U7. 01 SQL Creación de tablas

El objetivo de esta unidad es **crear, modificar y suprimir tablas** en SQL y todo lo que esto conlleva.

Las órdenes para poder realizar estas acciones se incluyen en el lenguaje de descripción de datos o **DDL (*Data Description Language*)**.

Al crear una tabla debemos planificar con antelación:

- Un nombre que identifique su contenido:
  - La denominación de la tabla puede tener de 1 a 30 caracteres de longitud.
  - Ha de ser un nombre único y no puede ser una palabra reservada de Oracle.
  - El primer carácter ha de ser alfabético y el resto pueden ser letras, números y el guion bajo (_)
- Los nombres de las columnas
- El tipo de dato y el tamaño que tendrá cada columna
- Las columnas obligatorias
- Los valores por defecto
- Las restricciones que han de cumplir los campos
- La clave principal de la tabla
- Las claves ajenas

1. Creación de una tabla

Para crear una tabla utilizamos la orden **CREATE TABLE**

CREATE TABLE NombreTabla
(
Columna1 Tipo_dato [NOT NULL],
Columna2 Tipo_dato [NOT NULL],
. . . . . .
Columnan Tipo_dato [NOT NULL]

) [TABLESPACE espacio_de_tabla];

Tablespace indica donde almacenamos la tabla, si no ponemos nada se almacena en el tablespace por defecto del usuario

Principales tipos de datos de Oracle:

CHAR	Cadena de caracteres de longitud fija. De 1 a 255 caracteres. El dato se rellena con blancos a la derecha hasta alcanzar la longitud definida Si no damos longitud permite un único carácter
VARCHAR2	Cadena de caracteres de longitud variable Máximo 2000 caracteres. Siempre hay que darle longitud. (Existe el tipo VARCHAR y funciona como VARCHAR2 pero Oracle recomienda el uso de este último por compatibilidad con futuras versiones)
NUMBER	Para datos numéricos, números enteros y reales. Admite hasta 38 dígitos Puede indicarse la precisión deseada NUMBER (p,s) "p" es el total de dígitos y "s" es el número de decimales.
DATE	Tipo de datos fecha Incluye día, mes, año, hora, minutos y segundos.

Ejemplo:

```
CREATE TABLE PRUEBA_TIPOS
(
        TIPO        CHAR,
        NOMBRE      CHAR (10),
        APELLIDOS   VARCHAR2 (20),
        EDAD        NUMBER (2),
        PESO        NUMBER (6,3), -- 6 dígitos en total, de ellos 3 son decimales
        SALARIO     NUMBER (*,2), -- cualquier cantidad de dígitos, 2 son decimales
        TOTAL       NUMBER,
        FECHA_NAC   DATE
);
```

Otros tipos de datos:

LONG	Cadena de caracteres de longitud variable. Longitud máxima 2Gb
RAW	Como VARCHAR2 pero en binario Longitud máxima 255 bytes
LONG RAW	Como el LONG pero en binario Soporte de datos multimedia
ROWID	Tipo de datos para almacenar direcciones virtuales de filas
INTEGER INT	Para números enteros
DECIMAL(p,s)	Como NUMBER
FLOAT REAL SMALLINT	Numéricos

Podemos consultar las tablas creadas por el usuario por medio de la vista **USER_TABLES**

Y visualizar los nombres de las tablas creadas mediante

SELECT **TABLE_NAME** FROM **USER_TABLES**;

2. Integridad de datos

Al almacenar datos en nuestras tablas podemos establecer una serie de restricciones predefinidas para mantener la integridad de los datos.

Integridad referencial: garantiza que los valores de una columna (o columnas) dependan de los de otra tabla. Garantiza la coherencia entre los valores de dos tablas que dependen una de otra. Por ejemplo: el valor de la columna dept_no de EMPLE ha de existir en DEPART.

Restricción de integridad: será una regla que restringe el rango de valores para una o más columnas. Por ejemplo: una columna que no puede tomar valores negativos.

3. Restricciones en CREATE TABLE

Vienen definidas mediante la cláusula **CONSTRAINT** (aunque a veces no sea necesario especificarlo)

Pueden ser para una sola columna (restricción de columna)

o implicar a varias columnas (restricción de tabla)

Pueden especificarse tras la definición de una columna (restricción de columna)

o tras todas las definiciones de columna (restricción de tabla)

Los distintos tipos son:

- Claves primarias (PK Primary Key)
- Claves foráneas o ajenas (FK Foreign Key)
- Obligatoriedad (NOT NULL)
- Valores por defecto (DEFAULT)
- Verificación de condiciones (CHECK)

3.1. Clave Primaria. La restricción PRIMARY KEY

- Una clave primaria dentro de una tabla es una columna o un conjunto mínimo de columnas que identifica unívocamente a cada fila.
- Debe ser única y no nula (obligatoria).
- Sólo puede haber una por tabla.
- Para definir una clave primaria utilizamos la restricción PRIMARY KEY

Podemos utilizar los siguientes **formatos** para definir claves primarias dependiendo de que queramos o no darle un nombre a la clave primaria y de que la clave esté compuesta por solo una o varias columnas.

Cuando la clave la forme una sola columna:

i.
```
CREATE TABLE Tabla
( ColumnaA        tipo      PRIMARY KEY,
  . . . . . );
```

y si queremos darle nombre a la clave elegimos la opción ii)

ii.
```
CREATE TABLE Tabla
( ColumnaA        tipo      CONSTRAINT  PK_Tabla  PRIMARY KEY,
  . . . . . );
```

Cuando la clave la forman varias columnas:

iii.
```
CREATE TABLE Tabla
( ColumnaA        tipo,
  ColumnaB        tipo,
  . . . . . ,
 PRIMARY KEY (ColumnaA, ColumnaB)  );
```

y si queremos darle nombre a la clave elegimos la opción iv)

iv.
```
CREATE TABLE Tabla
( ColumnaA        tipo,
  ColumnaB        tipo,
  . . . . . ,
 CONSTRAINT PK_TABLA PRIMARY KEY (ColumnaA, ColumnaB)  );
```

En el caso de que esté formada por varias columnas solo podemos elegir la opción (iii.) o (iv.)

Ejemplos:

Archivo Editar Hoja de trabajo Ayuda

```
drop table notas;
drop table asignaturas;
drop table alumnos;
CREATE TABLE ALUMNOS
( DNI VARCHAR2(10) PRIMARY KEY,
  APENOM VARCHAR2(30),
  DIREC VARCHAR2(30),
  POBLA  VARCHAR2(15),
  TELEF  VARCHAR2(10)
);

CREATE TABLE ASIGNATURAS
( COD NUMBER(2),
  NOMBRE VARCHAR2(25),
  CONSTRAINT pk_asignaturas PRIMARY KEY (COD)
);

CREATE TABLE NOTAS
( DNI VARCHAR2(10) NOT NULL,
  COD NUMBER(2) NOT NULL,
  NOTA NUMBER(2),
  CONSTRAINT pk_notas PRIMARY KEY (DNI,COD)
);
```

3.2. Obligatoriedad. Restricción NOT NULL

No permite el valor nulo para una determinada columna

Se resuelve añadiendo NOT NULL después de la definición de la columna

Ejemplo: Nombre VARCHAR2(20) NOT NULL

En el caso de que una columna sea o forme parte de una clave primaria es implícito el hecho de que sea no nula y no debemos poner esta restricción.

3.3. Valores por defecto. La especificación DEFAULT

Podemos especificar un valor por defecto para una columna, de este modo, si no se da un valor a la columna al insertar una fila, esta columna tomará el valor que se haya dado por defecto.

La sintaxis que se sigue es:

Columna tipo **DEFAULT** valor_por_defecto,

Ejemplo:

```
drop table T;

CREATE TABLE T
( A DATE DEFAULT SYSDATE,
  B NUMBER DEFAULT 0,
  C VARCHAR2(20) DEFAULT 'SIN NADA',
  D VARCHAR2(20) DEFAULT USER
);
```

3.4. Verificación de condiciones. Restricción CHECK

Se utilizan para limitar el rango de valores de una determinada columna o para que se cumplan determinadas condiciones.

Se especifican tras la definición de las columnas, es conveniente darles un nombre porque si no el sistema les da uno que no es fácilmente entendible.

La sintaxis que se sigue es:

[CONSTRAINT Nombre] CHECK (Condición)

Ejemplo:

```
drop table T;

CREATE TABLE T
( A DATE,
  B NUMBER,
  C VARCHAR2(20),
  D NUMBER,
  E DATE,
  CONSTRAINT CK_T_B_Positivo CHECK (B > 0),
  CONSTRAINT CK_T_D CHECK (D BETWEEN 10 AND 100),
  CONSTRAINT CK_T_E CHECK (E IS NULL OR E > A),
  CONSTRAINT CK_T_C_Mayuscula CHECK (C=UPPER(C))
);
```

3.5. Valores únicos. Restricción UNIQUE

Evita valores repetidos en la misma columna.

Puede ser para una sola columna o para uno conjunto de columnas.

Se especifica **UNIQUE** tras definir la columna o como una constraint al final de la definición de la tabla (como la primary key)

Ejemplo:

```
CREATE TABLE T
( num_emple NUMBER PRIMARY KEY,
  DNI VARCHAR2(9) UNIQUE,
  NOMBRE VARCHAR2(20),
  APELLIDOS VARCHAR2(20),
  CONSTRAINT T_NOM_APELL UNIQUE (NOMBRE,APELLIDOS)
);
```

3.6. Clave Foránea o Ajena. Restricción FOREIGN KEY

Una clave ajena está formada por una o varias columnas que están asociadas a una clave primaria de otra o la misma tabla.

Una clave ajena puede tomar valor NULL o un valor de la clave referenciada.

Partimos de una tabla T1 cuya clave es la columna C1
Queremos que la columna C2 de la tabla T2 sea clave ajena a T1

Podemos utilizar los siguientes formatos para crear una clave ajena:

i.
```
CREATE TABLE T2
(       A tipo,
        B  tipo,
        . . . .
        C2 tipo REFERENCES T1,
        . . .
);
```

ii. CREATE TABLE T2
(A tipo,
B tipo,
. . . .
C2 tipo,
. . .
CONSTRAINT FK_T2_T1 **FOREIGN KEY** (C2) **REFERENCES** T1
);

Ejemplo:

```
drop table notas;
drop table asignaturas;
drop table alumnos;
CREATE TABLE ALUMNOS
( DNI VARCHAR2(10) PRIMARY KEY,
  APENOM VARCHAR2(30),
  DIREC VARCHAR2(30),
  POBLA  VARCHAR2(15),
  TELEF  VARCHAR2(10)
);

CREATE TABLE ASIGNATURAS
( COD NUMBER(2),
  NOMBRE VARCHAR2(25),
  CONSTRAINT pk_asignaturas PRIMARY KEY (COD)
);

CREATE TABLE NOTAS
( DNI VARCHAR2(10) REFERENCES ALUMNOS,
  COD NUMBER(2) REFERENCES ASIGNATURAS,
  NOTA NUMBER(2),
  CONSTRAINT pk_notas PRIMARY KEY (DNI,COD)
);
```

O

```
CREATE TABLE NOTAS
( DNI VARCHAR2(10),
  COD NUMBER(2),
  NOTA NUMBER(2),
  CONSTRAINT pk_notas PRIMARY KEY (DNI,COD),
  CONSTRAINT FK_NOTAS_ALUMNOS FOREIGN KEY (DNI) REFERENCES ALUMNOS,
  CONSTRAINT FK_NOTAS_ASIGNATURAS FOREIGN KEY (COD) REFERENCES ASIGNATURAS
);
```

Los campos referenciados no tienen por qué llamarse igual pero sí ser del mismo tipo y tamaño.

```
CREATE TABLE NOTAS
( DNI_ALUMNO VARCHAR2(10),
  COD_ASIGNATURA NUMBER(2),
  NOTA NUMBER(2),
  CONSTRAINT pk_notas PRIMARY KEY (DNI_ALUMNO,COD_ASIGNATURA),
  CONSTRAINT FK_NOTAS_ALUMNOS FOREIGN KEY (DNI_ALUMNO) REFERENCES ALUMNOS,
  CONSTRAINT FK_NOTAS_ASIGNATURAS FOREIGN KEY (COD_ASIGNATURA) REFERENCES ASIGNATURAS
);
```

Una clave ajena puede estar compuesta de varias columnas. Es el caso de que la tabla a la que referencia tenga una clave formada por varias columnas.

Si se añade la cláusula ON DELETE CASCADE en la opción REFERENCES al borrar en la tabla principal una fila, se borrarán automáticamente las filas asociadas en la segunda tabla.

```
CREATE TABLE NOTAS
( DNI VARCHAR2(10),
  COD NUMBER(2),
  NOTA NUMBER(2),
  CONSTRAINT pk_notas PRIMARY KEY (DNI,COD),
  CONSTRAINT FK_NOTAS_ALUMNOS FOREIGN KEY (DNI) REFERENCES ALUMNOS,
  CONSTRAINT FK_NOTAS_ASIGNATURAS FOREIGN KEY (COD) REFERENCES ASIGNATURAS
  ON DELETE CASCADE
);
```

En este caso, al borrar una asignatura de la tabla ASIGNATURAS se borrarán todas las notas de esa asignatura.

En el caso de querer borrar un alumno que tenga notas el sistema no lo permitirá (dado que tiene notas habría que borrar primero las notas de ese alumno)

4. Acceder a las constraints almacenadas

Para ver las constraints de nuestras tablas tenemos la vista **USER_CONSTRAINTS**

De los campos que contiene los que nos interesan principalmente son:

- table_name : nombre de la tabla a la que pertenece
- constraint_name : nombre de la constraint
- constraint_type : tipo de la constraint
  - C para CHECK y NOT NULL
  - P para PRIMARY KEY
  - R para FOREIGN KEY
  - U para UNIQUE
- search_condition : para las de tipo check contiene la condición

```
select table_name, constraint_type, constraint_name, search_condition
 from user_constraints
order by table_name;
```

U7. 02 Modificación de tablas

Una vez creadas las tablas podemos **modificar su diseño**

- añadiendo nuevas columnas
- modificando el tipo de la columna
- borrando columnas
- añadir y borrar constraints

Ejemplo: create table ELEMENTOS
(codigo varchar2(6));

- **Añadir columna:**

  ALTER TABLE nombretabla

  ADD nombrecolumna tipo;

```
alter table ELEMENTOS
add  (descripcion varchar(10), unidades number);

alter table ELEMENTOS
add  precio number;
```

- **Modificar el tipo de una columna:**

  ALTER TABLE nombretabla

  MODIFY nombrecolumna nuevotipo;

```
alter table ELEMENTOS
modify (descripcion varchar(20),
        codigo number not null);
```

- **Borrar una columna:**

  ALTER TABLE nombretabla

  DROP COLUMN nombrecolumna;

```
alter table ELEMENTOS
drop column descripcion ;
```

- **Añadir una constraint:**

  ALTER TABLE nombretabla

  ADD CONSTRAINT nombre …..;

  alter table ELEMENTOS
  add (constraint pk_elem primary key (codigo),
  constraint ck_elem check (codigo>=1)) ;

- **Borrar una constraint:**

  ALTER TABLE nombretabla

  DROP CONSTRAINT nombre;

  alter table ELEMENTOS drop constraint ck_elem ;

U7. 03 Borrado de Tablas.

Para borrar una tabla ejecutamos la orden DROP

DROP TABLE Nombre_Tabla ;

DROP TABLE ELEMENTOS ;

En el caso de que la tabla esté siendo referenciada por otras tablas este borrado dará error.

Ejemplo :

```
drop  table  ELEMENTOS;
drop  table  CATEGORIAS;
create table CATEGORIAS
(tipo char (2) primary key, descripcion varchar2(20));
create table ELEMENTOS
 (codigo varchar2(6),
  tipo char(2) references CATEGORIAS);
```

Si ahora intentamos hacer DROP TABLE CATEGORIAS

```
drop table CATEGORIAS
           *
ERROR en línea 1:
ORA-02449: claves únicas/primarias en la tabla referidas por claves ajenas
```

Por esto es importante el orden de borrado en un script.

Existe una manera de borrar una tabla y a la vez borrar las foreign key que haya en otras tablas refiriendose a ella. Para ello utilizamos la claúsula CASCADE CONSTRAINTS

DROP TABLE Nombre_Tabla CASCADE CONSTRAINTS

```
drop table CATEGORIAS CASCADE CONSTRAINTS;

Tabla borrada.
```