Constrains

In this lesson we are going to learn the importance of using table constraints to assist us in maintaining the integrity of the data.

We have to understand that if the data is not accurate it become corrupted and thereby unusable.

If we allow ‘bad’ data into our tables we will find that it will proliferate or expand throughout the database.

So one of our main jobs to maintain the integrity of our data at all costs. We therefore use constraints.

What are constraints?

Constraints enforce the rules that we set up at the table level.

Constraints also prevent the deletion of a table if there are dependencies.

There are five (5) valid constraints:

NOT NULL

UNIQUE

PRIMARY KEY

FOREIGN KEY

CHECK

Constraint Descriptions

NOT NULL – Specifies that this column may not contain a null value.

UNIQUE – Specifies a column or combination of columns whose values must be unique for all rows in the table.

PRIMARY KEY – Uniquely identifies each row of the table.

FOREIGN KEY – Established and enforces a foreign key relationship between the column and a column of the referenced table.

CHECK – Specifies a condition that MUST be true.

Constraint Guidelines

We will usually name a constrain when we create it or the system will create a name for us. When the system names it we cannot be certain as to which constraint name goes with the constraint we set up.

We can create a constraint at the same time as the table is created or we can create them after the table is created.

Constraints can be defined at either the column level or the table level.

Defining Constraint

If we want to create a PRIMARY KEY constrain on the table EMP at the column level of the column EMPNO we would define it as follows:

```
CREATE TABLE emp
        (empno NUMBER(4)  CONSTRAINT
                emp_empno_pk PRIMARY KEY,
         ename VARCHAR2(10),
         deptno NUMBER(2) NOT NULL
);
```

We have created and named a primary key and ALSO created a NOT NULL constraint on empno.

Defining Constraint

Constraints are usually created as the same time as the table. However constraints can be added to a table after its creation and also temporarily disabled. With there being two levels of constraints we need to understand the difference:

COLUMN – References a single column and is defined within a specification for the owning column: Can define any type of integrity constraint.

TABLE – References one or more columns and is defined separately from the definitions of the column in the table: Can define any constraint except NOT NULL.

Naming Constraint

As a normal practice we have a certain way to name the constraints so the system does not do it for us. The name will consist of the table name, column name and the constraint abbreviation.

PK	The abbreviation for PRIMARY KEY
FK	The abbreviation for FOREIGN KEY
UK	The abbreviation for UNIQUE KEY
CK	The abbreviation for Check Constraint
NN	The abbreviation for NOT NULL.

Sample of Naming Constraints (table level)

Using PRIMARY KEY

```
CREATE TABLE dept
        (emp    NUMBER(4),
        ename VARCHAR2(10) NOT NULL,
        deptno NUMBER(7) NOT NULL,
CONSTRAINT dept_emp_pk PRIMARY KEY (emp));
```

The table level constraint is listed after the definitions of all columns. Since it is done this way we have to include the column name in the constraint description.

Sample of Naming Constraints (table level)

Using PRIMARY KEY and UNIQUE

```
CREATE TABLE dept
        (deptno         NUMBER(4),
        deptname        VARCHAR2(25),
        loc             Varchar2(13),
CONSTRAINT dept_deptno_pk PRIMARY KEY (deptno),
CONSTRAINT dept_deptname UNIQUE(deptname));
```

The table level constraint is listed after the definitions of all columns. Since it is done this way we have to include the column name in the constraint description.

Sample of Naming Constraints (Column level)

Using PRIMARY KEY

```
CREATE TABLE dept
 (emp NUMBER(4) CONSTRAINT dept_emp_pk PRIMARY KEY
  ename VARCHAR2(10) NOT NULL,
  deptno NUMBER(7) NOT NULL );
```

Since this is at the column lever we do not need to list the column name as in the previous slide.

Sample of Naming Constraints

Using NOT NULL and a FOREIGN KEY

```
CREATE TABLE emp
        (emp            NUMBER(4),
        ename           VARCHAR2(10) NOT NULL,
        deptno          NUMBER(7) NOT NULL
        CONSTRAINT emp_deptno_fk FOREIGN KEY
        (deptno) REFERENCES dept(deptno));
```

The NN constraint will show up in the logs as a system name that will end with _nn.

Sample of Naming Constraints

Using CHECK constraint

```
CREATE TABLE dept
(emp         NUMBER(4),
ename        VARCHAR2(10) NOT NULL,
deptno       NUMBER(7) NOT NULL
CONSTRAINT dept_deptno_ck CHECK
(deptno BETWEEN 10 and 99));
```

Sample of Naming Constraints

Using UNIQUE constraint

```
CREATE TABLE dept
(emp          NUMBER(4),
dname         VARCHAR2(10) NOT NULL,
deptno        NUMBER(7) NOT NULL
CONSTRAINT dept_dname_uk UNIQUE (dname));
```

NOT NULL Constraint

The NOT NULL constraint ensured that null value are NOT PERMITTED in that column.

If we try to add to a table column that has a not null constraint we have to have a value in that insert statement or the statement will fail.

Not null constraint also are automatically attached to some of the other constraints such as: FOREIGN KEY and PRIMARY KEY.

The not null constraint can be specified ONLY at the column level, NOT at the table level.

UNIQUE Constraint

A unique key integrity constraint requires that every value in a column or set of columns (Composite unique key) meaning that there are NO two rows of a table have duplicate values.

The unique key constraint DOES allow the input of NULL values unless we also define the NOT NULL constraint.

Defining constraints

We can also create constraint after the table is created. We do have to do this a little differently since we do not want to create the table again and lose the data that it contains.

To accomplish this we use the ALTER TABLE command.

ALTER TABLE emp

ADD CONSTRAINT constraint name then the specification.

ALTER TABLE emp

ADD CONSTRAINT emp_mgr_fk

FOREIGN KEY(mgr) REFERENCES emp (empno);