Subconsultas correlacionadas

Las subconsultas que hemos estudiado hasta ahora son independientes de la consulta más externa que las usa. Por eso entendemos que estas subconsultas podrían existir como consultas independientes.
En la base de datos de trabajadores, si nos piden "cuáles son los oficios de los trabajadores asignados al edificio 435" escribiríamos,

```
Select oficio
from trabajador
where id_trabajador in
        (select id_trabajador
        from asignacion where id_edificio=435)
```

La subconsulta "select id_trabajador ..... 435" se puede ejecutar independientemente y generaría el siguiente conjunto de Ids de trabajadores:

ID_TRABAJADOR
2920
1412
1311

de forma que la consulta principal equivale a:

```
Select oficio
from trabajador
where id_trabajador in ( 2920, 1412, 1311)
```

Ahora vamos a ver una clase de subconsultas cuyos valores en ejecución dependen de la fila que está siendo examinada por la consulta principal. Las llamaremos subconsultas correlacionadas.

Si en la base de datos de trabajadores nos piden "indicar los trabajadores que reciben una tarifa por hora mayor que la de su supervisor" sería imposible realizarlo con una subconsulta independiente.

La palabra clave es "SU" supervisor. Esto es, la fila del supervisor en la que hay que examinar la tarifa depende directamente de la fila del trabajador que está siendo examinada. Esta consulta puede resolverse usando una subconsulta correlacionada

```
Select nomb_trabajador
from trabajador as A
where A.tarifa_hr >
        (select B.tarifa_hr
        from trabajador as B
        where B.id_trabajador=A.id_supv)
```

Los pasos involucrados en la ejecución de esta consulta son los siguientes:

- El sistema hace dos copias de la tabla TRABAJADOR, A y B. A representará a los trabajadores y B a los supervisores.
- El sistema examina cada fila de A. Se selecciona una fila si satisface la condición WHERE. Esta condición establece que cada fila será selecciona si su TARIFA_HR es mayor que la TARIFA_HR generada por la subconsulta
- La subconsulta selecciona la TARIFA_HR de las filas de cuyo ID_TRABAJADOR sea igual al del ID_SUPV de la fila A que está siendo examinada por la consulta principal. Esta es la TARIFA_HR del supervisor de la fila de A

EXISTS Y NOT EXISTS (A Access no le gustan demasiado)

Supongamos que se quiere seleccionar a todos los trabajadores que no están asignados al edificio 435. Proponer una solución como

```
Select id_trabajador
from asignacion
where id_edificio <> 435
```

sería INCORRECTO, ya que esta consulta devuelve simplemente los Ids de los trabajadores que están trabajando en otros edificios que no son el 435 (aunque también trabajen o hayan trabajado en el 435)

Una solución podría ser:

```
Select id_trabajador
from asignacion
where id_edificio <>435
and id_trabajador not in    (select id_trabajador
                             from asignacion
                             where id_edificio=435)
```

Otra solución correcta sería utilizar el operador NOT EXISTS

```
Select id_trabajador
from trabajador
where NOT EXISTS
(select *
from asignacion
where asignacion.id_trabajador=trabajador.id_trabajador and id_edificio=435)
```

Los operadores EXISTS y NOT EXISTS siempre preceden a una subconsulta.
EXISTS evalúa verdadero si el subconjunto resultante de la subconsulta no es vacío mientras que NOT EXISTS evalúa verdadero si el conjunto resultante es vacío.
En este ejemplo, la subconsulta selecciona todas aquellas filas en ASIGNACION que tienen el mismo ID_TRABAJADOR que la fila de trabajador que está siendo

examinada por la consulta principal y que además tienen ID_EDIFICIO igual a 435. Si este subconjunto es vacío, entonces se selecciona la fila trabajador que se está examinando en la consulta principal, puesto que esto significa que el trabajador en cuestión no trabaja en el edificio 435

La solución con IN parecía más simple o al menos más fácil de comprender, por lo que algunos ya os estareis frotando las manos y pensando en olvidaros del EXISTS. Sin embargo y desgraciadamente, NOT EXISTS ofrece la única forma disponible de resolver ciertas consultas, en concreto, las que contengan el cuantificador "TODOS" en su condición.
Ejemplo: "Indicar los trabajadores que están asignados a todos los edificios" o lo que es lo mismo "Indicar los trabajadores tales que NO hay un edificio al cual NO estén asignados"

Para aclarar esta última solución, primero daremos una solución al problema "Identificar los edificios a lo que un trabajador hipotético "1235" no ha sido asignado

```
Select id_edificio
from edificio
where not EXISTS
(select *
from asignacion
where asignacion.id_edificio=edificio.id_edificio and
asignacion.id_trabajador=1235)
```

Si no hay ningún edificio que satisfaga esta consulta, entonces el trabajador 1235 satisface la consulta original. Para obtener una solución a la consulta original, el siguiente paso es generalizar la consulta del trabajador específico 1235 a una variable ID_TRABAJADOR y hacer que la consulta con dicha modificación pase a ser una subconsulta de una consulta más grande

```
Select id_trabajador
from trabajador
where not EXISTS
        (select id_edificio
        from edificio
        where not EXISTS
                (select *
                from asignacion
                where asignacion.id_edificio=edificio.id_edificio and
                        asignacion.id_trabajador=trabajador.id_trabajador))
```

Vemos que la subconsulta es idéntica a la consulta anterior salvo que se ha reemplazado el 1235 con trabajador.id_trabajador, de forma que la consulta principal puede leerse así: " Selecciona ID_TRABAJADOR de TRABAJADOR si no existe un edificio al cual ID_TRABAJADOR no está asignado"

Vemos que la solución a una consulta aparentemente sencilla y de frecuente uso no es precisamente intuitiva. Se necesitan investigaciones adicionales para desarrollar construcciones lingüísticas que permitan una solución más natural a estos tipos de consultas.

Como hemos visto, una consulta con IN o NOT IN puede ser reformulada por medio de una consulta EXISTS o NOT EXISTS, pero hay ciertas consultas que usan NOT EXISTS y que no tienen una equivalente IN.

TRABAJADOR

id_trabajador	Nomb_trabaja	tarifa_hr	oficio	id_supv
1235	M Faraday	12,5	Electricista	1311
1311	C Coulomb	15,5	Fontanero	1311
1412	C Nemo	13,75	Albañil	1520
1520	H Rickover	11,75	Carpintero	1520
2920	R Garret	10	Promotor	2920
3001	J Barrister	8,2	Electricista	3231
3231	P Mason	17,4	Carpintero	3231

ASIGNACIONES

id_trabajador	id_edificio	fecha_inicio	num_dias
1235	312	10/10/00	5
1235	515	17/10/00	22
1311	435	08/10/00	12
1311	460	23/10/00	24
1412	111	01/12/00	4
1412	210	15/11/00	12
1412	312	01/10/00	10
1412	435	15/10/00	15
1412	460	08/12/00	18
1412	515	05/11/00	8
1520	312	30/10/00	17
1520	515	09/10/00	14
2920	210	10/11/00	15
2920	435	28/10/00	10
2920	460	05/10/00	18
3001	111	08/10/00	14
3001	210	27/10/00	14
3231	111	10/10/00	8
3231	312	24/10/00	20

EDIFICIOS

id_edificio	dir_edificio	tipo	nivel_calidad	categoria
111	1213 Aspen	Oficina	4	1
210	1011 Birch	Oficina	3	1
312	123 Elm	Oficina	2	2
435	456 Maple	Comercio	1	1
460	1415 Beech	Almacén	3	3
515	789 Oak	Residencia	3	1