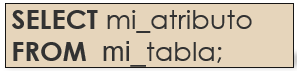
**TEMA 4**

**CONSULTAS SQL**

Se escriben con mayúsculas las palabras reservadas y con minúsculas el resto

Ejemplo:



**SELECT ->** identifica las columnas a recuperar – **EL QUE**

**FROM ->** identifica la tabla – **DE DONDE** **obtener los datos**

El resultado de una consulta es una tabla

**SELECCIÓN DE TODAS LAS COLUMNAS:**

El **\*** se usa para especificar la recuperación de todos los datos de las tablas

Ejemplo:

**SELECCIÓN SOLO DE ALGUNAS COLUMNAS:**

Ejemplo: seleccionar el código y nombre de las instituciones



**FILAS DUPLICADAS**

Cuando se recuperan todas las filas, estas incluyen a las duplicadas. Para eliminar los valores repetidos se debe usar la cláusula **DISTINCT**

**Ejemplo: seleccionar los distintos voluntarios que son coordinadores**



**ELIMINAR VALORES REPETIDOS**

**DISTINCT** se aplica a todas las columnas de la lista en el **SELECT**

Ejemplo: seleccionar los voluntarios coordinadores y las distintas instituciones de los empleados coordinados



**USO DE OPERADORES ARITMETICOS**

Una expresión aritmética puede contener nombres de columnas, valores numéricos y operadores aritméticos



* La multiplicación y la división tienen prioridad sobre la suma y la resta
* Los operadores de idéntica prioridad se evalúan de izquierda a derecha
* Los paréntesis se utilizan para forzar evaluaciones prioritarias y para clarificar las sentencias

Ej: seleccionar los nombres de las tareas y los rangos de duración (diferencia entre horas máximas y mínimas) de cada una



**RENOMBRADO DE COLUMNAS**

* El alias **AS** renombra un encabezamiento de columna o tabla.
* Si contiene espacios, caracteres especiales o es sensible a mayúsculas y minúsculas, se debe encerrar entre comillas dobles.

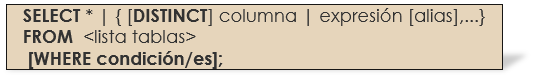


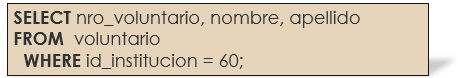
* Operador de concatenación || para cadenas de caracteres



**RESTRINGIR LAS FILAS RECUPERADAS**

* La cláusula **WHERE** se usa para realizar las restricciones.
* Una cláusula **WHERE** contiene una condición lógica, la cual usa
  + operadores de comparación **(<, >, =, <=, >=, <>** o **!=)**
  + operadores lógicos (**AND, OR, NOT**).
* Las filas recuperadas son aquellas cuyos datos que satisfacen la/s condición/es lógicas



**Ejemplo: recuperar el nro de voluntario, nombre y apellido de los voluntarios que trabajan en la institución cuyo identificador es 60.**

**CONDICIONES DE COMPARACION**

* Los operadores de comparación se utilizan en la cláusula **WHERE** para comparar expresiones.
* El resultado de la comparación puede ser
  + - Verdadero (T)
    - Falso (F)
    - Desconocido (U)
* Si se comparan valores nulos usando los operadores de comparación el resultado será siempre FALSO porque un valor nulo no puede ser igual, mayor, distinto, etc. a otro valor.



* Además de los operadores de comparación está disponible un operador especial [**NOT**] **BETWEEN**:
* **BETWEEN**trata a los valores de los extremos incluidos dentro del rango
  + - **BETWEEN** *x* **AND** *y* es equivalente a a >= x **AND**a <= y
    - **NOT BETWEEN** x**AND**y es equivalente a a < x **OR** a > y

**Ejemplo: Seleccionar los voluntarios cuyo número se encuentra entre 100 y 120**

**OPERADOR LIKE**

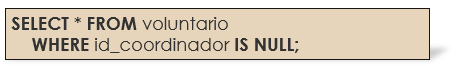
* No siempre se conoce el valor exacto a buscar.
* Se puede buscar coincidencias con un patrón de caracteres mediante el operador **LIKE**. También se emplea en la forma negativa **NOT LIKE**.
* Comodines
  + - % : cualquier secuencia de cero o más caracteres
    - \_ : denota un solo carácter

**Ejemplo: Seleccionar los voluntarios cuya segunda letra del nombre sea a y luego tenga una n como carácter final.**



**Operador IS [NOT] NULL**

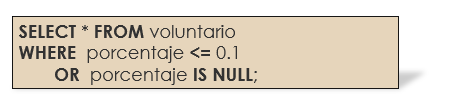
* Si una columna en particular carece de un valor se dice que contiene un **NULL**.
* **NULL** es un valor inaccesible, sin valor, desconocido o inaplicable.
* No representa ni un cero ni un espacio en blanco (el cero es un número y el espacio en blanco es un carácter). SOLO testean valores que son nulos.

**Ejemplo: listar los datos completos de los voluntarios que no tengan coordinador.**

* Si se comparan valores nulos usando los otros operadores (=, >, etc.) el resultado será siempre FALSO porque un valor nulo no puede ser igual, mayor, menor o distinto a otro valor.
* Si se desea incluir en el resultado los datos de aquellas columnas que tengan nulos hay que hacerlo explícitamente.

**Ejemplo listar los datos de los voluntarios cuyo porcentaje sea menor o igual que 0,10.**

* + - Algunos porcentajes pueden ser nulos
    - Si deseo incluirlos en el resultado debo explicitar IS NULL.



**CONDICIONES DE COMPARACION COMPUESTAS**

* Un operador lógico combina los resultados de dos condiciones para producir un único resultado basado en ellos, o invertir el resultado de una condición.
* Los operadores **AND** y **OR** se pueden usar para componer expresiones lógicas.
* El operador **AND** retorna VERDADERO si ambas condiciones evaluadas son VERDADERAS, mientras que el operador **OR** retorna VERDADERO si alguna de las condiciones es VERDADERA.
* El operador **NOT** invierte el resultado de la expresión.

**EJEMPLO: Seleccionar los voluntarios que son coordinados por los voluntarios nro 100 o 124 y están trabajando para la institución cuyo código es 50.**

**SELECT** nro\_voluntario, apellido, id\_institucion, id\_coordinador

**FROM** voluntario

**WHERE** (id\_coordinador=100

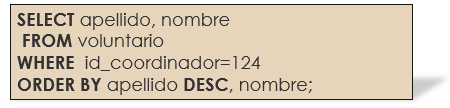
**OR** id\_coordinador=124)

**AND** id\_institucion=50;

**Orden de presentación de los registros**

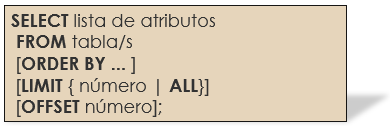
* ORDER BY puede usarse para ordenar las filas, y se debe colocar como última cláusula de la sentencia SELECT.
* Si no se especifica, el orden de los datos por defecto cuando se declara el ORDER BY es ascendente (ASC), pero se puede especificar también DESC. Se debe colocar después del nombre de la columna.
* Se puede ordenar el resultado de una consulta por más de una columna (pueden ser todas las de la tabla)

**Ejemplo: listar los apellidos ordenados descendentemente y nombres de los voluntarios que son coordinados por el voluntario 124**



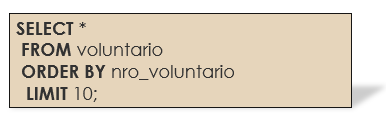
**LIMIT y OFFSET(POSTGRESQL)**

* Permiten recuperar solamente un subconjunto de filas del total de la consulta.



* Debería ser usado siempre con la cláusula **ORDER BY.**
* La cláusula **LIMIT** limita la cantidad de filas a retornar.
* La cláusula **OFFSET** determina a partir de qué fila del resultado se retorna.

**Ejemplo LIMIT: seleccionar los datos de los voluntarios que corresponden a los 10 primeros voluntarios.**



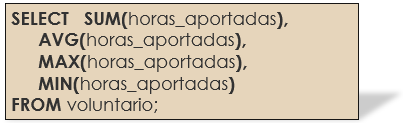
**Ejemplo OFFSET: seleccionar los datos de los voluntarios a partir del 15TO voluntario.**

**FUNCIONES DE GRUPO**

Estas funciones operan sobre conjuntos de filas para proporcionar un resultado por grupo

Permiten resumir el resultado de una consulta:

* + **SUM () →** *sumatoria de la columna especificada*
  + **AVG () →** *promedio de la columna especificada*
  + **STDDEV () *→*** *desvío estándar de la columna especificada*
  + **MAX () *→*** *valor máximo de la columna especificada*
  + **MIN () *→*** *valor mínimo de la columna especificada*
  + **COUNT () *→*** *cantidad de tuplas*
* **AVG,** **SUM** y ***STDDEV*** se usan para datos numéricos.
* **MIN** y **MAX** se pueden usar para cualquier tipo de dato



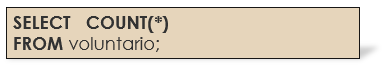
**Ejemplo: seleccionar el voluntario más joven y el más viejo.**

**SELECT MAX(**fecha\_nacimiento**) AS** voluntario\_mas\_joven**,**

**MIN(**fecha\_nacimiento**) AS** voluntario\_mas\_viejo

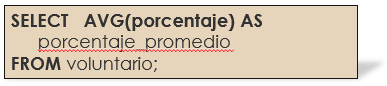
**FROM** voluntario;

* **COUNT (\*)** devuelve el número de filas de una tabla.
* **COUNT (**expr**)** devuelve el número de filas con valores no nulos para expr.

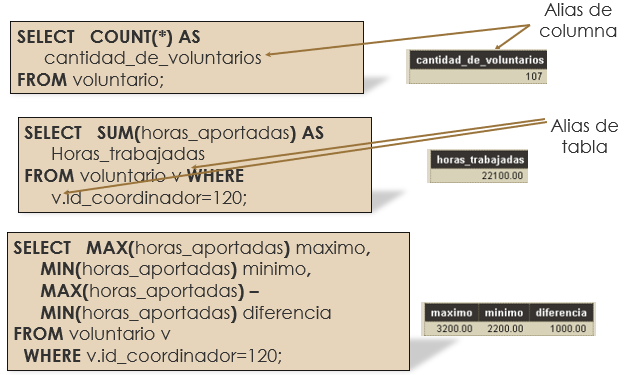


**Ejemplo: Liste el número de ciudades en la tabla dirección, excluyendo los valores nulos.**

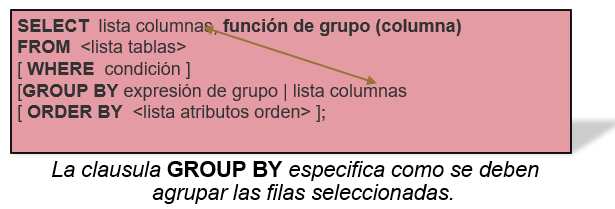
Las funciones de grupo ignoran los valores nulos del atributo



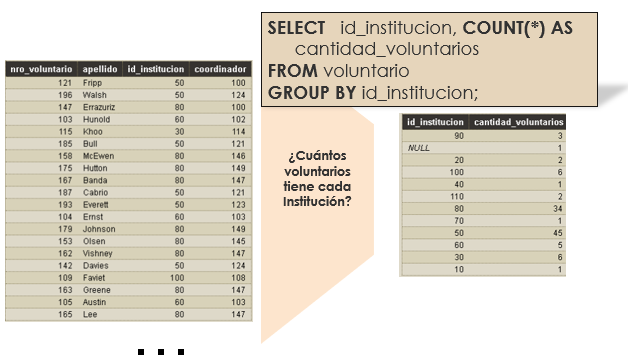
La función **COALESCE** (columna, valor\_reemplazo) en POSTGRESQL fuerzan a las funciones de grupo a que incluyan valores nulos, retornando un valor en ocurrencia de un nulo.

**Ejemplos**

**SELECCIÓN SOBRE GRUPOS DE DATOS**

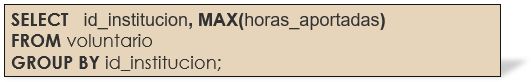
* Si se usa la cláusula **GROUP BY** en una sentencia **SELECT**, se dividen las filas de la tabla consultada en grupos
* Se aplica las funciones en la lista **SELECT** a cada grupo de filas y retorna una única fila por cada grupo.

**CREACION DE GRUPOS DE DATOS**



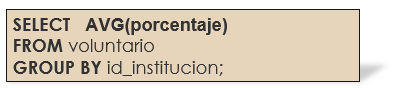
**SINTAXIS DE LA CLAUSULA GROUP**

***Todas las columnas de la lista SELECT, excepto las funciones de grupo, deben estar en la cláusula GROUP BY***

Ejemplo: liste las diferentes instituciones y el máximo de horas aportadas a cada una de ellas

Las columnas en la cláusula **GROUP BY** pueden no estar en la lista del **SELECT**

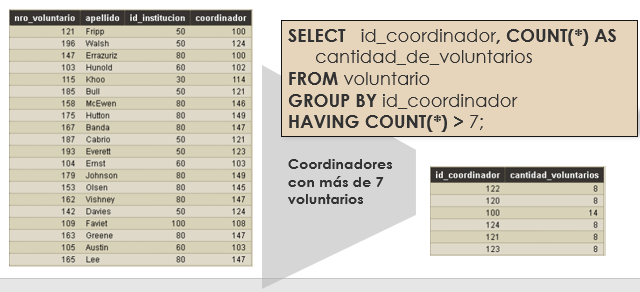
**Ejemplo: determine los porcentajes promedio de los voluntarios por institución.**



**RESTRINGIR LOS RESULTADOS DE LOS GRUPOS**

Se puede anexar la cláusula **HAVING** para restringir grupos

* 1. Las filas se agrupan por la/s columnas especificada/s
  2. Se aplica la función de grupo
  3. Se muestran los grupos que satisfacen la cláusula **HAVING**

****

**FUNCIONES DE GRUPOS NO VALIDAS**

* No se puede utilizar la cláusula **WHERE** para restringir grupos
* Se debe utilizar la cláusula **HAVING** para restringir grupos. No se pueden utilizar funciones de grupo en la cláusula **WHERE**

**SELECT** id\_coordinador**, COUNT(\*) AS** cantidad\_de\_voluntarios

**FROM** voluntario

**WHERE COUNT(\*)** >7

**GROUP BY** id\_coordinador;

