Prof: Ing. Alberto Espinoza Zamora alberto.espinoza@ucem.ac.cr

Lista de comandos SQL para Sql Server y MySQL

A continuación, veremos los principales comandos SQL utilizados durante la administración de las Bases de Datos y conocerás para qué sirven:

CREATE DATABASE: es utilizado para crear una base de datos vacía.

Ejemplo (crear una tabla):

```
# CREATE TABLE Empleado
(
id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
Nombre VARCHAR(50),
Apellido VARCHAR(50),
Direccion VARCHAR(255),
Ciudad VARCHAR(60),
Telefono VARCHAR(15),
Peso VARCHAR (5),
Edad (2),
Actividad Específica (100),
idCargo INT
)
```

DROP DATABASE

Es utilizado para eliminar íntegramente una base de datos existente.

La instrucción DROP DATABASE te permite eliminar una o más bases de datos con la siguiente sintaxis:

```
DROP DATABASE  [ IF EXISTS ]
    database_name
    [,database_name2,...];
```

Prof: Ing. Alberto Espinoza Zamora alberto.espinoza@ucem.ac.cr

CREATE TABLE

Es utilizado para crear una tabla nueva en una base de Datos existente.

*La sintaxis SQL para **CREATE TABLE** es*

```
CREATE TABLE "nombre_tabla"
("columna 1" "tipo_de_datos_para_columna_1",
"columna 2" "tipo_de_datos_para_columna_2",
... );
```

ALTER TABLE

Es utilizado para modificar una tabla que hemos creado previamente.

Este comando permite modificar la estructura de un objeto. Se pueden agregar/quitar campos a una tabla, modificar el tipo de un campo, agregar/quitar índices a una tabla, modificar un trigger, etc.

Ejemplo (agregar columna a una tabla):

```
# ALTER TABLE 'NOMBRE_TABLA' ADD NUEVO_CAMPO INT;
# ALTER TABLE 'NOMBRE_TABLA' DROP COLUMN NOMBRE_COLUMNA;
```

DROP TABLE

Es utilizado para eliminar por completo una tabla de nuestra base de Datos.

Este comando elimina un objeto de la base de datos. Puede ser una tabla, vista, índice, trigger, función, procedimiento o cualquier otro objeto que el motor de la base de datos soporte. Se puede combinar con la sentencia ALTER.

Ejemplo:

```
# DROP TABLE 'NOMBRE_TABLA';
# DROP SCHEMA 'ESQUEMA;'
# DROP DATABASE 'BASEDATOS';
```

***Un trigger:** es un procedimiento almacenado en la base de datos que se invoca automáticamente cada vez que ocurre un evento especial en la base de datos.*

Prof: Ing. Alberto Espinoza Zamora alberto.espinoza@ucem.ac.cr

Comandos Sql para manipular Datos

SELECT

Es utilizado para recuperar datos de una o más tablas, en otras palabras, nos permite hacer consultas de los registros de las tablas.

La sintaxis básica de una consulta de selección es:

```
# SELECT Campos FROM Tabla;

# SELECT Nombre, Telefono FROM Clientes;
```

INSERT

Es utilizado añadir o insertar registros a una tabla creada previamente, su complemento es Insert into.

Forma básica:

```
# INSERT INTO ''tabla'' (''columna1'', [''columna2,... '']) V
ALUES (''valor1'', [''valor2,...''])
```

UPDATE

Es utilizado para modificar los datos de un conjunto de registros existentes en una tabla

Ejemplo:

```
# UPDATE mi_tabla SET campo1 = 'nuevo valor campo1' WHERE cam
po2 = 'N';
```

DELETE

Es utilizado para eliminar o borrar todo o una parte de los datos de la tabla indicada por el argumento especificado después de la palabra clave FROM.

Forma básica:

```
# DELETE FROM 'tabla' WHERE 'columna1' = 'valor1'
```

Prof: Ing. Alberto Espinoza Zamora alberto.espinoza@ucem.ac.cr

TRUNCATE

Es utilizado para borrar todos los registros de una tabla, pero no la tabla, es decir que quedaría una tabla vacía.

Ejemplo:

```
# TRUNCATE TABLE 'NOMBRE_TABLA';
```

Clausulas básicas

Las cláusulas son condiciones y las utilizamos para especificar los datos que deseamos seleccionar o manipular.

FROM

Se utiliza para especificar la tabla de la cual se van a consultar los registros

ALL

Si no se incluye ninguno de los predicados se asume ALL. El Motor de base de datos selecciona todos los registros que cumplen las condiciones de la instrucción SQL:

```
    # SELECT ALL FROM Empleados;
```

```
    # SELECT * FROM Empleados;
```

DISTINCT

Omite los registros que contienen datos duplicados en los campos seleccionados. Para que los valores de cada campo listado en la instrucción SELECT se incluyan en la consulta deben ser únicos:

```
    # SELECT DISTINCT Apellido FROM Empleados;
```

GROUP BY

Se utiliza para separar en grupos específicos los registros consultados.

Combina los registros con valores idénticos, en la lista de campos especificados, en un único registro:

```
# SELECT campos FROM tabla WHERE criterio
GROUP BY campos del grupo
```

Todos los campos de la lista de campos de SELECT deben o bien incluirse en la cláusula GROUP BY o como argumentos de una función SQL agregada:

```
# SELECT Id_Familia, Sum(Stock)
FROM Productos GROUP BY Id_Familia;
```

ORDER BY

Se utiliza para ordenar los registros seleccionados tomando en cuenta los parámetros que le indiquemos.

*Se puede especificar el orden en que se desean recuperar los registros de las tablas mediante la clausula **ORDER BY**:*

```
# SELECT CodigoPostal, Nombre, Telefono FROM Clientes ORDER BY Nombre
;
```

WHERE

Se utilizar para determinar los registros seleccionados con la cláusula FROM

La cláusula WHERE puede usarse para determinar qué registros de las tablas enumeradas en la cláusula FROM aparecerán en los resultados de la instrucción SELECT. WHERE es opcional, pero cuando aparece debe ir a continuación de FROM:

```
# SELECT Apellidos, Salario FROM Empleados
WHERE Salario > 21000;
# SELECT Id_Producto, Existencias FROM Productos
WHERE Existencias <= Nuevo_Pedido;
```

HAVING

Es parecida a **WHERE**, ya que determina qué registros se seleccionan. Cuando los registros se han agrupado utilizando GROUP BY, la cláusula HAVING determina cuáles de ellos se van a mostrar.

```
# SELECT Id_Familia Sum(Stock) FROM Productos GROUP BY Id_F
amilia HAVING Sum(Stock) > 100 AND NombreProducto Like BOS*
```

Prof: Ing. Alberto Espinoza Zamora alberto.espinoza@ucem.ac.cr

Comandos para la cláusula Select

AVG

Se utiliza para determinar el promedio de los registros de un campo determinado, en una tabla especifica.

Calcula la media aritmética de un conjunto de valores contenidos en un campo especificado de una consulta:

```
Avg(expr)
```

*La función Avg no incluye ningún campo Null en el cálculo. Un ejemplo del funcionamiento de **AVG**:*

```
# SELECT Avg(Gastos) AS Promedio FROM
Pedidos WHERE Gastos > 100;
```

COUNT

Se utiliza para devolver el número de registros que se muestran en una consulta realizada.

A continuación, se muestra la sintaxis de la función COUNT():

```
COUNT([ALL | DISTINCT] expression)
```

En esta sintaxis:

ALL indica a la función COUNT() que se aplique a todos los valores. ALL es el valor predeterminado.

DISTINCT indica a la función COUNT() que devuelva el número de valores únicos no nulos.

expresión es una expresión de cualquier tipo excepto imagen, texto o ntext. Ten en cuenta que no puedes usar una subconsulta o una función agregada en la expresión.

La función COUNT() también se usa de la siguiente manera:

```
COUNT(*)
```

Usando esta sintaxis, COUNT(*) devuelve el número de filas en una tabla específica. COUNT(*) no admite DISTINCT y no toma parámetros. Cuenta cada fila por separado e incluye filas que contienen valores NULL.

SUM

Se utiliza para devolver la suma de todos los registros de un campo especifico, dentro de una tabla.

Devuelve la suma del conjunto de valores contenido en un campo especifico de una consulta. Su sintaxis es:

```
Sum(expr)
```

Por ejemplo:

```
# SELECT Sum(PrecioUnidad * Cantidad)
AS Total FROM DetallePedido;
```

MAX

Se utiliza para devolver el registro o cantidad mayores de un campo especifico, dentro de una tabla.

MIN

Se utiliza para devolver el registro o cantidad menores de un campo especifico, dentro de una tabla.

Devuelven el mínimo o el máximo de un conjunto de valores contenidos en un campo especifico de una consulta. Su sintaxis es:

```
Min(expr)
Max(expr)
```

Un ejemplo de su uso:

```
# SELECT Min(Gastos) AS ElMin FROM Pedidos
WHERE Pais = 'Costa Rica';
# SELECT Max(Gastos) AS ElMax FROM Pedidos
WHERE Pais = 'Costa Rica';
```

Prof: Ing. Alberto Espinoza Zamora alberto.espinoza@ucem.ac.cr

Operadores Lógicos

AND

Es el “y” lógico. Evalúa dos condiciones y devuelve un valor de verdad sólo si ambas son ciertas.

OR

Es el “o” lógico. Evalúa dos condiciones y devuelve un valor de verdad si alguna de las dos es cierta.

NOT

Negación lógica. Devuelve el valor contrario de la expresión.

En donde expresión1 y expresión2 son las condiciones para evaluar, el resultado de la operación varía en función del operador lógico:

```
# SELECT * FROM Empleados WHERE Edad > 25 AND Edad < 50;

# SELECT * FROM Empleados WHERE (Edad > 25 AND Edad < 50) OR Sueldo = 100;

# SELECT * FROM Empleados WHERE NOT Estado = 'Soltero';

# SELECT * FROM Empleados WHERE (Sueldo > 100 AND Sueldo < 500) OR (Provincia = 'Madrid' AND Estado = 'Casado');
```

Operadores de comparación

<	Menor que
>	Mayor que
<>	Distinto de
<=	Menor o igual que
>=	Mayor o igual que

BETWEEN

El operador BETWEEN se utiliza en la cláusula WHERE para seleccionar valores entre un rango de datos.

Sintaxis de SQL BETWEEN

```
SELECT columna

FROM tabla WHERE columna

BETWEEN valor1 AND valor2
```

Prof: Ing. Alberto Espinoza Zamora alberto.espinoza@ucem.ac.cr

LIKE

Se utiliza para comparar una expresión de cadena con un modelo en una expresión SQL. Su sintaxis es:

```
expresión LIKE modelo
```

In

Este operador devuelve aquellos registros cuyo campo indicado coincide con alguno de los indicados en una lista. Su sintaxis es:

```
expresión [Not] In(valor1, valor2, . . .)


# SELECT * FROM Pedidos WHERE Provincia In ('Madrid', 'Barcel
ona', 'Sevilla');
```

Consultas mediante JOIN

JOIN

La sentencia SQL JOIN se utiliza para relacionar varias tablas. Nos permitirá obtener un listado de los campos que tienen coincidencias en ambas tablas:

```
# select nombre, telefono, accion, cantidad from clientes joi
n acciones on clientes.cid=acciones.cid;
```

LEFT JOIN

La sentencia LEFT JOIN nos dará el resultado anterior mas los campos de la tabla de la izquierda del JOIN que no tienen coincidencias en la tabla de la derecha:

```
# select nombre, telefono, accion, cantidad from clientes lef
t join acciones on clientes.cid=acciones.cid;
```

RIGHT JOIN

Idéntico funcionamiento que en el caso anterior pero con la tabla que se incluye en la consulta a la derecha del JOIN:

```
# select nombre, telefono, accion, cantidad from clientes rig
ht join acciones on clientes.cid=acciones.cid;
```