Lab – 08

**Constraints**

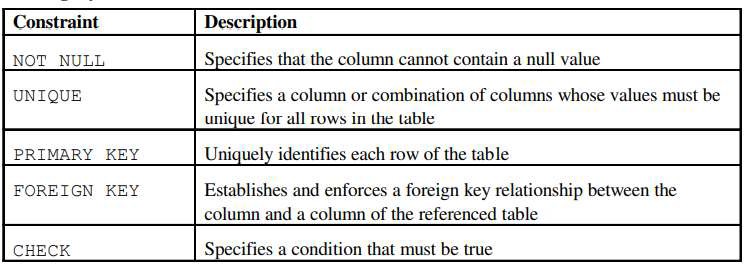
# Objective:

* Students will be able to apply multiple constraint on database designs and implementation of Relational model.

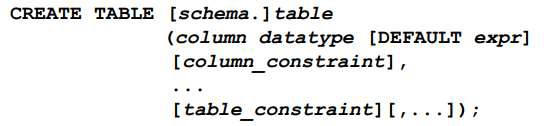
# Constraints:

* Constraints enforce rules at the table level.
* Constraints prevent the deletion of a table if there are dependencies.

# Data Integrity Constraints



**Defining Constraints:**

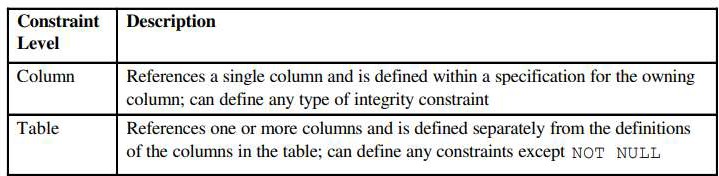


In the syntax:

* **schema** is the same as the owner’s name
* **table** is the name of the table DEFAULT expr specifies a
* **default value** to use if a value is omitted in the INSERT statement
* **column** is the name of the column
* **datatype** is the column’s data type and length
* **column\_constraint** is an integrity constraint as part of the column definition
* **table\_constraint** is an integrity constraint as part of the table definition

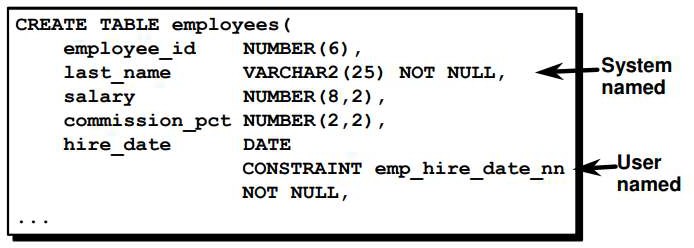
Constraints are usually created at the same time as the table. Constraints can be added to a table after its creation and also temporarily disabled.

# Constraints can be defined at one of two levels:



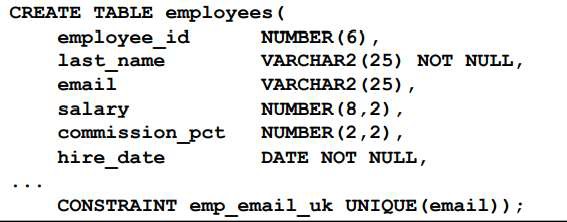
**The NOT NULL Constraint**

* The NOT NULL constraint ensures that the column contains no null values. Columns without the NOT NULL constraint can contain null values by default.
* The NOT NULL constraint can be specified only at the column level, not at the table level. The example applies the NOT NULL constraint to the LAST\_NAME and HIRE\_DATE columns of the EMPLOYEES table. Because these constraints are unnamed, the Oracle Server creates names for them.



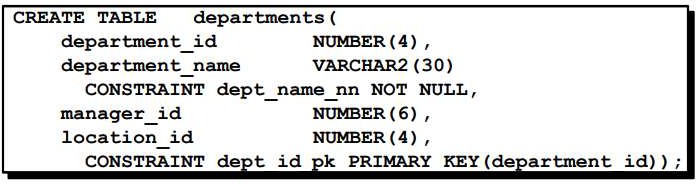
# The UNIQUE Constraint

* A UNIQUE key integrity constraint requires that every value in a column or set of columns (key) be unique: that is, no two rows of a table can have duplicate values in a specified column or set of columns. The column (or set of columns) included in the definition of the UNIQUE key constraint is called the unique key. If the UNIQUE constraint comprises more than one column, that group of columns is called a composite unique key.
* UNIQUE constraints allow the input of nulls unless you also define NOT NULL constraints for the same columns. In fact, any number of rows can include nulls for columns without NOT NULL constraints because nulls are not considered equal to anything. A null in a column (or in all columns of a composite UNIQUE key) always satisfies a UNIQUE constraint.
* UNIQUE constraints can be defined at the column or table level. A composite unique key is created by using the table level definition.
* The example applies the UNIQUE constraint to the EMAIL column of the EMPLOYEES table. The name of the constraint is EMP\_EMAIL\_UK.



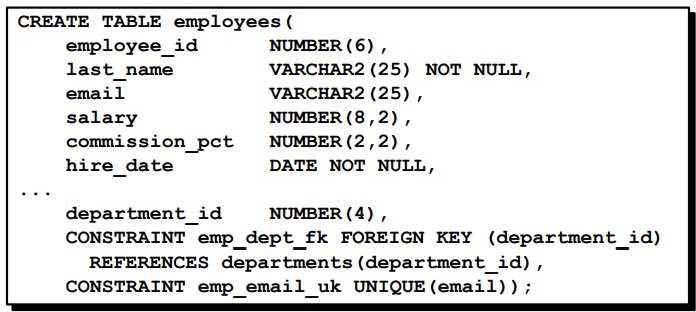
# The PRIMARY KEY Constraint

* A PRIMARY KEY constraint creates a primary key for the table. Only one primary key can be created for a each table. The PRIMARY KEY constraint is a column or set of columns that uniquely identifies each row in a table. This constraint enforces uniqueness of the column or column combination and ensures that no column that is part of the primary key can contain a null value.
* PRIMARY KEY constraints can be defined at the column level or table level. A composite PRIMARY KEY is created by using the table-level definition.
* A table can have only one PRIMARY KEY constraint but can have several UNIQUE constraints.
* The example on the slide defines a PRIMARY KEY constraint on the DEPARTMENT\_ID column of the DEPARTMENTS table. The name of the constraint is DEPT\_ID\_PK.



# The FOREIGN KEY Constraint

* The FOREIGN KEY, or referential integrity constraint, designates a column or combination of columns as a foreign key and establishes a relationship between a primary key or a unique key in the same table or a different table. In the example on the slide, DEPARTMENT\_ID has been defined as the foreign key in the EMPLOYEES table (dependent or child table); it references the DEPARTMENT\_ID column of the DEPARTMENTS table (the referenced or parent table).
* A foreign key value must match an existing value in the parent table or be NULL.
* Foreign keys are based on data values and are purely logical, not physical, pointers.
* FOREIGN KEY constraints can be defined at the column or table constraint level. A composite foreign key must be created by using the table-level definition.
* The example below defines a FOREIGN KEY constraint on the DEPARTMENT\_ID column of the EMPLOYEES table, using table-level syntax. The name of the constraint is EMP\_DEPTID\_FK.



# FOREIGN KEY Constraint Keywords

The foreign key is defined in the child table, and the table containing the referenced column is the parent table. The foreign key is defined using a combination of the following keywords:

* **FOREIGN KEY** is used to define the column in the child table at the table constraint level.
* **REFERENCES** identifies the table and column in the parent table.
* **ON DELETE CASCADE** indicates that when the row in the parent table is deleted, the dependent rows in the child table will also be deleted.
* **ON DELETE SET NULL** converts foreign key values to null when the parent value is removed.

The default behavior is called the restrict rule, which disallows the update or deletion of referenced data. Without the ON DELETE CASCADE or the ON DELETE SET NULL options, the row in the parent table cannot be deleted if it is referenced in the child table.

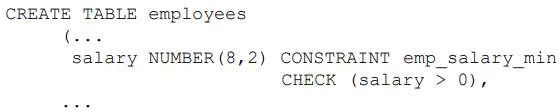
# The CHECK Constraint

The CHECK constraint defines a condition that each row must satisfy. The condition can use the same constructs as query conditions, with the following exceptions:

* References to the CURRVAL, NEXTVAL, LEVEL, and ROWNUM pseudo-columns
* Calls to SYSDATE, UID, USER, and USERENV functions
* Queries that refer to other values in other rows

A single column can have multiple CHECK constraints which reference the column in its definition. There is no limit to the number of CHECK constraints which you can define on a column.

CHECK constraints can be defined at the column level or table level.



**Adding a Constraint Syntax:**

Use the ALTER TABLE statement to:

* Add or drop a constraint, but not modify its structure
* Enable or disable constraints
* Add a NOT NULL constraint by using the MODIFY clause

# Syntax:



In the syntax:

* Table is the name of the table
* Constraint is the name of the constraint
* Type is the constraint type
* Column is the name of the column affected by the constraint

The constraint name syntax is optional, although recommended. If you do not name your constraints, the system will generate constraint names.

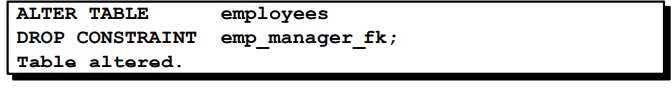
# Adding a Constraint:

* Add a FOREIGN KEY constraint to the EMPLOYEES table to indicate that a manager must already exist as a valid employee in the EMPLOYEES table.

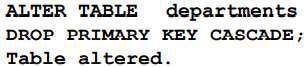


# Dropping a Constraint:

* Remove the manager constraint from the EMPLOYEES table.

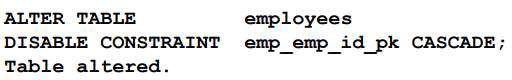


* Remove the PRIMARY KEY constraint on the DEPARTMENTS table and drop the associated FOREIGN KEY constraint on the EMPLOYEES.DEPARTMENT\_ID column.



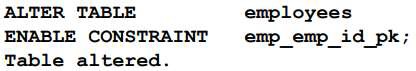
# Disabling Constraints:

* Execute the DISABLE clause of the ALTER TABLE statement to deactivate an integrity constraint.
* Apply the CASCADE option to disable dependent integrity constraints.



# Enabling Constraints:

* Activate an integrity constraint currently disabled in the table definition by using the ENABLE clause.



* A UNIQUE or PRIMARY KEY index is automatically created if you enable a UNIQUE key or PRIMARY KEY constraint.

# Cascading Constraints:

* The CASCADE CONSTRAINTS clause is used along with the DROP COLUMN clause.
* The CASCADE CONSTRAINTS clause drops all referential integrity constraints that refer to the primary and unique keys defined on the dropped columns.
* The CASCADE CONSTRAINTS clause also drops all multicolumn constraints defined on the dropped columns.

This statement illustrates the use of the CASCADE CONSTRAINTS clause. Assume table TEST1 is created as follows:

CREATE TABLE test1 (

pk NUMBER PRIMARY KEY, fk NUMBER,

col1 NUMBER, col2 NUMBER,

CONSTRAINT fk\_constraint FOREIGN KEY (fk) REFERENCES test1, CONSTRAINT ck1 CHECK (pk > 0 and col1 > 0),

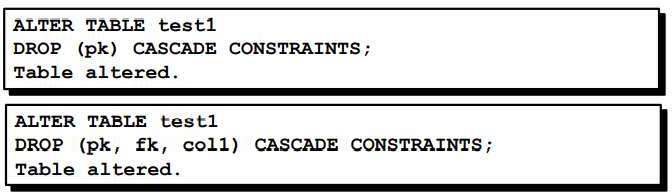
CONSTRAINT ck2 CHECK (col2 > 0));

An error is returned for the following statements:

ALTER TABLE test1 DROP (pk); (pk is a parent key)

ALTER TABLE test1 DROP (col1); (col1 is referenced by multicolumn constraint ck1)

# Example:



Submitting the following statement drops column PK, the primary key constraint, the fk\_constraint foreign key constraint, and the check constraint, CK1:

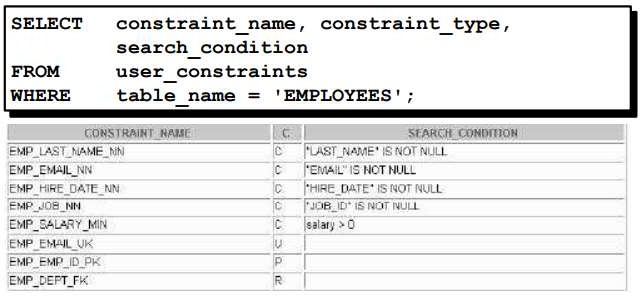
ALTER TABLE test1 DROP (pk) CASCADE CONSTRAINTS;

If all columns referenced by the constraints defined on the dropped columns are also dropped, then CASCADE CONSTRAINTS is not required. For example, assuming that no other referential constraints from other tables refer to column PK, it is valid to submit the following statement without the CASCADE CONSTRAINTS clause:

ALTER TABLE test1 DROP (pk, fk, col1);

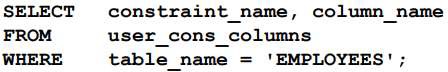
# Viewing Constraints

After creating a table, you can confirm its existence by issuing a DESCRIBE command. The only constraint that you can verify is the NOT NULL constraint. To view all constraints on your table, query the USER\_CONSTRAINTS table. The example given below displays the constraints on the EMPLOYEES table.



# Viewing the Columns Associated with Constraints:

* View the columns associated with the constraint names in the USER\_CONS\_COLUMNS view.



**Lab Tasks:**

* 1. Create a Table Student, it should have the following fields, choose appropriate DATA TYPES.
     + Std\_id
     + Std\_name
     + Std\_program
     + Std\_current\_semester Add a new column , cgpa.

CREATE TABLE Student(

Std\_id int PRIMARY KEY,

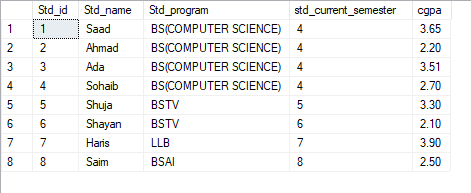
Std\_name varchar(255),

Std\_program varchar(255),

std\_current\_semester int,

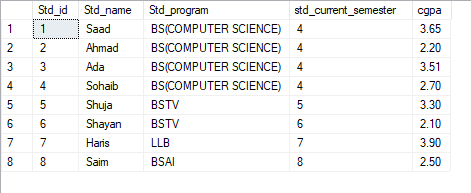
cgpa decimal );

* 1. Insert at least 8 records, (at least 4 records should have value of Std\_program as ‘bscs’)



* 1. Update bscs to BS(COMPUTER SCIENCE) in Student Table.

UPDATE Student SET Std\_program= 'BS(COMPUTER SCIENCE)' WHERE Std\_program = 'BSCS';



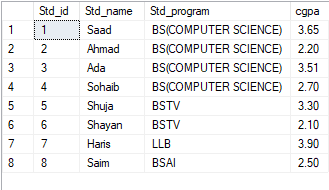
* 1. Rename table to student\_bahria

exec sp\_rename 'Student', 'Student\_Bahria';

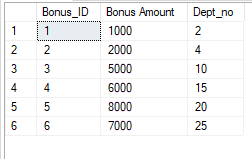


* 1. Delete Std\_current\_semester column from table.

ALTER TABLE Student\_Bahria DROP COLUMN std\_current\_semester;

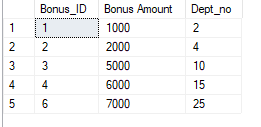


* 1. Create a table bonus (bonus Id, bonus amount, department no), insert information of multiple departments with department numbers 2, 4, 10, 15, 20 and 25.



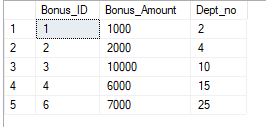
* 1. Delete row when department no is equal to 20,

DELETE FROM Bonus WHERE Dept\_no=20;



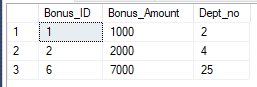
* 1. Increment bonus when department no is equal to 10 ,

UPDATE Bonus SET Bonus\_Amount+=Bonus\_Amount WHERE Dept\_no = 10;



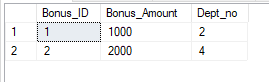
* 1. Delete row where department is between 10 and 20.

DELETE FROM Bonus WHERE Dept\_no BETWEEN 10 AND 20;



* 1. Delete rows where department number is greater than 20.

DELETE FROM Bonus WHERE Dept\_no > 20;



* 1. Add a table-level PRIMARY KEY constraint to the EMP table on the ID column. The constraint should be named at creation. Name the constraint my\_emp\_id\_pk. Hint: The constraint is enabled as soon as the ALTER TABLE command executes successfully.

CREATE TABLE EMP(

my\_emp\_id\_pk INT PRIMARY KEY,

emp\_name varchar(255) NOT NULL,

emp\_age int,

CHECK (emp\_age>=18)

);

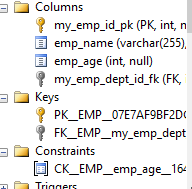
* 1. Create a PRIMARY KEY constraint to the DEPT table using the ID column. The constraint should be named at creation. Name the constraint my\_deptid\_pk. Hint: The constraint is enabled as soon as the ALTER TABLE command executes successfully.

ALTER TABLE DEPT ADD PRIMARY KEY(my\_deptid\_pk);

* 1. Add a column DEPT\_ID to the EMP table. Add a foreign key reference on the EMP table that ensures that the employee is not assigned to a nonexistent department. Name the constraint my\_emp\_dept\_id\_fk.

ALTER TABLE EMP ADD FOREIGN KEY(my\_emp\_dept\_id\_fk) REFERENCES DEPT(my\_deptid\_pk);

* 1. Confirm that the constraints were added by querying the USER\_CONSTRAINTS view. Note the types and names of the constraints. Save your statement text in a file called lab10\_4.sql.



* 1. Display the object names and types from the USER\_OBJECTS data dictionary view for the EMP and DEPT tables. Notice that the new tables and a new index were created.
  2. Modify the EMP table. Add a COMMISSION column of NUMBER data type, precision 2, scale 2. Add a constraint to the commission column that ensures that a commission value is greater than zero

ALTER TABLE EMP ADD COMMISSION numeric(2,2) CHECK(COMMISSION>0);