UNIVERSIDAD TECNOLÓGICA DE LA SELVA

BASE DE DATOS
UNIDAD III. LENGUAJE SQL
Parte 3.5 Consultas

Ing. Alejandro Vázquez Rodríguez

3.5 Consultas

- Identificar las cláusulas y sintaxis del DML para la generación de consultas y operaciones con los datos (select y funciones de agregado).
- Realizar consultas en Bases de Datos con el lenguaje SQL en Sistemas Gestores de Bases de Datos(SGBDs).

DML. Data Manipulation Language

- Este lenguaje manipula los datos almacenados en las tablas de la base de datos.
- Los comandos del DML del SQL los podemos dividir en dos tipos:
  - ✓ De consulta:
    - `SELECT` – extrae, selecciona, visualiza datos.
  - ✓ De actualización:
    - `INSERT` – inserta nuevos datos.
    - `UPDATE` – actualiza datos.
    - `DELETE` – borra datos.

Primero los datos

- Antes de realizar consultas con SELECT y operaciones con los datos de una o más tablas, estas deben contener datos.
- Aquí veremos la forma más simple de insertar datos en una tabla. Por supuesto, existen otros proceso para insertar grandes volúmenes de datos. Eso lo veremos en cuatrimestres más avanzados.
- Para evitar errores, los datos tipo texto como char y varchar, deben ir encerrados entre comillas dobles.
- Los datos tipo numérico como int, float, double, no llevan comillas.

Sintaxis de la sentencia INSERT en MySQL

```
INSERT [INTO] nom_tabla [(nom_columna1, nom_columna2,...)]
VALUES (valor1,valor2,valor3,...),
        (valor1,valor2,valor3,...),...
ó
INSERT [INTO] nom_tabla
SET nom_columna ={expresión | DEFAULT}, ...
ó
INSERT [INTO] nom_tabla [(nom_columna,...)]
SELECT ...
```

Ejemplos de INSERT

```
-- Aquí los valores deben ir en el mismo orden que los campos
INSERT INTO Alumnos(Matricula, Nombre, Apellido, Calificacion)
VALUES(“0931410629”,”Andrés”,”Pérez”,9.0);


-- Aquí los valores deben coincidir con el orden de los campos y tipos de datos
INSERT Alumnos VALUES(“0931410629”,”Andrés”,”Pérez”,9.0);


-- Aquí se realiza la inserción de varios registros, respetando el orden de los
-- campos sin mencionarlos.
INSERT Alumnos
VALUES(“0931410629”,”Andrés”,”Pérez”,9.0),
    (“0931410630”,”Bernardo”,”Lucas”,8.4),
    (“0931410631”,”Carlos”,”Jiménez”,7.5);
```

Ejemplos de INSERT

```
-- Aquí se llena la tabla Grupo1 con los datos de la tabla Grupos y deben coincidir
-- en el número de campos y tipos de datos.
INSERT INTO Grupo1(No_grupo, Fecha_Inicio, A_Paterno, A_Materno) SELECT NO_GRUPO,
   FECHA_INIC, A_PATERNO, A_MATERNO
     FROM Grupos
     WHERE NO_GRUPO=1;
```

Estructura básica de una consulta en SQL

- Una consulta también se le llama Query.

  SELECT $A_1, A_2, \ldots, A_n$

  FROM $r_1, r_2, \ldots, r_m$

  WHERE P

- Cada A_i representa un atributo, y cada r_i una relación(tabla). *P* es un predicado.

Estructura básica de una consulta(2)

- La consulta es equivalente a la expresión del algebra relacional:

$$\Pi\ A_1, A_2, \ldots, A_n\ (\sigma_p (r_1 \times r_2 \times \ldots \times r_m))$$

- Lo anterior se lee: Selecciona las columnas A_1, A_2, …, A_n donde se cumpla la condición P del producto cartesiano resultante de $r_1 X r_2 X \ldots X r_m$

Transformado una consulta AR a Query

- Al pasar la consulta de Algebra Relacional

  $\Pi\ A_1, A_2,\ldots,A_n\ (\sigma_p (r_1\ x\ r_2\ x\ \ldots\ x\ r_m))$

- A SQL, tenemos que:

  $\Pi\ A_1, A_2,\ldots,A_n$ equivale a SELECT

  σ_p equivale a WHERE [condición]

  p es la condición en el WHERE

  $r_1\ x\ r_2\ x\ \ldots\ x\ r_m$ equivale a FROM

Ejemplo consulta AR a SQL

- La consulta más simple es devolver todos los datos de una tabla.
- Tomando como base la tabla EMPLEADOS de la diapositiva 11 del Tema 3.4 tenemos:

π *Num_emp, Nombre, Apellido, Direccion, Ciudad, Telefono, Puesto, Fecha-nac, Salario* ***(EMPLEADOS)***

- Esta es una operación proyección, al transformarla a lenguaje SQL, queda:

```
-- Se pueden listar todos los campo o
-- la lista se reemplaza por el comodín *(asterisco)
SELECT * FROM EMPLEADOS
```

Ejemplo 2 consulta AR a SQL

- Tomando como base la tabla ALUMNOS del archivo 3.4 Ejercicios Algebra Relaciona.pdf, realizar la siguiente consulta:
  - ✓ Obtener los registros de los alumnos que estudian en PAI en el turno de la mañana.
  - ✓ $\sigma_{\text{Carrera}=\text{“PAI”} \wedge \text{Turno}=\text{“Matutino”}}(\text{ALUMNOS})$
- Debido a que solo nos piden que se cumpla una condición en los registros devueltos, esta es una operación selección, al transformarla a lenguaje SQL, queda:

```
SELECT * FROM ALUMNOS WHERE Carrera = “PAI” AND Turno = “Matutino”
```

Ejemplo 3 consulta AR a SQL

- Tomando como base el ejemplo de la diapositiva 10 del Tema 3.4 tenemos:

$$\pi_{Num\_emp,\ Nombre}\left(\sigma_{Salario<18,000}\left(EMPLEADOS\right)\right)$$

- Esta es una operación de composición de proyección y selección, al transformarla a SQL, queda:

```
SELECT Num_emp, Nombre FROM EMPLEADOS WHERE Salario < 18,000
```

Ejemplo 4 consulta AR a SQL

- ✓ Obtener el número de empleado, nombre y teléfono de aquellos que viven en Villarreal, cuyo puesto sea de supervisor y ganen $24,000.00:

$$\pi_{Num\_emp,\ Nombre,\ Telefono}(\sigma_{Ciudad = "Villarreal" \wedge Puesto = Supervisor \wedge Salario = 24{,}000.00}(EMPLEADOS))$$

- Esta es una operación de composición de proyección y selección, al transformarla a SQL, queda:

```
SELECT Num_emp, Nombre, Telefono
FROM EMPLEADOS
WHERE Ciudad = "Villarreal" AND Puesto = "Supervisor" AND Salario = 24,000.00
```

Cláusulas de una consulta SQL

- Una consulta puede constar de un máximo de 6 cláusulas, pero sólo son obligatorias las dos primeras: **SELECT** y **FROM**.
- Las cláusulas se especifican en el siguiente orden(las que están entre corchetes [...] son opcionales.)

```
SELECT <lista de atributos>
FROM <lista de tablas>
[WHERE <condiciones>]
[GROUP BY <atributo(s) de agrupación>]
[HAVING <condición de agrupación>]
[ORDER BY <lista de atributos>]
```

Sentencia SELECT

- **SELECT**
  - ✓ Identifica a las columnas que se desean en la consulta. Representa la operación Proyección(Π) del Algebra Relacional.
- **FROM**
  - ✓ Lista las tablas referidas en la consulta. Representa la operación Producto Cartesiano(X) del Algebra Relacional.
- **WHERE**
  - ✓ Especifica las condiciones para seleccionar las filas de las tablas especificadas en FROM. Representa la operación Selección(σ) del Algebra Relacional.

Cláusulas de SELECT

- **GROUP BY**
  - ✓ Agrupa a las filas que tengan un valor común, de manera lógica no física. Los atributos de agrupación deben aparecer en la cláusula SELECT.
- **HAVING**
  - ✓ Especifica una condición que deben cumplir los grupos seleccionados, no las tuplas individuales.
- **ORDER BY**
  - ✓ Ordena los valores de una o más columnas, puede ordenar los valores de manera ascendente o descendente.

Sintaxis de SELECT en MySQL

```
SELECT
[ * | DISTINCT | DISTINCTROW ]

FROM table_references
[WHERE where_definition]
[GROUP BY {col_name | expr | position}
[ASC | DESC], …]
[HAVING where_definition]
[ORDER BY {col_name | expr | position}
[ASC | DESC] , ...]
```

Operadores lógicos

- **AND**
  - ✓ La consulta se realiza si todas las condiciones se cumplen.

```
SELECT * FROM Estudiantes WHERE grado = "1" AND turno = "Vespertino"
```

- **OR**
  - ✓ La consulta se realiza si alguna o todas las condiciones se cumplen.

```
SELECT * FROM Estudiantes WHERE grado = "1" OR turno =
   "Vespertino"
```

- **NOT**
  - ✓ La consulta se realiza al negar una condición que puede cumplirse o no.

```
SELECT * FROM estudiantes WHERE nombre = "Juan" AND NOT edad =
   "20"
```

Operadores de comparación

- En SQL se utilizan los siguientes operadores de comparación: >, <, >=, <=, =, <>; la consulta se realiza al cumplirse la condición impuesta.

```
SELECT *
FROM alumnos
WHERE calificacion <= 7
```

Operadores de comparación(2)

- **BETWEEN.** La consulta se realiza si el atributo en la condición se encuentra dentro de un rango establecido o intérvalo de valores.

```
SELECT * FROM alumnos WHERE calificacion BETWEEN 7 AND 10
```

- **LIKE.** Hace coincidir los registros con un modelo especificado, se pueden utilizar caracteres comodines (%, ?,*, _ , etc.)

```
SELECT * FROM alumnos WHERE apellido LIKE "A%"
```

- Nota: El operador comodín varía de un SGBD a otro.

Operadores de comparación(3)

- **IN.** Se utiliza para recuperar registros que coinciden con una lista de valores.

```
SELECT *
FROM alumnos
WHERE grupo IN ("A", “B”)
```

- **NOT IN.** Se utiliza para recuperar registros que no coinciden con una lista de valores.

```
SELECT *
FROM alumnos
WHERE grupo NOT IN (“A”, “B”)
```

Funciones de agregado

- **SUM()**: suma los valores que contiene un atributo, estos deben ser numéricos.

```
SELECT SUM(salario) FROM empleado
```

- **COUNT()**: cuenta las filas de la tabla generada por una consulta.

```
SELECT COUNT(*) FROM empleado
```

- **MAX()**: encuentra el valor máximo de un atributo.

```
SELECT MAX(salario) FROM empleado
```

- **MIN()**: encuentra el valor mínimo de un atributo.

```
SELECT MIN(salario) FROM empleado
```

- **AVG()**: calcula el promedio de los valores que contiene un atributo.

```
SELECT AVG(salario) FROM empleado
```

Predicados

- **All.** Se utiliza para especificar explícitamente que no se eliminen los duplicados(ALL puede reemplazarse por *).

```
SELECT * FROM Empleado
```

- **Top.** Muestra solo unos cuantos registros de la parte superior o inferior de un conjunto grande de registros.

```
SELECT TOP 3 * FROM Empleado
```

- **Distinct.** Elimina filas repetidas en una consulta en función de un atributo.

```
SELECT DISTINCT Nombre FROM Empleado
```

- **Distinctrow.** Elimina filas repetidas en una consulta en función de todos los atributos.

```
SELECT DISTINCTROW FROM Empleado
```

Bibliografía

Silberschatz, A.; Korth, H.; Sudarshan, S. *Fundamentos de Bases de Datos.* (2006). Madrid. España. McGraw Hill.

Elmasri, R.; Navathe, S.B. *Sistemas de Bases de Datos. Conceptos fundamentales.* (2007). Madrid. España. Addison-Wesley.

- MySQL 5.0 Reference Manual
- Libros en pantalla de Microsoft SQL Server 2005
- www.w3schools.com/sql/default.asp