



Presentación TP N°2

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

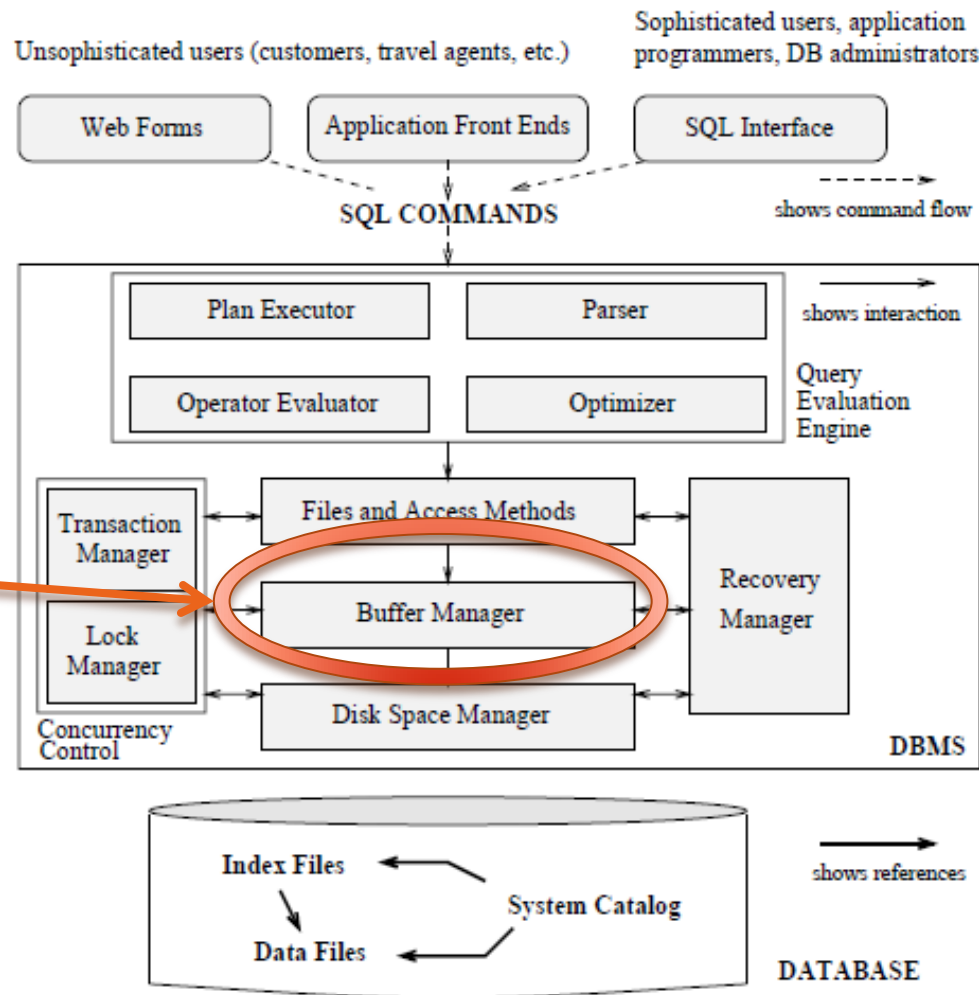
Plan

- Contexto
- UBADB
- UBADB – Buffer Manager
- Clases de interés para el TP



Contexto

Arquitectura de un motor de BD



Componentes

- **Disk Manager**

- Capa de más bajo nivel de la BD
- Maneja el espacio en disco
- Oculta los detalles de cómo se guardan las tablas en hardware
- Provee una abstracción a capas superiores, haciendo que las tablas se vean como una colección de páginas

Componentes

- **Buffer Manager**

- Responsable de traer las páginas del disco a la memoria
- Capa que administra un espacio de memoria de la BD
- Divide a la memoria en páginas de igual tamaño (formando uno o varios pooles de memoria)
- Capas superiores pueden usar las páginas sin preocuparse si están en disco o memoria, de eso se encargará este componente



UBADB

UBADB

- Motor de BD *en desarrollo* con fines académicos
 - Tecnologías que utiliza:
 - Java 1.7
 - Spring
 - Maven
 - Log4J
 - JUnit
- No es necesario conocerlos para el desarrollo del TP

UBADB

- Organización del código (paquetes)
 - **ubadb**
 - **common**: elementos comunes a toda la BD (Page, PageId, Table, etc)
 - **components**: componentes internos de la BD (Disk Manager, Buffer Manager, etc)
 - **exceptions**: excepciones generales
 - **util**: clases de utilidades para usar en cualquier parte de la aplicación

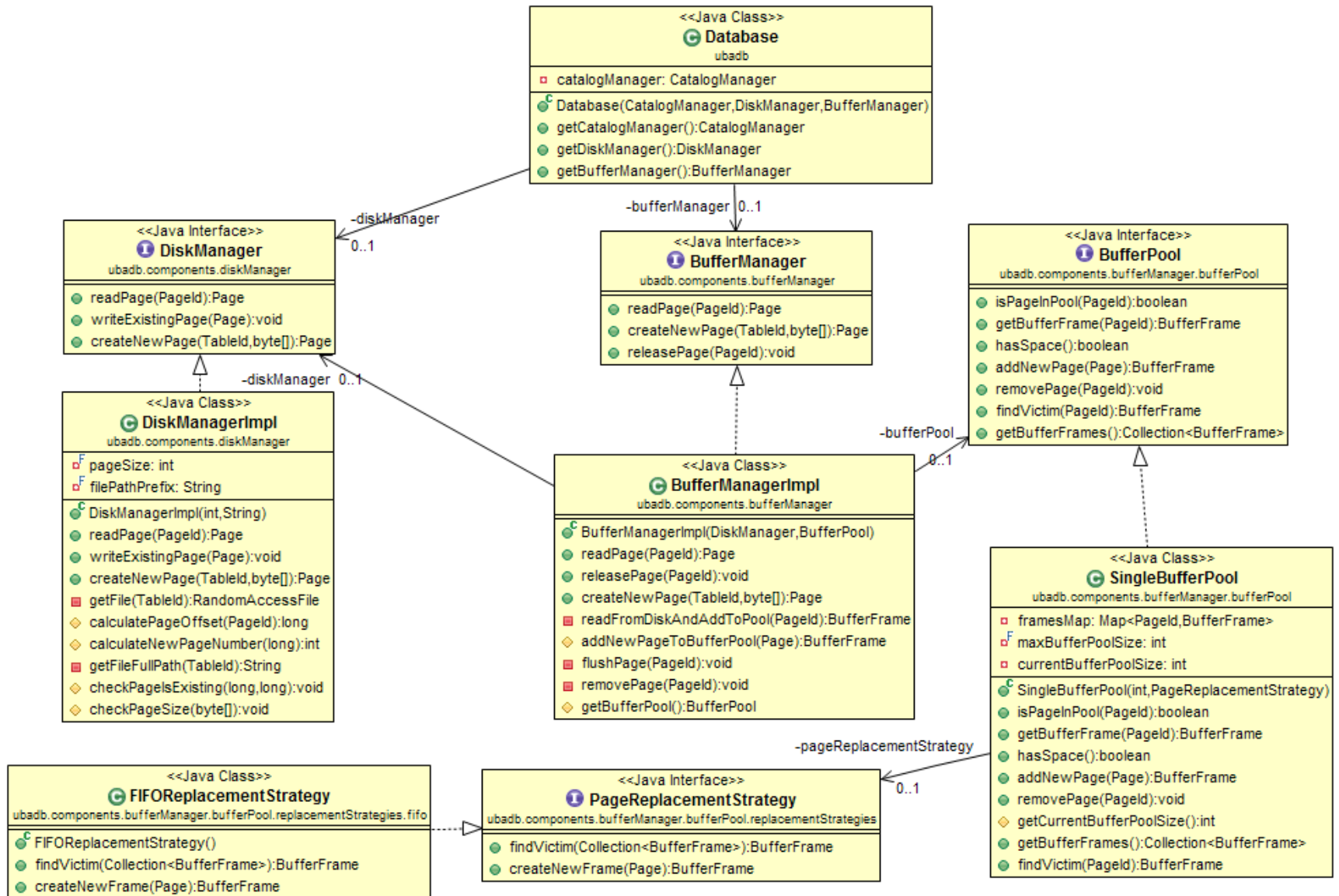
UBADB


- Organización del proyecto
 - **src/main/java**: código que implementa la BD
 - **src/main/resources**: archivos de configuración para el código principal
 - **src/test/java**: código que testea la implementación de la BD
 - **src/test/resources**: archivos útiles para los tests
 - **pom.xml**: configuración de Maven
 - **applicationContext.xml**: configuración de Spring
 - **config.properties**: configuración de la BD
 - **log4j.properties**: configuración de logueo

UBADB

- Entorno recomendado
 - Eclipse
 - Plugins útiles:
 - Maven - <http://m2eclipse.sonatype.org/sites/m2e>
 - SVN - http://subclipse.tigris.org/update_1.6.x
 - Java JDK 1.7

Diagrama de Clases





UBADB Buffer Manager

Buffer Manager

- Interactúa con el **Disk Manager** y administra el **Buffer Pool**
- **Todos** los pedidos de páginas pasan a través de él
- Servicios que provee:
 - `readPage` → devuelve la página solicitada (si no está en memoria, la buscará en disco)
 - `createNewPage` → crea una nueva página
 - `releasePage` → indica que no va a utilizar más una página dada
- *BufferManager* es una interfaz y la clase que la implementa es *BufferManagerImpl*

Buffer Pool


- Es el espacio de memoria que hace de *caché* de la BD
- *BufferPool* es una interfaz y puede tener varias implementaciones
- El pool se divide en Buffer Frames que alojarán a las páginas (además de información como el *pin count*)
- La implementación más sencilla es **SingleBufferPool**:
 - Mantiene un único pool de memoria compartido por todas las tablas
 - Es posible configurarlo para que utilice diferentes estrategias de reemplazo de páginas del pool (interfaz **PageReplacementStrategy**)
 - Hasta ahora, la única implementada es **FIFO**

Métodos importantes del BufferManager

Pseudocódigo

```
Page readPage (PageId pageId)
{
    if (bufferPool.isPageInPool (pageId))
        frame = bufferPool.getFrame (pageId)
    else
    {
        page = diskManager.getPage (pageId)
        frame = bufferPool.addPage (page)
    }

    pin (frame)
    return frame.getPage ()
}
```



En caso de no
haber espacio,
deberá reemplazar
a alguna

Métodos importantes del BufferManager

Pseudocódigo

```
Page releasePage (PageId pageId)
{
    unpin (bufferPool.getPage (pageId) )
}
```



Datos para TP



Clases de interés

- Dentro del paquete *ubadb.apps.bufferManagement* encontrarán clases útiles para la experimentación asociada al TP.
 - **PageReferenceTrace**: representa una traza de solicitudes de páginas
 - **PageReferenceTraceGenerator**: generador automático de trazas
 - **DiskManagerFaultCounterMock**: mock que simula al DiskManager pero, en lugar de ir a disco, cuenta las veces que fue llamado
 - **BufferManagementEvaluator**: sirve para evaluar el desempeño del BufferManager (según como se lo configure)

Ejemplo

- Pool de 4 frames
- Estrategias LRU y MRU
- Traza BNLJ
- Relación A de 4 páginas
- Relación B de 3 páginas
- Grupos de bloques de 2

Ejemplo - LRU

 = con pin (no se puede reemplazar)
 = sin pin (se puede reemplazar)

Frame 1	Frame 2	Frame 3	Frame 4	Acción	Hit / Miss
A-0				Request([A, 0])	MISS
A-0	A-1			Request([A, 1])	MISS
A-0	A-1	B-0		Request([B, 0])	MISS
A-0	A-1	B-0		Release([B, 0])	
A-0	A-1	B-0	B-1	Request([B, 1])	MISS
A-0	A-1	B-0	B-1	Release([B, 1])	
A-0	A-1	B-2	B-1	Request([B, 2])	MISS
A-0	A-1	B-2	B-1	Release([B, 2])	
A-0	A-1	B-2	B-1	Release([A, 0])	
A-0	A-1	B-2	B-1	Release([A, 1])	
A-0	A-1	B-2	A-2	Request([A, 2])	MISS
A-0	A-1	A-3	A-2	Request([A, 3])	MISS
B-0	A-1	A-3	A-2	Request([B, 0])	MISS
B-0	A-1	A-3	A-2	Release([B, 0])	
B-0	B-1	A-3	A-2	Request([B, 1])	MISS
B-0	B-1	A-3	A-2	Release([B, 1])	
B-2	B-1	A-3	A-2	Request([B, 2])	MISS
B-2	B-1	A-3	A-2	Release([B, 2])	

Ejemplo - MRU

 = con pin (no se puede reemplazar)

 = sin pin (se puede reemplazar)

Frame 1	Frame 2	Frame 3	Frame 4	Acción	Hit / Miss
A-0				Request([A, 0])	MISS
A-0	A-1			Request([A, 1])	MISS
A-0	A-1	B-0		Request([B, 0])	MISS
A-0	A-1	B-0		Release([B, 0])	
A-0	A-1	B-0	B-1	Request([B, 1])	MISS
A-0	A-1	B-0	B-1	Release([B, 1])	
A-0	A-1	B-0	B-2	Request([B, 2])	MISS
A-0	A-1	B-0	B-2	Release([B, 2])	
A-0	A-1	B-0	B-2	Release([A, 0])	
A-0	A-1	B-0	B-2	Release([A, 1])	
A-0	A-2	B-0	B-2	Request([A, 2])	MISS
A-3	A-2	B-0	B-2	Request([A, 3])	MISS
A-3	A-2	B-0	B-2	Request([B, 0])	HIT
A-3	A-2	B-0	B-2	Release([B, 0])	
A-3	A-2	B-1	B-2	Request([B, 1])	MISS
A-3	A-2	B-1	B-2	Release([B, 1])	
A-3	A-2	B-1	B-2	Request([B, 2])	HIT
A-3	A-2	B-1	B-2	Release([B, 2])	