Universidad de Buenos Aires
Facultad de
Ciencias Exactas y Naturales
Departamento de Computación

Base de Datos

Segundo Cuatrimestre de 2012

Trabajo práctico 2

Buffer Manager y estrategias de reemplazo de páginas

Grupo 2

Integrante	LU	Correo electrónico
Cammi, Martín	676/02	martincammi@gmail.com
De Sousa, Mariano	389/08	marian_sabianaa@hotmail.com

Índice

1. Investigación sobre estrategia de reemplazo de páginas de Oracle

1.1. Introducción

===Hacer referencia bien a la cachÃ© indicando que es y el termino cachÃ© del buffer de que buffer

A lo largo de este trabajo analizaremos el algoritmo de *Touch Count* diseñado por *Oracle* para administrar en memoria las páginas que se traen de la base de datos con el objetivo de evitar los accesos a disco utilizando dicha memoria de a manera más eficiente.

Inicialmente Oracle utilizaba para el algoritmo de manejo de *caché* lo que se conoce como algoritmo LRU (Least Recently Used). Este algoritmo, descripto más adelante se basa en priorizar en la caché las páginas de disco mÃ¡s referenciadas descartando las menos usadas recientemente (definición de LRU).

Un caso muy común que presenta un problema a este algoritmo son los full scan, los cuales recorren todos los registros de una tabla colocándolos en la *caché* y pisando todo historial previo. Así por ejemplo, si la cache del buffer tiene 300 bloques y un escaneo completo de una tabla está recibiendo 400 bloques en la cache del buffer, todos los bloques populares desaparecerán.

Para superar este problema, Oracle propuso un algoritmo modificado de LRU al que denominó *Touch Count*

1.2. Algoritmo *Touch Count*

El algoritmo *Touch Count* utiliza el mismo espíritu que sus predecesores LRU y MRU, solo que en una mezcla de ambos. Por un lado priorizará mantener en la caché las páginas más referenciadas y descartar las menos pero a su modo, cualquier página que ingrese deberá competir con otras existentes en la caché por permanecer en la misma, para ello el algoritmo llevarÃ¡ un conteo de cuantas veces fue referenciada. Este número de referencias a la página se denomina *Touch Count*.

1.3. Problemática que resuelve

Uno de los problemas que el *Touch Count* resuelve es el de las ráfagas de acceso a páginas. En un full scan por ejemplo múltiples páginas son referenciadas y en algoritmos como

1.4. Ventajas y desventajas

2. Implementación estrategias de reemplazo de páginas

2.1. Algoritmo de LRU

El algoritmo LRU (Least Recently Used) funciona de la siguiente manera, intentando mantener en memoria las páginas más recientemente usadas y en caso de necesitar remover alguna elegir de entre alguna de las menos recientemente usadas.

El siguiente es un ejemplo de como las caché se va actualizando mediante el algoritmo LRU.

El siguiente gráfico se lee de izquierda a derecha de arriba abajo. La primera hilera de números corresponde a la traza de páginas que intentan ser accedidas. Debajo de ella aparece en forma vertical la caché y como se va llenando a medida que cáda página de la traza es referenciada.

Marcaremos con amarillo cuando ocurra un *Hit* y con gris cuando ocurra un *Miss*. Cuando una página sea referenciada la marcaremos en la caché en negrita para saber que fue la más recientemente accedida.

Ingreso de pagina i	0	3	1	0	3	7	0	8	0	4	3
Caché (tamaño 3)	**0**	0	0	**0**	0	0	**0**	0	**0**	0	0
		3	3	3	**3**	3	3	**8**	8	8	**3**
			1	1	1	**7**	7	7	7	**4**	4

Así en el ejemplo,

- Se referencia inicialmente la página 0, como la caché se encuentra vacía se produce un *Miss*, y se prodece a agregar la página 0 a la caché.
- Se referencia a la página 3, como no existe en la caché (ya que solo posee la página 0 agregada anterioremente) se produce un *Miss* y se prodece a agregar la página 3 a la caché.
- Se referencia a la página 1, como no existe en la caché se produce un *Miss* y se prodece a agregar la página 1 a la caché.
- Se referencia a la página 0, como existe en la caché se produce un *Hit*.
- Se referencia a la página 3, como existe en la caché se produce un *Hit*.
- Se referencia a la página 7, como no existe en la caché se produce un *Miss* y se prodece a agregar la página 7 a la caché reemplazando la menos recientemente usada que es la página 1.

- Se referencia a la página 0, como existe en la caché se produce un *Hit*.
- Se referencia a la página 8, como no existe en la caché se produce un *Miss* y se prodece a agregar la página 8 a la caché reemplazando la menos recientemente usada que es la página 3.
- Se referencia a la página 0, como existe en la caché se produce un *Hit*.
- Se referencia a la página 4, como no existe en la caché se produce un *Miss* y se prodece a agregar la página 4 a la caché reemplazando la menos recientemente usada que es la página 7.
- Se referencia a la página 3, como no existe en la caché (porque ya fue desalojada) se produce un *Miss* y se prodece a agregar la página 3 a la caché reemplazando la menos recientemente usada que es la página 8.

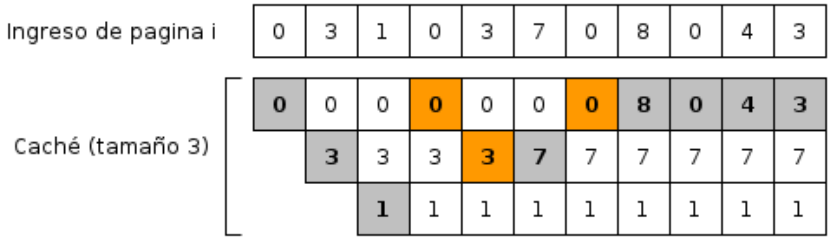
Es interesante notar que una página no será desalojada hasta tanto se hayan referenciado al menos una vez a todas las otras, ya que de esta forma la página inicial se convertirÃa en la menos recientemente usada.

Cantidad de Hits: 4
Cantidad de Miss: 7, con un total de 3 Miss iniciales y 4 Miss a lo largo de la traza.
Predicción: Este enfoque intenta predecir que páginas que fueron referenciadas posiblemente lo sean en un período corto de tiempo.
Desventajas: Una desventaja es que un full scan sobre una tabla barrerá por completo con todas las entradas de la caché.
Ventajas:

2.2. Algoritmo de MRU

El algoritmo MRU (Most Recently Used) funciona a la inversa de *LRU*, intentando mantener en memoria las páginas menos recientemente usadas y en caso de necesitar remover alguna elegir de entre alguna de las más recientemente usadas.

El siguiente es un ejemplo de como las caché se va actualizando mediante el algoritmo LRU.



El gráfico anterior se lee de izquierda a derecha de arriba abajo. La primera hilera de números corresponde a la traza de páginas que intentan ser accedidas. Debajo de ella aparece en forma vertical la caché y como se va llenando a medida que cáda página de la traza es referenciada.

Marcaremos con amarillo cuando ocurra un *Hit* y con gris cuando ocurra un *Miss*. Cuando una página sea referenciada la marcaremos en la caché en negrita para saber que fue la más recientemente accedida.

Así en el ejemplo,

- Se referencia inicialmente la página 0, como la caché se encuentra vacía se produce un *Miss*, y se prodece a agregar la página 0 a la caché.
- Se referencia a la página 3, como no existe en la caché (ya que solo posee la página 0 agregada anterioremente) se produce un *Miss* y se prodece a agregar la página 3 a la caché.
- Se referencia a la página 1, como no existe en la caché se produce un *Miss* y se prodece a agregar la página 1 a la caché.
- Se referencia a la página 0, como existe en la caché se produce un *Hit*.
- Se referencia a la página 3, como existe en la caché se produce un *Hit*.
- Se referencia a la página 7, como no existe en la caché se produce un *Miss* y se prodece a agregar la página 7 a la caché reemplazando la más recientemente usada que es la página 3.
- Se referencia a la página 0, como existe en la caché se produce un *Hit*.

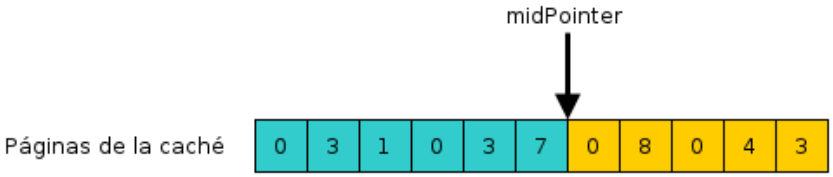
- Se referencia a la página 8, como no existe en la caché se produce un *Miss* y se prodece a agregar la página 8 a la caché reemplazando la más recientemente usada que es la página 0.
- Se referencia a la página 0, como no existe en la caché (porque ya fue desalojada) se produce un *Miss* y se prodece a agregar la página 0 a la caché reemplazando la más recientemente usada que es la página 8.
- Se referencia a la página 4, como no existe en la caché se produce un *Miss* y se prodece a agregar la página 4 a la caché reemplazando la más recientemente usada que es la página 0.
- Se referencia a la página 3, como no existe en la caché (porque ya fue desalojada) se produce un *Miss* y se prodece a agregar la página 3 a la caché reemplazando la más recientemente usada que es la página 4.

En este algoritmo se puede notar que cualquier referencia a una página existente en la caché la convierte en la potencial primera víctima para ser desalojada en caso de un próximo *Miss*.

Cantidad de Hits: 3
Cantidad de Miss: 8, con un total de 3 *Miss* iniciales y 5 *Miss* a lo largo de la traza.
Predicción: Este tipo de enfoque intenta basarse en que una vez referenciada una página no volverá a serlo al menos en un cierto período de tiempo en el cual si podrían llegar a serlo páginas más antiguas.
Desventajas:
Ventajas:

2.3. Algoritmo de Touch Count

Este algoritmo es la mejora del LRU implementada por *Oracle* describiremos un poco de la estructura interna que utiliza el TouchCount.



En la figura puede verse que Touch Count se cuenta con dos elementos, una lista y un puntero. La lista se utilizará para manejar las páginas y las prioridades que se le asignarán a cada una esta lista estará dividida en dos áreas o regiones. La región de la izquierda de la lista denominada *región fria* y la región inmediata contigua denominada *región caliente*.

Ambas regiones estarán separadas por un puntero, al que llamaremos midPointer, que se encargará de marcar dicha división. En realidad en la implementación el *midpointer* estará apuntando el primer elemento de la *región caliente* cuando haya elementos dicha región o a null en caso contrario.

Ahora describiremos su comportamiento en base a las operaciones que se realizan sobre la caché.

- Agregar una página Si la caché todavía posee espacio disponible.
- Encontrar vÃctima
- Remover una página

Peor caso del TouchCount

Si se referencian set de datos diferentes (tablas diferentes) cada vez, no se aclcanzaran para las paginas un aging count suficiente para salvarlas y todo el tiempo estaremos pisando las pÃ¡gina de la cachÃ© con pÃ¡ginas nuevas (al menos en la cola de frias)

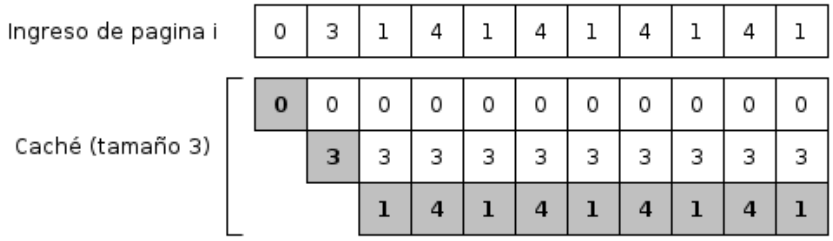
Si nada se pasa del agingHot lo que estÃ© en la Hot nunca se va a desaloar, sino que la cola cold serÃ¡ la que se renueve. Se le va a dar mÃ¡s importancia a las pÃ¡ginas con touch counte mayor al aging count mÃ¡s nuevas.

3. Test de Unidad

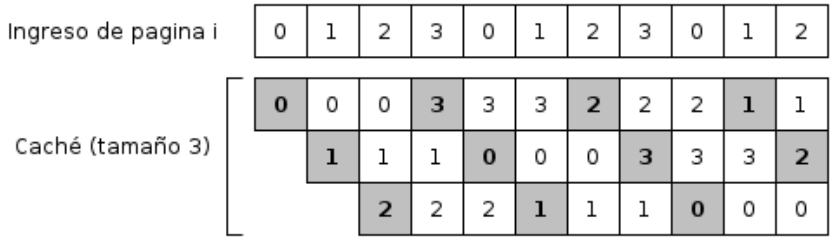
4. Comparación de Touch Count

4.1. Trazas

- Peor caso MRU: Traza MRUPathological



- Peor caso LRU



- TouchCount vs LRU
- Mejor caso LRU
- Mejor caso MRU
- Mejor caso *Touch Count*
- Peor caso *Touch Count*