Auxiliar de SQL

Problema 1

Sea el siguiente conjunto de esquemas que modelan asados:

```
ASADO(cod,lugar,fecha)
BEBIDA(cod,nom_bebida,litros)
ASISTE(cod,rut)
PERSONA(rut,nombre,apellido)
```

Conteste en SQL:

¿En dónde se hicieron asados las pasadas fiestas patrias? (17 al 21 de Septiembre)

Primero: hay un tipo fecha en SQL, el tipo DATE. Este tipo tiene la forma año-mes-día.
Segundo: en SQL siempre se pueden realizar comparaciones de desigualdad, pese al tipo de dato. Se comparará cronológicamente en fechas, alfabéticamente en cadenas, ordinalmente en números, etc. (Pero no se pueden comparar n-tuplas con desigualdades.)
Tercero: los tipos date se escriben entre comillas simples (apóstrofes), de lo contrario el *parser* entenderá una sustracción.
Cuarto: hay que empezar del 17 en vez del 18. Es sentido común. :P

Esta sencilla consulta es:

```
SELECT lugar
FROM ASADO
WHERE '2008-09-17'<=fecha AND fecha<='2008-09-21' ;
```

¿A qué asados asistió Boon Tobias?

Boon Tobias esté en PERSONA, pero la asistencia en ASISTE, por lo que hacemos un join y Liz Taylor:

```
SELECT A.cod
FROM ASISTE A, PERSONA P
WHERE A.rut=P.rut AND P.nombre="Boon" AND P.apellido="Tobias" ;
```

¿A qué asados no asistió Boon Tobias?

Esta consulta se puede hacer de dos maneras. Una solución consiste en usar una consulta anidada:

```
SELECT A.cod
FROM ASISTE A
WHERE A.rut NOT IN (SELECT P.rut
                    FROM PERSONA P
                    WHERE P.nombre="Boon" AND P.apellido="Tobias") ;
```

La otra forma sale de ver que este es el complemento, en términos de ASISTE, de la consulta anterior. Luego, utilizamos la función de sustracción, EXCEPT o MINUS:

```
SELECT A.cod FROM ASISTE A
EXCEPT
SELECT A.cod
FROM ASISTE A, PERSONA P
WHERE A.rut=P.rut AND P.nombre="Boon" AND P.apellido="Tobias" ;
```

Notemos que "ASISTE A" aparece dos veces. Esto no provoca conflictos dado que EXCEPT no anida. De hecho, ambos SELECT deben ser ejecutados separadamente con anterioridad a EXCEPT.

Estadísticas por asado: nro. Asistentes, nro. bebidas, total de litros de bebidas.

Esto es agregación con un poquito de agrupación, ya que hablamos "por asado":

```
SELECT AS.cod, COUNT(DISTINCT A.rut), COUNT(DISTINCT B.nom_bebida),
       AVG(B.litros)*COUNT(DISTINCT B.nom_bebida)
FROM ASADO AS, ASISTE A, BEBIDA B
WHERE AS.cod=A.cod AND AS.cod=B.cod ;
```

Notemos que este problema ofrece la dificultad de agregar dentro de joins y lo que pasa con los valores repetidos. *Propuesto*: qué ocurre cuando no hay asistentes(!) o bebidas(!!)? Parchar este caso.

¿Qué asados congregaron mas de 20 personas?

Esta es una consulta sobre un valor agregado o derivado. La condición sólo se permite dentro de un HAVING. Ojo que el HAVING sólo puede incluir operaciones agregadas, valores constantes y consultas agregadas, no atributos de tuplas.

```
SELECT A.cod
FROM ASISTE A
GROUP BY A.cod
HAVING COUNT(DISTINCT A.rut)>20 ;
```

¿Qué asado congregó la mayor cantidad de gente?

La forma fácil de comparar atributos agregados de distintos grupos es con anidación. Y eso es lo que necesitamos en el máximo: tener un atributo agregado que sea mayor o igual al todo grupo posible. (El mayor estricto no arroja nada cuando hay múltiples máximos.)

```
SELECT A.cod
FROM ASISTE A
GROUP BY A.cod
HAVING COUNT(DISTINCT A.rut)>=(SELECT COUNT(DISTINCT B.rut)
                               FROM ASISTE B
                               GROUP BY B.cod) ;
```

Nota: esta consulta tiene al menos un equivalente sin anidación. Encuentre uno.

Problema 2

Sea el siguiente conjunto de esquemas que modelan proyectos de inversión:

```
PROYECTO(#nro,van,inversion)
INTERFERENCIA(#nro1,#nro2,delta_van)
```

Conteste en SQL:

¿Cuántos proyectos hay en la BD?

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

Ordene los proyectos por IVAN. (IVAN=VAN/Inversion)

```
SELECT *
FROM PROYECTO
ORDER BY van/inversion ;
```

Los 10 proyectos con mayor VAN.

```
SELECT *
FROM PROYECTO P
WHERE 10 <= (SELECT COUNT(DISTINCT Q.nro)
             FROM PROYECTO Q
             WHERE Q.van>=P.van) ;
```

¿Obtenga el VAN de todos los pares de proyectos posibles. (Solo [x,y] ó [y,x]).

Propuesto. (Hecho en clase.)

¿Qué pares de proyectos tienen el mayor VAN conjunto?

Propuesto. (Similar al siguiente.)

¿Qué pares de proyectos tienen el mayor IVAN conjunto?

Propuesto. (Hecho en clase.)

Problema 3

Introducción a la optimización de consultas. Se contó brevemente:

- cómo SQL es procesado por la base de datos;
- cómo influye el *tipo de archivo* de los datos en la eficiencia de las consultas;
- los 5 *archivos* generales: heap, sorted, clustered, B+-tree, hash;
- cómo el optimizador de la base de datos optimiza una consulta SQL, al crear una consulta equivalente *óptima* en álgebra relacional y luego escoger los algoritmos *óptimos* de evaluación.