Ayudantía 8

Hashing, índices y reescritura de consultas

IIC2413, Bases de Datos
Benjamín Saavedra

Convenciones

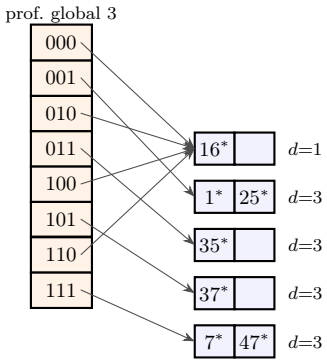
- **Extendable hashing.** Los datos son naturales en binario y $h_n(k)$ son los *últimos* n bits de k. Un bucket guarda a lo más B datos. Se parte con profundidad global 0 y un bucket vacío de profundidad local 0. Se cuenta un *split* cada vez que un bucket se divide y un *doblamiento* cada vez que la profundidad global aumenta en uno. Si después de un split el dato sigue sin caber, se aplica de nuevo el algoritmo de inserción hasta que quepa; por eso una misma inserción puede provocar varios splits y varios doblamientos.
- **B+tree de orden d.** Hojas con entre d y $2d$ datos y directorios (salvo la raíz) con entre d y $2d$ separadores. Split de hoja: d datos a la izquierda, $d+1$ a la derecha, y el separador se *copia* al padre. Split de directorio: el separador *sube*. Se inserta sin redistribución.
- **Operadores con duplicados.** Como en la clase 16, los operadores existen en dos versiones, sin y con duplicados. SQL trabaja con duplicados por defecto: una tupla puede aparecer más de una vez en una relación. Si t aparece r veces en R y s veces en S, entonces aparece $r+s$ veces en $R \cup S$ (como `UNION ALL`) y mín(r, s) veces en $R \cap S$ (`INTERSECT ALL`). El δ deja una sola copia de cada tupla y π no elimina duplicados.
- **Costos.** Se miden en páginas leídas. Cada pregunta indica el modelo que corresponde.

Pregunta 1: La historia de un extendable hashing

Un extendable hashing con $B = 2$ recibió *solo inserciones* de los datos

$$S = \{1, 7, 16, 25, 35, 37, 47\},$$

en un orden que **no conocemos**. Su estado final es:



(a) Sin conocer el orden, determine cuántos splits y cuántos doblamientos ocurrieron en total. Justifique que esos números no dependen del orden de inserción.

(b) Dé un orden de inserción de S en el que *una sola inserción* provoque todos los doblamientos. Justifique por qué ninguna inserción, en ningún orden, puede provocar más doblamientos que esos.

(c) Dé un orden de inserción de S en el que ninguna inserción provoque más de un split.

(d) Para el estado final de la figura, determine si existe un dato $k \notin S$ cuya inserción provoque: (i) ningún split; (ii) un split sin doblamiento; (iii) un split con doblamiento; (iv) dos splits. Si existe, dé una; si no existe, justifique. Si (ii) no existe, ¿cuál es la secuencia más corta de inserciones que lo provoca?

Pregunta 2: B+tree versus extendable hashing ★

Considere un B+tree de orden $d = 2$ y un extendable hashing con buckets de capacidad $B = 4$ (sin páginas de overflow). En el B+tree cada página (hoja o directorio) guarda a lo más 4 datos o 4 separadores; en el extendable hashing cada página del directorio guarda 5 punteros. El número de páginas de un índice es la suma de las de datos (hojas o buckets) y las de directorio. Se insertan números positivos.

(a) Dé una secuencia de 12 inserciones tal que el B+tree quede con *más* páginas que el extendable hashing.

(b) Dé una secuencia de 12 inserciones tal que el extendable hashing quede con *más* páginas que el B+tree.

(c) Para cada secuencia, dibuje el estado final de ambos índices y cuente sus páginas.

(d) Explique, en una frase por secuencia, qué característica de los datos explotó cada una y qué índice castiga.

(adaptada de la Interrogación 1 de 2022)

Pregunta 3: Diseño de índices para una carga de trabajo

La relación **Pedidos**(id, cliente, fecha, monto, estado) tiene 2 000 000 de tuplas, con 50 tuplas por página ($P = 40\,000$ páginas). Se pueden crear a lo más **3 índices**, **a lo más uno clustered**. Modelo de costos (en páginas leídas) para una consulta que devuelve t tuplas; el optimizador elige siempre el método más barato, incluido el *full scan*. Para el hash se asume que los buckets guardan entradas (llave, puntero), 200 por página; que las páginas del bucket solo contienen entradas de la llave buscada; y que el directorio está en memoria, no en disco (su lectura no cuenta):

Método	Costo
Full scan	$P = 40\,000$
Hash sobre el atributo (solo igualdad; siempre unclustered)	$\lceil t/200 \rceil + t$
B+tree unclustered (hojas con 200 data entries, 2 niveles de directorio)	$2 + \lceil t/200 \rceil + t$
B+tree clustered (las hojas son las páginas de datos, 50 tuplas)	$2 + \lceil t/50 \rceil$

La carga de trabajo, con su frecuencia por hora:

	Consulta	Tuplas devueltas t	Frecuencia
Q_1	`WHERE id = ...`	1	20 000
Q_2	`WHERE cliente = ...`	20	5 000
Q_3	`WHERE fecha BETWEEN ...` (un mes)	50 000	50
Q_4	`WHERE estado = 'pendiente'`	40 000	20

El costo total por hora es $\sum_i f_i \cdot (\text{costo de } Q_i)$.

(a) Calcule, para cada consulta, el costo con un índice sobre su atributo en cada una de las tres variantes (hash, B+tree unclustered, B+tree clustered) y con full scan.

(b) Elija los índices (a lo más 3, a lo más uno clustered) que minimizan el costo total por hora. Dé ese costo y diga qué método usa cada consulta.

(c) Justifique por qué el índice clustered conviene sobre fecha y no sobre cliente, estado o id.

(d) Si solo se permitiera *un* índice, ¿cuál sería? Y si Q_4 devolviera solo 200 tuplas, ¿valdría la pena un índice sobre estado?

Pregunta 4: ¿Es correcta esta equivalencia?

Esquema (llaves subrayadas):

$$\textbf{Players}(\underline{\mathsf{id}}, \mathsf{name}, \mathsf{year}, \mathsf{tid}), \qquad \textbf{Teams}(\underline{\mathsf{id}}, \mathsf{name}, \mathsf{country}),$$
$$\textbf{Matches}(\underline{\mathsf{id}}, \mathsf{stadium}, \mathsf{date}), \qquad \textbf{PlayersMatches}(\underline{\mathsf{pid}}, \underline{\mathsf{mid}}, \mathsf{goals}).$$

Abreviamos P, T, M y PM. Considere además dos relaciones PM_a y PM_b con el mismo esquema que PM (por ejemplo, los registros de dos temporadas). Un compañero propone las siguientes igualdades entre planes lógicos. Trabaje con duplicados, como en SQL. Para cada una, indique si es **correcta** (está bien definida y es equivalente a la original para toda base de datos) y **explique por qué**:

- si es verdadera, demuéstrela;

- si es falsa, diga si el problema es (A) que la expresión *no está bien definida* (a algún operador le falta un atributo) o (B) que está bien definida pero *no es equivalente*, y en ese caso dé un contraejemplo.

(a) $\pi_{name,goals}(P \bowtie_{P.id=PM.pid} PM) \stackrel{?}{=} \pi_{name}(P) \bowtie_{P.id=PM.pid} \pi_{goals}(PM)$. Si es incorrecta, corríjala.

(b) $\pi_{P.id}(P \bowtie_{P.id=PM.pid} PM) \stackrel{?}{=} \pi_{P.id}\big(\pi_{P.id}(P) \bowtie_{P.id=PM.pid} \pi_{PM.pid}(PM)\big)$

(c) $\sigma_{P.year \geq 1985 \,\vee\, PM.goals \geq 2}(P \bowtie_{P.id=PM.pid} PM) \stackrel{?}{=}$
$\sigma_{P.year \geq 1985}(P \bowtie_{P.id=PM.pid} PM) \cup \sigma_{PM.goals \geq 2}(P \bowtie_{P.id=PM.pid} PM)$

(d) $\pi_{P.name,M.stadium}\big((P \bowtie_{P.id=PM.pid} PM) \bowtie_{PM.mid=M.id} M\big) \stackrel{?}{=}$
$\pi_{P.name,M.stadium}\Big(\big(\pi_{P.id,P.name}(P) \bowtie_{P.id=PM.pid} \pi_{PM.pid,PM.mid}(PM)\big)$
$\bowtie_{PM.mid=M.id} \pi_{M.id,M.stadium}(M)\Big)$

(e) $\pi_{pid}(PM_a) \cap \pi_{pid}(PM_b) \stackrel{?}{=} \pi_{pid}(PM_a \cap PM_b)$

(f) $\delta(\pi_{tid}(P)) \stackrel{?}{=} \pi_{tid}(\delta(P))$. Si es falsa, dé una condición sobre P que la haga verdadera y justifíquela.

Pregunta 5: Reescritura y optimización con estadísticas

Esquema (llaves subrayadas):

$$\textbf{Players}(\underline{\mathsf{id}}, \mathsf{name}, \mathsf{year}, \mathsf{tid}), \qquad \textbf{Teams}(\underline{\mathsf{id}}, \mathsf{name}, \mathsf{country}),$$
$$\textbf{Matches}(\underline{\mathsf{id}}, \mathsf{stadium}, \mathsf{date}), \qquad \textbf{PlayersMatches}(\underline{\mathsf{pid}}, \underline{\mathsf{mid}}, \mathsf{goals}).$$

Abreviamos P, T, M y PM. Considere la consulta

```
SELECT DISTINCT P.name, M.stadium
FROM Teams T, Players P, PlayersMatches PM, Matches M
WHERE T.id = P.tid AND P.id = PM.pid AND PM.mid = M.id
  AND T.country = 'Chile' AND PM.goals >= 2 AND M.date >= '2020-01-01'
```

Estadísticas del catálogo: $|T| = 300$ equipos (12 chilenos, 10 páginas); $|P| = 9\,000$, repartidos uniformemente entre los equipos; $|PM| = 150\,000$ (3 000 páginas), de las cuales 15 000 cumplen $goals \geq 2$, repartidas uniformemente entre los jugadores; $|M| = 5\,000$ (100 páginas), de los cuales 1 000 cumplen $date \geq$ 2020-01-01. Cada fila de PM referencia exactamente un jugador y un partido, repartidos uniformemente. El *tamaño* de un plan es la suma de las cardinalidades de los resultados de sus *joins*.

(a) Dibuje el plan lógico directo y un plan reescrito completo (selecciones y proyecciones lo más abajo posible, productos convertidos en equijoins).

(b) Calcule el tamaño de los planes de join

- A: $\big((T' \bowtie P) \bowtie PM'\big) \bowtie M'$,
- B: $T' \bowtie \big(P \bowtie (PM' \bowtie M')\big)$,

- C: $(T' \bowtie P) \bowtie (PM' \bowtie M')$,

donde T', PM' y M' son las relaciones ya filtradas por sus condiciones. ¿Cuál es el mejor y por qué?

(c) ¿Es necesario el δ de la consulta? Dé un ejemplo si es necesario. ¿Y si el `SELECT` fuera `DISTINCT P.id, M.id`? Justifique usando las llaves.

(d) Para elegir cómo leer cada relación filtrada, use el modelo de costos de la Pregunta 3 (full scan = páginas; hash: $\lceil t/200 \rceil + t$; B+tree unclustered: $2 + \lceil t/200 \rceil + t$; B+tree clustered: $2 + \lceil t/\text{tuplas por página} \rceil$). El catálogo indica: un hash sobre Teams.country, un B+tree unclustered sobre PlayersMatches.goals y un B+tree clustered sobre Matches.date. Decida, para cada una de las tres selecciones, si conviene usar el índice o hacer full scan.