## RESUMEN SQL

# Resumen SQL

## Introducción

SQL es un lenguaje de consultas de BD, que está compuesto por dos submódulos fundamentales:

* Módulo para definición del modelo de datos, denominado DDL con el cual se pueden definir las tablas, atributos, integridad referencial, índices y restricciones del modelo.
* Módulo para la operatoria normal de la BD, denominado DML, con el cual se pueden realizar las consultas, altas, bajas y modificaciones de datos sobre las tablas

## Lenguaje de definición de datos

El modelo de datos relacional utiliza el concepto de relación, tupla y atributo, en tanto en SQL los términos corresponden a tabla, fila y columna. Para administrar un modelo físico de datos, SQL presenta tres cláusulas básicas:

* CREATE (TABLE|DATABASE)
* DROP (TABLE|DATABASE)
* ALTER TABLE

## Lenguaje de manipulación de datos

La estructura básica de una consulta es:  
**SELECT** lista\_atributos **FROM** lista\_tablas **WHERE** predicado

Lista\_tablas indica las tablas de la BD necesarias para resolver la consulta; sobre las tablas indicadas en la lista se realiza un producto cartesiano. De éstas cláusulas sólo **SELECT** y **FROM** son obligatorias. Cuándo en la cláusula **FROM** hay más de una tabla es necesario agregar una cláusula **WHERE** para quedarse sólo con las filas con sentido, utilizando las FK y así obteniendo como resultado un producto natural.

### Otras cláusulas u operadores

* **DISTINCT**: Elimina las tuplas repetidas en el resultado otorgado por un select. Ej: **SELECT DISTINCT** atributo\_id
* **ALL**: Muestra todas las tuplas, aún las repetidas, cuándo no se especfíca el operador se asume ALL. Ej: **SELECT ALL**
* **UNION**: Reúne los resultados de dos consultas, las tuplas duplicadas quedan descartadas, para conservarlas se puede usar la cláusula **UNION ALL**
* **EXCEPT**: Análoga a la diferencia de conjuntos.
* **INTERSECT**: Análoga a la intersección de conjuntos.
* **ORDER BY**: Ordena el conjunto resultado de tuplas de menor a mayor, es posible ordenar al conjunto por más de un atributo, el segundo criterio se aplica en caso de empato en el primero. Si se desea hacerlo de mayor a menor hay que utilizar el operador DESC. EJ: **ORDER BY** atributo1, atributo2 **DESC**

### Funciones de agregación

Las funciones de agregación son funciones que operan sobre un conjunto de tuplas de entrada y producen un único valor de salida. SQL define cinco:

* **AVG**: Retorna como resultado el promedio del atributo indicado para todas las tuplas del conjunto.
* **COUNT**: Retorna como resultado la cantidad de tuplas involucradas en el conjunto de entrada.
* **MAX**: Retorna como resultado el valor más grande dentro del conjunto de tuplas para el atributo indicado.
* **MIN**: Retorna como resultado el valor más pequeño dentro del conjunto de tuplas para el atributo indicado.
* **SUM**: Retorna como resultado la suma del valor del atributo indicado para todas las tuplas del conjunto.

Las funciones de agregación se aplican sobre un conjunto de tuplas, por éste motivo **NO** pueden estar definidas dentro de una cláusula where. Una función de agregación siempre retorna un resultado. No siempre es posible retornar además de dicho, el valor de otro atributo, dado que no se puede determinar a que tupla, del conjunto sobre el que trabajó la función de agregación, pertenece el atributo.

### Funciones de agrupación

En muchos casos resulta necesario agrupar las tuplas de una consulta por algún criterio, para éstas situaciones se puede utilizar la cláusula **GROUP BY**. Además se puede aplicar una función de agregación sobre cada grupo. Cuándo se genera un agrupamiento por un criterio (atributo) es posible proyectar en el **SELECT** el atributo por el cual se genera el grupo. También se pueden filtrar a los grupos que cumplan determinado predicado mediante la cláusula **HAVING**. Dentro de un **HAVING** se puede utilizar tambièn una función de agregación.

### Subconsultas

Una consulta puede contener otras consultas dentro de la misma. Las subconsultas se pueden utilizar en las siguientes situaciones:

1. Cuando se desea comprobar la pertenencia o no en un conjunto
2. Cuando se desea analizar la cardinalidad de un conjunto
3. Cuando se desea analizar la existencia o no de elementos en un conjunto.

Cuándo una consulta retorna un solo resultado es posible compararlo contra un valor simple. Ej

**SELECT** nombre  
**FROM** carrera  
**WHERE** duracion\_años = (  
**SELECT** MAX(duracion\_años)  
**FROM** carreras  
)

SQL define el operador IN para comprobar si un elemento es parte o no de un conjunto.

**SELECT** nombre  
**FROM** alumnos  
**WHERE** id\_carrera **IN** (  
**SELECT** id\_carrera  
**FROM** carreras  
**WHERE** duracion\_años = 5  
)

La operación contraria puede ser realizada utilizando el operador **NOT IN**. Además, se pueden utilizar operadores de comparación de elementos simples sobre conjuntos. Estos operadores similares a los racionales (>, <, =, >=, <=, <>) estás combinados con las palabras reservadas **SOME** y **ALL**.

La cláusula **EXISTS** se utiliza para comprobar si una subconsulta generó o no alguna tupla como respuesta. El resultado de la cláusula es verdadero si la subconsulta tiene al menos una tupla. No es importante para la subconsulta el o los atributos proyectados, debido a que sólo se chequea existencia o no de las tuplas. La operación contraria es **NOT EXISTS**

### Operaciones con Valores Nulos

Existe un operador **IS NULL** y su negación **IS NOT NULL**

**SELECT** nombre **FROM** alumnos **WHERE** id\_localidad **IS NULL**

### Productos Naturales

Las sentencias disponibles al efecto son:

* **INNER JOIN**: Realiza un producto natural clásico, reuniendo las tuplas de las relaciones que tienen sentido,
* **LEFT JOIN**: Similar al INNER JOIN sólo que si alguna tupla de la tabla 1 no se reúne con una de la tabla 2 igualmente aparece en el resultado
* **RIGHT JOIN**: Similar al INNER JOIN sólo que si alguna tupla de la tabla 2 no se reúne con una de la tabla 1 igualmente aparece en el resultado
* **FULL JOIN**: Reúne las las tuplas con sentido, luego agrega las tuplas de la tabla 1 sin correspondencia en la tabla 2 y, por último, reúne las tuplas de tabla 2 sin correspondencia en tabla 1. Es una fusión de INNER JOIN, LEFT JOIN, RIGTH JOIN.

### Otras operaciones

* **Inserción**: se utiliza la cláusula **INSERT INTO** tabla (list\_atributos) **VALUES** (lista\_valores)

Si el usuario intentara asignar un valor a un atributo definido como autoincremental, la operación de inserción falla. Se debe tener cuidado además con los atributos que representan claves foráneas, en caso de haber exigido integridad referencial estricta estas claves deben ser válidas o la consulta fallará. La cláusul genera una tupla cada vez que es ejecutada.

* **Borrado**: **DELETE FROM** tabla **WHERE** condición

Debe tenerse en cuenta también a la integridad referencial. Solamente pueden borrarse tuplas que cumplan con las condiciones de IR definidas.

* **Modificación** **UPDATE** tabla **SET** atributo = valor **WHERE** condición

Si varias tuplas cumplen con la condición del WHERE todas ellas serán modificadas.

### Vistas

Una vista de una BD es un resultado de una consulta SQL que puede involucrar una o varias tablas. Las vistas tienen la misma estructura que una tabla: tuplas y atributos y son el resultado de guardar en forma temporaría el resultado de una consulta SQL. La única diferencia es que sólo se almacena de ellas la definición de la consulta, no los datos. Cada vez que la vista es utilizada, la consulta es recalculada para obtener una imagen actual de la BD.

Las vistas se crean por diversos motivos. Uno de ellos es lograr modularizar las consultas. Otro motivo es la seguridad de la BD, el DBA puede decidir que para determinados usuarios de la BD no es conveniente que operen sobre todos los atributos de todas las tablas. Se generan así vistas, que solamente tienen definidos aquellos atributos que el usuario puede manipular. Se les asigna derechos de trabajo sobre esas vistas y por ende para esos usuarios sólo existen los atributos que son visibles. Para crear una vista SQL prevee la cláusula **CREATE VIEW**