Guía de ejercicios de SQL

Profesor. Luis Rojas Pino

Ayudante Abraham Marianjel

Modelamiento de la información

[P1] Corretaje de propiedades

Enunciado Sean las siguientes tablas de una base de datos de una corredora de propiedades:

```
Arrendatario(RUT,Nombre,Apellido)
Arrienda(RUT,Id_casa,Deuda) Ojo: Deuda >=0 (si es 0, no hay deuda)
Telefonos(RUT,Fono)
Dueño(RUT,Nombre,Apellido)
Casa(Id_casa,RUT,Nro,Calle,Comuna)
```

Al respecto, conteste las siguientes preguntas:

1. Los arrendatarios que arriendan la casa ubicada en la calle Carrera nº 1024, Santiago.
2. ¿Cuánto le deben a María Pérez?
3. ¿Cuál es la deuda total para cada dueño?
4. Liste todas las personas de la base de datos
5. Indique los dueños que poseen tres o más casas.
6. Liste los dueños que tengan deudores en todas sus casas.

1. Este es el tipo más sencillo de consulta posible.

```
SELECT A.RUT, A.Nombre, A.Apellido
FROM Arrendatario A, Arrienda B, Casa C
WHERE A.RUT=B.RUT AND B.Id_casa=C.Id_casa
AND C.Calle='Carrera' AND C.Nro='1024' AND C.Comuna='Santiago' ;
```

2. Se supondrá que María Pérez hay una sola.

```
SELECT SUM(A.Deuda) FROM Arrienda A, Casa B, Dueño C
WHERE A.Id_casa=B.Id_casa AND B.RUT=C.RUT
AND C.Nombre='María' AND C.Apellido='Pérez' ;
```

3. Aquí es necesario agrupar la información, así la suma se hará dentro de cada grupo indicado. Entregué toda la información necesaria en el SELECT, aunque con el RUT del dueño bastaría (si en el trabajo le piden algo así, entregue todo).

```
SELECT SUM(A.Deuda), C.RUT, C.Nombre, C.Apellido
FROM Arrienda A, Casa B, Dueño C
WHERE A.Id_casa=B.Id_casa AND B.RUT=C.RUT
GROUP BY C.RUT ;
```

4. Las personas de la BD son los arrendatarios y los dueños. Para entregar ambos, hay que realizar una unión. Nota: para realizar una unión, los esquemas deben ser compatibles (atributos con mismo nombre y
dominio). Afortunadamente, éste es el caso.

```
SELECT * FROM Arrendatario UNION SELECT * FROM Dueño ;
```

5. Hay dos maneras de hacer esto: con agregación y sin ésta. El caso sin agregación (menos evidente en general) consiste en hacer un *join* de tres tablas.

Sin agregación:

```
SELECT A.RUT, A.Nombre, A.Apellido
FROM Dueño A, Casa C1, Casa C2, Casa C3
WHERE A.RUT=C1.RUT AND C1.RUT=C2.RUT AND C2.RUT=C3.RUT
AND C1.Id_casa<>C2.Id_casa AND C1.Id_casa<>C3.Id_casa
AND C2.Id_casa<>C3.Id_casa ;
```

Con agregación: en este caso, es necesario utilizar HAVING. HAVING es el WHERE pero para funciones agregadas. En el HAVING sólo pueden aparecer funciones agregadas y constantes.

```
SELECT A.RUT, A.Nombre, A.Apellido
FROM Dueño A, Casa C
WHERE A.RUT=C.RUT
GROUP BY A.RUT
HAVING COUNT(DISTINCT C.Id_casa)>=3 ;
```

6. Jugando con la semántica vemos que un dueño con deudores en todas sus casas equivale a un dueño con deuda en todas sus casas. Y el complemento de eso son los dueños con casas sin deudas.

```
SELECT D.RUT, D.Nombre, D.Apellido
FROM Dueño D, Casa C
WHERE D.RUT=C.RUT
EXCEPT
SELECT D.RUT, D.Nombre, D.Apellido
FROM Dueño D, Casa C, Arrienda A
WHERE D.RUT=C.RUT AND C.Id_casa=A.Id_casa AND A.Deuda>0
```

Otra manera consiste en exigir que la deuda de cada casa del dueño sea positiva. En este caso, una consulta
anidada exigiendo igualdad sobre ALL basta.

```
SELECT D.RUT, D.Nombre, D.Apellido
FROM Dueño D, Casa C
WHERE D.RUT=C.RUT
AND 0 = ALL (  SELECT A.Deuda
               FROM Arrienda A
               WHERE C.Id_casa=A.Id_casa)
OR NOT EXISTS ( SELECT *
                  FROM Arrienda A
                  WHERE C.Id_casa=A.Id_casa) ;
```

Esto es equivalente a lo anterior: que no exista (NOT EXISTS) una casa con deuda para este dueño. De hecho, es más eficiente.

```
SELECT D.RUT, D.Nombre, D.Apellido
FROM Dueño D, Casa C
WHERE D.RUT=C.RUT
```

```
AND NOT EXISTS ( SELECT *
                 FROM Arrienda A
                 WHERE C.Id_casa=A.Id_casa AND A.Deuda>0) ;
```

[P2] Sistema de e-learning

Enunciado *Noob Saiborg*[2] está desarrollando un sistema de evaluación automático como parte de un paquete de e-learning comercial. En este sistema, los *tests* consisten de 1 ó más preguntas con alternativas. Para hacer el sistema flexible, y para aumentar la dificultad de los problemas, uno de los requerimientos es que las preguntas -de las que consisten los tests- tengan una o más soluciones. (Por ejemplo, una pregunta del tipo "¿Cuál de las siguientes propiedades presenta el paradigma OOP?" requiere la selección de varias alternativas.) Actualmente, Noob S. cuenta con el siguiente modelo de datos relacional para modelar los tests:

```
test(tnum,titulo,curso,descripcion,autor)
preg(tnum,pnum,enunciado)
pregalt(tnum,pnum,alt,texto,correcta)     //correcta es booleana
resp(user,tnum,pnum,alt)
alumno(user,nombre,apellido)
cursa(user,curso)
contesta(user,tnum)
cursos(curso,nombre)
```

En este sencillo (primitivo) modelo de datos, Noob S. necesita resolver los siguientes problemas con SQL:

1. Sobrescribir el test X con otro test Y. (X,Y son tnum)
2. Conocer el número de tests por curso.
3. Conocer los cursos sin tests.
4. Determinar los tests con falencias. Un test tiene falencias si no tiene preguntas, si su primera pregunta (pnum) no está numerada 1, si las preguntas no son consecutivas (ej. 1,2,4,5,8), si hay preguntas con 1 ó menos alternativas[3], si todas las alternativas son verdaderas o si todas las alternativas son falsas.
5. Corregir los test cuya única falencia radica en la numeración de las preguntas.
6. Cuántos alumnos hay en cada curso.
7. Qué alumnos han contestado tests que no les corresponden (de cursos que no cursan).
8. Obtener el puntaje no normalizado de cada rendición de test. El puntaje no normalizado ha sido definido (requerimiento) como: P = buenas – malas/4. Si un alumno no contesta en una pregunta exactamente lo mismo que se ha definido como correcto, la pregunta cuenta como mala a menos que el alumno haya omitido.
9. Obtener el puntaje normalizado, o sea, de 1,0 a 7,0.

Solución Para resolver este problema es necesario saber cómo escribir consultas y cómo modificar datos con SQL.

[1] Sobrescribir el test X con el test Y se hace borrando el test X y copiando los datos del test Y con el tnum X. Haremos el proceso completo de borrado de un test, considerando si el test fue contestado, y copiaremos los datos de Y.

```
delete from test where tnum=X;
delete from preg where tnum=X;
delete from alt where tnum=X;
delete from contesta where tnum=X;
delete from resp where tnum=X;
insert into test select X, titulo, curso, descripcion, autor
                from test where tnum=Y;
insert into preg select X, pnum, enunciado
                from preg where tnum=Y;
insert into pregalt select X, pnum, alt, texto, correcta
                   from pregalt where tnum=Y;
```

[2] El número de tests por curso tiene la forma (curso, número). Es una simple y vil agrupación con count(*).

```
select curso, count(*)
from test
group by curso;
```

[3] Una simple sustracción.

```
select curso from cursos
except
select curso from test;
```

[4] Cada exigencia del enunciado puede ser consultada por separado y luego unida para obtener los tnum que corresponden a tests inválidos.

```
(
 select tnum from test
 EXCEPT select tnum from preg
)
UNION
(
 select tnum from preg
 group by tnum having min(pnum)<1
)
```

```
UNION
(
 select tnum from preg
 group by tnum having max(pnum)<>count(pnum)
)
UNION
(
 select tnum from pregalt
 group by tnum,pnum having count(alt)=1
)
UNION
(
 select tnum from (select tnum,preg from preg
                   EXCEPT select tnum,preg from pregalt)
)
UNION
(
 select tnum from pregalt
 group by tnum,alt having count(distinct cierta)=1
);
```

El resto de las preguntas se dejan como ejercicios propuestos.