Bases de Datos
Trabajo Práctico N°2

Fecha de Entrega: 16/11/2011
Fecha de Recuperatorio: 14/12/2011

Objetivos

- Evaluar el impacto de la utilización de diferentes estrategias de reemplazo de páginas en la implementación de ciertas operaciones sobre una BD.

- Analizar la importancia del módulo *Buffer Manager* en el contexto de un motor de BD.

Enunciado

El **Buffer Manager** es uno de los componentes más importantes dentro de un motor de BD. Su principal función es administrar un espacio de memoria de la BD, utilizado como una especie de memoria *caché*. El objetivo es que las diferentes aplicaciones que usan la BD y requieren páginas de disco, puedan recuperar la página de este espacio de memoria y accedan lo menos posible al disco.

El espacio de memoria administrado por el *Buffer Manager* puede ser organizado de diferentes formas y la estrategia para decidir cuál página reemplazar cuando ya no queda más espacio también puede variar.

Para este trabajo consideraremos que la memoria contiene un único *pool* de páginas compartido y nos centraremos en comparar algunas estrategias de reemplazo de páginas.

1. Implementar estrategias de reemplazo de páginas

En esta parte deberán implementar diferentes estrategias para reemplazar páginas en memoria. El *Buffer Manager* deberá poder ser configurado para trabajar con los siguientes algoritmos:

- **LRU** (*Least Recently Used*): se remueve la página de fecha de referencia más antigua.
- **MRU** (*Most Recently Used*): opuesto a LRU, se remueve la página de fecha de referencia más reciente.

Consideraremos que una página es referenciada cuando se realiza un *pin* o un *unpin* sobre ella.

En este punto, pueden tomar como guía la implementación **FIFO** que viene con el código fuente entregado.

Por último, su implementación deberá contener **tests de unidad** (deberán usar *JUnit*).

2. Evaluación de estrategias

Aquí el objetivo es comparar el comportamiento de estos algoritmos en diferentes escenarios. Deberán realizar experimentos considerando diferentes tipos de trazas de accesos a páginas, evaluando el *hit rate* (# páginas encontradas en memoria / # páginas solicitadas) para distintos tamaños de memoria. Los resultados de esos experimentos deben volcarse en el informe (por ejemplo gráficamente), presentando además un análisis sobre los resultados. En base a sus experimentos, indicar si consideran que alguno de los algoritmos se comporta mejor para las distintas clases de trazas.

Las trazas a considerar son:

- Trazas de un ***File Scan***

Ej: Suponiendo que la relación A tiene 5 páginas.

```
Request([A, 0])
Release([A, 0])
Request([A, 1])
Release([A, 1])
Request([A, 2])
Release([A, 2])
Request([A, 3])
Release([A, 3])
Request([A, 4])
Release([A, 4])
```

- Trazas de un ***Index Scan Clustered***

Ej: Suponiendo que el índice tiene 3 niveles y se recuperan 3 páginas de A.

```
Request([A_index, 0])
Release([A_index, 0])
Request([A_index, 1])
Release([A_index, 1])
Request([A_index, 2])
Release([A_index, 2])
Request([A, 3])
Release([A, 3])
Request([A, 4])
Release([A, 4])
Request([A, 5])
Release([A, 5])
```

- Trazas de un ***Index Scan Unclustered***

Ej: Suponiendo que el índice tiene 3 niveles y se recuperan 3 páginas de A.

```
Request([A_index, 0])
Release([A_index, 0])
Request([A_index, 1])
Release([A_index, 1])
Request([A_index, 2])
Release([A_index, 2])
Request([A, 7])
Release([A, 7])
Request([A, 2])
Release([A, 2])
Request([A, 2])
Release([A, 2])
```

- Trazas de un ***BNLJ***

Ej: Suponiendo que A tiene 7 páginas, B tiene 2 y los grupos de bloques son de tamaño 3.

```
Request([A, 0])
Request([A, 1])
Request([A, 2])
Request([B, 0])
Release([B, 0])
Request([B, 1])
Release([B, 1])
Release([A, 0])
Release([A, 1])
Release([A, 2])
Request([A, 3])
Request([A, 4])
Request([A, 5])
Request([B, 0])
Release([B, 0])
Request([B, 1])
Release([B, 1])
Release([A, 3])
Release([A, 4])
Release([A, 5])
Request([A, 6])
Request([B, 0])
Release([B, 0])
Request([B, 1])
Release([B, 1])
Release([A, 6])
```

Consideraciones

Para la corrección no sólo se evaluará que lo implementado funcione correctamente sino también la calidad del código generado. Con calidad nos referimos a usar comentarios, nombre declarativo para las variables o métodos, uso de métodos auxiliares, evitar duplicar código, etc. La idea es seguir utilizando este código en futuros cuatrimestres, por lo que es importante que traten de seguir las convenciones utilizadas a lo largo del código.

Entrega

La entrega deberá contar con el código que implementa lo pedido, un breve informe con detalles de implementación, decisiones tomadas y todo lo que crean conveniente y, finalmente, la comparación empírica de los diferentes algoritmos.

En los laboratorios, se llevará a cabo una demo con su tutor asignado.

Bibliografía

- Libro "Database Management Systems", Raghu Ramakrishnan – Johannes Gehrke, 2° Edición, Capítulo 7.
- Paper "Principles of Database Buffer Management", Effelssberg & Haerder, ACM Transactions on Database Systems (TODS), Volume 9 Issue 4, 1984.