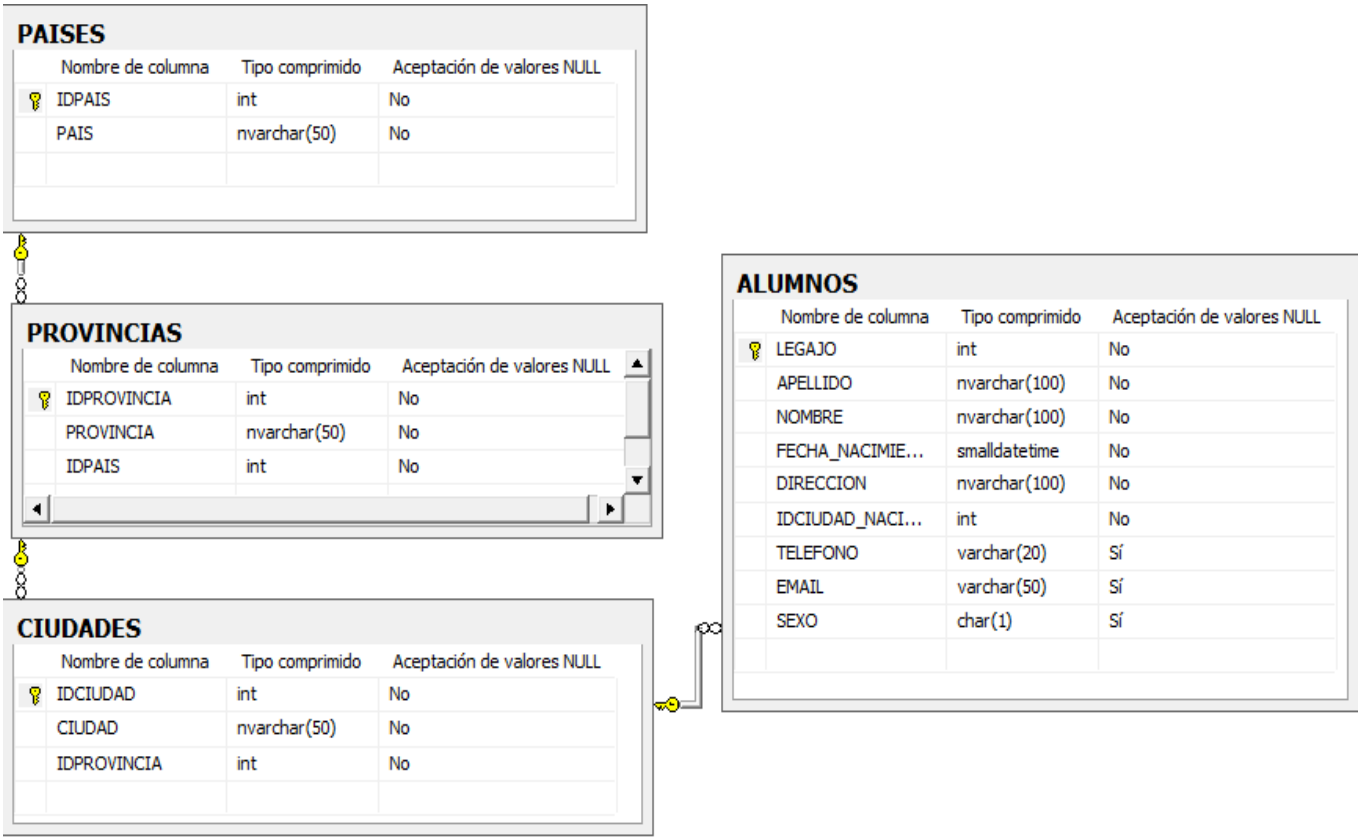
Carrera: Técnico Superior en Programación
Materia: Laboratorio de Computación III
Tema: Consultas de selección - Parte 2

Consultas de selección - Parte 2

Cláusulas JOIN

Hasta el momento, sólo estuvimos obteniendo datos de nuestra base de datos provenientes de una sola tabla. Sin embargo, es muy común querer obtener datos de más de una tabla y así poder elaborar un listado más descriptivo. Por ejemplo, es usual querer obtener la descripción principal de cada campo en lugar de su clave principal.

Utilizaremos la misma base de datos que en el apunte 'Consultas de selección - Parte 1' para ejemplificar el objetivo y los tipos de JOIN con los que trabajaremos.



Antes de comenzar a utilizar las cláusulas del tipo JOIN veamos qué ocurre cuando no las utilizamos y queremos obtener datos de diferentes tablas. Supongamos que queremos obtener Apellido, nombre y nombre de la ciudad de nacimiento de cada alumno. En el diagrama que aparece arriba podemos observar claramente como el apellido y nombre se pueden obtener de la tabla alumnos pero el nombre de la ciudad proviene de la tabla ciudades.

En principio veamos las siguientes consultas para analizar cómo sería el resultado óptimo:

```
SELECT * FROM ALUMNOS
```

	LEGAJO	APELLIDO	NOMBRE	FECHA_NACIMIENTO	DIRECCION	IDCIUDAD_NACIMIENTO	TELEFONO	EMAIL	SEXO
1	1000	PEREZ	JUAN	1990-05-05 00:00:00	ITALIA 342	1	NULL	NULL	M
2	1100	MENDEZ	JULIETA	1984-05-10 00:00:00	LIBERTAD 123	2	NULL	NULL	F
3	1300	FERNANDEZ	NATALIA	1984-09-11 00:00:00	JUNIN 12	1	NULL	FERNATALIA@FAKEMAIL.COM	F
4	2000	GUTIERREZ	CARLOS	1991-03-02 00:00:00	BELGRANO 890	3	1234567	NULL	M
5	3000	SIMON	ANGEL	1986-10-02 00:00:00	ASD	NULL	NULL	NULL	M

Como podemos observar en el diagrama de la página 1, existe integridad referencial entre la tabla alumnos y la tabla ciudades. Sin embargo, en el quinto registro el campo IDCIUDAD_NACIMIENTO acepta valores nulos. Esto es correcto y significa que el campo puede contener el valor nulo pero si contiene algún otro tipo de valor, este debe existir en la tabla ciudades.

```
SELECT * FROM CIUDADES
```

	IDCIUDAD	CIUDAD	IDPROVINCIA
1	1	SAN ISIDRO	1
2	2	VICENTE LOPEZ	1
3	3	CÓRDOBA	2
4	4	LUJÁN	1

Recordemos que lo que necesitaríamos es un listado de la misma cantidad de registros que existan en la tabla de alumnos, pero que en lugar de mostrarnos el ID de ciudad de nacimiento nos muestre el nombre de la ciudad.

```
SELECT ALUMNOS.apellido, ALUMNOS.nombre, CIUDADES.ciudad FROM ALUMNOS, CIUDADES
```

	apellido	nombre	ciudad
1	PEREZ	JUAN	SAN ISIDRO
2	MENDEZ	JULIETA	SAN ISIDRO
3	FERNANDEZ	NATALIA	SAN ISIDRO
4	GUTIERREZ	CARLOS	SAN ISIDRO
5	SIMON	ANGEL	SAN ISIDRO
6	PEREZ	JUAN	VICENTE LOPEZ
7	MENDEZ	JULIETA	VICENTE LOPEZ
8	FERNANDEZ	NATALIA	VICENTE LOPEZ
9	GUTIERREZ	CARLOS	VICENTE LOPEZ
10	SIMON	ANGEL	VICENTE LOPEZ
11	PEREZ	JUAN	CÓRDOBA
12	MENDEZ	JULIETA	CÓRDOBA
13	FERNANDEZ	NATALIA	CÓRDOBA
14	GUTIERREZ	CARLOS	CÓRDOBA
15	SIMON	ANGEL	CÓRDOBA
16	PEREZ	JUAN	LUJÁN
17	MENDEZ	JULIETA	LUJÁN
18	FERNANDEZ	NATALIA	LUJÁN
19	GUTIERREZ	CARLOS	LUJÁN
20	SIMON	ANGEL	LUJÁN

Al ejecutar la consulta obtenemos los datos de más de una tabla y por eso es que necesitamos indicar de qué tabla obtenemos cada columna. Pero, en lugar de obtener los datos de los alumnos incluidos los nombres de las ciudades donde nacieron, obtenemos el producto de los

registros de ambas tablas. Esto quiere decir que para cada registro de la tabla alumnos obtenemos su combinación con todos los registros de la tabla ciudades. Por lo tanto, el total de registros que obtendremos será de cantidad de filas de alumnos multiplicado por la cantidad de filas de ciudades.

El problema resulta por el uso de más de una tabla dentro del FROM, precisamente porque no especificamos que queremos todos los registros de alumnos pero sólo un registro de ciudad que se encuentre relacionado a cada registro de alumno en particular.
Para lograr esto sería necesario indicarlo mediante una condición dentro del WHERE. Para ello nos basamos en la relación que existe entre la clave foránea de la tabla ALUMNOS mediante la columna IDCIUDAD_NACIMIENTO y la clave primaria de la tabla CIUDADES, la cual es IDCIUDAD.

La consulta quedaría así:

```
SELECT A.apellido, A.nombre, C.ciudad FROM ALUMNOS AS A, CIUDADES AS C WHERE
A.idciudad_nacimiento = C.idciudad
```

	apellido	nombre	ciudad
1	PEREZ	JUAN	SAN ISIDRO
2	MENDEZ	JULIETA	VICENTE LOPEZ
3	FERNANDEZ	NATALIA	SAN ISIDRO
4	GUTIERREZ	CARLOS	CÓRDOBA

Aquí podemos observar varias cuestiones. La primera es la posibilidad de crear un alias también para las tablas. De manera que la tabla ALUMNOS será representada por el alias A mientras que la tabla CIUDADES será representada por el alias C. Luego, que a diferencia del ejemplo anterior (con el cual obteníamos resultados indeseados) aquí obtenemos los registros tal y como los necesitamos.

Esto se debe a que a cada fila se le aplica la condición del WHERE. La cual indica que la columna IDCIUDAD_NACIMIENTO de la tabla ALUMNOS debe contener el mismo valor que en la columna IDCIUDAD de la tabla CIUDADES.

INNER JOIN

Como vimos en el ejemplo anterior, logramos obtener los resultados esperados pero tuvimos que utilizar una cláusula WHERE obligatoriamente. Si bien su funcionamiento es correcto, se recomienda utilizar la cláusula INNER JOIN cuyo objetivo es específicamente el de relacionar dos tablas mediante un campo que sirve de nexo. La recomendación proviene de la mayor eficiencia del INNER frente al WHERE.

Ahora veamos cómo quedaría el ejemplo anterior utilizando la cláusula INNER JOIN:

```
SELECT A.apellido, A.nombre, C.ciudad FROM ALUMNOS AS A
INNER JOIN CIUDADES AS C
ON A.idciudad_nacimiento = C.idciudad
```

A diferencia de la consulta anterior que teníamos más de una tabla en la cláusula FROM, aquí sólo tenemos una tabla (ALUMNOS). Luego indicamos que dicha tabla tendrá una relación con la tabla CIUDADES mediante la cláusula INNER JOIN y su nexo (ON) serán los campos

idciudad_nacimiento para la tabla alumnos y el campo idciudad para la tabla ciudades.

De ésta forma, el resultado de esta consulta será el siguiente:

	apellido	nombre	ciudad
1	PEREZ	JUAN	SAN ISIDRO
2	MENDEZ	JULIETA	VICENTE LOPEZ
3	FERNANDEZ	NATALIA	SAN ISIDRO
4	GUTIERREZ	CARLOS	CÓRDOBA

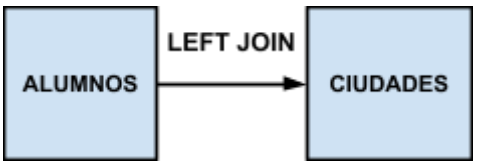
La pregunta es: ¿Qué paso con el registro nro 5 de la tabla alumnos? Aquel que tenía como idciudad_nacimiento el valor NULL.
No aparece ya que es la manera en que funciona la cláusula INNER JOIN. Va a exigir que el valor de los campos que relacionemos existan en ambas tablas. Obtendrá todos los alumnos que se puedan relacionar con alguna ciudad de lo contrario lo excluirá. Por lo que si tuviese un valor de idciudad_nacimiento que no existiera en la tabla ciudades o bien éste tuviese el valor NULL (como en el caso del ejemplo) entonces dicho registro no es incorporado al listado de selección.

¿No es posible obtener los datos si el campo no tiene un valor relacionado con las tablas de la selección?

Sí, es posible y para ello se utilizan las cláusulas LEFT JOIN y RIGHT JOIN.

LEFT JOIN

La cláusula LEFT JOIN al igual que INNER JOIN es utilizada para indicar la relación existente entre dos tablas mediante dos columnas que sirven como nexo. Sin embargo, LEFT JOIN no exige que exista el valor de la columna de la tabla de la izquierda en el de la tabla de la derecha para poder obtener los datos.
Supongamos este diagrama:



Suponiendo que queremos obtener Apellido, nombre y nombre de la ciudad de nacimiento. Mediante el uso de LEFT JOIN obtendremos todos los registros de ALUMNOS (Apellido y nombre), si existe el código de ciudad en la tabla de ciudades obtendrá el nombre de la ciudad relacionada. Pero si no existe el código de ciudad o el campo contiene NULL completará el nombre de la ciudad con el valor NULL incorporando el registro al listado.

Veamos el código que lo ejemplifica:

```
SELECT A.apellido, A.nombre, C.ciudad FROM ALUMNOS AS A
LEFT JOIN CIUDADES AS C
ON A.idciudad_nacimiento = C.idciudad
```

Left Join: Si tiene valor NULL en la tabla izquierda (ciudad) la incluye igual

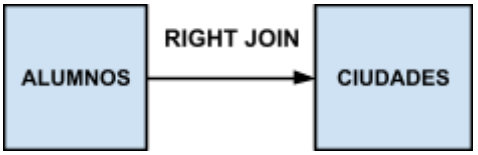
	apellido	nombre	ciudad
1	PEREZ	JUAN	SAN ISIDRO
2	MENDEZ	JULIETA	VICENTE LOPEZ
3	FERNANDEZ	NATALIA	SAN ISIDRO
4	GUTIERREZ	CARLOS	CÓRDOBA
5	SIMON	ANGEL	NULL

En este caso, el registro nro 5 de la tabla de alumnos, aquel que no tenía un idciudad_nacimiento relacionado con la tabla de ciudades, es incluido en el listado. Pero como mencionamos anteriormente el valor a mostrar en lo que representaría el nombre de la ciudad es completado con el valor NULL.

RIGHT JOIN

La cláusula RIGHT JOIN al igual que el LEFT JOIN e INNER JOIN es utilizada para relacionar dos tablas mediante dos columnas utilizadas como nexo. RIGHT JOIN no exige que exista el valor de la tabla de la columna de la derecha en el de la tabla de la izquierda para poder obtener los datos.

Según el siguiente diagrama:



Suponiendo que queremos obtener Apellido, nombre y nombre de la ciudad de nacimiento. Mediante el uso de RIGHT JOIN obtendremos los registros de ALUMNOS (Apellido y nombre) y nombre de la ciudad si existe el código de ciudad en la tabla de ciudades. Además obtendrá los registros de ciudades que no se encuentren relacionados con algún alumno completando con valor NULL en los campos Apellido y nombre.

Código de ejemplo:

```
SELECT A.apellido, A.nombre, C.ciudad FROM ALUMNOS AS A
RIGHT JOIN CIUDADES AS C
ON A.idciudad_nacimiento = C.idciudad
```

	apellido	nombre	ciudad
1	PEREZ	JUAN	SAN ISIDRO
2	FERNANDEZ	NATALIA	SAN ISIDRO
3	MENDEZ	JULIETA	VICENTE LOPEZ
4	GUTIERREZ	CARLOS	CÓRDOBA
5	NULL	NULL	LUJÁN

Right Join: Si tiene valor NULL en la tabla derecha (Aumnos) la incluye igual

En este caso, como se puede observar, se realiza la relación entre el código de ciudad de la tabla alumnos con el código de ciudad de la tabla ciudades. Pero cada registro de ciudad que no se relaciona con algún alumno es incorporado al listado, agregando NULL en los campos de apellido y nombre.

FULL JOIN

Por último, resta ver el uso de FULL JOIN que como su nombre lo indica es el listado completo comprendido por la combinación de LEFT JOIN y RIGHT JOIN.

Ejemplo:

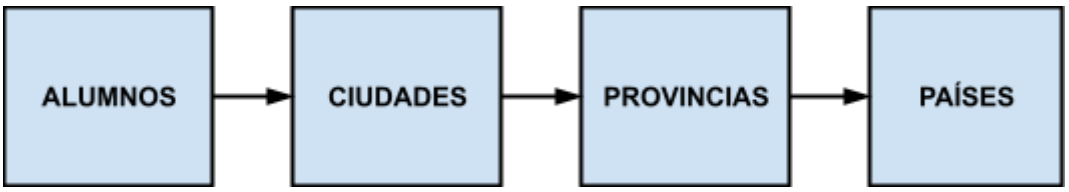
```
SELECT A.apellido, A.nombre, C.ciudad FROM ALUMNOS AS A
FULL JOIN CIUDADES AS C
ON A.idciudad_nacimiento = C.idciudad
```

	apellido	nombre	ciudad
1	PEREZ	JUAN	SAN ISIDRO
2	MENDEZ	JULIETA	VICENTE LOPEZ
3	FERNANDEZ	NATALIA	SAN ISIDRO
4	GUTIERREZ	CARLOS	CÓRDOBA
5	SIMON	ANGEL	NULL
6	NULL	NULL	LUJÁN

Full Join: Acepta NULL en ambas tablas

Anidamiento de JOINS

Es probable que uno haya normalizado lo suficiente una base de datos de manera tal que para obtener algún valor haya que recorrer diferentes niveles de relaciones entre más de dos tablas. Esto es posible hacerlo combinando los JOINS correctamente. Supongamos que queremos obtener Nombre, apellido y país de nacimiento de cada alumno. Para ello tenemos que tener en cuenta que en nuestras tablas existe el siguiente diagrama para la relación:



O sea que para poder obtener nuestro valor de 'Nombre de país de nacimiento' la transitividad de las tablas nos obliga a recorrer ciudad de nacimiento, provincia y luego país. Claro está, que no necesariamente hay que mostrar los valores de ciudad y provincia. Hay que tener en cuenta que el listado solicitado sólo necesita mostrar los valores Apellido, nombre y país de nacimiento.

Ejemplo:

```
SELECT A.apellido, A.nombre, PA.pais FROM ALUMNOS AS A
INNER JOIN CIUDADES AS C
ON C.idciudad = A.IDCIUDAD_NACIMIENTO

INNER JOIN PROVINCIAS AS PR
ON PR.IDPROVINCIA = C.IDPROVINCIA

INNER JOIN PAISES AS PA
ON PA.IDPAIS = PR.IDPAIS
```

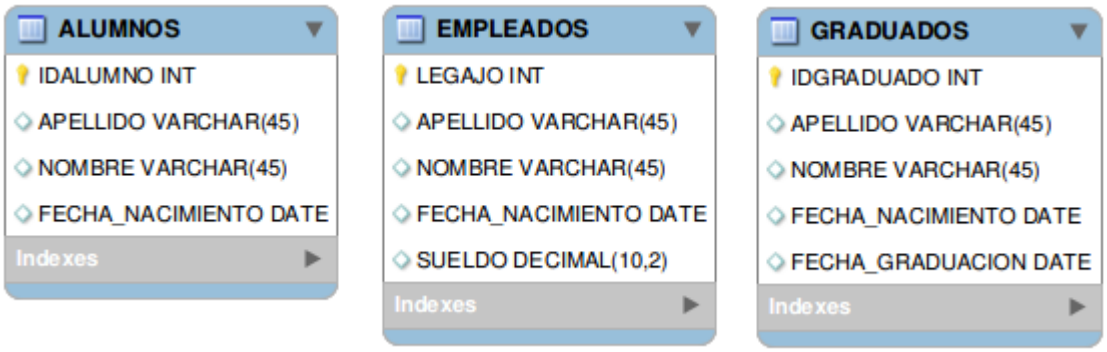
	apellido	nombre	pais
1	PEREZ	JUAN	ARGENTINA
2	MENDEZ	JULIETA	ARGENTINA
3	FERNANDEZ	NATALIA	ARGENTINA
4	GUTIERREZ	CARLOS	ARGENTINA

El ejemplo anterior ejemplifica que se pueden combinar las cláusulas JOIN para poder obtener los resultados deseados y que no hace falta que las columnas estén especificadas dentro del SELECT (como se hace con apellido, nombre y nombre de país) para poder utilizarlas dentro de las cláusulas de JOIN, WHERE u ORDER BY.

Cláusula UNION

Combina los resultados de dos consultas de SELECT, todos los resultados de una consulta se unen a los resultados de la otra. No debe de confundirse la operación de UNION con la cláusula JOIN ya que esta última se encarga de combinar columnas de dos tablas.

Supongamos el siguiente caso:



Aquí tenemos tres tablas que podrían representar las personas que trabajan, estudian y estudiaron en una universidad. La idea sería poder tener un listado de las personas que pertenecen o pertenecieron a la facultad.
La primer alternativa que se nos puede ocurrir es ejecutar una consulta como la siguiente:

```
SELECT * FROM ALUMNOS
SELECT * FROM GRADUADOS
SELECT * FROM EMPLEADOS
```

Resultados | Mensajes

	IDALUMNO	APELLIDO	NOMBRE	FECHA_NACIMIENTO
1	1	SIMON	ANGEL	1986-10-02
2	2	LOPEZ	JULIETA	1989-01-01

	IDGRADUADO	APELLIDO	NOMBRE	FECHA_NACIMIENTO	FECHA_GRADUACION
1	1	PEREZ	ROBERTO	1976-08-04	2000-05-04

	LEGAJO	APELLIDO	NOMBRE	FECHA_NACIMIENTO	SUELDO
1	1000	KLOSTER	DANIEL	1960-05-04	9999999.00
2	4000	PEREZ	ROBERTO	1976-08-04	50000.00

Aquí como podemos observar estamos realizando tres consultas de SELECT a la vez, pero los resultados obtenidos no son los deseados ya que, como se puede apreciar, los listados que obtenemos son independientes entre sí. Es decir, obtenemos tres listados distintos.
La solución a esto es la cláusula UNION. Pero antes de comenzar a desarrollarla tenemos que comprender que si vamos a unir filas éstas deben compartir las mismas columnas, de lo contrario carecería de sentido.
En principio tenemos que determinar qué estructura de listado vamos a querer. En principio definir qué columnas van a formar parte del listado y luego qué filtros vamos a querer aplicar al mismo.
En este ejemplo vamos a determinar que queremos obtener:
A) el código identificatorio, apellido, nombre, fecha de nacimiento, sueldo y qué tipo de rol cumple en la institución de todos los alumnos.
B) Un listado similar con apellido, nombre y fecha de nacimiento pero de quienes hayan nacido entre 1976 y 1986.

Como podemos observar, la mayoría de los campos los obtenemos de sus respectivas tablas a excepción del sueldo para los graduados y alumnos. Así como el rol que no figura en ninguna tabla. No obstante, como vimos en los primeros apuntes de selección, nosotros podemos obtener valores constantes en nuestra consulta de SELECT que en este caso servirían de 'relleno' para las columnas faltantes.

Veamos como quedaría la consulta A:

```
SELECT IDALUMNO AS ID, APELLIDO, NOMBRE, FECHA_NACIMIENTO, 0 AS SUELDO,
'ALUMNO' AS ROL FROM ALUMNOS
UNION
SELECT IDGRADUADO AS ID, APELLIDO, NOMBRE, FECHA_NACIMIENTO, 0 AS SUELDO,
'GRADUADO' AS ROL FROM GRADUADOS
UNION
SELECT LEGAJO AS ID, APELLIDO, NOMBRE, FECHA_NACIMIENTO, SUELDO, 'EMPLEADO' AS
ROL FROM EMPLEADOS
```

	ID	APELLIDO	NOMBRE	FECHA_NACIMIENTO	SUELDO	ROL
1	1	PEREZ	ROBERTO	1976-08-04	0.00	GRADUADO
2	1	SIMON	ANGEL	1986-10-02	0.00	ALUMNO
3	2	LOPEZ	JULIETA	1989-01-01	0.00	ALUMNO
4	1000	KLOSTER	DANIEL	1960-05-04	9999999.00	EMPLEADO
5	4000	PEREZ	ROBERTO	1976-08-04	50000.00	EMPLEADO

Como podemos observar, obtener en un mismo listado los cinco registros que antes teníamos en tres listados. La claúsula UNION se encargó de combinarlos ya que todos los números de columnas coincidían. Para darle homogeneidad a los listados se completó con valores constantes aquellos campos en los que faltaban datos.

Veamos el caso de la consulta B:

```
SELECT APELLIDO, NOMBRE, FECHA_NACIMIENTO FROM ALUMNOS
WHERE YEAR(FECHA_NACIMIENTO) BETWEEN 1976 AND 1986
UNION
SELECT APELLIDO, NOMBRE, FECHA_NACIMIENTO FROM GRADUADOS
WHERE YEAR(FECHA_NACIMIENTO) BETWEEN 1976 AND 1986
UNION
SELECT APELLIDO, NOMBRE, FECHA_NACIMIENTO FROM EMPLEADOS
WHERE YEAR(FECHA_NACIMIENTO) BETWEEN 1976 AND 1986
```

	APELLIDO	NOMBRE	FECHA_NACIMIENTO
1	PEREZ	ROBERTO	1976-08-04
2	SIMON	ANGEL	1986-10-02

Este ejemplo es un caso muy particular, primero hay que mencionar que si queremos que los listados sean filtrados a partir de aquellas personas que nacieron entre 1976 y 1986 es necesario agregar la cláusula WHERE en cada consulta de SELECT.
Luego, podemos observar como tanto en la tabla de graduados como en la de empleados tenemos un registro (Perez, Roberto) que tiene exactamente la misma fecha de nacimiento (probablemente porque se trate de la misma persona). Sin embargo, cuando obtenemos los

registros mediante UNION, sólo obtenemos uno de ellos. Esto se debe a que UNION realiza el 'DISTINCT' por defecto, por lo que si encuentra un registro duplicado lo elimina del listado.

Si quisiéramos que nuestra consulta admita los registros duplicados utilizamos la cláusula UNION ALL

```
SELECT APELLIDO, NOMBRE, FECHA_NACIMIENTO FROM ALUMNOS
WHERE YEAR(FECHA_NACIMIENTO) BETWEEN 1976 AND 1986
UNION ALL
SELECT APELLIDO, NOMBRE, FECHA_NACIMIENTO FROM GRADUADOS
WHERE YEAR(FECHA_NACIMIENTO) BETWEEN 1976 AND 1986
UNION ALL
SELECT APELLIDO, NOMBRE, FECHA_NACIMIENTO FROM EMPLEADOS
WHERE YEAR(FECHA_NACIMIENTO) BETWEEN 1976 AND 1986
```

	APELLIDO	NOMBRE	FECHA_NACIMIENTO
1	SIMON	ANGEL	1986-10-02
2	PEREZ	ROBERTO	1976-08-04
3	PEREZ	ROBERTO	1976-08-04