SQL

o SQL. Cláusulas y operadores. Vídeo 3

Cláusulas

Cláusula	Descripción
FROM	Especifica la tabla de la que se quieren obtener los registros
WHERE	Especifica las condiciones o criterios de los registros seleccionados
GROUP BY	Para agrupar los registros seleccionados en función de un campo
HAVING	Especifica las condiciones o criterios que deben cumplir los grupos
ORDER BY	Ordena los registros seleccionados en función de un campo

o SQL. Cláusulas y operadores. Vídeo 3

Operadores de comparación

Operador	Significado
<	Menor que
>	Mayor que
=	Igual que
>=	Mayor o igual que
<=	Menor o igual que
<>	Distinto que
BETWEEN	Entre. Utilizado para especificar rangos de valores
LIKE	Cómo. Utilizado con caracteres comodín (? *)
In	En. Para especificar registros en un campo en concreto

CREATE TABLE

```
/**crear tabla identificando columnas)  **/
CREATE TABLE groceries (id INTEGER PRIMARY KEY, name TEXT, quantity INTEGER );
/**  crear tabla  configurando  id autogenerada**/
CREATE TABLE exercise_logs
   (id INTEGER PRIMARY KEY AUTOINCREMENT,
   type TEXT);
```

INSERT INTO __ VALUES /

/**añadir valores para todos los campos**/

INSERT INTO groceries VALUES (1, "Bananas", 4);

/** añadir solo en campos concretos , poniendo id autogenerada**/

INSERT INTO exercise_logs(type, minutes, calories, heart_rate) VALUES ("biking", 30, 100, 110);

SELECT __ FROM

/** mostrar toda la tabla : **/

SELECT * FROM groceries;

/** mostrar solo los campos seleccionados **/

SELECT name,quantity FROM groceries

ORDER BY

/** ordenar lista por pasillo: ORDER BY **/

SELECT * FROM groceries ORDER BY aisle;

WHERE

/** Filtrar pasillo >5 **/

SELECT * FROM groceries WHERE aisle > 5 ORDER BY aisle;

SUM , MAX(quantity)

/** suma (o otras operaciones) elementos de una columna **/

SELECT SUM(quantity) FROM groceries;

/** cantidad máxima de un elemento (quantity en el eejmplo) **/

SELECT MAX(quantity) FROM groceries;

GROUP BY

/** agrupa pasillos y luego muestra la suma de los elementos de cada pasillo sin especificar pasillo **/

SELECT SUM(quantity) FROM groceries GROUP BY aisle;

/** mustra pasillo y la suma de los elemntos de ese pasillo (ailse). **/

SELECT aisle, SUM(quantity) FROM groceries GROUP BY aisle;

AND / OR

```
/** AND tiene prioridad sobre OR, pero con paréntesis podemos lograr la expresión deseada **/

/** AND  selecciona calorias >50 y más de 30 minutos **/
SELECT * FROM exercise_logs WHERE calories > 50 AND minutes < 30;

/* *OR selcciona >50calorias y ritmo cardíaco>100  * */
SELECT * FROM exercise_logs WHERE calories > 50 OR heart_rate > 100;
```

IN / NOT IN

```
/** para seleccionar varios valores . Primero vemos como es sin IN **/

SELECT * FROM exercise_logs WHERE type = "biking" OR type = "hiking" OR type = "tree climbing" OR
type = "rowing";


/* Con IN */
SELECT * FROM exercise_logs WHERE type IN ("biking", "hiking", "tree climbing", "rowing");
/** también podemos poner los valores que no cumplen con NOT IN **/
SELECT * FROM exercise_logs WHERE type NOT IN ("biking", "hiking", "tree climbing", "rowing");
```

CASE

```
/* CASE  funciona como un switch o iF para seleccionar opciones */
SELECT type, heart_rate,
   CASE
      WHEN heart_rate > 220-30 THEN "above max"
      WHEN heart_rate > ROUND(0.90 * (220-30)) THEN "above target"
      WHEN heart_rate > ROUND(0.50 * (220-30)) THEN "within target"
      ELSE "below target"
   END as "hr_zone"
FROM exercise_logs;

/** para agrupar esas consultas por zonas **/
SELECT COUNT(*),
   CASE
      WHEN heart_rate > 220-30 THEN "above max"
      WHEN heart_rate > ROUND(0.90 * (220-30)) THEN "above target"
      WHEN heart_rate > ROUND(0.50 * (220-30)) THEN "within target"
      ELSE "below target"
   END as "hr_zone"
FROM exercise_logs
GROUP BY hr_zone;
```

SUBCONSULTAS

/** si tenemos dos tablas y queremos hacer consulta entre elals **/

```
CREATE TABLE exercise_logs
    (id INTEGER PRIMARY KEY AUTOINCREMENT,
    type TEXT,
    minutes INTEGER,
    calories INTEGER,
    heart_rate INTEGER);

INSERT INTO exercise_logs(type, minutes, calories, heart_rate) VALUES ("biking", 30, 100, 110);
INSERT INTO exercise_logs(type, minutes, calories, heart_rate) VALUES ("dancing", 15, 200, 120);
INSERT INTO exercise_logs(type, minutes, calories, heart_rate) VALUES ("tree climbing", 30, 70, 90);
INSERT INTO exercise_logs(type, minutes, calories, heart_rate) VALUES ("rowing", 30, 70, 90);
INSERT INTO exercise_logs(type, minutes, calories, heart_rate) VALUES ("hiking", 60, 80, 85);

/* IN */
SELECT * FROM exercise_logs WHERE type IN ("biking", "hiking", "tree climbing", "rowing");

/** nueva tabla **/
CREATE TABLE drs_favorites
    (id INTEGER PRIMARY KEY,
    type TEXT,
    reason TEXT);

INSERT INTO drs_favorites(type, reason) VALUES ("biking", "Improves endurance and flexibility.");
INSERT INTO drs_favorites(type, reason) VALUES ("hiking", "Increases cardiovascular health.");
/** comprovamos  que campos type tenemos **/
SELECT type FROM drs_favorites;
/** miramos en la primera tabla esos campos que coinciden **/
SELECT * FROM exercise_logs WHERE type IN ("biking", "hiking");
```

/** con este método no se actualizan los campos si se modifican en la tabla, por eso hacemos una consulta anidada simplemente pegando la subconsulta dentro de l consulta **/

```
SELECT * FROM exercise_logs WHERE type IN (SELECT type FROM drs_favorites);
```

LIKE

/** LIKE buscará un valor (entre % %) dentro de una frase **/

SELECT * FROM exercise_logs WHERE type IN (
SELECT type FROM drs_favorites WHERE reason LIKE "%cardiovascular%");

LIKE

/** compara de modo flexible buscando solo elementos deseados dentro de una frase **/

```
SELECT * FROM exercise_logs WHERE type IN (
    SELECT type FROM drs_favorites WHERE reason = "Increases cardiovascular health");
```

/* LIKE */

```
SELECT * FROM exercise_logs WHERE type IN (
    SELECT type FROM drs_favorites WHERE reason LIKE "%cardiovascular%");
```

BETWEN

/** consulta entre dos parámetros. En el ejemplo son fechas **/

```
SELECT * FROM PRODUCTOS WHERE FECHA BETWEEN '2000-03-01' AND '2000-04-30'
```

/** equivale a: **/

```
SELECT * FROM PRODUCTOS WHERE FECHA >='2000-03-01' AND FECHA <='2000-04-30'
```

AS

/** crea un nombre para la columna (resultado de sumar calorías) **/

SELECT type, SUM(calories) AS total_calories FROM exercise_logs GROUP BY type;

HAVING

/** filtra valores para un resultado AGRUPADO, no para cada valor individual de la tabla . La suma de calorías la ponemos en una columna llamada total_calories i luego con HAVING pedimos que el resulta total de esas sumas >150. (Es fácil confundir HAVING con WHERE **/

SELECT type, SUM(calories) AS total_calories FROM exercise_logs
GROUP BY type HAVING total_calories > 150

HAVING COUNT

/** COUNT: mostrará type que tenga 2 o más valores **/

SELECT type FROM exercise_logs GROUP BY type HAVING COUNT(*) >= 2;

```
/** Harry Potter **/
CREATE TABLE books (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    author TEXT,
    title TEXT,
    words INTEGER);

INSERT INTO books (author, title, words)
    VALUES ("J.K. Rowling", "Harry Potter and the Philosopher's Stone", 79944);
INSERT INTO books (author, title, words)
    VALUES ("J.K. Rowling", "Harry Potter and the Chamber of Secrets", 85141);
INSERT INTO books (author, title, words)
    VALUES ("J.K. Rowling", "Harry Potter and the Prisoner of Azkaban", 107253);
INSERT INTO books (author, title, words)
    VALUES ("J.K. Rowling", "Harry Potter and the Goblet of Fire", 190637);
INSERT INTO books (author, title, words)
    VALUES ("J.K. Rowling", "Harry Potter and the Order of the Phoenix", 257045);
INSERT INTO books (author, title, words)
    VALUES ("J.K. Rowling", "Harry Potter and the Half-Blood Prince", 168923);
INSERT INTO books (author, title, words)
    VALUES ("J.K. Rowling", "Harry Potter and the Deathly Hallows", 197651);

INSERT INTO books (author, title, words)
    VALUES ("Stephenie Meyer", "Twilight", 118501);
INSERT INTO books (author, title, words)
    VALUES ("Stephenie Meyer", "New Moon", 132807);
INSERT INTO books (author, title, words)
    VALUES ("Stephenie Meyer", "Eclipse", 147930);
INSERT INTO books (author, title, words)
    VALUES ("Stephenie Meyer", "Breaking Dawn", 192196);

INSERT INTO books (author, title, words)
    VALUES ("J.R.R. Tolkien", "The Hobbit", 95022);
INSERT INTO books (author, title, words)
    VALUES ("J.R.R. Tolkien", "Fellowship of the Ring", 177227);
INSERT INTO books (author, title, words)
    VALUES ("J.R.R. Tolkien", "Two Towers", 143436);
INSERT INTO books (author, title, words)
    VALUES ("J.R.R. Tolkien", "Return of the King", 134462);
```