CONSTRAINT PromoIDUNQ UNIQUE,
PromoName VARCHAR2(20), PromoCategory VARCHAR2(15),
PromoCost NUMBER(10,2), PromoBegDate DATE, PromoEndDate DATE
);
```

Illustration: 2

Table Level Syntax:

```
SQL> CREATE Table Promotions
(Promo_ID NUMBER(6),
PromoName VARCHAR2(20), PromoCategory VARCHAR2(15),
PromoCost NUMBER(10,2), PromoBegDate DATE, PromoEndDate DATE,
CONSTRAINT PromoIDUNQ UNIQUE(Promo_ID) );
```

Illustration: 3

Composite UNIQUE Constraint Syntax:

```
SQL> CREATE Table WareHouse
 (WareHouseID NUMBER(6),
WareHouseName VARCHAR2(30), Area NUMBER(4), DockType VARCHAR2(50),
WaterAccess VARCHAR2(10), RailAccess VARCHAR2(10), Parking VARCHAR2(10),
Vclearance NUMBER(4),CONSTRAINT WareHouseUNQ
UNIQUE(WareHouseID,WareHouseName));
```

PRIMARY KEY Constraint:

- A PRIMARY KEY Constraint Designates a Column as the PRIMARY KEY of a TABLE or VIEW.
- A COMPOSITE PRIMARY KEY Designates a Combination of Columns as the PRIMARY KEY.
- When the Constraint is Declared At Column Level only PRIMARY KEY Keyword is Enough.

- A Composite PRIMARY KEY is Always Defined At Table Level only.
- A PRIMARY KEY Constraint Combines a NOT NULL and UNIQUE Constriant in one Declaration.

Restrictions:

- A TABLE or VIEW Can Have only one PRIMARY KEY.
- PRIMARY KEY Cannot Be Implemented on Columns Having…
  - LOB, LONG, LONG RAW, VARRAY, NESTED TABLE, OBJECT, BFILE, REF, TIMESTAMP WITH TIME ZONE.
- A Composite PRIMARY KEY Cannot Have More Than 32 Columns.
- the Same Column or Combination of Columns Cannot Be Designated Both as PRIMARY KEY and UNIQUE KEY.
- PRIMARY KEY Cannot Be Specified When Creating a Sub Table or Sub View in an Inheritance Hierarchy.
- the PRIMARY KEY Can be Specified only for the top Level (ROOT) TABLE or VIEW.

Syntax:

```
SQL> CREATE Table <Table_Name>

 (Column_Name 1 <Data Type> (Width) CONSTRAINT ColNamePK PRIMARY KEY,
Column_Name 2 <Data Type> (Width), Column_Name N <Data Type>(Width) ):
```

Illustration:1

Column Level Syntax:

```
SQL> CREATE TABLE Locations
(LocationID NUMBER(4) CONSTRAINT LocIDPK PRIMARY KEY, StAddress
VARCHAR2(40) NOT NULL, PostalCode VARCHAR2(6) CONSTRAINT PCNN NOT
NULL, City VARCHAR2(30) CONSTRAINT CityNN NOT NULL);
```

Illustration:2

Column Level Syntax:

```
SQL> CREATE TABLE Locations
(LocationID NUMBER(4) NOT NULL, StAddress VARCHAR2(40) NOT NULL,
PostalCode VARCHAR2(6) NOT NULL, City VARCHAR2(30) CONSTRAINT CityNN
NOT NULL, CONSTRAINT LocIDPK PRIMARY KEY(LocationID));
```

Analyze The Following Data for Primary Key:

Sale ID	Cust ID	ProdID	Qty	SaleDate	SaleDesc
S001	C001	P001	250.00	01-AUG-07	CASH
S001	C001	P002	125.00	01-AUG-07	CASH
S002	C002	P003	50.00	01-AUG-07	CASH
S002	C002	P004	75.00	01-AUG-07	CREDIT
S002	C002	P010	225.00	01-AUG-07	CREDIT
S002	C002	P003	125.00	01-AUG-07	CASH
S003	C001	P005	200.00	01-AUG-07	CREDIT
S003	C001	P002	25.00	01-AUG-07	CASH
S003	C001	P015	354.00	01-AUG-07	CREDIT
S004	C003	P100	245.00	02-AUG-07	CASH
S005	C001	P002	125.00	03-AUG-07	CASH
S006	C002	P004	75.00	03-AUG-07	CASH

Illustration:3

COMPOSITE PRIMARY KEY Syntax:

```
SQL> CREATE TABLE SalesInfo
   (SaleID NUMBER(6), CustID NUMBER(6), ProdID NUMBER(6), Quantity
NUMBER(6) NOT NULL, SaleDate DATE NOT NULL, SaleDesc LONG NOT NULL,
CONSTRAINT ProdCustIDPK PRIMARY KEY (SaleID, ProdID, CustID) );
```

FOREIGN KEY Constraint:

- It is Also Called as REFERENTIAL INTEGRITY CONSTRAINT.
- It Designates a Column as FOREIGN KEY and Establishes a RELATION Between the FOREIGN KEY and a Specified PRIMARY or UNIQUE KEY.
- A COMPOSITE FOREIGN KEY Designates a Combination of Columns as the FOREIGN KEY.

- the TABLE or VIEW Containing the FOREIGN KEY is Called the Child Object.
- the TABLE or VIEW Containing the REFERENCED KEY is Called the Parent Object.
- the FOREIGN KEY and the REFERENCED KEY Can Be in the Same TABLE or VIEW.
- the Corresponding Column or Columns of the FOREIGN KEY and the REFERENCED KEY Must Match in ORDER and DATA TYPE.
- A FOREIGN KEY CONSTRAINT Can Be Defined on a Single Key Column Either Inline or Out of Line Style.
- We Can Designate the Same Column or Combination of Columns as Both a FOREIGN Key and a PRIMARY or UNIQUE KEY.
- A COMPOSITE FOREIGN KEY CONSTRAINT, Must Refer to a COMPOSITE UNIQUE KEY or a COMPOSITE PRIMARY KEY in the PARENT TABLE or VIEW.

Restrictions:

- the FOREIGN KEY Columns Cannot be Applied on…
  - LOB, LONG, LONG RAW, VARRAY, NESTED TABLE, OBJECT, BFILE, REF, TIMESTAMP WITH TIME ZONE.
- the REFERENCED UNIQUE/PRIMARY KEY Constraint on the PARENT TABLE OR VIEW Must Already Be Detected.
- A COMPOSITE FOREIGN KEY Cannot Have More Than 32 Columns.
- the Child and Parent Tables Must Be on the Same Database.
- to Enable REFERENTIAL INTEGRITY Across Nodes of a Distributed Database TRIGGERS Are Used.

REFERENCES Clause:

- the REFERENCES CLAUSE Should be Used When the FOREIGN KEY Constraint is INLINE.
- When the Constraint is OUT OF LINE, We Must Specify the FOREIGN KEY, Key Word.

on DELETE Clause:

- the ON DELETE Clause Determines How ORACLE Automatically Maintains REFERENTIAL INTEGRITY if the REFERENCED PRIMARY or UNIQUE KEY Value is Removed from Master Table.
- CASCADE Option Can Be Specified if We Want ORACLE to Remove DEPENDENT FOREIGN KEY Values.

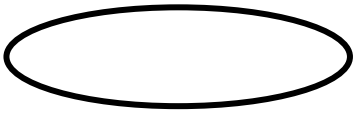
- Specify SET NULL if We Want ORACLE to Convert Dependent FOREIGN KEY Values to NULL.
- ON DELETE Clause Cannot be Specified for a View Constraint.
- Declared on FOREIGN KEY Column only.

Few Things to Note Before We Apply Relations:

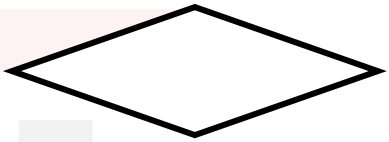
Relation Model Symbols:

- Entity OR Table →

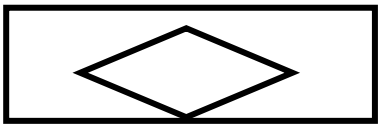
- Column OR Attribute →



- Relation OR association →



- associative Entity OR Table →



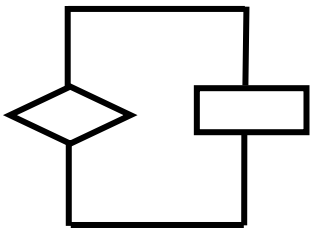
- Connecting →

Types of Relations:

i. Unary Relation.
ii. Binary Relation.
iii. Ternary Relation.
iv. N'Ary Relation.

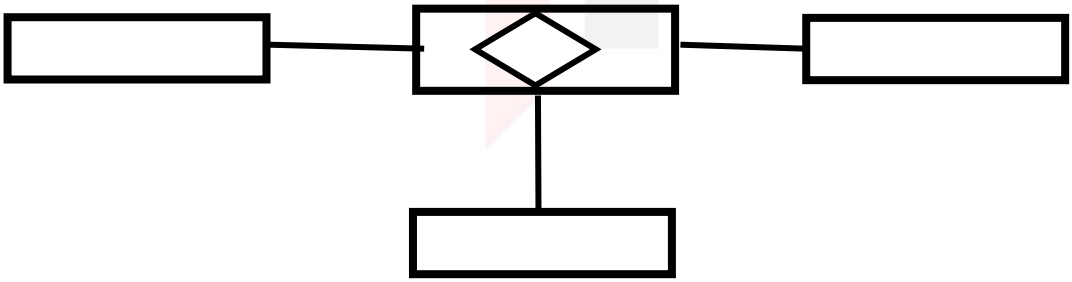
Relation Model Representation:

i. Unary Relation:



ii. Binary Relation:

iii. Ternary Relation:



SRelation Cardinality:

- one - to - one ----> 1..1
- one - to - Many ---- > 1..* OR 1:M
- Many - to - Many ---- > *..* OR M:N OR M:N

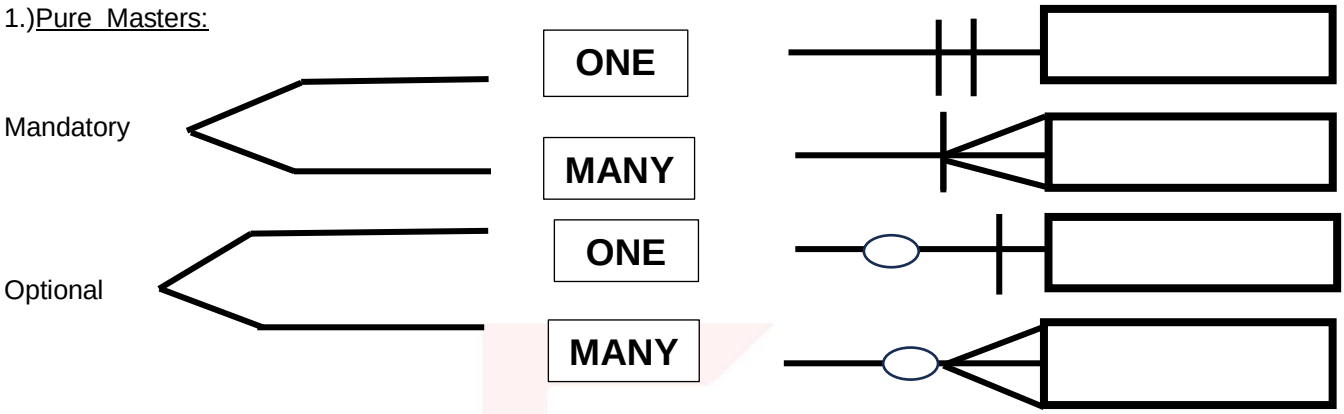
Relation State:

- Mandatory State
- Optional State

Relation State with Cardinality:

Types of Tables to Be Identified in Design Process:

1.)Pure Masters:



- these Are Tables Which Contain only Primary Keys, and All the Remaining Columns are Non Keys.

2.) Master Details:

- these Are Tables Containing their Own Primary Key and are Also Related to them Selves or Other Tables with foreign keys.

3.) Pure Details:

- these are Tables Which Contain only foreign Keys, Related to Other Table or Tables Primary Key.

Steps Followed for Creating foreign Key Constraint are as Follows:

Step 1: Create Primary Master's /Pure Master's

```
SQL> CREATE  TABLE  Dept
  ( Deptno  NUMBER(2)  CONSTRAINT  Deptno_Pk  PRIMARY  KEY,
    Dname  VARCHAR2(20)  CONSTRAINT  Dname_NN  NOT  NULL,
    Location  VARCHAR2(20)  CONSTRAINT  Loc_NN  NOT  NULL
   );
```

Step 2: Create Detailed/ Child Table

- these Are Tables Which Can Contain Primary Key of their Own as Well as foreign Key's Referring to Other Primary Master's or to them Selves.
- these Tables Are Also Called as Dependent Tables or Referential Tables.

```
SQL> CREATE TABLE Employee(
EmployeeID NUMBER(6) CONSTRAINT Emp_ID_PK PRIMARY KEY,
Ename VARCHAR2(30) CONSTRAINT Ename_NN NOT NULL,
 Designation VARCHAR2(30) CONSTRAINT Ename_NN NOT NULL,
ManagerID NUMBER(6) CONSTRAINT Mgr_ID_FK_Self REFERENCES
Employee(EmployeeID) ON DELETE SET NULL,
Hiredate DATE CONSTRAINT Hiredate_NN NOT NULL,
Commission NUMBER(7,2), DeptID NUMBER(2) CONSTRAINT Dept_ID_FK
REFERENCES Dept(Deptno) ON DELETE CASCADE
);
```

Working with Composite Keys:

Step 1: Create Pure Masters

```
SQL> CREATE TABLE SampleMaster1
( SampleID1 NUMBER(4) CONSTRAINT Samp_ID1_PK PRIMARY KEY,
  SampName1 VARCHAR2(20) CONSTRAINT SampName1_NN NOT NULL,
 SampDate1 DATE CONSTRAINT SampDate1_NN NOT NULL
);
```

```
SQL> CREATE TABLE SampleMaster2
( SampleID2 NUMBER(4) CONSTRAINT Samp_ID2_PK PRIMARY KEY,
  SampName2 VARCHAR2(20) CONSTRAINT SampName2_NN NOT NULL,
 SampDate2 DATE CONSTRAINT SampDate2_NN NOT NULL
);
```

Step 2: Create the Pure Details

```
SQL> CREATE TABLE SampRef

(SampIDRef1 NUMBER(4) CONSTRAINT SampIDRef1_FK REFERENCES
SampMaster1(SampID1),

SampIDRef2 NUMBER(4) CONSTRAINT SampIDRef2_FK REFERENCES
SampMaster2(SampID2),
```

```
SampNameRef    VARCHAR2(20),
SampDateRef   DATE,
CONSTRAINT   SampRef_Comp_PK   PRIMARY   KEY(SampIDRef1, SampIDRef2)
);
```

CHECK Constraint:

- It Defines a Condition That Each Row Must Satisfy.
- to Satisfy the Constraint, Each Row in the Table Must Make the Condition Either TRUE or UNKNOWN.
- ORACLE Does Not Verify That CHECK CONDITIONS Are Mutually Exclusive.

Restrictions:

- the Condition of a CHECK Constraint Can Refer to any Column in the Same Table, But It Cannot Refer to Columns of Other Tables.
- the Constructs That Cannot Be Included are…
    - Queries to Refer to Values in Other Rows.
    - Calls to Functions SYSDATE, UID, USER, USERENV.
    - the Pseudo Columns CURRVAL, NEXTVAL, LEVEL or ROWNUM.
    - DATE Constant That Are Not Fully Specified.
- A Single Column Can Have Multiple CHECK Constraints That Can Be Defined on a Column.
- the CHECK Constraints Can Be Defined At the Column Level or Table Level.

Default Option:

- the DEFAULT Option is Given to Maintain a DEFAULT Value in a Column.
- the Option Prevents NULL Values from Entering the Columns, if a Row is Inserted without a Value for a Column.
- the Default Value Can Be a Literal, an Expression or a SQL function.
- the DEFAULT Value Can be a Literal, an Expression or a SQL Function.
- the DEFAULT Expression Must Match the Data Type of the COLUMN.

Example:

```
SQL> CREATE TABLE Dept (Deptno NUMBER(2) CONSTRAINT CHK_Deptno
CHECK(Deptno BETWEEN 10 AND 90),
Dname VARCHAR2(15) CONSTRAINT Chk_Dname_Up CHECK(Dname=UPPER(Dname)),
Loc VARCHAR2(15) CONSTRAINT Chk_Loc CHECK(Loc IN('DALLAS','BOSTON','NEW
YORK','CHICAGO'))
);
```

Constraints Maintenance:

Adding Constraints to a Table:

- A Constraint Can Be Added to a Table At any Time After the Table Was Created by Using ALTER TABLE Statement, Using ADD Clause.

```
SQL> ALTER TABLE <Table Name> ADD [CONSTRAINT <ConstraintName>]
          CONS_TYPE(Column_Name);
```

- the Constraint Name in the Syntax is Optional, But Recommended.

Guidelines:

- We Can ADD, DROP, ENABLE, or DISABLE a Constraint, but Cannot Modify the Physical Structure of the Table.
- A NOT NULL Can Be Added to Existing Column by Using the MODIFY Clause of the ALTER TABLE Statement.
- NOT NULL Can Be Defined only When the Table Contains No Rows.

Example:

```
SQL>ALTER TABLE Emp ADD CONSTRAINT Emp_Mgr_FK FOREIGN KEY(Mgr)
REFERENCES Emp(Empno);
```

DROPPING Constraints:

- to Drop a Constraint, Identify the Constraint Name from the
  - USER_CONSTRAINTS
  - USER_CONS_COLUMNS Data Dictionary Views.
- the ALTER TABLE Statement is Used with the DROP Clause.
- the CASCADE Option of Th DROP Clause Causes any Dependent Constraints Also to Be Dropped.

- When a Constraint is Dropped, the Constraint is No Longer Enforced and is No Longer Available in the Data Dictionary.

Syntax:

```
SQL> ALTER TABLE Dept DROP PRIMARY KEY CASCADE;
SQL> ALTER TABLE Emp DROP CONSTRAINT Emp_Mgr_FK;
```

ENABLING Constraints:

- the Constraint Can Be Enabled without Dropping it or Re-Creating it.
- the ALTER TABLE Statement with the ENABLE Clause is Used for the Purpose.

Syntax:

```
SQL> ALTER TABLE <Table Name> ENABLE CONSTRAINT <Constraint Name>;
```

Guidelines:

- Enabling a Constraint Applies to All the Data in the Table At a Time.
- When a UNIQUE or PRIMARY KEY Constraint is ENABLED , the UNIQUE or PRIMARY KEY Index is Automatically Created.
- the ENABLE Clause Can Be Used Both in CREATE TABLE as Well as ALTER TABLE Statements.

Examples:

```
SQL> ALTER TABLE Emp ENABLE CONSTRAINT Emp_Empno_FK;
```

VIEWING Constraints:

- to View All Constraints on a Table Query Upon the Data Dictionary USER_CONSTRAINTS.
- the Codes That Are Revealed Are…
  - **C** --- > CHECK
  - **P** --- > PRIMARY KEY
  - **R** --- > REFERENTIAL INTEGRITY
  - **U** --- > UNIQUE KEY

Example:

```
SQL> SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE, SEARCH_CONDITION FROM
USER_CONSTRAINTS WHERE TABLE_NAME ='EMP';
```

VIEWING the Columns associated with Constraints:

- the Names of the Columns That Are Involved in Constraints Can Be Known by Querying the USER_CONS_COLUMNS Data Dictionary View.

Example:

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