

Algoritmo UNDO (Immediate update)

Ejemplo

Ejemplo

Considere el siguiente solapamiento de transacciones.

Suponga que los valores iniciales de los ítems son $A = 60$, $B = 44$, $C = 38$.

Transacción T_1	Transacción T_2	Transacción T_3
begin leer_item(B) $B = B + 4$ escribir_item(B) commit	begin leer_item(A) leer_item(C) $A = A \div 2$ $C = C + 10$ escribir_item(A) escribir_item(C) commit	begin leer_item(B) $B = B + 5$ escribir_item(B) leer_item(A) $A = A \times 1,10$ escribir_item(A) commit

Ejercicio 1

Escriba la secuencia de registros de un *log* UNDO (omita los registros de lectura).

Respuesta 1

```
01 (BEGIN, T1);  
02 (WRITE, T1, B, 44);  
03 (BEGIN, T2);  
04 (WRITE, T2, A, 60);  
05 (WRITE, T2, C, 38);  
06 (BEGIN, T3);  
07 (COMMIT, T1);  
08 (WRITE, T3, B, 48);  
09 (COMMIT, T2);  
10 (WRITE, T3, A, 30);  
11 (COMMIT, T3);
```

Ejercicio 2

¿Hasta qué momento pueden guardarse los datos modificados por T_1 en disco?

Respuesta 2

T_1 sólo modifica B . B debe ser guardado en disco antes del *commit* de T_1 , es decir, antes de escribir (*COMMIT, T1*) en el *log* en disco.

Ejercicio 3

¿Cómo reacciona el sistema ante una falla inmediatamente después del *commit* de T1?

Respuesta 3

Cuando el sistema reinicie, será necesario deshacer (UNDO) T2 y T3, que quedarán abortadas. Para ello se deberá escribir 38 en el ítem C y 60 en el ítem A en disco. Luego se escribe en el *log* (*ABORT*, T2) y (*ABORT*, T3) y se hace *flush* del *log* a disco.

Algoritmo REDO (Deferred update)

Ejemplo

Ejercicio 1

Escriba la secuencia de registros de un *log* REDO para el mismo ejercicio considerado anteriormente (omita los registros de lectura).

Respuesta 1

```
01 (BEGIN, T1);
02 (WRITE, T1, B, 48);
03 (BEGIN, T2);
04 (WRITE, T2, A, 30);
05 (WRITE, T2, C, 48);
06 (BEGIN, T3);
07 (COMMIT, T1);
08 (WRITE, T3, B, 53);
09 (COMMIT, T2);
10 (WRITE, T3, A, 33);
11 (COMMIT, T3);
```

Ejercicio 2

¿Cómo reacciona el sistema ante una falla después del *commit* de T1?

Respuesta 2

Cuando el sistema reinicie, será necesario rehacer (REDO) T1. Para ello se deberá escribir 48 en el ítem B. Las transacciones T2 y T3 no tienen su COMMIT hecho, por lo tanto se escribe en el *log* (*ABORT*, T2) y (*ABORT*, T3) y se hace *flush* del *log* a disco.

Algoritmo UNDO/REDO

Ejemplo

Ejercicio 1

Para la siguiente secuencia de registros de *log*, indique qué ítems deben/pueden haber cambiado su valor en disco. Luego aplique el algoritmo de recuperación UNDO/REDO e indique cómo queda el archivo de *log*.

```
01 (BEGIN, T1);  
02 (WRITE, T1, A, 10, 15);  
03 (BEGIN, T2);  
04 (WRITE, T2, B, 30, 25);  
05 (WRITE, T1, C, 35, 32);  
06 (WRITE, T2, D, 14, 12);  
07 (COMMIT, T2);
```

Respuesta

Todos los ítems pueden haber cambiado su valor en disco, pero no necesariamente deben haberlo cambiado.

Al aplicar UNDO/REDO deberemos abortar T_1 . Para ello, en la fase de UNDO debemos reescribir el valor 35 en *C* y el valor 10 en *A*, en disco. Luego, en la fase de REDO debemos reescribir 25 en *B* y 12 en *D*. Por último, debemos agregar al *log* la línea (*ABORT*, T_1) y hacer *flush* del *log* a disco.

Algoritmo UNDO

Checkpointing activo

Ejemplo

Considere la siguiente secuencia de registros de un *log* UNDO con *checkpointing* activo. El sistema falla después de logear el último de ellos en disco.

```
01 (BEGIN, T1);  
02 (WRITE, T1, X, 50);  
03 (BEGIN, T2);  
04 (WRITE, T1, Y, 15);  
05 (WRITE, T2, X, 8);  
06 (BEGIN, T3);  
07 (WRITE, T3, Z, 3);  
08 (COMMIT, T1);  
09 (BEGIN CKPT, T2, T3);  
10 (WRITE, T2, X, 7);  
11 (WRITE, T3, Y, 4);
```

Ejercicio 1

¿Hasta qué línea será necesario volver atrás?

Respuesta 1

Hasta la línea 03.

Ejercicio 2

Indique cómo será el procedimiento de recuperación.

Respuesta 2

Es necesario deshacer las transacciones 2 y 3. Debemos escribir 3 en el ítem Z, 8 en el ítem X y 4 en el ítem Y, en disco. Finalmente agregamos (ABORT, T2) y (ABORT, T3) al *log*, y lo volcamos a disco.

Algoritmo REDO

Checkpointing activo

Ejemplo

Considere la siguiente secuencia de registros de un *log* REDO con *checkpointing* activo. El sistema falla después de loguear el último de ellos en disco.

```
01 (BEGIN, T1);
02 (WRITE, T1, A, 10);
03 (BEGIN, T2);
04 (WRITE, T2, B, 5);
05 (WRITE, T1, C, 7);
06 (BEGIN, T3);
07 (WRITE, T3, D, 8);
08 (COMMIT, T1);
09 (BEGIN CKPT, ....);
10 (BEGIN, T4);
11 (WRITE, T2, E, 5);
12 (COMMIT, T2);
13 (WRITE, T3, F, 7);
14 (WRITE, T4, G, 15);
15 (END CKPT);
16 (COMMIT, T3);
17 (BEGIN, T5);
18 (WRITE, T5, H, 20);
19 (BEGIN CKPT, ....);
20 (COMMIT, T5);
```

Ejercicio 1

Complete los listados de transacciones en los (BEGIN CKPT).

Respuesta 1

```
01 (BEGIN, T1);
02 (WRITE, T1, A, 10);
03 (BEGIN, T2);
04 (WRITE, T2, B, 5);
05 (WRITE, T1, C, 7);
06 (BEGIN, T3);
07 (WRITE, T3, D, 8);
08 (COMMIT, T1);
09 (BEGIN CKPT, T2, T3);
10 (BEGIN, T4);
11 (WRITE, T2, E, 5);
12 (COMMIT, T2);
13 (WRITE, T3, F, 7);
14 (WRITE, T4, G, 15);
15 (END CKPT);
16 (COMMIT, T3);
17 (BEGIN, T5);
18 (WRITE, T5, H, 20);
19 (BEGIN CKPT, T4, T5);
20 (COMMIT, T5);
```

Ejercicio 2

¿Hasta qué línea será necesario volver atrás?

Respuesta 2

Hasta la línea 03, en que comienza la transacción 2.

Ejercicio 3

Indique cómo será el procedimiento de recuperación.

Respuesta 3

Es necesario rehacer las transacciones 2, 3 y 5 (que son las que commitearon) desde la línea 03. Entonces, asignamos $B = 5$, $D = 8$, $E = 5$, $F = 7$, $H = 20$. Finalmente agregamos (ABORT, T4) al log.

Algoritmo UNDO/REDO

Checkpointing activo

Ejemplo

Considere la siguiente secuencia de registros de un *log* UNDO/REDO con *checkpointing* activo.

```
01 (BEGIN, T1);
02 (WRITE, T1, A, 60, 61);
03 (COMMIT, T1);
04 (BEGIN, T2);
05 (WRITE, T2, A, 61, 62);
06 (BEGIN, T3);
07 (WRITE, T3, B, 20, 21);
08 (WRITE, T2, C, 30, 31);
09 (BEGIN, T4);
10 (WRITE, T3, D, 40, 41);
11 (WRITE, T4, F, 70, 71);
12 (COMMIT, T3);
13 (WRITE, T2, E, 50, 51);
14 (COMMIT, T2);
15 (WRITE, T4, B, 21, 22);
16 (COMMIT, T4);
```

Ejercicio 1

Suponga que se agrega un registro (BEGIN CKPT, T1) justo después de la línea 02. ¿En qué posición del listado podría escribirse el registro (END CKPT)?

Respuesta 1

El registro (END CKPT) podría escribirse en cualquier posición después del (BEGIN CKPT, T1), siempre que ya se hayan guardado a disco todos los ítems modificados con anterioridad al (BEGIN CKPT).

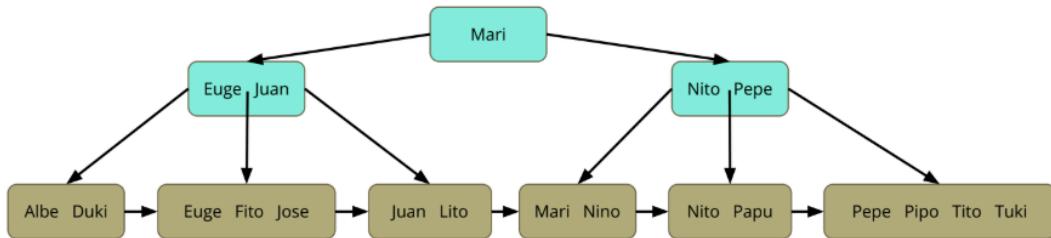
Ejercicio 2

Con ese *checkpoint* iniciado, hasta donde deberemos retroceder si se reinicia el sistema después de escribir en el *log* la línea (WRITE, T2, A, 61, 62)? Describa el procedimiento de reinicio.

Respuesta 2

Deberemos retroceder hasta el (BEGIN, T1). Hay que hacer el UNDO de la transacción *T2* y el REDO de la transacción *T1*. Debemos entonces asignar *A* = 61. Luego debemos escribir (ABORT, T2).

Índices - Árboles B+ - Ejemplo de búsqueda



- Buscar a "Papu" implica leer la raíz, ir al puntero de la derecha (Papu > Mari), luego al del medio (Nito < Papu < Pepe) y ahí se llegó al bloque que contiene "Papu"

Selección - Ejemplo

- En base a los siguientes datos de la tabla de Publicaciones, estime el costo de las siguientes consultas
 - σ Autor='Mariano Beiró' (Publicaciones)

- σ Autor='Mariano Beiró' (Publicaciones)
- σ Tipo='C' (Publicaciones)
- σ Autor='Mariano Beiró' \wedge Tipo='C' (Publicaciones)

Los índices no son de clustering

n(Publicaciones)	1,000,000
B(Publicaciones)	100,000
V(Autor, Publicaciones)	200,000
V(Tipo, Publicaciones)	10
Height(I(Autor,Publicaciones))	4
Height(I(Tipo,Publicaciones))	2

Selección - Ejemplo resuelto

- σ Autor='Mariano Beiró' (Publicaciones)

El costo es $4 + 1,000,000 / 200,000 = 9$

- σ Tipo='C' (Publicaciones)

De usar índice el costo sería $2 + 100,000$

Conviene un file scan con costo 100,000

- σ Autor='Mariano Beiró' \wedge Tipo='C' (Publicaciones)

Igual costo que el primer caso: 9

* El índice de tipo no se aprovecha (me devolvería todos los bloques que el de autor)

n(Publicaciones)	1,000,000
B(Publicaciones)	100,000
V(Autor, Publicaciones)	200,000
V(Tipo, Publicaciones)	10
Height(I(Autor,Publicaciones))	4
Height(I(Tipo,Publicaciones))	2

Selección - Ejemplo con histogramas

- Clientes(id, nombre, país,)
 - El índice por país no es de clustering

n(Clientes)	100,000
B(Clientes)	10,000
V(país, Clientes)	50
Height(l(país, Clientes))	3

Cost(σ país='Brasil' (Clientes)) = ?
 Cost(σ país='Argentina' (Clientes)) = ?
 Cost(σ país='Tonga' (Clientes)) = ?

- Histograma con 3 valores más frecuentes de país

	Argentina	Uruguay	Brasil
Clientes.pais	95,000	3,000	1,000

Selección - Distribución no uniforme de los valores

- Para Brasil, se accederan a 1,000 bloques ya que el histograma nos dice que mil filas son de brasil
 $Cost(\sigma \text{ país='Brasil' (Clientes)}) = 3 + 1000 = 1003$
- Para Argentina conviene File Scan
 $Cost(\sigma \text{ país='Argentina' (Clientes)}) = 10,000$
- Para tonga, se usa la fórmula quitando los valores conocidos
 $Cost(\sigma A_i=v(R)) = 3 + \lceil 1,000 / 47 \rceil = 3 + 22 = 25$

100K clientes menos los 95K de Argentina, 3K de Uruguay 1K de Brasil

Se sabe que son mil por el histograma

50 países menos los 3 de los que se conoce su frecuencia

Selección - Cardinalidad y Bloques en el ejemplo

- F(Clientes): $100,000 / 10,000 = 10$ (Entrar 10 filas por bloque)
- Para Brasil y Argentina se usa el histograma para la cardinalidad
 $n(\sigma \text{ país='Brasil' (Clientes)}) = 1,000$ (usa 100 bloques)
 $n(\sigma \text{ país='Argentina' (Clientes)}) = 95,000$ (usa 9,500 bloques)
- Para tonga, se estima quitando los valores conocidos
 $n(\sigma \text{ país='Tonga' (Clientes)}) = \lceil (100,000 - 99,000) / (50 - 3) \rceil$
 $n(\sigma \text{ país='Tonga' (Clientes)}) = \lceil 1,000 / 47 \rceil = 22$ (usa 3 bloques)

Join - Ejercicio

- Estimar el costo de la junta natural Clientes \bowtie Ordenes utilizando el método de loop con único índice.
- Indicar desde qué valor de memoria M es conveniente resolverlo con el método de loops anidados

Clientes (nro_cliente, nombre)		Ordenes (nro_orden, nro_cliente,)	
n(Clientes)	5,000	n(Ordenes)	10,000
F(Clientes)	20	F(Ordenes)	25
		V(nro_cliente, Ordenes)	2,500

- Hay un índice de clustering por nro_cliente en Ordenes con altura 4

Join - Ejercicio - Resolución con loop con único índice

- $B(\text{Clientes}) = 5,000 / 20 = 250$
- $B(\text{Ordenes}) = 10,000 / 25 = 400$
- $\text{Cost}(R \bowtie S) = 250 + 5000 * (4 + \lceil 400 / 2500 \rceil) = 25,250$

Clientes (nro_cliente, nombre)		Ordenes (nro_orden, nro_cliente,)	
n(Clientes)	5,000	n(Ordenes)	10,000
F(Clientes)	20	F(Ordenes)	25
		V(nro_cliente, Ordenes)	2,500

- Hay un índice de clustering por nro_cliente en Ordenes con altura 4

Join - Ejercicio - Resolución con loops anidados

- $B(\text{Clientes}) = 5,000 / 20 = 250$
- $B(\text{Ordenes}) = 10,000 / 25 = 400$
- $\text{Cost}(R \bowtie S) = 250 + \lceil 250 / (M-2) \rceil * 400$
 - Busco M tal que el costo sea menor a 25,250
 - Con $M \geq 7$ comienza a convenir usar loops anidados
 - El costo sería $250 + 50 * 400 = 20,250$

Join - Junta Hash Grace - Ejemplo

- Se quiere hacer un join entre las tablas de alumno y de notas por padrón
 - $B(\text{Alumnos}) = 100, B(\text{Notas}) = 300, M = 50$
- Se pueden armar 10 particiones basándonos en el último dígito del padrón
 - Se partitionan ambas tablas
 - Cada partición de alumnos ocupa 10 bloques y cada partición de notas, 30 bloques

Join - Junta Hash Grace - Ejemplo primera etapa

- En la **primera etapa** se divide ambas tablas en 10 particiones
 - Se precisan 11 bloques de memoria, 1 para leer la tabla y 10 para acumular particiones
- Se lee bloque a bloque cada tabla, y fila a fila se agrega al bloque de la partición que le corresponde.
- Cuando se llena un bloque de partición, se graba
- El costo es de escribir y leer ambas tablas: $2*(B(R) + B(S))$

Join - Junta Hash Grace - Ejemplo segunda etapa

- En la **segunda etapa**, se procesa la primera partición (padrón terminando en 0)
 - Se efectúa join por loops anidados entre la primera partición de alumnos y la primera partición de notas
 - Alumnos y sus notas quedan en la misma partición, ya que su padrón termina con el mismo dígito
 - Cada partición de alumnos tiene 10 bloques así que entra completa en memoria
- Se repite para las otras 9 particiones
 - Se lee, en total $B(R) + B(S)$ bloques entre todas

Join - Junta Hash Grace - Costo del ejemplo

- En total el costo fue de $3 * (B(R) + B(S))$
 - Se leyeron ambas tablas en la primera etapa
 - Se guardaron la misma cantidad de bloques (distribuidos en particiones) en la primera etapa
 - Se leyeron la misma cantidad de bloques en la segunda etapa
 - Cada join tuvo costo $B(PRI) + B(PSI)$

Join - Resumen de costos

- Loops anidados

$$\text{Cost}(R \bowtie S) = B(R) + \lceil B(R) / (M-2) \rceil * B(S)$$

- Único loop con índice

$$\text{Cost}(R \bowtie S) = B(R) + n(R) * (\text{Costo Index scan})$$

- Sort Merge

$$\text{Cost}(R \bowtie S) = 2 * B(R) * \lceil \log_{M-1}(B(R)) \rceil + B(R) + 2 * B(S) * \lceil \log_{M-1}(B(S)) \rceil + B(S)$$

- Junta Hash Grace

$$\text{Cost}(R \bowtie S) = 3 * B(R) + 3 * B(S)$$

Join - Ejercicio comparativo

- Calcule el costo de hacer el join entre R y S sabiendo que no se poseen índices a aprovechar, que $B(R) = 1,000$ y que $B(S) = 10,000$. Indique qué método es conveniente para tres casos distintos de memoria:
 1. $M = 12$
 2. $M = 102$
 3. $M = 1,002$

Join - Ejercicio comparativo - Loops anidados

- Usando la fórmula, el costo es
 $\text{Cost}(R \bowtie S) = 1,000 + \lceil 1,000 / (M-2) \rceil * 10,000$
- Para cada caso el costo termina siendo
 1. $M = 12 \Rightarrow \text{Cost}(R \bowtie S) = 1,001,000$
 2. $M = 102 \Rightarrow \text{Cost}(R \bowtie S) = 101,000$
 3. $M = 1,002 \Rightarrow \text{Cost}(R \bowtie S) = 11,000$
- Loop con único índice no puede hacerse (no hay índice)

Join - Ejercicio comparativo - Sort Merge

- Usando la fórmula, el costo es
$$\begin{aligned} \text{Cost}(R \bowtie S) &= 2,000 * \lceil \log M - 1(1,000) \rceil + 1,000 \\ &\quad + 20,000 * \lceil \log M - 1(10,000) \rceil + 10,000 \end{aligned}$$
- Para cada caso el costo termina siendo
 1. $M = 12 \Rightarrow \text{Cost}(R \bowtie S) = 97,000$
 2. $M = 102 \Rightarrow \text{Cost}(R \bowtie S) = 55,000$
 3. $M = 1,002 \Rightarrow \text{Cost}(R \bowtie S) = 53,000$

Logaritmos	1,000	10,000
$\lceil \log 11(\dots) \rceil$	3	4
$\lceil \log(101(\dots)) \rceil$	2	2
$\lceil \log 1001(\dots) \rceil$	1	2

Join - Ejercicio comparativo - Hash Grace

- Usando la fórmula, el costo es siempre igual
 $\text{Cost}(R \bowtie S) = 3 * (1,000 + 10,000)$
- Con $M=12$ se pueden hacer 11 particiones, las particiones de R tendrán 90 bloques con lo que no entran en memoria
- Recién con 102 bloques se puede hacer hash grace
 2. $M = 102 \Rightarrow \text{Cost}(R \bowtie S) = 33,000$
 3. $M = 1,002 \Rightarrow \text{Cost}(R \bowtie S) = 33,000$

M	Loops anidados	Sort Merge	Hash Grace	Mejor método
12	1,001,000	97,000		Sort Merge
102	101,000	55,000	33,000	Hash Grace
1002	11,000	53,000	33,000	Loops Anidados

Join - Ejemplo cardinalidad sin histogramas

- Clientes(id, nombre, país, ...) y Proveedores(id, nombre, país, ...)
- $n(\text{Clientes} \bowtie_{\text{clientes.pais} = \text{Proveedores.pais}} \text{Proveedores}) = ?$

n(Clientes)	100,000
V(país, Clientes)	50
B(Clientes)	10,000
n(Proveedores)	10,000
V(país, Proveedores)	40
B(Proveedores)	2,500

Join - Ejemplo cardinalidad sin histogramas

- Clientes(id, nombre, país, ...) y Proveedores(id, nombre, país, ...)
- $n(\text{Clientes} \bowtie_{\text{clientes.pais} = \text{Proveedores.pais}} \text{Proveedores}) = ?$

n(Clientes)	100,000
V(país, Clientes)	50
B(Clientes)	10,000
n(Proveedores)	10,000
V(país, Proveedores)	40
B(Proveedores)	2,500

- $100,000 * 10,000 / \max(50, 40)$
- $n(\text{Clientes} \bowtie_{\text{clientes.pais} = \text{Proveedores.pais}} \text{Proveedores}) = 20,000,000$

Join - Ejemplo cardinalidad con histogramas

- Clientes(id, nombre, país, ...) y Proveedores(id, nombre, país, ...)
- $n(\text{Clientes} \bowtie_{\text{clientes.pais} = \text{Proveedores.pais}} \text{Proveedores}) = ?$
- Histograma de 3 valores más frecuentes en cada tabla

n(Clientes)	100,000
V(país, Clientes)	50
B(Clientes)	10,000
n(Proveedores)	10,000
V(país, Proveedores)	40
B(Proveedores)	2,500

	Argentina	Uruguay	Brasil
Clientes.pais	95,000	3,000	1,000
	Argentina	Brasil	Perú
Proveedores.pais	8,000	1,000	900

Join - Ejemplo cardinalidad con histogramas

- Armamos una única tabla, agregando el “otros”
- Los valores conocidos se juntan entre sí
 - $95,000 * 8,000$ combinaciones para argentina
 - $1,000 * 1,000$ para brasil

	Argentina	Uruguay	Brasil	Perú	Otros	V(Otros)
Clientes.pais	95,000	3,000	1,000		1,000	47
Proveedores.pais	8,000		1,000	900	100	37

- Estimamos los proveedores de uruguay y clientes de perú

Join - Ejemplo cardinalidad con histogramas

- Usamos $n(\text{otros}) / V(\text{otros})$ para la estimación
- Quitamos de “otros” lo que pusimos en cada país y restamos la variabilidad de otros también
 - El n y el V de la tabla deben ser los originales

	Argentina	Uruguay	Brasil	Perú	Otros	V(Otros)
Clientes.pais	95,000	3,000	1,000	21	979	46
Proveedores.pais	8,000	3	1,000	900	97	36

- Ahora tenemos $3,000 * 3$ filas de uruguay y $21 * 900$ de perú

Join - Ejemplo cardinalidad con histogramas

- Para estimar cómo se combinan los otros, usar la fórmula sin histograma
 - En este caso $979 * 97 / \max(46, 36)$
- Finalmente se suman los de los países y los de otros

	Argentina	Uruguay	Brasil	Perú	Otros	V(Otros)
Clientes.pais	95,000	3,000	1,000	21	979	46
Proveedores.pais	8,000	3	1,000	900	97	36

Join - Ejemplo cardinalidad con histogramas

- Argentina: 760,000,000
- Uruguay: 9,000
- Otros: 2,065
- Brasil: 1,000,000
- Perú: 18,900

	Argentina	Uruguay	Brasil	Perú	Otros	V(Otros)
Clientes.pais	95,000	3,000	1,000	21	979	46
Proveedores.pais	8,000	3	1,000	900	97	36

- $n(\text{Clientes} \bowtie_{\text{clientes.pais}} \text{Proveedores}) = 761,029,965$

Join - Ejemplo cantidad de bloques

- $F(\text{Clientes}) = 10$ y $F(\text{Proveedores}) = 4$
- $F(\text{Clientes} \bowtie \text{Proveedores}) = \lfloor 1 / (0,35) \rfloor = 2$

n(Clientes)	100,000
V(país, Clientes)	50
B(Clientes)	10,000
n(Proveedores)	10,000
V(país, Proveedores)	40
B(Proveedores)	2,500

- $n(\text{Clientes} \bowtie_{\text{clientes.pais}} \text{Proveedores.pais} \text{Proveedores}) = 761,029,965$
- $B(\text{Clientes} \bowtie_{\text{clientes.pais}} \text{Proveedores.pais} \text{Proveedores}) = 380,514,983$

Proyección - Ejemplo completo

- Estime el costo, cardinalidad y cantidad de bloques de la proyección sin repetidos $\pi_{\text{Nombre, Apellido}}(\text{Clientes})$
- Sabiendo que:
 - Se dispone de $M=11$ bloques de memoria
 - El nombre y apellido ocupan un 20% del tamaño de la fila completa de Clientes
 - $n(\text{Clientes}) = 100,000$
 - $B(\text{Clientes}) = 10,000$
 - $V(\text{Nombre, Clientes}) = 200$
 - $V(\text{Apellido, Clientes}) = 400$

Proyección - Ejemplo completo resuelto

- $B(\pi_{\text{Nombre}, \text{Apellido}}(\text{Clientes})) = 10,000 * 0.2 = 2,000 (*)$

$\text{Cost}(B(\pi_{\text{X}}(R))) = 2 * B(\pi_{\text{X}}(R)) * \lceil \log_{10}(B(\pi_{\text{X}}(R))) \rceil - 2 * B(\pi_{\text{X}}(R)) + B(R)$

$\text{Cost}(B(\pi_{\text{X}}(R))) = 4,000 * \lceil \log_{10}(2,000) \rceil - 4,000 + 10,000$

$\text{Cost}(B(\pi_{\text{X}}(R))) = 4,000 * 4 + 6,000 = 22,000$

- $n(\pi_{\text{Nombre}, \text{Apellido}}(\text{Clientes})) = 200 * 400 = 80,000$
- $F(\pi_{\text{Nombre}, \text{Apellido}}(\text{Clientes})) = 10 / 0.2 = 50$
- $B(\pi_{\text{Nombre}, \text{Apellido}}(\text{Clientes})) = 1,600$

* Para el costo no consideraremos eliminar duplicados

Operadores de conjunto - Ejemplo completo

- Estime el costo, cardinalidad y cantidad de bloques de la siguiente consulta que devuelve los ids de cliente que son Premium y/o gold:

Premium \cup Gold

Asuma que un 50% de los clientes premium también son gold y que se tiene M=11 bloques de memoria

Premium(idcliente)		Gold(idcliente)	
n(Premium)	5,000	n(Gold)	100,000
B(Premium)	50	B(Gold)	1,000

Operadores de conjunto - Resolución

- El costo de ordenar cada tabla es

$\text{Cost}(\text{Ord11}(\text{Premium})) = 2 * 50 * \lceil \log_{10}(50) \rceil = 200$

$\text{Cost}(\text{Ord11}(\text{Gold})) = 2 * 1,000 * \lceil \log_{10}(1,000) \rceil = 6,000$

- El costo total es $200 + 6,000 + 50 + 1,000 = 7,250$

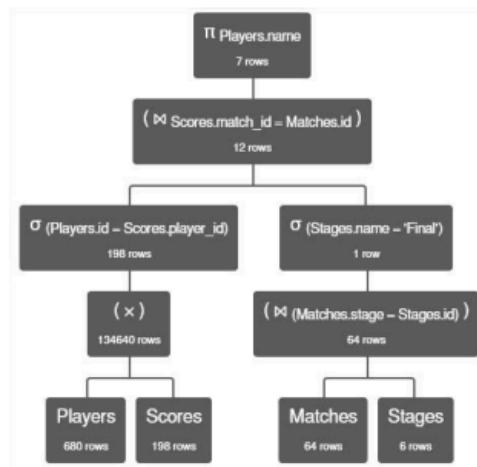
- La intersección entre ambos es la mitad de los premium, es decir 2,500.

- Entonces la unión devuelve $100,000 + 5,000 - 2,500 = 102,500$ filas

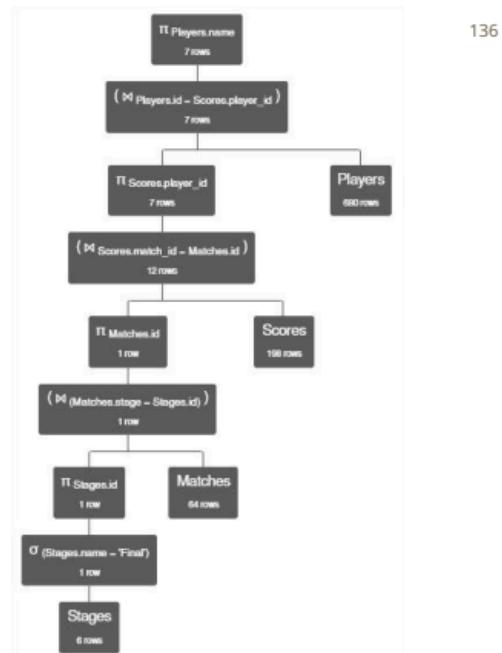
- Entran 100 por bloque entonces se devolverán 1,025 bloques

Optimización de consultas - Ejemplo

- Optimice el plan del siguiente árbol de consulta que obtiene el nombre de los jugadores que hicieron gol en la final



Optimización de consultas - Ejemplo resuelto



Pipelining - Ejemplo sencillo

- Estime el costo de la siguiente consulta:

Costo(σ padron=12345 \wedge nombre='Lucas' \wedge apellido='Román'
(Alumnos))

- La tabla tiene 1000 bloques y un índice con altura 4 sobre el padrón
- El costo es igual al de únicamente buscar por padrón y luego revisar si el nombre y apellido coinciden también
 - Costo = 4 + 1 = 5

Pipelining - Ejemplo final

- Calcule el costo de la siguiente consulta que devuelve el id de los clientes con préstamos viejos (y el id del préstamo)

Clientes(id, nombre, apellido, email,)
Prestamos(id, fecha, id_cliente, monto)

$\Pi_{\text{clientes.id}, \text{Prestamos.id}} (\text{Clientes} \bowtie_{\text{Clientes.id} = \text{Prestamos.id_cliente}} \sigma_{\text{fecha} < '2020-01-01'} (\text{Prestamos}))$

Pipelining - Ejemplo final - Metadatos

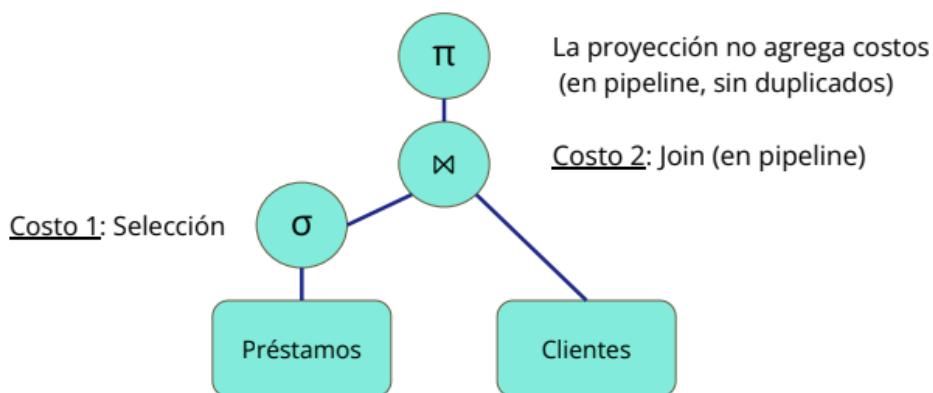
$\Pi_{\text{clientes.id}, \text{Prestamos.id}} (\text{Clientes} \bowtie_{\text{Clientes.id} = \text{Prestamos.id_cliente}} \sigma_{\text{fecha} < '2020-01-01'} (\text{Prestamos}))$

Clientes		Prestamos	
n(Clientes)	1,000,000	n(Prestamos)	5,000
B(Clientes)	100,000	B(Prestamos)	250
		V(id_cliente, Prestamos)	5,000
H(l(id_cliente, Clientes))	4	H(l(id_cliente, Prestamos))	2

- Los índices no son de clustering
- Un 10% de los préstamos son anteriores a 2020

Pipelining - Ejemplo final - Árbol de consulta

$\Pi_{\text{clientes.id}, \text{Prestamos.id}} (\text{Clientes} \bowtie_{\text{Clientes.id} = \text{Prestamos.id_cliente}} \sigma_{\text{fecha} < '2020-01-01'} (\text{Prestamos}))$



Pipelining - Ejemplo final - Costo 1

- La selección no puede aprovechar el índice
 - Se recorrerán todos los préstamos
 - Sólo se devuelve el 10% que cumple la condición de la fecha
- El costo es el del file scan
 - $C1 = B(\text{Prestamos}) = 250$
- Salen 500 préstamos (un 10% de los que habían) en 25 bloques

Pipelining - Ejemplo final - Costo 2

- Para el join podría usarse el índice
 - Si el costo es menor que recorrer una vez la tabla de clientes, entonces le gana a los otros métodos
- A medida que cada fila de préstamo es devuelta por la selección se hace el index scan
 - Se debe adaptar la fórmula para no considerar el costo $B(R)$
- $C2 = n(\text{selección}) * \text{Index_Scan}$
 $C2 = 500 * (4 + 1) = 2,500$
 - Conviene el loop con índices

Pipelining - Ejemplo final - Costo Final

- La proyección no precisa evitar duplicados
- Se hace en pipeline y no suma costos
 - Luego del join, únicamente nos quedamos con los atributos necesarios
- Costo total = $C1 + C2 = 250 + 2,500 = 2,750$