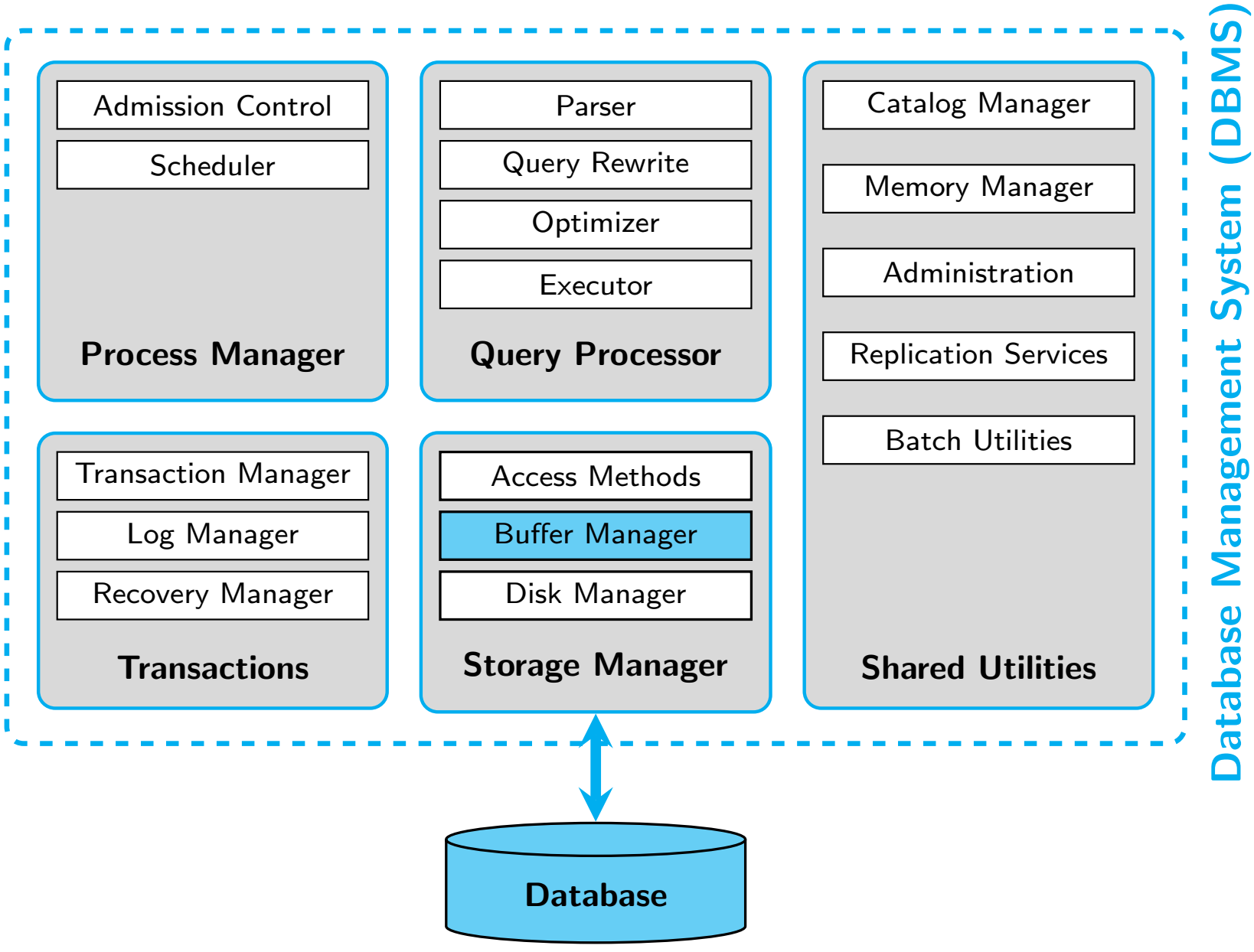
Organización de los datos

Clase 03

IIC 3413

Prof. Cristian Riveros

Organización de los datos



Considere el siguiente esquema relacional:

Players(pId, pName, pBirthdate, pDescription)
Matches(mId, mStadium, mDate)
Players_Matches(pId, mId, pGoals)

¿cómo almacenamos las tuplas de cada relación en disco?

¿cómo almacenamos las tuplas de cada relación en disco?

Posibilidades de diseño:

- Un archivo del SO para todas las relaciones (ej. SQLite).
- Un archivo del SO por cada relación.
- Varios archivos del SO por cada relación.

¿alguna desventaja?

Solución: almacenar cada relación
en un conjunto de **páginas** (bloques) de datos.

Almacenamiento basado en páginas

Una **página** corresponde a un **bloque** en disco.

Cada relación corresponde a un set de páginas que contienen un subconjunto de las tuplas.

Ventajas:

- Manejo granular del contenido en disco.
- Optimización del acceso al disco.
- Facilidad para el manejo de transacciones.

Outline

Heap files

Alternativas

Outline

Heap files

Alternativas

Heapfiles: varias aristas del problema

Heapfiles = estructura de datos para almacenar relaciones en páginas.

1. ¿cómo representamos una tupla?
2. ¿cómo almacenamos varias tuplas en una página?
3. ¿cómo almacenamos una relación en varias páginas?

Representación de una tupla

Suponga que queremos guardar una tupla de la relación:

Relación:

Players(pId, pName, pBirthdate, pDescription)

Tuplas: (ejemplo)

(7, Carlos Caszely, 5/7/1950, Jugador conocido por ...)

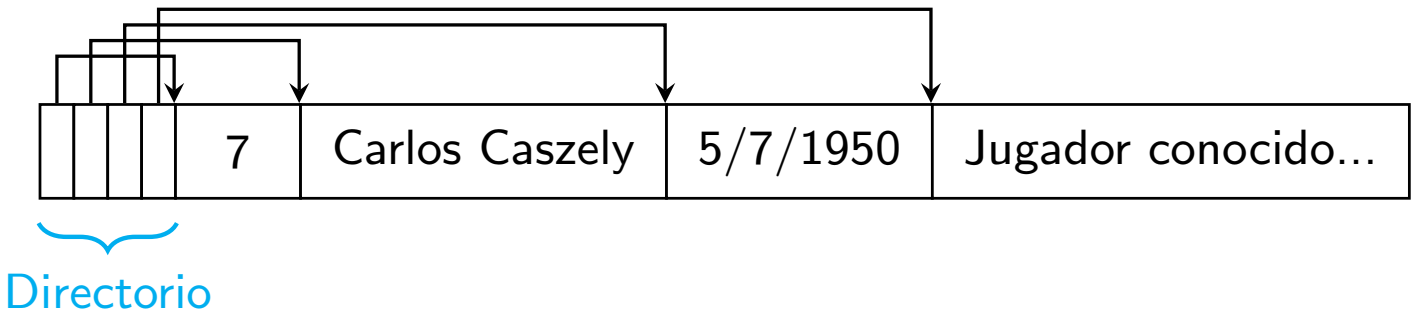
Primer approach:

#	7	$	Carlos Caszely	$	5/7/1950	$	Jugador conocido...	#

¿alguna desventaja?

Representación de una tupla

Segundo approach:

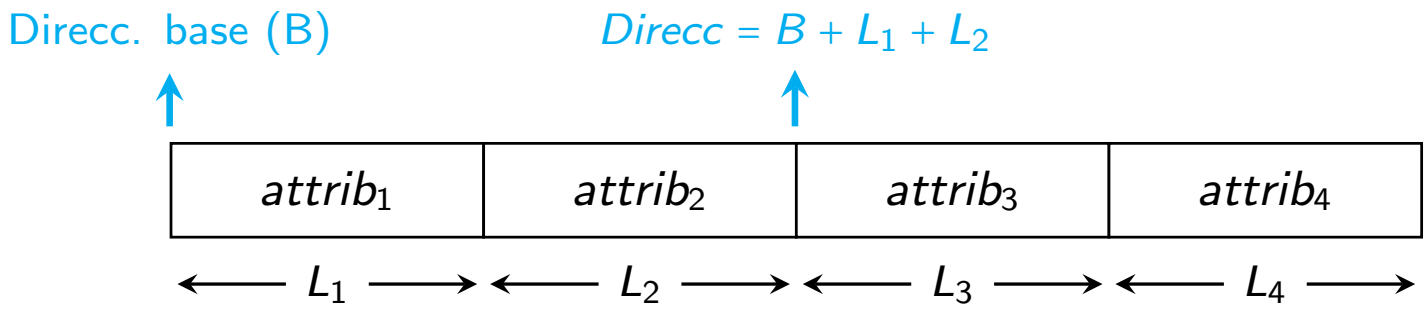


- ¿qué ocurre con los valores null?

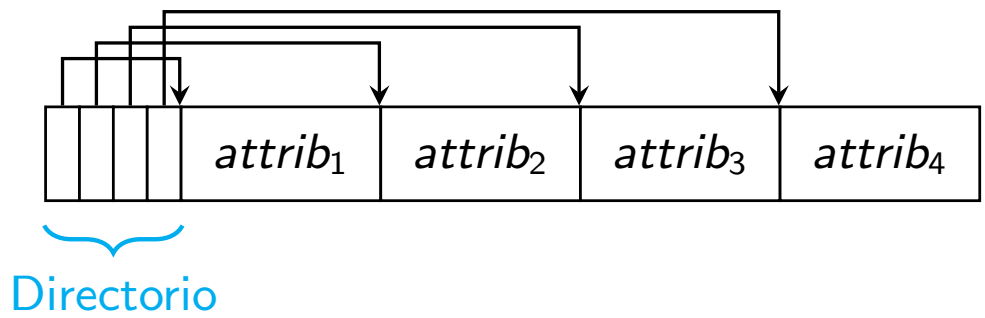
¿desventajas de este approach?

Representación de una tupla (diseño final)

Atributos de tamaño fijo:



Atributos de tamaño variable:



Heapfiles: varias aristas del problema

1. ¿cómo representamos una tupla? ✓
2. ¿cómo almacenamos varias tuplas en una página?
3. ¿cómo almacenamos una relación en varias páginas?

Record ID

- Identificador único de cada tupla.
- Necesario para mantener unicidad de cada tupla.

¿cuál es una buena elección de RID?

Usualmente, RID = $(PageID, NumSlot)$.

Tipo de almacenamiento de tuplas

1 página = 1 bloque en disco = tamaño fijo ($\approx$ 8 KB)

Tamaño de tuplas:

- Tamaño fijo.
- Tamaño variable.

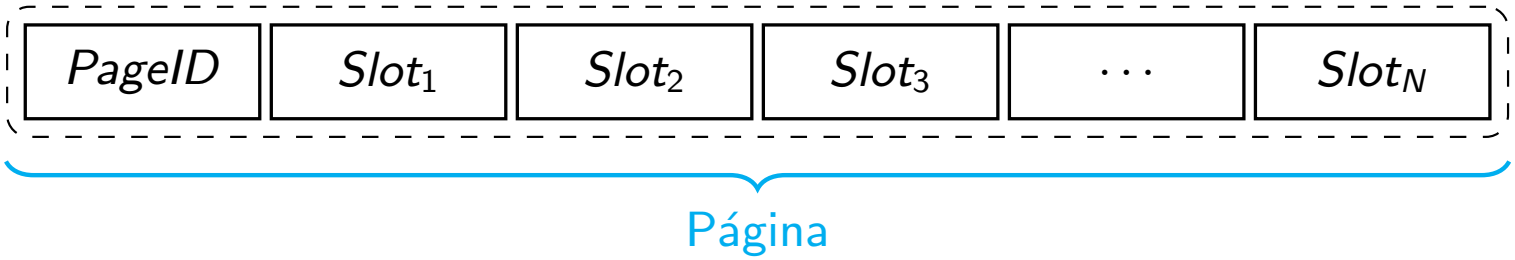
Atomicidad de una tupla:

- Cada tupla en una página (spanned).
- Una tupla en multiples páginas (unspanned).

Tipo de tuplas en una misma página:

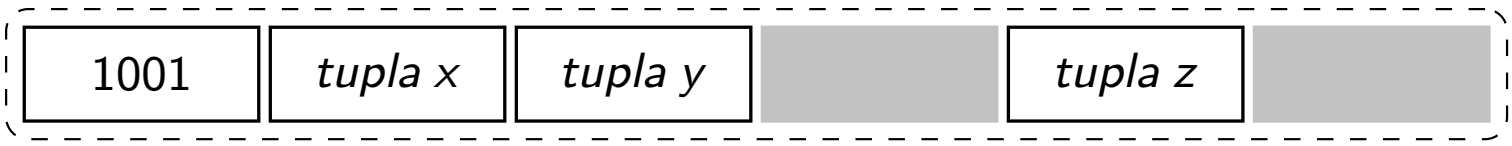
- Solo una relación (homogeneous).
- Multiples relaciones (non-homogeneous).

Formato página: tuplas de tamaño fijo



Record ID (RID) para una tupla en el $slot_n$ viene dado por: $(PageID, n)$.

Ejemplo página:

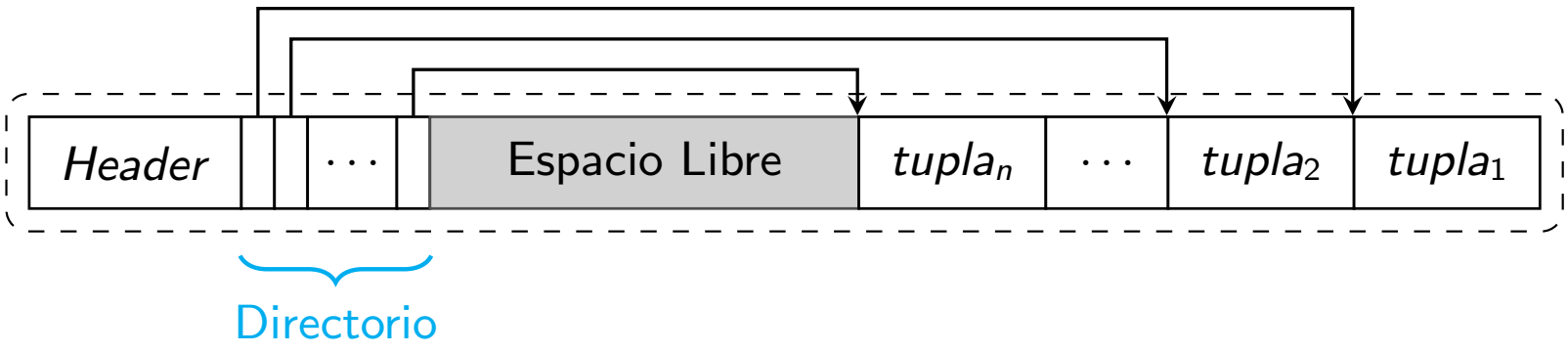


¿cómo insertamos/eliminamos una tupla en este formato?

Formato página: tuplas de tamaño variable

¿cómo almacenamos varias tuplas de tamaño variable en una página?

Formato página: tuplas de tamaño variable

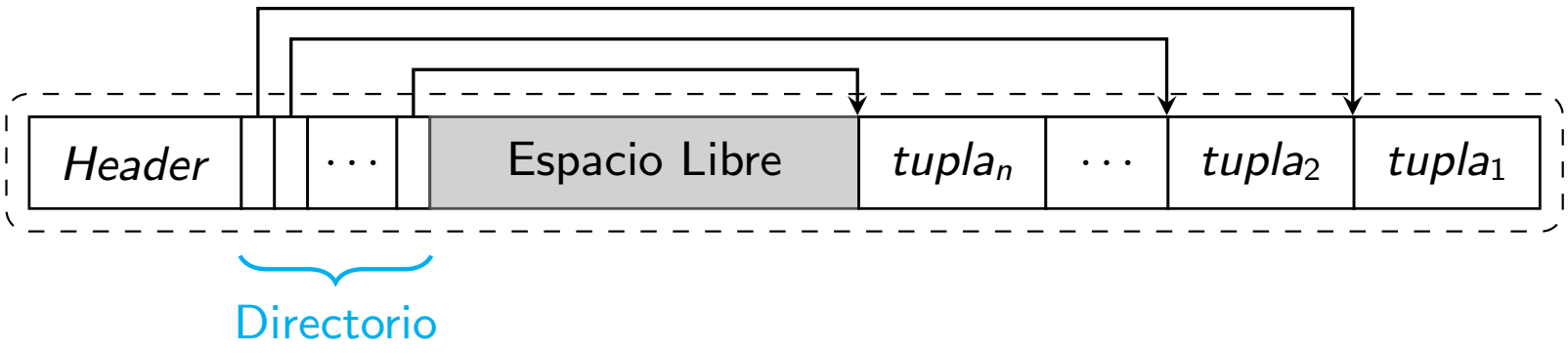


Header contiene el *PageID* + número de tuplas + tamaño directorio, etc.

Record ID (RID) = (*PageID*, *DirectoryID*).

¿cómo insertamos/eliminamos una tupla en este formato?

Formato página: tuplas de tamaño variable



¿Ventajas de este formato?

- Manejo de tuplas de tamaño variable.
- Eliminación de tuplas.
- Mover tuplas dentro de una página (indireccionamiento), o dentro de distintas páginas (forwarding address).

Formato página: tuplas de tamaño variable

Suposición del formato anterior:

Tuplas de tamaño variable
pero no mayor *al tamaño de una página.*

¿cómo almacenamos tuplas/atributos de tamaño mayor a una página?

`char(n)` vs. `varchar(n)` vs. `clob(n)`

BLOB = Binary Large OBject
CLOB = Character Large OBject

- Soportado por la mayoría de DBMS modernos.
- Incluyen: imagenes, videos, textos, etc.

¿cuál es la diferencia entre `char(n)`, `varchar(n)`, y `clob(n)`?

Implementación estandar para `clob`:

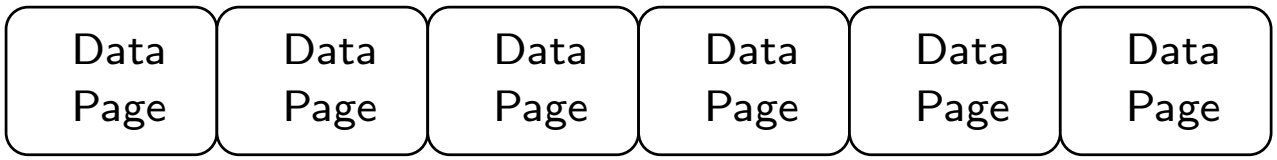
- Almacenar información del atributo en una o varias páginas.
- Guardar puntero a página en el atributo BLOB/CLOB.

Heapfiles: varias aristas del problema

1. ¿cómo representamos una tupla? ✓
2. ¿cómo almacenamos varias tuplas en una página? ✓
3. ¿cómo almacenamos una relación en varias páginas?

Heap file: Implementación 1

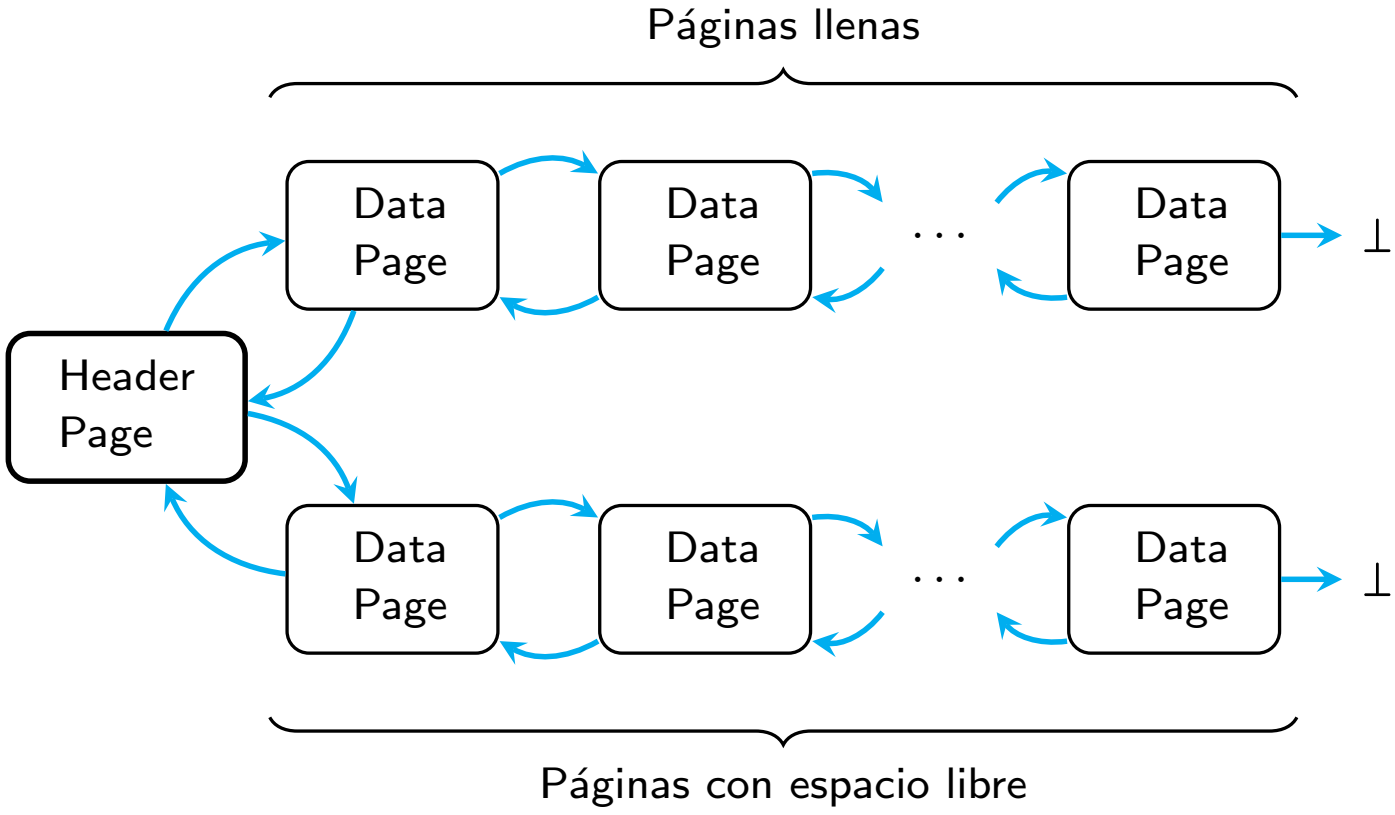
Páginas consecutivas:



¿Problemas?:

- Requiere un espacio contiguo en disco.
- Fragmentación de espacio en las páginas.
- Es necesario buscar en todas las páginas para encontrar espacio.

Heap file: Implementación 2

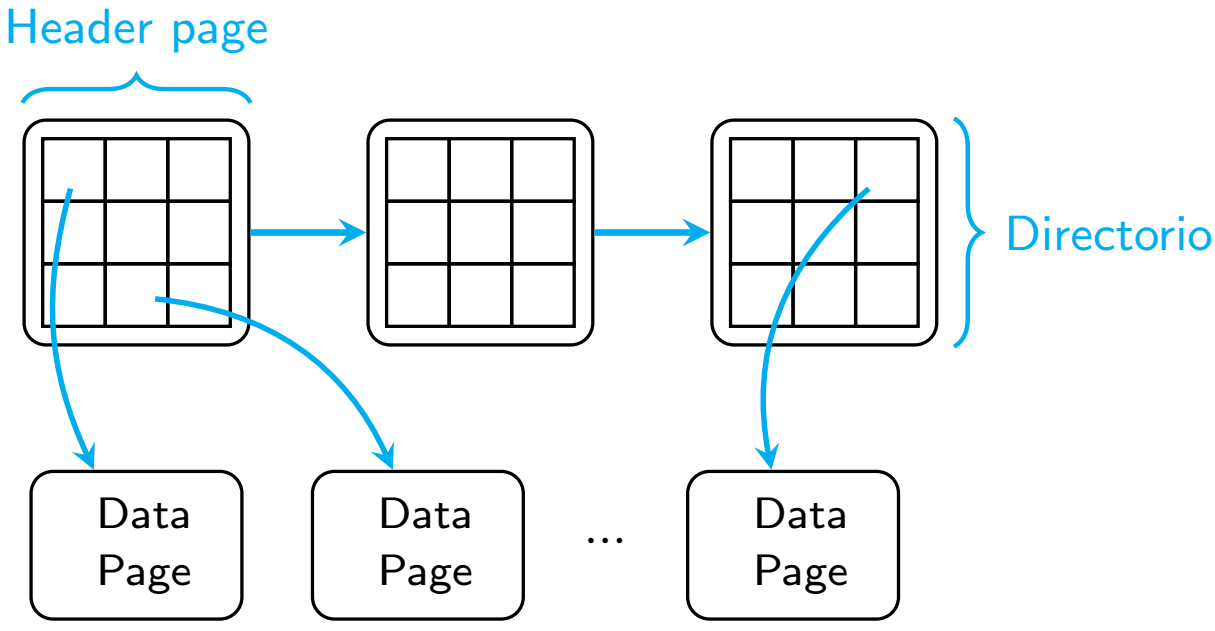


¿Problemas?

- Todavía es necesario buscar en todas las páginas vacías por espacio para una tupla.

Heap file: Implementación 3

Directorio de páginas:



Ventajas:

- Directorio contiene meta-información sobre las páginas (espacio libre).
- Mayor eficiencia en la busqueda de espacio libre.

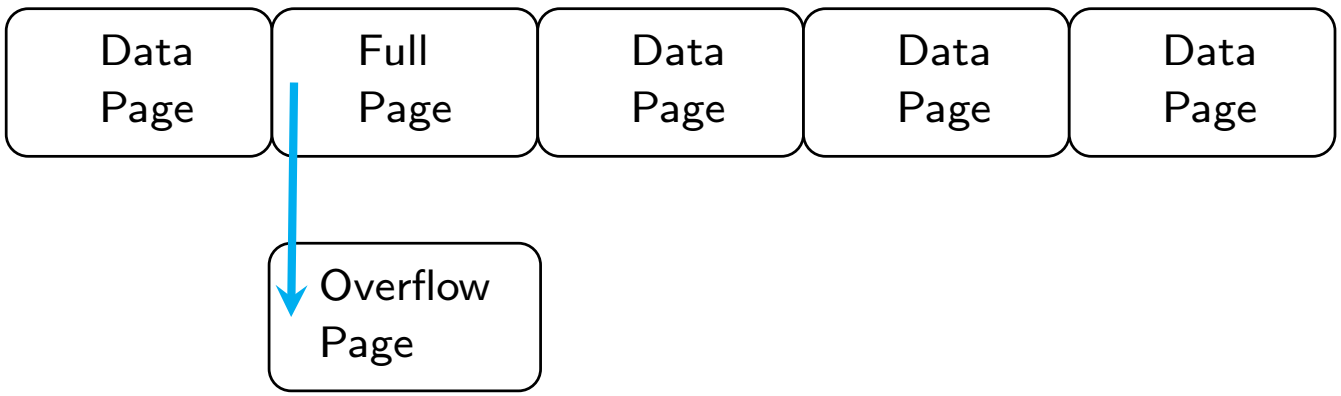
Heap file: modificaciones

Heapfiles no ordenados:

- Insertar, modificar o borrar tuplas es sencillo.

Heapfiles ordenados:

- Para insertar o modificar tuplas: usar **overflow pages**.



Outline

Heap files

Alternativas

Alternativas de almacenamiento

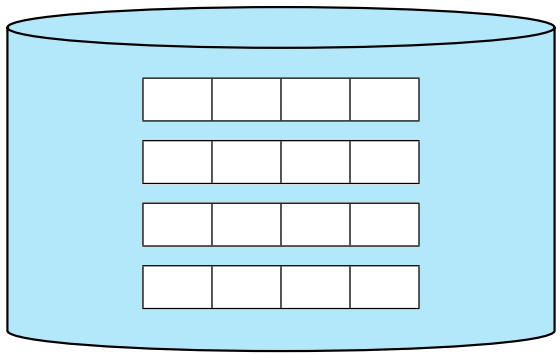
Hasta el momento hemos asumido que la información se guarda en filas.

¿qué sucedería si almacenamos la información por columnas?

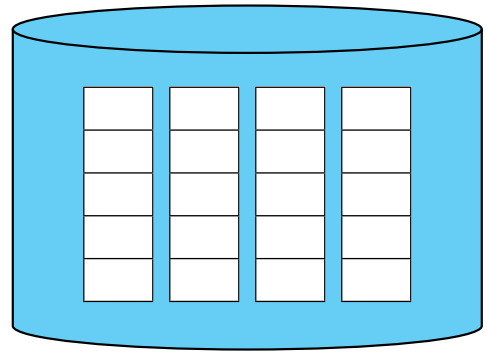
BD orientadas a columnas

Column-oriented DBMS

Basado en filas



Basado en columnas



BD orientadas a columnas

Ventajas de base de datos orientadas a **columnas**:

- Información puede ser compactada.
- Mayor eficiencia en operaciones basadas en columnas.

Ventajas de base de datos orientadas a **filas**:

- Mayor eficiencia en escritura.
- Mayor eficiencia en acceso por tuplas completas.

BD orientadas a columnas

Propuesto en el articulo:

- "C-Store: A column oriented DBMS" por Stonebreaker et all.

Existen sistemas comerciales y académicos en el mercado:

- Vertica (C-store) – inicialmente desarrollado en MIT.
- MonetDB – desarrollado por CWI Amsterdam.