Manual SQL

Iker Martín

ÍNDICE

1.- Tipos de datos

Numéricos

INT(n) ➔ Números enteros

FLOAT(n,n) ➔ Números con decimales

Ejemplo: FLOAT(6,2) ➔ 1002,25

Fecha y hora

DATE ➔ aaaa-mm-dd (año, mes, día)

CURDATE() ➔ Fecha actual

TIME ➔ hh:mm:ss (horas, minutos, segundos)

DATETIME ➔ DATE + TIME

YEAR ➔ aaaa (año)

YEAR(CURDATE()) ➔ Año actual

Caracteres

CHAR(n) ➔ Números y letras (estáticos)

VARCHAR(n) ➔ Números y letras (dinámicos)

ENUM ➔ Valores especificados

Ejemplo: ENUM (‘valor1’, ‘valor2’, ‘valor3’, ‘valor4’)

2.- Tratamiento de datos

2.1.- Añadir datos

Insertar datos, especificando campos concretos.

```
INSERT INTO tabla1 (campo1, campo3) VALUES
        (123, 'hola'),
        (456, 'adiós');
```

Sin especificar campos, siempre y cuando se rellenen los obligatorios.

```
INSERT INTO tabla1 VALUES
        (123, 54.2, 'hola'),
        (456, 12.02, 'adiós');
```

Insertar datos obtenidos a través de una consulta.

```
INSERT INTO tabla1
        SELECT * FROM tabla2 WHERE…;
```

2.2.- Borrar datos

Borrar todos los datos.

```
DELETE FROM tabla1
```

Borrar únicamente los que cumplan una o varias condiciones.

```
DELETE FROM tabla1
        WHERE campo1 LIKE 'Hola';
```

2.3.- Modificar datos

Cambiar valor de uno o varios campos.

```
UPDATE tabla1 SET campo1 = 'Hola y adiós'
        WHERE campo2 = 123;
```

```
UPDATE tabla1 SET campo1 = 'Hola y adiós', campo2 = 123;
```

3.- Consultas simples

Estructura de consultas.

```
SELECT + FROM + WHERE (AND, OR, etc);
```

```
SELECT campo1, campo2 AS 'nombre_campo2' FROM tabla1;
        WHERE campo1 LIKE '_A%'    / NOT LIKE…
        WHERE campo1 > 2
        WHERE campo1 IS NULL       / IS NOT NULL…
        WHERE YEAR(campo2) > 1996
        LIMIT 1 ➔ Número de resultados (filas)
```

4.- Consultas con varios valores

```
        WHERE campo1 = 'HOLA' OR campo1 = 'ADIOS'
        WHERE campo1 IN ('HOLA', 'ADIOS')          / NOT IN…
        WHERE campo1 BETWEEN 1 AND 5               /NOT BETWEEN…
```

5.- Consultas con tablas relacionadas

```
SELECT e1.nombre, e2.nombre, e2.apellido
        FROM tabla1 e1, tabla2 e2
        WHERE e1.nombre = e2.nombre;
```

```
SELECT e1.nombre, e2.nombre, e2.apellido
        FROM tabla1 AS e1
        JOIN tabla2 AS e2 USING (nombre);
```

```
SELECT tabla1.nombre, tabla2.nombre, tabla3.nombre
        FROM tabla1
        JOIN tabla2 USING (nombre)
        JOIN tabla3 USING (nombre);
```

```
SELECT tabla1.nombre, tabla2.nombre, tabla2.apellido
        FROM tabla1 AS e1
        INNER JOIN tabla2 ON tabla1.nombre = tabla2.nombre;
```

6.- Ordenar por campo

SELECT + FROM + WHERE + **ORDER BY** campo1 + **DESC**, **ASC**;

ASC ➔ Arriba 1 y debajo 5, por ejemplo.

DESC ➔Arriba 5 y debajo 1, por ejemplo.

7.- Contar filas y cálculos

Contar todas las filas.

```
SELECT COUNT(*) FROM tabla1;
```

Contar cuantas comisiones existen, por ejemplo.

```
COUNT(comisión)
```

Máximo / mínimo

```
SELECT MAX(salario)
SELECT MIN(salario)
```

Suma total de valores.

```
SELECT SUM(salario)
```

Media de todos los valores.

```
SELECT AVG(salario)
```

Obtener la posición del valor 2, en este caso.

```
SELECT INSTR(salario, 2)
```

Juntar o concatenar valores en un único campo.

```
SELECT CONCAT(nombre, ' ', apellidos)
```

Mostrar cálculo en la propia consulta.

```
SELECT comisión, salario, (comisión*100/salario) AS 'CalculoMedia' FROM …
```

Redondear el resultado.

```
ROUND(comisión*100/salario)
```

8.- Agrupación

Estructura de consulta con agrupación, condición, ordenación…

```
SELECT + FROM + WHERE + GROUP BY + HAVING + ORDER BY;
```

Agrupar valores por cada campo.

```
SELECT color, AVG(campo1), COUNT(*)
        FROM tabla1
        GROUP BY color;
```

Resultado:

Rojo	1	5
Verde	1	4

Añadir condición a la propia agrupación.

```
        GROUP BY campo1 HAVING COUNT(campo2) > 15;
```

9.- Subconsultas

Consultas dentro de otras consultas.

```
SELECT + FROM + WHERE campo1 = (SELECT ...);
```

10.- Procedimientos

Los procedimientos son un conjunto de instrucciones que se ejecutan cuando se hace la llamada **CALL**.

10.1.- Creación

```
DELIMITER // (opcional)
CREATE PROCEDURE procedimiento1 ()
BEGIN
    SELECT campo1 FROM tabla1
        WHERE campo1 LIKE (SELECT MIN(campo1) FROM tabla1);
    SELECT CONCAT ('Esto es un texto para mostrar') TítuloDelMensaje;
END;//
```

```
CREATE PROCEDURE procedimiento2 (parámetro1 VARCHAR(30), parámetro2 DATE)
BEGIN
    INSERT INTO tabla1 VALUES (parámetro1, parámetro2);
    SELECT CONCAT ('Dato 1:', parámetro1, 'Dato2: ', parámetro2) Mensaje;
END;
```

10.2.- Llamada o ejecución

```
CALL procedimiento1();
```

```
CALL procedimiento2('texto1', '2015-04-01');
```

```
CALL procedimiento3('texto', 'fecha', 5.90, 7, CURRENT_DATE);
```

11.- Funciones

Las funciones son un conjunto de instrucciones que se ejecutan cuando se hace la llamada **SELECT** y siempre devuelve un valor.

11.1.- Creación

```
CREATE FUNCTION función1 () RETURNS INT(3)
BEGIN
DECLARE variable1 INT(3);
SELECT campo1 INTO variable1 FROM tabla1
        WHERE campo2 LIKE …
RETURN variable1;
END;
```

```
CREATE FUNCTION función2 (param1 CHAR(2), param2 INT(2)) RETURNS INT(3)
BEGIN
RETURN (SELECT campo1 INTO variable1 FROM tabla1
        WHERE campo2 LIKE parámetro1 AND campo3 = parámetro2);
END;
```

11.2.- Llamada o ejecución

```
SELECT función1 () DescripciónDeLaFunción;
```

```
SELECT función2 ('p1', CURRENT_DATE) Descripción;
```

12.- Variables

Declarar las variables.

```
DECLARE variable1 FLOAT(6,2);
DECLARE precio FLOAT(6,2) NOT NULL DEFAULT 5.0;
SELECT ... FROM ... WHERE campo1 LIKE (variable1);
SELECT campo1 INTO variable1 FROM tabla1
        WHERE campo2 LIKE …
RETURN variable1;
```

Dar valor a cada variable.

```
SET variable1 = 1.55;
SET variable2 = 'Texto';
SET variable3 = '2015-01-02';
```

13.- Estructuras de control

Condición IF

```
IF variable1 > 2 THEN
        SELECT, UPDATE, etc.
END IF;
```

Condición IF ELSE

```
IF variable1 > 2 THEN
        SELECT, UPDATE, etc.
ELSE
        SELECT, UPDATE, etc.
END IF;
```

Condición IF ELSEIF

```
IF variable1>2 THEN
        SELECT, UPDATE, etc.
ELSEIF variable1=3 THEN
        SELECT, UPDATE, etc.
END IF;
```

14.- Excepciones

Error 1062 - Entrada duplicada.

```
DECLARE encontrado BOOL DEFAULT 1;

DECLARE CONTINUE HANDLER FOR 1062 SET encontrado = 0

INSERT ….

IF encontrado = 0 THEN ➔ Está duplicado
```

Error 1329 - Cero filas encontradas.

```
DECLARE desconocida BOOL DEFAULT 1;

DECLARE CONTINUE HANDLER FOR 1329 SET desconocida = 0;

SELECT dato1 INTO numero FROM tabla1;

IF desconocida = 0 THEN ➔ No existe
```

Error 1451 - No se puede modificar o eliminar una fila padre.

Error 1452 - No se puede modificar o insertar una fila hija.

```
DECLARE noborrar BOOL DEFAULT 0;

DECLARE CONTINUE HANDLER FOR 1451 SET noborrar = 1;

DELETE …

IF noborrar = 0 THEN ➔ No se puede borrar
```

15.- Cursores

Almacenan los resultados de una consulta para revisar fila por fila.

CREATE PROCEDURE

BEGIN

DECLARE campo1;

DECLARE campo2;

DECLARE fin BOOL DEFAULT 0;

DECLARE c **CURSOR** FOR **SELECT dato1, dato2 FROM …**

DECLARE CONTINUE HANDLER FOR NOT FOUND SET fin = 1;

OPEN c; ➔**Inicia el cursor**

FETCH c INTO campo1, campo2; ➔**Obtiene el primer registro, en este caso**

WHILE fin = 0 **DO**

    SELECT CONCAT(campo1, campo2) Mensaje;

    FETCH c INTO campo1, campo2; ➔**Obtiene el siguiente registro, en este caso**

END WHILE;

CLOSE c; ➔**Cierra el cursor**

16.- Usuarios

16.1.- Crear

CREATE USER nombre

IDENTIFIED BY 'contraseña'

DEFAULT ROLE rol

PASSWORD EXPIRE DEFAULT (NEVER, INTERVAL n DAY)

WITH MAX_QUERIES_PER_HOUR n ➔**Número de consultas y actualizaciones que puede realizar cada hora.**

WITH MAX_UPDATES_PER_HOUR n ➔**Número de actualizaciones que puede realizar cada hora.**

WITH MAX_CONNECTIONS_PER_HOUR n ➔**Número de conexiones que puede realizar el usuario como máximo cada hora.**

WITH MAX_USER_CONNECTIONS n ➔**Número máximo de conexiones simultáneas.**

16.2.- Modificar

ALTER USER nombre

WITH MAX….

PASSWORD EXPIRE DEFAULT / NEVER, INTERVAL n DAY

16.3.- Asignar contraseña

SET PASSWORD FOR nombre = '123456';

SET PASSWORD FOR nombre = 'nueva' **REPLACE** 'contraseña_actual';

16.4.- Cambiar nombre

```
RENAME USER nombre@localhost TO nombre_nuevo@localhost;

RENAME USER nombre_antiguo1 TO nombre_nuevo1, nombre_antiguo2 TO nombre_nuevo2;
```

16.5.- Borrar

```
DROP USER nombre_usuario;
```

17.- Permisos

17.1.- Tipos

ALL [PRIVILEGES]	Todos los permisos excepto GRANT OPTION
ALTER	Permite el uso de ALTER TABLE
ALTER ROUTINE	Permite modificar o borrar rutinas almacenadas (procedimientos y funciones).
CREATE	Permite el uso de CREATE TABLE
CREATE ROUTINE	Permite crear rutinas almacenadas
CREATE TABLESPACE	Permite crear tablespaces, alterarlos y borrarlos.
CREATE TEMPORARY TABLES	Permite el uso de CREATE TEMPORARY TABLE
CREATE USER	Permite el uso de CREATE USER, DROP USER, RENAME USER y REVOKE ALL PRIVILEGES
CREATE VIEW	Permite el uso de CREATE VIEW
DELETE	Permite el uso de DELETE
DROP	Permite el uso de DROP TABLE
EVENT	Permite crear, modificar, eliminar y visualizar eventos.
EXECUTE	Permite ejecutar rutinas almacenadas
FILE	Permite el uso de SELECT … INTO OUTFILE y LOAD DATA INFILE
GRANT OPTION	Permite conceder o retirar a otros usuarios los privilegios poseídos.
INDEX	Permite el uso de CREATE INDEX y DROP INDEX
INSERT	Permite el uso de INSERT
LOCK TABLES	Permite el uso de LOCK TABLES en tablas para las que tenga el permiso SELECT
PROCESS	Permite el uso de SHOW FULL PROCESSLIST y de SHOW ENGINE.
REFERENCES	Permite crear claves ajenas al atributo especificado
RELOAD	Permite el uso de FLUSH
REPLICATION CLIENT	Permite al usuario preguntar dónde están los servidores maestro o esclavo
REPLICATION SLAVE	Es necesario para los esclavos de replicación (para leer eventos del log binario desde el maestro)
SELECT	Permite el uso de SELECT
SHOW DATABASES	Permite el uso de SHOW DATABASES, que muestra todas las bases de datos
SHOW VIEW	Permite el uso de SHOW CREATE VIEW
SHUTDOWN	Permite el uso de mysqladmin shutdown (para parar el servidor)
SUPER	Permite el uso de CHANGE MASTER, KILL, PURGE BINARY LOGS, SET GLOBAL y el comando mysqladmin admin debug, que permite conectar incluso si se llega *a max_connections*.
TRIGGER	Permite crear, eliminar, ejecutar y mostrar triggers.
UPDATE	Permite el uso de UPDATE
USAGE	Es un sinónimo de ningún privilegio

17.2.- Mostrar

```
SHOW GRANTS FOR usuario;
```

17.3.- Quitar

```
REVOKE INSERT ON *.* FROM usuario;
```

18.- Roles

```
CREATE ROLE nombre_rol;

GRANT SELECT, INSERT ON bbdd1.tabla1 TO nombre_rol;
```

```
GRANT nombre_rol TO usuario@localhost;
```

19.- Vistas

19.1.- Crear

```
CREATE VIEW nombre_vista
        AS SELECT campo1, campo2 FROM …
```

```
CREATE VIEW OR REPLACE nombre_vista
        AS SELECT campo1, campo2 FROM …
```

```
CREATE VIEW nombre_vista (Campo 1, Campo 2)
        AS SELECT campo1, campo2 FROM …
```

19.2.- Modificar

```
ALTER VIEW nombre_vista (campo1, campo3)
        AS SELECT campo1, campo3 FROM …
```

19.3.- Eliminar

```
DROP VIEW IF EXISTS nombre_vista;
DROP VIEW nombre_vista;
```

19.4.- Sentencias sobre vista

```
INSERT INTO nombre_vista
        VALUES (35, 48);
```

```
DELETE FROM nombre_vista
        WHERE campo2 < 40;
```

```
UPDATE nombre_vista
SET campo1 = 24;
```

20.- Triggers

Los triggers o disparadores son procesos que se ejecutan cuando se realiza una acción concreta.

```
CREATE TRIGGER nombre
AFTER INSERT ON tabla FOR EACH ROW
UPDATE tabla2 SET contador = contador + 1 WHERE nombre = NEW.nombre;
(Cuando inserta, se recoge el nombre más reciente ➔ NEW)
```

```
CREATE TRIGGER nombre
BEFORE DELETE ON tabla FOR EACH ROW
UPDATE tabla2 SET contador = contador – 1 WHERE nombre = OLD.nombre;
```