Apuntes SQL

El lenguaje SQL (Structure Query Languaje) es un lenguaje con el cual se escriben todas las acciones u operaciones que se realizan sobre los motores de base de datos relacionales. Es un lenguaje que se utiliza para las consultas programación de la base de datos. Se lo utiliza para acceder a los datos y para consultar, actualizar y gestionar sistemas de base de datos relacionales.
SQL se divide en dos partes: el DDL (Data Definition Lenguaje), Lenguaje de definición de datos; y el DML (Data Manipulation Lenguaje), que es lenguaje de manipulación de datos o tratamiento de datos.

Componentes del SQL

Le lenguaje SQL está compuesto por comandos, cláusulas, operadores y funciones de agregado. Estos elementos se combinan en las instrucciones para crear, actualizar y manipular las bases de datos.

Comandos

Existen dos tipos de comandos SQL:

Comandos DDL: permiten crear y definir nuevas base de datos, campos e índices.	
Comando	**Descripción**
CREATE	Utilizado para crear nuevas tablas, campos e índices.
DROP	Se lo emplea para eliminar tablas e índices.
ALTER	Utilizado para modificar las tablas agregando campos o cambiando la definición de los campos.

Comandos DML: permiten generar consultas para ordenar, filtrar y extraer datos de la base de datos.	
Comando	**Descripción**
SELECT	Utilizado para consultar registros de la base de datos que satisfagan un criterio determinado.
INSERT	Utilizado para cargar datos en la base de datos en una única operación
UPDATE	Utilizado para modificar los valores de los campos y registros especificados.
DELETE	Utilizado para eliminar registros de una tabla de una base de datos

Clausulas

Las clausulas son condiciones de modificación utilizadas para definir los datos que desea seleccionar o manipular.

Clausula	**Descripción**
FROM	Utilizada para especificar la tabla de la cual se van a seleccionar los registros.
WHERE	Utilizada para especificar las condiciones que deben reunir los registros que se van a seleccionar
GROUP BY	Utilizada para separar los registros seleccionados en grupos específicos
HAVING	Utilizada para expresar la condición que debe satisfacer cada grupo.
ORDER BY	Utilizada para ordenar los registros seleccionados de acuerdo con un orden especifico

SELECT

La sentencia SELECT nos permite consultar los datos almacenados en una tabla de la base de datos.
El formato de la sentencia en SQL es:

SELECT campo1, campo2,… FROM nombre_tabla

Donde campos es la lista de campos que se desea recuperar y tabla es el origen de los mismos, por ej:

SELECT *FROM socios (el asterisco significa que se van a recuperar todos los campos de la tabla)

Esta consulta devuelve una visualización de la tabla Socios con todos sus campos y registros.

Si se quiere recuperar solo algunos campos, por ejemplo: soccod y socnom, para ello vamos a crear una nueva consulta en donde escribiremos lo siguiente:

SELECT soccod, socnom FROM socios

INSERTAR UN NUEVO REGISTRO

La sintaxis utilizada para ingresar un nuevo registro es la siguiente:
Insert Into nombre_tabla (nombre_campo1, nombre_campo2,…) values (valor_campo1, valor_campo2,…)

Ejemplo: ingresar un nuevo socio
Insert Into socios (soccod, socape, socnom, fec_nac, socloc, codpostal) values (123, Ruiz, Elias, '1988-09-02', 'S.M de Tuc', 4000)
Observemos que los campos no numéricos van entre comillas simples.

ORDENAR REGISTROS

Podemos especificar un orden para recuperar los registros de las tablas mediante ORDER BY y la lista de campos a ordenar.
Por ejemplo: mostrar los socios ordenados por apellido

SELECT soccod, socape, socloc FROM socios ORDER BY socape

Esta consulta devuelve los campos soccod, socape y socloc de la tabla Socios ordenados por el campo socape.
También podemos ordenar los registros por más de un campo, como por ejemplo:

SELECT soccod, socape, socloc FROM socios ORDER BY socape, socloc

Incluso podemos especificar el orden de los registros para que sea de manera ascendente ò descendente (ASC-DES). Si no especificamos por defecto toma el valor ASC.
Y la sentencia SQL sería:

SELECT soccod, socape, socloc FROM socios ORDER BY socape DESC

CRITERIOS DE SELECCIÓN

WHERE

Indica a SQL que incluya únicamente ciertos registros en el resultado de la consulta. WHERE es necesario para recuperar datos de varias tablas.

Para las condiciones de filtro se puede utilizar cualquiera de los siguientes operadores:

Operador	Comparación
=	Igual
==	Exactamente igual
LIKE	SQL LIKE
<>	Distinto de
>	Mayor que

>=	Mayor o igual que
<	Menor
<=	menor o igual que

Por ejemplo:

1) si deseamos saber todos los socios cuyo código sea mayores a 5, la sentencia en SQL será:

SELECT soccod, socape, socnom FROM socios WHERE soccod > 5

SELECT soccod, socape, socnom: selecciona los campos soccod, socape, socnom de la tabla

FROM socios: establece el origen de los datos para la consulta.

WHERE soccod > 5 : es el criterio de fitro para la consulta

2) si necesitamos conocer el nombre y el precio de los artículos cuyo precio es hasta $1500, la consulta seria:

SELECT art_nom AS 'Artículos de precio hasta 1500', art_precio FROM artículos WHERE art_precio <= 1500

AS 'Artículos de precio hasta 1500': crea un título más significativo (alias) para el campo art_nom. Se debe tomar en cuenta que como es una cadena de caracteres, la misma debe ir entre comillas simples.

3) Listar todos los socios que vivan en San Miguel de Tucumán (código postal =4000)

SELECT socnom AS Socio, socdom AS Domicilio, soccodpos AS Codigo Postal FROM socios WHERE soccodpos=4000

La consulta me devolverá todos los socios con código postal igual a 4000.

4) Listar todos los socios que vivan en S.M de Tucumán y en Monteros.

SELECT socnom AS Socio, socdom AS Domicilio, soccodpos AS Codigo Postal FROM socios WHERE soccodpos=4000 OR soccodpos=4165

También se puede hacer la consulta de la siguiente manera con el operador IN:

SELECT socnom AS Socio, socdom AS Domicilio, soccodpos AS Codigo Postal FROM socios WHERE soccodpos IN (4000,4165)

La consulta me devolverá como resultado solo los socios con código postal igual a 4000 e igual a 4165

IN: utilizado para especificar registros de una base de datos.

BETWEEN

Utilizamos este operador para indicar que deseamos recuperar los registros según el intervalo de valores de un campo. Su sintaxis sería la siguiente:

Campo BETWEEN valor1 AND valor2

En este caso la consulta nos devolvería los registros que contengan en "campo" un valor incluido en el intervalo valor1, valor2 (ambos inclusive). Si anteponemos la condición NOT, devolverá aquellos valores NO incluidos en el intervalo.

Por ejemplo: necesitamos saber los datos de los socios que tienen DNI entre 12000000 y 26000000. La consulta será:

SELECT socnom AS Socio, socdom AS Domicilio, socdni AS 'Numero de Documento' FROM socios WHERE socdni BETWEEN 12000000 AND 26000000

MODIFICAR DATOS (UPDATE)

Esta instrucciòn nos sirve para modificar los registros de una tabla. Por medio de Where especificamos cuales son los registros en los que necesitamos realizar las modificaciones. Además deberemos especificar cuáles son los nuevos valores de los campos que vamos a actualizar.

La sintaxis es la siguiente:

Update nombre_tabla Set nombre_campo1= valor_campo1, nombre_campo2=valor campo2,… where condiciones de selección.

Ejemplo: Cambiar el apellido del socio cuyo código es 120.

Update socios Set socape='Ruiz' where soccod=120

Mediante esta sentencia hemos cambiado el apellido del socio por el de 'Ruiz' solo en el registro cuyo codigo de socio es 120.

***Importante:** hay que ser cuidadoso de no olvidarse de usar Where, de lo contrario modificaremos todos los registros de nuestra tabla.*

COUNT

Calcula el número de registros devueltos por una consulta.

Su sintaxis es la siguiente: **COUNT (expr)**

En donde expr contiene el nombre del campo que desea contar. Los operadores de expr pueden incluir el nombre de un campo de una tabla, una constante o una función. Aunque expr puede realizar un cálculo sobre un campo, COUNT simplemente cuenta el número de registros sin tener en cuenta que valores almacenan en los registros. La función COUNT no cuenta los registros que tienen campos NULL.

SELECT COUNT (campo) AS nombre_alias FROM tabla

Ejemplo: calcular cuántos autores hay por nacionalidad

SELECT autape, autnom, autnac, COUNT (*) AS 'cantidad de autores por nacionalidad' FROM autores

OPERADOR LIKE

Este operador se utiliza para comparar una expresión de cadena con un modelo en una expresión SQL.

Su sintaxis es: **expresión Like modelo**.

En donde expresión es una cadena modelo o campo contra el que compara expresión. Se puede utilizar el operador Like para encontrar valores en los campos que coincidan con el modelo especificado. Por modelo puede especificar un valor completo (Ana Maria), o se puede utilizar caracteres comodin para encontrar un rango de valores (LIKE 'AN%').

El operador Like se puede utilizar en una expresión para comparar un valor de un campo con una expresión cadena. Por ejemplo, si se introduce LIKE 'C%' en una consulta SQL, la consulta devuelve todos los valores de campo que comiencen con la letra C. En una con parámetros, puede hacer que el usuario escriba el modelo que se va a utilizar. Puede utilizar el signo (%) y subrayado (_) como parte de la expresión. El signo de procentaje representa a cualquier secuencia de caracteres desconocidos en la cadena. El subrayado representa un solo carácter desconocido en la cadena.

Ejemplo: Mostrar todos los socios cuyos nombres comiencen con G

Select socnom from socios where socnom LIKE 'G%'

BORRAR UN REGISTRO

Para eliminar un registro utilizamos la instrucción Delete. En este caso debemos especificar cual o cuales son los registros que queremos borrar, la podemos hacer a través de la cláusula where.

La sintaxis será:

Delete from nombre_tabla where condiciones_de_seleccion

Ejemplo: eliminar todos los registros de los socios que vivan en 'S.M. de Tucumán'

Delete from socios where socloc='S.M. de Tucumán'

También podemos considerar por código postal: **Delete from socios where codpostal=4000**

***Importante:** tener cuidado con esta instrucción ya que si no especificamos una condición con Where, lo que estaríamos haciendo es borrar toda la tabla.

Consultas combinadas: JOINS

Cuando necesitamos recuperar información de una base de datos nos encontramos con que dicha información se encuentra repartida en varias tablas, referenciadas a través de varios códigos. De este modo si tuviéramos una tabla de ventas con un campo cliente, dicho campo contendría el código del cliente de la tabla cliente.

Sin embargo esta forma de almacenar la información no resulta muy útil a la hora de consultar los datos. SQL nos proporciona una forma fácil de mostrar la información repartida en varias tablas, las consultas combinadas o JOINS. Las consultas combinadas pueden ser de tres tipos:

- Combinación interna
- Combinación externa
- Uniones

Consultas de unión Internas

Las vinculaciones entre tablas se realizan mediante la cláusula INNER que combina registros de dos tablas siempre que haya concordancia de valores en un campo común.
Su sintaxis es:

SELECT campos FROM tabla1 INNER JOIN tabla2 ON tabla1.campo1 OPERADOR tabla2.campo2

En donde:
Tabla1, tabla2 son los nombres de las tablas desde las que se combinan los registros.
Campo1, campo2 son los nombres de los campos que se combinan. Si son numéricos, los campos deben ser del mismo tipo de datos y contener el mismo tipo de datos, pero no tienen que tener el mismo nombre.
Operador: es cualquier operador de comparación relacional: =, <,>,>=, <, <=, ò <>.
Se puede utilizar una operación INNER JOIN en cualquier clausula FROM. Esto crea una combinación por equivalencia, conocida también como unión interna. Las combinaciones de equivalencia son las más comunes; estas combinan los registros de dos tablas siempre que haya concordancia de valores en un campo común a ambas tablas. Se puede utilizar INNER JOIN con las tablas autores y libros para seleccionar todos los libros de un autor ò los libros de un autor en particular.
Más formalmente INNER JOIN específica que el resultado de la consulta contenga solo filas para una tabla con la que coincidan una o varias filas en otra tabla.

Por ejemplo:
1) visualizar la cantidad de ejemplares que posee un libro

SELECT libros.libcod, libtit, COUNt(*) AS 'cantidad de ejemplares' FROM libros
INNER JOIN ejemplares ON libros.libcod=ejemplares.libcod

2) Visualizar el codigo de un cliente, nombre, código postal y la descripción de la localidad

SELECT clicod, clinom, localidades.loccodpos, localidades.locdes FROM clientes
INNER JOIN localidades ON clientes.loccodpos=localidades.loccodpos