Consultas Simples

Unidad 4
Bases de Datos - 1º DAW

Contenidos

1. El lenguaje DML de SQL
2. La declaración SELECT
3. Funciones en la SELECT
4. **Modificadores, cláusulas y operadores**
5. Tratamiento de los valores nulos

4. Modificadores, cláusulas y operadores

U4. Consultas Simples

Bases de Datos
1º DAW

Francisco Javier Castillo Jiménez

4.1.1 Modificadores

En la cláusula SELECT se pueden añadir los modificadores **ALL**, **DISTINCT** y **DISTINCTROW**.

- **ALL** se utiliza para incluir todas las filas, inclusive las repetidas. No hace falta indicarlo puesto que es una opción que se incluye por defecto.
- **DISTINCT** no va a mostrar las filas repetidas en el resultado de la consulta.
- **DISTINCTROW** se emplea como un sinónimo de DISTINCT.

Si en la SELECT se utilizan varias columnas, se considera un valor repetido aquel que es misma combinación en diferentes filas.

4.1.2 Ejemplo de modificadores

- **Incluir todos los nombres de los vigilantes.** SELECT ALL nombreVig FROM vigilante;
- **Omitir los nombres repetidos.** SELECT DISTINCT nombreVig FROM vigilante;
- **Incluir todos los nombres y apellidos de los vigilantes** SELECT ALL nombreVigilante, ape1, ape2 FROM vigilante;
- **Incluir los nombres y apellidos sin que se repitan** SELECT DISTINCT nombreVigilante, ape1, ape2 FROM vigilante;

4.2 Cláusula ORDER BY

La cláusula **ORDER BY** va a permitir ordenar las filas que se muestran en el resultado de una consulta.

Estos resultados se pueden ordenar de forma ascendente o descendente, además de permitir ordenar por varias columnas con diferentes niveles de ordenación.

La sintaxis oficial que aparece en la documentación de MySQL es:

[ORDER BY {col_name | expr | position} [ASC | DESC], ...]

SELECT * FROM fortines **ORDER BY** altura ASC, diametro DESC;

4.2 Cláusula LIMIT

La cláusula **LIMIT** va a permitir acotar a un límite el número de filas que ofrece el resultado de una consulta.

La sintaxis oficial que aparece en la documentación de MySQL es:

[LIMIT {[*offset*,] *row_COUNT* | *row_COUNT* OFFSET *offset*}] sustituyendo offset por el número de filas consecutivas que no muestran y row_COUNT por el número de filas que se quieren obtener.

-- Muestra tres filas a partir de la quinta

SELECT * FROM fortines **LIMIT** 5,3;

4.2 Cláusula WHERE

La cláusula **WHERE** va a permitir añadir filtros a las consultas para seleccionar únicamente las filas que cumplen una condición.

A estas condiciones se les denomina predicados condicionales y su resultado puede ser **true**, **false** o **unknown** (para valores nulos).

Existen varios tipos de predicados que puede tener esta cláusula:

- Comparativos de valores, expresiones o expresiones regulares.
- Comprobaciones en rangos y conjuntos.
- Verificador de valores nulos.

En estas consultas se utilizarán operadores aritméticos, lógicos, ...

4.3 Operadores

Los operadores empleados en los predicados condicionales pueden ser aritméticos, de comparación o lógicos.

Con ellos se construirá este predicado condicional que se valorará como **true**, **false** ó **unknown**.

Así, sólo se mostrarán en el resultado de la consulta las filas que cumplan la condición de verdadero en el predicado.

A continuación se muestran los operadores más utilizados en MySQL para realizar las consultas.

4.3.1 Operadores

Op.	Aritméticos
+	Suma
-	Resta
*	Multiplicación
/	División
%	Módulo

Op.	Comparación
<	Menor que
<=	Menor o igual
>	Mayor que
>=	Mayor o igual
<>	Distinto
!=	Distinto
=	Igual que

Op.	Lógicos
AND	Y lógica
&&	Y lógica
OR	O lógica
||	O lógica
NOT	Negación lógica
!	Negación lógica

4.3.2 Ejemplos con operadores

Siguiendo con la base de datos de los *fuertes neomedievales*, se tiene:

- **Nombre de todos los vigilantes cuyo apellido sea Díaz.** SELECT nombreVig FROM vigilante WHERE ape1=”Díaz”;
- **Nombre de todos los vigilantes cuyo apellido sea Díaz ó Pérez.** SELECT nombreVig FROM vigilante WHERE ape1=”Díaz” OR ape1=”Pérez”;
- **Nombre de todos los vigilantes nacidos después del 1 mayo 1920.** SELECT nombreVig FROM vigilante WHERE fechaNacimiento >= “1920/05/01”;

4.4 Operador BETWEEN

El operador **BETWEEN** se emplea para comprobar si un valor determinado se encuentra en un rango de valores, por ejemplo entre dos fechas.

```
SELECT nombreVig
FROM vigilante
WHERE fechaNacimiento BETWEEN “1920-01-01” AND “1920-12-31”;
```

También se puede emplear para valores numéricos además de poder añadirle el operador NOT delante de BETWEEN para especificar que no esté en ese rango.

4.5 Operador IN

El operador **IN** se emplea para comprobar si un valor determinado se encuentra en una lista de valores. A diferencia del BETWEEN, el operador **IN** se emplea más para cadenas de caracteres.

```
SELECT nombreVig
FROM vigilante
WHERE ape1 IN (“Mcintyre”, “Petersen”, “Higgins”);
```

También se puede emplear para valores numéricos además de poder añadirle el operador NOT delante de IN para especificar que no esté en esa lista.

4.6 Operador LIKE

El operador **LIKE** se emplea para comparar si una cadena de caracteres coincide con un patrón dado.

Ese patrón va formado por una cadena de caracteres, utilizando dos símbolos:

- %: equivale a cualquier conjunto de caracteres.
- _: equivale a un caracter.

Muestra todos los datos de los vigilantes cuyo nombre empieza por A
SELECT * FROM vigilantes WHERE nombreVig **LIKE** “A%”;

4.7 Operador IS

El operador **IS** permite comprobar si una columna posee en un momento dado un valor nulo o no.

SELECT * FROM vigilante WHERE ape2 **IS NULL**;

ó

SELECT * FROM vigilante WHERE ape2 **IS NOT NULL**;