DEPARTAMENTO
DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Práctico 2

17 de Julio de 2013 — Bases de Datos

Integrante	LU	Correo electrónico
Agustina Ciraco	630/06	agusciraco@gmail.com
Nadia Heredia	589/08	heredianadia@gmail.com
Pablo Antonio	290/08	pabloa@gmail.com
Vanesa Stricker	159/09	vanesastricker@gmail.com

1. Introducción

En este trabajo práctico, analizaremos el impacto de las diferentes estrategias para manejar los buffer pools. Nos concentraremos en el módulo *Buffer Manager*.

Las diferentes estrategias que consideraremos son

- un solo buffer
- múltiples buffers (distribución de Oracle)

Veremos las ventajas y desventajas de cada uno según la situación.

1.1. Formato

En primera instancia, presentaremos una breve explicación del uso de los múltiples buffer, luego se explicarán ciertos detalles de implementación, así como también las decisiones tomadas durante el desarrollo.

Por último, daremos nuestras conclusiones.

2. Múltiples Buffers

En esta sección se explica el funcionamiento de los múltiples buffer y su finalidad.

Basamos nuestro análisis en la bibliografía mencionada en la sección correspondiente.

Puede pasar que diferentes bloques interfieran entre sí, por lo cual en Oracle, se decidió utilizar *múltiples buffers*. Estos múltiples buffers en general se adecuan a la mayoría de los sistemas. Oracle implementa *múltiples buffers* utilizando una división de buffers basada en patrones de acceso atipico [1]. El tamaño de cada pool es configurable, pero los pool en sí son fijos, y son los siguientes:

- **Keep pool** Utilizado para objetos que siempre deberían estar en memoria, ya que son muy frecuentemente usados.
- **Recycle pool** Utilizado para objetos poco usados que no se desea que interfieran con el resto. En general son datos que no van a ser accedidos frecuentemente, y no se quiere que estos ocupen lugar en el pool default.
- **Default pool** Utilizado para todo lo demás.

[1] http://www.exploreoracle.com/2009/04/02/keep-buffer-pool-and-recycle-buffer-pool

3. Detalles de Implementación

En esta sección se explican algunos detalles de las implementaciones de los algoritmos.

3.1. PoolDescriptor

Teniendo en cuenta la idea de la clase `TableDescriptor`, creamos la clase `PoolDescriptor`, que almacena el nombre y el tamaño de un pool.

```
public class PoolDescriptor
{
    private String name;
    private Integer size;

    ...
}
```

3.2. Catalog

Agregamos una lista de `PoolDescriptor` a la ya existente lista de `TableDescriptor`. Agregamos un método que, dado el nombre de un Pool devuelve su tamaño, buscándolo en la lista poolDesciptors.

```
public class Catalog
{
    private List<PoolDescriptor> poolDescriptors;
    private List<TableDescriptor> tableDescriptors;

    ....

    public Integer getSizeOfPool(String name) throws Exception
    {
        for (PoolDescriptor p:poolDescriptors)
        {
            if (p.getName() == name)
            {
                return p.getSize();
            }
        }

        throw new Exception("Name does not exists");
    }
}
```

4. Biblografía

- **Oracle Docs** Oracle Database Concepts
  http://docs.oracle.com/cd/ E11882_01/server.112/e25789/toc.htm
- **Oracle Docs** Oracle Database Performance Tuning Guide
  http://docs.oracle.com/cd/ E11882_01/server.112/e16638/toc.htm
- **Artículo** Principles of Database Buffer Management, Effelssberg & Haerder
- **Reference to Oracle Technologies** Keep Buffer Pool and Recycle Buffer Pool
  http://www.exploreoracle.com/2009/04/02/ keep-buffer-pool-and-recycle-buffer-pool/
- **Praetoriate Oracle Support** Create Multiple Data Buffer Pools
  http://www.praetoriate.com/ t_oracle_net_data_buffer_pools.htm