# Crear una Base de Datos Relacional

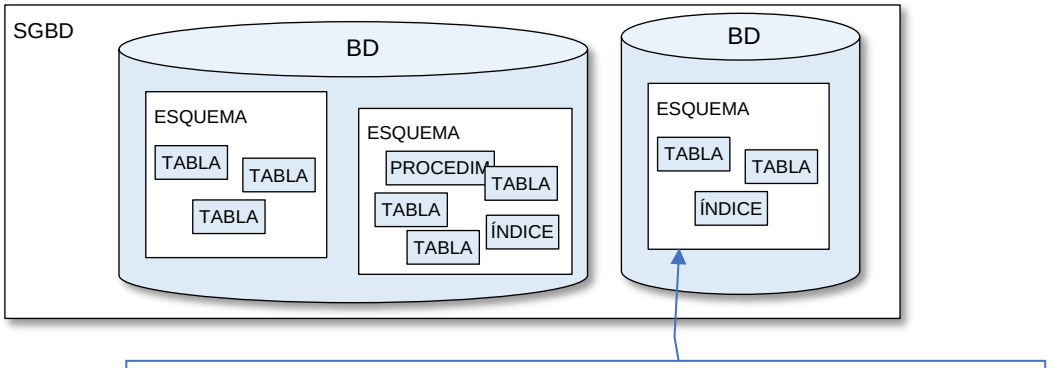
## ÍNDICE

En este documento, a no ser que se indique lo contrario, se reflejan los comandos SQL referidos al Sistema Gestos de Base de Datos (SGBD) Oracle Database.

# 1. Database

## 1.1. Base de datos, esquemas y objetos

Una Base de Datos dentro del contexto del Sistema Gestor de Base de Datos (SGBD) es un conjunto de objetos que se utilizarán para gestionar los datos. Los objetos principales son tablas, pero pueden haber otros (vistas, índices, enlaces, procedimientos, secuencias, enlaces…). Estos objetos están contenidos en esquemas. Normalmente un esquema está asociado a un usuario.




En Oracle, cada usuario de una base de datos tiene un esquema, que tendrá el mismo nombre que el usuario con el que se ha accedido y sirve para almacenar los objetos que posea ese usuario. Solo el usuario propietario y los administradores tienen acceso a los objetos del esquema, aunque pueden modificarse los permisos.

Crear una base de datos implica indicar los archivos y ubicaciones que se van a utilizar además de otras indicaciones técnicas y administrativas. Todo esto sólo lo puede realizar si se tiene privilegio de Administrador.

## 1.2. Instrucciones Database

| | |
|---|---|
| `CREATE DATABASE nombreBD;` | Crear base de datos |
| `SHOW DATABASES;` | Ver bases de datos disponibles |
| `USE nombreBD;` | Entrar a base de datos * |
| `DROP DATABASE nombreBD;` | Borrar base de datos |

Los comandos anteriores pertenecen al estándar SQL, sin embargo puede haber diferencias en los SGBD.

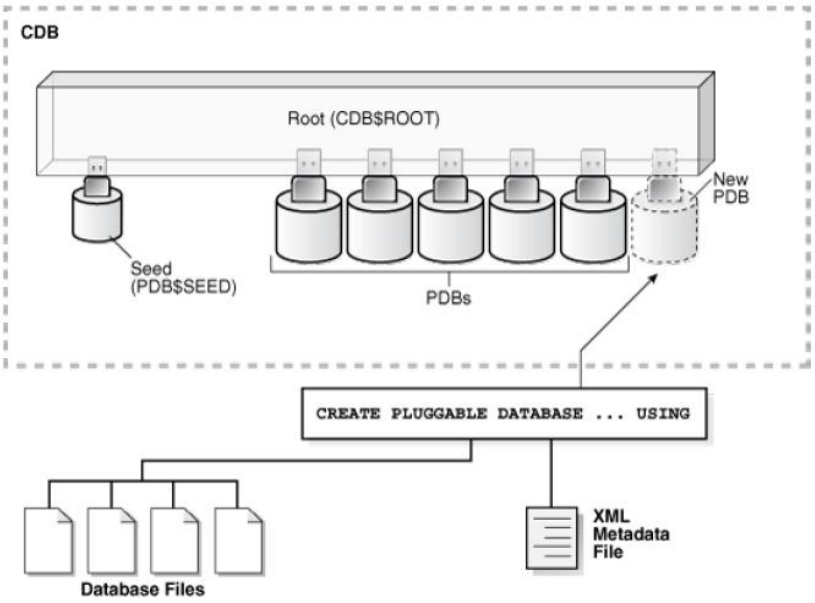
- **Mysql**

Por ejemplo, en Mysql para visualizar la base de datos actual (a la que se está contectado):

```
select database() from dual;
```

- **Oracle Database**

En Oracle Database las bases de datos se denominan PDB (Pluggable Database):




Algunos comandos para manejar bases de datos (PDBs) en Oracle Database:

| | |
|---|---|
| `create pluggable database …;` | Crear base de datos PDB. Pueden crearse a partir de un modelo (Seed), clonando otra PDB o utilizando ficheros de configuración. |
| `select * from v$pdbs;` | Ver bases de datos disponibles |
| `alter session set container=nombrePDB;` | Usar base de datos (acceder a ella)* |
| `show con_name;` | Ver base de datos actual. |
| `drop pluggable database nombrePDB including datafiles;` | Borrar base de datos |

(*) Normalmente es posible establecer la conexión con el SGBD directamente dentro de una BD concreta. Por ejemplo accediendo desde terminal de comandos:

| | |
|---|---|
| mysql -u *usuario* -p *nombreBD* | Acceso a una BD dentro de mysql |
| sqlplus *usuario*/*password*@//*máquina*:*puerto*/nombre*PBD* | Acceso a una PDB dentro de Oracle Database |

### Actividad. Acceder a una PDB

> Conéctate al contenedor CDB de Oracle con el usuario administrador SYSTEM y revisa las PDBs existentes.
>
> Comprueba que el usuario "alumno" no tiene permisos para acceder al CDB. Conéctate con el usuario "alumno" a la PDB XEPDB1.

## 2. Table

Una tabla es un objeto dentro de una base de datos que representa una tabla o relación del modelo relacional. Está compuesta por columnas o campos, cada uno de los cuales con un nombre y tipo de datos.

| NUM_EMP | NOMBRE | FECHA_NAC | FECHA_ALTA | OFICINA | VENTAS | JEFE_VENTAS |
|---|---|---|---|---|---|---|
| 101 | Adrián Alonso Gutierrez | 18/08/79 | 03/05/00 | 1 | 75000 | (null) |
| 110 | Lucía Rodríguez Otura | 21/08/83 | 01/09/11 | 1 | 45000 | 101 |
| 112 | Alejandra Miere Moya | 06/07/84 | 05/10/15 | 1 | 56020 | 110 |

### 2.1. SQL. Creación de tablas

Formato de creación de tabla simple:

```
CREATE TABLE nombreTabla
(
  Columna1 tipo_dato,
  Columna2 tipo_dato,
  …
);
```

En Oracle Database el nombre de la tabla debe tener entre 1 y 30 caracteres de longitud, no debe ser una palabra reservada. El primer carácter debe ser alfanumérico y el resto pueden ser letras, números y el carácter de subrayado.

Más adelante se completará el formato añadiendo restricciones.

```
DESCRIBE nombreTabla;
```

Con este comando puede verse la estructura de una tabla.

### 2.2. Tipos de datos

El tipo de dato establece de qué clase es el dato que guarda en un campo de una tabla. Los principales son: texto, numérico y fecha.

En cada SGDB se establecen los distintos tipos. Un ejemplo de algunos tipos en mysql y Oracle Database:

| | Tipo de dato | | Descripción |
|---|---|---|---|
| | mysql | oracle | |
| numérico | TINYINT<br>SMALLINT<br>MEDIUMINT<br>INT, INTEGER<br>BIGINT | NUMBER | Número entero con o sin signo |
| | FLOAT<br>DOUBLE | NUMBER<br>BINARY_FLOAT<br>BINARY_DOUBLE | Número decimal |
| | BIT | | Número binario |
| texto | CHAR | CHAR<br>NCHAR | Cadena con longitud fija |
| | VARCHAR | VARCHAR2<br>NVARCHAR2 | Cadena |
| fecha | DATE | DATE | Fecha |
| | DATETIME<br>TIMESTAMP | TIMESTAMP | Fecha y hora |
| | TIME | | Hora |
| | YEAR | | Año |
| Datos binarios | TINYBLOB<br>MEDIUMBLOB<br>BLOB<br>LONGBLOB | BLOB<br>CLOB<br>NCLOB<br>BFILE<br>RAW | Datos de archivos (LOB: objeto grande) |

Cada tipo tiene una longitud definida que se mide en bytes, aunque en ocasiones se puede configurar el tamaño al utilizar el tipo.

## Actividad. Crear la primera tabla.

a) Crea la siguiente tabla CLIENTE**, cuya clave primaria es el campo NIF**:

```
CREATE TABLE CLIENTE
(
  NIF varchar2(9) primary key,
  nombre varchar2(30),
  apellido varchar2(30),
  telefono number(9)
);
```

Observa:

- Las definiciones individuales de campos se separan por comas
- No se pone coma en la última definición de campo
- Es indiferente usar mayúsculas o minúsculas
- Son indiferentes los saltos de línea
- Las sentencias acaban con punto y coma

b) Comprueba la estructura de la tabla (comando DESCRIBE).

c) Intenta crear una segunda tabla con el mismo nombre (repite la ejecución del comando). ¿Qué ocurre?

d) Borra la tabla (comando DROP).

- **¿Cómo se almacenan los strings? Diferencia entre CHAR y VARCHAR/VARCHAR2.**

Un carácter ocupa 1 byte de memoria.

- CHAR(n) tiene una longitud fija de n bytes.
- VARCHAR(n)/VARCHAR2(n) ocupa los bytes del dato guardado más uno de fin de string.

Por ejemplo:

| | Tamaño ocupado de memoria según el tipo de dato | |
|---|---|---|
| Valor a almacenar | CHAR(4) | VARCHAR(4) o VARCHAR2(4) |
| '' (vacío) | 4 bytes | 1 bytes |
| 'ab' | 4 bytes | 3 bytes |
| 'abcd' | 4 bytes | 5 bytes |
| 'abcde' | No es posible guardar ese dato | No es posible guardar ese dato |

Ejercicios: crear tablas sencillas. Ejercicio 1.

- **Números: Precisión y escala**

El formato es NUMBER (precisión, escala), siendo:

Precisión: número total de dígitos
Escala: número de decimales. Si la escala es un número negativo se realiza un redondeo a la unidad indicada (-1 redondea a las decenas, -2 redondea a las centenas, etc)

Por ejemplo:

| Dato actual | Formato | Almacenamiento |
|---|---|---|
| 7456123.89 | NUMBER | 7456123.89 |
| 7456123.89 | NUMBER(9) | 7456124 |
| 7456123.89 | NUMBER(9,2) | 7456123.89 |
| 7456123.89 | NUMBER(9,1) | 7456123.9 |
| 7456123.8 | NUMBER(6) | ERROR-Valor mayor que el que permite la precisión especificada para esta columna. |
| 7456123.8 | NUMBER(15,1) | 7456123.8 |
| 7456123.89 | NUMBER(7,-2) | 7456100 |
| 7456123.89 | NUMBER(-7,2) | ERROR-Especificador de precisión numérica está fuera de rango (1 a 38). |

Ejercicios: crear tablas. Campos numéricos con precisión y escala. Ejercicios 2, 3 y 4.

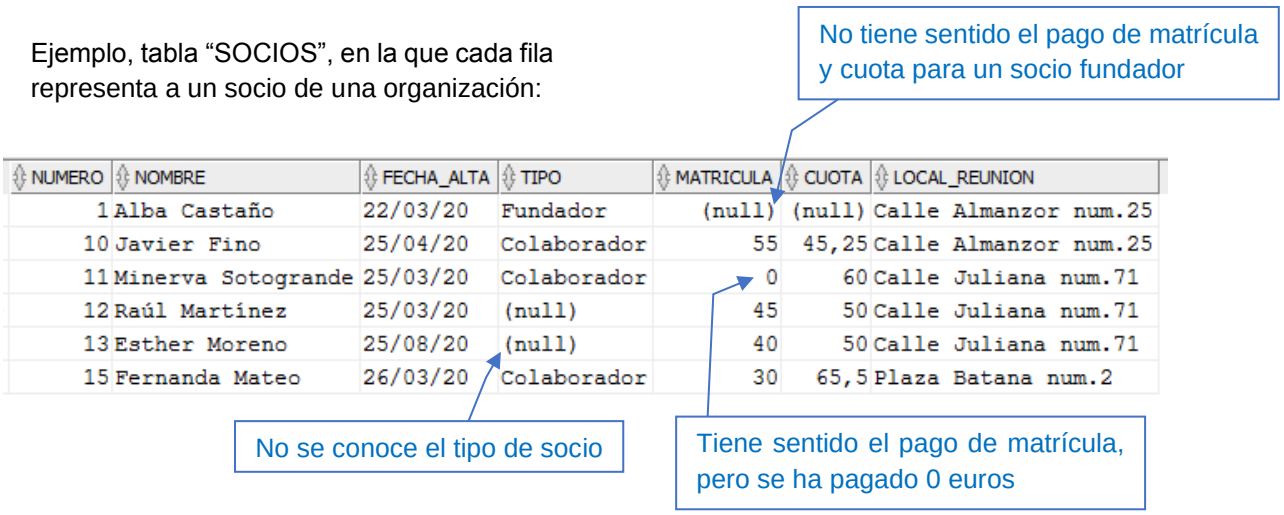
Nota: para realizar los ejercicios es necesario insertar y visualizar datos. Puedes revisar el formato simple de los comandos LMD en el anexo.

## 2.3. Valor null o nulo

Null es un valor especial que puede tomar un campo, independientemente del dominio y tipo de datos que tenga definido, y que significa que ese dato no está definido (ausencia de valor).

Puede ser que no se conozca el dato o bien que carezca de sentido en esa fila.

Ejemplo, tabla “SOCIOS”, en la que cada fila representa a un socio de una organización:




**No es lo mismo valor NULO que espacio en blanco.** Un espacio en blanco es un carácter que se tratará como una letra cualquiera, no es ausencia de valor.

**No es lo mismo valor NULO que espacio en cero.** Un cero es un número que indica una cantidad de 0 unidades, no es ausencia de valor.

Cuando se opera o compara un valor null, hay que seguir reglas específicas. Por ejemplo:

| | |
|---|---|
| 5 + null | = null |
| Verdadero AND null | = null |
| Falso AND null | = falso |
| Verdadero OR null | = falso |
| Falso OR null | = falso |
| NOT null | = null |

Se emplea el operador específico IS NULL o IS NOT NULL para comprobar si un valor o una expresión tiene el valor null. **No se utilizan los operadores habituales de comparación** igual (=) y distinto (!= <>).

### Actividad. Valor null

a) Crea la tabla MATERIAL, que tiene como clave primaria el campo “código”, con la siguiente instrucción:

```
CREATE TABLE MATERIAL
(
  codigo number(2) primary key,
  precio number(5,2),
  nombre varchar2(30)
);
```

b) Inserta dos registros mediante los siguientes comandos:

```
insert into material values (33, 0, 'Alumnio');
insert into material (codigo, nombre) values (34, 'Hierro');
```

Observa el formato del último comando. Se ha dado valor a las columnas “código” y “nombre”, pero no a “precio”. Al no indicar un valor se almacenará el valor NULL.

Comprueba el contenido de la tabla (comando SELECT) y compara el valor de la columna “precio” en cada registro. No es lo mismo 0 que NULL.

c) Inserta dos registros mediante los siguientes comandos:

```
insert into material values (35, 25.99, ' ');
insert into material (codigo, precio) values (36, 25.10);
```

Observa el formato del último comando. Se ha dado valor a las columnas “código” y “precio”, pero no a “nombre”. Al no indicar un valor se almacenará el valor NULL.

Comprueba el contenido de la tabla (comando SELECT) y compara el valor de la columna “nombre” en cada registro. No es lo mismo un espacio en blanco que NULL.

## 2.4. Restricciones (constraints)

- **¿Qué es una restricción o constraint?**

Una restricción es una condición que los datos de una o varias columnas de una tabla deben cumplir obligatoriamente. Se crea en el momento de crear la tabla (CREATE TABLE), aunque puede añadirse posteriormente (ALTER TABLE).

Tipos de restricciones:

- Primary Key (clave primaria)
- Foreign Key (clave externa)
- Unique (no repetido)
- Not Null (obligatorio)
- Check (validación)

- **Restricción de columna y de tabla**

Según la forma de definición de la restricción se distingue:

➞ **Restricción de columna:** en el comando de CREATE TABLE la restricción se define junto a la definición del campo al que afecta la restricción. Este tipo de restricción solo puede afectar a un campo.

```
CREATE TABLE nombreTabla
(
      Columna1 tipo_dato      [CONSTRAINT nombre_restricción]
                  [NOT NULL]
                  [UNIQUE]
                  [PRIMARY KEY]
                  [DEFAULT valor]
                  [REFERENCES nombreTabla [(columna [,columna])]
                    [{ON DELETE CASCADE | ON DELETE SET NULL}] ]
                  [CHECK (condición)],
      …
);
```

➞ **Restricción de tabla:** en el comando de CREATE TABLE la restricción se define de manera independiente a los campos. Este tipo de restricción puede afectar a más de un campo.

```
CREATE TABLE nombreTabla
(
    Columna1 tipo_dato,
    Columna2 tipo_dato,
    …
    [CONSTRAINT nombre_restricción] {UNIQUE | PRIMARY KEY}
        [(columna [,columna])],

    [CONSTRAINT nombre_restricción] FOREIGN KEY [(columna [,columna])]
        REFERENCES nombreTabla [(columna [,columna])] [{ON DELETE
        CASCADE | ON DELETE SET NULL}],

    [CONSTRAINT nombre_restricción] CHECK (condición),
    …
);
```

- **Primary Key**

Esta restricción define la clave primaria de una tabla o relación.

Ejemplos:

| | Restricción de columna | Restricción de tabla |
|---|---|---|
| Primary Key | CREATE TABLE CLIENTE<br>(<br>NIF varchar2(9) primary key,<br>nombre varchar2(30),<br>apellido varchar2(30),<br>telefono number(9)<br>); | CREATE TABLE CLIENTE<br>(<br>NIF varchar2(9),<br>nombre varchar2(30),<br>apellido varchar2(30),<br>telefono number(9),<br>primary key (NIF)<br>); |

Ejemplo de clave primaria compuesta:

```
CREATE TABLE CLIENTE (
      NIF varchar2(9),
      nombre varchar2(30),
      apellido varchar2(30),
      telefono number(9),
      primary key (nombre, apellido, telefono)
  );
```

## Actividad. Primary key simple

a) Crea la siguiente tabla PROVIN, en la que se van a almacenar datos de provincias.

```
Create table provin
(
 Codigo number(2) primary key,
 Nombre varchar2(25)
);
```

b) Inserta los datos:

Madrid, 91
Barcelona, 93
Córdoba, 14
Girona, 17

c) Intenta insertar la provincia y analiza qué ocurre:

Gerona, 17

## Actividad. Primary key compuesta

a) Crea la tabla BLOQUESPISOS, en la que se van a almacenar datos de viviendas. Los campos son los siguientes.

La información que identifica de forma unívoca a un piso es la calle, el número, el piso y la puerta (campos en negrita).

| Nombre columna | Representa | Tipo |
|---|---|---|
| **CALLE** | Calle donde está el bloque | VARCHAR2(30) |
| **NUMERO** | Número donde está el bloque | NUMBER(3) |
| **PISO** | Número de planta | NUMBER(2) |
| **PUERTA** | Puerta | CHAR(1) |
| CODIGO_POSTAL | Código postal | NUMBER(5) |
| METROS | Metros de la vivienda | NUMBER(5) |
| COMENTARIOS | Otros datos de la vivienda | VARCHAR2(60) |

¿Qué formato de restricción has utilizado y por qué? (columna/tabla)

b) Inserta las siguientes pisos. Si hay algún problema interpreta el error.

| calle | Número | Piso | Puerta | CP | Metros | comentarios |
|---|---|---|---|---|---|---|
| Mayor | 25 | 3 | A | 28005 | 89 | Con balcón |
| Mayor | 25 | 3 | B | 28005 | 80 | Interior |
| Mayor | 25 | 3 | B | 28005 | 75 | Reformado |

c) ¿Cuántas claves primarias tiene esta tabla?

- **Foreign Key**

Esta restricción define la clave externa de una tabla o relación. Es una restricción de integridad.

Cuando se trata de una restricción de columna el formato es el siguiente. Observa que en este caso no se emplea la cláusula FOREIGN KEY:

```
Columna1 Tipo_dato [CONSTRAINT nombre_restricción] REFERENCES
nombreTabla [(columna [,columna])] {[ON DELETE CASCADE | ON DELETE SET
NULL] }]
```

Cuando se trata de una restricción de tabla el formato es:

```
[CONSTRAINT nombre_restricción] [FOREIGN KEY [(columna [,columna])]
REFERENCES nombreTabla [(columna [,columna])] {[ON DELETE CASCADE | ON
DELETE SET NULL] }]
```

Los tipos de borrado que se definen (ON DELETE) representan la **integridad referencial** de la base de datos, que se estudiará más adelante.

De momento no vamos a indicar el tipo de borrado (por defecto se considera borrado RESTRINGIDO). Ejemplos:

| | Restricción de columna | Restricción de tabla |
|---|---|---|
| Foreign Key | CREATE TABLE FACTURA<br>(<br>numero number(4) primary key,<br>precio number(6,2),<br>cliente varchar2(9) references<br>CLIENTE(nif),<br>fecha date<br>); | CREATE TABLE FACTURA<br>(<br>numero number(4) primary key,<br>precio number(6,2),<br>cliente varchar2(9),<br>fecha date,<br>foreign key (cliente) references CLIENTE(nif)<br>); |

Es importante remarcar **que el tipo de datos y dominio de un campo foreign key debe ser igual que el tipo de datos y dominio de la primary key a la que referencia**, ya que se trata del mismo dato.

En la referencia se indica la tabla y campo a la que se hace referencia. Si se omite el campo se considera la clave primaria de esa tabla. Ejemplo:

```
cliente varchar2(9) references CLIENTE
```

Al no indicar el campo al que se hace referencia en la tabla CLIENTE, se considera la clave primaria de cliente, es decir “dni”.

## Actividad. Foreign key simple

a) Crea la tabla PERSONAS1, en la que se van a almacenar datos de personas:

| Nombre de columna | Representa | Tipo |
|---|---|---|
| **NIF** | NIF de la persona | 9 carácteres alfanuméricos |
| Nombre | Nombre completo de la persona | Texto de 30 caracteres |
| población | Ciudad de residencia | Texto de 20 caracteres |
| **provincia** | Código de la provincia de residencia | Número de 2 dígitos |

Las personas se identifican por su NIF.

Las provincias deben estar previamente dadas de alta en la tabla PROVIN creada anteriormente. Define el campo “provincia” como foreign key que referencie a la tabla PROVIN. Utiliza el formato de restricción de columna.

b) Inserta las siguientes personas. Si hay algún problema interpreta el error.

49635201G, Pedro Martínez, Coslada, provincia de código 91 (Madrid)
56008932L, Sonia Gil, Guadalajara, provincia de código 19 (Guadalajara)

## Actividad. Foreign key compuesta

a) Crea la tabla PERSONAS2 con los siguientes campos:

| Nombre de columna | Representa | Tipo |
|---|---|---|
| **NIF** | NIF de la persona | 9 carácteres alfanuméricos |
| Nombre | Nombre completo de la persona | Texto de 30 caracteres |
| **calle** | Residencia: calle donde está el bloque de pisos | Varchar2(30) |
| **numero** | Residencia: número donde está el bloque | Number(3) |
| **piso** | Residencia: número de planta dentro del bloque | Number(2) |
| **puerta** | Residencia: puerta del piso | Char(1) |

Las personas se identifican por su NIF.

Los campos: calle, número, piso y puerta indican la vivienda de residencia de la persona. Esa vivienda debe estar previamente dada de alta en la tabla BLOQUESPISOS, creada anteriormente. Define la foreign key que referencie a la tabla BLOQUESPISOS.

¿Qué formato de restricción has utilizado y por qué? (columna/tabla)

b) Inserta las siguientes personas. Si hay algún problema interpreta el error.

| NIF | Nombre | calle | Número | Piso | Puerta |
|---|---|---|---|---|---|
| 49635201G | Pedro Martínez | Mayor | 25 | 3 | A |
| 56008932L | Sonia Gil | Mayor | 15 | 6 | B |

c) ¿Cuántas claves externas tiene esta tabla?

- **Not null**

Esta restricción define que el valor de un campo no puede tener el valor null, es decir, es un campo **obligatorio**. En este caso, el formato de definición debe ser siempre como restricción de columna.

Ejemplo:

| | Restricción de columna |
|---|---|
| Not null | `CREATE TABLE CLIENTE`<br>`(`<br>`NIF varchar2(9) primary key,`<br>`nombre varchar2(30) not null,`<br>`apellido varchar2(30),`<br>`telefono number(9)`<br>`);` |

## Actividad. Not null

a) Crea la tabla EJEMPLO_NOTNULL:

```
create table ejemplo_notnull (
  dni varchar2(9) primary key,
  nombre varchar2(30) not null,
  edad number(3)
);
```

b) Inserta el siguiente registro y analiza qué ocurre:

```
insert into ejemplo_notnull (dni, edad) values ('11111A', 53);
```

Ejercicios: crear tablas con relaciones (PK, FK y not null). Ejercicios 5, 6 y 7.

- **Unique**

Esta restricción define que el valor de un campo o de un conjunto de campos no puede repetirse (debe ser único).

Ejemplos:

| | Restricción de columna | Restricción de tabla |
|---|---|---|
| Unique | CREATE TABLE CLIENTE<br>(<br>NIF varchar2(9) primary key,<br>nombre varchar2(30),<br>apellido varchar2(30),<br>telefono number(9) unique<br>); | CREATE TABLE CLIENTE<br>(<br>NIF varchar2(9) primary key,<br>nombre varchar2(30),<br>apellido varchar2(30),<br>telefono number(9),<br>unique (telefono)<br>); |

Ejemplo de UNIQUE compuesto:

```
CREATE TABLE CLIENTE
(
      NIF varchar2(9) primary key,
      nombre varchar2(30),
      apellido varchar2(30),
      telefono number(9),
      unique (nombre,apellido)
);
```

## Actividad. Unique

a) Crea la tabla EJEMPLO_UNIQUE:

```
create table ejemplo_unique (
  dni varchar2(9) primary key,
  nombre varchar2(30) unique,
  Edad number(3)
);
```

b) Inserta los siguientes datos y analiza qué ocurre:

```
insert into ejemplo_unique values ('11111A', 'John', 53);
insert into ejemplo_unique values ('11111B', 'John', 22);
```

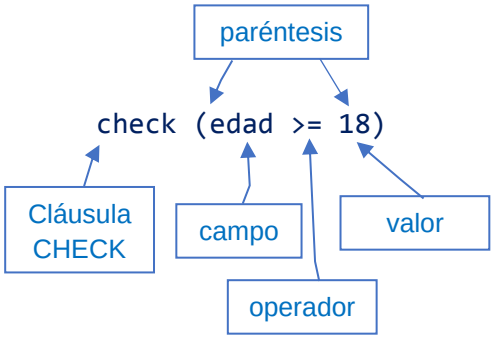
c) ¿Puede insertarse un registro con el nombre a valor NULL? ¿y varios registros con ese valor?

- **Check**

Esta restricción define que una validación para un campo. Esa validación se construye utilizando la cláusula CHECK, el campo que se quiere validar, un operador y un valor de comparación. Pueden anidarse condiciones con operadores lógicos.

Ejemplos

```
check (edad >= 18)
check (edad <> 18)
check (edad >= 18 and edad<=55)
check (edad between 20 and 30)
check (edad not in (10,20,30))
```




Ver operadores en el anexo.

**Solo puede realizarse la validación de un campo con CHECK** (no existe "check compuesto").

Ejemplos:

| | Restricción de columna | Restricción de tabla |
|---|---|---|
| | CREATE TABLE CLIENTE<br>(<br>NIF varchar2(9) primary key,<br>nombre varchar2(30),<br>apellido varchar2(30) check (apellido not like 'M%'),<br>telefono number(9)<br>); | CREATE TABLE CLIENTE<br>(<br>NIF varchar2(9) primary key,<br>nombre varchar2(30),<br>apellido varchar2(30),<br>telefono number(9),<br>check (apellido not like 'M%')<br>); |

## Actividad. Check

a) Crea la tabla EJEMPLO_CHECK:

```
create table ejemplo_check (
  nif varchar2(9) primary key,
  nombre varchar2(30) not null,
  edad number(3) check (edad between 12 and 18),
  curso number(1),
  check (curso in (1,2,3))
);
```

Observa que se han creado dos restricciones CHECK, una como restricción de columna y otra como restricción de tabla. En ambos casos se está definiendo un **dominio restringido**.

Inserta datos en la tabla violando las restricciones definidas, y comprueba el mansaje que aparece.

b) Intenta insertar un registro en la tabla en el que el campo “edad” o el campo “curso”, ambos con restricciones check, tomen el valor nulo (NULL). ¿es posible?

## Actividad. Crear tablas relacionadas con restricciones check

a) Crea las siguientes tablas relacionadas entre sí:

| TABLA FABRICANTES | | TABLA ARTICULOS | |
|---|---|---|---|
| | | articulo | varchar2(20) |
| cod_fabricante | number(3) | cod_fabricante | number(3) |
| nombre | varchar2(15) | peso | number(3) |
| país | varchar2(15) | categoria | varchar2(10) |
| | | precio_venta | number(6,2) |
| | | existencias | number(5) |

Restricciones para FABRICANTES:

- La clave primaria es cod_fabricante
- El país es un campo obligatorio

Restricciones para ARTICULOS:

- La clave primaria está formada por: artículo, cod_fabricante, peso y categoría
- Cod_fabricante es una clave externa que referencia a FABRICANTES
- Precio_venta y peso deben ser valores mayores de cero
- Categoría puede ser “Primera”, “Segunda” o “Tercera”.

b) Inserta al menos un registro en la tabla ARTICULOS (inventa los datos).

Ejercicios: crear tablas con restricciones. Ejercicios 8, 9 y 10.

- **Nombre de restricción**

Cada restricción tendrá un **nombre** asociado que puede ser configurado por el usuario al crear la restricción, o asignado automáticamente por el SGBD. Es decir, **indicar un nombre de restricción es opcional. Se puede indicar tanto en el formato de restricción de tabla como de columna.** En caso de indicarlo se emplea la cláusula CONSTRAINT con el formato:

```
CONSTRAINT nombre_restricción restricción
```

El nombre que se ponga para una constraint es elegido por el diseñador de la base de datos.

Normalmente se elige un literal que describa de alguna forma la restricción, por ejemplo "mayor_de_edad", "legal_age" o "edad_min_18" servirían para identificar la restricción de que un campo "edad" tenga que tener un valor de al menos 18 años:

```
constraint mayor_de_edad check (edad >= 18)
```

Es importante señalar que el nombre de una constraint **no puede repetirse en toda la base de datos**.

Ejemplos:

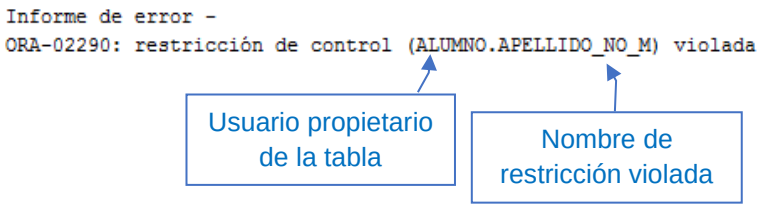
| | Restricción de columna | Restricción de tabla |
|---|---|---|
| Primary key | CREATE TABLE CLIENTE<br>(<br>NIF varchar2(9) constraint<br>clave_Cliente primary key,<br>nombre varchar2(30),<br>apellido varchar2(30),<br>telefono number(9)<br>); | CREATE TABLE CLIENTE<br>(<br>NIF varchar2(9),<br>nombre varchar2(30),<br>apellido varchar2(30),<br>telefono number(9),<br>constraint clave_Cliente primary key (NIF)<br>); |
| Foreign key | CREATE TABLE FACTURA<br>(<br>numero number(4) primary key,<br>precio number(6,2),<br>cliente varchar2(9) constraint<br>cliente_existente references<br>CLIENTE(nif),<br>fecha date<br>); | CREATE TABLE FACTURA<br>(<br>numero number(4) primary key,<br>precio number(6,2),<br>cliente varchar2(9),<br>fecha date,<br>constraint cliente_existente foreign key<br>(cliente) references CLIENTE(nif)<br>); |

| | | |
|---|---|---|
| Unique | CREATE TABLE CLIENTE<br>(<br>NIF varchar2(9) primary key,<br>nombre varchar2(30),<br>apellido varchar2(30),<br>telefono number(9) constraint<br>telefono_unico unique<br>); | CREATE TABLE CLIENTE<br>(<br>NIF varchar2(9) primary key,<br>nombre varchar2(30),<br>apellido varchar2(30),<br>telefono number(9),<br>constraint telefono_unico unique<br>(telefono)<br>); |
| Not null | CREATE TABLE CLIENTE<br>(<br>NIF varchar2(9) primary key,<br>nombre varchar2(30) constraint<br>nombre_oblig not null,<br>apellido varchar2(30),<br>telefono number(9)<br>); | |
| Check | CREATE TABLE CLIENTE<br>(<br>NIF varchar2(9) primary key,<br>nombre varchar2(30),<br>apellido varchar2(30) constraint<br>apellido_no_M check (apellido<br>not like 'M%'),<br>telefono number(9)<br>); | CREATE TABLE CLIENTE<br>(<br>NIF varchar2(9) primary key,<br>nombre varchar2(30),<br>apellido varchar2(30),<br>telefono number(9),<br>constraint apellido_no_M check (apellido<br>not like 'M%'),<br>); |

**¿Para qué sirve dar un nombre de restricción?**

Principalmente para identificar esa restricción y para que al intentar introducir un registro que viole la restricción, el mensaje de error sea más específico.

Es más fácil identificar una violación de restricción si se le da un nombre al definirla.




Nota: la restricción NOT NULL, aunque tenga nombre asignado, no lo mostrará.

## Actividad. Definir nombre de Constraint

a) Crea las siguientes tablas relacionadas entre sí:

```
create table provincias(
cod_provincia number(2) constraint pk_prov primary key,
nom_provincia varchar2(15));

create table personas(
dni number(8) constraint pk_per primary key,
nombre varchar2(15),
edad number(3) constraint mayor_edad check (edad >= 18),
provincia number(2),
constraint fk_prov foreign key (provincia) references
  provincias(cod_provincia));
```

b) Realiza las siguientes inserciones y analiza el mensaje de error que aparece:

```
insert into provincias values (28,'MADRID');
insert into provincias values (28,'SEVILLA');
```

c) Ejecuta el siguiente comando y analiza el mensaje de error que aparece:

```
insert into personas values (1133, 'Luis', 35, 22);
```

d) Ejecuta el siguiente comando y analiza el mensaje de error que aparece:

```
insert into personas values (1155, 'Ana', 15, 28);
```

La restricción NOT NULL, aunque tenga nombre asignado, no lo mostrará.

## Actividad. Definir nombre de Constraint en la restricción NOT NULL

a) Crea la siguiente tabla:

```
create table ejemplo_const_notnull(
nif varchar2(9) primary key,
nombre varchar2(15) not null,
edad number(3) constraint edad_obligat not null);
```

Observa que se han definido dos restricciones NOT NULL, una de ellas con nombre de constraint y otra sin él.

b) Realiza las siguientes inserciones y analiza el mensaje de error que aparece:

```
insert into ejemplo_const_notnull(nif,edad) values ('1111A',20);
insert into ejemplo_const_notnull(nif,nombre) values ('2222B', 'Jose');
```

- **Visualizar restricciones ya creadas**

¿Cómo podrían revisarse las restricciones en tablas ya creadas?

Existen vistas creadas por Oracle que contienen información sobre las restricciones definidas en las tablas:

- USER_CONSTRAINTS: restricciones en tablas propiedad del usuario. Entre otras cosas, se aprecia el tipo de restricción en el campo "constraint_type":

| | |
|---|---|
| C Restricción CHECK | R Restricción FOREIGN KEY (References) |
| P Restricción PRIMARY KEY | U Restricción UNIQUE |

- USER_CONS_COLUMNS: información sobre las restricciones, en especial se refleja el campo o campos a los que afectan.

Las vistas se manejan de forma similar a las tablas, para visualizar su contenido se emplea el comando SELECT.

En las vistas anteriores aparecen las restricciones de todas las tablas creadas por el usuario (cada registro es una restricción). Para filtrar los registros y solo visualizar los de una tabla en concreto, se pueden seleccionar aquellas filas cuyo campo "table_name" sea el nombre de la tabla deseada. El formato de la instrucción sería:

```
SELECT * FROM Nombre_vista WHERE table_name='NOMBRE_TABLA';
```

Nota: los nombres de los objetos internamente en Oracle se guradan con mayúsculas. En este caso el nombre de la tabla es un dato a buscar, por lo que al ser un texto, aparece entre comillas simples y escrito en mayúscula.

## Actividad. Visualizar constraints existentes

a) Visualiza las restricciones de la tabla EJEMPLO_CONST_NOTNULL creada en una actividad anterior. Para ello, ejecuta las siguientes instrucciones e interpreta su resultado. Presta atención a los **nombres de las constraints** que aparecen.

```
select * from user_constraints where table_name = 'EJEMPLO_CONST_NOTNULL';
select * from user_cons_columns where table_name = 'EJEMPLO_CONST_NOTNULL';
```

b) Visualiza las restricciones de la tabla PERSONAS creada en una actividad anterior. Presta especial atención a las **restricciones compuestas.**

Ejercicios: Crear tablas con nombre de restricción. Ejercicio 11.

## 2.5. Valor por defecto (Default)

Cuando se define una columna de una tabla, es posible asignarle un valor por defecto, de forma que si no se indicara un valor para esa columna se asignaría el valor por defecto automáticamente.

El formato es similar al de una restricción de columna, pero no puede indicarse nombre de constraint:

```
CREATE TABLE nombreTabla
(
  Columna1 tipo_dato DEFAULT valor,
  …
);
```

Ejemplo. Cuando no se conozca el nombre de un cliente la base de datos automáticamente guardará el texto “Desconocido” en la columna “nombre”:

```
CREATE TABLE CLIENTE
(
 NIF varchar2(9) primary key,
 nombre varchar2(30) default 'Desconocido',
 apellido varchar2(30),
 telefono number(9)
);
```

### Actividad. Default

a) Crea la siguiente tabla:

```
create table ejemplo_default(
nif varchar2(9) primary key,
nombre varchar2(15) not null,
edad number(3) default 18,
fecha_alta date default sysdate);
```

SYSDATE devuelve la fecha actual

b) Realiza las siguiente inserción y después comprueba el contenido de la tabla (SELECT):

```
insert into ejemplo_default(nif,nombre) values ('1111A', 'Isabel');
```

Ejercicios: Crear tablas con nombre de restricción y default. Ejercicio 12.

## 2.6. SQL. Borrado de tablas

```
TRUNCATE TABLE nombreTabla [{DROP | REUSE} STORAGE];
```

TRUNCATE TABLE borra el contenido de una tabla (solo los datos, no su estructura). Es similar a DELETE FROM nombreTabla, sin la cláusula WHERE (borrado de todos los datos de una tabla).

TRUNCATE es una sentencia DDL que suprime todas las filas de una tabla liberando el espacio para otros usos sin que desaparezca la definición de la tabla en la base de datos. No genera información de retroceso (no permite rollback) ni activa disparadores DELETE. Por todo ello, la eliminación de datos con TRUNCATE es más rápida que con DELETE (sentencia LMD).

DROP STORAGE libera todo el espacio (opción por defecto) mientras que REUSE STORAGE mantiene el espacio reservado para nuevas filas.

### Actividad. Truncado

a) En actividades anteriores se creó la tabla MATERIAL. Revisa la tabla con el comando DESCRIBE para ver su estructura, y comprueba si tiene datos (SELECT).

   Trunca la tabla y comprueba de nuevo su estructura y contenido.

b) En actividades anteriores se creó la tabla PROVIN. Revisa la tabla con el comando DESCRIBE para ver su estructura, y comprueba si tiene datos (SELECT).

   Trunca la tabla y comprueba de nuevo su estructura y contenido. Interpreta lo que ocurre.

```
DROP TABLE nombreTabla [cascade constraints];
```

DROP TABLE borra completamente una tabla (datos y objeto).

Cuando una tabla está referenciada por otra, no es posible borrar esa tabla sin violar restricciones de integridad.

Por ejemplo si se borrara la tabla CLIENTE, el campo “cliente” de la tabla FACTURA quedaría con referencias inexistentes. No es posible hacer ese borrado.

CLIENTE

| NIF | Nombre | Apellido | Teléfono |
|---|---|---|---|
| 452110B | Ana | Ruiz | 6325142 |
| 125852D | Eva | Casado | 6399632 |
| 588774A | Jaime | Vega | 6698523 |

FACTURA

| Número | Precio | cliente | fecha |
|---|---|---|---|
| 1003 | 175 | 125852D | 29/03/18 |
| 1004 | 66 | 125852D | 06/05/18 |
| 1005 | 325 | 452110B | 06/06/18 |

Cliente es clave externa en la tabla FACTURA, hace referencia a NIF de la tabla CLIENTE.

La solución es añadir la cláusula CASCADE CONSTRAINTS. En ese caso, antes de proceder al borrado de la tabla CLIENTE, la base de datos de forma automática borrará la restricción FOREIGN KEY que relaciona ambas tablas.

### Actividad. Borrado Cascade Constraints

En actividades anteriores se crearon las tablas PERSONAS2 y BLOQUESPISOS.

PERSONAS2 guarda información de personas, incluyendo su vivienda de residencia, que referencia a BLOQUESPISOS.

a) Para ambas tablas recopila e interpreta la siguiente información:

- Estructura de la tabla (DESCRIBE)
- Contenido (SELECT)
- Restricciones (USER_CONSTRAINTS y USER_CONS_COLUMNS)

b) Borra la tabla BLOQUESPISOS con el comando:

```
Drop table bloquespisos;
```

¿Qué ocurre?

c) Utiliza la cláusula CASCADE CONSTRAINTS para borrar la tabla. ¿Qué implica este borrado?

Para la tabla PERSONAS2, que no se ha borrado, recopila de nuevo la información del apartado a) y comprueba si han habido cambios.

## 2.7. SQL. Renombrar tablas

```
RENAME nombreAnterior TO nombreNuevo;
```

Este comando permite hacer un cambio en el nombre de una tabla.

### Actividad. Renombrar tabla

a) En una actividad anterior se creó la tabla MATERIAL.

Renombra la tabla al nombre MATERIALES mediante la siguiente instrucción:

```
Rename material to materiales;
```

b) Comprueba si el nombre se ha actualizado en las vistas de restricciones (USER_CONSTRAINTS y USER_CONS_COLUMNS).

Ejercicios: crear tablas. Renombrar, borrado y truncado. Ejercicio 13.

# 3. Vistas

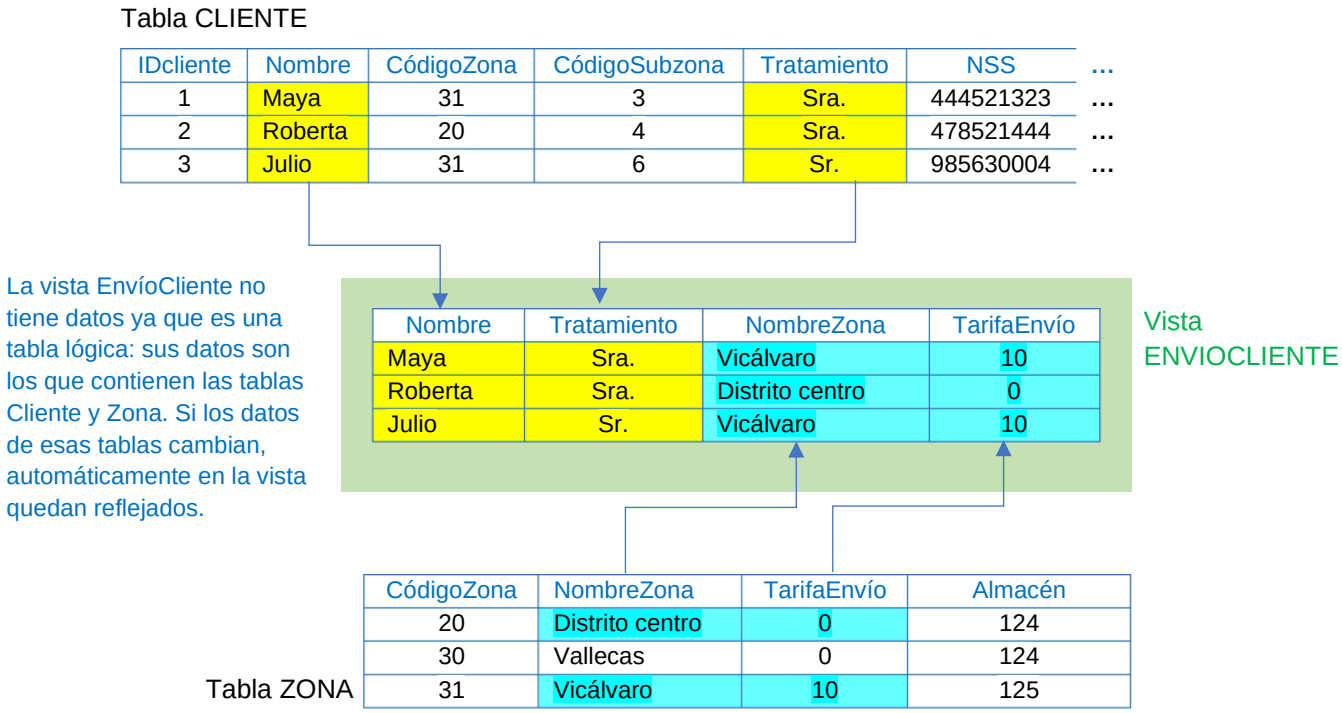
## 3.1. Definición de vista

Una vista es una **tabla lógica** que permite acceder a la información de una o varias tablas. No contiene información por sí misma, sino que su información está basada en lo que contienen otras tablas, llamadas **tablas base**, y siempre refleja los datos de estas tablas.

Permite simplificar el acceso a los datos: a través de una consulta simple permite obtener información que, de otro modo, hubiese requerido una consulta compleja.

Una vista tiene la misma estructura que una tabla: filas y columnas, y **permite las mismas operaciones LMD** (Lenguaje de Manipulación de Datos): seleccionar filas (SELECT), borrar filas (DELETE), actualizar filas (UPDATE), insertar filas (INSERT).

Ejemplo:



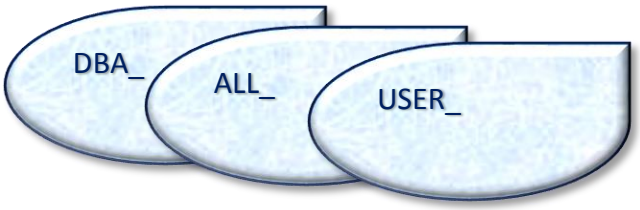

Se ha de respetar la lógica de las tablas base: por ejemplo, si se inserta una fila en la vista, se insertarán filas en las tablas base asociadas, pero debe cumplirse la condición de que se de valor a todos los campos obligatorios. Por otro lado, no se podrá insertar, modificar o actualizar filas en una vista creada con funciones de grupo (MAX, MIN, AVG…)

Si se suprime una tabla base, la vistas asociadas a ella se invalidan.

## 3.2. Vistas predefinidas

Los SGBD incluyen vistas que aportan información útil a los usuarios.

Dependiendo de los permisos del usuario se podrá acceder a unas vistas u otras. Por ejemplo, solo los usuarios administradores tienen acceso a las vistas con el prefijo ”DBA_”.




Los prefijos DBA_, ALL_ y USER_ permiten visualizar objetos de administración, de cualquier usuario y del usuario propio, respectivamente. Se podrá acceder en función de los privilegios que se tengan.

Ya se ha trabajado anteriormente con las vistas USER_CONSTRAINT y USER_CONS_COLUMNS para acceder a las restricciones de las tablas creadas por un usuario.

Las vistas **USER_OBJECTS** y **ALL_OBJECTS** muestran información resumida sobre cualquier objeto de base de datos (no se verán objetos de otros usuarios si no se tiene permiso).

Otras vistas: USER_INDEXES, USER_IND_COLUMNS, USER_VIEWS, USER_SOURCE, USER_VIEWS, USER_TABLES…

Listado completo de vistas predefinidas: https://docs.oracle.com/cd/B28359_01/nav/catalog_views.htm

### Actividad. Ver vistas predefinidas

Prueba las siguientes instrucciones:

Ver los distintos tipos de objetos que tiene creado nuestro usuario:

```
select distinct object_type from user_objects;

select * from user_tables; --Ver las tablas creadas por el usuario:
select count(*) from user_tables; --Ver número de tablas creadas por el usuario

select * from all_tables;  --Ver todas las tablas (a las que se tiene permiso)
select count(*) from all_tables; --Ver número de tablas

describe user_tables;       --Ver estructura de la vista
```

## 3.3. SQL. Creación, borrado y cambio de nombre de vistas

Se pueden crear vistas propias.

```
CREATE [OR REPLACE] VIEW nombreVista [(columna [,columna])]
AS consulta
[WITH {CHECK OPTION | READ ONLY} CONSTRAINT nombreRestricción];
```

[(columna [,columna])] son los nombres de las columnas que va a contener la vista. Si se omiten, tendrán el mismo nombre que las columnas devueltas por la consulta.

[OR REPLACE] crea de nuevo la vista si ya existía

[WITH CHECK OPTION] si se indica, el gestor realizará una comprobación cada vez que se haga un INSERT o UPDATE de la vista: revisará que los nuevos datos satisfagan los criterios de la consulta de la definición de la vista. Si no los satisfacen, el INSERT o UPDATE fallará.

[WITH READ ONLY] si se indica, sólo es posible hacer SELECT de la vista.

CONSTRAINT nombre Restricción da un nombre a la restricción WITH CHECK OPTION o WITH READ ONLY. Es opcional.

```
RENAME nombreAnterior TO nombreNuevo;

DROP VIEW nombreVista;
```

Ejemplos:

```
create view cliente_zona31
as select * from emple where codigozona=31;
```

Este tipo de consultas se verán más adelante

```
create view cliente_resumen
as select tratamiento, idcliente, nombre, nombrezona from cliente, zona
where cliente.codigozona=zona.codigozona;

rename cliente_zona31 to cliente_z31;

drop view cliente_z31;
```

# 4. Índices

## 4.1. Definición de índice

Un índice es un objeto de la base de datos que se asocia a una tabla, la cual indexa a través de una o varias columnas.

Proporciona un acceso rápido y directo a las filas de la tabla: es un elemento de **optimización de la base de datos**.

Crea una estructura de índice, que se cargará en memoria, para hacer búsquedas más rápidas. Contiene un elemento para cada valor que aparece en la columna o columnas indexadas de la tabla.

**Por defecto, cuando se crea una PRIMARY KEY o una restricción UNIQUE, se crea un índice para el campo o campos.** El nombre del índice será el nombre de la restricción. En la vista USER_CONSTRAINTS puede verse que la restricción tiene un índice asociado.

Por ejemplo:

```
create table prueba (
  num_ref number(2) constraint PK_PRUEBA primary key,
  valor number(5));
```

Al crear la tabla PRUEBA se creará automáticamente un índice de nombre PK_PRUEBA.

### Actividad. Visualizar índices de una tabla.

a) En actividades anteriores se creó la tabla EJEMPLO_CHECK. Revisa la tabla con el comando DESCRIBE para ver su estructura.

Revisa, en la vista USER_CONSTRAINTS, las columnas con el prefijo “INDEX”, y observa cuántos índices tiene esa tabla y sus nombres.

```
select * from user_constraints where table_name='EJEMPLO_CHECK';
```

b) Repite el apartado anterior para la tabla EJEMPLO_UNIQUE.

c) Accede a la vista **USER_INDEXES** y revisa os parámetros de los índices de la tabla EJEMPLO_UNIQUE.

## 4.2. Funcionamiento del índice

Si se hace una búsqueda en una tabla, a través de un campo no indexado (sin índice asociado), la búsqueda es secuencial, leyendo registro a registro.

En cambio, si el campo por el que se hace la búsqueda está indexado, se recorrerá la estructura de índice de forma mucho más óptima, obteniendo el ROWID (identificador de fila) y accediendo directamente al registro buscado.

**Un índice optimiza mucho las búsquedas, sin embargo consume muchos recursos**, por lo que no deben crearse índices de forma indiscriminada. Se aconseja:

- Indexar tablas con muchas filas
- No indexar columnas que sean modificadas (UPDATE) frecuentemente
- No indexar tablas en las que realicen manipulación de datos frecuentes (UPDATE, INSERT, DELETE), ya que cada cambio conlleva actualizar el índice.

**Ejemplo simplificado**: se tiene la tabla CLIENTES, cuya clave primaria es "num_cliente".

| ROWID | **num_cliente** | Empresa | Categoría | CIF | Localización | … |
|---|---|---|---|---|---|---|
| 1 | 125 | ELECTR.SA | A | 100002-A | Toledo | … |
| 2 | 126 | HOS-AMP | B | 360000-J | Logroño | … |
| 3 | 158 | TELEC.SL | A | 725000-X | Madrid | … |
| 4 | 201 | SINTREL | F | 450000-H | Teruel | … |
| 5 | 355 | TELCOS | E | **100010-G** | Toledo | … |
| 6 | 380 | DISP SA | A | 100000-R | Murcia | … |
| 7 | 388 | GESTI-LIMP | B | 100000-V | Tarragona | … |
| | …. | | | | | |

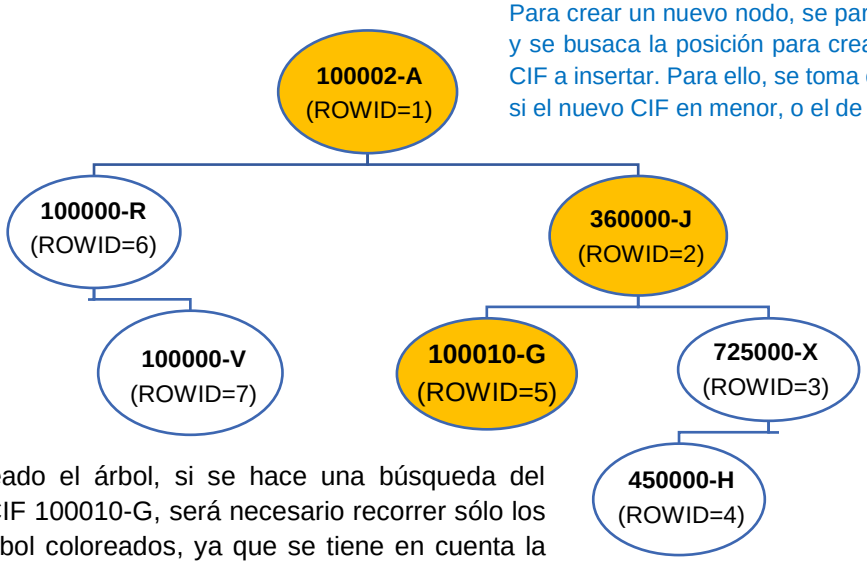
Si se hace una búsqueda del cliente con CIF 100010-G, será necesario recorrer las filas secuencialmente, desde la 1 hasta la 5, para encontrarlo. Eso son 5 accesos a la tabla. Si la tabla es grande el tiempo de búsqueda puede ser muy alto.

Por otro lado, si se crea un índice sobre el CIF, ya que se desea acelerar las búsquedas por ese campo, el gestor de base de datos implementará una estructura en memoria, a partir de ese campo. Existen distintas estrategias de indexación. A modo de ejemplo, se expone un árbol binario:

**Creación del árbol binario (simplificado):**

Cada nodo del árbol tendrá el dato del campo indexado (CIF) y el ROWID (identificador de fila en la que se encuentra ese dato en la tabla).

Para crear un nuevo nodo, se parte siempre del nodo raíz y se busaca la posición para crear el nodo para el nuevo CIF a insertar. Para ello, se toma el camino de la izquierda si el nuevo CIF en menor, o el de la derecha si es mayor.




Una vez creado el árbol, si se hace una búsqueda del cliente con CIF 100010-G, será necesario recorrer sólo los nodos del árbol coloreados, ya que se tiene en cuenta la estrategia (izquierda/derecha según si el CIF buscado es menor/mayor).

Una vez localizado el nodo del árbol, se lee el ROWID, en este caso el 5, y se accede directamente a la fila 5 de la tabla CLIENTES. **Sólo se ha requerido un acceso a la tabla.**

## 4.3. SQL. Creación y borrado de índices

Creación de un índice:

```
CREATE INDEX nombreÍndice
ON nombreTabla (columna[ASC|DESC][,columna[ASC|DESC]]…);
```

Es un formato simplificado. Se crea un índice para una o varias columnas de la tabla nombreTabla. Para cada columna se puede especificar el orden de indexación (ascendente o descendente).

Los índices que se crean manualmente no aparecen en la vista USER_CONSTRAINTS, ya que no están asociados a una restricción de la tabla. Todos los índices aparecen en la vista **USER_INDEXES** y **USER_IND_COLUMNS**.

Borrado de un índice:

```
DROP INDEX nombreÍndice;
```

Atributo UNIQUENESS de un índice:

- UNIQUE el campo no puede tener valores repetidos
- NONUNIQUE el valor del campo se puede repetir

### Actividad. Crear un índice simple.

a) Define la siguiente tabla:

```
create table EMPLEADOS (
codigo varchar2(5) primary key,
nombre varchar2(50),
apellidos varchar2(50),
sueldo number(4));
```

Crea un índice en el campo “nombre”:

```
create index indice_nombre on empleados(nombre);
```

b) Comprueba que se ha creado el índice en la vista USER_INDEXES. ¿Cuántos índices aparecen?

c) Comprueba que el índice creado manualmente no está en la vista USER_CONSTRAINTS, ya que no está asociado a ninguna restricción de la tabla. En esa vista solo debe aparecer el índice de la clave primaria.

### Actividad. Crear un índice compuesto.

a) Utiliza la tabla EMPLEADOS creada en la actividad anterior. Define un índice compuesto por dos campos: nombre y apellidos. El nombre del índice será “índice_nombre_completo”.

b) Comprueba que se ha creado el índice en la vista USER_INDEXES

c) ¿Cómo podría saberse a qué campo o campos está asociado un índice? Accede a la vista USER_IND_COLUMNS.

# 5. Anexos

## 5.1. Operadores

| Operadores aritméticos | |
|---|---|
| + | suma |
| - | resta |
| * | multiplicación |
| / | división |

| Operadores lógicos | |
|---|---|
| and | Devuelve TRUE si las dos condiciones son verdaderas |
| or | Devuelve TRUE si al menos una de las dos condiciones es verdadera |
| not | Devuelve TRUE si la condición es falsa y FALSE si es verdadera |

| Operadores de comparación | |
|---|---|
| = | Igual a |
| > | Mayor que |
| >= | Mayor o igual que |
| < | Menor que |
| <= | Menos o igual que |
| != <> | Distinto de |

| Operadores del valor null | |
|---|---|
| Is null | Comprueba si un valor es null |
| Is not null | Comprueba si un valor no es null (es distinto de null) |

| Operadores de comparación de cadenas de caracteres | |
|---|---|
| like | Compara con un texto |
| _ | Cualquier carácter |
| % | Cualquier cadena de caracteres |

Ejemplos. “valor” es un campo o una expresión:

valor LIKE ‘Hola’ se cumple sólo en el caso de que el valor sea ‘Hola’
valor LIKE ‘Tela_’ se cumple para valor: ‘Tela1’, ‘Tela2’, TelaX’, etc
valor LIKE ‘A%’ se cumple para cualquier valor que comience con A, por ejemplo: ‘Anastasia’, ‘Aleteo’, ‘Andrajoso’, ‘A123’

| Operadores de conjuntos de valores | |
|---|---|
| in | Comprueba si un valor pertenece a un conjunto de valores (lista) |
| Between and | Comprueba si un valor está en un rango de valores |

Ejemplos. “valor” es un campo o una expresión:

valor in (2, 3, 8) se cumple sólo en el caso de que valor sea 2, 3 u 8
valor between 3 and 8 se cumple para valor: 3, 4, 5, 6, 7 u 8
valor between ‘A’ and ‘C’ se cumple para valor: ‘A’, ‘B’ o ‘C’

## 5.2. Insertar y visualizar datos

Aunque los comandos LMD se verán más adelante, se expone aquí una introducción a los comandos de inserción y visualización de datos, con la finalidad de realizar algunos ejercicios.

- Insertar datos. Se indicarán valores para todas las columnas de la tabla, en el mismo orden en que fueron definidas:

```
INSERT INTO nombreTabla VALUES (valor1, valor2..);
```

Recuerda que los textos y fechas deben estar entre comillas simples

Ejemplo:

```
CREATE TABLE CLIENTE (
   NIF varchar2(9) primary key,
   nombre varchar2(30),
   apellido varchar2(30),
   telefono number(9)
);
insert into cliente values ('45896352K', 'Julián', 'Pérez', 625411489);
```

- Insertar datos. No se indican valores para todas las columnas de la tabla:

```
INSERT INTO nombreTabla (campo1, campo3,…) VALUES (valor1, valor3..);
```

Ejemplo:

```
insert into cliente (dni, apellido) values ('45896357N', 'Pérez');
```

- Visualizar todos los datos de una tabla:

```
SELECT * FROM nombreTabla;
```

## 5.3. Resumen de definición de restricciones

| Primary key | Sin nombre de restricción | Con nombre de restricción |
|---|---|---|
| Restricción de columna | CREATE TABLE CLIENTE<br>(<br>NIF varchar2(9) primary key,<br>nombre varchar2(30),<br>apellido varchar2(30),<br>telefono number(9)<br>); | CREATE TABLE CLIENTE<br>(<br>NIF varchar2(9) constraint clave_Cliente<br>primary key,<br>nombre varchar2(30),<br>apellido varchar2(30),<br>telefono number(9)<br>); |

| | | |
|---|---|---|
| Restricción de tabla | CREATE TABLE CLIENTE<br>(<br>NIF varchar2(9),<br>nombre varchar2(30),<br>apellido varchar2(30),<br>telefono number(9),<br>primary key (NIF)<br>); | CREATE TABLE CLIENTE<br>(<br>NIF varchar2(9),<br>nombre varchar2(30),<br>apellido varchar2(30),<br>telefono number(9),<br>constraint clave_Cliente primary key (NIF)<br>); |

| Foreign key | Sin nombre de restricción | Con nombre de restricción |
|---|---|---|
| Restricción de columna | CREATE TABLE FACTURA<br>(<br>numero number(4) primary key,<br>precio number(6,2),<br>cliente varchar2(9)<br>references CLIENTE(nif),<br>fecha date<br>); | CREATE TABLE FACTURA<br>(<br>numero number(4) primary key,<br>precio number(6,2),<br>cliente varchar2(9) constraint<br>cliente_existente references<br>CLIENTE(nif),<br>fecha date<br>); |
| Restricción de tabla | CREATE TABLE FACTURA<br>(<br>numero number(4) primary key,<br>precio number(6,2),<br>cliente varchar2(9),<br>fecha date,<br>foreign key (cliente)<br>references CLIENTE(nif)<br>); | CREATE TABLE FACTURA<br>(<br>numero number(4) primary key,<br>precio number(6,2),<br>cliente varchar2(9),<br>fecha date,<br>constraint cliente_existente foreign key<br>(cliente) references CLIENTE(nif)<br>); |

| unique | Sin nombre de restricción | Con nombre de restricción |
|---|---|---|

| | | |
|---|---|---|
| Restricción de columna | CREATE TABLE CLIENTE<br>(<br>NIF varchar2(9) primary key,<br>nombre varchar2(30),<br>apellido varchar2(30),<br>telefono number(9) unique<br>); | CREATE TABLE CLIENTE<br>(<br>NIF varchar2(9) primary key,<br>nombre varchar2(30),<br>apellido varchar2(30),<br>telefono number(9) constraint telefono_unico<br>unique<br>); |
| Restricción de tabla | CREATE TABLE CLIENTE<br>(<br>NIF varchar2(9) primary key,<br>nombre varchar2(30),<br>apellido varchar2(30),<br>telefono number(9),<br>unique (telefono)<br>); | CREATE TABLE CLIENTE<br>(<br>NIF varchar2(9) primary key,<br>nombre varchar2(30),<br>apellido varchar2(30),<br>telefono number(9),<br>constraint telefono_unico unique (telefono)<br>); |

not null

| | Sin nombre de restricción | Con nombre de restricción |
|---|---|---|
| Restricción de columna | CREATE TABLE CLIENTE<br>(<br>NIF varchar2(9) primary key,<br>nombre varchar2(30) not null,<br>apellido varchar2(30),<br>telefono number(9)<br>); | CREATE TABLE CLIENTE<br>(<br>NIF varchar2(9) primary key,<br>nombre varchar2(30) constraint nombre_oblig<br>not null,<br>apellido varchar2(30),<br>telefono number(9)<br>); |

check

| | Sin nombre de restricción | Con nombre de restricción |
|---|---|---|
| Restricción de columna | CREATE TABLE CLIENTE<br>(<br>NIF varchar2(9) primary key,<br>nombre varchar2(30),<br>apellido varchar2(30) check<br>(apellido not like 'M%'),<br>telefono number(9)<br>); | CREATE TABLE CLIENTE<br>(<br>NIF varchar2(9) primary key,<br>nombre varchar2(30),<br>apellido varchar2(30) constraint<br>apellido_no_M check (apellido not like<br>'M%'),<br>telefono number(9)<br>); |

| | | |
|---|---|---|
| Restricción de tabla | CREATE TABLE CLIENTE<br>(<br>NIF varchar2(9) primary key,<br>nombre varchar2(30),<br>apellido varchar2(30),<br>telefono number(9),<br>check (apellido not like 'M%')<br>); | CREATE TABLE CLIENTE<br>(<br>NIF varchar2(9) primary key,<br>nombre varchar2(30),<br>apellido varchar2(30),<br>telefono number(9),<br>constraint apellido_no_M check (apellido not like 'M%'),<br>); |