Práctica: Procesamiento de queries

CI-0127 Bases de Datos, Universidad de Costa Rica

Sivana Hamer

Importante

Este documento recopila contenidos de diversos de sitios web especializados, académicos y documentos compartidos por universidades. Toda la información es utilizada con fines estrictamente académicos.

Admisión UCR

Para que uno pueda entrar en una carrera en la Universidad de Costa Rica (UCR) existe un proceso de admisión. Para estudiantes de nuevo ingreso, se debe tomar un examen de admisión que es administrado anualmente. Luego, las personas que cumplan una nota minima pueden concursar para empadronarse a una carrera. El sistema de concurso de carrera tiene el modelo relacional de la Fig. 1.

Carrera:

Código	Nombre	Facultad	Escuela	Recinto

Aplicante:

Cédula	Nombre	Correo	NombreColegio	NotaColegio	NotaAdmisión

Concurso:

CédulaAplicante	CódigoCarrera	Año	Admitido

Figure 1: Parte del relacional del sistema de Admisión UCR

Álgebra relacional

Escriba los siguientes queries utilizando los operadores de álgebra relacional. Además, se recomienda escribir las consultas en SQL.

1. Obtenga los nombres de las personas aplicantes.
2. Obtenga el código y nombre de las carreras de la universidad.
3. Obtenga las carreras que son de la facultad de ingeniería.
4. Obtenga los nombres de aplicantes con una nota de admisión mayor que 600 y una nota de colegio mayor que 9.
5. Obtenga todos los concursos a la carrera con código CI (computación) del año 2021.

6. Obtenga todas las personas estudiantes que aplicaron a la carrera de computación en el 2021 que no fueron admitidos.

7. Obtenga el nombre de las personas estudiantes que fueron admitidos en el 2021 a una carrera de la facultad de ingeniería.

8. Obtenga los nombres de las personas que aplicaron en el 2020 **o** en el 2021 a la carrera de computación.

9. Obtenga los nombres de las personas que aplicaron en el 2020 **y** en el 2021 con una nota de admisión mayor a 720.

10. Obtenga el nombre de todas las carreras donde ningún aplicante aplicó en el 2021.

Heurísticas

Para las siguientes expresiones, vamos a usar A para la relación $APLICANTE$, CO para la relación $CONCURSO$ y CA para la relación $CARRERA$. Se resumen algunos atributos por efectos de espacio.

1. Represente los siguientes queries en el *query tree* inicial.
   (a) $\pi_{Escuela}(CA)$
   (b) $\pi_{Cedula,Nombre}(A)$
   (c) $\pi_{Escuela}(\sigma_{Facultad=\text{`}Ingeneria\text{'}}(CA))$
   (d) $\sigma_{CA.Co=CO.CoC}(CA \times CO)$
   (e) $\pi_{NombreColegio}(A \bowtie_{Cedula=CedulaAplicante} CO)$

2. Para los siguientes queries indique cúal es el resultado de aplicar cada paso de las heurísticas.
   (a) Para la expresión $\sigma_{NotaAdmision>420 AND NombreColegio=\text{`High School Musical'}}(A)$ el resultado de **solamente** romper σ es:
      ☐ $\pi_{NotaAdmision>420 AND NombreColegio=\text{`High School Musical'}}(A)$
      ☐ $\sigma_{NombreColegio=\text{`High School Musical'} AND NotaAdmision>420}(A)$
      ☐ $\sigma_{NotaAdmision>420}(\sigma_{NombreColegio=\text{`High School Musical'}}(A))$
      ☐ $\sigma_{NotaAdmision>420 AND NombreColegio=\text{`High School Musical'}}(A)$
   (b) Para la expresión $\sigma_{NotaAdmision>533}(\sigma_{Cedula=CedulaAplicante}(A \times CO))$ el resultado de **solamente** mover σ para abajo es:
      ☐ $((\sigma_{NotaAdmision>533}(A)) \bowtie_{Cedula=CedulaAplicante} CO)$
      ☐ $\sigma_{Cedula=CedulaAplicante}(A \times (\sigma_{NotaAdmision>533}(CO)))$
      ☐ $\sigma_{NotaAdmision>533}((\sigma_{Cedula=CedulaAplicante}(A)) \times CO)$
      ☐ $\sigma_{Cedula=CedulaAplicante}((\sigma_{NotaAdmision>533}(A)) \times CO)$
   (c) Para la expresión $(CA \bowtie_{CA.Co=CO.CoC} (\sigma_{CO.Admitido=\text{`}Y\text{'}}(CO)) \bowtie_{A.Ce=CO.CeA} A)$ el resultado de **solamente** mover σ restrictivo primero es:
      ☐ $(CA \bowtie_{CA.Co=CO.CoC} (\sigma_{CO.Admitido=\text{`}Y\text{'}}(CO)) \bowtie_{A.Ce=CO.CeA} A)$
      ☐ $(A \bowtie_{A.Ce=CO.CeA} (\sigma_{CO.Admitido=\text{`}Y\text{'}}(CO)) \bowtie_{CA.Co=CO.CoC} CA)$
      ☐ $(A \bowtie_{A.Ce=CO.CeA} ((\sigma_{CO.Admitido=\text{`}Y\text{'}}(CO)) \bowtie_{CA.Co=CO.CoC} CA))$
      ☐ $((\sigma_{CO.Admitido=\text{`}Y\text{'}}(CO)) \bowtie_{CA.Co=CO.CoC} CA \bowtie_{A.Ce=CO.CeA} A)$
   (d) Para la expresión $\sigma_{NotaAdmision>533}(\sigma_{Cedula=CedulaAplicante}(A \times CO))$ el resultado de **solamente** convertir $\bowtie$ es:
      ☐ $\sigma_{NotaAdmision>533}(\sigma_{Cedula=CedulaAplicante}(A \bowtie CO))$

- ☐ $\sigma_{NotaAdmision>533}(A \bowtie_{Cedula=CedulaAplicante} CO)$
- ☐ $\sigma_{Cedula=CedulaAplicante}(A \bowtie_{NotaAdmision>533} CO)$
- ☐ $((\sigma_{NotaAdmision>533}(A)) \bowtie_{Cedula=CedulaAplicante} CO)$

(e) Para la expresión $\pi_{A.Correo}(CO \bowtie_{A.Ce=CO.CeA} A)$ el resultado de **solamente** mover π para abajo es:

- ☐ $(\pi_{A.Ce}(A)) \bowtie_{A.Ce=CO.CeA} (\pi_{A.Correo,CO.CeA}(A))$
- ☐ $\pi_{A.Correo}((\pi_{A.Ce}(A)) \bowtie_{A.Ce=CO.CeA} (\pi_{CO.Correo,CO.CeA}(A)))$
- ☐ $\pi_{A.Correo}(CO \bowtie_{A.Ce=CO.CeA} (\pi_{A.Correo}(A)))$
- ☐ $CO \bowtie_{A.Ce=CO.CeA} (\pi_{A.Correo}(A))$

3. Optimize en su **totalidad** los siguientes queries.
   (a) $\pi_{NombreColegio}(\sigma_{NotaAdmision>750}(A))$
   (b) $\pi_{NombreColegio}(A \bowtie_{Cedula=CedulaAplicante} CO)$
   (c) $\pi_{A.Nombre}(\sigma_{C.Rec=\text{'RFB'} AND A.Ce=CO.CeA AND DCA.Co=CO.CoCAND CO.Adm=\text{'Y'}}(CO \times A \times CA))$

Sorting

Existe una base de datos con tres millones de bloques ($b = 3,000,000$) que se quiere ordenar usando *external merge sort*. n_B son el número de buffers.

1. Si se tienen cuatro buffers, ¿cuántas corridas se requieren para ordenar los bloques?
   ☐ 10 ☐ 15 ☐ 120 ☐ 500,000 ☐ 750,000 ☐ 1,000,000

2. Si se tienen seis buffers, ¿cuántos bloques se pueden ordenar en cada paso del merge?
   ☐ 5 ☐ 6 ☐ 100 ☐ 500,000 ☐ 750,000

3. ¿Cuántos son los buffers mínimos requeridos para poder ordenar con external merge sort?
   ☐ 1 ☐ 2 ☐ 3 ☐ 5 ☐ 10

4. Si se tienen diez buffers, ¿cuántas pasadas se deben realizar para ordenar el archivo?
   ☐ 2 ☐ 4 ☐ 6 ☐ 10 ☐ 100

5. Si se tienen diez buffers, ¿cuánto es el costo en lecturas y escrituras de disco de ordenar?
   ☐ 1,000,000 ☐ 6,000,000 ☐ 18,000,000 ☐ 42,000,000 ☐ 69,000,000

6. ¿Cuánto es el costo en lecturas y escrituras de disco de ordenar con los mínimos requerimientos de merge sort?
   ☐ 31,000,000 ☐ 42,000,000 ☐ 69,000,000 ☐ 120,000,000 ☐ 1,000,000,000

7. ¿Cuál es el número más pequeño de buffers que se pueden tener para que se pueda ordenar en solo dos corridas?
   ☐ 555 ☐ 556 ☐ 1,732 ☐ 1,733 ☐ 1,734 ☐ 2,007 ☐ 2,008 ☐ 2,009

8. Si se tienen veinte buffers, ¿cuál es el número de bloques más grandes que pueden ser ordenados en cuatro corridas?
   ☐ 71 ☐ 556 ☐ 1,001 ☐ 3,702 ☐ 3,703 ☐ 50,007 ☐ 137,180

Join

Suponga que se está realizando un join entre la relación aplicantes (A) y la relación concurso (CO) sobre el atributo $Cedula$ y $CedulaAplicante$. Calcule los costos de lectura y escritura a disco. No incluya los costos de escribir en memoría los resultados.

- Se tienen $n_B = 40$ bloques en el buffer.
- Se ocupan $b_A = 1,300$ bloques para guardar la relación A.
- Se ocupan $b_{CO} = 3,900$ bloques para guardar la relación CO.

1. ¿Cuál es el costo de realizar un *nested loop join* con A como la relación exterior y CO como la relación interior?

   ☐ $124,000$ ☐ $158,000$ ☐ $137,800$ ☐ $210,000$ ☐ $269,900$

2. ¿Cuál es el costo de realizar un *nested loop join* con CO como la relación exterior y A como la relación interior?

   ☐ $124,000$ ☐ $158,000$ ☐ $137,800$ ☐ $210,000$ ☐ $269,900$

3. ¿Cuál es el costo de particionar en *partition-hash join*?

   ☐ $2,080$ ☐ $4,160$ ☐ $6,240$ ☐ $8,320$ ☐ $10,400$

4. ¿Cuál es el costo de sondear en *partition-hash join*?

   ☐ $1,900$ ☐ $3,000$ ☐ $4,100$ ☐ $5,200$ ☐ $6,300$

5. ¿Cuál es el costo total de *partition-hash join*?

   ☐ $3,980$ ☐ $4,160$ ☐ $7,160$ ☐ $10,340$ ☐ $15,600$ ☐ $16,700$

6. ¿Cuál es el costo de ordenar A para el atributo $Cedula$?

   ☐ $5,200$ ☐ $6,400$ ☐ $7,600$ ☐ $8,800$ ☐ $10,000$

7. ¿Cuál es el costo de ordenar CO para el atributo $CedulaAplicante$?

   ☐ $5,400$ ☐ $10,600$ ☐ $20,100$ ☐ $23,400$ ☐ $25,600$

8. ¿Cuál es el costo de realización del merge en *sort-merge* join entre A y CO si no hay duplicados?

   ☐ $1,300$ ☐ $3,900$ ☐ $5,200$ ☐ $10,400$ ☐ $14,400$

9. En el peor caso, ¿cuál es el costo de *sort-merge* join?

   ☐ $50,720,000$ ☐ $72,600,000$ ☐ $83,000,200$ ☐ $111,200,000$ ☐ $121,680,000$

10. ¿Cuál es el costo total de *sort-merge* join entre A y CO si no hay duplicados?

    ☐ $5,200$ ☐ $28,600$ ☐ $30,120$ ☐ $33,800$ ☐ $38,600$

11. ¿Cuál es el algoritmo más eficiente? Explique.

    ☐ *Sort-merge join* ☐ *Nested loop join* con A como relación exterior. ☐ *Nested loop join* con CO como relación exterior. ☐ *Partition-hash join.*

Steam

Steam es uno de los distribuidores de juegos más grande del mundo. La compañía ha distribuido más de $50,000$ juegos de PC en centenas de géneros. Dado la calidad de su plataforma, actualmente cuenta con más de $1,000,000,000$ (miles de millones) de cuentas. Cada cuenta puede comprar diversos juegos de los cuales la plataforma recolecta información de uso. La base de datos de Steam tiene el modelo relacional de la Fig. 2 para las tablas que se relacionan con la distribución de uso.

JUEGO (J):

ID	Nombre	Descripcion	FechaLanzamiento	Genero

CUENTA (CU):

ID	Username	Correo	Nivel	Pais	Saldo

COMPRA (CO):

Juego	Cuenta	MinutosJugados	FechaCompra	UltimaFecha

Figure 2: Parte del relacional del sistema de Steam

1. Escriba la expresión de álgebra relacional que obtenga todos los géneros de juegos que se han jugado por más de 50 horas ($3,000$ minutos).

2. Se tiene la siguiente expresión de álgebra relacional:

$$\pi_{J.Nombre}(\sigma_{J.ID=CO.Juego\ AND\ CU.ID=CO.Cuenta\ AND\ CU.Pais='CR'\ AND\ CO.UF>='01-01-2021'}(CU \times CO \times J))$$

   Nota: $CO.UF = CO.UltimaFecha$

   (a) Explique brevemente, en sus propias palabras, qué consulta realiza la expresión de álgebra relacional.

   (b) Dibuje el árbol canónico de la expresión de álgebra relacional.

   (c) Optimice en totalidad, utilizando heurísticas, la expresión de álgebra relacional. Debe escribir el resultado de cada paso en un árbol de consulta (*query tree*).

3. Suponga que se está realizando un join entre CUENTA (CU) y COMPRA (CO) sobre los atributos ID y $Cuenta$, respectivamente. Estime el costo de lectura y escritura a disco. No debe incluir los costos de escribir en memoría los resultados.

   - Se tienen $n_B = 100$ bloques en el buffer.
   - Se ocupan $b_{CU} = 5,500$ bloques para guardar CU.
   - Se ocupan $b_{CO} = 10,200$ bloques para guardar CO.

   (a) ¿Cuál es el costo de realizar un *nested loop join* con CU como relación exterior y CO como la relación interior?

   (b) ¿Cuál es el costo de realizar un *nested loop join* con CO como relación exterior y CU como la relación interior?

   (c) ¿Cuál es el costo de realizar un *sort-merge join* entre CO y CU si no hay duplicados? Las relaciones **NO** se encuentran ordenadas por los atributos ID y $Cuenta$ en memoría.

   (d) ¿Cuál es el costo de realizar un *partition-hash join* con CU y CO?

   (e) Para estas estimaciones, ¿cuál es el algoritmo más eficiente? Explique.

