Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación
Primer Semestre de 2019

IIC 2413 – Bases de Datos
Rúbrica Interrogación 1

Pregunta 1: Álgebra Relacional

Esquema:

- `Usuarios(uid INT PRIMARY KEY, nombre VARCHAR(100), edad INT)`
- `Libros(lid INT PRIMARY KEY, nombre VARCHAR(100), número_páginas INT, precio INT)`
- `Ha_Comprado(uid INT, lid INT, PRIMARY KEY(uid, lid))`

(1 pt) Entregue el nombre de todos los usuarios que han leído todos los libros de más de 800 páginas.

Para esta pregunta hay que hacer una división entre `Ha_Comprado` y los libros de más de 800 páginas.

$$\rho(\text{Libros\_800}, \sigma_{\text{Libros.número\_páginas} \geq 800}(\text{Libros}))$$

Para obtener el id de todos los usuarios que han leído dichos libros:

$$\text{Ha\_Comprado} / \pi_{lid}(\text{Libros\_800})$$

Luego obtenemos los nombres de los usuarios 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.

(0.5 pts) Entregue el libro más caro.

En esta consulta es necesario hacer un producto cruz de `Libros` consigo mismo. Vamos a sacar el `lid` de todos los libros que no son los más caros:

$$\rho(\text{L1}, \text{Libros})$$

$$\rho(\text{L2}, \text{Libros})$$

$$\rho(\text{Libros\_no\_más\_caro}, \pi_{L1.lid}(\sigma_{\text{L1.precio} < \text{L2.precio}}(\text{L1} \times \text{L2})))$$

Y para obtener el más caro:

$$\pi_{lid}(\text{Libros}) - \text{Libros\_no\_más\_caro}$$

(1 pt) Entregue los dos libros más caros.

Haremos un *rename* del *join* del `lid` del libro más caro con la tabla de libros:

$$\rho(\text{Libro\_más\_caro}, (\pi_{lid}(\text{Libros}) - \text{Libros\_no\_más\_caro}) \bowtie \text{Libros})$$

Luego podemos armar una tabla de Libros que no contenga al más caro, le llamaremos `Libros_sin_caro_1`:

$$\rho(\text{Libros\_sin\_caro\_1}, \pi_{lid}(\text{Libros}) - \text{Libro\_más\_caro})$$

Ahora repetimos el proceso de sacar el libro más caro, pero para la tabla `Libros_sin_caro_1`, lo que nos dará el segundo libro más caro. Luego hay que hacer una unión entre el libro más caro y el segundo más caro.

(0.5 pts) Usando el operador $caros_n(\text{Libros})$, entregue el nombre de todos los lectores entre 15 y 20 años (inclusive) que han leído los libros que tienen alguno de los n precios más caros, y tengan más de 500 páginas.

Para esto podemos hacer la siguiente consulta:

$$\sigma_{15 \leq \text{edad} \leq 20}(\text{Usuarios}) \bowtie \text{Ha\_Comprado} \bowtie (caros_n(\text{Libros}) \bowtie \sigma_{\text{número\_páginas}}(\text{Libros}))$$

(3 pts) Entregue una idea de cómo expresar el operador en álgebra relacional que defina al operador $caros_n(\text{Libros})$ para un n dado. Diga si este operador es o no monótono, justificando su respuesta.

(1.5 pts) Para expresar el operador repetimos el procedimiento anterior para sacar el segundo más caro:

- A la tabla Libros le restamos los $n-1$ más caros. A esta tabla le llamamos `Libros_sin_caro_(n-1)`.
- Sacamos el más caro de la tabla `Libros_sin_caro_(n-1)`. Esto corresponde al *n-ésimo* libro más caro.
- Luego podemos unir el más caro, el segundo más caro, ..., el n más caro y obtener el resultado deseado.

(1.5 pts) La consulta no es monótona. Supongamos una instancia I. Vamos a crear una instancia I' que es la misma instancia, pero después de insertar un libro más caro que todos los que estaban en la base de datos. Claramente $I \subseteq I'$. Sin embargo, si usamos $caros_1(\text{Libros})$ sobre la instancia I, el resultado será distinto que $caros_1(\text{Libros})$ sobre la instancia I', porque el libro más caro es distinto para ambas instancias. Por lo mismo, la consulta $caros_1(\text{Libros})$ sobre I no es subconjunto de $caros_1(\text{Libros})$ sobre I', a pesar de que $I \subseteq I'$, por lo que $caros_n(\text{Libros})$ no es monótono.

Pregunta 2: SQL

Considerando el esquema:

```
Alumno(numero_alumno VARCHAR(20) PRIMARY KEY, nombre varchar(100), edad int)

Curso(sigla VARCHAR(10) PRIMARY KEY, nombre VARCHAR(100), profesor VARCHAR(100))

Toma_Curso(numero_alumno VARCHAR(10), sigla VARCHAR(100),
    año INT, semestre INT,
    nota_I1 FLOAT, nota_I2 FLOAT, nota_I3 FLOAT, nota_examen FLOAT,
    PRIMARY KEY(numero_alumno, sigla, año, semestre))
```

(0.5 pts) Entregue el nombre de cada alumno, junto al nombre de cada curso que ha tomado a lo largo de su carrera.

```
SELECT Alumno.nombre, Curso.nombre
FROM Alumno, Toma_Curso, Curso
WHERE Alumno.numero_alumno = Toma_Curso.numero_alumno
    AND Curso.sigla = Toma_Curso.sigla
```

(1 pt) Para cada alumno que haya realizado un curso más de una vez, entregue cada nombre de alumno junto al nombre de los cursos que ha realizado más de una vez.

```
SELECT DISTINCT Alumno.nombre, Curso.nombre
FROM Alumno, Curso, (
  SELECT TC1.numero_alumno, TC1.sigla
  FROM Toma_Curso TC1, Toma_Curso TC2
  WHERE TC1.numero_alumno = TC2.numero_alumno
      AND TC1.sigla = TC2.sigla
      AND (TC1.año <> TC2.año OR TC1.semestre <> TC2.semestre)
) AS TC
WHERE Alumno.numero_alumno = TC.numero_alumno
    AND Curso.sigla = TC.sigla
```

(0.5 pts) Entregue el número de alumno, el nombre y la calificación final de cada alumno en el curso con sigla"IIC2413" durante el primer semestre del 2019. Puede asumir que la calificación final es $(I1 + I2 + I3 + Examen)/4$.

```
SELECT Alumno.nombre, Curso.nombre,
  ((TC.nota_I1, TC.nota_I2, TC.nota_I3, TC.nota_examen) / 4) AS NF
FROM Alumno, Toma_Curso TC, Curso
WHERE Alumno.numero_alumno = Toma_Curso.numero_alumno
    AND Curso.sigla = Toma_Curso.sigla
    AND Curso.sigla = 'IIC2413'
```

(1 pt) Entregue el nombre y la calificación final de todos los alumnos en el curso con sigla"IIC2413" durante el primer semestre del 2019, tal que su calificación final está por sobre el promedio del curso. El promedio del curso se calcula como el promedio de todas las calificaciones finales de los alumnos en el curso.

Supongamos que el resultado de la consulta anterior se almacena en una vista `AuxIIC2413`.

```
SELECT Alumno.nombre, Curso.nombre,
  ((TC.nota_I1, TC.nota_I2, TC.nota_I3, TC.nota_examen) / 4) AS NF
FROM Alumno, Toma_Curso TC, Curso
WHERE Alumno.numero_alumno = Toma_Curso.numero_alumno
    AND Curso.sigla = Toma_Curso.sigla
    AND Curso.sigla = 'IIC2413' AND Curso.semestre = 1 AND Curso.año = 2019
    AND NF >= (SELECT AVG(NF) FROM AuxIIC2413)
```

(1 pts) Entregue cada número de alumno, junto al nombre del alumno y el promedio acumulado de cada alumno, ordenados de forma decreciente según promedio acumulado. El promedio acumulado es el promedio de las calificaciones finales de todos los cursos realizados por el alumno.

Vamos a crear una vista llamada `CalificacionesFinales`:

```
CREATE VIEW CalificacionesFinales AS (
  SELECT  Alumno.numero_alumno, Alumno.nombre,
    Curso.sigla, Curso.nombre
    ((TC.nota_I1, TC.nota_I2, TC.nota_I3, TC.nota_examen) / 4) AS NF
  FROM Alumno, Toma_Curso TC, Curso
  WHERE Alumno.numero_alumno = Toma_Curso.numero_alumno
      AND Curso.sigla = Toma_Curso.sigla
)
```

Y ahora la consulta final:

```
SELECT Alumno.numero_alumno, Alumno.nombre, AVG(NF)
FROM CalificacionesFinales
GROUP BY Alumno.numero_alumno, Alumno.nombre
```

(2 pts) Entregue el número de alumno, junto al nombre y el promedio acumulado de todos los alumnos con los i promedios acumulados más altos.

Vamos a crear una vista llamada PPA con el resultado de la consulta anterior:

```
CREATE VIEW PPA AS (
  SELECT Alumno.numero_alumno, Alumno.nombre, AVG(NF) AS PAcumulado
  FROM CalificacionesFinales
  GROUP BY Alumno.numero_alumno, Alumno.nombre
)
```

Para luego hacer la consulta final:

```
SELECT * FROM PPA P1 WHERE i >= (
  SELECT COUNT(*) FROM PPA P2
  WHERE P1.PAcumulado < P2.PAcumulado
)
```

Pregunta 3: Modelación

El diagrama es:

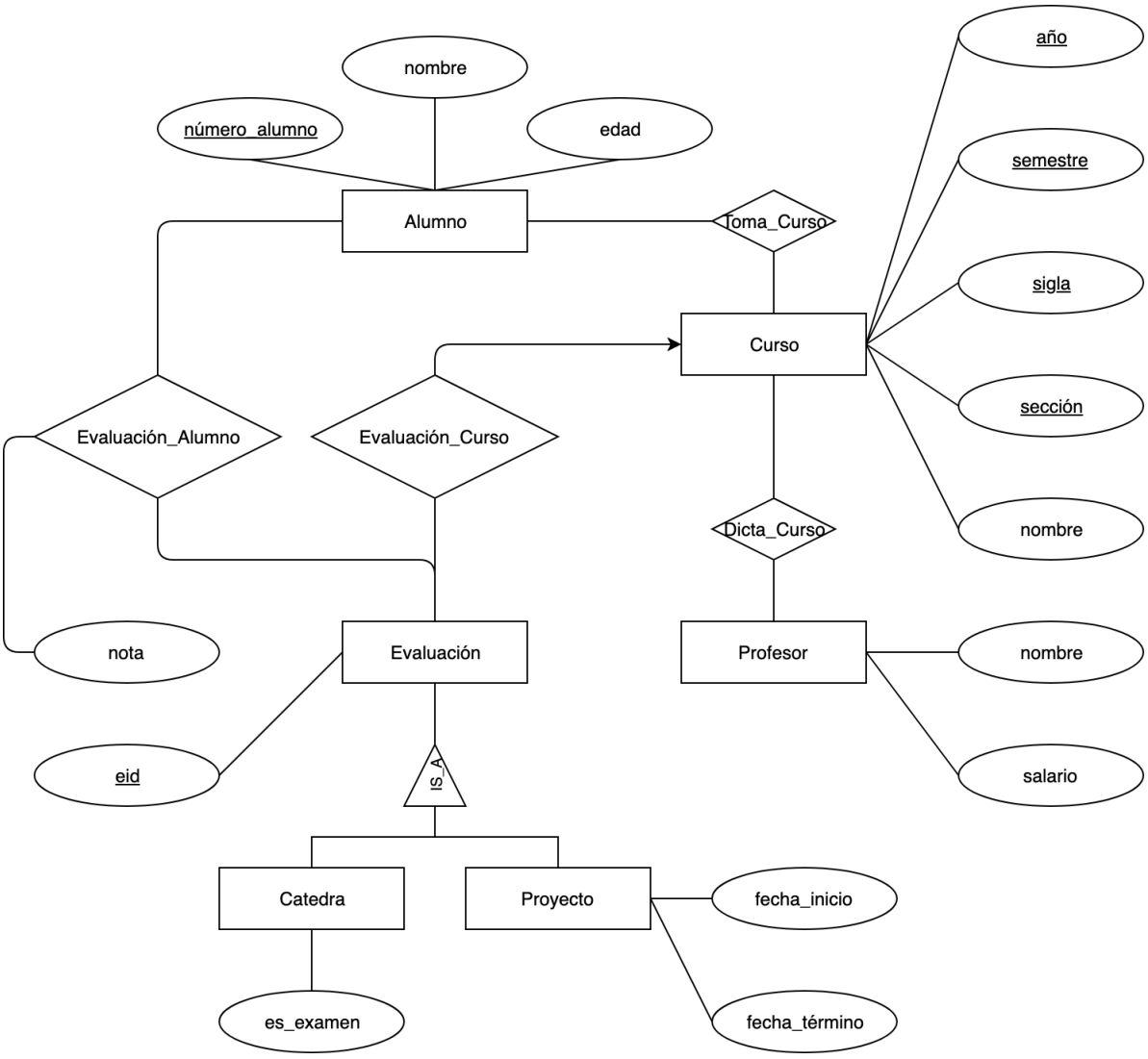

Figura 1: Diagrama P3

El esquema es inmediato a partir del diagrama, pero se lo siguiente:

Existe una tabla:

```
Evaluación(eid VARCHAR(20) PRIMARY KEY, nota float, sigla VARCHAR(20),
    FOREIGN KEY(sigla) REFERENCES Curso(sigla))
```

Que representa una evaluación. Estas varían de curso en curso, y cada versión del curso tiene distintas evaluaciones. Incluso si hay dos secciones de cierto ramo en simultáneo, a nivel de datos, la I1 de una de las versiones es distinta a la I1 de otra de las versiones. Estas evaluaciones se vinculan con los alumnos, y en la relación intermedia se agrega la nota. Las relaciones hijas se ven de la siguiente forma:

`Catedra(eid eid VARCHAR(20) PRIMARY KEY, es_examen BOOLEAN)`

`Proyecto(eid VARCHAR(20) PRIMARY KEY, fecha_inicio DATE, fecha_término DATE)`

Para la consulta la idea es la siguiente:

- Hacer una vista que haga el *join* entre `Curso`, `Evaluación`, `Cátedra`, `Evaluación_Alumno` y `Alumno`. Se filtra que el curso sea Bases de Datos en el semestre, año y sección señalados. Luego se agrega por promedio de nota, agrupado por número de alumno y nombre.
- Repetir el proceso anterior para la nota de cátedra.
- Hacer un join entre ambas vistas por número de alumno. En ese resultado, promediar la nota final de cátedra y la de proyecto.

Bonus

La empanada era sabor tomate - queso.