**SQL**

**¿Qué es? ¿Para qué sirve?**

SQL (Structured Query Language) es un lenguaje declarativo utilizado en programación; y diseñado para administrar sistemas de gestión de bases de datos relacionales. Una de sus principales características es el manejo del álgebra y el cálculo relacional para efectuar consultas con el fin de recuperar, de forma sencilla, información de bases de datos, así como realizar cambios en ellas.

**¿Qué es DML, DLL, DCL, TCL?**

DML: Es un lenguaje de Manipulación de Datos (Data Manipulation Language, DML) que es proporcionado Sistema de Gestión de Base de Datos, que permite a los usuarios visualizar y modificar los datos de las tablas y el lenguaje de manipulación de datos más popular hoy en día es SQL.

Algunos comandos son:

* SELECT: seleccionar registros de una tabla
* INSERT: insertar nuevos registros
* UPDATE: actualizar / Modificar registros existentes
* DELETE: eliminar registros existentes

DLL: Es un lenguaje de definición de datos (Data Definition Language) proporcionado por el Sistema de Gestión de Base de Datos y permite realizar consultas SQL para crear y modificar objetos.

Algunos comandos son:

* CREATE: crear una nueva tabla, base de datos, esquema
* ALTER: alterar la tabla existente y la descripción de la columna
* DROP: eliminar objetos existentes de la base de datos

DCL: Es un Lenguaje de Control de Datos (Data Control Language) proporcionado por el Sistema de Gestión de Base de Datos que incluye una serie de comandos SQL que permiten al administrador controlar el acceso a los datos contenidos en la Base de Datos.

Algunos comandos son:

* GRANT: Permite dar permisos a uno o varios usuarios o roles para realizar tareas determinadas.
* REVOKE: Permite eliminar permisos que previamente se han concedido con GRANT.

TCL: Es un lenguaje de control de transacciones (Transaction Control Language) que permiten controlar y administrar las transacciones para mantener la integridad de los datos dentro de las declaraciones SQL.

Algunos comandos son:

* BEGIN Transaction: abre una transacción
* COMPRAR Transaction: confirma una transacción
* ROLLBACK Transaction: retrocede una transacción en caso de error

**Motor de bases de datos y bases de datos**

**¿Qué son?**

El Motor de base de datos es el servicio principal para almacenar, procesar y proteger los datos, El motor es quien interpreta y ejecuta las consultas, mantiene los índices, proporciona acceso controlado y procesamiento de transacciones rápidas para cumplir con los requisitos de las aplicaciones consumidoras de datos más exigentes de su empresa.

Los más comunes son:

* MySQL
* PostgreSQL
* MSSQL
* SQLite
* MS Access

**¿Qué motores ofrece sqlzoo.net ?**

* MySQL
* ORACLE
* SQL SERVER

**¿Qué bases de datos ofrece sqlzoo?**

* World
* Nobel
* Game
* Goal
* Eteam
* Teacher
* Dept

**En este laboratorio, ¿en qué escribimos? ¿Por qué?**

Escribimos en SQL ya que en la actualidad es el lenguaje más utilizado en bases de datos y por eso mismo la manipulación de casi cualquier base de datos es más segura además SQL es un lenguaje declarativo estándar internacional de comunicación dentro de las bases de datos, donde se puede integrar a lenguajes de programación, por ejemplo, ASP o PHP, y en combinación con cualquier base de datos específica, por ejemplo, MySQL, SQL Server, MS Access, etc. Además, la gestión de Base de Datos es una habilidad cada vez más demandada y recompensada en el mundo de los negocios en nuestros días.

**B. Practica**

**Select**

La instrucción SELECT se utiliza para seleccionar datos de una base de datos.

Los datos devueltos se almacenan en una tabla de resultados, llamada conjunto de resultados.

* Calculo: {n:Customers |: n.CustomerName, n.City}
* {n:Customers|:n}
* Algebra: ,
* SQL: select City, PostalCode from Customers;

**Distinct**

La instruccion Distinct es usada para seleccionar únicamente valores diferentes

* Calculo: {n.Costumers|:n.Country }
* Algebra:
* SQL: Select Disctinct City from Customers;

**Where**

Es usada para filtrar los registros con una condicion

* Calculo:{n.Costumers |n.Country= ’Mexico’ :n}
* {n.Costumers |n.CostumerID= 1 :n}
* Algebra:
* SQL: Select CostumersID from Costumers Where mod(CostumersID,2)=0;

**AND, OR and NOT**

El operador AND muestra un registro si todas las condiciones separadas por AND son VERDADERAS.

El operador OR muestra un registro si alguna de las condiciones separadas por O es VERDADERO.

El operador NOT muestra un registro si la (s) condición (es) NO SON VERDADERAS.

* Calculo: {n.Costumers| Country='Germany' AND City='Berlin':n}
* {n.Costumers| City='Munich' OR City='Berlin':n}
* {n.Costumers| NOT Country='Germany':n}
* {n.Costumers| Country='Germany' AND (City='Berlin' OR City='München':n}
* {n.Costumers| NOT Country='Germany' AND NOT Country='USA':n}
* Algebra:
* select \* from customers where city=” London” and not PostalCode=”WA1 1DP”;

**ORDER BY**

El comando ORDER BY se usa para ordenar el conjunto de resultados en orden ascendente o descendente (asc,desc).

* SQL: select Country from Customers order by Country asc;
* Select Country from Customers order by Country asc, PostalCode desc;

**SELECT TOP**

el commando select top se usa para especificar el número de registros a devolver.

* Calculo:{n.customers|
* Algebra :
* SQL: select top 3 Country from Customers order by City asc;

**MIN () and MAX ()**

La función MIN () devuelve el valor más pequeño de la columna seleccionada.

La función MAX () devuelve el valor más grande de la columna seleccionada.

* Calculo: {{n.Products|:n.Price}}
* {{n.Products|:n.Price}}
* Algebra:
* SQL: select max(length(Country)) from Customers;

**COUNT (), AVG () and SUM ()**

La función COUNT () devuelve el número de filas que coinciden con un criterio especificado.

La función AVG () devuelve el valor promedio de una columna numérica.

La función SUM () devuelve la suma total de una columna numérica.

* Calculo: {{n.Products|:n.ProductID}}
* Algebra:
* SQL: select count (distinct City) from Customers;

**LIKE**

El operador LIKE se utiliza en una cláusula WHERE para buscar un patrón específico en una columna.

* SQL: select CustomerName from Costumers where CustomerName like “a%a”;

**Wildcards**

Representa un carácter comodín que se utiliza para sustituir cualquier otro carácter (s) en una cadena.

Los caracteres comodín se usan con el operador LIKE.

% - El signo de porcentaje representa cero, uno o múltiples caracteres

\_ - El guión bajo representa un solo carácter

[charlist] - Define conjuntos y rangos de caracteres que deben coincidir

[^ charlist] o [! charlist] - Define conjuntos y rangos de caracteres que NO coinciden

* SQL: Select City from Customers where City like “\_[eo]%”;

**BETWEEN**

El operador BETWEEN selecciona valores dentro de un rango determinado. Los valores pueden ser números, texto o fechas. Es un operador inclusivo, tanto el inicio como el final estan dentro del rango de alcance.

* Calculo: {n.products|
* {n.products|
* Algebra:
* SQL: Select \* from Orders where (OrderDate between #07/04/1996# and #07/09/1996#) and (OrderID between 10200 and 10300);

**GROUP BY**

La instrucción GROUP BY a menudo se usa con funciones agregadas (COUNT, MAX, MIN, SUM, AVG) para agrupar el conjunto de resultados por una o más columnas.

* SQL: select round (avg(price)), SupplierID from [products] group by SupplierID

**HAVING**

El commando HAVING permite hacer restriccion tal como lo hace el commando WHERE, pero esta admite funciones agregadas (COUNT, MAX, MIN, AVG).

* **SQL**

select price, ProductName, Supplierid from [products] group by ProductName having min(price)<10;

**Comments**

Los comentarios se utilizan para explicar secciones de codigo SQL o para evitar la ejecución de consultas SQL.

Se pueden hacer en una linea de la siguiente forma:

--Select all:

O se pueden aplicar a varias lineas de la siguiente forma: Selecciona todas las columnas empezando con /\* y terminando con \*/.

**B.)**

* Select CategoryID from Products where mod(CategoryID,2) =0;
* Select concat (‘the number of the category of’, ProductName, ‘is’, CategoryID) as desc from Products;
* Select SupplierName from Suppliers where city=’Londona’ or SupplierName like 'e%’;
* Select Cast (price as int) from Products where Select Cast (price as int) <124;
* Select Suppliername, Phone from [Suppliers] where supplierID between 10 and 16;
* Select current\_date (), ProductName,Price from Products;
* select Productname, Supplierid, Price, case when price < 30 then "justo lo que necesitamos" else "demasiado costoso" end from Products;
* select count(country), country from [Suppliers] group by country order by count(country) desc;

**Anidadas:**

Select SupplierName, ContactName, Phone from Suppliers where SupplierID in (select SupplierID from Products where price <40);

Select ProductName, SupplierID from (Select \* from Products where price<20 order by ProductName asc) where length(ProductName)=5;

Select \* from (Select count(SupplierName) as "#suppliers", country from suppliers group by country) where length(country)=3;