Introducción a las Bases de Datos y Bases de Datos

CONSULTAS SQL – PARTE 2

TEORÍA 5

2017

Tecnicaturas TUPAR y TUDAI

CONSULTAS SQL

- ✓ Repaso: la sentencia del lenguaje empleada para la recuperación de datos: **SELECT**.

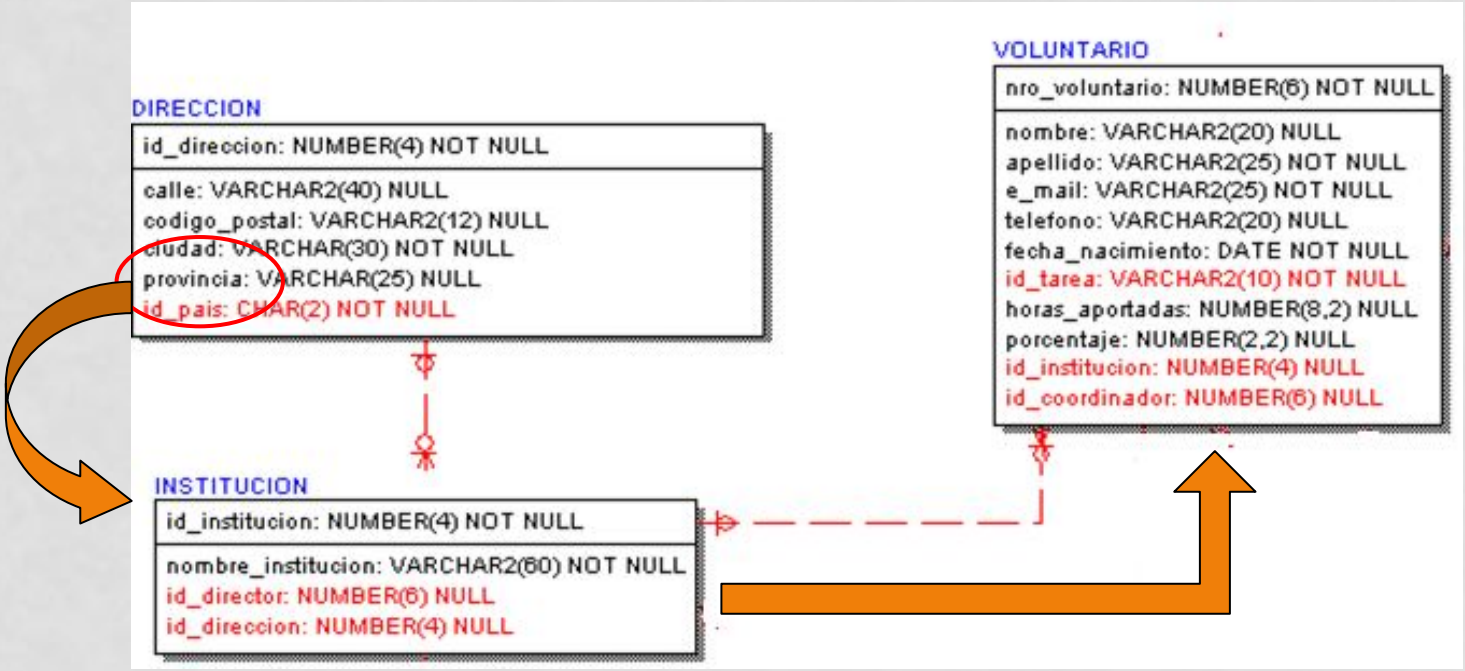
- ✓ Sintáxis:

SELECT * | { [**DISTINCT**] columna | expresion [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]]**

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 entre las distintas tablas.



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'
```

Condiciones de ensamble = (cantidad de tablas – 1)

CONSULTAS ANIDADAS

- 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

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

Solo se puede utilizar si la subconsulta devuelve UNA SOLA FILA

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

- **Muy importante…..(**que ocurriría si hay mas de un resultado para Texas?**)**
  - Usar **single-rows** operadores para subqueries que retornan una fila (=, >, <, <>, >=, <=)
  - Usar **multiple-rows** operadores para subqueries que retornan varias filas (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

SUBCONSULTAS DE UNA SOLA FILA

Ejemplo: Se desea seleccionar los voluntarios que realizan la misma tarea que el voluntario 141 y que aportan más horas que el voluntario 143

```
SELECT nombre, id_tarea, horas_aportadas
FROM   voluntario
WHERE id_tarea = (SELECT id_tarea
                  FROM   voluntario
                   WHERE id_voluntario = 141)
AND horas_aportadas > (SELECT horas_aportadas
                       FROM   voluntario
                        WHERE id_voluntario = 143);
```

CONSULTAS ANIDADAS

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 0 o + de 1 fila, dará error

CONSULTAS ANIDADAS

La cláusula HAVING en subconsultas

Se ejecuta en primer lugar las subconsultas.

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

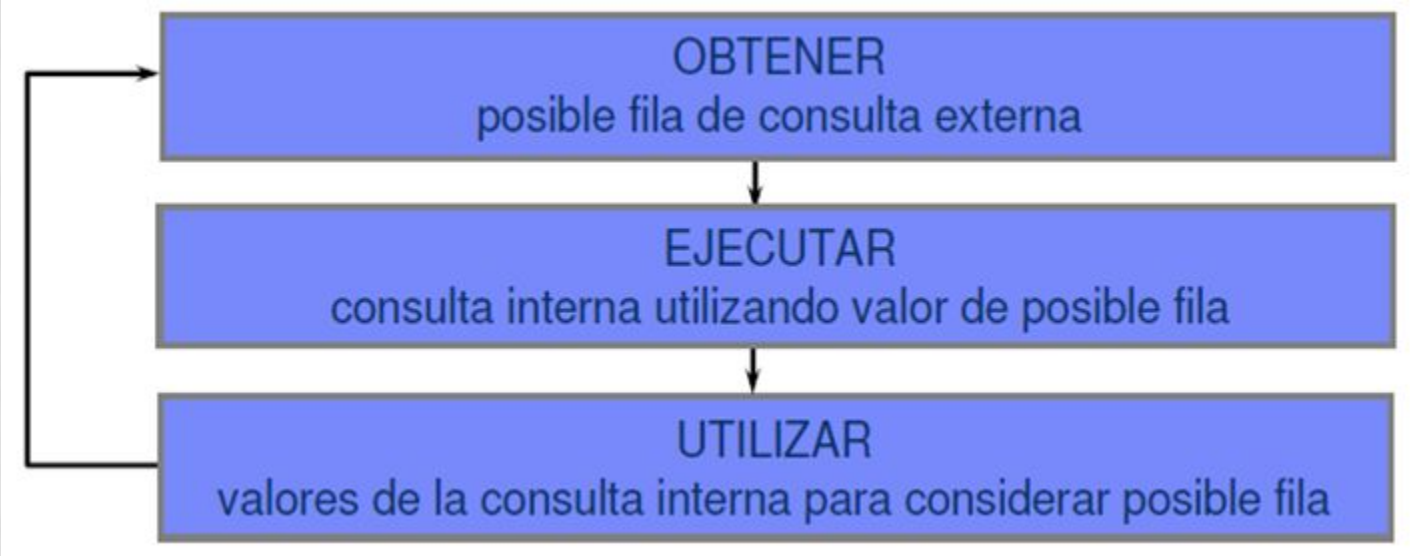
```
SELECT id_institucion, MIN(horas_aportadas)
FROM   voluntario
GROUP BY id_institucion
HAVING MIN(horas_aportadas) > (SELECT MIN(horas_aportadas)
                               FROM voluntario
                                    WHERE id_institucion = 40);
```

CONSULTAS ANIDADAS

Operador	Significado
IN	Igual a cualquier miembro de la lista
ANY	Requiere que la expresión se satisfaga para al menos un valor de la subconsulta
ALL	Requiere que la expresión se satisfaga para cada valor de la subconsulta

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: Busque todos los voluntarios que aportan mas horas que el promedio de horas de los voluntarios de la misma institucion.

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

APELLIDO	HORAS_APORTADAS	ID_INSTITUCION
King	24000	90
Hunold	9000	60
Ernst	6000	60
Greenberg	12000	100
Faviet	9000	100
Raphaely	11000	30
Weiss	8000	50
Fripp	8200	50
Kaufling	7900	50
Vollman	6500	50
Mourgos	5800	50
Ladwig	3800	50
Rajs	3500	50
Russell	14000	80

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

USO DEL OPERADOR EXISTS

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

USO DEL OPERADOR EXISTS

Ejemplo: Buscar los voluntarios que coordinan al menos a una persona.

```
SELECT nro_voluntario, apellido, id_tarea
FROM   voluntario externa
WHERE  EXISTS ( SELECT *
                FROM  voluntario
                WHERE  id_coordinador =
                   externa.nro_voluntario);
```

NRO_VOLUNTARIO	APELLIDO	ID_TAREA
100	King	AD_PRES
101	Kochhar	AD_VP
102	De Haan	AD_VP
103	Hunold	IT_PROG
108	Greenberg	FI_MGR
114	Raphaely	PU_MAN
120	Weiss	ST_MAN
121	Fripp	ST_MAN
122	Kaufling	ST_MAN
123	Vollman	ST_MAN
124	Mourgos	ST_MAN
145	Russell	SA_MAN
146	Partners	SA_MAN
147	Errazuriz	SA_MAN
NRO_VOLUNTARIO	APELLIDO	ID_TAREA
148	Cambrault	SA_MAN
149	Zlotkey	SA_MAN
201	Hartstein	MK_MAN
205	Higgins	AC_MGR

DIVISIÓN EN SQL

Obtener los Investigadores que trabajan en todos los Proyectos

```
SELECT * FROM INVESTIGADOR I
WHERE NOT EXISTS
    ( SELECT * FROM PROYECTO P
      WHERE NOT EXISTS
          ( SELECT * FROM TRABAJA T
            WHERE T.id_inv = I.id_inv
              AND P.proy_id = T.proy_id ) );
```

INVESTIGADOR

id_inv	nom_inv	categ
1	Inv1	2
2	Inv2	3
3	Inv3	2
4	Inv4	3

PROYECTO

proy_id	nombre	tema	duracion	id_inst
1	Proy1	Análisis de Imágenes	24	2
2	Proy2	Bases de Datos	24	1
3	Proy3	Análisis de Sistemas	12	2
4	Proy4	Bases de Datos	12	2
5	Proy5	Análisis Bacteriológico	24	3
6	Proy6	Análisis de Imágenes	24	2

TRABAJA

id_inv	proy_id
1	3
1	5
2	5
3	2

ENSAMBLES INTERNOS

- Por medio del operador JOIN podemos combinar dos tablas según una condición para obtener tuplas compuestas por atributos de las dos relaciones combinadas. Existen diferentes maneras hacerlo:
- **INNER JOIN** (De equivalencia o Equi-join)**:** Es un caso particular de INNER JOIN en el que 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, solo 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.

ENSAMBLES INTERNOS

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

```
SELECT nombre, apellido

FROM voluntario v INNER JOIN institucion i
        ON (v.id_institucion = i.id_institucion)

        NATURAL JOIN direccion d

  AND d.provincia = 'Texas'
```

ENSAMBLE EXTERNO

El ensamble interno (*inner join*) sólo se queda con las filas que tienen valores idénticos en las columnas de las tablas que compara.

Pero….puede suceder que perdamos alguna fila interesante de alguna de las dos tablas; por ejemplo, porque poseen valores NULL…..

Por esto se SQL dispone del ensamble externo (*outer join*), que nos permite obtener todos los valores de la tabla que hemos puesto a la derecha, los de la tabla que hemos puesto a la izquierda o los valores de ambas tablas.

Su formato es:

```
SELECT nombre_columnas_a_seleccionar
FROM t1 [NATURAL] [LEFT|RIGHT|FULL] [OUTER] JOIN t2
{ON condiciones| [USING (columna [,columna...])}
[WHERE condiciones];
```

ENSAMBLE EXTERNO

Ensamble externo: Operador RIGHT JOIN

Listar las entregas de peliculas y todos los videos.

```
SELECT *
FROM entrega RIGHT JOIN video;
```

Ensamble externo: Operador LEFT JOIN

Listar todas las empresas productoras junto con las películas que producen:

```
SELECT *
FROM empresa_productora LEFT JOIN pelicula;
```

ENSAMBLE EXTERNO

Ensamble externo: Operador FULL JOIN

Listar todos los distribuidores nacionales e internacionales

```
SELECT *
FROM nacional N FULL JOIN internacional I
        ON (N.id_distrib_mayorista = I.id_distribuidor) ;
```

Listara todos los distribuidores nacionales tengan o no distribuidor mayorista y tambien se incluiran los distribuidores internacionales que no estaban asociados a ningun distribuidor nacional.

Nota: la cláusula ON la debo especificar cuando las columnas de ensamble (join) no coinsiden en nombre

CLÁUSULA UNION

Permite realizar la intersección de los resultados producidos por 2 o mas sentencias SELECT FROM. El formato es:

```
SELECT columnas
FROM tabla
[WHERE condiciones]
INTERSECT [ALL]
SELECT columnas
FROM tabla
[WHERE condiciones];
```

Mismas consideraciones que para la UNIÓN

CLÁUSULA INTERSECT

Permite realizar la intersección de los resultados producidos por 2 o mas sentencias SELECT FROM. El formato es:

```
SELECT columnas
FROM tabla
[WHERE condiciones]
INTERSECT [ALL]
SELECT columnas
FROM tabla
[WHERE condiciones];
```

Mismas consideraciones que para la UNIÓN

INTERSECCIÓN UTILIZANDO IN O EXISTS

Es una de las operaciones del SQL que se puede hacer de otra forma. Podríamos encontrar la intersección con IN o EXISTS:

```
SELECT c.ciudad
FROM clientes c
WHERE c.ciudad IN (SELECT d.ciudad_dep FROM departamentos d);

SELECT c.ciudad
FROM clientes c
WHERE EXISTS (SELECT *
        FROM departamentos d WHERE c.ciudad = d.ciudad_dep);
```

CLÁUSULA EXCEPT

- Para encontrar la diferencia entre dos o más sentencias SELECT FROM podemos utilizar la cláusula EXCEPT, que tiene este formato:

```
SELECT columnas
FROM tabla
[WHERE condiciones]
EXCEPT [ALL]
SELECT columnas
FROM tabla
[WHERE condiciones];
```

- Mismas consideraciones que para la UNIÓN
- Tambien se puede realizar la diferencia utilizando las cláusulas NOT IN o NOT EXISTS