



FACULTAD DE CIENCIAS  
**EXACTAS**  
UNIVERSIDAD NACIONAL DEL CENTRO  
DE LA PROVINCIA DE BUENOS AIRES

# Tecnicatura Universitaria en Desarrollo de Aplicaciones Informáticas (TUDAI)

## Base de Datos

Tema 4: Consultas SQL - Parte 2

2  
0  
2  
5

# Repaso

## SQL

**SQL**, or *Structured Query Language*, is a language to talk to databases. It allows you to select specific data and to build complex reports. Today, SQL is a universal language of data. It is used in practically all technologies that process data.

## SAMPLE DATA

COUNTRY			
id	name	population	area
1	France	66600000	640680
2	Germany	80700000	357000
...	...	...	...

CITY				
id	name	country_id	population	rating
1	Paris	1	2243000	5
2	Berlin	2	3460000	3
...	...	...	...	...

## QUERYING SINGLE TABLE

Fetch all columns from the country table:

```
SELECT *  
FROM country;
```

Fetch id and name columns from the city table:

```
SELECT id, name  
FROM city;
```

Fetch city names sorted by the rating column in the default ASCending order:

```
SELECT name  
FROM city  
ORDER BY rating [ASC];
```

Fetch city names sorted by the rating column in the DESCending order:

```
SELECT name  
FROM city  
ORDER BY rating DESC;
```

# SQL

**SQL**, or *Structured Query Language*, is a language to talk to databases. It allows you to select specific data and to build complex reports. Today, SQL is a universal language of data. It is used in practically all technologies that process data.

## SAMPLE DATA

COUNTRY			
id	name	population	area
1	France	66600000	640680
2	Germany	80700000	357000
...	...	...	...

CITY				
id	name	country_id	population	rating
1	Paris	1	2243000	5
2	Berlin	2	3460000	3
...	...	...	...	...

## TEXT OPERATORS

Fetch names of cities that start with a 'P' or end with an 's':

```
SELECT name
FROM city
WHERE name LIKE 'P%'
      OR name LIKE '%s';
```

Fetch names of cities that start with any letter followed by 'ublin' (like Dublin in Ireland or Lublin in Poland):

```
SELECT name
FROM city
WHERE name LIKE '_ublin';
```

# Repaso

## FILTERING THE OUTPUT

### COMPARISON OPERATORS

Fetch names of cities that have a rating above 3:

```
SELECT name
FROM city
WHERE rating > 3;
```

Fetch names of cities that are neither Berlin nor Madrid:

```
SELECT name
FROM city
WHERE name != 'Berlin'
      AND name != 'Madrid';
```

# Repaso

## SQL

**SQL**, or *Structured Query Language*, is a language to talk to databases. It allows you to select specific data and to build complex reports. Today, SQL is a universal language of data. It is used in practically all technologies that process data.

## SAMPLE DATA

COUNTRY			
id	name	population	area
1	France	66600000	640680
2	Germany	80700000	357000
...	...	...	...

CITY				
id	name	country_id	population	rating
1	Paris	1	2243000	5
2	Berlin	2	3460000	3
...	...	...	...	...

## ALIASES

### COLUMNS

```
SELECT name AS city_name
FROM city;
```

### TABLES

```
SELECT co.name, ci.name
FROM city AS ci
JOIN country AS co
  ON ci.country_id = co.id;
```

## OTHER OPERATORS

Fetch names of cities that have a population between 500K and 5M:

```
SELECT name
FROM city
WHERE population BETWEEN 500000 AND 5000000;
```

Fetch names of cities that don't miss a rating value:

```
SELECT name
FROM city
WHERE rating IS NOT NULL;
```



# SQL

**SQL**, or *Structured Query Language*, is a language to talk to databases. It allows you to select specific data and to build complex reports. Today, SQL is a universal language of data. It is used in practically all technologies that process data.

## SAMPLE DATA

COUNTRY			
id	name	population	area
1	France	66600000	640680
2	Germany	80700000	357000
...	...	...	...

CITY				
id	name	country_id	population	rating
1	Paris	1	2243000	5
2	Berlin	2	3460000	3
...	...	...	...	...

## AGGREGATION AND GROUPING

**GROUP BY** groups together rows that have the same values in specified columns. It computes summaries (aggregates) for each unique combination of values.

CITY		
id	name	country_id
1	Paris	1
101	Marseille	1
102	Lyon	1
2	Berlin	2
103	Hamburg	2
104	Munich	2
3	Warsaw	4
105	Cracow	4



CITY	
country_id	count
1	3
2	3
4	2

## AGGREGATE FUNCTIONS

- **avg(expr)** – average value for rows within the group
- **count(expr)** – count of values for rows within the group
- **max(expr)** – maximum value within the group
- **min(expr)** – minimum value within the group
- **sum(expr)** – sum of values within the group

# Repaso

# Repaso

## EXAMPLE QUERIES

Find out the number of cities:

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

Find out the number of cities with non-null ratings:

```
SELECT COUNT(rating)  
FROM city;
```

Find out the number of distinctive country values:

```
SELECT COUNT(DISTINCT country_id)  
FROM city;
```

Find out the smallest and the greatest country populations:

```
SELECT MIN(population), MAX(population)  
FROM country;
```

Find out the total population of cities in respective countries:

```
SELECT country_id, SUM(population)  
FROM city  
GROUP BY country_id;
```

Find out the average rating for cities in respective countries if the average is above 3.0:

```
SELECT country_id, AVG(rating)  
FROM city  
GROUP BY country_id  
HAVING AVG(rating) > 3.0;
```

## AGGREGATION AND GROUPING

GROUP BY **groups** together rows that have the same values in specified columns. It computes summaries (aggregates) for each unique combination of values.

CITY		
id	name	country_id
1	Paris	1
101	Marseille	1
102	Lyon	1
2	Berlin	2
103	Hamburg	2
104	Munich	2
3	Warsaw	4
105	Cracow	4



CITY	
country_id	count
1	3
2	3
4	2

# Consultas SQL

*Repaso:* la sentencia del lenguaje empleada para la recuperación de datos: **SELECT** y cada una de sus cláusulas.

```
SELECT * | { [DISTINCT] columna | expresión [alias], función_grupo...}  
FROM lista de tablas  
[WHERE condiciones]  
[GROUP BY expresión de agrupamiento]  
[HAVING condición de grupo]  
[ORDER BY lista de columnas [ASC|DESC]]  
[LIMIT { nro filas| ALL }] [OFFSET fila desde ]
```

# Ensamblares (JOIN)

El operador **JOIN** combina dos tablas según una condición para obtener registros compuestos por atributos de las dos tablas combinadas. Existen diferentes maneras hacerlo:

## **JOIN INTERNO**

- **INNER JOIN** donde la condición que acota el resultado es una comparación de igualdad.
- **NATURAL JOIN**: Es un caso especial de equi-join en el que en el caso de existir columnas con el mismo nombre en las relaciones que se combinan, sólo se incluirá una de ellas en el resultado de la combinación.
- Si los nombres de columnas se repiten, hay que anteponer el nombre de la tabla para evitar ambigüedades.



# Ensamblados (JOIN)

El tipo de JOIN anterior sólo se queda con los registros que tienen valores iguales en las columnas que compara.

Pero puede suceder que perdamos registros que nos interesan de alguna de las dos tablas por ejemplo si quisiéramos recuperar todos los voluntarios.

Por eso es que SQL dispone de **JOIN EXTERNO** (OUTER JOIN)

# Vamos a crear datos de prueba...

**--Vamos a crear una tabla VOLUNTARIO con algunos voluntarios, primero sólo los que son directores de instituciones (11 en total)**

```
create table voluntario as
select nro_voluntario, apellido, nombre, id_tarea, id_institucion
from unc_esq_voluntario.voluntario
where nro_voluntario in (select id_director from unc_esq_voluntario.institucion);
```

**--Vamos a crear una tabla INSTITUCIONES con algunos instituciones (las <= al id 140)**

```
create table institucion
as select id_institucion, nombre_institucion
from unc_esq_voluntario.institucion
where id_institucion <= 140;
```

**--Vamos a crear una tabla TAREA con todas las tareas**

```
create table tarea
as select id_tarea, nombre_tarea
from unc_esq_voluntario.tarea;
```

**--Cambiamos algunos datos**

```
update voluntario set id_tarea= null where nro_voluntario = 108;
update voluntario set id_institucion= null where nro_voluntario = 145;
```

id_tarea	nombre_tarea
AC_ACCOUNT	FISCALIZACION DE RECURSOS NATURALES
AC_MGR	ORG.CAMPAÑAS LIMPIEZA
AD_ASST	AISTENCIA ANCIANOS
AD_PRES	PROMOCION
AD_VP	PREVENCION
FI_ACCOUNT	PLANTACION
FI_MGR	FORESTACION
HR_REP	MAESTRO ESPECIAL
IT_PROG	CONSTRUCTOR
MK_MAN	ASISTENCIA A ENFERMOS
MK_REP	COCINERO
OT_NEW	Nueva Tarea
PR_REP	RELACIONES INSTITUCIONALES
PU_CLERK	CLASIFICACION DE ALIMENTOS
PU_MAN	ORGANIZACION DE COLECTAS
SA_MAN	CLASES ESPECIALES
SA_REP	ORGANIZACION CAMPAMENTOS RECREATIVOS
SH_CLERK	AYUDA DISCAPACITADOS
ST_CLERK	ATENCION DE ROPERITOS
ST_MAN	ATENCION DE COMEDORES

Se puede ver que cada voluntario en la tabla de voluntario tiene una tarea, excepto **Nancy Greenberg**. Cada voluntario tiene una institución, excepto **John Russell**. Hay instituciones de la tabla instituciones que no están asignadas a ningún voluntario (80, 120, 130, 140). Y, Nancy Greenberg tiene una institución asignada, pero no tiene tarea.

nro_voluntario	nombre	apellido	id_tarea	id_institucion
100	Steven	King	AD_PRES	90
103	Alexander	Hunold	IT_PROG	60
114	Den	Raphaely	PU_MAN	30
121	Adam	Fripp	ST_MAN	50
200	Jennifer	Whalen	AD_ASST	10
201	Michael	Hartstein	MK_MAN	20
203	Susan	Mavris	HR_REP	40
204	Hermann	Baer	PR_REP	70
205	Shelley	Higgins	AC_MGR	110
108	Nancy	Greenberg	NULL	100
145	John	Russell	SA_MAN	NULL

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

# Qué es un múltiple JOIN?

```
SELECT v.apellido, v.nombre
FROM voluntario v INNER JOIN tarea t ON
    v.id_tarea = t.id_tarea
    INNER JOIN institucion i ON
    v.id_institucion = i.id_institucion;
```

apellido	nombre
Whalen	Jennifer
Hartstein	Michael
Raphaely	Den
Mavris	Susan
Fripp	Adam
Hunold	Alexander
Baer	Hermann
King	Steven
Higgins	Shelley

9 fila(s)

En la consulta anterior, utilizamos el JOIN múltiple para recuperar sólo aquellos **voluntarios** a los que se les asignó un tarea y una institución. De los 11 registros de la tabla voluntario, sólo 9 se retornaron; ni Nancy Greenberg ni John Russell están en el resultado de la consulta.

nro_voluntario	nombre	apellido	id_tarea	id_institucion
100	Steven	King	AD_PRES	90
103	Alexander	Hunold	IT_PROG	60
114	Den	Raphaely	PU_MAN	30
121	Adam	Fripp	ST_MAN	50
200	Jennifer	Whalen	AD_ASST	10
201	Michael	Hartstein	MK_MAN	20
203	Susan	Mavris	HR_REP	40
204	Hermann	Baer	PR_REP	70
205	Shelley	Higgins	AC_MGR	110
108	Nancy	Greenberg	NULL	100
145	John	Russell	SA_MAN	NULL

# JOIN

Tenga en cuenta que todas las operaciones de JOIN se realizan de izquierda a derecha.

1. Primer paso, las tablas del primer JOIN coinciden (tablas voluntario y tarea). Como resultado, se crea una tabla intermedia.
2. Segundo paso, esta tabla intermedia (tratada como la tabla izquierda) se une con la otra tabla (tabla institución) utilizando el segundo JOIN.

Recuerde que un solo JOIN (cualquiera sea su tipo) produce una sola tabla intermedia durante una consulta de varios JOINS.



# LEFT-RIGHT-FULL JOIN

Es posible hacer diferentes tipos de combinaciones en una consulta de ensambles múltiple.

***Supongamos que queremos consultar todas las tareas que realizan los voluntarios de alguna institución y también las tareas que no realiza nadie.***

¿Cómo lo hacemos? Podríamos hacer esto:

```
SELECT nombre_tarea, apellido, nombre  
FROM tarea t LEFT JOIN voluntario v  
                ON (t.id_tarea = v.id_tarea)  
JOIN institucion i ON (i.id_institucion = v.id_institucion);
```

# LEFT-RIGHT-FULL JOIN

¿Qué pasó?

apellido	nombre	id_tarea	id_institucion
Whalen	Jennifer	AD_ASST	10
Hartstein	Michael	MK_MAN	20
Raphaely	Den	PU_MAN	30
Mavris	Susan	HR_REP	40
Fripp	Adam	ST_MAN	50
Hunold	Alexander	IT_PROG	60
Baer	Hermann	PR_REP	70
King	Steven	AD PRES	90
Higgins	Shelley	AC_MGR	110

9 fila(s)

# LEFT-RIGHT-FULL JOIN

¿Qué pasó? La tabla derivada producto del LEFT JOIN entre tarea y voluntario, posee **todas** las tareas, pero al ensamblarla con institución (INNER JOIN) quedan sólo las que coinciden en el id\_institucion.

¿Cómo resolvemos este problema?

```
SELECT nombre_tarea, apellido, nombre  
FROM voluntario v JOIN institucion i  
    ON (i.id_institucion = v.id_institucion)  
RIGHT JOIN tarea t ON (t.id_tarea = v.id_tarea);
```

Alguna otra solución?

# LEFT-RIGHT-FULL JOIN

Para el caso del FULL JOIN completará con nulos para aquellos tareas que no sean realizadas por ningún voluntario, al igual que las instituciones.

```
SELECT apellido, nombre, t.id_tarea,
i.id_institucion
FROM tarea t FULL JOIN voluntario v
  ON (t.id_tarea = v.id_tarea)
FULL JOIN institucion i
  ON (i.id_institucion = v.id_institucion)
```

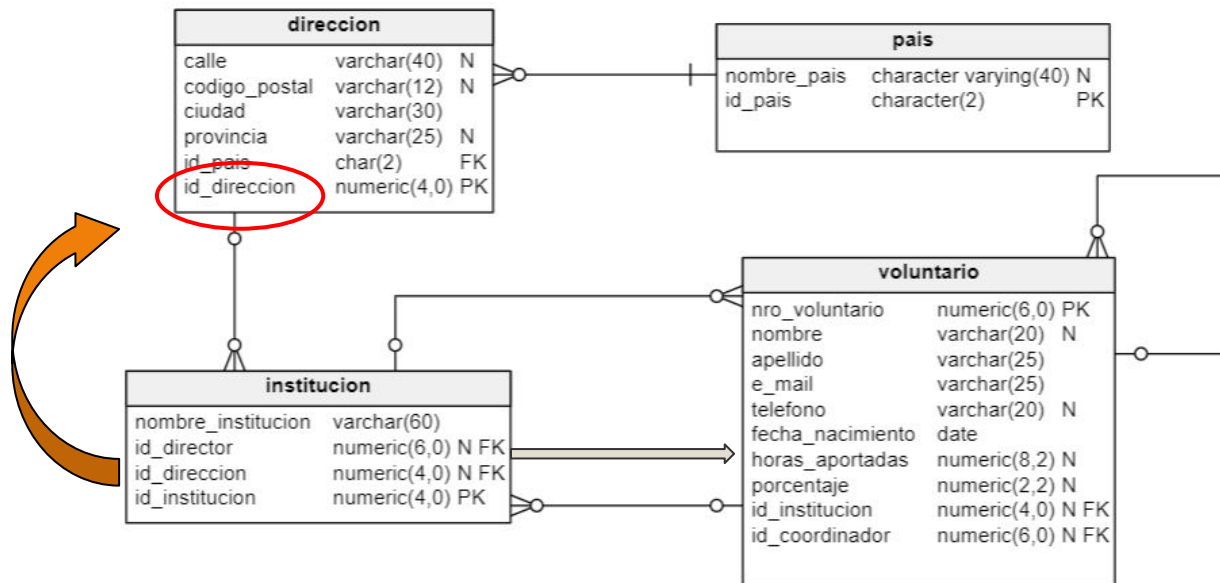
apellido	nombre	id_tarea	id_institucion
Baer	Hermann	PR_REP	70
Fripp	Adam	ST_MAN	50
Greenberg	Nancy	NULL	100
Hartstein	Michael	MK_MAN	20
Higgins	Shelley	AC_MGR	110
Hunold	Alexander	IT_PROG	60
King	Steven	AD_PRES	90
Mavris	Susan	HR_REP	40
Raphaely	Den	PU_MAN	30
Russell	John	SA_MAN	NULL
Whalen	Jennifer	AD_ASST	10
NULL	NULL	AC_ACCOUNT	NULL
NULL	NULL	AD_VP	NULL
NULL	NULL	FI_ACCOUNT	NULL
NULL	NULL	FI_MGR	NULL
NULL	NULL	MK_REP	NULL
NULL	NULL	OT_NEW	NULL
NULL	NULL	PU_CLERK	NULL
NULL	NULL	SA_REP	NULL
NULL	NULL	SH_CLERK	NULL
NULL	NULL	ST_CLERK	NULL
NULL	NULL	NULL	80
NULL	NULL	NULL	120
NULL	NULL	NULL	130
NULL	NULL	NULL	140

25 fila(s)

# Consultas de más de una tabla

Ejemplo: seleccionar el nombre y apellido de los voluntarios del estado (provincia) de Texas.

Tenemos que revisar desde el esquema las condiciones de ensamble o subconsulta entre las distintas tablas.





# Consultas Anidadas (subconsulta)

La cláusula WHERE puede contener un SELECT anidado, como una consulta conjunta en 2 pasos.


**Ejemplo:** seleccionar el nombre de la/s instituciones del estado (provincia) de Texas.



**Una UNICA consulta que resuelva ...:**

```
SELECT id_direccion  
FROM direccion d  
WHERE d.provincia = 'Texas';  
1400
```

```
SELECT nombre_institucion  
FROM institucion i  
WHERE i.id_direccion = 1400;
```



# Consultas Anidadas (subconsulta)

Seleccionar el nombre de la/s instituciones del estado (provincia) de Texas.

```
SELECT nombre_institucion  
FROM institucion i  
WHERE i.id_direccion = (SELECT id_direccion      1400  
                        FROM direccion d  
                        WHERE d.provincia = 'Texas');;
```

Solo se puede utilizar si  
la subconsulta devuelve  
**UNA SOLA FILA**

**Muy importante!!!.....**qué ocurriría si hay más de un resultado para Texas?

Hay que utilizar operadores para subqueries que retornam **multiple-rows** (IN, ANY, ALL)

# Subconsultas de una Sola Fila

Es posible utilizar los siguientes operadores de comparación:

Operador	Significado
=	Igual que
>	Mayor que
>=	Mayor o igual que
<	Menor que
<=	Menor o igual que
<>	No igual a



# Consultas Anidadas

Ejemplo con uso de funciones de grupo:

Seleccionar todos los voluntarios que aportan la mínima cantidad de horas:

```
SELECT nombre, apellido  
FROM voluntario  
WHERE horas_aportadas = (SELECT MIN(horas_aportadas)  
                        FROM voluntario);
```

Es responsabilidad de quien escribe el query asegurar que el subquery devolverá una sola fila. Si el subquery devuelve más de 1 fila, dará error



# Consultas Anidadas

Cláusula **HAVING** en subconsultas

Se ejecuta en primer lugar la subconsulta. Devuelve resultados a la cláusula **HAVING** (correspondiente a la consulta principal) que luego se usan para chequear la condición de grupo.

Ejemplo: Instituciones donde la **mínima cantidad de horas que aportan sus voluntarios** es mayor que la *mínima cantidad de horas que aportan los de la institución 40*.

```
SELECT id_institucion, MIN(horas_aportadas)
FROM voluntario
GROUP BY id_institucion
HAVING MIN(horas_aportadas) > (SELECT MIN(horas_aportadas)
                                FROM voluntario WHERE id_institucion = 40);
```

# Ejercicio

¿Cuáles son las tareas cuyo promedio de horas aportadas por tarea de los voluntarios nacidos a partir del año 1995 es superior al promedio general de dicho grupo de voluntarios?

- 1 - Promedio de horas aportadas por tarea de los voluntarios nacidos a partir del año 1995.
- 2 - Promedio general de las horas aportadas por los voluntarios nacidos a partir del año 1995

# Consultas de más de una Tabla

Seleccionar el nombre y apellido de los voluntarios del estado (provincia) de Texas

```
SELECT nombre, apellido  
FROM voluntario v, institucion i, direccion d
```

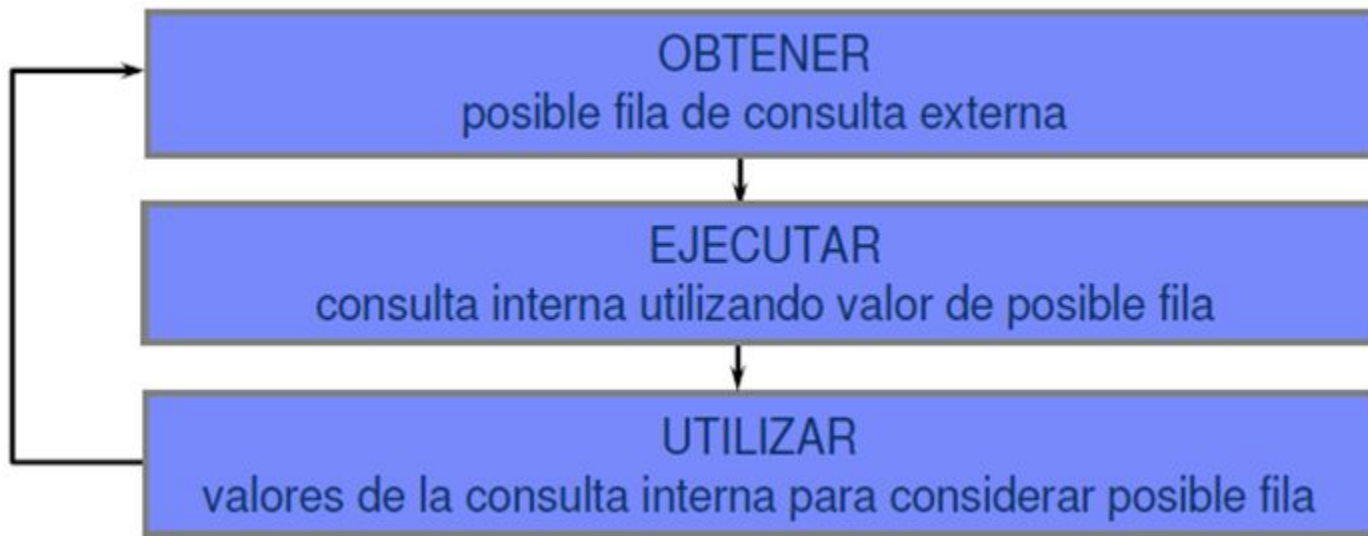
```
WHERE v.id_institucion = i.id_institucion  
       AND i.id_direccion = d.id_direccion
```

```
AND d.provincia = 'Texas'
```

**Generalmente condiciones  
de ensamble = pares PK/FK  
(cantidad de tablas – 1)**

# Subconsultas Correlacionadas

Se utilizan para el procesamiento fila a fila. Cada subconsulta se ejecuta una vez por cada fila de la consulta externa



# Uso de Subconsulta Correlacionada

Ejemplo: Buscar todos los voluntarios que aportan más horas que el promedio de horas aportadas por los voluntarios de la institución a la que pertenecen.

Cada vez que se procesa una fila de la consulta externa, se evalúa la consulta interna

apellido	horas_aportadas	id_institucion	
Hartstein	13000.00	20	9500.00
Raphaely	11000.00	30	4150.00
Ladwig	3600.00	50	3475.56
Rajs	3500.00	50	3475.56
Sarchand	4200.00	50	3475.56
Bull	4100.00	50	3475.56
Chung	3800.00	50	3475.56
Weiss	8000.00	50	3475.56
Fripp	8200.00	50	3475.56
Kaufling	7900.00	50	3475.56
Vollman	6500.00	50	3475.56
Mourgos	5800.00	50	3475.56
Dilly	3600.00	50	3475.56
Bell	4000.00	50	3475.56
Everett	3900.00	50	3475.56
Ernst	6000.00	60	5760.00
Hunold	9000.00	60	5760.00
Fox	9600.00	80	8955.88

```
SELECT apellido, horas_aportadas, id_institucion  
FROM voluntario V1  
WHERE horas_aportadas >  
      (SELECT AVG(horas_aportadas)  
        FROM voluntario V2  
        WHERE V2.id_institucion = V1.id_institucion);
```



# Operador IN

El operador IN permite especificar múltiples valores en una cláusula WHERE. Podría decirse que es una forma abreviada de varias condiciones OR. La forma es:

```
SELECT columna(s)  
FROM tabla  
WHERE columna(s) IN (valor(es), valor(es)2, ...);
```

# Operador IN

Ejemplo: Listar los voluntarios que realizan las tareas ST\_CLERK, SA\_MAN, SA\_REP o IT\_PROG

nro_voluntario	apellido	nombre
134	Rogers	Michael
135	Gee	Ki
136	Philtanker	Hazel
137	Ladwig	Renske
138	Stiles	Stephen
139	Seo	John
140	Patel	Joshua
141	Rajs	Trenna
142	Davies	Curtis
143	Matos	Randall
144	Vargas	Peter
145	Russell	John
146	Partners	Karen
147	Errazuriz	Alberto
148	Cambrault	Gerald
149	Zlotkey	Eleni
150	Tucker	Peter
151	Bernstein	David
152	Hall	Peter

```
SELECT nro_voluntario, apellido, nombre  
FROM voluntario  
WHERE id_tarea IN ('ST_CLERK', 'SA_MAN',  
                  'SA_REP', 'IT_PROG');
```

# Uso del Operador IN

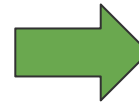
La lista de valores puede ser reemplazada por una subconsulta (consulta interna), que retorna una lista de valores que la consulta exterior luego emplea para recuperar los datos

Se recomienda probar las subconsultas antes de incluirlas en una consulta exterior, así puede verificar que retorna lo necesario, porque a veces resulta difícil verlo como consultas anidadas

*Ejemplo:* Liste el apellido y nombre de los empleados que trabajan en departamentos de Argentina.

# Uso del Operador IN

```
SELECT e.nombre, e.apellido
FROM empleado e
WHERE (e.id_departamento, e.id_distribuidor) IN (
    SELECT d.id_departamento, d.id_distribuidor
    FROM departamento d
    WHERE d.id_ciudad IN (
        SELECT c.id_ciudad
        FROM ciudad c
        WHERE c.id_pais IN (
            SELECT p.id_pais
            FROM pais p
            WHERE nombre_pais = 'ARGENTINA'
        )
    )
);
```



Pais	
id_pais	
AR	

# Uso del Operador IN

```
SELECT e.nombre, e.apellido
FROM empleado e
WHERE (e.id_departamento, e.id_distribuidor) IN (
    SELECT d.id_departamento, d.id_distribuidor
    FROM departamento d
    WHERE d.id_ciudad IN (
        SELECT c.id_ciudad
        FROM ciudad c
        WHERE c.id_pais IN (
            SELECT p.id_pais
            FROM pais p
            WHERE p.id_pais = 'AR'
        )
    )
)
```

2

id_pais
AR

La sub-consulta  
retorna 200 filas en  
7,364 ms.

id_ciudad
752
910
30
427
516
1560
...

# Uso del Operador IN

```
SELECT e.nombre, e.apellido
FROM empleado e
WHERE (e.id_departamento, e.id_distribuidor) IN (
  SELECT d.id_departamento, d.id_distribuidor
  FROM departamento d
  WHERE d.id_ciudad IN (
```

3

id_ciudad
752
910
30
...

La sub-consulta  
retorna 6 filas en  
7,208 ms

id_depto	id_dist
21	999
55	702
71	194
32	860
90	428
71	376

```
)
);
```

# Uso del Operador IN

```
SELECT e.nombre, e.apellido
```

```
4 FROM empleado e
```

```
WHERE (e.id_departamento, e.id_distribuidor) IN (
```

id_depto	id_dist
21	999
55	702
71	194
32	860
90	428
71	376



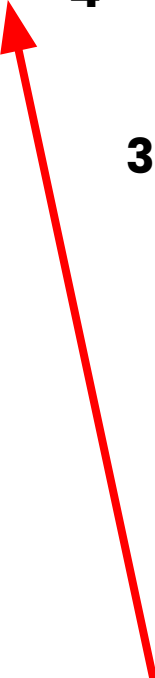
La sub-consulta  
retorna 227 filas en  
18,128 ms

nombre	apellido
Nicolas	Casta#O
Maria Belen	Schoeder
Hilda	Agostinelli
Marcelo	Bayle
Horacio A.	Arisnavarreta
...	....

```
) ;
```



# Uso del Operador IN



```
SELECT e.nombre, e.apellido
FROM empleado e
WHERE (e.id_departamento, e.id_distribuidor) IN (
    SELECT d.id_departamento, d.id_distribuidor
    FROM departamento d
    WHERE d.id_ciudad IN (
        SELECT c.id_ciudad
        FROM ciudad c
        WHERE c.id_pais IN (
            SELECT p.id_pais
            FROM pais p
            WHERE nombre_pais = 'ARGENTINA'
        )
    )
);
```

4

3

2

1

# Uso del Operador EXISTS

- El operador **EXISTS** comprueba la existencia de filas en el conjunto de filas del resultado de la consulta.
- Si se encuentra un valor de fila de la subconsulta:
  - La búsqueda no continúa en la consulta interna.
  - Se señala a la condición como **TRUE**.
- Si no se encuentra un valor de fila de la subconsulta:
  - Se señala a la condición como **FALSE**.
  - La búsqueda continúa en la consulta interna.

# Uso del Operador EXISTS

```
1 SELECT e.nombre, e.apellido
FROM empleado e
WHERE EXISTS (
    SELECT 'X'
    FROM departamento d
    WHERE e.id_departamento=d.id_departamento
    AND e.id_distribuidor=d.id_distribuidor
    AND EXISTS (
        SELECT 'Y'
        FROM ciudad c
        WHERE d.id_ciudad = c.id_ciudad
        AND EXISTS (
            SELECT 'Z'
            FROM pais p
            WHERE p.id_pais=c.id_pais
            AND nombre_pais = 'ARGENTINA'))));
```

Empleado

nombre	apellido	dep	dist	...
Marcelo	Bayle	21	999	...
...	...	..	..	..

**Toma la primer tupla de “Empleado”.**

# Uso del Operador EXISTS

```
SELECT e.nombre, e.apellido
FROM empleado e
WHERE EXISTS (
    SELECT 'X'
    FROM departamento d
    WHERE e.id_departamento=d.id_departamento
    AND e.id_distribuidor=d.id_distribuidor
    AND EXISTS (
        SELECT 'Y'
        FROM ciudad c
        WHERE d.id_ciudad = c.id_ciudad
        AND EXISTS (
            SELECT 'Z'
            FROM pais p
            WHERE p.id_pais=c.id_pais
            AND nombre_pais = 'ARGENTINA')));
```

Empleado

nombre	apellido	dep	dist	...
Marcelo	Bayle	21	999	...
...	...	..	..	..

busca la tupla con  
id\_departamento = 21  
e id\_distribuidor = 999

Departamento

dep	dist	ciu	...
....	...	...	...
21	237	15736	...
...	...	...	...

# Uso del Operador EXISTS

```
SELECT e.nombre, e.apellido
FROM empleado e
WHERE EXISTS (
    SELECT 'X'
    FROM departamento d
    WHERE e.id_departamento=d.id_departamento
    AND e.id_distribuidor=d.id_distribuidor
    AND EXISTS (
        SELECT 'Y'
        FROM ciudad c
        WHERE d.id_ciudad = c.id_ciudad
        AND EXISTS (
            SELECT 'Z'
            FROM pais p
            WHERE p.id_pais=c.id_pais
            AND nombre_pais = 'ARGENTINA'))));
```

3

Departamento

dep	dist	ciu	...
....	...	...	...
21	237	15736	...
...	...	...	...

Ciudad

ciu	pais	...
...	...	...
15736	AR	..
...	...	...

# Uso del Operador EXISTS

```
SELECT e.nombre, e.apellido
FROM empleado e
WHERE EXISTS (
    SELECT 'X'
    FROM departamento d
    WHERE e.id_departamento=d.id_departamento
    AND e.id_distribuidor=d.id_distribuidor
    AND EXISTS (
        SELECT 'Y'
        FROM ciudad c
        WHERE d.id_ciudad = c.id_ciudad
        AND EXISTS (
            SELECT 'Z'
            FROM pais p
            WHERE p.id_pais=c.id_pais
            AND nombre_pais = 'ARGENTINA'))));
```

4

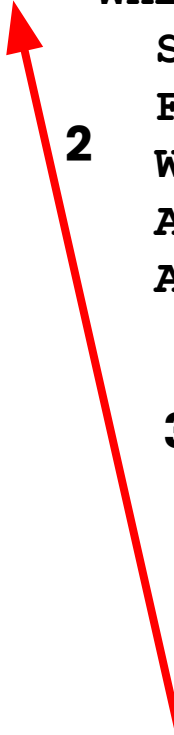
Ciudad

ciu	pais	...
...	...	...
15736	AR	..
...	...	...

Pais

pais	nombre
...	...
AR	...
...	...

# Uso del Operador EXISTS



```
1 SELECT e.nombre, e.apellido
   FROM empleado e
   WHERE EXISTS (
       SELECT 'X'
       FROM departamento d
       WHERE e.id_departamento=d.id_departamento
       AND e.id_distribuidor=d.id_distribuidor
       AND EXISTS (
           SELECT 'Y'
           FROM ciudad c
           WHERE d.id_ciudad = c.id_ciudad
           AND EXISTS (
               SELECT 'Z'
               FROM pais p
               WHERE p.id_pais=c.id_pais
               AND nombre_pais = 'ARGENTINA'))));
```

Como la subconsulta 4 encontró una tupla que cumple la restricción, devuelve 'Z' a la subconsulta 3. Entonces, la subconsulta 3 encontró un "Z" y devuelve 'Y' a la subconsulta 2. La subconsulta 2 encontró un "Y", entonces devuelve 'X' a la subconsulta 1. Finalmente la consulta 1 agrega la tupla a la tabla de resultados y continua el proceso con la siguiente tupla de la tabla Empleado.



# Uso del Operador EXISTS

nombre	apellido
Nicolas	Casta#O
Maria Belen	Schoeder
Hilda	Agostinelli
Marcelo	Bayle
Norma Susana	Dutto
Eduardo Oscar	Perez
Horacio A.	Arisnavarreta
Armando Ramon	Puricelli
...	....

La consulta retorna 227  
filas en 18,444 ms

# NOT IN vs. NOT EXISTS

## NOT IN

- Tener presente que la subconsulta de un NOT IN puede retornar valores nulos.

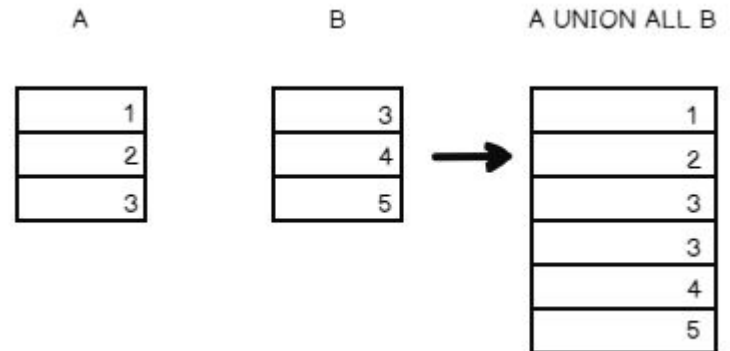
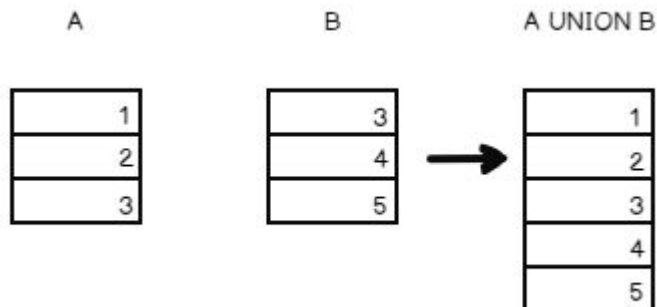
## NOT EXISTS

- La subconsulta no retornará ningún dato, devuelve valores VERDADEROS o FALSOS que dependen de la verificación de existencia de los valores de la subconsulta.
- Para hacerlo debe recorrer TODA la subconsulta

# Cláusula UNION

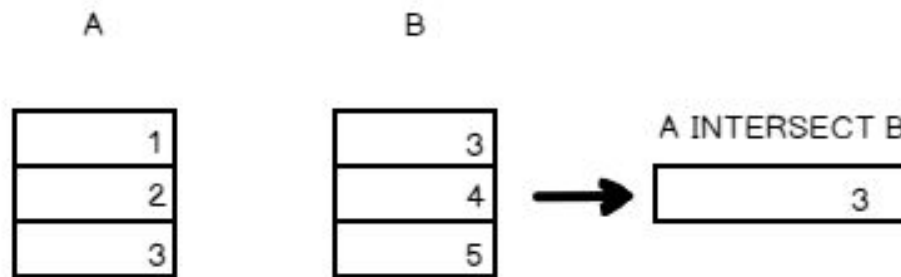
Unión combina los resultados de dos o más consultas en un único conjunto de resultados que incluye todas las filas que pertenecen a todas las consultas en la Unión.

Hay dos opciones para hacerlo UNION y UNION ALL



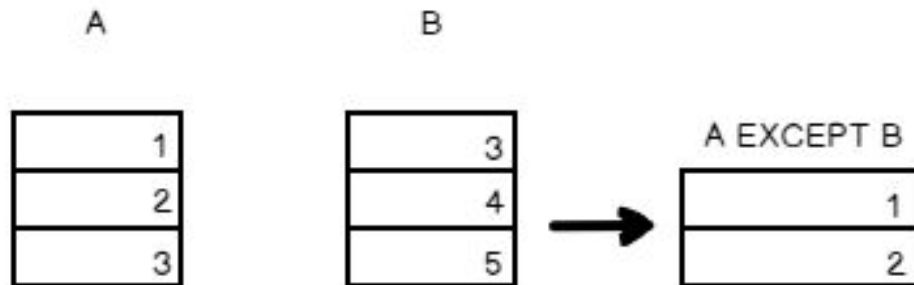
# Cláusula INTERSECT

Recupera las filas que son comunes a todas las tablas.



# Cláusula EXCEPT

Recupera las filas que están en la primer tabla y que no están en la segunda.



# LEFT JOIN vs. EXCEPT

## NOT IN vs. NOT EXISTS vs. LEFT JOIN vs. EXCEPT

### LEFT JOIN

Devuelve **todos** los registros de la primera tabla de la izquierda, los registros coincidentes de la segunda tabla de la derecha y los valores NULL del lado derecho para los registros de la tabla de la izquierda que no tienen coincidencias en la tabla de la derecha.

### EXCEPT

Devuelve todos los registros distintos de la primera instrucción SELECT que no se devuelven de la segunda instrucción SELECT; cada instrucción SELECT se considerará como un conjunto de datos separado.

# Ejemplo

Planteo de una misma consulta resuelta con NOT IN - NOT EXISTS - LEFT JOIN - EXCEPT

Listar todas las instituciones que no poseen voluntarios

Podría ser el siguiente query...

```
SELECT *  
FROM institucion  
WHERE id institucion NOT IN  
      (SELECT distinct id_institucion  
        FROM voluntario)
```

Porqué no retorna  
datos?





# Ejemplo

Listar todas las instituciones que no poseen voluntarios

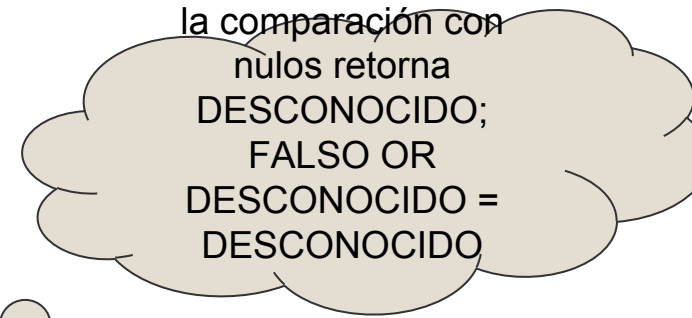
Porque el NOT IN es equivalente a colocar  
SELECT \*

FROM institucion

WHERE id\_institucion <> 10 AND  
id\_institucion <> 20 AND

.....

id\_institucion <> NULL



la comparación con  
nulos retorna  
DESCONOCIDO;  
FALSO OR  
DESCONOCIDO =  
DESCONOCIDO

# Ejemplo

Listar todas las instituciones que no poseen voluntarios,  
otra forma