Bases de datos I

Módulo 3. SQL
Profesor: Lucas Catardo

Sesión 8: Introducción a DDL y DML – Queries SM

Agosto 26 de 2021

DDL: Lenguaje de definición de datos

Sentencias para la creación de tablas y registros. Se utilizan para realizar modificaciones sobre la estructura de la base de datos.

DML: Lenguaje de manipulación de datos

Sentencias para la consulta, actualización y borrado de datos.
Se utilizan para realizar consultas y modificaciones sobre los registros almacenados dentro de cada una de las tablas.

CREATE, DROP, ALTER

CREATE DATABASE

```
CREATE DATABASE db_peliculas;
USE db_peliculas;
```

CREATE TABLE

```
CREATE TABLE pelicula (
        id INT PRIMARY KEY AUTO_INCREMENT,
        titulo VARCHAR(60) NOT NULL,
        calificacion INT DEFAULT 3,
        director_id
        FOREIGN KEY (director_id) INT REFERENCES director(id)
);
```

DROP TABLE

```
DROP TABLE IF EXIST director;
```

ALTER TABLE

Opciones: ADD / MODIFY / DROP

```
ALTER TABLE pelicula
ADD rating DECIMAL(3,1) NOT NULL;

ALTER TABLE pelicula
MODIFY rating DECIMAL(4,1) NOT NULL;

ALTER TABLE pelicula
DROP rating;
```

INSERT, UPDATE, DELETE

INSERT

Insertar datos en todas las columnas:

```
INSERT INTO pelicula
VALUES (DEFAULT, 'La bruja', 5, 23);
```

Insertar datos en algunas columnas:

```
INSERT INTO pelicula (titulo, director)
VALUES
('El faro', 18),
('Los otros', 45);
```

UPDATE

```
UPDATE pelicula
SET calificacion = 5, director_id = 18
WHERE titulo = 'Hereditary';
```

DELETE

```
DELETE FROM pelicula
WHERE titulo = 'Us';
```

SELECT

Importar una base de datos:
- Server
- Data Import
- Import from Self-Contained File
- Seleccionar el archivo
- Siguiente…

SELECT

SELECT columna_1, columna_2
FROM nombre_tabla;

WHERE Y ORDER BY

WHERE (Filtrar) / ORDER BY (Ordenar)

SELECT titulo, calificacion
FROM pelicula
WHERE ano > 2000
ORDER BY ano DESC, calificacion;

Operadores lógicos y de comparación:
>, >=, <, <=, =, !=, <>
AND, OR

Otros operadores:
IS NULL, BETWEEN, IN, LIKE

Formato de fecha
"2010-01-01"

Sincrónico

Para nombrar FK:

FK_turno_paciente

Para generar un autonincremental a partir del 10:

ALTER TABLE nombre_tabla AUTO_INCREMENT=10;

Sesión 9. Agosto 27 de 2021.

Sincrónico.

Sesión 10. Checkpoint 1.

Sesión 11. Uso de DML. Queries DML.

- SELECT
- BETWEEN Y LIKE

BETWEEN:
Funciona con números, textos y fechas.

Ejemplo:
SELECT nombre, edad
FROM alumnos
WHERE edad BETWEEN 6 AND 12

Comodines (wildcards)

%: Es un sustituto que representa cero, uno o varios caracteres

_: Es un sustituto para un solo caracteres

Ejemplos:

SELECT nombre
FROM usuarios
WHERE edad LIKE '_a%'; // Devuelve los nombres que tengan la 'a' como segundo caracter

SELECT nombre
FROM usuarios
WHERE direccion LIKE '%Monroe%'; // Devuelve las direcciones que incluyan la palabra "Monroe"

SELECT nombre
FROM clientes
WHERE nombre LIKE 'Los%s'; // Devuelve los nombres de los clientes que empiezan con "Los" y terminan con "s"

LIMIT Y OFFSET

```
SELECT id, nombre, apellido
FROM alumnos
LIMIT 10
OFFSET 20;  // Recupera 10 registros omitiendo los 20 primeros de la consulta original
```

ALIAS

Alias para una columna:

```
SELECT razon_social_cliente AS nombre
FROM cliente
WHERE nombre LIKE 'a%';
```

Alias para una tabla:

```
SELECT nombre, apellido, edad
FROM alumnos_comision_inicial AS alumnos;
```

En general, el alias no debe tener más de una pálabra; en caso contrario separar las palabras con guion bajo o encerrar entre comillas el alias completo.

Operadores adicionales

IS NULL: Es nulo
BETWEEN: Entre dos valores
IN: Lista de valores
LIKE: Se ajusta a…

Sincrónico

Operadores:

```
IS NULL //Es nulo
BETWEEN //Entre dos valores
IN //Lista de valores
LIKE
```

El alias funciona en el ORDER BY, pero en general funciona en el WHERE.

SELECT PrecioUnitario, PrecioUnitario * 1,1 AS ‘Aumento del 10’ FROM productos;

Sesión 12.

Septiembre 3 de 2021.

Sincrónico.

Sesión 13. Informes

Septiembre 6 de 2021

Funciones de agregación
Realizan cálculos sobre un conjunto de datos y devuelven un único resultado.

COUNT – AVG - MIN - MAX – SUM

Ejemplos:

```
SELECT COUNT(*) FROM movies;

SELECT COUNT(id) AS total FROM movies WHERE genre_id=3;

SELECT AVG(rating) FROM movies;

SELECT SUM(length) FROM movies;

SELECT MIN(rating) FROM movies;

SELECT MAX(length) FROM movies;
```

GROUP BY

Se usa para agrupar los registros de la tabla resultante de una consulta por una o más columnas

Ejemplos

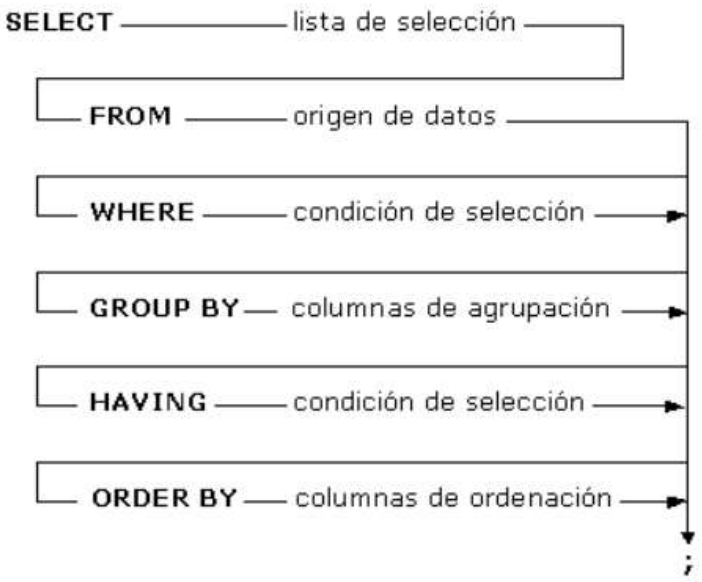
```
SELECT marca
FROM coche
WHERE anio_fabricacion>= 2010
GROUP BY marca;


SELECT marca, MAX(precio) AS precio_maximo
FROM coche
GROUP BY marca;
```

HAVING

Permite imponer condiciones sobre los datos agrupados

```
SELECT pais, COUNT(clienteId)
FROM clientes
GROUP BY pais
HAVING COUNT(clienteId)>=3;
```



Sincrónico

Utilizar el GROUP BY con el nombre de una columna y una función de agregación.

Funciones de alteración:

https://www.w3schools.com/mysql/mysql_ref_functions.asp

Sesión 14. DML – Queries agregadas

Septiembre 9 de 2021

Table Reference

Cuando hay relación muchos a uno:

```
SELECT clientes.id AS ID, clientes.nombre, ventas.fecha
FROM clientes, ventas
WHERE clientes.id = ventas.cliente_id;
```

Cuando la relación es muchos a muchos:

```
SELECT title, first_name, last_name
FROM movies, actor_movie, actors
WHERE movie_id = movies.id
AND actor_id = actors.id
```

JOIN

INNER JOIN

```
SELECT movies.id, title, genre_id, genres.id, name
FROM movies
INNER JOIN genres ON genre_id = genres.id
```

LEFT JOIN

```
SELECT movies.id, title, genre_id, genres.id, name
FROM movies
LEFT JOIN genres ON genre_id = genres.id
// Incluye también las películas que no tienen género asignado
```

RIGHT JOIN

```
SELECT movies.id, title, genre_id, genres.id, name
FROM movies
RIGHT JOIN genres ON genre_id = genres.id
// Incluye además los géneros que no tienen películas asignadas.
```

INNER JOIN EN RELACIÓN MUCHOS A MUCHOS

```
SELECT title, first_name, last_name
FROM movies
INNER JOIN actor_movie ON movie_id = movies.id
INNER JOIN actors ON actor_id = actors.id
```

DISTINCT
Elimina los datos repetidos de una consulta:

SELECT DISTINCT pais FROM usuarios;
// Devuelve la lista de los países (sin repetir) que tengan algún usuario asociado.

FUNCIONES DE ALTERACIÓN
No alteran los registros almacenados en la base de datos:

- CONCAT
- COALESCE
- DATEDIFF
- TIMEDIFF
- EXTRACT
- REPLACE
- DATE_FORMAT
- DATE_ADD
- DATE_SUB
- CASE

CONCAT
Permite concatenar datos:

SELECT CONCAT(first_name, " ", last_name) AS "Nombre completo"
FROM actors

COALESCE
Permite agregar un valor para reemplazar los valores nulos:

SELECT title, COALESCE(name, "No tiene género") FROM movies
LEFT JOIN genres ON genre_id = genres.id

SELECT id, apellido, nombre, COALESCE(telefono_movil, telefono_fijo, 'Sin datos')
AS telefono FROM cliente;
// Si no tiene teléfono móvil, presenta el fijo; si tampoco tiene fijo, despliega el texto 'Sin datos'

NOW, DATEDIFF, TIMEDIFF

SELECT title, DATEDIFF(NOW(), release_date)
FROM movies
// La columna muestra la diferencia de días entre las dos fechas

DATE_FORMAT, YEAR, EXTRACT, DATE_ADD, DATE_SUB

```
SELECT title, DATE_FORMAT(release_date, "%d/%m/%Y")
FROM movies;

SELECT title, YEAR(release_date)
FROM movies;

SELECT title, EXTRACT(day FROM release_date)
FROM movies;
//También SECOND, MINUTE, HOUR, DAY, WEEK, MONTH, QUARTER, YEAR
```

REPLACE

```
SELECT REPLACE(title, "La Guerra de las galaxias", "Star wars")
FROM movies;
```

LENGTH

```
SELECT title
FROM movies
WHERE LENGTH(title) > 10
// Las funciones se pueden utilizar dentro del WHERE o el ORDER BY
```

CASE

```
SELECT title, rating
CASE
        WHEN rating < 5 THEN "Mala"
        WHEN rating < 7 THEN "Buena"
        ELSE "Muy buena"
END AS Calificación
FROM movies
```

Sesión 15

Septiembre 10 de 2021

Sincrónico

HAVING: Para filtrar funciones de agregación