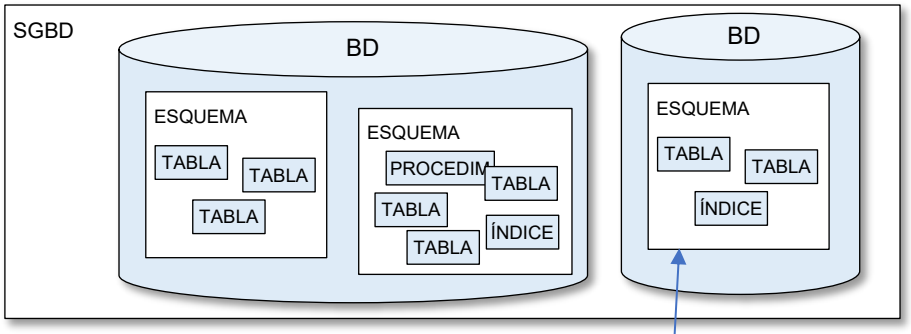
Crear una Base de Datos Relacional

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 puede 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.

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

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

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.

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

Para dar una longitud máxima a un texto se escribe el valor entre paréntesis, por ejemplo, varchar2(50) define un texto de 50 caracteres como máximo.

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

Actividad: crea tu primera tabla con varias columnas con tipos de datos char(12), varchar2(50) y date. Inventa el nombre de la tabla y los campos, intentando que tenga sentido.

Prueba a insertar varios registros (puedes ver los comandos en el cuadro de abajo). Recuerda que los campos de texto y fechas se escriben entre comillas simples.

➡ Comandos básicos LMD para manipular datos (se estudiarán más adelante, pero se muestra aquí una introducción mínima para poder realizar las actividades):

Insertar datos en una tabla (todos los campos del registro):

```
insert into nombre_tabla values (valor1, valor2…);
```

Insertar datos en una tabla (no todos los campos del registro):

```
insert into nombre_tabla(campo1,campo3…) values (valor1, valor3…);
```

Ver los registros ya insertados:

```
select * from nombre_tabla;
```

En el caso de los números, 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).

Actividad: crea una tabla con al menos una columna de tipo number, y define una escala y precisión. Prueba a instar varios registros con distintos valores y comprueba los errores y cómo quedan insertados (comando SELECT) siguiendo lo explicado arriba.

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.

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

a) Crea la tabla MATERIAL, que tiene como clave primaria el campo "código", con:

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

b) Inserta dos registros mediante los siguientes comandos (observa el formato):

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

c) Inserta dos registros mediante los siguientes comandos:

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

Comprueba el contenido de la tabla (comando SELECT * from material;).

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: primsry key, foreign key, unique, not null, check.

- **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. **Todas las tablas en el modelo relacional deben tener clave primaria, que es única.**

Ejemplos:

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

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. Crea una tabla con clave primaria e intenta insertar dos registros con el mismo valor de ese campo. Observa el error que aparece.

Actividad. Crea una tabla con una clave primaria compuesta por varias columnas.

a) Intenta insertar dos registros en los que se repita el valor de una de las columnas de la clave primaria, ¿qué ocurre?

b) Intenta insertar dos registros en los que todas las columnas de la clave primaria sean iguales, ¿qué ocurre?

- **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`
	`(`
	`numero number(4) primary key,`
	`precio number(6,2),`
	`cliente varchar2(9) references`
	`CLIENTE(nif),`
	`fecha date`
	`);`
		`CREATE TABLE FACTURA`
		`(`
		`numero number(4) primary key,`
		`precio number(6,2),`
		`cliente varchar2(9),`
		`fecha date,`
		`foreign key (cliente) references CLIENTE(nif)`
		`);`

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.

Actividad

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) Crea la tabla PERSONAS1, en la que se van a almacenar datos de personas. Las provincias de las personas 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.

Nombre de columna	Representa	Tipo
NIF	NIF de la persona	9 caracteres 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

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. Crea dos tablas relacionadas con una foreign key (FK), como en la actividad anterior. Inventa las tablas, con un ejemplo que tenga sentido. Pero la FK en este caso debe ser compuesta. **Para ello la PK debe ser compuesta también** y, por supuesto, declararse con la sintaxis de restricción de tabla.

Nota: cuando la PK es compuesta, la FK que la apunte debe ser única y compuesta. No es correcto referenciar mediante la constraint "foreign key" cada columna por separado.

- **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 (NIF varchar2(9) primary key, nombre varchar2(30) not null, apellido varchar2(30), telefono number(9));

Actividad. Crea una tabla en la que un campo sea declarado "not null" e inserta varios registros con esa columna a valor "null". Hay varias posibilidades sintácticas para ello. Recuerda que no es lo mismo valor "null" que un espacio en blanco.

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

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. Crea una tabla con un campo “unique”

a) Intenta insertar dos registros con el mismo valor de ese campo. Observa el error que aparece.

b) ¿Es posible insertar “null” en un campo definido como “unique”? Haz la prueba.

Actividad. Repite la actividad anterior creando una restricción “unique” compuesta de varias columnas.

a) Intenta insertar dos registros en los que se repita el valor de una de las columnas de la restricción “unique”, ¿qué ocurre?

b) Intenta insertar dos registros en los que todas las columnas de la restricción sean iguales, ¿qué ocurre?

- **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.

Ejemplos:

	Restricción de columna	Restricción de tabla

	`CREATE TABLE CLIENTE`
	`(`
	`NIF varchar2(9) primary key,`
	`nombre varchar2(30),`
	`apellido varchar2(30) check (apellido not like 'M%'),`
	`telefono number(9)`
	`);`
		`CREATE TABLE CLIENTE`
		`(`
		`NIF varchar2(9) primary key,`
		`nombre varchar2(30),`
		`apellido varchar2(30),`
		`telefono number(9),`
		`check (apellido not like 'M%')`
		`);`

Actividad. 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**.

a) 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?

- **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:

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

);
```

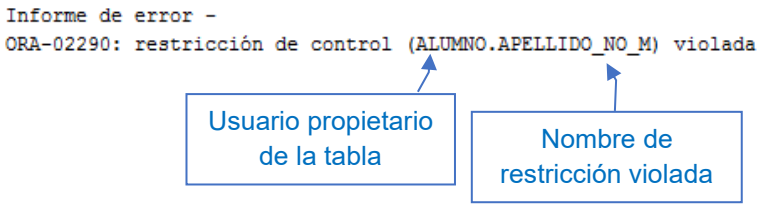
```
CREATE TABLE CLIENTE
(
NIF varchar2(9) primary key,
nombre varchar2(30),
apellido varchar2(30),
telefono number(9),
constraint telefono_unico unique (telefono)

);
```

¿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. Observa y 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) constraint con_nombre not null,,
edad number(3) constraint mayor_edad check (edad >= 18),
provincia number(2),
constraint fk_prov foreign key (provincia) references
  provincias(cod_provincia));
```

Inserta registros de modo que violen todas las restricciones definidas (de una en una) y observa los mensajes de error.

- **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 sobre los que están creados.

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. Visualiza las restricciones de alguna tabla que ya tengas creada. Elige una que tenga alguna restricción compuesta (si no tienes ninguna, créala). Comprueba como necesitas acceder a las dos vistas para obtener la información completa de la restricción.

Actividad. Las restricciones de tipo "R" son foreign key que apuntan a primary key. Busca, en la información de las vistas, cómo podrías saber, sin tener los comandos de creación de tablas, a quién está apuntando una foreign key concreta. Elige cualquier tabla con FK que tengas creada e intenta deducir a que tabla y columna/s está apuntando, solo consultando las vistas USER_CONSTRAINTS y USER_CONS_COLUMNS.

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. Crea una tabla con un campo que tenga un valor por defecto.

a) Prueba a insertar un registro sin indicar el valor de ese campo (null) y observa como automáticamente se inserta el valor por defecto. Recuerda que no es lo mismo un valor “null” que un espacio en blanco.

b) Inserta un registro de manera que no se especifique el valor de la columna con valor por defecto, pero forzando que se guarde un valor “null” (no un espacio en blanco).Tendrás que unas una sintaxis de comando algo diferente al apartado anterior.

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. Trunca alguna tabla de las que hemos usado en los ejemplos y actividades.

```
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.

Una posible 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. En una actividad anterior creamos las tablas PROVINCIAS y PERSONAS.

a) Para ambas tablas recopila la siguiente información:

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

b) Utiliza DROP para borrar la tabla PROVINCIAS. ¿Qué ocurre?

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

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

SQL. Renombrar tablas

```
RENAME nombreAnterior TO nombreNuevo;
```

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

Actividad. Renombra una tabla que hayas creado anteriormente. Después, comprueba en las vistas USER_CONSTRAINTS y USER_CONS_COLUMNS si ha cambiado el nombre de esa tabla.

Vistas

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).

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, las vistas asociadas a ella se invalidan.

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. Visualiza el contenido de algunas vistas predefinidas como user_objects, user_tables, all_tables… Puedes utilizar el usuario de trabajo y el usuario administrador, ya que es posible que haya diferencias entre lo que ve cada uno.

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;
```

Índices

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, al crear la tabla PRUEBA se creará automáticamente un índice de nombre PK_PRUEBA.:

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

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;`

Actividad. Busaca una tabla que tengas creada con las restricciones primary key e unique. Si no tienes ninguna, crea una tabla nueva.

a) Comprueba qué índices ya tiene creados esa tabla, accediendo a la vista USER_INDEXES. ¿Cómo podría saberse a qué campo o campos está asociado un índice? Accede a la vista USER_IND_COLUMNS.

b) Crea un nuevo índice sobre un campo de la tabla y accede a ambas vistas de nuevo (USER_INDEXES. Y USER_IND_COLUMNS).

c) Crea un nuevo índice compuesto sobre varios campos de la tabla y accede a ambas vistas de nuevo (USER_INDEXES. Y USER_IND_COLUMNS).

Anexo. 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’