SQL - Condiciones de Búsqueda, Agrupación y Resumen de Datos

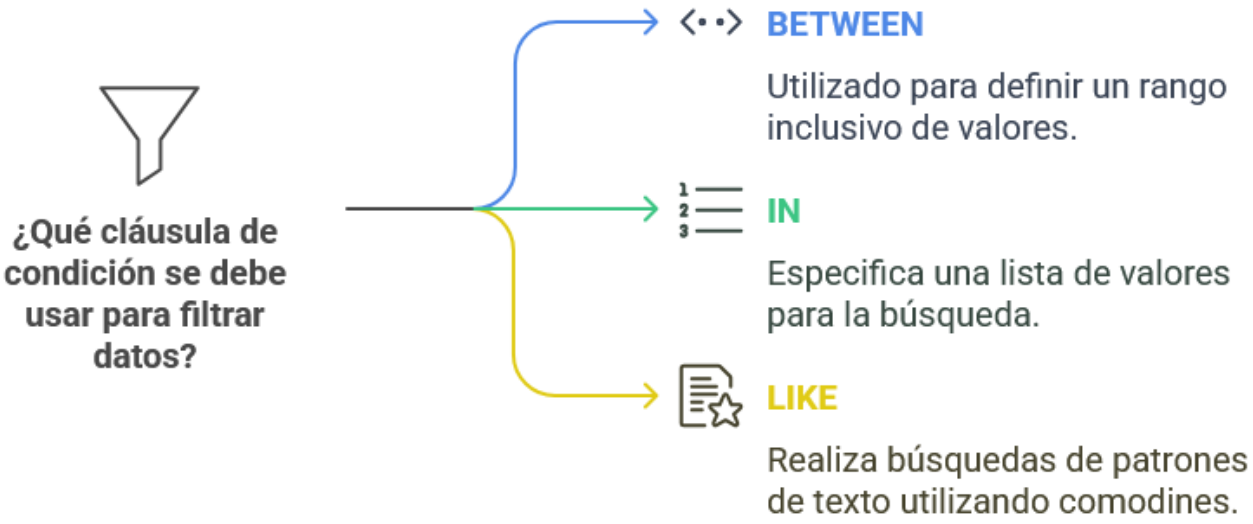
Resumen Ejecutivo:

Este documento presenta un resumen detallado de los conceptos fundamentales de SQL, específicamente en lo que respecta a la filtración de datos mediante la cláusula WHERE y sus diversas condiciones, la manipulación de valores NULL, el uso de operadores lógicos, la eliminación de duplicados con DISTINCT, el ordenamiento de resultados con ORDER BY, las funciones de agregación para resumir datos, la agrupación de filas con GROUP BY, el filtrado de grupos con HAVING, y las cláusulas COMPUTE y COMPUTE BY para generar totales y subtotales.

Temas Principales y Detalles Clave:

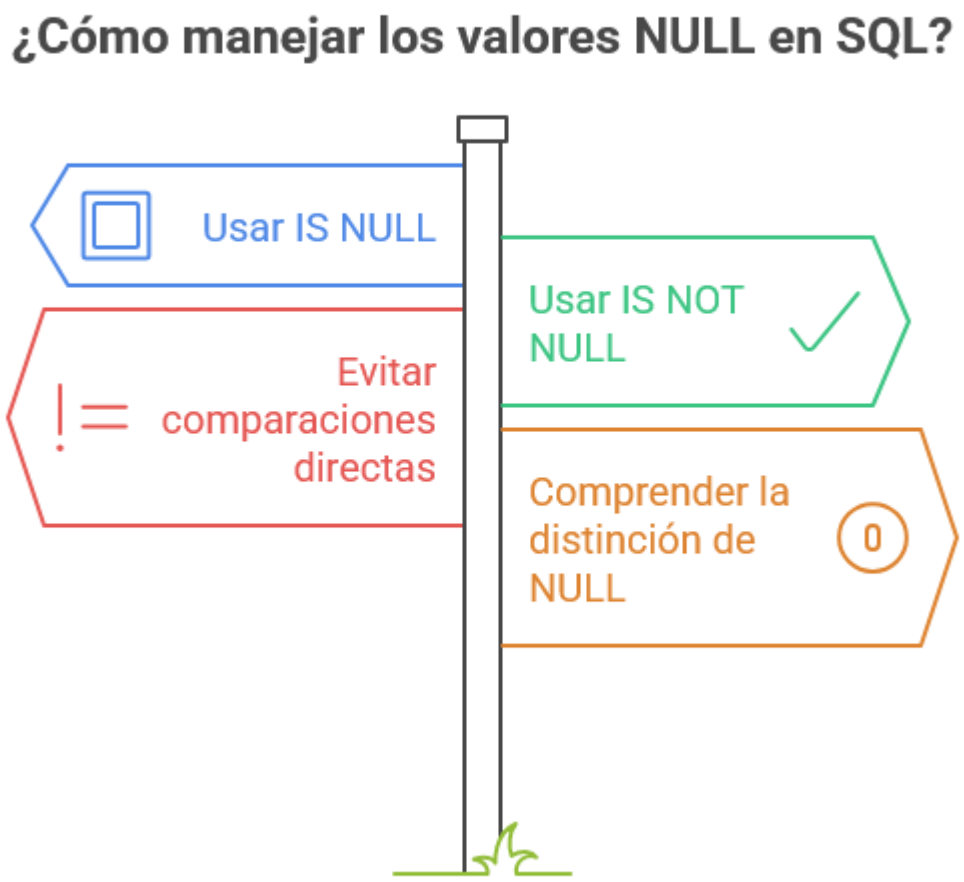
1. Filtrando Datos con la Cláusula WHERE:

- La cláusula WHERE se utiliza para especificar qué registros se deben incluir en el resultado de una consulta SELECT.
- "La cláusula WHERE en la sentencia SELECT especifica que registros se quieren visualizar en la consulta y cuales no."
- Sintaxis básica: SELECT columnas FROM tabla WHERE condición
- **Condiciones de Comparación:BETWEEN:** Define un rango inclusivo de búsqueda.
- "La cláusula BETWEEN especifica un rango de búsqueda."
- Sintaxis: test_expression [NOT] BETWEEN begin_expression AND end_expression
- **IN:** Especifica una lista de valores para la búsqueda.
- "La cláusula IN especifica un rango de búsqueda." (Nota: Aunque el texto dice "rango", se refiere a una lista de valores).
- Sintaxis: test_expression [NOT] IN (expression [,...n])
- **LIKE:** Realiza búsquedas de patrones de texto utilizando comodines.
- "LIKE selecciona los registros los cuales aquellos campos que se especifiquen contengan una porción de una determinada cadena de caracteres."
- Sintaxis: match_expression [NOT] LIKE pattern [ESCAPE escape_character]



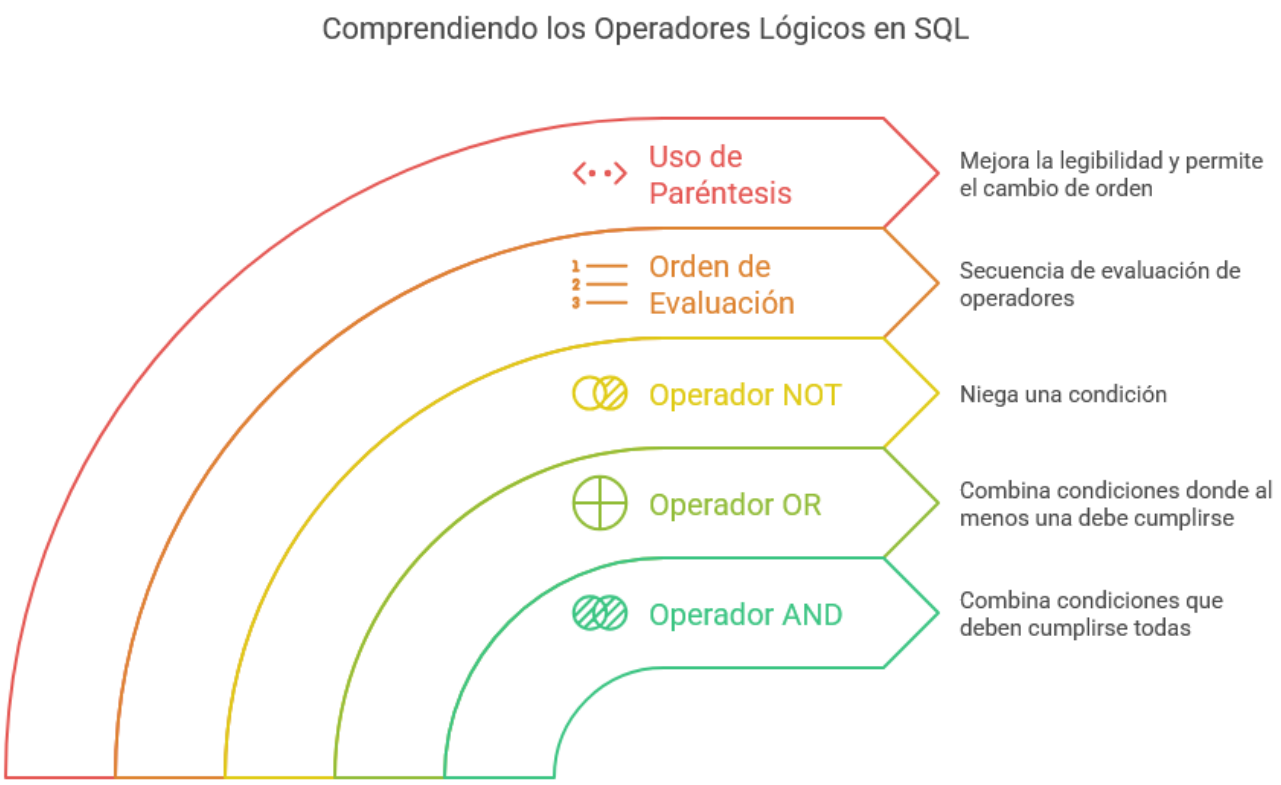
2. Valores NULL:

- NULL representa un valor desconocido o la ausencia de un valor en un campo.
- "NULL significa valor desconocido."
- "En otras palabras cuando el usuario o una aplicación no cargan ningún valor en el campo de una tabla, el mismo toma el valor NULL."
- Para buscar valores NULL, se utiliza la cláusula IS [NOT] NULL.
- SELECT * FROM tabla WHERE columna IS [NOT] NULL
- Importante distinción: NULL no es igual a 0 o a una cadena vacía ("").
- "NULL no significa 0 “cero” o blanco “”"
- Las comparaciones directas con NULL siempre resultan en falso.
- "Los valores NULL fallan con todas las consultas. No se puede utilizar ninguna de las condiciones de comparación:" (Se listan los operadores de comparación que no funcionan directamente con NULL).
- "NULL no es igual a NULL. O sea la comparación (NULL = NULL) es Falsa."



3. Operadores Lógicos AND, OR y NOT:

- Se utilizan para combinar múltiples condiciones de búsqueda en la cláusula WHERE.
- "Son utilizados para realizar búsquedas con varios argumentos."
- "El use de los operadores AND y OR le permiten combinar argumentos de búsquedas."
- "Los operadores lógicos AND y OR son utilizados para conectar diferentes expresiones de búsquedas dentro de la cláusula WHERE"
- NOT niega una expresión.
- "NOT niega la expresión"
- Orden de evaluación de los operadores lógicos: NOT (primero), AND (luego), OR (finalmente).
- "Cuando se utiliza mas de una operador lógico, NOT es evaluado primero, luego AND y finalmente OR"
- Los paréntesis se pueden utilizar para alterar el orden de evaluación y mejorar la legibilidad.
- "Los paréntesis pueden ser utilizados para cambiar el orden de la evaluaciones y también poder hacer mas legible las expresiones"
- Sintaxis: SELECT * FROM tabla WHERE expresión AND | NOT [NOT] expresión



4. Eliminación de Registros Duplicados con DISTINCT:

- La cláusula DISTINCT se utiliza en la sentencia SELECT para mostrar solo registros únicos, eliminando las filas duplicadas en el conjunto de resultados.
- "Se utiliza para eliminar los registros duplicados de los resultados de las consultas."
- Sintaxis mencionada incorrectamente, la sintaxis correcta es SELECT DISTINCT columna1, columna2, ... FROM tabla;

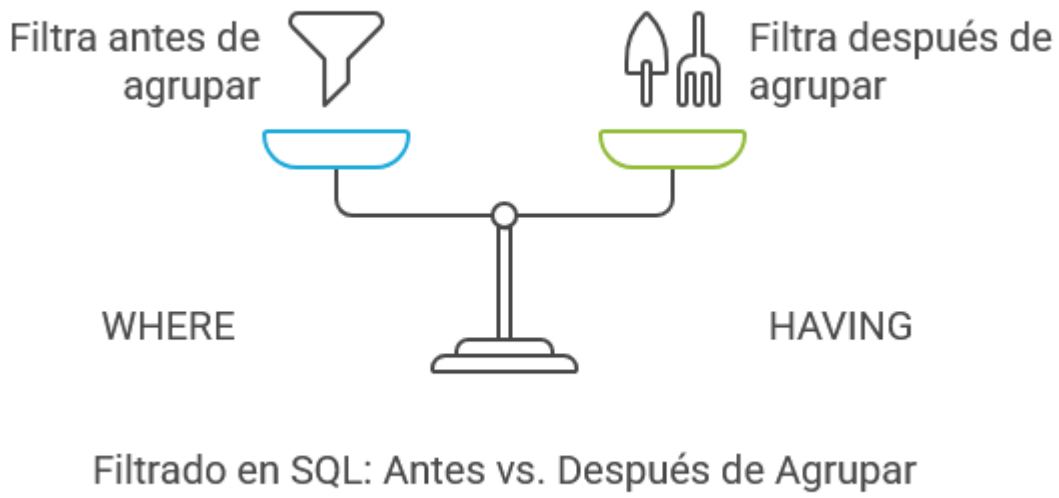
5. Ordenando Resultados con ORDER BY:

- La cláusula ORDER BY especifica el orden en el que se mostrarán los resultados de una consulta SELECT.
- "Especifica el orden de los resultados de la sentencia SELECT"
- "Ordena los resultados de una consulta."
- Se puede ordenar por una o más columnas.
- "Se puede realizar el ordenamiento por una o mas columnas"
- No es válida para las vistas.
- "Esta cláusula es invalida para las vistas"
- El orden puede ser ascendente (ASC, por defecto) o descendente (DESC).
- "Se puede determinar que el orden sea Ascendente o Descendente"
- Sintaxis: [ORDER BY { order_by_expression [ASC | DESC] } [,...n]

6. Agrupando y Resumiendo Datos:

- **Funciones Agregadas:** Operan sobre conjuntos de filas para devolver un único valor por grupo.
- "Las funciones de agrupación operan sobre conjuntos de filas para dar un resultado por grupo. Es por eso que se llaman de agrupación, porque agrupa un conjunto de registros devolviendo un único resultado final."
- Funciones mencionadas: AVG, MAX, MIN, SUM, COUNT, COUNT_BIG, VAR, VARP, STDEV, STDEVP.
- **COUNT():** Devuelve la cantidad de registros. COUNT(*) cuenta todas las filas, mientras que COUNT(nombre_columna) cuenta solo las filas con valores no nulos en la columna especificada.
- Ejemplo: "select count(*) from ventas” devolverá el numero entero 6 como resultado.
- **AVG():** Devuelve el promedio de los valores numéricos de una columna.
- Ejemplo: "select AVG(monto) from ventas”
- **MIN():** Devuelve el valor mínimo de una columna.
- Ejemplo: "select MIN(monto) from ventas”
- **MAX():** Devuelve el valor máximo de una columna.
- Ejemplo: "select MAX(monto) from ventas”
- **SUM():** Devuelve la suma de los valores numéricos de una columna.
- Ejemplo: "select SUM(monto) from ventas”
- Los valores NULL son ignorados por las funciones de agregación (excepto COUNT(*)).
- Ejemplo: "select count(CONCEPTOID) from ventas” El resultado será: 5 (asumiendo un valor NULL en esa columna).
- **Cláusula GROUP BY:** Agrupa las filas de una consulta en conjuntos basados en los valores de una o más columnas.
- "SQL nos permite agrupar las filas resultado de una consulta en conjuntos y aplicar funciones sobre esos conjuntos de filas. En la cláusula GROUP BY se colocan las columnas por las que vamos a agrupar."
- Se utiliza cuando se seleccionan múltiples columnas y al menos una función agregada. Todas las columnas no agregadas en la lista SELECT deben estar en la cláusula GROUP BY.
- "Cuando esto sucede, necesitamos GROUP BY todas las otras columnas seleccionadas, es decir, todas las columnas excepto aquella(s) que se operan por un operador aritmético."
- Ejemplo: "Select producto, sum(monto) as total from ventas GROUP BY producto”
- Características importantes:
- Especifica los grupos para las filas de salida.
- Organiza los resúmenes de datos agrupados.
- Si se usan funciones agregadas en el SELECT, GROUP BY calcula un resumen por cada grupo.
- Los valores NULL en las columnas de GROUP BY se agrupan.
- GROUP BY puede referirse a columnas no incluidas en el SELECT (siempre que haya una función agregada).
- Sintaxis: [GROUP BY [ALL] group_by_expression [,...n] [WITH { CUBE | ROLLUP }]] (El argumento ALL incluye todos los grupos, incluso aquellos sin filas que cumplan la condición WHERE, pero no se puede usar con CUBE o ROLLUP).
- El orden de los grupos resultantes no está garantizado sin la cláusula ORDER BY.
- **Cláusula HAVING:** Filtra los grupos creados por la cláusula GROUP BY basándose en una condición.
- "La consulta SQL HAVING es utilizada junto con SELECT para especificar una condición de búsqueda para un grupo."
- Funciona de manera similar a WHERE, pero opera sobre grupos en lugar de filas individuales.

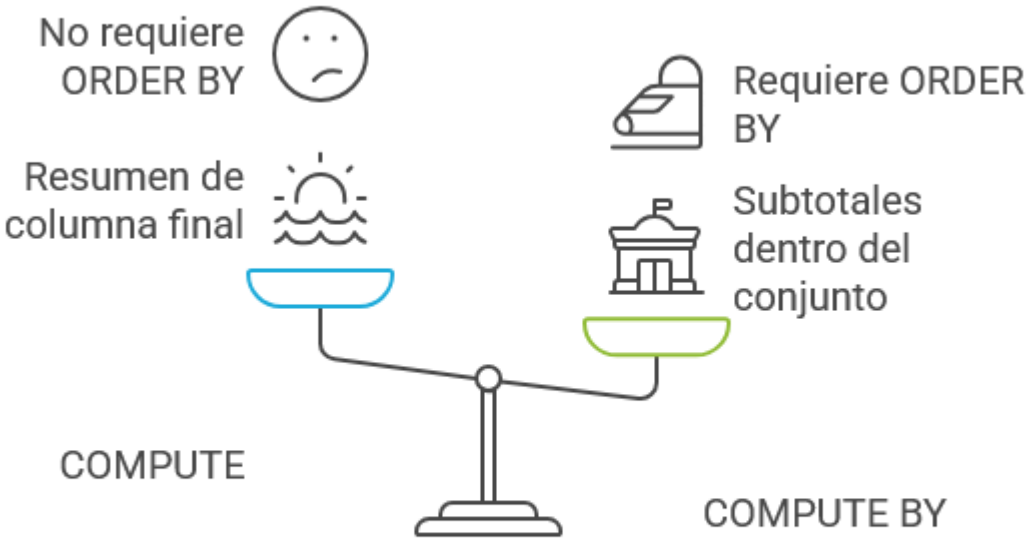
- "HAVING se comporta como WHERE, pero se aplica a grupos (las filas o tuplas en el conjunto de resultados representan grupos). La cláusula WHERE se aplica a filas o tuplas individuales, NO a grupos."
- Se utiliza para filtrar resultados *después* de la agrupación.
- "O sea HAVING se utiliza para filtrar filas de un resultado ya agrupado a través de la cláusula GROUP BY."
- Ejemplo: "Select producto, sum(monto) as total from ventas GROUP BY producto HAVING sum(monto) >= 500”.
- Características:
- Permite filtrar grupos generados por GROUP BY.
- Excluye grupos que no cumplen la condición de búsqueda.
- Las columnas en la cláusula HAVING deben retornar un solo valor (generalmente a través de una función agregada).
- Diferencia clave con WHERE: WHERE filtra filas antes de la agrupación, HAVING filtra grupos después de la agrupación.
- GROUP BY ALL muestra todos los grupos, incluso aquellos excluidos por WHERE.



Filtrado en SQL: Antes vs. Después de Agrupar

7. Cláusulas COMPUTE y COMPUTE BY:

- **COMPUTE:** Genera totales que aparecen como columnas de resumen adicionales al final del conjunto de resultados.
- "COMPUTE genera totales que aparecen como columnas de resumen adicionales al final del conjunto de resultados."
- Sintaxis: [COMPUTE { { AVG | COUNT | MAX | MIN | STDEV | STDEVP | VAR | VARP | SUM } (expression) } [,...n] [BY expression [,...n]]]
- Las funciones agregadas utilizadas son las mismas que las funciones de agrupación.
- **COMPUTE BY:** Genera interrupciones de control y subtotales dentro del conjunto de resultados. Requiere el uso de ORDER BY.
- "Cuando se utiliza COMPUTE BY, genera interrupciones de control y subtotales en el"
- "Puede especificar COMPUTE BY y COMPUTE en la misma consulta."
- Las expresiones en COMPUTE BY deben ser idénticas o un subconjunto de las expresiones en ORDER BY, y en el mismo orden.
- **Restricciones de COMPUTE y COMPUTE BY:**
- Si se usa COMPUTE BY, se debe usar ORDER BY.
- Las expresiones en COMPUTE BY deben coincidir o ser un subconjunto de las de ORDER BY.
- Omiten los valores NULL.
- No se permite DISTINCT en las funciones agregadas de filas cuando se usa COMPUTE.
- No se puede usar COMPUTE en una instrucción SELECT INTO.
- No se puede usar COMPUTE si la instrucción SELECT forma parte de una instrucción DECLARE CURSOR.



Comparando Resúmenes y Subtotales en SQL

Conclusiones:

El material revisado proporciona una base sólida para comprender cómo filtrar, ordenar, agrupar y resumir datos en SQL. La correcta utilización de las cláusulas WHERE, ORDER BY, GROUP BY y HAVING, junto con las funciones de agregación, permite realizar consultas complejas y obtener información valiosa de las bases de datos. Es crucial comprender las diferencias entre WHERE y HAVING, así como el manejo de los valores NULL para escribir consultas efectivas y precisas. Las cláusulas COMPUTE y COMPUTE BY ofrecen funcionalidades adicionales para generar resúmenes y subtotales dentro de los resultados, aunque con ciertas restricciones en su uso.