3.7. Borrado de bases de datos

El comando DROP DATABASE es el utilizado para eliminar bases de datos. En MySQL, el comando se acompaña del nombre de la base de datos a eliminar:

```
>mysql -u root -p
drop database Proveedores;
```

En Oracle, no hay que especificarlo, tan solo es necesario conectarse a la instancia y escribir:

```
>sqlplus / as sysdba
shutdown abort; --parada de la instancia
startup mount exclusive restrict; --reinicio en modo exclusivo
drop database; --borrado
exit; --salir
```

Otra forma menos elegante de borrar una base de datos consiste en borrar físicamente todos los ficheros binarios de la base de datos.

3.8. Creación de tablas

Para crear tablas se usa el comando CREATE TABLE:

```
CREATE [TEMPORARY] TABLE [esquema.]nombre_tabla
    [(definición_create,...)]
    [opciones_tabla]

definición_create:
    definición_columna
  | [CONSTRAINT [símbolo]] PRIMARY KEY (nombre_columna,...)
  | [CONSTRAINT [símbolo]] FOREIGN KEY (nombre_columna,...)
      [definición_referencia]

definición_columna:
    nombre_columna tipo_datos [NOT NULL | NULL] [DEFAULT valor]
        [UNIQUE [KEY] | [PRIMARY] KEY]
        [definición_referencia]

definición_referencia:
    REFERENCES nombre_tabla [(nombre_columna,...)]
               [ON DELETE {CASCADE | SET NULL | NO ACTION} ]
               [ON UPDATE {CASCADE | SET NULL | NO ACTION} ]
```

El formato que aquí se presenta es un formato reducido, compatible con la mayoría de los gestores de bases de datos, para más información sobre un comando CREATE TABLE para un gestor en particular, se debe recurrir al manual de referencia del gestor.

La porción de sentencia *(definición_create,...)* especificará la definición de los campos que va a contener la tabla y sus restricciones. Los puntos suspensivos sugieren que *definición_create* se puede repetir tantas veces como sea necesario, por ejemplo si hay una tabla con 5 columnas, habrá que especificar 5 veces 'definición_create'.

El token opcional TEMPORARY se utiliza para crear tablas temporales (esto es, una tabla invisible al resto de usuarios y que se borrará en el momento de la desconexión del usuario que la creó). La primera cláusula que lleva la definición es *definición_columna*, donde se especificará la definición de un campo o columna de la tabla:

```
definición_columna:
    nombre_columna tipo_datos [NOT NULL | NULL] [DEFAULT valor]
        [UNIQUE [KEY] | [PRIMARY] KEY]
        [definición_referencia]
```

Se indica el nombre de columna y el tipo de datos, por ejemplo, el Codigo Postal podría ser un tipo NUMERIC(5,0) de MySQL, puesto que los códigos postales tienen 5 dígitos enteros y 0 decimales.

NOT NULL y NULL especifican que la columna admite o no admite valores nulos, es decir si un campo puede quedar sin valor, o con valor desconocido.

DEFAULT indica el valor por defecto que toma el campo si no es especificado de forma explícita. Por ejemplo, si se declara la siguiente columna:

```
CodigoPostal Numeric(5,0) NOT NULL DEFAULT 28941
```

Se indica que el campo Código Postal es un número de 5 dígitos enteros, que no admite valores nulos, y que en caso de no especificarlo, por ejemplo, en una inserción, el valor que tomará es 28941.

UNIQUE KEY especifica la creación de un índice, esto es, una estructura de datos que permite un acceso rápido a la información de una tabla y además, controla que no haya ningún valor repetido en el campo. Se producirá un error si se intenta insertar un elemento que ya coincida con un valor anterior. A efectos prácticos, la diferencia con una clave primaria es básicamente que un campo UNIQUE puede admitir valores NULL, mientras que un campo que se define como clave primaria no puede admitir nulos (debe ser NOT NULL).

PRIMARY KEY indica que la columna definida es la clave primaria. Una tabla tan solo puede tener una clave primaria. Si se especifica a nivel de *column_definition*, la clave solo puede tener un campo, si la clave se define a nivel de tabla, la clave puede ser multicolumna.

3.8.1. Implementación de restricciones

En la sintaxis de CREATE TABLE, definición_referencia sirve para crear una clave foránea. De esta manera, se enlaza el campo a su campo origen, es decir se crea una referencia:

Si existe la siguiente tabla creada:

```
create table clientes(
    dni varchar(9) PRIMARY KEY,
    nombre varchar(50),
    direccion varchar(60)
);
```

Se puede crear otra tabla 'mascotas' que contenga el registro de las mascotas de una tienda veterinaria y con un campo que haga referencia al dueño de la mascota:

```
create table mascotas(
    codigo integer PRIMARY KEY,
    nombre varchar(50),
    raza varchar(50),
    cliente varchar(9) REFERENCES clientes(dni)
);
```

Las opciones ON DELETE y ON UPDATE establecen el comportamiento del gestor en caso de que las filas de la tabla padre (es decir, la tabla referenciada) se borren o se actualicen. Los comportamientos pueden ser CASCADE, SET NULL y NO ACTION.

- Si se usa NO ACTION y se intenta un borrado o actualización sobre la tabla padre, la operación se impide, rechazando el borrado o la actualización.
- Si se especifica CASCADE, la operación se propaga en cascada a la tabla hija, es decir, si se actualiza la tabla padre, se actualizan los registros relacionados de la tabla hija, y si se borra un registro de la tabla padre, se borran aquellos registros de la tabla hija que estén referenciando al registro borrado.

- Si se indica SET NULL, se establece a NULL la clave foránea afectada por un borrado o modificación de la tabla padre.

```
create table mascotas(
    codigo integer PRIMARY KEY,
    nombre varchar(50),
    raza varchar(50),
    cliente varchar(9) REFERENCES clientes(dni)
      ON DELETE CASCADE ON UPDATE SET NULL
);
```

Si no se especifica ON DELETE u ON UPDATE por defecto se actúa como NO ACTION.

Oracle no implementa la opción ON UPDATE por lo que hay que recurrir a otros métodos para realizar las acciones de actualización, como por ejemplo, mediante TRIGGERS (disparadores).

La siguiente parte de la sintaxis de CREATE TABLE hace referencia a las declaraciones globales sobre la tabla, esto es, restricciones, claves primarias y foráneas compuestas, etc.

```
[CONSTRAINT [símbolo]] PRIMARY KEY (nombre_columna,...)
| [CONSTRAINT [símbolo]] FOREIGN KEY (nombre_columna,...)
    [definición_referencia]
```

En SQL, todas las restricciones pueden tener un nombre para facilitar su posterior referencia, por tanto cuando aparezca el opcional 'CONSTRAINT [símbolo]' se indica que la definición de una restricción tiene un nombre.

PRIMARY KEY sirve para especificar la creación de una clave primaria a nivel de tabla. Existe la opción de crearla a nivel de columna, mediante *definición_columna* o definirla aquí. Por ejemplo, es posible definir la siguiente tabla de dos formas:

```
#primera forma - nivel de columna
create table vehiculo(
    matricula varchar(7) primary key,
    marca varchar(20),
    modelo varchar(20),
    precio numeric(7,2)
);
```

```
#segunda forma - nivel de tabla
create table vehiculo(
    matricula varchar(7),
    marca varchar(20),
    modelo varchar(20),
    precio numeric(7,2)
    primary key (matricula)
);
```

Ambas formas son equivalentes, la diferencia es que mediante la segunda forma, es posible crear claves primarias compuestas por varios campos. Ejemplo:

```
#Clave primaria = dni + n_ss
create table empleado(
    dni varchar(9),
    n_ss varchar(15),
    nombre varchar(40),
    PRIMARY KEY (dni,n_ss) #compuesta
);
```

De forma análoga, también se puede crear claves foráneas a nivel de tabla, haciendo uso de:

```
[CONSTRAINT [symbol]] FOREIGN KEY (index_col_name,...)
   [reference_definition]
```

```
create table mascotas(
    codigo integer PRIMARY KEY,
    nombre varchar(50),
    raza varchar(50),
    cliente varchar(9),
    FOREIGN KEY (cliente) references clientes(dni)
);
```

◇ **Actividad 3.8**: Conéctate a MySQL y prueba a ejecutar los comandos CREATE TABLE anteriores. Posteriormente, conéctate a Oracle mediante SQL*Plus y ejecútalos de nuevo. ¿Son compatibles?

Para terminar, cada gestor de base de datos efectúa sus propias modificaciones al formato de la sintaxis create table. La cláusula *opciones_tabla* permite especificar las peculiaridades de cada gestor con respecto al almacenamiento en soporte físico de sus tablas. Además, cada gestor incorpora diversas características, por ejemplo, Oracle y DB2 implementan tipos de datos distintos a MySQL o a SQL Server y Access.

3.8.2. Tipos de Datos

Los tipos de datos que pueden usarse tanto para MySQL como para Oracle en la definición de una columna son los siguientes:

Tipo de dato	Naturaleza	Tamaño/formato	MySQL	Oracle
TINYINT [UNSIGNED]	Entero	1 byte	X	X
SMALLINT [UNSIGNED]	Entero	2 bytes	X	X
MEDIUMINT [UNSIGNED]	Entero	3 bytes	X	X
INT [UNSIGNED]	Entero	4 bytes	X	X
BIGINT [UNSIGNED]	Entero	8 bytes	X	X
INTEGER [UNSIGNED]	Entero	4 bytes	X	X
DOUBLE [UNSIGNED]	Real Aproximado	8 bytes	X	X
FLOAT [UNSIGNED]	Real Aproximado	4 bytes	X	X
DECIMAL(longitud,decimales)	Real Exacto	Variable	X	
NUMERIC(longitud,decimales)	Real Exacto	Variable	X	
NUMBER(longitud[,decimales])	Real Exacto	Variable		X
DATE	Fecha	'aaaa-mm-dd'	X	X
TIME	Hora	'hh:mm:ss'	X	
TIMESTAMP	Fecha y Hora	'aaaa-mm-dd hh:mm:ss'	X	X
DATETIME	Fecha y hora	'aaaa-mm-dd hh:mm:ss'	X	
CHAR(longitud)	caracteres	Longitud Fija	X	X
VARCHAR(longitud)	caracteres	Longitud Variable	X	X
VARCHAR2(longitud)	caracteres	Longitud Variable		X
BLOB	Objetos binarios	Longitud Variable	X	X
TEXT	Campos Memo	Longitud Variable	X	
CLOB	Campos Memo	Longitud Variable		X
ENUM(valor1,valor2,valor3...)	Enumeraciones	Lista de valores	X	
SET(valor1, valor2, valor3...)	Conjuntos	Conjuntos de valores	X	

Cuadro 3.1: Tipos de datos en MySQL y Oracle/DB2.

Puede verse que no todos los tipos de datos están en todos los gestores. Así por ejemplo, el tipo de datos ENUM no está disponible en Oracle y sí en MySQL. Se utiliza para crear enumeraciones, es decir, campos que admiten solo valores fijos, por ejemplo *Color ENUM('rojo','amarillo','verde')*. En Oracle, en su lugar, se puede implementar utilizando la directiva CHECK.

3.8.3. Características de la creación de tablas para MySQL

Las opciones de tabla para MySQL son las siguientes:

```
opciones_tabla: opción_tabla [opción_tabla] ...
opción_tabla:
    ENGINE = nombre_motor
  | AUTO_INCREMENT = valor
  | [DEFAULT] CHARACTER SET juego_caracteres [COLLATE colación]
  | CHECKSUM = {0 | 1}
  | COMMENT = 'string'
  | MAX_ROWS = valor
  | MIN_ROWS = valor
```

El almacenamiento físico de una tabla en MySQL está controlada por un software especial denominado *Motor de almacenamiento*. Mediante la opción 'ENGINE= nombre_motor' se indica el motor de almacenamiento para la tabla. Puede ser, entre otros, *innodb*, que son tablas transaccionales con bloqueo de registro y claves foráneas, *myIsam* por defecto usado por MySQL y que genera tablas operadas a gran velocidad, pero sin control de integridad referencial y *Memory*, que genera tablas que están almacenadas en memoria en lugar de un archivo físico.

La opción AUTO_INCREMENT permite indicar el valor inicial para campos de tipo AUTO_INCREMENT. AUTO_INCREMENT indica que ese campo es autoincrementado después de cada inserción. Un campo definido como auto_increment debe ser numérico; de esta manera, cuando en ese campo se indica el valor NULL en una inserción, el campo toma automáticamente el último valor incrementado en una unidad. Este campo es muy útil para los campos *código* donde se especifican valores clave autogenerados. En MySQL para definir un campo AUTO_INCREMENT se especifica la palabra clave justo a continuación del tipo de datos. Para simular este comportamiento en Oracle, se utiliza una secuencia (SEQUENCE).

'[DEFAULT] CHARACTER SET' Especifica el conjunto de caracteres para la tabla y COLLATE define la colación por defecto de la tabla. Si se especifica CHECKSUM, MySQL mantiene una suma de verificación para todos los registros. El comando CHECKSUM TABLE muestra esa suma de verificación (solo para motores de almacenamiento MyISAM).

COMMENT es un comentario para la tabla, hasta 60 caracteres. También es posible crear comentarios para cada una de las columnas. Mediante el token COMMENT

se puede documentar el diccionario de datos. MySQL dispone de este token para comentar no solo tablas, sino también columnas.

MAX_ROWS es el máximo número de registros que se quiere almacenar en la tabla. No es un límite absoluto, sino un indicador que la tabla debe ser capaz de almacenar al menos estos registros. MIN_ROWS es el mínimo número de registros que se planea almacenar en la tabla.

Por último, al igual que en la creación de base de datos, MySQL dispone de la cláusula *IF NOT EXISTS*, para que solamente se cree la tabla si no está creada previamente.

```
#Ejemplo de creación de tabla en MySQL
        create table if not exists Pedido(
           codigo int auto_increment primary key,
           fecha datetime,
           estado enum('Pendiente','Entregado','Rechazado')
        )
        comment = 'tabla de pedidos a proveedores'
        auto_increment = 10000
        max_rows=1000000
        checksum=1
        engine=innodb;
```

3.8.4. Características de la creación de tablas para Oracle

En Oracle, la mayoría de las opciones tienen que ver con su almacenamiento físico. Por ejemplo, las tablas deben ser almacenadas en un contenedor llamado *tablespace* (Espacio de tablas). Por defecto, si no se indican opciones de almacenamiento, la tabla se ubica en el tablespace del usuario, pero si se quiere ubicar en otro tablespace, se puede incluir la opción *tablespace nombre* para designar otro tablespace. Además, se puede definir cómo se reserva el espacio en disco para controlar el crecimiento desmedido del tamaño de una tabla mediante la cláusula storage. Estas y muchas otras opciones, son propias de Oracle. Por último, cabe destacar la capacidad de Oracle de permitir la creación de restricciones de tipo CHECK, que permite validar el valor de un campo mediante una expresión.

```
--Ejemplo de creación de tablas con opciones propias de Oracle
        create table Pedido(
           codigo integer primary key,
           fecha date,
           estado varchar(10),
```

```
    constraint c_estado
        check (estado IN ('Pendiente','Entregado','Rechazado'))
)
tablespace Administracion
storage (initial 100k next 100k minextents 1
         maxextents unlimited pctincrease 0);
```

3.8.5. Consulta de las tablas de una base de datos

Para consultar las tablas disponibles de una base de datos en MySQL, se utiliza el comando SHOW TABLES

```
mysql> show tables;
+----------------------+
| Tables_in_jardineria |
+----------------------+
| Clientes             |
| DetallePedidos       |
| ...                  |
| Productos            |
+----------------------+
```

En Oracle, se pueden consultar las vistas user_tables, dba_tables y all_tables. En Oracle, las vistas que comienzan por user_, aportan información sobre los objetos que posee el usuario conectado. Las vistas que comienzan por dba_ son solo accesibles por los administradores de bases de datos y muestran información sobre todos los objetos. Finalmente, las vistas que comienzan por all_ muestran la información sobre los objetos a los que el usuario tiene acceso, sean suyos o no.

```
SQL> select table_name from user_tables;
TABLE_NAME
------------------------------
PARTIDOS
ESTADISTICAS
JUGADORES
EQUIPOS
```

3.8.6. Consulta de la estructura de una tabla

Para conocer la estructura de una tabla ya creada es posible utilizar el comando

```
DESCRIBE [esquema.]nombre_tabla
```

Este comando muestra un listado con las columnas de la tabla, aportando información sobre los tipos de datos, restricciones, etc.

```
SQL> describe nba.equipos;
 Nombre                                    Nulo     Tipo
 ----------------------------------------- -------- ------------
 NOMBRE                                    NOT NULL VARCHAR2(20)
 CIUDAD                                             VARCHAR2(20)
 CONFERENCIA                                        VARCHAR2(4)
 DIVISION                                           VARCHAR2(9)
```

3.9. Modificación de tablas

El comando para modificar una tabla es ALTER TABLE y su sintaxis también es bastante compleja:

```
ALTER TABLE nombre_tabla
    especificación_alter [, especificación_alter] ...

especificación_alter:
    ADD definición_columna [FIRST | AFTER nombre_columna ]
  | ADD (definición_columna,...)
  | ADD [CONSTRAINT [símbolo]]
        PRIMARY KEY (nombre_columna,...)
  | ADD [CONSTRAINT [símbolo]]
        UNIQUE (nombre_columna,...)
  | ADD [CONSTRAINT [símbolo]]
        FOREIGN KEY (nombre_columna,...)
        [definición_referencia]
  | CHANGE [COLUMN] anterior_nombre_columna definición_columna
        [FIRST|AFTER nombre_columna]
  | RENAME COLUMN anterior_nombre_columna TO nuevo_nombre_columna
  | MODIFY definición_columna [FIRST | AFTER nombre_columna]
  | DROP COLUMN nombre_columna
  | DROP PRIMARY KEY
  | DROP FOREIGN KEY fk_símbolo
  | opciones_tabla
```

Al ver esta sintaxis se puede caer en la tentación de pensar que es mejor borrar una tabla y volver a crearla haciendo las modificaciones oportunas. Esto es posible si la tabla no tiene datos, pero... ¿qué ocurre si hay un centenar o un millar de registros? No queda otra opción que ejecutar una sentencia ALTER TABLE para evitar la pérdida de datos.

- La opción ADD permite añadir una columna, se puede especificar el lugar donde se va a insertar mediante las cláusulas AFTER (después de una columna) y FIRST (la primera columna). Oracle no admite las cláusulas AFTER y FIRST.

- Con la opción MODIFY se cambia el tipo de datos de una columna y se añaden restricciones.

- Con la opción DROP se pueden eliminar las restricciones de claves foráneas y primarias, dejando el tipo de dato y su contenido intacto.

- Las opciones de tabla, al igual que en CREATE TABLE varían con cada SGBD, y sirven principalmente para modificar las características del almacenamiento físico.

- Para cambiar el nombre de una columna, Oracle usa la cláusula RENAME y MySQL la cláusula CHANGE.

Por ejemplo, en MySQL, para añadir a la tabla mascotas el campo especie después del campo Raza, habría que ejecutar la siguiente instrucción:

```
ALTER TABLE Mascotas ADD Especie VARCHAR(10) AFTER Raza;
```

Para eliminar la columna con clave primaria *CodigoCliente* de la tabla *Clientes* en Oracle y establecer como clave primaria el campo *NIF* , hay que ejecutar las siguientes sentencias:

```
ALTER TABLE Clientes DROP PRIMARY KEY;
ALTER TABLE Clientes DROP CodigoCliente;
ALTER TABLE Clientes ADD COLUMN Nif VARCHAR(10) PRIMARY KEY FIRST ;
```

3.10. Borrado de tablas

El formato de instrucción en MySQL para el borrado de una tabla es muy sencillo:

```
DROP [TEMPORARY] TABLE
    tbl_name [, tbl_name] ...
```

Ejemplo:

```
DROP TABLE Mascotas;
DROP TABLE Clientes, Empleados;
```

En Oracle, cambia un poco pero es muy similar:

```
DROP [TEMPORARY] TABLE tbl_name [CASCADE CONSTRAINT]
```

No se permite el borrado de varias tablas en la misma sentencia, y, de forma opcional, CASCADE CONSTRAINT exige a Oracle que elimine las claves foráneas de las tablas relacionadas (en cascada).

```
--Elimina la tabla Partidos
    DROP TABLE Partidos;
--Elimina la tabla Jugadores y
--borra las claves foráneas de otras tablas
    DROP TABLE Jugadores CASCADE CONSTRAINT;
```

3.11. Renombrado de tablas

Para renombrar una tabla en MySQL se usa el comando RENAME TABLE:

```
RENAME TABLE nombre_tabla TO nuevo_nombre_tabla
    [, nombre_tabla TO nuevo_nombre_tabla] ...
```

Ejemplo:

```
RENAME TABLE Mascotas TO Animales;
```

En Oracle, no hay que indicar el token TABLE, basta con poner:

```
RENAME Jugadores TO Baloncestistas;
```