

Facultad: Ingeniería
Escuela: Computación
Asignatura: Base de datos I

Tema: Uso de sentencias SQL

Objetivo

- Modificar, eliminar e insertar registros en una tabla
- Listar los registros de una tabla utilizando la sentencia select

Materiales y Equipo

- Computadora con SQL Server 2008.
- Guía Número 5

Introducción

En esta guía terminaremos de examinar los elementos que pueden ir en una instrucción SELECT así como las instrucciones que nos ayudan a definir una estructura de una base de datos y las instrucciones para poder realizar modificaciones a los objetos de la base de datos.

Sintaxis general de la consulta SELECT

```
SELECT <lista de columnas>  
[FROM <tabla(s) de origen >]  
[WHERE <condición restrictiva >]  
[GROUP BY <nombre de columna o expresión que utiliza una  
columna en la lista de selección>]  
[HAVING <condición restrictiva basadas en los resultados de  
GROUP BY>]  
[ORDER BY <listas de columna>]
```

Ejemplo:

```
USE NORTHWIND  
SELECT OrderID, Quantity As [Sin nombre de columna]  
FROM [Order Details]  
WHERE OrderID BETWEEN 11000 AND 11002
```

Nota: Observe el uso de corchetes en esta consulta. Se debe de utilizar corchetes si el nombre de un objeto (en este caso una tabla) tiene espacios de por medio en él, tenemos que delimitar el nombre utilizando corchetes o comillas simples, lo que permite a SQL Server saber dónde empieza y dónde termina el nombre.

Resultado de la consulta anterior es el siguiente:

<u>ORDER ID</u>	<u>Sin nombre de columna</u>
11000	25
11000	30
11000	30
11001	60
11001	25
11001	25
11001	6
11002	56
11002	15
11002	24
11002	40

Aunque se ha solicitado sólo tres pedidos, se están viendo cada línea individual de detalle del pedido. Podríamos utilizar una calculadora o podríamos utilizar la cláusula GROUP BY con un agregado (en este caso vamos a utilizar **SUM ()**) para obtener el total deseado.

```
SELECT OrderID, SUM (Quantity) As [Sin nombre de columna]
FROM [Order Details]
WHERE OrderID BETWEEN 11000 AND 11002
GROUP BY OrderID
```

Así obtiene lo que se está buscando:

<u>ORDER ID</u>	<u>Sin nombre de columna</u>
11000	85
11001	116
11002	135

Lo que hizo fue agrupar en una suma los ORDER ID de los resultados, es decir en la primera consulta aparecían

ORDER ID	Sin nombre de columna
11000	25
11000	30
11000	30

Con la función sum lo que hace es agrupar(en este caso por el ORDER ID) y realizar la suma, por eso da el resultado

ORDER ID	Sin nombre de columna
11000	85

TIPS: Puedo agregarle un título a la columna sin nombre con AS Como es de suponer, la función **SUM** devuelve totales; ¿pero totales de qué? Si no hubiésemos suministrado la cláusula **GROUP BY**, la función **SUM** habría sumado todos los valores de todas las filas para la columna con nombre. Sin embargo, en este caso hemos suministrado una cláusula **GROUP BY** y, por tanto, el total proporcionado por la función **SUM** es el total de cada grupo.

También podemos agrupar basándonos en múltiples columnas. Para ello, sólo tenemos que añadir una coma y el nombre de la siguiente columna.

Funciones de agregado

Las funciones de agregado realizan un cálculo sobre un conjunto de valores y devuelven un solo valor. Con la excepción de COUNT, las funciones de agregado omiten los valores NULL. Las funciones de agregado se suelen utilizar con la cláusula GROUP BY de la instrucción SELECT.

COUNT:

Devuelve el número de elementos de un grupo (conjunto de resultados).

SELECT COUNT(*) FROM Employees

	(No column name)
1	9

AVG

Devuelve la media de los valores de un grupo. Los valores nulos se pasan por alto al probar a ejecutar la misma consulta que ejecutamos anteriormente, pero ahora vamos a modificarla para que devuelva el promedio de la cantidad por pedido en lugar del total de cada pedido:

```
SELECT OrderID, AVG (Quantity) AS promedio
FROM [Order Details]
WHERE OrderID BETWEEN 11000 AND 11002
GROUP BY OrderID
```

El resultado es:

OrderID	promedio
11000	28
11001	29
11002	33

MAX / MIN:

MAX devuelve el valor máximo de la expresión o del valor de una columna y **MIN** lo contrario.

```
SELECT OrderID, MIN(Quantity) AS promedio
FROM [Order Details]
WHERE OrderID BETWEEN 11000 AND 11002
GROUP BY OrderID
```

NOTA: En realidad todas las funciones de agregado ignoran los valores NULL excepto COUNT (*).

Agrupar condiciones con la cláusula **HAVING**. Hasta el momento, hemos aplicado todas nuestras condiciones a filas específicas. Si una determinada columna en una fila no tiene un valor específico o no se encuentra dentro de un rango de valores, se omitirá toda la fila, y todo ello antes de pensar siquiera en las agrupaciones.

¿Qué pasaría si deseáramos agrupar condiciones? Es decir, ¿Qué pasaría si deseáramos que todas las filas se agregasen a un grupo pero sólo cuando los grupos se hubiesen acumulado completamente estaríamos preparados para aplicar la condición? Bueno, aquí es donde entra en acción la cláusula **HAVING**.

La cláusula **HAVING** sólo se utiliza si también existe una cláusula **GROUP BY** en la consulta. Mientras la cláusula **WHERE** se aplica a todas las filas antes incluso de tener la oportunidad de convertirse en parte de un grupo, la cláusula **HAVING** se aplica al valor agregado de dicho grupo.

Ejemplo:

Primero vamos a realizar un ejemplo que no lleve la cláusula HAVING y después vamos a utilizar el mismo ejemplo, pero ahora con la cláusula HAVING.

Sin HAVING

```
SELECT OrderID, SUM(Quantity) AS TOTAL
FROM [Order Details]
GROUP BY OrderID
```

El resultado es:

OrderID	TOTAL
10248	27
10249	49
10250	60
10251	41
10252	105
10253	102
10254	57
10255	110
10256	27
10257	46
10258	121
10259	11
10260	102
10261	40
10262	29
10263	184
10264	60
10265	50
10266	12
10267	135
10268	14
10269	80

(830
filas afectadas)

Lamentablemente, es bastante difícil analizar una lista tan larga. Por tanto, vamos a dejar que SQL Server reduzca esta lista para ayudarnos a realizar nuestro análisis. Supongamos que sólo estamos interesados en cantidades de pedidos más grandes. ¿Podemos modificar la consulta para que devuelva la misma información pero limitándose a los pedidos cuya cantidad total esté por encima de 300? La respuesta es sí, y es tan fácil como añadir la cláusula HAVING.

Con HAVING

```
SELECT OrderID, SUM(Quantity) AS TOTAL
FROM [Order Details]
GROUP BY OrderID
HAVING SUM(Quantity) > 300
```

El resultado es:

	OrderID	TOTAL
1	10895	346
2	11030	330

Como puede comprobar, podemos reducir rápidamente la lista hasta los elementos que nos interesan.

INSTRUCCION SELECT

Ejemplo:

```
USE Northwind
SELECT * FROM Employees
```

En la instrucción anterior la primera línea indica que utilizaremos la base de datos **Northwind**.

La segunda línea se ha pedido seleccionar información (SELECT puede pensar en ello como en pedir que se muestre o consulte información). El asterisco * podría parecer algo extraño, pero realmente funciona igual que en cualquier otro sitio: es un **comodín**.

Cuando escribimos **SELECT *** estamos indicando que deseamos seleccionar todas las columnas de la tabla (En este caso la tabla **Employees**.) **FROM** indica que hemos terminado de especificar los elementos que deseamos ver y estamos a punto de indicar el origen de la información (el origen es la tabla **Employees**).

Ejemplo:

```
USE AdventureWorks
SELECT FirstName, Lastname, EmailAddress FROM Person.Contact
```

En esta consulta solamente se piden los campos **FirstName**, **Lastname** y **EmailAddress** de la table *Person.Contact*

LA CLAUSULA WHERE.

Esta cláusula nos permite colocar condiciones sobre los resultados deseados.

Los ejemplos que hemos visto hasta el momento son consultas **SELECT** sin restricciones en el sentido de que se han incluido en el resultado todas las filas de la tabla especificada. Las consultas sin restricciones son muy útiles para rellenar cuadros de lista y cuadros combinados y en otros escenarios en los que intentamos proporcionar un listado de dominio.

Ejemplo:

Ahora vamos a buscar información más específica de la tabla *Person.Address* de la base de datos *AdventureWorks*:

```
USE AdventureWorks
SELECT City, AddressLine1, PostalCode, StateProvinceID
FROM Person.Address
WHERE StateProvinceID = 74
```

Los primeros 3 resultados son:

City	AddressLine1	PostalCode
Nevada	2487 Riverside Drive	84407
Salt Lake City	683 Larch Ct.	84101
Salt Lake City	6119 11th	84101

Ejemplo:

Esta sentencia listara productos que tengan un precio único mayor a 60

```
USE NORTHWIND
SELECT * FROM dbo.Products WHERE UnitPrice >60.00
```

Esta sentencia listara todas los campos de la tabla categoria donde el nombre de categoria tenga coincidencia con la palabra produ(en cualquier parte del nombre)

```
SELECT * FROM dbo.Categories WHERE CategoryName LIKE '%produ%'
```

OPERADORES QUE PODEMOS USAR CON LA CLAUSULA WHERE

Operador	Efecto
=, >, <, >=, <=, <>, !=, >, <	<p>Operadores de comparación estándar: funcionan como lo hacen en cualquier lenguaje de programación con un par de puntos importantes:</p> <ol style="list-style-type: none"> 1) Lo que constituye "mayor que", "menor que" e "igual a" puede cambiar, dependiendo del orden de comparación seleccionado. Por ejemplo, "ROMEY" = "romey" es verdadero cuando se ha seleccionado una ordenación en la que no se tienen en cuenta las mayúsculas y minúsculas. Pero "ROMEY" <> "romey" es una situación en la que si se distinguen las mayúsculas de las minúsculas. 2) Tanto != como <> significan <u>no igual a</u>. 3) >, < significan <u>no menor que</u> y <u>no mayor que</u> respectivamente.

Insertando datos en una Tabla.

Operador	Efecto
AND, OR, NOT	Valores lógicos booleanos estándar. Puede utilizarlos para combinar múltiples condiciones en una cláusula WHERE. Con respecto al orden NOT se evalúa primero, posteriormente AND y por último OR. Si tiene que cambiar el orden de evaluación, puede utilizar paréntesis. Tenga en cuenta que no se admite XOR.
BETWEEN	<p>La comparación es verdadera, TRUE, si el primer valor se encuentra comprendido entre el segundo y tercer valor, ambos inclusive. Es la equivalencia funcional de $A \geq B \text{ AND } A \leq C$. cualquier valor especificado puede ser un nombre de columna, una variable o un literal.</p> <p>Ej: <columna> BETWEEN 1 AND 5</p>
LIKE	<p>Utiliza los caracteres % y _ como comodines.</p> <ol style="list-style-type: none"> 1) % indica que un valor de cualquier longitud puede reemplazar al carácter % (Ej: LIKE "ROM%"). 2) _ indica que cualquier otro carácter puede reemplazar al carácter _. 3) La escritura de caracteres entre corchetes, [] indica que cualquier carácter único dentro de dichos corchetes es correcto ([a-c] significa que a, b y c son valores correctos. [ab] indica que los valores de a o b son correctos). 4) ^ funciona como operador NOT (indicando que se excluye el siguiente carácter).

El comando **INSERT** inserta una nueva fila en la tabla, al llenar las columnas con valores específicos.

Sintaxis:

INSERT INTO <tabla> [(lista de columnas)] **VALUES** (valores_de_datos)

Ejemplo:

INSERT INTO demo(cod_sucursal,nombre,apellido,telefono) **VALUES**
(14,'Julia','Morales','22552100')

Eliminando datos de una tabla.

El comando **DELETE** elimina filas de una tabla o vista, que satisfagan una condición específica.

Sintaxis:

DELETE FROM table_name **WHERE** condición.

Ejemplo:

DELETE FROM demo **where** nombre='julia'

Si no se incluye la cláusula **WHERE**, se eliminarán todas las filas en la tabla indicada.

Actualizando datos de una tabla.

Este comando permite la actualización de uno o más campos de una fila o grupo de filas de una tabla o vista. Las sentencias **UPDATE** se utilizan para modificar datos existentes.

Sintaxis:

UPDATE nombre_tabla | nombre_vista **SET** (nombre_columna = expresion | **DEFAULT** | **NULL**)
WHERE (condición)

Ejemplo:

UPDATE demo **SET** cod_sucursal = '55' **WHERE** apellido= 'Morales'

Set: especifica la lista de columnas que se actualizarán.

Procedimiento

Ejercicio No 1

Realizar lo siguiente:

1. Crear una Base de datos con el siguiente nombre: **SUCARNET_PRACTICA3**.

Nota: recuerde que SUCARNET se refiere a su código de carné como estudiante de la UDB

2. Poner en uso la base de datos que acaba de crear con el comando **USE**.

Ejemplo: **USE SUCARNET_PRACTICA3**

3. Crear las siguientes tablas dentro de la base de datos:

TABLA : ESTUDIANTES		
CAMPO	TIPO DE DATO	LONGITUD
IdEstudiante	INT	
Nombres	VARCHAR	25
Apellidos	VARCHAR	25
Direccion	VARCHAR	50
Telefono	VARCHAR	8
sexo	char	1

TABLA : MATERIAS		
CAMPO	TIPO DE DATO	LONGITUD
IdMateria	INT	
Materia	VARCHAR	30

TABLA : NOTAS		
CAMPO	TIPO DE DATO	LONGITUD
IdEstudiante	INT	

IdMateria	INT	
Nota	DECIMAL	(10,2)

4. Insertar los siguientes datos a las tablas creadas anteriormente.

TABLA ESTUDIANTES					
IdEstudiante	Nombres	Apellidos	Dirección	Teléfono	Sexo
01	Maria	HERNANDEZ	COL. SANTA ISABEL	2254212	F
02	OSCAR	MEJIA	FINAL 4 CALLE OTE.	2609834	M
03	HILARIO	URRUTIA	Fnal cl progreso	2907834	M
04	JOSE JOSE	QUEZADA	Mejicanos	23663322	M
05	ELIAS ALFREDO	URRUTIA	Santa Tecla	2778934	M

TABLA MATERIAS	
IdMateria	Materias
111	Base de datos II
114	Ingenieria del Software
115	SQL SERVER

TABLA NOTAS		
IdEstudiante	IdMateria	Nota
01	111	7
01	114	6.0
01	115	4
02	111	6
02	114	10
02	115	8.0

5. Eliminar registros.

Eliminar los registros de la tabla NOTAS; su IdMateria sea igual a 111

DELETE FROM NOTAS **WHERE** IdMateria = 111

6. Actualizar registros.

Modificar los registros de la tabla NOTAS para el IdEstudiante = 2 y IdMateria = 115, la nota a modificar es 8.0 a 9.0

UPDATE NOTAS **SET** Nota=9.0 **WHERE** IdEstudiante =2 **AND** IdMateria=115

7-Modificar el nombre de la materia Ingeniería del Software por Desarrollo del Software**8- Eliminar a todos los alumnos que tengan apellido URRUTIA****9-Listar todos los estudiantes la tabla estudiantes****10-listar los apellidos y la direccion de los estudiantes que tienen id=1****ASIGNACION**

A continuación deberá realizar diferentes consultas con distintos niveles de dificultad, utilizando en algunas de ellas diferentes funciones asociadas con la sentencia SELECT:

- 1) Se desea un listado de los alumnos ordenados ascendentemente apellido
- 2) Se desea conocer cuántos estudiantes hay por cada género
- 3) Se desea conocer la nota promedio del alumno con id 1
- 4) Se desea conocer la nota mayor de las evaluaciones
- 5) De la tabla notas que estudiantes tienen promedios mayor o igual que 6

Bibliografía

Guía 5: USO DE SENTENCIAS SQL

 Hoja de cotejo: **5**

Alumno:

Máquina No:

Docente:

GL:

Fecha:

EVALUACION					
	%	1-4	5-7	8-10	Nota
CONOCIMIENTO	Del 20 al 30%	Conocimiento deficiente de los fundamentos teóricos	Conocimiento y explicación incompleta de los fundamentos teóricos	Conocimiento completo y explicación clara de los fundamentos teóricos	
APLICACIÓN DEL CONOCIMIENTO	Del 40% al 60%				
ACTITUD					
	Del 15% al 30%	No tiene actitud proactiva.	Actitud propositiva y con propuestas no aplicables al contenido de la guía.	Tiene actitud proactiva y sus propuestas son concretas.	
TOTAL	100%				