Teoría de Bases de datos
Tarea

Estudiante: Pérez Gutiérrez Sandra Susana
Grupo: 1
4 de enero de 2021
sandraconnors271198@gmail.com

Semestre 2021-1

Tarea de Bases de datos.

Índice

1. Select

La sentencia SELECT es la base para realizar consultas en cualquier base de datos de lenguaje de consulta estructurado (SQL).

1.1. Sintaxis

$$SELECT [DISTINCT] select\_list$$
$$FROM from\_clause$$
$$[WHERE search\_condition]$$
$$[GROUP BY column, column$$
$$[HAVING search\_condition]]$$
$$[ORDER BY column, column]$$

En donde:

- select_list es la lista de columnas especificada en la solicitud.
- FROM from_clause es la lista de tablas de la solicitud. Puede incluir determinada información para la solicitud.
- WHERE search_condition especifica cualquier combinación de condiciones para formar una prueba condicional. Una cláusula WHERE actúa como filtro que le permite restringir una solicitud para obtener los resultados que responden a una pregunta concreta. Junto a las columnas que seleccione, los filtros determinan lo que contendrán los resultados.
- GROUP BY column , column especifica una columna (o alias) que pertenece a una tabla definida en el origen de datos.
- HAVING search_condition especifica cualquier combinación de condiciones para formar una prueba condicional. La sintaxis es idéntica a la de la sentencia WHERE.
- ORDER BY column , column especifica las columnas por las que se ordenarán los resultados.

1.2. Casos de uso

Se puede usar select para:

- Recuperar filas y columnas.
- Con encabezados de columna y cálculos.
- Hacer subconsultas correlacionadas.
- Crear tablas con SELECT INTO.
- Usar DISTINCT con SELECT para evitar la recuperación de títulos duplicados.

1.3. Ejemplos

En el primer ejemplo se muestran consultas que son semánticamente equivalentes para demostrar la diferencia entre el uso de la palabra clave EXISTS y la palabra clave IN. Ambos son ejemplos de subconsultas válidas que recuperan una instancia de cada nombre de producto cuyo modelo es un jersey de manga larga con logotipo y cuyos números de ProductModelID coinciden en las tablas Product y ProductModel.

```
    USE AdventureWorks2012;
GO
SELECT DISTINCT Name
FROM Production.Product AS p
WHERE EXISTS
(SELECT *
FROM Production.ProductModel AS pm
WHERE p.ProductModelID = pm.ProductModelID
AND pm.Name LIKE 'Long-Sleeve Logo Jersey %');
GO
```

```
    – OR
USE AdventureWorks2012;
GO
SELECT DISTINCT Name
FROM Production.Product
WHERE ProductModelID IN
(SELECT ProductModelID
FROM Production.ProductModel AS pm
WHERE p.ProductModelID = pm.ProductModelID
AND Name LIKE 'Long-Sleeve Logo Jersey %');
GO
```

2. From

El from es un comando que se usa para especificar la tabla en la cual se operará una consulta o borrado.

2.1. Sintaxis

Su sintaxis es:

```
SELECT select_list FROM from_clause;
```

o

```
DELETE FROM from_clause
WHERE condition';
```

En donde:

- select_list es la lista de columnas especificada en la solicitud.

- from_clause es la tabla a operar.
- condition son la condición a cumplir para la operación de borrado
- WHERE especifica cualquier combinación de condiciones para formar una prueba condicional. Una cláusula WHERE actúa como filtro que le permite restringir una solicitud para obtener los resultados que responden a una pregunta concreta.

2.2. Casos de uso

Se debe de usar select para:

- Especificar la tabla a operar.
- Especificar la acción de borrado de la tbla, de otra manera la información se perderá.

2.3. Ejemplos

Consulta el contenido de las columnas CustomerName y City de la tabla Customers.

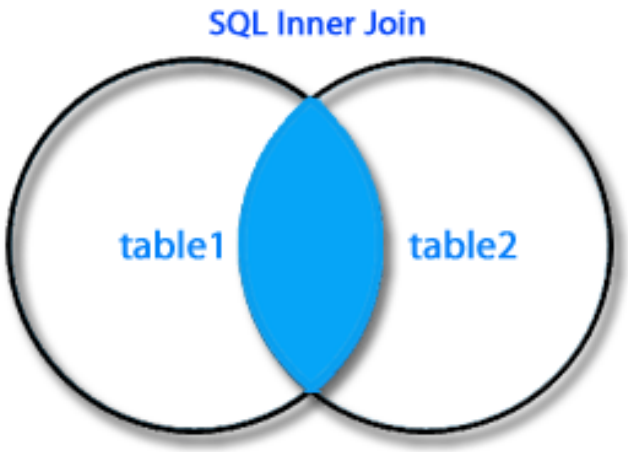
```
SELECT CustomerName, City FROM Customers;
```

Borra la información de la tabla especificada con el FROM.

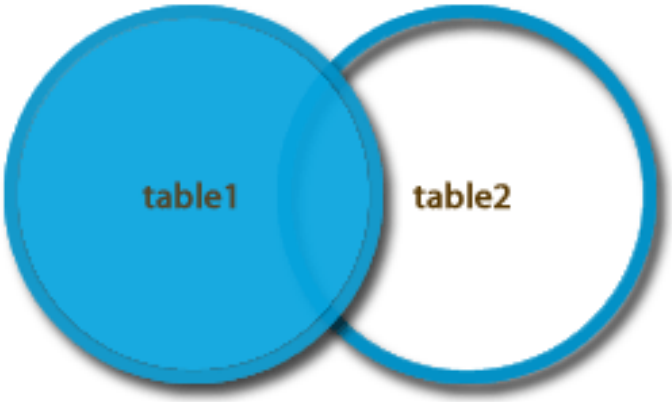
```
DELETE FROM Customers
WHERE CustomerName='Alfreds Futterkiste';
```

3. Join

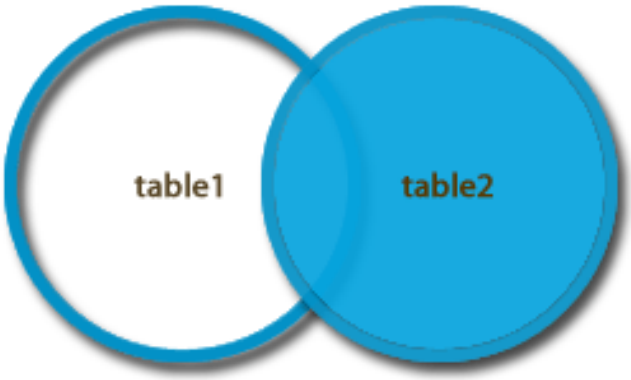
El comando INNER JOIN devuelve filas que tienen valores coincidentes en ambas tablas. INNER JOIN selecciona todas las filas de las dos columnas siempre y cuando haya una coincidencia entre las columnas en ambas tablas. Es el tipo de JOIN más común.



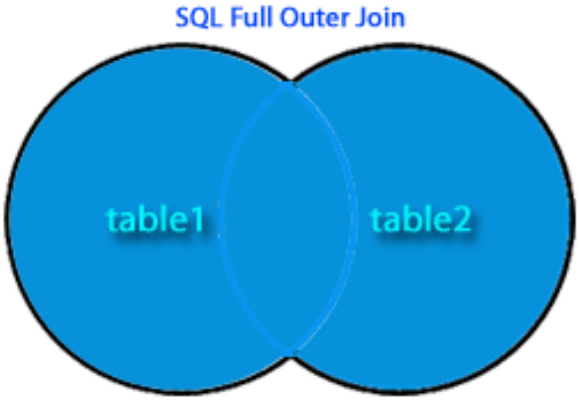
LEFT JOIN mantiene todas las filas de la tabla izquierda (la tabla1). Las filas de la tabla derecha se mostrarán si hay una coincidencia con las de la izquierda. Si existen valores en la tabla izquierda pero no en la tabla derecha, ésta mostrará null.



Es igual que LEFT JOIN pero al revés. Ahora se mantienen todas las filas de la tabla derecha (tabla2). Las filas de la tabla izquierda se mostrarán si hay una coincidencia con las de la derecha. Si existen valores en la tabla derecha pero no en la tabla izquierda, ésta se mostrará null.



OUTER JOIN o FULL OUTER JOIN devuelve todas las filas de la tabla izquierda (tabla1) y de la tabla derecha (tabla2). Combina el resultado de los joins LEFT y RIGHT. Aparecerá null en cada una de las tablas alternativamente cuando no haya una coincidencia.



3.1. Sintaxis

- INNER JOIN:
  Su sintaxis es:
  SELECT nombreColumna(s)
  FROM tabla1
  INNER JOIN tabla2
  ON tabla1.nombreColumna=table2.nombreColumna;

  En donde:

  - nombreColumna es la lista de columnas especificada en la solicitud.
  - from es la tabla a operar.
  - ON especifica una condición de unión. Esto nos permite especificar condiciones de combinación separadas de cualquier condición de búsqueda o filtro en la cláusula WHERE.

- LEFT JOIN:
  Su sintaxis es:
  SELECT nombreColumna(s)
  FROM tabla1
  LEFT JOIN tabla2
  ON tabla1.nombreColumna=tabla2.nombreColumna;

  En donde:

  - nombreColumna es la lista de columnas especificada en la solicitud.
  - from es la tabla a operar.
  - ON especifica una condición de unión. Esto nos permite especificar condiciones de combinación separadas de cualquier condición de búsqueda o filtro en la cláusula WHERE.

- RIGHT JOIN:
  Su sintaxis es:

```
SELECT nombreColumna(s)
FROM tabla1
RIGHT JOIN tabla2
ON tabla1.nombreColumna=tabla2.nombreColumna;
```

En donde:

- nombreColumna es la lista de columnas especificada en la solicitud.
- from es la tabla a operar.
- ON especifica una condición de unión. Esto nos permite especificar condiciones de combinación separadas de cualquier condición de búsqueda o filtro en la cláusula WHERE.

- OUTER JOIN o FULL OUTER JOIN:
  Su sintaxis es:

```
SELECT nombreColumna(s)
FROM tabla1
OUTER JOIN tabla2
ON tabla1.nombreColumna=tabla2.nombreColumna;
```

En donde:

- nombreColumna es la lista de columnas especificada en la solicitud.
- from es la tabla a operar.
- ON especifica una condición de unión. Esto nos permite especificar condiciones de combinación separadas de cualquier condición de búsqueda o filtro en la cláusula WHERE.

3.2. Casos de uso

Los JOINs en SQL sirven para combinar filas de dos o más tablas basándose en un campo común entre ellas, devolviendo por tanto datos de diferentes tablas. Un JOIN se produce cuando dos o más tablas se juntan en una sentencia SQL.

Existen más tipos de joins en SQL que los que aquí se explican, como CROSS JOIN, O SELF JOIN, pero no todos ellos están soportados por todos los sistemas de bases de datos. Los más importantes son los siguientes:

- INNER JOIN: Devuelve todas las filas cuando hay al menos una coincidencia en ambas tablas.
- LEFT JOIN: Devuelve todas las filas de la tabla de la izquierda, y las filas coincidentes de la tabla de la derecha.
- RIGHT JOIN: Devuelve todas las filas de la tabla de la derecha, y las filas coincidentes de la tabla de la izquierda.
- OUTER JOIN: Devuelve todas las filas de las dos tablas, la izquierda y la derecha. También se llama FULL OUTER JOIN.

3.3. Ejemplos

El siguiente SQL selecciona todos los pedidos con información del cliente.

```
SELECT Orders.OrderID, Customers.CustomerName
FROM Orders
INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID;
```

El siguiente SQL devolverá a todos los empleados y cualquier pedido que hayan realizado:

```
SELECT Orders.OrderID, Employees.LastName, Employees.FirstName
FROM Orders
RIGHT JOIN Employees ON Orders.EmployeeID = Employees.EmployeeID
ORDER BY Orders.OrderID;
```

4. Where

El comando WHERE filtra un conjunto de resultados para incluir solo los registros que cumplen una condición específica. El from es un comando que se usa para especificar la tabla en la cual se operará una consulta o borrado.

4.1. Sintaxis

Su sintaxis es:

```
SELECT select_list FROM from_clause;
WHERE condition';
```

En donde:

- select_list es la lista de columnas especificada en la solicitud.
- condition son la condición a cumplir para la operación de borrado
- WHERE especifica cualquier combinación de condiciones para formar una prueba condicional. Una cláusula WHERE actúa como filtro que le permite restringir una solicitud para obtener los resultados que responden a una pregunta concreta.

4.2. Casos de uso

Es una clausula para especificar condiciones que deben de cumplir las tablas a operar y sus atributos.

4.3. Ejemplos

Borra la información de la tabla especificada con el FROM.

```
DELETE FROM Customers
WHERE CustomerName='Alfreds Futterkiste';
```

5. Having

El comando HAVING se usa en lugar de WHERE con funciones agregadas.Esta cláusula condicional devuelve filas donde los resultados de la función agregada coinciden solo con las condiciones dadas. Se agregó en el SQL porque la cláusula WHERE no se puede combinar con resultados agregados, por lo que tiene un propósito diferente. El propósito principal de la Cláusula WHERE es tratar con registros individuales o no agregados.

5.1. Sintaxis

Su sintaxis es:
[HAVING ≺ search condition≻]
En donde:

- ≺ search_condition ≻ Especifica uno o más predicados que los grupos o agregados deben satisfacer.

5.2. Casos de uso

Se debe de usar select para:

- La cláusula HAVING siempre se utiliza en combinación con la cláusula GROUP BY.
- La cláusula HAVING restringe los datos de los registros de grupo en lugar de los registros individuales.
- WHERE y HAVING se pueden utilizar en una sola consulta.

5.3. Ejemplos

El siguiente SQL enumera el número de clientes en cada país. Incluya solo países con más de 5 clientes:

```
SELECT COUNT(CustomerID), Country
FROM Customers
GROUP BY Country
HAVING COUNT(CustomerID) ≻ 5;
```

6. Correlaciones

Devuelve el coeficiente de correlación de los pares de valores X-Y evaluados sobre un conjunto.

6.1. Sintaxis

Su sintaxis es:
Correlation(Set_Expression, Numeric_Expression_y [,Numeric_Expression_x])
En donde:

- Set_Expression:
  Expresión MDX (Expresiones multidimensionales) válida que devuelve un conjunto.

- Numeric_Expression_y:
  Expresión numérica válida que suele ser una expresión MDX de las coordenadas de celdas que devuelven un número que representa los valores del eje Y.

- Numeric_Expression_x:
  Expresión numérica válida que suele ser una expresión MDX de las coordenadas de celdas que devuelven un número que representa los valores del eje X.

6.2. Casos de uso

La función de correlación calcula el coeficiente de correlación de dos pares de valores al evaluar primero el conjunto especificado con la primera expresión numérica para obtener los valores del eje y. A continuación, la función evalúa el conjunto especificado con la segunda expresión numérica, si está presente, para obtener el conjunto de valores del eje x. Si no se especifica la segunda expresión numérica, la función utiliza el contexto actual de las celdas del conjunto especificado como valores para el eje X.

6.3. Ejemplos

Encuentre todos los empleados que ganan más que el salario promedio en su departamento.

```
SELECT last_name, salary, department_id
FROM employees outer
WHERE salary ≻
(SELECT AVG(salary)
FROM employees
WHERE department_id =
outer.department_id);
```

7. Referencias

1. Microsoft, "*Correlation (MDX)*", `https://cutt.ly/Wjsi330`, 4 de enero de 2021.
2. GeeksforGeeks, "*SQL Correlated Subqueries*", `https://cutt.ly/Ljsoiwo`, 4 de enero de 2021.
3. Weschools, "*SQL HAVING Keyword*", `https://www.w3schools.com/sql/sql_ref_having.asp`, 4 de enero de 2021.
4. Diego Lazaro, "*Principales tipos de JOINS en SQL*", `https://cutt.ly/cjsoUTf`, 4 de enero de 2021.
5. Oracle, "*Sintaxis y Semántica de SQL*", `https://cutt.ly/ijsoGVh`, 4 de enero de 2021.