Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación
Segundo Semestre de 2018

IIC 2413 – Bases de Datos
Rúbrica Interrogación 1

Pregunta 1: Álgebra Relacional

Esquema:

- `Ciudad(cid int PRIMARY KEY, nombre varchar(100), país varchar(100))`
- `Aerolínea(aid int PRIMARY KEY, nombre varchar(100))`
- `Vuelo_Directo(cid_inicio int, cid_final int, aid int)`

(1.5 pts) El nombre de las aerolíneas que tengan vuelos directos hacia todas las ciudades Estados Unidos presentes en la tabla `Ciudad`.

Para esta pregunta la idea era hacer una división entre los vuelos de las aerolíneas y todas las ciudades de USA.

$$\rho(\text{Vuelos\_USA}, \sigma_{\text{Ciudad.país='USA'}}(\text{Vuelo\_Directo} \bowtie_{\text{Ciudad.cid=Vuelo\_directo.cid\_final}} \text{Ciudad}))$$

$$\rho(\text{Vuelos\_USA\_2}, \pi_{aid,cid\_final}(\text{Vuelos\_USA}))$$

$$\rho(\text{Ciudades\_USA}, \pi_{cid}(\sigma_{\text{Ciudad.país='USA'}}(\text{Ciudad})))$$

Para obtener el id de todas las aerolíneas que tengan vuelos hacia todas las ciudades de USA, hacemos la siguiente división

$$\text{Vuelos\_USA\_2}/\pi_{cid}(\text{Ciudades\_USA})$$

Luego obtenemos los nombres de las aerolíneas con un join. Recordar que para la división el orden de los atributos es importante, y tiene que ser una relación binaria con una unaria. También se podía hacer una versión extendida sin división usando sólo soperadores básicos.

(1 pt) Entregue todos los nombres de los pares de ciudades que están a una escala, junto con la aerolínea que realiza el vuelo.

En esta consulta es necesario hacer un join de `Vuelo_directo` consigo misma.

$$\rho(\text{V1}, \text{Vuelo\_directo})$$

$$\rho(\text{V2}, \text{Vuelo\_directo})$$

$$\text{V1} \bowtie_{\text{V1.cid\_final=V2.cid\_inicio}} \text{V2}$$

Luego para obtener los nombres, se hace el join de la relación anterior con las respectivas relaciones. Se podía asumir que la aerolínea tenía que ser la misma, para eso se agrega esa condición al join anterior.

(0.5 pts) Usando el operador $escalas_n(Vuelos)$, entregue todos los nombres de los pares de ciudades a las que puedo llegar con una o dos escalas, pero que no tienen un vuelo directo que las conecte.

Esta consulta era para familiarizarse con el operador.

$$escalas_2(\text{Vuelo\_directo}) \cup escalas_3(\text{Vuelo\_directo}) - \text{Vuelo\_directo}$$

Notar que escala 2 implica una ciudad intermedia y escala 3 implica dos.

(3 pts) Entregue una expresión en álgebra relacional que defina al operador. En base a su construcción, muestre que el operador es monótono. Piense cómo generalizar la consulta de la pregunta anterior.

Notar que $escala_n$ requiere de $n-1$ ciudades intermedias, por lo que debemos hacer $n-1$ joins. Debemos hacer un rename de `Vuelo_directo` n veces, proyectando el id de la ciudad de inicio y de la última ciudad.

$$\rho(\text{V1}, \text{Vuelo\_directo}) \ \ldots \ \rho(\text{V}_n, \text{Vuelo\_directo})$$

$$\pi_{\text{V1.cid\_inicio},\text{V}_n\text{.cid\_final}}(\text{V1} \bowtie_{\text{V1.cid\_final=V2.cid\_inicio}} \text{V2} \bowtie_{\text{V2.cid\_final=V3.cid\_inicio}} \ \cdots \ \bowtie_{\text{V}_{n-1}\text{.cid\_final=V}_n\text{.cid\_inicio}} \text{V}_n)$$

La monotonía viene dada porque la consulta sólo utiliza operadores que en clases vimos que eran monótonos.

Pregunta 2: SQL

Considerando el esquema:

```
Jugador(jid int PRIMARY KEY, nombre varchar(100), edad int)

Partido(pid int PRIMARY KEY, jid1 int, jid2 int,
        sets_j1 int, sets_j2 int,
        estadio varchar(100), fecha date,
        FOREIGN KEY jid1 REFERENCES Jugador(jid),
        FOREIGN KEY jid2 REFERENCES Jugador(jid))
```

(0.5 pts) Entrega cada identificador de partido, junto al nombre de los dos jugadores que participan en él, junto al estadio en el que jugaron.

```
SELECT P.pid, J1.nombre, J2.nombre, P.estadio
FROM Partido P, Jugador J1, Jugador J2
WHERE P.jid1 = J1.jid AND P.jid2 = J2.jid
```

(1 pt) Dado un identificador i de jugador, entrega su nombre junto al número de *Sets* que ha ganado y el número de *Sets* que ha perdido.

```
SELECT jid, SUM(sum_set_gana), SUM(sum_set_pierde) FROM (
  SELECT jid1 AS jid, sum(sets_j1) AS sum_set_gana, sum(sets_j2) AS sum_set_pierde
  FROM Partido
  GROUP BY jid1

  UNION

  SELECT jid2 as jid, sum(sets_j2) AS sum_set _gana, sum(sets_j1) AS sum_set_pierde
  FROM Partido GROUP BY jid2) AS foo
GROUP BY jid;
```

En la consulta anterior los alias a los atributos no son necesarios, pero así se evita ambigüedad.

(1.5 pt) Entrega el nombre de todos los jugadores que hayan jugado con todos los otros jugadores de la liga.

```
SELECT jid FROM(
  SELECT J1.jid, J2.jid AS jid2 FROM Jugador J1, Jugador J2
  WHERE J1.jid <> J2.jid
  EXCEPT (
    SELECT jid1, jid2 FROM Partido
    UNION
    SELECT jid2, jid1 FROM Partido)) AS Foo
```

Llamemos al `string` de la consulta anterior `Descalificados`. Primero generamos todos los posibles pares de partidos, excluyendo las tuplas que indicarían que un jugador juegue consigo mismo. Luego restamos los partidos que han ocurrido, pero debemos hacer la unión para obtener los partidos en ambas direcciones. Al hacer la proyección nos quedan los id de jugador que no han jugado con todos los jugadores. Finalmente la consulta:

```
SELECT nombre FROM Jugador WHERE Jugador.jid NOT IN(Descalificados)
```

(1.5 pt) Entrega el identificador de cada jugador, junto a su nombre y al número de partidos que ha ganado el año 2018. Si un jugador no ha jugado ese año o no tiene partidos ganados, puedes omitirlo.

```
SELECT jid, sum(sumset) FROM(
  SELECT jid1 AS jid, COUNT(sets_j1) AS sumset
  FROM Partido
  WHERE sets_j1 > sets_j2 AND EXTRACT(year FROM fecha)=2018
  GROUP BY jid1

  UNION  ALL

  SELECT jid2 AS jid, COUNT(sets_j2) AS sumset
  FROM Partido
  WHERE sets_j2 > sets_j1 AND EXTRACT(year FROM fecha)=2018
  GROUP BY jid2) AS Foo
GROUP BY jid
```

Con la consulta anterior obtenemos el id de cada jugador junto al número de partidos que ha ganado. Para el nombre hacemos el join de la consulta anterior con la tabla `Jugador`. El `UNION ALL` es necesario porque pueden venir filas duplicadas, pero no se penalizará si no se utiliza. La sintaxis para obtener el año de la fecha tampoco es estricta (por ejemplo, se podía escribir `fecha.year`).

(1.5 pts) Para los 10 jugadores que más partidos han ganado el año 2018, entrega el identificador de cada jugador, junto a su nombre y al número de partidos que ha ganado en dicho año. Puedes suponer que el resultado de la consulta anterior lo almacenaste en la vista V.

Suponemos que los atributos de V son `V(jid, nombre, sumset)`.

```
SELECT * FROM V V1 WHERE 0 >= (
  SELECT COUNT(*) FROM V V2
  WHERE V1.sumset < V2.sumset
)
```

Pregunta 3: Modelación

El diagrama es:

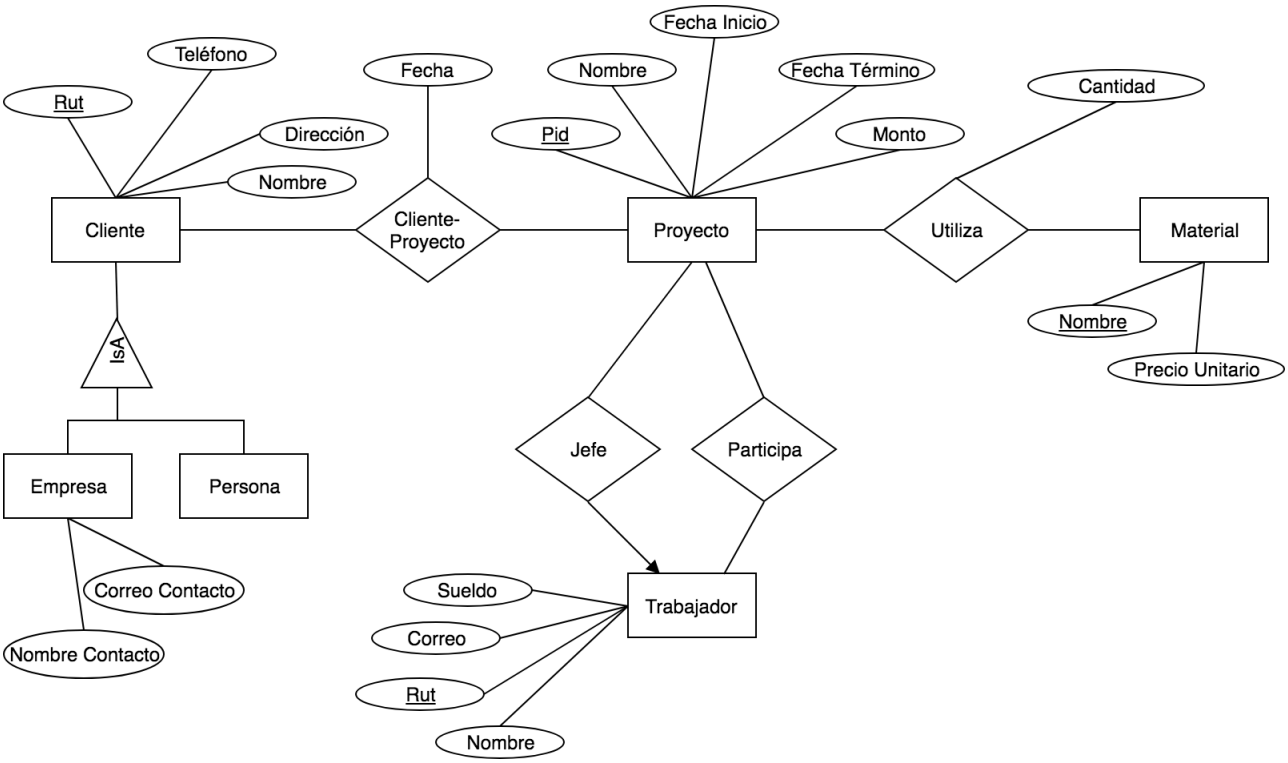

Figura 1: Diagrama P3

El esquema es inmediato a partir del diagrama, pero se aclaran las siguientes cosas:

1. Existe una tabla:

   ```
   Cliente(rut varchar(20) PRIMARY KEY,
     teléfono varchar(20), dirección varchar(100),
     nombre varchar(20))
   ```

   Que contiene todos los clientes con los atributos comunes a sus hijos. Sus hijos se ven de la siguiente forma:

   ```
   Empresa(rut varchar(20) PRIMARY KEY, cnombre varchar(20), ccorreo varchar(100))

   Persona(rut varchar(20) PRIMARY KEY)
   ```

2. No es necesario crear una tabla para la relación intermedia `Jefe`, basta con agregar un *id* de jefe a la tabla de Proyecto.

Luego para la consulta, se hace el join de `Cliente` con `Proyecto` y se suma el `monto` agrupando por el *id* del cliente.

Para las dependencias funcionales, es evidente que a cada tabla le va a corresponder una única dependencia funcional, que señala que la llave primaria determina todo lo demás. Por lo tanto el esquema está en BCNF.

Bonus (2 décimas en la interrogación)

¿A qué autor italiano se le asocia la frase "Oh vosotros los que entráis, abandonad toda esperanza"?

La frase es de Dante Alighieri, la dijo en el tercer canto de Inferno, que es la primera parte de la Divina Comedia.