TECNOLÓGICO NACIONAL DE MÉXICO®

Ciencia de DATOS

DIPLOMADO “CIENCIA DE DATOS”

Módulo 3

ANÁLISIS Y MANIPULACIÓN DE BASE DE DATOS

1. Introducción
2. Exploración de datos
3. **Restricciones y ordenamiento de datos**

3.1 Restringir datos

La cláusula WHERE en SQL se utiliza en una consulta SELECT para filtrar los resultados basándose en una o varias condiciones. Permite especificar criterios de búsqueda para seleccionar solo los registros que cumplan con ciertas condiciones, excluyendo aquellos que no satisfacen los criterios definidos.

La sintaxis básica de la cláusula WHERE es la siguiente:

```
SELECT *|{[DISTINCT] columna|expresión [alias],...}
FROM    tabla
WHERE   condición:
```

- SELECT identifica las columnas
- FROM Identifica la tabla
- WHERE Condiciona los renglones

Ejemplo 1.

Esta consulta recupera a todos los alumnos de séptimo semestre:

```
ELECT matricula AS numero, nombre nombre_completo, semestre
ROM   alumnus
/HERE  semestre = 7;
```

NUMERO	NOMBRE_COMPLETO	SEMESTRE
406	GONZALEZ MARQUEZ JOSE ANTONIO	7
407	GONZALEZ LARA ALEJANDRO	7
408	GUERRERO TORRES JUAN DIEGO	7

Operadores de relación

Los operadores de relación en SQL se utilizan en la cláusula WHERE de una consulta SELECT para establecer condiciones de comparación entre valores. Estos operadores permiten filtrar los resultados de una consulta basándose en relaciones de igualdad, desigualdad o comparación entre los valores de una o varias columnas:

Operador	Descripción
>	Mayor
>=	Mayor que
<	Menor
<=	Menor que
<>	Diferente
!=	Diferente
=	Igual
BETWEEN AND	Rango
IN	En una lista
LIKE	Patron
IS NULL	No es nulo

Operador BETWEEN

El operador BETWEEN en SQL se utiliza para evaluar si un valor se encuentra dentro de un rango específico, que puede incluir valores numéricos, fechas o texto. Este operador es especialmente útil cuando deseas filtrar registros que tienen un valor dentro de un rango específico en una columna. La sintaxis básica del operador BETWEEN es la siguiente:

```
SELECT columnas
FROM   tabla
WHERE  columna BETWEEN valor  inicial AND valor  final:
```

Ejemplo 1.

Esta consulta recupera a todos los alumnos de tercer a octavo semestre

```
SELECT matricula, nombre
FROM   alumnos
WHERE  semestre BETWEEN 3 AND 8;
```

MATRICULA	NOMBRE
403	FLORES MARTIEZ JORGE
404	FUENTES OLGUIN KARLA
405	GONZALEZ DIAZ ROBERTO
406	GONZALEZ MARQUEZ JOSE ANTONIO
407	GONZALEZ LARA ALEJANDRO
408	GUERRERO TORRES JUAN DIEGO
409	GUERRERO LOPEZ ALMA
410	HERNANDEZ DIAZ ROCIO
411	FERNANDEZ RODRIGUEZ GAEL

Ejemplo 2.

Supongamos que tenemos una tabla "empleados" con una columna llamada "salario". Si queremos seleccionar los empleados cuyo salario está dentro del rango de $40,000 y $60,000, la consulta sería:

```
SELECT nombre, salario
FROM   empleados
```

Ejemplo 3.

Si la tabla contiene una columna "fecha_ingreso" de tipo fecha, y queremos seleccionar los registros cuya fecha de ingreso está entre el 1 de enero de 2023 y el 31 de diciembre de 2023, la consulta sería:

```
SELECT nombre, fecha_ingreso
FROM   empleados
```

Operador IN

El operador IN en SQL se utiliza para verificar si un valor se encuentra dentro de un conjunto específico de valores. Es especialmente útil cuando deseas realizar una consulta que incluya varios valores posibles para una columna determinada

La sintaxis básica del operador IN es la siguiente:

```
SELECT columnas
FROM   table
```

Ejemplo 1

Si queremos recuperar la matrícula y nombre de todos los estudiantes de quinto, séptimo y noveno semestre:

```
SELECT matricula, nombre
FROM   alumnos
```

MATRICULA	NOMBRE
400	BALLESTEROS REYES MARIA
401	CISNEROS BAEZA ALEJANDRO
402	ESQUIVEL RUIZ BRANDON
406	GONZALEZ MARQUEZ JOSE ANTONIO
407	GONZALEZ LARA ALEJANDRO
408	GUERRERO TORRES JUAN DIEGO

Ejemplo 2.

Si la tabla contiene una columna "producto" y queremos seleccionar los registros cuyo producto es "A", "B" o "C", la consulta sería:

```
SELECT nombre, producto
FROM   productos
```

Ejemplo 3.

Si deseamos seleccionar empleados que tienen un rango específico de edades, podemos utilizar el operador IN junto con una lista de valores numéricos:

```
SELECT nombre, edad
FROM   empleados
```

Operador LIKE

El operador LIKE en SQL se utiliza para realizar coincidencias parciales en una columna que contiene datos de tipo texto (cadenas de caracteres). Permite buscar registros que contengan un patrón específico dentro de un campo de texto

El operador LIKE utiliza comodines para definir el patrón de búsqueda

- % (porcentaje): Representa cualquier secuencia de caracteres, incluyendo cero caracteres. Es útil para buscar coincidencias que empiecen o terminen con un conjunto de caracteres o para buscar subcadenas en cualquier posición.
- _ (guion bajo): Representa un solo carácter. Es útil para buscar palabras de una longitud específica.

Ejemplo 1.

% (porcentaje): Representa cualquier secuencia de caracteres, incluyendo cero caracteres. Es útil para buscar coincidencias que empiecen o terminen con un conjunto de caracteres o para buscar subcadenas en cualquier posición.

Consultar a todos los alumnos cuyo nombre empiece con la letra A:

```
SELECT matricula, nombre
FROM   alumnos
WHERE  nombre LIKE ‘F%:
```

MATRICULA	NOMBRE
403	FLORES MARTIEZ JORGE
404	FUENTES OLGUIN KARLA

Ejemplo 2.

_ (guión bajo): Representa un solo carácter. Es útil para buscar palabras de una longitud específica.

Esta consulta seleccionará los clientes cuyos nombres tengan dos caracteres y el tercero sea la letra "a".

```
SELECT nombre
FROM   clientes
WHERE  nombre LIKE '    a':
```

Operador AND

El operador AND en SQL se utiliza en la cláusula WHERE de una consulta SELECT para combinar múltiples condiciones y filtrar los resultados basándose en todas las condiciones especificadas. El operador AND requiere que ambas condiciones sean verdaderas para que el registro sea incluido en los resultados de la consulta.

La sintaxis básica del operador AND es la siguiente:

```
SELECT columnas
FROM   tabla
WHERE  condicion1 AND condicion2;
```

Ejemplo 1.

Consultar a todos los alumnos cuyo nombre empiece con la letra A y sean de octavo semestre en adelante

```
SELECT matricula, nombre
FROM   alumnos
WHERE  nombre LIKE 'B%
AND    semestre > 8;
```

MATRICULA	NOMBRE
400	BALLESTEROS REYES MARIA

Ejemplo 2.

Supongamos que tenemos una tabla "empleados" con las columnas "departamento" y "salario". Si queremos seleccionar los empleados que pertenecen al departamento de "Ventas" y tienen un salario mayor a $40,000, la consulta sería:

```
SELECT nombre, salario
FROM   empleados
WHERE  departamento = 'Ventas'
AND    salario > 40000;
```

Ejemplo 3.

Si la tabla contiene una columna "edad" y "experiencia", y queremos seleccionar los registros de empleados que tienen una edad mayor a 25 años y más de 5 años de experiencia, podemos utilizar el operador AND:

```
SELECT nombre, edad, experiencia
FROM   empleados
WHERE  edad > 25
AND    experiencia > 5;
```

Operador OR

El operador OR en SQL se utiliza en la cláusula WHERE de una consulta SELECT para combinar múltiples condiciones y filtrar los resultados basándose en al menos una de las condiciones especificadas. El operador OR requiere que al menos una de las condiciones sea verdadera para que el registro sea incluido en los resultados de la consulta.

La sintaxis básica del operador OR es la siguiente:

```
SELECT columnas
FROM   tabla
WHERE  condicion1 OR condicion2;
```

Ejemplo 1.

Consultar la matrícula y el nombre de los alumnos con estatus “Regular” o que se encuentre en quinto semestre

```
SELECT matricula, nombre
FROM   alumnos
WHERE  estatus = ’R’
OR     semestre = 5;
```

Ejemplo 2.

Supongamos que tenemos una tabla "empleados" con las columnas "departamento" y "salario". Si queremos seleccionar los empleados que pertenecen al departamento de "Ventas" o tienen un salario mayor a $40,000, la consulta sería:

```
SELECT nombre, salario
FROM   empleados
WHERE  departamento = 'Ventas'
OR     salario > 40000;
```

Ejemplo 3.

Si la tabla contiene una columna "edad" y "experiencia", y queremos seleccionar los registros de empleados que tienen una edad mayor a 25 años o más de 5 años de experiencia, podemos utilizar el operador OR:

```
SELECT nombre, edad, experiencia
FROM   empleados
WHERE  edad > 25
OR     experiencia > 5;
```

Operador NOT

El operador NOT en SQL se utiliza en la cláusula WHERE de una consulta SELECT para negar una condición específica. Es decir, el operador NOT invierte el resultado de una condición, de modo que si la condición original es verdadera, el operador NOT la convierte en falsa, y viceversa.

La sintaxis básica del operador NOT es la siguiente:

```
SELECT columnas
FROM   tabla
WHERE  NOT condicion;
```

Ejemplo 1.

Supongamos que tenemos una tabla "empleados" con una columna llamada "estado_civil", que puede tener valores "casado" o "soltero". Si queremos seleccionar los empleados que NO están casados, la consulta sería:

```
SELECT nombre, estado_civil
FROM   empleados
WHERE  NOT estado_civil = 'casado';
```

Ejemplo 2.

Si la tabla contiene una columna "edad", y queremos seleccionar los registros de empleados que NO tienen 30 años, podemos utilizar el operador NOT de la siguiente manera:

```
SELECT nombre, edad
FROM empleados
WHERE NOT edad = 30;
```

Ejemplo 3.

Consultar a los alumnos que no están en quinto, séptimo y noveno semestre:

```
SELECT matricula, nombre
FROM   alumnos
WHERE  semestre NOT IN ( 5,7,9 );
```

MATRICULA	NOMBRE
403	FLORES MARTIEZ JORGE
404	FUENTES OLGUIN KARLA
405	GONZALEZ DIAZ ROBERTO
409	GUERRERO LOPEZ ALMA
410	HERNANDEZ DIAZ ROCIO
411	FERNANDEZ RODRIGUEZ GAEL

3.2 Ordenar datos

Clausula ORDER BY

La cláusula ORDER BY en SQL se utiliza para ordenar los resultados de una consulta SELECT según una o varias columnas especificadas. Con ORDER BY, puedes organizar los registros en el conjunto de resultados en un orden ascendente (de menor a mayor) o descendente (de mayor a menor) basándote en los valores de una o varias columnas.

La sintaxis básica de ORDER BY es la siguiente:

```
SELECT columnas
FROM   tabla
ORDER BY columna [ASC|DESC];
```

- **ASC** (opcional): Especifica que los resultados se ordenen en orden ascendente (de menor a mayor). Es el orden predeterminado si no se especifica ASC o DESC.
- **DESC**: Especifica que los resultados se ordenen en orden descendente (de mayor a menor).

Ejemplo 1.

Supongamos que tenemos una tabla "empleados" con las columnas "nombre" y "salario". Si queremos seleccionar los empleados y ordenarlos por su salario en orden ascendente, la consulta sería:

```
SELECT nombre, salario
FROM   empleados
ORDER BY salario ASC;
```

Ejemplo 2.

Si deseamos ordenar los empleados por su salario en orden descendente, podemos utilizar DESC:

```
SELECT nombre, salario
FROM   empleados
ORDER BY salario DESC;
```

Ejemplo 3.

Si la tabla contiene una columna "fecha_nacimiento" de tipo fecha, y queremos seleccionar los registros y ordenarlos por su fecha de nacimiento en orden ascendente, la consulta sería:

Puedes utilizar ORDER BY con una sola columna o con varias columnas para ordenar los resultados en función de múltiples criterios. Por ejemplo, para ordenar primero por una columna y luego por otra en caso de empate:

En este caso, los empleados se ordenarían primero por el nombre del departamento en orden ascendente y luego, dentro de cada departamento, por salario en orden descendente.

Ejemplo 4.

Consultar a los alumnos de primero, segundo y tercer semestre ordenados por nombre: