Bases de Datos I

Facultad de Ciencias Exactas
UNIVERSIDAD NACIONAL DEL CENTRO DE LA PROVINCIA DE BUENOS AIRES

Consultas de Datos - Parte 1

INGENIERÍA DE SISTEMAS

Cursada 2019

Lenguaje SQL

Algunas funciones del estandar SQL:

DDL
- ✓ Definición de datos
  - Creación de objetos (CREATE)
  - Modificación de objetos definidos (ALTER)
  - Eliminación de objetos definidos (DROP)

Los objetos pueden ser:
- *tablas,*
- *vistas,*
- *índices,*
- *dominios,*
- *etc.*

DML
- ✓ Actualización de los datos
  - Inserción (INSERT)
  - Actualización (UPDATE)
  - Eliminación (DELETE)
- ✓ Consulta de datos
  - Selección (SELECT)

Lenguaje de Consulta SQL

- ✓ Para la recuperación de los datos a partir de tablas cargadas en la base de datos se utiliza la sentencia **SELECT**.
- ✓ El formato más simple y básico es:

```
SELECT * | { [DISTINCT] columna | expresion [alias],...}
FROM  <lista tablas>
```

- ✓ En una consulta se especifica qué información se requiere, sin especificar cómo obtenerla, no le especificamos métodos de acceso a los datos.
- ✓ **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*** (si es un número, se considera como una tabla con una fila y una columna).

Escritura de Sentencias SQL

- **NO** son sensibles a mayúsculas/minúsculas.
- Pueden ocupar una o más líneas.
- Las palabras clave **NO** se pueden abreviar ni dividir entre líneas.
- Las cláusulas suelen colocarse en líneas separadas y con sangría, para mejorar la legibilidad.
- Se estila escribir con MAYUSCULAS las palabras reservadas y con minúsculas el resto.
- Por ejemplo:

```
SELECT mi_atributo
FROM  mi_tabla;
```

Como funcionan todos los DBMS ?

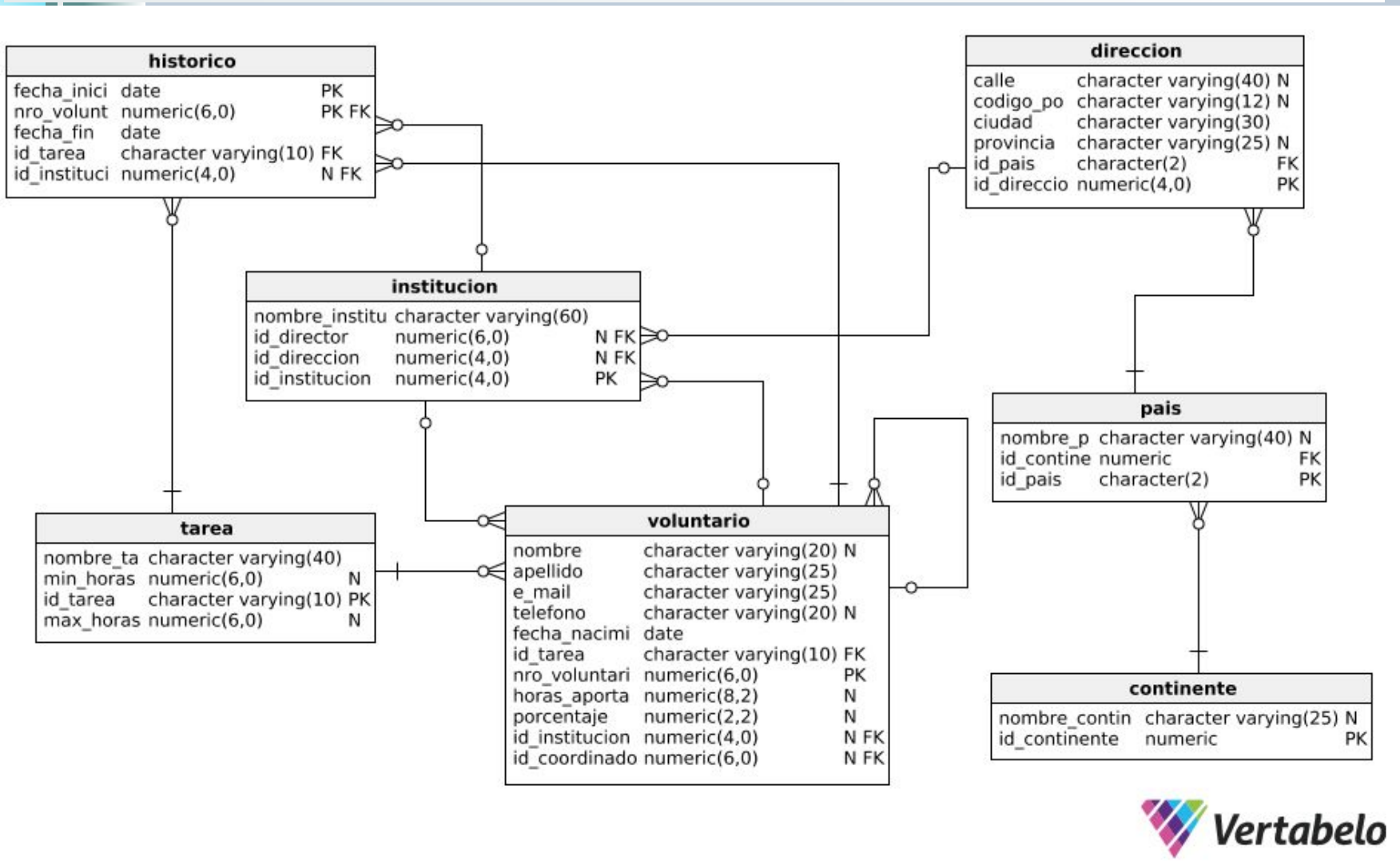
Todos, absolutamente todos los DBMS para resolver una consulta siguen el siguiente esquema.

1) Origen de Datos, el DBMS recupera todas las filas y columnas con las que va a trabajar (FROM)

2) Filtrado del Origen de Datos (WHERE)

3) Proyección (SELECT)

Esquema Ejemplo DERE BD Voluntarios



Vertabelo

Consultas SQL básicas

- ✓ Selección de todas las columnas. Ejemplo: selección de los datos completos de las instituciones.

```
SELECT *
FROM institucion;
```

nombre_institucion	id_director	id_direccion	id_institucion
CASA DE LA PROVIDENCIA	200	1700	10
CORPORACION URRACAS DE EMAUS	201	1800	20
FUNDACION CIVITAS	114	1700	30
FUNDACION LAS ROSAS DE AYUDA FRATERNA	203	2400	40
FUNDACION HOGAR DE CRISTO	121	1500	50
FUNDACION MI CASA	103	1400	60
CORPORACION SOLIDARIDAD Y DESARROLLO	204	2700	70
FUNDACION REGAZO	145	2500	80
FUNDACION ALERTA BOSQUES	100	1700	90
BOSQUEDUCA	108	1700	100
COMITE NACIONAL PRO DEFENSA DE LA FLORA Y LA FAUNA	205	1700	110
CONSEJO ECOLOGICO COMUNAL	*NULL*	1700	120
CORPORACION AMBIENTAL	*NULL*	1700	130
FUNDACION VIDA RURAL	*NULL*	1700	140
CENTRO DE AYUDA MAPUCHE	*NULL*	1700	150
SIERRAS PROTEGIDAS	*NULL*	1700	160
CENTRO DE EDUCACION AMBIENTAL	*NULL*	1700	170
RENACE- RED DE ACCION ECOLOGICA	*NULL*	1700	180
Contracting	*NULL*	1700	190
CONSEJO NACIONAL DE LA JUVENTUD	*NULL*	1700	200
DEFENSA DE LOS DERECHOC DEL NIÑO	*NULL*	1700	210
FUNDACION CHILDREN	*NULL*	1700	220
CORPORACION ANGLICANA	*NULL*	1700	230
CORPORACION EVANGELICA	*NULL*	1700	240
CENTRO ECUMENICO	*NULL*	1700	250

Se usa para especificar la recuperación de todos los datos de la/s tabla/s

Porque se recomienda **evitar** su uso? Y en su lugar seleccionar **SOLO** los campos necesarios…

Ademas:27 fila(s)
Tiempo total de ejecución: 2.589 ms
SQL ejecutada.

Consultas SQL básicas

- Selección SÓLO de algunas columnas. Ejemplo: *seleccionar el código y el nombre de las instituciones.*

```
SELECT id_institucion, nombre_institucion
FROM   institucion;
```

id_institucion	nombre_institucion
10	CASA DE LA PROVIDENCIA
20	CORPORACION URRACAS DE EMAUS
30	FUNDACION CIVITAS
40	FUNDACION LAS ROSAS DE AYUDA FRATERNA
50	FUNDACION HOGAR DE CRISTO
60	FUNDACION MI CASA
70	CORPORACION SOLIDARIDAD Y DESARROLLO
80	FUNDACION REGAZO
90	FUNDACION ALERTA BOSQUES
100	BOSQUEDUCA
110	COMITE NACIONAL PRO DEFENSA DE LA FLORA Y LA FAUNA
120	CONSEJO ECOLOGICO COMUNAL
130	CORPORACION AMBIENTAL
140	FUNDACION VIDA RURAL
150	CENTRO DE AYUDA MAPUCHE
160	SIERRAS PROTEGIDAS
170	CENTRO DE EDUCACION AMBIENTAL
180	RENACE- RED DE ACCION ECOLOGICA
190	Contracting

Filas Duplicadas

- Por defecto, ante una consulta, se recuperan todas las filas, **incluidas** las filas duplicadas. Ejemplo: *Seleccionar los voluntarios que son coordinadores.*

```
SELECT id_coordinador
FROM  voluntario;
```

id_coordinador
NULL
100
100
102
103
103
103
103
101
108
108
108
108
108
100
114
114
114
114
114
100
100
100
100
100
120
120
120
120

...

- Para eliminar los valores repetidos se debe usar la cláusula **DISTINCT**. Ejemplo: *Seleccionar los distintos voluntarios que son coordinadores.*

```
SELECT DISTINCT id_coordinador
FROM  voluntario;
```

id_coordinador
NULL
205
122
120
101
103
108
145
100
201
124
114
121
123
102
146
147
148
149

Eliminar los valores repetidos

- La cláusula **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.*

```
SELECT DISTINCT id_institucion,
                id_coordinador
FROM  voluntario;
```

id_institucion	id_coordinador
90	*NULL*
110	101
100	101
40	101
80	100
110	205
100	108
30	100
50	100
80	147
80	148
60	102
30	114
50	124
80	149
50	121
50	123
80	146
60	103
20	100
20	201
90	100
70	101
80	145
10	101
50	122
NULL	149
50	120

Ejercicios

1. Cuantos registros tiene la tabla tareas?
2. Cual es el resultados si se listan los códigos de las tareas que están realizando efectivamente los voluntarios? Cuantos registros retorna la consulta?
3. Y si se listan los distintos códigos de las tareas que están realizando efectivamente los voluntarios?

http://dbases.exa.unicen.edu.ar/phppgadmin

Filtrar filas recuperadas

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

```
SELECT * | { [DISTINCT] columna | expresion [alias],...}
FROM  <lista tablas>
 [WHERE condicion/es];
```

Filtrar filas recuperadas

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

```
SELECT nro_voluntario, nombre, apellido
FROM  voluntario
WHERE id_institucion= 60;
```

Resultado de la consulta

nro_voluntario	nombre	apellido
103	Alexander	Hunold
104	Bruce	Ernst
105	David	Austin
106	Valli	Pataballa
107	Diana	Lorentz

Condiciones de comparación

- Al utilizar los operadores de comparación el resultado puede ser:
  - Verdadero (T)
  - Falso (F)
  - Desconocido (U)
- Tener presente que 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.

```
SELECT nombre, apellido,
       e-mail
FROM  voluntario
WHERE id_tarea= 'ST_MAN';
```

```
SELECT nombre, apellido,
       e-mail
FROM  voluntario
WHERE id_tarea != 'ST_MAN';
```

Ejercicios

Muestre los apellidos, nombres, y e_mail de los voluntarios que llevan aportadas más de 1.000 horas, ordenados por apellido.

- Cual tabla consultar?
- Cuáles atributos son necesarios y en qué cláusula?

http://dbases.exa.unicen.edu.ar/phppgadmin

Otros operadores de comparación

- 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 las voluntarios cuyo número se encuentra entre 100 y 120*

```
SELECT nombre, apellido, e-mail
FROM voluntario
WHERE nro_voluntario BETWEEN 100 AND 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 y _ : denota un solo carácter

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

```
SELECT nombre, apellido, e_mail
FROM voluntario
WHERE nombre LIKE '_a%n';
```

nombre	apellido	e_mail	telefono	fecha_nacimiento	id_tarea	nro_voluntario	horas_aportadas
Karen	Colmenares	KCOLMENA	515.127.4566	1999-08-10	PU_CLERK	119	2500.00
Jason	Mallin	JMALLIN	650.127.1934	1996-06-14	ST_CLERK	133	3300.00
Karen	Partner	KPARTNER	011.44.1344.467268	1997-01-05	SA_MAN	146	13500.00

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 caracter). SOLO testean valores que son nulos.

Ejemplo: listar los voluntarios que no tengan coordinador.

```
SELECT nombre, apellido, e_mail
FROM voluntario
WHERE id_coordinador IS NULL;
```

nombre	apellido	e_mail	telefono	fecha_nacimiento	id_tarea	nro_voluntario	horas_aportadas	porcentaje
Steven	King	SKING	515.123.4567	1987-06-17	AD_PRES	100	24000.00	*NULL*

Operador IS [NOT] NULL

- Si se comparan valores nulos usando los otros operadores (=, >, etc.)el resultado será siempre FALSO porque un valor nulo no puede ser igual, mayor, distinto, etc. 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 sea menor o igual que 0,10.
  - Algunos porcentajes pueden ser nulos
  - Si deseo incluirlos en el resultado debo explicitar IS NULL.

```
SELECT * FROM voluntario
WHERE  porcentaje<=0.1
       OR  porcentaje IS NULL;
```

Condiciones de comparación 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.

Condiciones de comparación compuestas

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;
```

nro_voluntario	apellido	id_institucion	id_coordinador
120	Weiss	50	100
121	Fripp	50	100
122	Kaufling	50	100
123	Vollman	50	100
124	Mourgos	50	100
141	Rajs	50	124
142	Davies	50	124
143	Matos	50	124
144	Vargas	50	124
196	Walsh	50	124
197	Feeney	50	124
198	OConnell	50	124
199	Grant	50	124

Importante:

-Indentar las cláusulas

-Uso de paréntesis en las condiciones

Qué sucede si se eliminan ()?

...al eliminar ()

nro_voluntario	apellido	id_institucion	id_coordinador
101	Kochhar	90	100
102	De Haan	90	100
114	Raphaely	30	100
120	Weiss	50	100
121	Fripp	50	100
122	Kaufling	50	100
123	Vollman	50	100
124	Mourgos	50	100
141	Rajs	50	124
142	Davies	50	124
143	Matos	50	124
144	Vargas	50	124
145	Russell	80	100
146	Partners	80	100
147	Errazuriz	80	100
148	Cambrault	80	100
149	Zlotkey	80	100
196	Walsh	50	124
197	Feeney	50	124
198	OConnell	50	124
199	Grant	50	124
201	Hartstein	20	100

Estas filas no corresponden a la institución cuyo identificador es 50!!
No es válida la consulta sin los paréntesis en la condición!!!

Orden en la presentación de las tuplas

- El orden de las filas listadas en una consulta es indefinido, se puede utilizar la cláusula **ORDER BY** para ordenar las filas, y se debe colocar como última cláusula del **SELECT**.
- Por defecto si no se especifica el orden es ascendente (**ASC**), pero se puede especificar también **DESC** luego del nombre de la columna, para especificar un orden descendente.
- 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.*

```
SELECT apellido, nombre
 FROM voluntario
WHERE  id_coordinador=124
ORDER BY apellido DESC, nombre;
```

apellido	nombre
Walsh	Alana
Vargas	Peter
Rajs	Trenna
OConnell	Donald
Matos	Randall
Grant	Douglas
Feeney	Kevin
Davies	Curtis

LIMIT and OFFSET (PostgreSQL)

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

```
SELECT lista de atributos
 FROM tabla/s
 [ORDER BY ... ]
 [LIMIT {numero | ALL}]
 [OFFSET numero];
```

- Debe 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.

LIMIT and OFFSET (PostgreSQL)

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

```
SELECT apellido, nombre
FROM voluntario
ORDER BY nro_voluntario
LIMIT 10;
```

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

```
SELECT apellido, nombre
FROM voluntario
ORDER BY nro_voluntario
LIMIT ALL
OFFSET 15;
```

¿Qué son Funciones de Grupo o Agregación?

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

nro_voluntario	apellido	id_institucion	coordinador
121	Fripp	50	100
196	Walsh	50	124
147	Errazuriz	80	100
103	Hunold	60	102
115	Khoo	30	114
185	Bull	50	121
158	McEwen	80	146
175	Hutton	80	149
167	Banda	80	147
187	Cabrio	50	121
193	Everett	50	123
104	Ernst	60	103
179	Johnson	80	149
153	Olsen	80	145
162	Vishney	80	147
142	Davies	50	124
109	Faviet	100	108
163	Greene	80	147
105	Austin	60	103
165	Lee	80	147

Cantidad de horas aportadas por todos los voluntarios

total_de_horas_aportadas
691400.00

■ ■ ■

Funciones de Grupo o Agregación

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*

SELECT [**columna, ...**] función de grupo(columna), ...
FROM tabla/s
[**WHERE** condicion/es]

Funciones de Grupo o Agrupamiento

AVG, SUM y ***STDDEV*** se usan para datos numéricos.

```
SELECT   SUM(horas_aportadas),
     AVG(horas_aportadas),
     MAX(horas_aportadas),
     MIN(horas_aportadas)
FROM voluntario;
```

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;
```

Funciones de Grupo o Agrupamiento

- **COUNT(*)** devuelve el número de filas de una tabla.

```
SELECT   COUNT(*)
FROM voluntario;
```

- **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.*

```
SELECT   COUNT(ciudad) AS cantidad__de_ciudades
FROM direccion;
```

Funciones de Grupo y valores nulos

- Las funciones de grupo ignoran los valores nulos del atributo.

```
SELECT   AVG(porcentaje) AS
     Porcentaje_promedio
FROM voluntario;
```

porcentaje_promedio
0.22285714285714285714

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

```
SELECT   AVG(COALESCE(porcentaje, 0)) AS
     Porcentaje_promedio
FROM voluntario;
```

porcentaje_promedio
0.07289719626168224299

Ejemplos de Funciones de Grupo o Agrupamiento

```
SELECT   COUNT(*) AS
    cantidad_de_voluntarios
FROM voluntario;
```

cantidad_de_voluntarios
107

Alias de columna

```
SELECT   SUM(horas_aportadas) AS
    Horas_trabajadas
FROM voluntario v WHERE
    v.id_coordinador=120;
```

Alias de tabla

horas_trabajadas
22100.00

```
SELECT   MAX(horas_aportadas) maximo,
    MIN(horas_aportadas) minimo,
    MAX(horas_aportadas) –
    MIN(horas_aportadas) diferencia
FROM voluntario v
  WHERE v.id_coordinador=120;
```

maximo	minimo	diferencia
3200.00	2200.00	1000.00

Recomendaciones

- ✓ Completar con textos de la bibliografía recomendada (en todos se explica SQL y se plantean ejemplos)
- ✓ Consultar la Documentación del SQL estándar y de PostgreSQL.
- ✓ Resolver ejercicios (propuestos en la práctica, laboratorio, etc.)
- ✓ Probar ejercicios en PostgreSQL (accesible vía web… en la página está la URL y los datos para conexión), aprovechar TODAS las prácticas para consultar!!

Practicar…

Practicar…

Practicar… !!!

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.

SELECT lista columnas, **función de grupo (columna)**
FROM <lista tablas>
[**WHERE** condición]
[**GROUP BY** expresión de grupo | lista columnas
[**ORDER BY** <lista atributos orden>];

La clausula **GROUP BY** *especifica como se deben agrupar las filas seleccionadas.*

Creación de Grupos de Datos o Agregaciones

nro_voluntario	apellido	id_institucion	coordinador
121	Fripp	50	100
196	Walsh	50	124
147	Errazuriz	80	100
103	Hunold	60	102
115	Khoo	30	114
185	Bull	50	121
158	McEwen	80	146
175	Hutton	80	149
167	Banda	80	147
187	Cabrio	50	121
193	Everett	50	123
104	Ernst	60	103
179	Johnson	80	149
153	Olsen	80	145
162	Vishney	80	147
142	Davies	50	124
109	Faviet	100	108
163	Greene	80	147
105	Austin	60	103
165	Lee	80	147

■ ■ ■

```
SELECT   id_institucion, COUNT(*) AS
     cantidad_voluntarios
FROM voluntario
GROUP BY id_institucion;
```

¿Cuántos voluntarios tiene cada Institución?

id_institucion	cantidad_voluntarios
90	3
NULL	1
20	2
100	6
40	1
110	2
80	34
70	1
50	45
60	5
30	6
10	1

Sintaxis de la clausula GROUP BY

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*

```
SELECT   id_institucion, MAX(horas_aportadas)
FROM voluntario
GROUP BY id_institucion;
```

Sintaxis de la clausula GROUP BY

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

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

```
SELECT   AVG(porcentaje)
FROM voluntario
GROUP BY id_institucion;
```

Restringir los resultados de los grupos

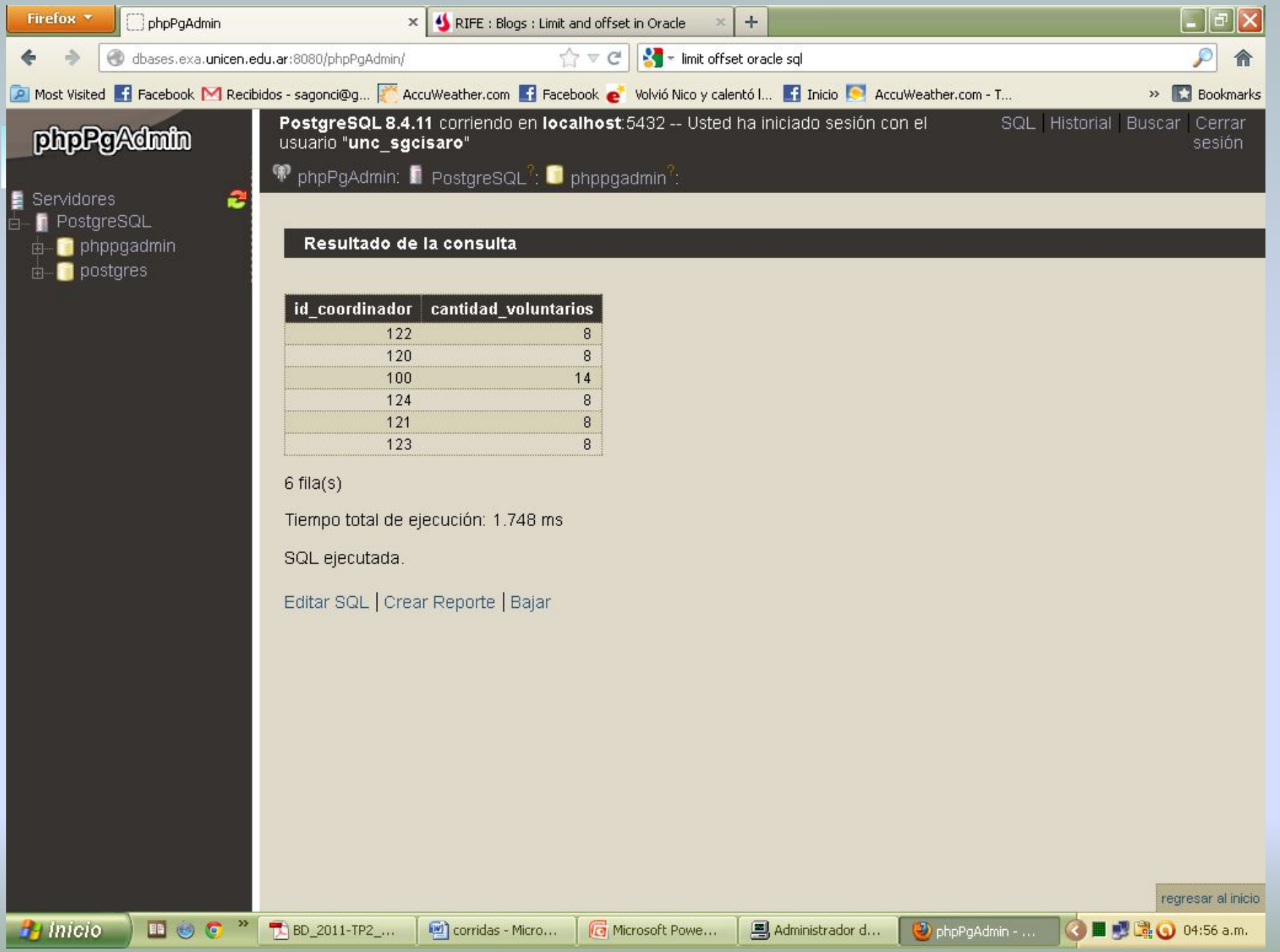
Se puede anexar la cláusula **HAVING** para restringir grupos

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**

nro_voluntario	apellido	id_institucion	coordinador
121	Fripp	50	100
196	Walsh	50	124
147	Errazuriz	80	100
103	Hunold	60	102
115	Khoo	30	114
185	Bull	50	121
158	McEwen	80	146
175	Hutton	80	149
167	Banda	80	147
187	Cabrio	50	121
193	Everett	50	123
104	Ernst	60	103
179	Johnson	80	149
153	Olsen	80	145
162	Vishney	80	147
142	Davies	50	124
109	Faviet	100	108
163	Greene	80	147
105	Austin	60	103
165	Lee	80	147

Coordinadores con más de 7 voluntarios

```
SELECT   id_coordinador, COUNT(*) AS
     cantidad_de_voluntarios
FROM voluntario
GROUP BY id_coordinador
HAVING COUNT(*) > 7;
```


Firefox
phpPgAdmin
RIFE : Blogs : Limit and offset in Oracle
dbases.exa.unicen.edu.ar:8080/phpPgAdmin/
limit offset oracle sql
Most Visited
Facebook
Recibidos - sagonci@g...
AccuWeather.com
Facebook
Volvió Nico y calentó l...
Inicio
AccuWeather.com - T...
Bookmarks
phpPgAdmin
PostgreSQL 8.4.11 corriendo en localhost:5432 -- Usted ha iniciado sesión con el usuario "unc_sgcisaro"
SQL | Historial | Buscar | Cerrar sesión
phpPgAdmin: PostgreSQL: phppgadmin:
Servidores
PostgreSQL
phppgadmin
postgres
Resultado de la consulta
id_coordinador | cantidad_voluntarios
122 | 8
120 | 8
100 | 14
124 | 8
121 | 8
123 | 8
6 fila(s)
Tiempo total de ejecución: 1.748 ms
SQL ejecutada.
Editar SQL | Crear Reporte | Bajar
regresar al inicio
Inicio
BD_2011-TP2_...
corridas - Micro...
Microsoft Powe...
Administrador d...
phpPgAdmin - ...
04:56 a.m.

Funciones de grupo no válidas

- ✓ No se puede utilizar la cláusula **WHERE** para restringir grupos.
- ✓ Se debe utilizar la cláusula **HAVING** para restingir 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;
```

Sentencia NO VALIDA

Error de SQL:

```
ERROR:  aggregates not allowed in WHERE clause at character 105
```

En la declaración:

```
SELECT id_coordinador, count(*) as Cantidad_voluntarios

 FROM  unc_esq_voluntario.voluntario  where  count(*) >7
group by id_coordinador

;
```

Para Recordar!!!

- La sentencia SQL empleada para la recuperación de los datos a partir de las tablas cargadas en la base de datos es el **SELECT**.
- **SELECT** identifica las columnas a recuperar – **EL QUE**
- **FROM** identifica la/s tabla/s - **DE DONDE** obtener los datos**.**
- **WHERE** se usa para realizar las restricciones sobre los datos
- Para eliminar los valores repetidos se debe usar la cláusula **DISTINCT**.
- Los operadores de comparación se utilizan en la cláusula WHERE para comparar expresiones. Usar paréntesis y sangrías para mejorar la legibilidad.
- **ORDER BY** puede usarse para ordenar las filas, y se debe colocar como última cláusula de la sentencia SELECT
- Las funciones de agregación operan sobre conjuntos de filas para proporcionar un resultado por grupo.
- GROUP BY *especifica como se deben agrupar las filas seleccionadas. Todas las columnas de la lista SELECT, excepto las funciones de grupo, deben estar en la clausula GROUP BY.* **No** se pueden utilizar funciones de grupo en la cláusula WHERE.

Recomendaciones

- ✓ Completar con textos de la bibliografía recomendada (en todos se explica SQL y se plantean ejemplos)
- ✓ Consultar la Documentación del SQL estándar y de PostgreSQL.
- ✓ Resolver ejercicios (propuestos en la práctica, laboratorio, etc.)
- ✓ Probar ejercicios en PostgreSQL (accesible vía web… en la página está la URL y los datos para conexión), aprovechar TODAS las prácticas para consultar!!

Practicar…

Practicar…

Practicar… !!!