What Are Constraints?

- Constraints enforce rules at the table level.
- Constraints prevent the deletion of a table if there are dependencies.
- The following constraint types are valid:
  - NOT NULL
  - UNIQUE
  - PRIMARY KEY
  - FOREIGN KEY
  - CHECK

Constraint	Description
NOT NULL	Specifies that the column cannot 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	Establishes 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

- Name a constraint or the Oracle server generates a name by using the SYS_C*n* format.
- Create a constraint either:
  - At the same time as the table is created, or
  - After the table has been created.
- Define a constraint at the column or table level.
- View a constraint in the data dictionary.

You can view the constraints defined for a specific table by looking at the USER_CONSTRAINTS data dictionary table.

Defining Constraints

```
CREATE TABLE [schema.]table
             (column datatype [DEFAULT expr]
             [column_constraint],
             ...
             [table_constraint][,...]);
```

```
CREATE TABLE employees(
             employee_id  NUMBER(6),
             first_name   VARCHAR2(20),
             ...
             job_id       VARCHAR2(10) NOT NULL,
             CONSTRAINT emp_emp_id_pk
                          PRIMARY KEY (EMPLOYEE_ID));
```

- **Column constraint level:**

```
column [CONSTRAINT constraint_name] constraint_type,
```

- **Table constraint level:**

```
column,...
  [CONSTRAINT constraint_name] constraint_type
  (column, ...),
```

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.

Constraint Level	Description
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 columns in the table; can define any constraints except `NOT NULL`

In the syntax:

constraint_name is the name of the constraint

constraint type is the type of the constraint

The NOT NULL Constraint

Ensures that null values are not permitted for the column

100	King	SKING	515.123.4567	17-JUN-87	AD_PRES	24000		
101	Kochhar	NKOCHHAR	515.123.4568	21-SEP-89	AD_VP	17000		100
102	De Haan	LDEHAAN	515.123.4569	13-JAN-93	AD_VP	17000		100
103	Hunold	AHUNOLD	590.423.4567	03-JAN-90	IT_PROG	9000		102
104	Ernst	BERNST	590.423.4568	21-MAY-91	IT_PROG	6000		103
205	[illegible]	[illegible]	[illegible]	JUN-94	[illegible]	[illegible]		101
206	Gietz	WGIETZ	515.123.8181	07-JUN-94	AC_ACCOUNT	8300		205

20 rows selected.

NOT NULL constraint (No row can contain a null value for this column.)

NOT NULL constraint

Absence of NOT NULL constraint (Any row can contain null for this column.)

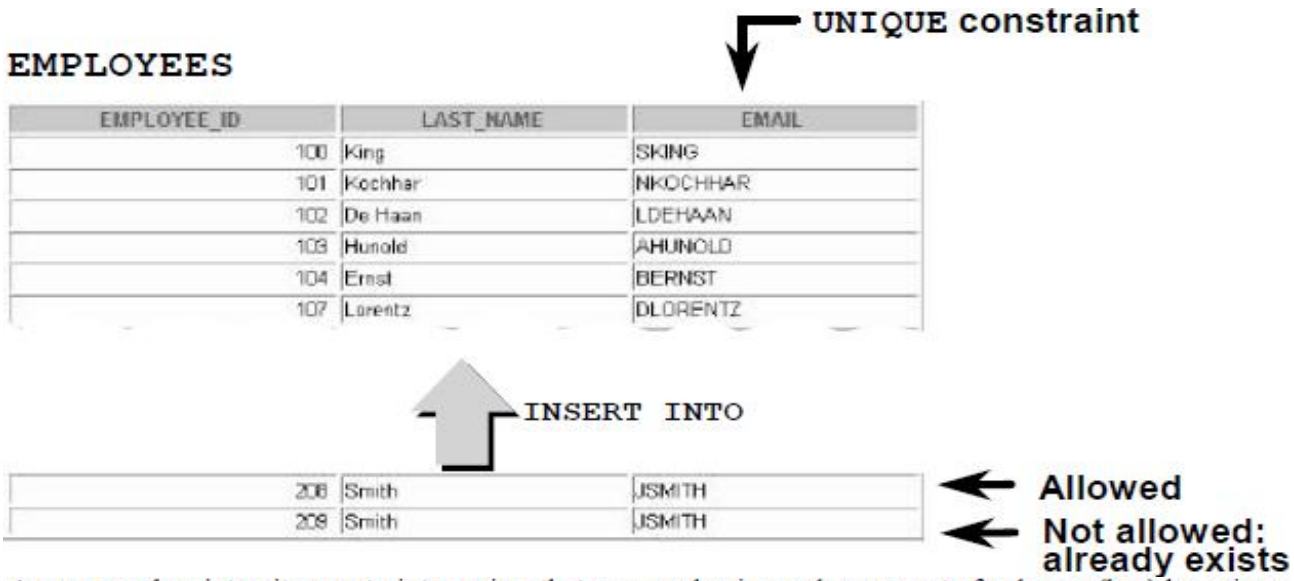
Is defined at the column level

```
CREATE TABLE employees(
    employee_id     NUMBER(6),
    last_name       VARCHAR2(25) NOT NULL,
    salary          NUMBER(8,2),
    commission_pct  NUMBER(2,2),
    hire_date       DATE
                    CONSTRAINT emp_hire_date_nn
                    NOT NULL,
...
```

System named

User named

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.

Note: Because of the search mechanism for UNIQUE constraints on more than one column, you cannot have identical values in the non-null columns of a partially null composite UNIQUE key constraint.

Is defined at either the table level or the column level

```
CREATE TABLE employees(
    employee_id      NUMBER(6),
    last_name        VARCHAR2(25) NOT NULL,
    email            VARCHAR2(25),
    salary           NUMBER(8,2),
    commission_pct   NUMBER(2,2),
    hire_date        DATE NOT NULL,
...
    CONSTRAINT emp_email_uk UNIQUE(email));
```

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 on the slide applies the UNIQUE constraint to the EMAIL column of the EMPLOYEES table. The name of the constraint is EMP_EMAIL_UK.

Note: The Oracle Server enforces the UNIQUE constraint by implicitly creating a unique index on the unique key column or columns.

The PRIMARY KEY Constraint

DEPARTMENTS

PRIMARY KEY

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
50	Shipping	124	1500
60	IT	103	1400
80	Sales	149	2500

Not allowed (null value)

INSERT INTO

	Public Accounting		1400
50	Finance	124	1500

Not allowed (50 already exists)

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.

Is defined at either the table level or the column level

```
CREATE TABLE   departments(
    department_id         NUMBER(4),
    department_name       VARCHAR2(30)
      CONSTRAINT dept_name_nn NOT NULL,
    manager_id            NUMBER(6),
    location_id           NUMBER(4),
      CONSTRAINT dept id pk PRIMARY KEY(department id));
```

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.

Note: A UNIQUE index is created automatically for a PRIMARY KEY column.

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.

Is defined at either the table level or the column level

```
CREATE TABLE employees(
    employee_id      NUMBER(6),
    last_name        VARCHAR2(25) NOT NULL,
    email            VARCHAR2(25),
    salary           NUMBER(8,2),
    commission_pct   NUMBER(2,2),
    hire_date        DATE NOT NULL,
...
    department_id    NUMBER(4),
    CONSTRAINT emp_dept_fk FOREIGN KEY (department_id)
      REFERENCES departments(department_id),
    CONSTRAINT emp_email_uk UNIQUE(email));
```

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 on the slide 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.

The foreign key can also be defined at the column level, provided the constraint is based on a single column. The syntax differs in that the keywords FOREIGN KEY do not appear. For example:

```
CREATE TABLE employees
(...
department_id NUMBER(4) CONSTRAINT emp_deptid_fk
    REFERENCES departments(department_id),
...
)
```

FOREIGN KEY Constraint Keywords

- **FOREIGN KEY**: Defines the column in the child table at the table constraint level
- **REFERENCES**: Identifies the table and column in the parent table
- **ON DELETE CASCADE**: Deletes the dependent rows in the child table when a row in the parent table is deleted
- **ON DELETE SET NULL**: Converts dependent foreign key values to null

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

- Defines a condition that each row must satisfy
- The following expressions are not allowed:
  - References to CURRVAL, NEXTVAL, LEVEL, and ROWNUM pseudocolumns
  - Calls to SYSDATE, UID, USER, and USERENV functions
  - Queries that refer to other values in other rows

```
..., salary  NUMBER(2)
     CONSTRAINT emp_salary_min
                CHECK (salary > 0),...
```

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.

```
CREATE TABLE employees
      (...
       salary NUMBER(8,2) CONSTRAINT emp_salary_min
                          CHECK (salary > 0),
      ...
```

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

```
ALTER TABLE table
ADD [CONSTRAINT constraint] type (column);
```

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

Guidelines

- You can add, drop, enable, or disable a constraint, but you cannot modify its structure.
- You can add a NOT NULL constraint to an existing column by using the MODIFY clause of the ALTER TABLE statement.

Note: You can define a NOT NULL column only if the table is empty or if the column has a value for every row.

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.

```
ALTER TABLE      employees
ADD CONSTRAINT   emp_manager_fk
  FOREIGN KEY(manager_id)
  REFERENCES employees(employee_id);
Table altered.
```

Dropping a Constraint

- Remove the manager constraint from the EMPLOYEES table.

```
ALTER TABLE      employees
DROP CONSTRAINT  emp_manager_fk;
Table altered.
```

- Remove the PRIMARY KEY constraint on the DEPARTMENTS table and drop the associated FOREIGN KEY constraint on the EMPLOYEES.DEPARTMENT_ID column.

```
ALTER TABLE  departments
DROP PRIMARY KEY CASCADE;
Table altered.
```

To drop a constraint, you can identify the constraint name from the USER_CONSTRAINTS and USER_CONS_COLUMNS data dictionary views. Then use the ALTER TABLE statement with the DROP clause. The CASCADE option of the DROP clause causes any dependent constraints also to be dropped.

Syntax

```
ALTER TABLE  table
DROP  PRIMARY KEY | UNIQUE (column) |
      CONSTRAINT   constraint  [CASCADE];
```

In the syntax:

table	is the name of the table
column	is the name of the column affected by the constraint
constraint	is the name of the constraint

When you drop an integrity constraint, that constraint is no longer enforced by the Oracle Server and is no longer available in the data dictionary.

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.**

```
ALTER TABLE          employees
DISABLE CONSTRAINT   emp_emp_id_pk CASCADE;
Table altered.
```

Guidelines

- You can use the DISABLE clause in both the CREATE TABLE statement and the ALTER TABLE statement.
- The CASCADE clause disables dependent integrity constraints.
- Disabling a unique or primary key constraint removes the unique index.

Enabling Constraints

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

```
ALTER TABLE          employees
ENABLE CONSTRAINT    emp_emp_id_pk;
Table altered.
```

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

Guidelines

- If you enable a constraint, that constraint applies to all the data in the table. All the data in the table must fit the constraint.
- If you enable a UNIQUE key or PRIMARY KEY constraint, a UNIQUE or PRIMARY KEY index is created automatically.
- You can use the ENABLE clause in both the CREATE TABLE statement and the ALTER TABLE statement.
- Enabling a primary key constraint that was disabled with the CASCADE option does not enable any foreign keys that are dependent upon the primary key.

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

```
ALTER TABLE test1
DROP (pk) CASCADE CONSTRAINTS;
Table altered.
```

```
ALTER TABLE test1
DROP (pk, fk, col1) CASCADE CONSTRAINTS;
Table altered.
```

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

Query the USER_CONSTRAINTS table to view all constraint definitions and names.

```
SELECT    constraint_name, constraint_type,
          search_condition
FROM      user_constraints
WHERE     table_name = 'EMPLOYEES';
```

CONSTRAINT_NAME	C	SEARCH_CONDITION
EMP_LAST_NAME_NN	C	"LAST_NAME" IS NOT NULL
EMP_EMAIL_NN	C	"EMAIL" IS NOT NULL
EMP_HIRE_DATE_NN	C	"HIRE_DATE" IS NOT NULL
EMP_JOB_NN	C	"JOB_ID" IS NOT NULL
EMP_SALARY_MIN	C	salary > 0
EMP_EMAIL_UK	U	
EMP_EMP_ID_PK	P	
EMP_DEPT_FK	R	

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 in the slide displays the constraints on the EMPLOYEES table.

Note: Constraints that are not named by the table owner receive the system-assigned constraint name. In constraint type, C stands for CHECK, P for PRIMARY KEY, R for referential integrity, and U for UNIQUE key. Notice that the NOT NULL constraint is really a CHECK constraint.

Viewing the Columns Associated with Constraints

View the columns associated with the constraint names in the USER_CONS_COLUMNS view.

```
SELECT   constraint_name, column_name
FROM     user_cons_columns
WHERE    table_name = 'EMPLOYEES';
```

CONSTRAINT_NAME	COLUMN_NAME
EMP_DEPT_FK	DEPARTMENT_ID
EMP_EMAIL_NN	EMAIL
EMP_EMAIL_UK	EMAIL
EMP_EMP_ID_PK	EMPLOYEE_ID
EMP_HIRE_DATE_NN	HIRE_DATE
EMP_JOB_FK	JOB_ID
EMP_JOB_NN	JOB_ID
EMP_LAST_NAME_NN	LAST_NAME
EMP_MANAGER_FK	MANAGER_ID
EMP_SALARY_MIN	SALARY

You can view the names of the columns involved in constraints by querying the USER_CONS_COLUMNS data dictionary view. This view is especially useful for constraints that use system-assigned names.

ASSIGNMENTS

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.

2. 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.

3. 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`.

4. 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`.

CONSTRAINT_NAME	C
MY_DEPT_ID_PK	P
SYS_C002541	C
MY_EMP_ID_PK	P
MY_EMP_DEPT_ID_FK	R

5. 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.

If you have time, complete the following exercise:

6. 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.