شركــة تحالـف الإمــارات للحــلـــول التـقـنـيــة ذ.م.م.
TAHALUF AL EMARAT TECHNICAL SOLUTIONS L.L.C.

Database Design and Programming

Tahaluf Training Center 2021

شركــة تحالـف الإمــارات للحــلـــول التـقـنـيــة ذ.م.م.
TAHALUF AL EMARAT TECHNICAL SOLUTIONS L.L.C.

Chapter 02

constraint

Use a **constraint** to define an integrity constraint--a rule that restricts the values in a database.

Oracle Database lets you create six types of constraints and lets you declare them in two ways.

constraint

شركة تحالف الإمارات للحلول التقنية ذ.م.م.
TAHALUF AL EMARAT TECHNICAL SOLUTIONS L.L.C.

The six types of integrity constraint are described briefly here :

- **NOT NULL** constraint prohibits a database value from being null.

- **unique** constraint prohibits multiple rows from having the same value in the same column or combination of columns but allows some values to be null.

- **primary key** constraint combines a NOT NULL constraint and a unique constraint in a single declaration. That is, it prohibits multiple rows from having the same value in the same column or combination of columns and prohibits values from being null.

constraint

- **foreign key** constraint requires values in one table to match values in another table.
- **check** constraint requires a value in the database to comply with a specified condition.
- **REF** column by definition references an object in another object type or in a relational table. A REF constraint lets you further describe the relationship between the REF column and the object it references.

constraint

You can define constraints syntactically in two ways:

1. As part of the definition of an individual column or attribute. This is called **inline** specification.

2. As part of the table definition. This is called **out-of-line** specification.

NOT NULL constraints must be declared inline. All other constraints can be declared either inline or out of line.

Example of a column-level constraint

```
CREATE TABLE employees(
employee_id NUMBER(6)
CONSTRAINT emp_emp_id_pk PRIMARY KEY,
first_name VARCHAR2(20),
PhoneNumber Number(10)
)
```

شركة تحالف الإمارات للحلول التقنية ذ.م.م.
TAHALUF AL EMARAT TECHNICAL SOLUTIONS L.L.C.

Example of a table-level 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));
```

Foreign Key Constraint Table 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 Constraint Column level

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

Foreign Key Constraint

شركة تحالف الإمارات للحلول التقنية ذ.م.م.
TAHALUF AL EMARAT TECHNICAL SOLUTIONS L.L.C.

A **foreign key** is a way to enforce referential integrity within your Oracle database. A foreign key means that values in one table must also appear in another table.

The referenced table is called the **parent table** while the table with the foreign key is called the **child table**. The foreign key in the child table will generally reference a primary key in the parent table.

A foreign key can be defined in either a **CREATE TABLE** statement or an **ALTER TABLE** statement

Foreign Key Constraint

```
CREATE TABLE supplier
( supplier_id numeric(10) not null,
  supplier_name varchar2(50) not null,
  contact_name varchar2(50),
  CONSTRAINT supplier_pk PRIMARY KEY (supplier_id)
);

CREATE TABLE products
( product_id numeric(10) not null,
  supplier_id numeric(10) not null,
  CONSTRAINT fk_supplier
    FOREIGN KEY (supplier_id)
    REFERENCES supplier(supplier_id)
);
```

Foreign Key Constraint

```
CREATE TABLE supplier
( supplier_id numeric(10) not null,
  supplier_name varchar2(50) not null,
  contact_name varchar2(50),
  CONSTRAINT supplier_pk PRIMARY KEY (supplier_id, supplier_name)
);

CREATE TABLE products
( product_id numeric(10) not null,
  supplier_id numeric(10) not null,
  supplier_name varchar2(50) not null,
  CONSTRAINT fk_supplier_comp
    FOREIGN KEY (supplier_id, supplier_name)
    REFERENCES supplier(supplier_id, supplier_name)
);
```

Foreign Keys with Cascade Delete

foreign key with **cascade delete** means that if a record in the parent table is deleted, then the corresponding records in the child table will automatically be deleted. This is called a cascade delete in Oracle.

A foreign key with a cascade delete can be defined in either a **CREATE TABLE** statement or an **ALTER TABLE** statement.

Foreign Keys with Cascade Delete

```
CREATE TABLE supplier
( supplier_id numeric(10) not null,
  supplier_name varchar2(50) not null,
  contact_name varchar2(50),
  CONSTRAINT supplier_pk PRIMARY KEY (supplier_id)
);

CREATE TABLE products
( product_id numeric(10) not null,
  supplier_id numeric(10) not null,
  CONSTRAINT fk_supplier
    FOREIGN KEY (supplier_id)
    REFERENCES supplier(supplier_id)
    ON DELETE CASCADE
);
```

Foreign Keys with Cascade Delete

TAHALUF AL EMARAT TECHNICAL SOLUTIONS L.L.C.

```
CREATE TABLE supplier
( supplier_id numeric(10) not null,
  supplier_name varchar2(50) not null,
  contact_name varchar2(50),
  CONSTRAINT supplier_pk PRIMARY KEY (supplier_id)
);

CREATE TABLE products
( product_id numeric(10) not null,
  supplier_id numeric(10) not null,
  CONSTRAINT fk_supplier
    FOREIGN KEY (supplier_id)
    REFERENCES supplier(supplier_id)
    ON DELETE CASCADE
);
```

Using an ALTER TABLE statement

TAHALUF AL EMARAT TECHNICAL SOLUTIONS L.L.C.

```
ALTER TABLE products
ADD CONSTRAINT fk_supplier
  FOREIGN KEY (supplier_id)
  REFERENCES supplier(supplier_id)
  ON DELETE CASCADE;
```

We could also create a foreign key (with a cascade delete) with more than one field as in the example below:

```
ALTER TABLE products
ADD CONSTRAINT fk_supplier
  FOREIGN KEY (supplier_id, supplier_name)
  REFERENCES supplier(supplier_id, supplier_name)
  ON DELETE CASCADE;
```

Foreign Keys with Set Null on Delete

A foreign key with "**set null on delete**" means that if a record in the parent table is deleted, then the corresponding records in the child table will have the foreign key fields set to null. The records in the child table will not be deleted.

A foreign key with a "**set null on delete**" can be defined in either a CREATE TABLE statement or an ALTER TABLE statement.

Foreign Keys with Set Null on Delete

```
CREATE TABLE supplier
( supplier_id numeric(10) not null,
  supplier_name varchar2(50) not null,
  contact_name varchar2(50),
  CONSTRAINT supplier_pk PRIMARY KEY (supplier_id)
);

CREATE TABLE products
( product_id numeric(10) not null,
  supplier_id numeric(10),
  CONSTRAINT fk_supplier
    FOREIGN KEY (supplier_id)
    REFERENCES supplier(supplier_id)
    ON DELETE SET NULL
);
```

Foreign Keys with Set Null on Delete

```
CREATE TABLE supplier
( supplier_id numeric(10) not null,
  supplier_name varchar2(50) not null,
  contact_name varchar2(50),
  CONSTRAINT supplier_pk PRIMARY KEY (supplier_id, supplier_name)
);

CREATE TABLE products
( product_id numeric(10) not null,
  supplier_id numeric(10),
  supplier_name varchar2(50),
  CONSTRAINT fk_supplier_comp
    FOREIGN KEY (supplier_id, supplier_name)
    REFERENCES supplier(supplier_id, supplier_name)
    ON DELETE SET NULL
);
```

Using an ALTER TABLE statement

In this example, we've created a foreign key "**with a set null on delete**" called fk_supplier that references the supplier table based on the supplier_id field.

```
ALTER TABLE products
ADD CONSTRAINT fk_supplier
  FOREIGN KEY (supplier_id)
  REFERENCES supplier(supplier_id)
  ON DELETE SET NULL;
```

Using an ALTER TABLE statement

We could also create a foreign key "**with a set null on delete**" with more than one field as in the example below:

```
ALTER TABLE products
ADD CONSTRAINT fk_supplier
  FOREIGN KEY (supplier_id, supplier_name)
  REFERENCES supplier(supplier_id, supplier_name)
  ON DELETE SET NULL;
```

Add constraint after table creation

```
ALTER TABLE location
ADD CONSTRAINT PK_location PRIMARY KEY (lnumber,LName);

ALTER TABLE Department
ADD CONSTRAINT FK_manager
FOREIGN KEY (mg_ssn) REFERENCES employee (SSN);

ALTER TABLE mytable
MODIFY (mynumber NUMBER(8,2) CONSTRAINT
my_cons_name NOT NULL);

ALTER TABLE supplier
ADD CONSTRAINT supplier_unique UNIQUE (supplier_id);
```

Add constraint after table creation

```
ALTER TABLE suppliers
ADD CONSTRAINT check_supplier_name
CHECK (supplier_name IN ('IBM', 'Microsoft', NVIDIA'));
```

View Constraint

```
select constraint_name,constraint_type,table_name
from user_constraints
```

Enable and disable Constraint

شركة تحالف الإمارات للحلول التقنية ذ.م.م.
TAHALUF AL EMARAT TECHNICAL SOLUTIONS L.L.C.

1- alter table table_name **ENABLE** constraint constraint_name;

2- alter table table_name **DISABLE** constraint constraint_name;

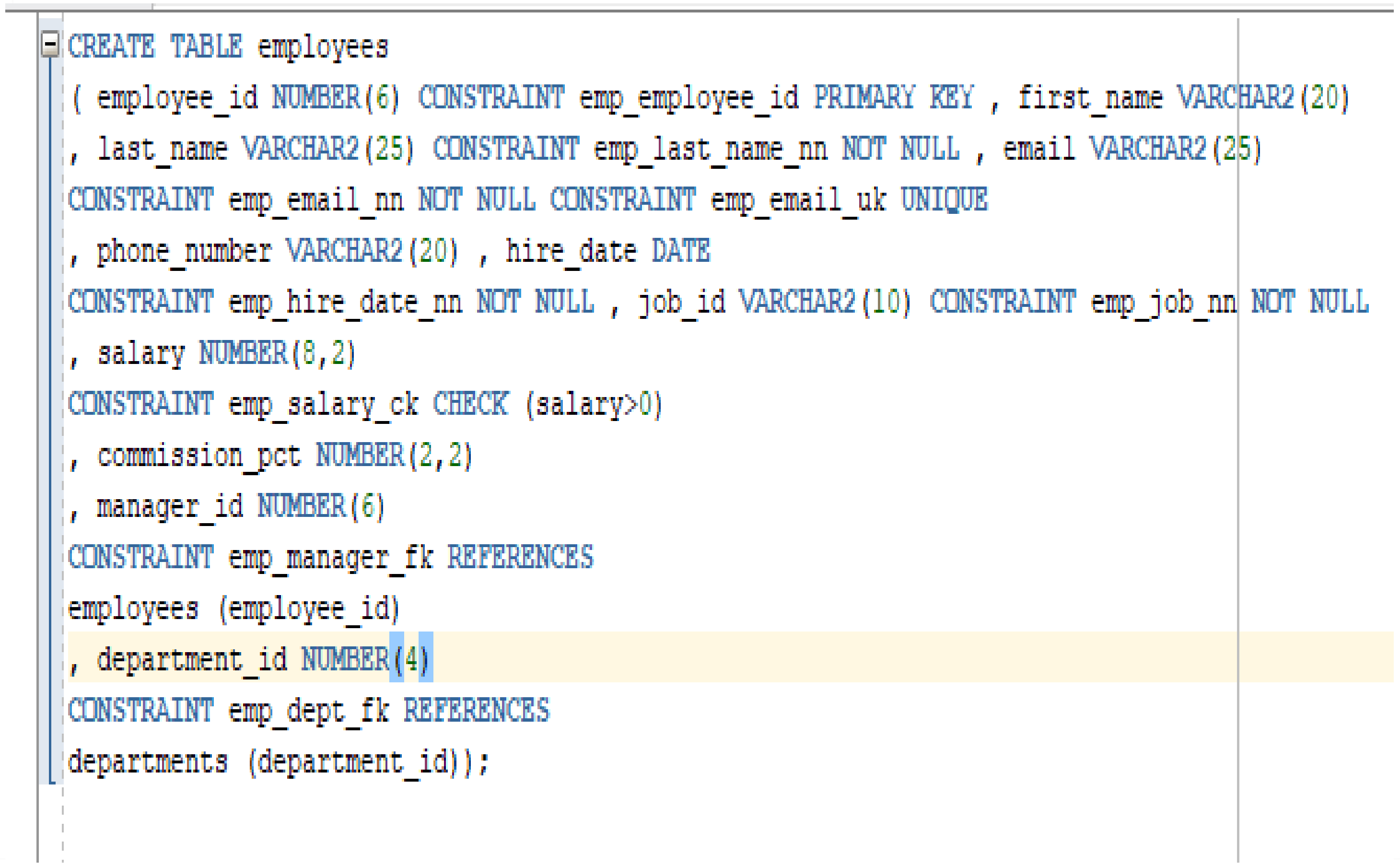
select constraint_name,constraint_type,table_name, Status from user_constraints

Drop constraint

```
ALTER TABLE table_name
DROP CONSTRAINT constraint_name;

ALTER TABLE location
DROP CONSTRAINT PK_location;
```

General Example



```
CREATE TABLE employees
( employee_id NUMBER(6) CONSTRAINT emp_employee_id PRIMARY KEY , first_name VARCHAR2(20)
, last_name VARCHAR2(25) CONSTRAINT emp_last_name_nn NOT NULL , email VARCHAR2(25)
CONSTRAINT emp_email_nn NOT NULL CONSTRAINT emp_email_uk UNIQUE
, phone_number VARCHAR2(20) , hire_date DATE
CONSTRAINT emp_hire_date_nn NOT NULL , job_id VARCHAR2(10) CONSTRAINT emp_job_nn NOT NULL
, salary NUMBER(8,2)
CONSTRAINT emp_salary_ck CHECK (salary>0)
, commission_pct NUMBER(2,2)
, manager_id NUMBER(6)
CONSTRAINT emp_manager_fk REFERENCES
employees (employee_id)
, department_id NUMBER(4)
CONSTRAINT emp_dept_fk REFERENCES
departments (department_id));
```

ALTER TABLE

- Add a new column
- Modify an existing column definition
- Define a default value for the new column
- Drop a column
- Rename a column
- Change table to read-only status

ALTER TABLE

```
ALTER TABLE employees READ ONLY

ALTER TABLE employees READ WRITE
```

ALTER TABLE

1. ALTER TABLE table_name
ADD column_name datatype;

2. ALTER TABLE DROP COLUMN

3. ALTER TABLE table_name
MODIFY column_name datatype;

ALTER TABLE "table_name"
RENAME COLUMN “oldname" TO “newname ”;

Drop table

Use the **DROP TABLE** statement to move a table or object table to the recycle bin or to remove the table and all its data from the database entirely.

DROP TABLE table name