

# Chapter 2

## Storage

Disks, Buffer Manager, Files...

*Architecture and Implementation of Database Systems*  
Summer 2013

Torsten Grust  
Wilhelm-Schickard-Institut für Informatik  
Universität Tübingen



### Storage

Torsten Grust



#### Magnetic Disks

Access Time  
Sequential vs. Random  
Access

#### I/O Parallelism

RAID Levels 1, 0, and 5

#### Alternative Storage Techniques

Solid-State Disks  
Network-Based Storage

#### Managing Space

Free Space Management

#### Buffer Manager

Pinning and Unpinning  
Replacement Policies

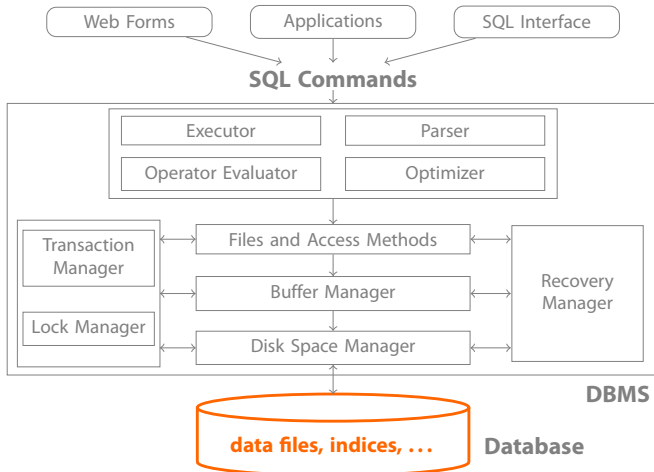
#### Databases vs. Operating Systems

#### Files and Records

Heap Files  
Free Space Management  
Inside a Page  
Alternative Page Layouts

#### Recap

# Database Architecture



## Storage

Torsten Grust



## Magnetic Disks

Access Time  
Sequential vs. Random Access

## I/O Parallelism

RAID Levels 1, 0, and 5

## Alternative Storage Techniques

Solid-State Disks  
Network-Based Storage

## Managing Space

Free Space Management

## Buffer Manager

Pinning and Unpinning  
Replacement Policies

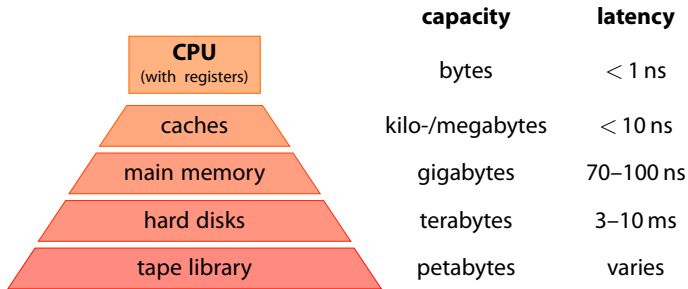
## Databases vs. Operating Systems

## Files and Records

Heap Files  
Free Space Management  
Inside a Page  
Alternative Page Layouts

## Recap

## The Memory Hierarchy



- Fast—but expensive and small—memory close to CPU
- Larger, slower memory at the periphery
- DBMSs try to **hide latency** by using the fast memory as a **cache**.

### Storage

Torsten Grust



### Magnetic Disks

Access Time  
Sequential vs. Random  
Access

### I/O Parallelism

RAID Levels 1, 0, and 5

### Alternative Storage Techniques

Solid-State Disks  
Network-Based Storage

### Managing Space

Free Space Management

### Buffer Manager

Pinning and Unpinning  
Replacement Policies

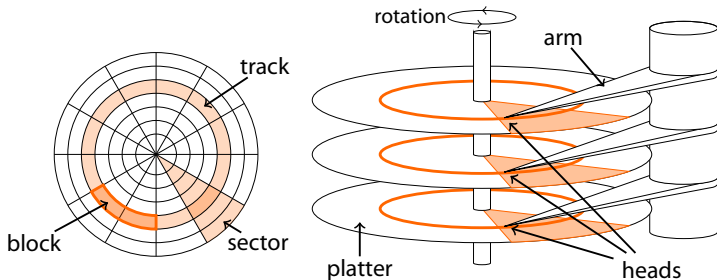
### Databases vs. Operating Systems

### Files and Records

Heap Files  
Free Space Management  
Inside a Page  
Alternative Page Layouts

### Recap

## Magnetic Disks



- A stepper motor positions an array of disk heads on the requested track
- Platters (disks) steadily rotate
- Disks are managed in blocks: the system reads/writes data one block at a time



### Storage

Torsten Grust



### Magnetic Disks

Access Time

Sequential vs. Random Access

### I/O Parallelism

RAID Levels 1, 0, and 5

### Alternative Storage Techniques

Solid-State Disks

Network-Based Storage

### Managing Space

Free Space Management

### Buffer Manager

Pinning and Unpinning

Replacement Policies

### Databases vs. Operating Systems

### Files and Records

Heap Files

Free Space Management

Inside a Page

Alternative Page Layouts

### Recap

## Access Time

*Data blocks can only be read and written if disk heads and platters are positioned accordingly.*

- This design has implications on the **access time** to read/write a given block:

### Definition (Access Time)

- 1 Move disk arms to desired track (**seek time**  $t_s$ )
- 2 Disk controller waits for desired block to rotate under disk head (**rotational delay**  $t_r$ )
- 3 Read/write data (**transfer time**  $t_{tr}$ )

$$\Rightarrow \text{access time: } t = t_s + t_r + t_{tr}$$

### Storage

Torsten Grust



### Magnetic Disks

#### Access Time

Sequential vs. Random Access

#### I/O Parallelism

RAID Levels 1, 0, and 5

#### Alternative Storage Techniques

Solid-State Disks

Network-Based Storage

#### Managing Space

Free Space Management

#### Buffer Manager

Pinning and Unpinning

Replacement Policies

#### Databases vs. Operating Systems

#### Files and Records

Heap Files

Free Space Management

Inside a Page

Alternative Page Layouts

#### Recap

## Example: Seagate Cheetah 15K.7 (600 GB, server-class drive)

- Seagate Cheetah 15K.7 performance characteristics:
  - 4 disks, 8 heads, avg. 512 kB/track, 600 GB capacity
  - rotational speed: 15 000 rpm (revolutions per minute)
  - average seek time: 3.4 ms
  - transfer rate  $\approx$  163 MB/s



What is the access time to read an 8 KB data block?

### Storage

Torsten Grust



### Magnetic Disks

#### Access Time

Sequential vs. Random Access

#### I/O Parallelism

RAID Levels 1, 0, and 5

#### Alternative Storage Techniques

Solid-State Disks

Network-Based Storage

#### Managing Space

Free Space Management

#### Buffer Manager

Pinning and Unpinning

Replacement Policies

#### Databases vs. Operating Systems

#### Files and Records

Heap Files

Free Space Management

Inside a Page

Alternative Page Layouts

#### Recap

## Example: Seagate Cheetah 15K.7 (600 GB, server-class drive)

- Seagate Cheetah 15K.7 performance characteristics:
  - 4 disks, 8 heads, avg. 512 kB/track, 600 GB capacity
  - rotational speed: 15 000 rpm (revolutions per minute)
  - average seek time: 3.4 ms
  - transfer rate  $\approx$  163 MB/s



### What is the access time to read an 8 KB data block?

average seek time  $t_s = 3.40$  ms

average rotational delay:  $\frac{1}{2} \cdot \frac{1}{15\,000 \text{ min}^{-1}}$   $t_r = 2.00$  ms

transfer time for 8 KB:  $\frac{8 \text{ kB}}{163 \text{ MB/s}}$   $t_{tr} = 0.05$  ms

**access time** for an 8 kB data block  $t = 5.45$  ms

#### Storage

Torsten Grust



#### Magnetic Disks

##### Access Time

Sequential vs. Random Access

##### I/O Parallelism

RAID Levels 1, 0, and 5

##### Alternative Storage Techniques

Solid-State Disks

Network-Based Storage

##### Managing Space

Free Space Management

##### Buffer Manager

Pinning and Unpinning

Replacement Policies

##### Databases vs. Operating Systems

##### Files and Records

Heap Files

Free Space Management

Inside a Page

Alternative Page Layouts

##### Recap

## Sequential vs. Random Access

### Example (Read 1 000 blocks of size 8 kB)

- **random access:**

$$t_{\text{rnd}} = 1\,000 \cdot 5.45 \text{ ms} = 5.45 \text{ s}$$

- **sequential read of adjacent blocks:**

$$\begin{aligned} t_{\text{seq}} &= t_s + t_r + 1\,000 \cdot t_{tr} + 16 \cdot t_{s,\text{track-to-track}} \\ &= 3.40 \text{ ms} + 2.00 \text{ ms} + 50 \text{ ms} + 3.2 \text{ ms} \approx 58.6 \text{ ms} \end{aligned}$$

The Seagate Cheetah 15K.7 stores an average of 512 kB per track, with a 0.2 ms track-to-track seek time; our 8 kB blocks are spread across 16 tracks.

- ⇒ Sequential I/O is **much** faster than random I/O
- ⇒ **Avoid random I/O** whenever possible
- ⇒ As soon as we need at least  $\frac{58.6 \text{ ms}}{5,450 \text{ ms}} = 1.07\%$  of a file, we better read the **entire** file sequentially



### Storage

Torsten Grust



### Magnetic Disks

Access Time

Sequential vs. Random Access

### I/O Parallelism

RAID Levels 1, 0, and 5

### Alternative Storage Techniques

Solid-State Disks

Network-Based Storage

### Managing Space

Free Space Management

### Buffer Manager

Pinning and Unpinning

Replacement Policies

### Databases vs. Operating Systems

### Files and Records

Heap Files

Free Space Management

Inside a Page

Alternative Page Layouts

### Recap

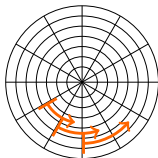


## Performance Tricks

- Disk manufacturers play a number of tricks to improve performance:

### track skewing

Align sector 0 of each track to avoid rotational delay during longer sequential scans



### request scheduling

If **multiple requests** have to be served, choose the one that requires the smallest arm movement (SPTF: shortest positioning time first, elevator algorithms)

### zoning

Outer tracks are longer than the inner ones. Therefore, divide outer tracks into more sectors than inner tracks

## Storage

Torsten Grust



### Magnetic Disks

Access Time

Sequential vs. Random Access

### I/O Parallelism

RAID Levels 1, 0, and 5

### Alternative Storage Techniques

Solid-State Disks

Network-Based Storage

### Managing Space

Free Space Management

### Buffer Manager

Pinning and Unpinning

Replacement Policies

### Databases vs. Operating Systems

### Files and Records

Heap Files

Free Space Management

Inside a Page

Alternative Page Layouts

### Recap

## Evolution of Hard Disk Technology

Disk seek and rotational latencies have only marginally improved over the last years ( $\approx 10\%$  per year)

### But:

- Throughput (i.e., transfer rates) improve by  $\approx 50\%$  per year
- Hard disk capacity grows by  $\approx 50\%$  every year

### Therefore:

- Random access cost hurts even more as time progresses

### Example (5 Years Ago: Seagate Barracuda 7200.7)

Read 1K blocks of 8 kB sequentially/randomly: 397 ms / 12 800 ms

#### Storage

Torsten Grust



#### Magnetic Disks

Access Time

Sequential vs. Random Access

#### I/O Parallelism

RAID Levels 1, 0, and 5

#### Alternative Storage Techniques

Solid-State Disks

Network-Based Storage

#### Managing Space

Free Space Management

#### Buffer Manager

Pinning and Unpinning

Replacement Policies

#### Databases vs. Operating Systems

#### Files and Records

Heap Files

Free Space Management

Inside a Page

Alternative Page Layouts

#### Recap

## Ways to Improve I/O Performance

The latency penalty is hard to avoid

### But:

- Throughput can be increased rather easily by exploiting **parallelism**
- **Idea:** Use multiple disks and access them in parallel, try to hide latency

### DB2<sup>®</sup> TPC-C: An industry benchmark for OLTP

A recent #1 system (IBM DB2 9.5 on AIX) uses

- 10,992 disk drives (73.4 GB each, 15,000 rpm) (!)  
plus 8 146.8 GB internal SCSI drives,
- connected with 68 4 Gbit fibre channel adapters,
- yielding 6 mio transactions per minute

#### Storage

Torsten Grust



#### Magnetic Disks

Access Time

Sequential vs. Random  
Access

#### I/O Parallelism

RAID Levels 1, 0, and 5

#### Alternative Storage Techniques

Solid-State Disks

Network-Based Storage

#### Managing Space

Free Space Management

#### Buffer Manager

Pinning and Unpinning

Replacement Policies

#### Databases vs. Operating Systems

#### Files and Records

Heap Files

Free Space Management

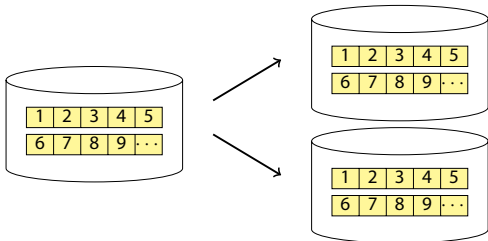
Inside a Page

Alternative Page Layouts

#### Recap

## Disk Mirroring

- **Replicate** data onto multiple disks:



- Achieves I/O parallelism only for **reads**
- Improved failure tolerance—can survive one disk failure
- This is also known as **RAID 1** (mirroring without parity) (**RAID**: Redundant Array of Inexpensive Disks)

### Storage

Torsten Grust



### Magnetic Disks

Access Time

Sequential vs. Random Access

### I/O Parallelism

RAID Levels 1, 0, and 5

### Alternative Storage Techniques

Solid-State Disks

Network-Based Storage

### Managing Space

Free Space Management

### Buffer Manager

Pinning and Unpinning

Replacement Policies

### Databases vs. Operating Systems

### Files and Records

Heap Files

Free Space Management

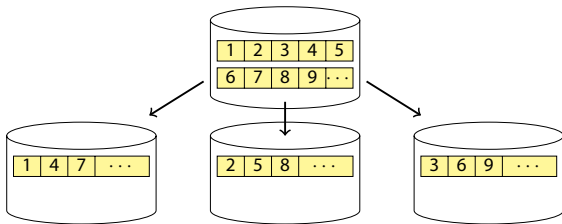
Inside a Page

Alternative Page Layouts

### Recap

## Disk Striping

- **Distribute** data over disks:



- Full I/O parallelism for **read and write** operations
- High failure risk (here: 3 times risk of single disk failure)!
- Also known as **RAID 0** (striping without parity)

### Storage

Torsten Grust



### Magnetic Disks

Access Time

Sequential vs. Random  
Access

### I/O Parallelism

RAID Levels 1, 0, and 5

### Alternative Storage Techniques

Solid-State Disks  
Network-Based Storage

### Managing Space

Free Space Management

### Buffer Manager

Pinning and Unpinning  
Replacement Policies

### Databases vs. Operating Systems

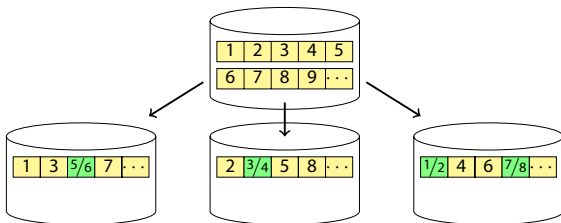
### Files and Records

Heap Files  
Free Space Management  
Inside a Page  
Alternative Page Layouts

### Recap

## Disk Striping with Parity

- **Distribute** data and **parity** information over  $\geq 3$  disks:



- High I/O parallelism
- Fault tolerance: any **one disk may fail** without data loss (with dual parity/RAID 6: two disks may fail)
- Distribute parity (e.g., XOR) information over disks, separating data and associated parity
- Also known as **RAID 5** (striping with distributed parity)

### Storage

Torsten Grust



### Magnetic Disks

Access Time

Sequential vs. Random Access

### I/O Parallelism

RAID Levels 1, 0, and 5

### Alternative Storage Techniques

Solid-State Disks

Network-Based Storage

### Managing Space

Free Space Management

### Buffer Manager

Pinning and Unpinning

Replacement Policies

### Databases vs. Operating Systems

### Files and Records

Heap Files

Free Space Management

Inside a Page

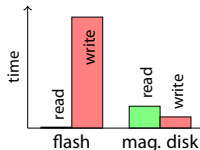
Alternative Page Layouts

### Recap

## Solid-State Disks

Solid state disks (SSDs) have emerged as an alternative to conventional hard disks

- SSDs provide **very low-latency random read access** ( $< 0.01$  ms)
- Random writes**, however, are significantly **slower** than on traditional magnetic drives:
  - (Blocks of) Pages have to be **erased** before they can be updated
  - Once pages have been erased, sequentially writing them is almost as fast as reading



### Storage

Torsten Grust



### Magnetic Disks

Access Time

Sequential vs. Random Access

### I/O Parallelism

RAID Levels 1, 0, and 5

### Alternative Storage Techniques

Solid-State Disks

Network-Based Storage

### Managing Space

Free Space Management

### Buffer Manager

Pinning and Unpinning  
Replacement Policies

### Databases vs. Operating Systems

### Files and Records

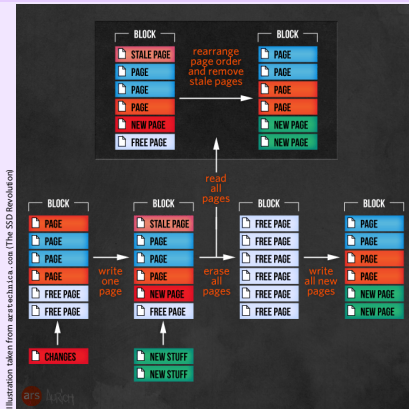
Heap Files  
Free Space Management  
Inside a Page  
Alternative Page Layouts

### Recap

## SSDs: Page-Level Writes, Block-Level Deletes

- Typical **page size**: 128 kB
- SSDs erase **blocks of pages**: block  $\approx$  64 pages (8 MB)

### Example (Perform block-level delete to accommodate new data pages)



#### Storage

Torsten Grust



#### Magnetic Disks

Access Time  
Sequential vs. Random  
Access

#### I/O Parallelism

RAID Levels 1, 0, and 5

#### Alternative Storage Techniques

Solid-State Disks

Network-Based Storage

#### Managing Space

Free Space Management

#### Buffer Manager

Pinning and Unpinning  
Replacement Policies

#### Databases vs. Operating Systems

#### Files and Records

Heap Files  
Free Space Management  
Inside a Page  
Alternative Page Layouts

#### Recap



## Example: Seagate Pulsar.2 (800 GB, server-class solid state drive)

- Seagate Pulsar.2 performance characteristics:
  - NAND flash memory, 800 GB capacity
  - standard 2.5" enclosure, no moving/rotating parts
  - data read/written in pages of 128 kB size
  - transfer rate  $\approx$  370 MB/s



What is the access time to read an 8 KB data block?

### Storage

Torsten Grust



### Magnetic Disks

Access Time

Sequential vs. Random  
Access

### I/O Parallelism

RAID Levels 1, 0, and 5

### Alternative Storage Techniques

Solid-State Disks

Network-Based Storage

### Managing Space

Free Space Management

### Buffer Manager

Pinning and Unpinning  
Replacement Policies

### Databases vs. Operating Systems

### Files and Records

Heap Files  
Free Space Management  
Inside a Page  
Alternative Page Layouts

### Recap

## Example: Seagate Pulsar.2 (800 GB, server-class solid state drive)

- Seagate Pulsar.2 performance characteristics:
  - NAND flash memory, 800 GB capacity
  - standard 2.5" enclosure, no moving/rotating parts
  - data read/written in pages of 128 kB size
  - transfer rate  $\approx 370$  MB/s

### What is the access time to read an 8 KB data block?

no seek time  $t_s = 0.00$  ms

no rotational delay:  $t_r = 0.00$  ms

transfer time for 8 KB:  $\frac{128 \text{ kB}}{370 \text{ MB/s}}$   $t_{tr} = 0.30$  ms

**access time** for an 8 kB data block  $t = 0.30$  ms

#### Storage

Torsten Grust



#### Magnetic Disks

Access Time

Sequential vs. Random  
Access

#### I/O Parallelism

RAID Levels 1, 0, and 5

#### Alternative Storage Techniques

Solid-State Disks

Network-Based Storage

#### Managing Space

Free Space Management

#### Buffer Manager

Pinning and Unpinning  
Replacement Policies

#### Databases vs. Operating Systems

#### Files and Records

Heap Files  
Free Space Management  
Inside a Page  
Alternative Page Layouts

#### Recap

## Sequential vs. Random Access with SSDs

### Example (Read 1 000 blocks of size 8 kB)

- **random access:**

$$t_{\text{rnd}} = 1\,000 \cdot 0.30 \text{ ms} = 0.3 \text{ s}$$

- **sequential read of adjacent pages:**

$$t_{\text{seq}} = \left\lceil \frac{1\,000 \cdot 8 \text{ kB}}{128 \text{ kB}} \right\rceil \cdot t_{\text{tr}} \approx 18.9 \text{ ms}$$

The Seagate Pulsar.2 (sequentially) reads data in 128 kB chunks.

- ⇒ Sequential I/O still beats random I/O (but random I/O is more feasible again)
- Adapting database technology to these characteristics is a current research topic

#### Storage

Torsten Grust



#### Magnetic Disks

Access Time  
Sequential vs. Random Access

#### I/O Parallelism

RAID Levels 1, 0, and 5

#### Alternative Storage Techniques

Solid-State Disks

Network-Based Storage

#### Managing Space

Free Space Management

#### Buffer Manager

Pinning and Unpinning  
Replacement Policies

#### Databases vs. Operating Systems

#### Files and Records

Heap Files  
Free Space Management  
Inside a Page  
Alternative Page Layouts

#### Recap

## Network-Based Storage

Today the network is **not** a bottleneck any more:

Storage media/interface	Transfer rate
Hard disk	100–200 MB/s
Serial ATA	375 MB/s
Ultra-640 SCSI	640 MB/s
10-Gbit Ethernet	1,250 MB/s
Infiniband QDR	12,000 MB/s
For comparison (RAM):	
PC2–5300 DDR2–SDRAM	10.6 GB/s
PC3–12800 DDR3–SDRAM	25.6 GB/s

⇒ Why not use the network for database storage?

### Storage

Torsten Grust



### Magnetic Disks

Access Time

Sequential vs. Random  
Access

### I/O Parallelism

RAID Levels 1, 0, and 5

### Alternative Storage Techniques

Solid-State Disks

Network-Based Storage

### Managing Space

Free Space Management

### Buffer Manager

Pinning and Unpinning  
Replacement Policies

### Databases vs. Operating Systems

### Files and Records

Heap Files  
Free Space Management  
Inside a Page  
Alternative Page Layouts

### Recap

## Storage Area Network (SAN)

- **Block-based** network access to storage:
  - SAN emulate interface of block-structured disks (*"read block #42 of disk 10"*)
  - This is unlike network file systems (e.g., NFS, CIFS)
- SAN storage devices typically abstract from RAID or physical disks and present logical drives to the DBMS
  - Hardware acceleration and simplified maintainability
- Typical setup: local area network with multiple participating servers and storage resources
  - Failure tolerance and increased flexibility

### Storage

Torsten Grust



### Magnetic Disks

Access Time

Sequential vs. Random  
Access

### I/O Parallelism

RAID Levels 1, 0, and 5

### Alternative Storage Techniques

Solid-State Disks

### Network-Based Storage

### Managing Space

Free Space Management

### Buffer Manager

Pinning and Unpinning  
Replacement Policies

### Databases vs. Operating Systems

### Files and Records

Heap Files  
Free Space Management  
Inside a Page  
Alternative Page Layouts

### Recap

## Grid or Cloud Storage

Internet-scale enterprises employ clusters with 1000s commodity PCs (e.g., Amazon, Google, Yahoo!):

- **system cost**  $\leftrightarrow$  **reliability** and **performance**,
- use **massive replication** for data storage

Spare CPU cycles and disk space are sold as a **service**:

### Amazon's Elastic Computing Cloud (EC2)

Use Linux-based compute cluster by the hour ( $\sim 10$  ¢/h).

### Amazon's Simple Storage System (S3)

"Infinite" store for objects between 1 B and 5 TB in size, organized in a map data structure (key  $\mapsto$  object)

- Latency: 100 ms to 1 s (not impacted by load)
- pricing  $\approx$  disk drives (but addl. cost for access)

$\Rightarrow$  Building a database on S3?  
( $\nearrow$  Brantner *et al.*, SIGMOD 2008)

#### Storage

Torsten Grust



#### Magnetic Disks

Access Time

Sequential vs. Random Access

#### I/O Parallelism

RAID Levels 1, 0, and 5

#### Alternative Storage Techniques

Solid-State Disks

Network-Based Storage

#### Managing Space

Free Space Management

#### Buffer Manager

Pinning and Unpinning  
Replacement Policies

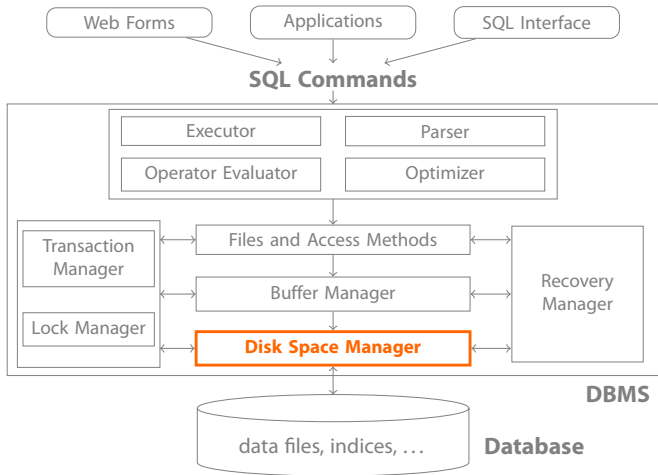
#### Databases vs. Operating Systems

#### Files and Records

Heap Files  
Free Space Management  
Inside a Page  
Alternative Page Layouts

#### Recap

## Managing Space



### Storage

Torsten Grust



### Magnetic Disks

Access Time  
Sequential vs. Random  
Access

### I/O Parallelism

RAID Levels 1, 0, and 5

### Alternative Storage Techniques

Solid-State Disks  
Network-Based Storage

### Managing Space

Free Space Management

### Buffer Manager

Pinning and Unpinning  
Replacement Policies

### Databases vs. Operating Systems

### Files and Records

Heap Files  
Free Space Management  
Inside a Page  
Alternative Page Layouts

### Recap

## Managing Space

### Definition (Disk Space Manager)

- Abstracts from the gory details of the underlying storage (disk space manager talks to I/O controller and initiates I/O)
- DBMS issues allocate/deallocate and read/write commands
- Provides the concept of a **page** (typically 4–64 KB) as a unit of storage to the remaining system components
- Maintains a locality-preserving mapping

page number  $\mapsto$  physical location ,

where a physical location could be, *e.g.*,

- an OS file name and an offset within that file,
- head, sector, and track of a hard drive, or
- tape number and offset for data stored in a tape library

### Storage

Torsten Grust



### Magnetic Disks

Access Time

Sequential vs. Random Access

### I/O Parallelism

RAID Levels 1, 0, and 5

### Alternative Storage Techniques

Solid-State Disks

Network-Based Storage

### Managing Space

Free Space Management

### Buffer Manager

Pinning and Unpinning

Replacement Policies

### Databases vs. Operating Systems

### Files and Records

Heap Files

Free Space Management

Inside a Page

Alternative Page Layouts

### Recap



## Empty Pages

The disk space manager also keeps track of **used/free blocks** (deallocation and subsequent allocation may create **holes**):

- 1 Maintain a **linked list** of free pages
  - When a page is no longer needed, add it to the list
- 2 Maintain a **bitmap** reserving one bit for each page
  - Toggle bit  $n$  when page  $n$  is (de-)allocated

### Allocation of contiguous pages

To exploit **sequential access**, it is useful to allocate **contiguous** sequences of pages.

Which of the techniques (1 or 2) would you choose to support this?

#### Storage

Torsten Grust



#### Magnetic Disks

Access Time

Sequential vs. Random Access

#### I/O Parallelism

RAID Levels 1, 0, and 5

#### Alternative Storage Techniques

Solid-State Disks

Network-Based Storage

#### Managing Space

Free Space Management

#### Buffer Manager

Pinning and Unpinning

Replacement Policies

#### Databases vs. Operating Systems

#### Files and Records

Heap Files

Free Space Management

Inside a Page

Alternative Page Layouts

#### Recap

## Empty Pages

The disk space manager also keeps track of **used/free blocks** (deallocation and subsequent allocation may create **holes**):

- 1 Maintain a **linked list** of free pages
  - When a page is no longer needed, add it to the list
- 2 Maintain a **bitmap** reserving one bit for each page
  - Toggle bit  $n$  when page  $n$  is (de-)allocated

### Allocation of contiguous pages

To exploit **sequential access**, it is useful to allocate **contiguous** sequences of pages.

Which of the techniques (1 or 2) would you choose to support this?

This is a lot easier to do with a free page bitmap (option 2).

#### Storage

Torsten Grust



#### Magnetic Disks

Access Time

Sequential vs. Random Access

#### I/O Parallelism

RAID Levels 1, 0, and 5

#### Alternative Storage Techniques

Solid-State Disks

Network-Based Storage

#### Managing Space

Free Space Management

#### Buffer Manager

Pinning and Unpinning

Replacement Policies

#### Databases vs. Operating Systems

#### Files and Records

Heap Files

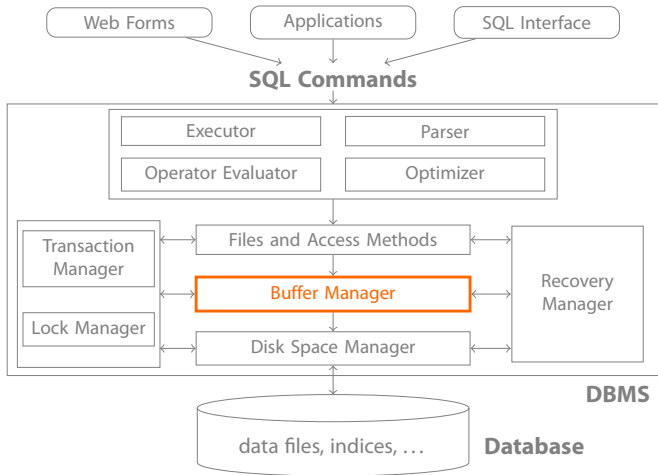
Free Space Management

Inside a Page

Alternative Page Layouts

#### Recap

## Buffer Manager



### Storage

Torsten Grust



### Magnetic Disks

Access Time  
Sequential vs. Random  
Access

### I/O Parallelism

RAID Levels 1, 0, and 5

### Alternative Storage Techniques

Solid-State Disks  
Network-Based Storage

### Managing Space

Free Space Management

### Buffer Manager

Pinning and Unpinning  
Replacement Policies

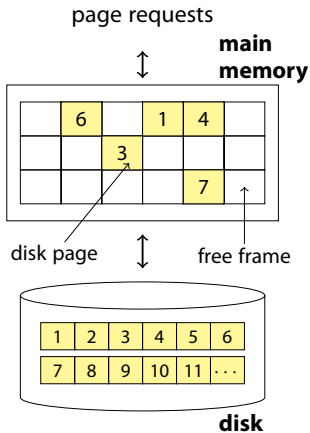
### Databases vs. Operating Systems

### Files and Records

Heap Files  
Free Space Management  
Inside a Page  
Alternative Page Layouts

### Recap

## Buffer Manager



### Definition (Buffer Manager)

- Mediates between external storage and main memory,
- Manages a designated main memory area, the **buffer pool**, for this task

Disk pages are brought into memory as needed and loaded into memory **frames**

A **replacement policy** decides which page to evict when the buffer is full

### Storage

Torsten Grust



### Magnetic Disks

Access Time  
Sequential vs. Random Access

### I/O Parallelism

RAID Levels 1, 0, and 5

### Alternative Storage Techniques

Solid-State Disks  
Network-Based Storage

### Managing Space

Free Space Management

### Buffer Manager

Pinning and Unpinning  
Replacement Policies

### Databases vs. Operating Systems

### Files and Records

Heap Files  
Free Space Management  
Inside a Page  
Alternative Page Layouts

### Recap

## Interface to the Buffer Manager

Higher-level code requests (pins) pages from the buffer manager and releases (unpins) pages after use.

### *pin* (*pageno*)

Request page number *pageno* from the buffer manager, load it into memory if necessary and mark page as clean (*-dirty*). Returns a reference to the frame containing *pageno*.

### *unpin* (*pageno*, *dirty*)

Release page number *pageno*, making it a candidate for eviction. Must set *dirty* = true if page was modified.

### Why do we need the *dirty* bit?

#### Storage

Torsten Grust



#### Magnetic Disks

Access Time

Sequential vs. Random Access

#### I/O Parallelism

RAID Levels 1, 0, and 5

#### Alternative Storage Techniques

Solid-State Disks

Network-Based Storage

#### Managing Space

Free Space Management

#### Buffer Manager

Pinning and Unpinning

Replacement Policies

#### Databases vs. Operating Systems

#### Files and Records

Heap Files

Free Space Management

Inside a Page

Alternative Page Layouts

#### Recap

## Interface to the Buffer Manager

Higher-level code requests (pins) pages from the buffer manager and releases (unpins) pages after use.

### *pin* (*pageno*)

Request page number *pageno* from the buffer manager, load it into memory if necessary and mark page as clean ( $\neg$ *dirty*). Returns a reference to the frame containing *pageno*.

### *unpin* (*pageno*, *dirty*)

Release page number *pageno*, making it a candidate for eviction. Must set *dirty* = true if page was modified.

### Why do we need the *dirty* bit?

Only **modified** pages need to be written back to disk upon eviction.

#### Storage

Torsten Grust



#### Magnetic Disks

Access Time  
Sequential vs. Random Access

#### I/O Parallelism

RAID Levels 1, 0, and 5

#### Alternative Storage Techniques

Solid-State Disks  
Network-Based Storage

#### Managing Space

Free Space Management

#### Buffer Manager

Pinning and Unpinning  
Replacement Policies

#### Databases vs. Operating Systems

#### Files and Records

Heap Files  
Free Space Management  
Inside a Page  
Alternative Page Layouts

#### Recap

## Proper pin ()/unpin () Nesting

- Any database transaction is required to properly “bracket” any page operation using pin () and unpin () calls:

### A read-only transaction

```
a ← pin (p);  
{  
  ∴  
  read data on page at memory address a;  
  ∴  
}  
unpin (p, false);
```

- Proper bracketing enables the systems to keep a count of active users (e.g., transactions) of a page

#### Storage

Torsten Grust



#### Magnetic Disks

Access Time  
Sequential vs. Random  
Access

#### I/O Parallelism

RAID Levels 1, 0, and 5

#### Alternative Storage Techniques

Solid-State Disks  
Network-Based Storage

#### Managing Space

Free Space Management

#### Buffer Manager

Pinning and Unpinning  
Replacement Policies

#### Databases vs. Operating Systems

#### Files and Records

Heap Files  
Free Space Management  
Inside a Page  
Alternative Page Layouts

#### Recap

## Implementation of `pin()`

### Function `pin(pageno)`

```
1 if buffer pool already contains pageno then
2   | pinCount(pageno) ← pinCount(pageno) + 1;
3   | return address of frame holding pageno;
4 else
5   | select a victim frame v using the replacement policy;
6   | if dirty(page in v) then
7   |   | write page in v to disk;
8   |   | read page pageno from disk into frame v;
9   |   | pinCount(pageno) ← 1;
10  |   | dirty(pageno) ← false;
11  |   | return address of frame v;
```

#### Storage

Torsten Grust



#### Magnetic Disks

Access Time  
Sequential vs. Random  
Access

#### I/O Parallelism

RAID Levels 1, 0, and 5

#### Alternative Storage Techniques

Solid-State Disks  
Network-Based Storage

#### Managing Space

Free Space Management

#### Buffer Manager

Pinning and Unpinning  
Replacement Policies

#### Databases vs. Operating Systems

#### Files and Records

Heap Files  
Free Space Management  
Inside a Page  
Alternative Page Layouts

#### Recap



## Implementation of `unpin()`

### Function `unpin(pageno, dirty)`

```
1 pinCount(pageno) ← pinCount(pageno) - 1;  
2 dirty(pageno) ← dirty(pageno) ∨ dirty;
```

### Why don't we write pages back to disk during `unpin()`?

#### Storage

Torsten Grust



#### Magnetic Disks

Access Time

Sequential vs. Random Access

#### I/O Parallelism

RAID Levels 1, 0, and 5

#### Alternative Storage Techniques

Solid-State Disks

Network-Based Storage

#### Managing Space

Free Space Management

#### Buffer Manager

Pinning and Unpinning

Replacement Policies

#### Databases vs. Operating Systems

#### Files and Records

Heap Files

Free Space Management

Inside a Page

Alternative Page Layouts

#### Recap

## Implementation of `unpin()`

### Function `unpin(pageno, dirty)`

```
1 pinCount(pageno) ← pinCount(pageno) - 1 ;  
2 dirty(pageno) ← dirty(pageno) ∨ dirty ;
```

### Why don't we write pages back to disk during `unpin()`?

Well, we could ...

- + recovery from failure would be a lot simpler
- higher I/O cost (every page write implies a write to disk)
- bad response time for writing transactions

This discussion is also known as **force** (or **write-through**) vs. **write-back**. Actual database systems typically implement write-back.

#### Storage

Torsten Grust



#### Magnetic Disks

Access Time

Sequential vs. Random Access

#### I/O Parallelism

RAID Levels 1, 0, and 5

#### Alternative Storage Techniques

Solid-State Disks

Network-Based Storage

#### Managing Space

Free Space Management

#### Buffer Manager

Pinning and Unpinning

Replacement Policies

#### Databases vs. Operating Systems

#### Files and Records

Heap Files

Free Space Management

Inside a Page

Alternative Page Layouts

#### Recap

## Concurrent Writes?

### Conflicting changes to a block

Assume the following:

- 1 The same page  $p$  is requested by more than one transaction (i.e.,  $\text{pinCount}(p) > 1$ ), and
- 2 ... those transactions perform conflicting writes on  $p$ ?

Conflicts of this kind are resolved by the system's **concurrency control**, a layer on top of the buffer manager (see "Introduction to Database Systems", transaction management, locks).

The buffer manager may assume that everything is in order whenever it receives an `unpin( $p$ , true)` call.

#### Storage

Torsten Grust



#### Magnetic Disks

Access Time  
Sequential vs. Random Access

#### I/O Parallelism

RAID Levels 1, 0, and 5

#### Alternative Storage Techniques

Solid-State Disks  
Network-Based Storage

#### Managing Space

Free Space Management

#### Buffer Manager

Pinning and Unpinning  
Replacement Policies

#### Databases vs. Operating Systems

#### Files and Records

Heap Files  
Free Space Management  
Inside a Page  
Alternative Page Layouts

#### Recap

## Replacement Policies

The effectiveness of the buffer manager's **caching** functionality can depend on the **replacement policy** it uses, *e.g.*,

### Least Recently Used (LRU)

Evict the page whose latest unpin () is longest ago

### LRU-*k*

Like LRU, but considers the *k* latest unpin () calls, not just the latest

### Most Recently Used (MRU)

Evict the page that has been unpinned most recently

### Random

Pick a victim randomly

### Rationales behind each of these policies?

#### Storage

Torsten Grust



#### Magnetic Disks

Access Time

Sequential vs. Random Access

#### I/O Parallelism

RAID Levels 1, 0, and 5

#### Alternative Storage Techniques

Solid-State Disks

Network-Based Storage

#### Managing Space

Free Space Management

#### Buffer Manager

Pinning and Unpinning

Replacement Policies

#### Databases vs. Operating Systems

#### Files and Records

Heap Files

Free Space Management

Inside a Page

Alternative Page Layouts

#### Recap

## Example Policy: Clock (“Second Chance”)

- Simulate an LRU policy with less overhead (no LRU queue reorganization on every frame reference):

### Clock (“Second Chance”)

- 1 Number the  $N$  frames in the buffer pool  $0, \dots, N - 1$ ; initialize  $\text{current} \leftarrow 0$ , maintain a bit array  $\text{referenced}[0, \dots, N - 1]$  initialized to all 0
- 2 In  $\text{pin}(p)$ , assign  $\text{referenced}[p] \leftarrow 1$
- 3 To find the next victim, consider page  $\text{current}$ ; if  $\text{referenced}[\text{current}] = 0$ ,  $\text{current}$  is the victim; otherwise,  $\text{referenced}[\text{current}] \leftarrow 0$ ,  $\text{current} \leftarrow \text{current} + 1 \bmod N$ , repeat 3

#### Storage

Torsten Grust



#### Magnetic Disks

Access Time

Sequential vs. Random Access

#### I/O Parallelism

RAID Levels 1, 0, and 5

#### Alternative Storage Techniques

Solid-State Disks

Network-Based Storage

#### Managing Space

Free Space Management

#### Buffer Manager

Pinning and Unpinning

Replacement Policies

#### Databases vs. Operating Systems

#### Files and Records

Heap Files

Free Space Management

Inside a Page

Alternative Page Layouts

#### Recap

## Heuristic Policies Can Fail

The mentioned policies, including LRU, are **heuristics** only and may fail miserably in certain scenarios.



### Example (A Challenge for LRU)

A number of transactions want to scan the same sequence of pages (consider a repeated `SELECT * FROM R`).

Assume a buffer pool capacity of 10 pages.

- 1 Let the size of relation `R` be 10 or less pages.  
How many I/O (actual disk page reads) do you expect?
- 2 Now grow `R` by one page.  
How about the number of I/O operations in this case?

#### Storage

Torsten Grust



#### Magnetic Disks

Access Time

Sequential vs. Random Access

#### I/O Parallelism

RAID Levels 1, 0, and 5

#### Alternative Storage Techniques

Solid-State Disks

Network-Based Storage

#### Managing Space

Free Space Management

#### Buffer Manager

Pinning and Unpinning

Replacement Policies

#### Databases vs. Operating Systems

#### Files and Records

Heap Files

Free Space Management

Inside a Page

Alternative Page Layouts

#### Recap

## More Challenges for LRU

- 1 Transaction  $T_1$  repeatedly accesses a fixed set of pages; transaction  $T_2$  performs a sequential scan of the database pages.
- 2 Assume a B<sup>+</sup>-tree-indexed relation R. R occupies 10,000 data pages  $R_i$ , the B<sup>+</sup>-tree occupies one root node and 100 index leaf nodes  $I_k$ .  
Transactions perform repeated random index key lookups on R  $\Rightarrow$  **page access pattern** (ignores B<sup>+</sup>-tree root node):

$I_1, R_1, I_2, R_2, I_3, R_3, \dots$

 **How will LRU perform in this case?**

### Storage

Torsten Grust



### Magnetic Disks

Access Time  
Sequential vs. Random  
Access

### I/O Parallelism

RAID Levels 1, 0, and 5

### Alternative Storage Techniques

Solid-State Disks  
Network-Based Storage

### Managing Space

Free Space Management

### Buffer Manager

Pinning and Unpinning  
Replacement Policies

### Databases vs. Operating Systems

### Files and Records

Heap Files  
Free Space Management  
Inside a Page  
Alternative Page Layouts

### Recap

## More Challenges for LRU

- 1 Transaction  $T_1$  repeatedly accesses a fixed set of pages; transaction  $T_2$  performs a sequential scan of the database pages.
- 2 Assume a B<sup>+</sup>-tree-indexed relation R. R occupies 10,000 data pages  $R_i$ , the B<sup>+</sup>-tree occupies one root node and 100 index leaf nodes  $I_k$ .  
Transactions perform repeated random index key lookups on R  $\Rightarrow$  **page access pattern** (ignores B<sup>+</sup>-tree root node):

$$I_1, R_1, I_2, R_2, I_3, R_3, \dots$$

### How will LRU perform in this case?

With LRU, 50 % of the buffered pages will be pages of R. However, the probability of re-accessing page  $R_i$  only is  $1/10,000$ .

### Storage

Torsten Grust



### Magnetic Disks

Access Time  
Sequential vs. Random  
Access

### I/O Parallelism

RAID Levels 1, 0, and 5

### Alternative Storage Techniques

Solid-State Disks  
Network-Based Storage

### Managing Space

Free Space Management

### Buffer Manager

Pinning and Unpinning  
Replacement Policies

### Databases vs. Operating Systems

### Files and Records

Heap Files  
Free Space Management  
Inside a Page  
Alternative Page Layouts

### Recap



## Buffer Management in Practice

### Prefetching

Buffer managers try to anticipate page requests to overlap CPU and I/O operations:

- **Speculative prefetching:** Assume sequential scan and automatically read ahead.
- **Prefetch lists:** Some database algorithms can instruct the buffer manager with a list of pages to prefetch.

### Page fixing/hating

**Higher-level code** may request to **fix** a page if it may be useful in the near future (*e.g.*, nested-loop join).

Likewise, an operator that **hates** a page will not access it any time soon (*e.g.*, table pages in a sequential scan).

### Partitioned buffer pools

*E.g.*, maintain separate pools for indexes and tables.

#### Storage

Torsten Grust



#### Magnetic Disks

Access Time

Sequential vs. Random Access

#### I/O Parallelism

RAID Levels 1, 0, and 5

#### Alternative Storage Techniques

Solid-State Disks

Network-Based Storage

#### Managing Space

Free Space Management

#### Buffer Manager

Pinning and Unpinning

Replacement Policies

#### Databases vs. Operating Systems

#### Files and Records

Heap Files

Free Space Management

Inside a Page

Alternative Page Layouts

#### Recap

## Databases vs. Operating Systems

**Wait!** Didn't we just re-invent the operating system?



**Yes,**

- disk space management and buffer management very much look like **file management** and **virtual memory** in OSs.

**But,**

- a DBMS may be much more aware of the **access patterns** of certain operators (prefetching, page fixing/hating),
- concurrency control often calls for a **prescribed order** of write operations,
- technical reasons may make OS tools unsuitable for a database (*e.g.*, file size limitation, platform independence).

### Storage

Torsten Grust



### Magnetic Disks

Access Time  
Sequential vs. Random  
Access

### I/O Parallelism

RAID Levels 1, 0, and 5

### Alternative Storage Techniques

Solid-State Disks  
Network-Based Storage

### Managing Space

Free Space Management

### Buffer Manager

Pinning and Unpinning  
Replacement Policies

### Databases vs. Operating Systems

### Files and Records

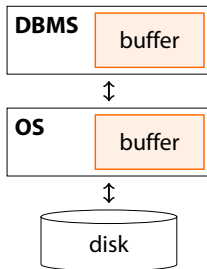
Heap Files  
Free Space Management  
Inside a Page  
Alternative Page Layouts

### Recap

## Databases vs. Operating Systems

In fact, databases and operating systems sometimes interfere:

- Operating system and buffer manager effectively buffer the same data twice.
  - Things get really bad if parts of the DBMS buffer get swapped out to disk by OS VM manager.
  - Therefore, database systems try to **turn off** certain OS features or services.
- ⇒ **Raw disk** access instead of OS files.



### Storage

Torsten Grust



### Magnetic Disks

Access Time  
Sequential vs. Random  
Access

### I/O Parallelism

RAID Levels 1, 0, and 5

### Alternative Storage Techniques

Solid-State Disks  
Network-Based Storage

### Managing Space

Free Space Management

### Buffer Manager

Pinning and Unpinning  
Replacement Policies

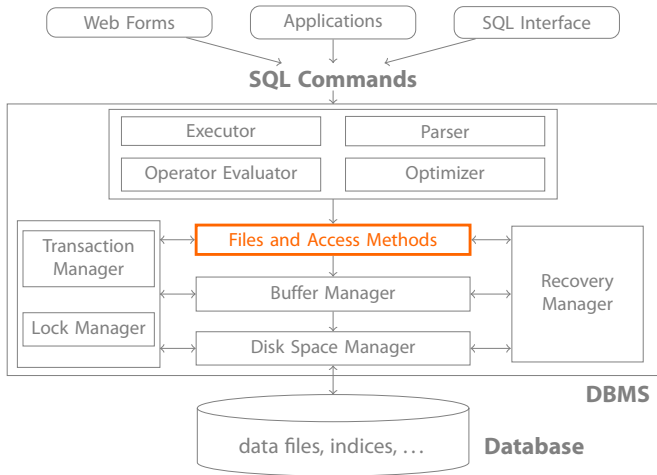
### Databases vs. Operating Systems

### Files and Records

Heap Files  
Free Space Management  
Inside a Page  
Alternative Page Layouts

### Recap

## Files and Records



### Storage

Torsten Grust



### Magnetic Disks

Access Time  
Sequential vs. Random  
Access

### I/O Parallelism

RAID Levels 1, 0, and 5

### Alternative Storage Techniques

Solid-State Disks  
Network-Based Storage

### Managing Space

Free Space Management

### Buffer Manager

Pinning and Unpinning  
Replacement Policies

### Databases vs. Operating Systems

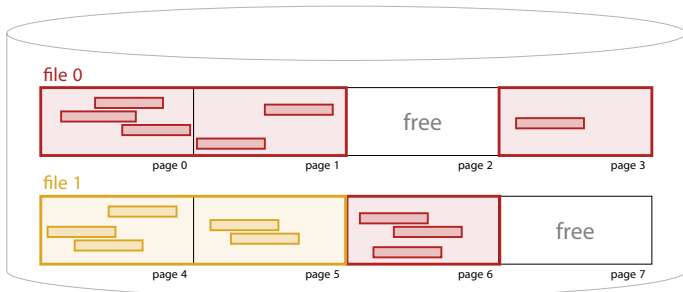
### Files and Records

Heap Files  
Free Space Management  
Inside a Page  
Alternative Page Layouts

### Recap

## Database Files

- So far we have talked about **pages**. Their management is oblivious with respect to their actual content.
- On the conceptual level, a DBMS primarily manages **tables of tuples** and **indexes**.
- Such tables are implemented as **files of records**:
  - A **file** consists of **one or more pages**,
  - each **page** contains **one or more records**,
  - each **record** corresponds to **one tuple**:



### Storage

Torsten Grust



### Magnetic Disks

Access Time

Sequential vs. Random Access

### I/O Parallelism

RAID Levels 1, 0, and 5

### Alternative Storage Techniques

Solid-State Disks

Network-Based Storage

### Managing Space

Free Space Management

### Buffer Manager

Pinning and Unpinning

Replacement Policies

### Databases vs. Operating Systems

### Files and Records

Heap Files

Free Space Management

Inside a Page

Alternative Page Layouts

### Recap

## Database Heap Files

The most important type of files in a database is the **heap file**. It stores records in **no particular order** (in line with, e.g., the SQL semantics):

### Typical heap file interface

- **create/destroy** heap file  $f$  named  $n$ :  
 $f = \text{createFile}(n), \text{deleteFile}(n)$
- **insert** record  $r$  and return its  $rid$ :  
 $rid = \text{insertRecord}(f, r)$
- **delete** a record with a given  $rid$ :  
 $\text{deleteRecord}(f, rid)$
- **get** a record with a given  $rid$ :  
 $r = \text{getRecord}(f, rid)$
- initiate a **sequential scan** over the whole heap file:  
 $\text{openScan}(f)$

**N.B. Record ids** ( $rid$ ) are used like **record addresses** (or pointers). The heap file structure maps a given  $rid$  to the page containing the record.

### Storage

Torsten Grust



### Magnetic Disks

Access Time  
Sequential vs. Random  
Access

### I/O Parallelism

RAID Levels 1, 0, and 5

### Alternative Storage Techniques

Solid-State Disks  
Network-Based Storage

### Managing Space

Free Space Management

### Buffer Manager

Pinning and Unpinning  
Replacement Policies

### Databases vs. Operating Systems

### Files and Records

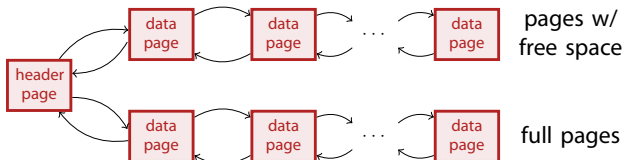
Heap Files  
Free Space Management  
Inside a Page  
Alternative Page Layouts

### Recap

## Heap Files

### (Doubly) Linked list of pages:

Header page allocated when `createFile(n)` is called—initially both page lists are empty:



+ easy to implement

- most pages will end up in free page list
- might have to search many pages to place a (large) record

#### Storage

Torsten Grust



#### Magnetic Disks

Access Time

Sequential vs. Random Access

#### I/O Parallelism

RAID Levels 1, 0, and 5

#### Alternative Storage Techniques

Solid-State Disks

Network-Based Storage

#### Managing Space

Free Space Management

#### Buffer Manager

Pinning and Unpinning

Replacement Policies

#### Databases vs. Operating Systems

#### Files and Records

Heap Files

Free Space Management

Inside a Page

Alternative Page Layouts

#### Recap

## Heap Files

### Operation `insertRecord(f, r)` for linked list of pages

- 1 Try to find a page  $p$  in the free list with free space  $> |r|$ ; should this fail, ask the disk space manager to allocate a new page  $p$
- 2 Write record  $r$  to page  $p$
- 3 Since, generally,  $|r| \ll |p|$ ,  $p$  will belong to the list of pages with free space
- 4 A unique  $rid$  for  $r$  is generated and returned to the caller

### Generating sensible record ids ( $rid$ )

Given that  $rids$  are used like record addresses: what would be a feasible  $rid$  generation method?

#### Storage

Torsten Grust



#### Magnetic Disks

Access Time  
Sequential vs. Random  
Access

#### I/O Parallelism

RAID Levels 1, 0, and 5

#### Alternative Storage Techniques

Solid-State Disks  
Network-Based Storage

#### Managing Space

Free Space Management

#### Buffer Manager

Pinning and Unpinning  
Replacement Policies

#### Databases vs. Operating Systems

#### Files and Records

##### Heap Files

Free Space Management  
Inside a Page  
Alternative Page Layouts

#### Recap



## Heap Files

### Operation `insertRecord(f, r)` for linked list of pages

- 1 Try to find a page  $p$  in the free list with free space  $> |r|$ ; should this fail, ask the disk space manager to allocate a new page  $p$
- 2 Write record  $r$  to page  $p$
- 3 Since, generally,  $|r| \ll |p|$ ,  $p$  will belong to the list of pages with free space
- 4 A unique *rid* for  $r$  is generated and returned to the caller

### Generating sensible record ids (*rid*)

Given that *rids* are used like record addresses: what would be a feasible *rid* generation method?

Generate a **composite** *rid* consisting of the address of page  $p$  and the placement (offset/slot) of  $r$  inside  $p$ :

$$\langle \text{pageno } p, \text{slotno } r \rangle$$

## Storage

Torsten Grust



### Magnetic Disks

Access Time

Sequential vs. Random Access

### I/O Parallelism

RAID Levels 1, 0, and 5

### Alternative Storage Techniques

Solid-State Disks

Network-Based Storage

### Managing Space

Free Space Management

### Buffer Manager

Pinning and Unpinning

Replacement Policies

### Databases vs. Operating Systems

### Files and Records

Heap Files

Free Space Management

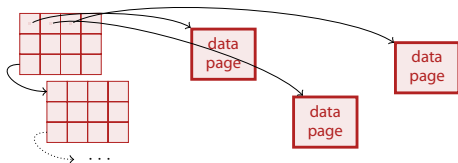
Inside a Page

Alternative Page Layouts

### Recap

## Heap Files

### Directory of pages:



- Use as **space map** with information about free space on each page
  - granularity as trade-off space  $\leftrightarrow$  accuracy (may range from *open/closed* bit to exact information)
- + free space search more efficient
- memory overhead to host the page directory

### Storage

Torsten Grust



### Magnetic Disks

Access Time  
Sequential vs. Random  
Access

### I/O Parallelism

RAID Levels 1, 0, and 5

### Alternative Storage Techniques

Solid-State Disks  
Network-Based Storage

### Managing Space

Free Space Management

### Buffer Manager

Pinning and Unpinning  
Replacement Policies

### Databases vs. Operating Systems

### Files and Records

#### Heap Files

Free Space Management  
Inside a Page  
Alternative Page Layouts

### Recap

## Free Space Management

Which page to pick for the insertion of a new record?

### Append Only

Always insert into last page. Otherwise, create a new page.

### Best Fit

Reduces fragmentation, but requires searching the entire free list/space map for each insert.

### First Fit

Search from beginning, take first page with sufficient space.  
(⇒ These pages quickly fill up, system may waste a lot of search effort in these first pages later on.)

### Next Fit

Maintain **cursor** and continue searching where search stopped last time.

#### Storage

Torsten Grust



#### Magnetic Disks

Access Time

Sequential vs. Random Access

#### I/O Parallelism

RAID Levels 1, 0, and 5

#### Alternative Storage Techniques

Solid-State Disks

Network-Based Storage

#### Managing Space

Free Space Management

#### Buffer Manager

Pinning and Unpinning

Replacement Policies

#### Databases vs. Operating Systems

#### Files and Records

Heap Files

Free Space Management

Inside a Page

Alternative Page Layouts

#### Recap

## Free Space Witnesses

We can accelerate the search by remembering **witnesses**:

- Classify pages into **buckets**, *e.g.*, “75 %–100 % full”, “50 %–75 % full”, “25 %–50 % full”, and “0 %–25 % full”.
- For each bucket, remember some **witness pages**.
- Do a regular best/first/next fit search only if no witness is recorded for the specific bucket.
- Populate witness information, *e.g.*, as a side effect when searching for a best/first/next fit page.

### Storage

Torsten Grust



### Magnetic Disks

Access Time

Sequential vs. Random Access

### I/O Parallelism

RAID Levels 1, 0, and 5

### Alternative Storage Techniques

Solid-State Disks

Network-Based Storage

### Managing Space

Free Space Management

### Buffer Manager

Pinning and Unpinning

Replacement Policies

### Databases vs. Operating Systems

### Files and Records

Heap Files

Free Space Management

Inside a Page

Alternative Page Layouts

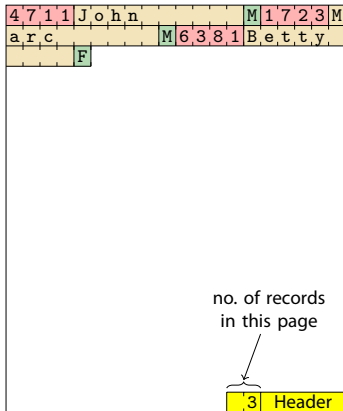
### Recap

## Inside a Page — Fixed-Length Records

Now turn to the **internal page structure**:

ID	NAME	SEX
4711	John	M
1723	Marc	M
6381	Betty	F

- **Record identifier** (*rid*):  
*(pageno, slotno)*
- Record position (within page):  
 $slotno \times bytes\ per\ slot$
- Record **deletion**?
  - record id should **not** change $\Rightarrow$  **slot directory** (bitmap)



### Storage

Torsten Grust



### Magnetic Disks

Access Time  
Sequential vs. Random  
Access

### I/O Parallelism

RAID Levels 1, 0, and 5

### Alternative Storage Techniques

Solid-State Disks  
Network-Based Storage

### Managing Space

Free Space Management

### Buffer Manager

Pinning and Unpinning  
Replacement Policies

### Databases vs. Operating Systems

### Files and Records

Heap Files  
Free Space Management

### Inside a Page

Alternative Page Layouts

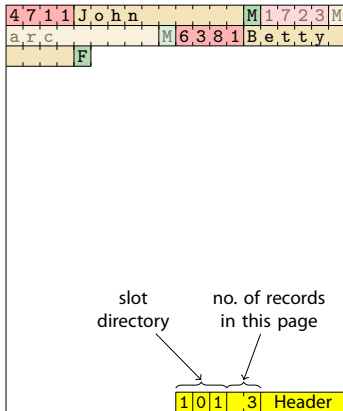
### Recap

## Inside a Page — Fixed-Length Records

Now turn to the **internal page structure**:

ID	NAME	SEX
4711	John	M
1723	Marc	M
6381	Betty	F

- **Record identifier** (*rid*):  
 $\langle \text{pageno}, \text{slotno} \rangle$
- Record position (within page):  
 $\text{slotno} \times \text{bytes per slot}$
- Record **deletion**?
  - record id should **not** change $\Rightarrow$  **slot directory** (bitmap)



### Storage

Torsten Grust



### Magnetic Disks

Access Time  
Sequential vs. Random  
Access

### I/O Parallelism

RAID Levels 1, 0, and 5

### Alternative Storage Techniques

Solid-State Disks  
Network-Based Storage

### Managing Space

Free Space Management

### Buffer Manager

Pinning and Unpinning  
Replacement Policies

### Databases vs. Operating Systems

### Files and Records


Heap Files  
Free Space Management

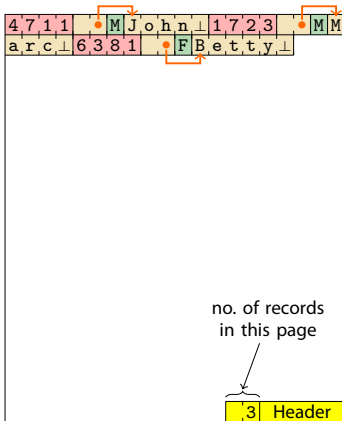
### Inside a Page

Alternative Page Layouts

### Recap

## Inside a Page—Variable-Sized Fields

- Variable-sized fields moved to **end** of each record.
  - Placeholder points to location.
  -  **Why?**



### Storage

Torsten Grust



### Magnetic Disks

Access Time  
Sequential vs. Random  
Access

### I/O Parallelism

RAID Levels 1, 0, and 5

### Alternative Storage Techniques

Solid-State Disks  
Network-Based Storage

### Managing Space

Free Space Management

### Buffer Manager

Pinning and Unpinning  
Replacement Policies

### Databases vs. Operating Systems

### Files and Records


Heap Files  
Free Space Management

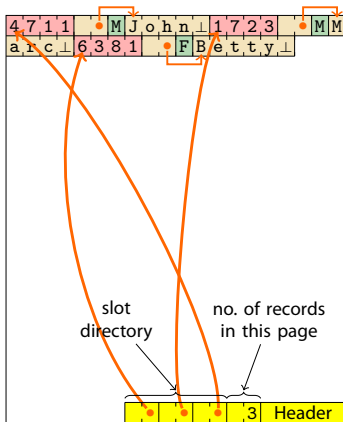
### Inside a Page

Alternative Page Layouts

### Recap

## Inside a Page—Variable-Sized Fields

- Variable-sized fields moved to **end** of each record.
  - Placeholder points to location.
  -  **Why?**
- Slot directory points to start of each record.



### Storage

Torsten Grust



### Magnetic Disks

Access Time

Sequential vs. Random Access

### I/O Parallelism

RAID Levels 1, 0, and 5

### Alternative Storage Techniques

Solid-State Disks

Network-Based Storage

### Managing Space

Free Space Management

### Buffer Manager

Pinning and Unpinning

Replacement Policies

### Databases vs. Operating Systems

### Files and Records

Heap Files

Free Space Management


### Inside a Page

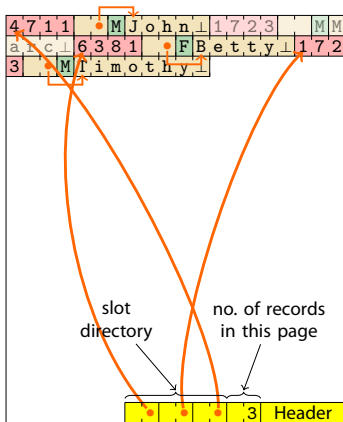
Alternative Page Layouts

### Recap



## Inside a Page—Variable-Sized Fields

- Variable-sized fields moved to **end** of each record.
  - Placeholder points to location.
  -  **Why?**
- Slot directory points to start of each record.
- Records **can move** on page.
  - *E.g.*, if field size changes or page is compacted.



### Storage

Torsten Grust



### Magnetic Disks

Access Time  
Sequential vs. Random  
Access

### I/O Parallelism

RAID Levels 1, 0, and 5

### Alternative Storage Techniques

Solid-State Disks  
Network-Based Storage

### Managing Space

Free Space Management

### Buffer Manager

Pinning and Unpinning  
Replacement Policies

### Databases vs. Operating Systems

### Files and Records



Heap Files  
Free Space Management

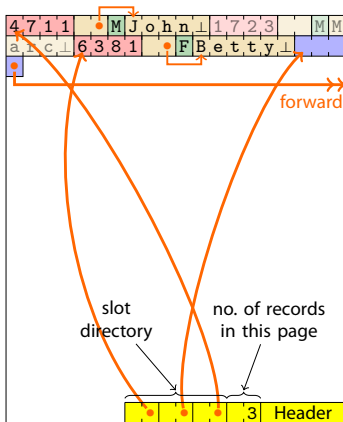
### Inside a Page

Alternative Page Layouts

### Recap

## Inside a Page—Variable-Sized Fields

- Variable-sized fields moved to **end** of each record.
  - Placeholder points to location.
  -  **Why?**
- Slot directory points to start of each record.
- Records **can move** on page.
  - *E.g.*, if field size changes or page is compacted.
- Create **“forward address”** if record won't fit on page.
  -  **Future updates?**



### Storage

Torsten Grust



### Magnetic Disks

Access Time  
Sequential vs. Random  
Access

### I/O Parallelism

RAID Levels 1, 0, and 5

### Alternative Storage Techniques

Solid-State Disks  
Network-Based Storage

### Managing Space

Free Space Management

### Buffer Manager

Pinning and Unpinning  
Replacement Policies

### Databases vs. Operating Systems

### Files and Records



Heap Files  
Free Space Management

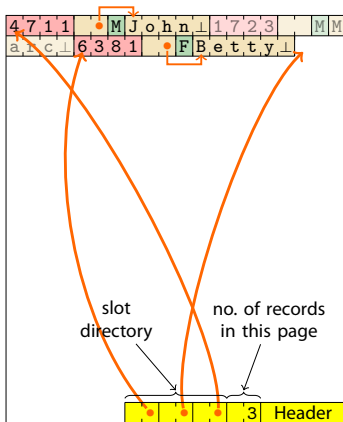
### Inside a Page

Alternative Page Layouts

### Recap

## Inside a Page—Variable-Sized Fields

- Variable-sized fields moved to **end** of each record.
  - Placeholder points to location.
  -  **Why?**
- Slot directory points to start of each record.
- Records **can move** on page.
  - *E.g.*, if field size changes or page is compacted.
- Create **“forward address”** if record won't fit on page.
  -  **Future updates?**
- Related issue: space-efficient representation of NULL values.



### Storage

Torsten Grust



### Magnetic Disks

Access Time

Sequential vs. Random Access

### I/O Parallelism

RAID Levels 1, 0, and 5

### Alternative Storage Techniques

Solid-State Disks

Network-Based Storage

### Managing Space

Free Space Management

### Buffer Manager

Pinning and Unpinning

Replacement Policies

### Databases vs. Operating Systems

### Files and Records

Heap Files

Free Space Management


### Inside a Page

Alternative Page Layouts

### Recap

## IBM DB2 Data Pages

### DB2. Data page and layout details

- Support for **4 K, 8 K, 16 K, 32 K data pages** in separate table spaces. Buffer manager pages match in size.
- 68 bytes of database manager overhead per page. On a 4 K page: maximum of 4,028 bytes of user data (maximum record size: 4,005 bytes). Records do *not* span pages.
- **Maximum table size:** 512 GB (with 32 K pages). Maximum number of columns: 1,012 (4 K page: 500), maximum number of rows/page: 255.  **IBM DB2 RID format?**
- Columns of type LONG VARCHAR, CLOB, etc. maintained outside regular data pages (pages contain descriptors only).
- **Free space management:** first-fit order. Free space map distributed on every 500th page in FSCR (free space control records). Records updated in-place if possible, otherwise uses forward records.

#### Storage

Torsten Grust



#### Magnetic Disks

Access Time

Sequential vs. Random Access

#### I/O Parallelism

RAID Levels 1, 0, and 5

#### Alternative Storage Techniques

Solid-State Disks

Network-Based Storage

#### Managing Space

Free Space Management

#### Buffer Manager

Pinning and Unpinning

Replacement Policies

#### Databases vs. Operating Systems

#### Files and Records

Heap Files

Free Space Management

#### Inside a Page

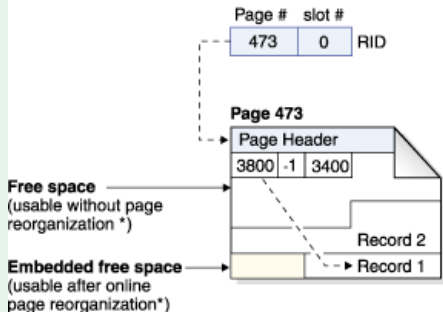
Alternative Page Layouts

#### Recap

## IBM DB2 Data Pages

**DB2.** Taken directly from the DB2 V9.5 Information Center

### Data page and RID format



**Supported page sizes:**  
4KB, 8KB,  
16KB, 32KB  
Set on table space creation.  
Each table space must be  
assigned a buffer pool with  
a matching page size.

\* Exception: Any space reserved by an uncommitted DELETE is not usable.

<http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/>

### Storage

Torsten Grust



#### Magnetic Disks

Access Time  
Sequential vs. Random  
Access

#### I/O Parallelism

RAID Levels 1, 0, and 5

#### Alternative Storage Techniques

Solid-State Disks  
Network-Based Storage

#### Managing Space

Free Space Management

#### Buffer Manager

Pinning and Unpinning  
Replacement Policies

#### Databases vs. Operating Systems

#### Files and Records

Heap Files  
Free Space Management

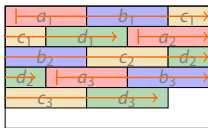
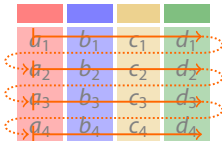
#### Inside a Page

Alternative Page Layouts

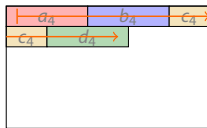
#### Recap

## Alternative Page Layouts

We have just populated data pages in a **row-wise** fashion:

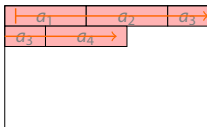
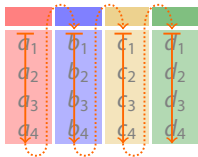


page 0

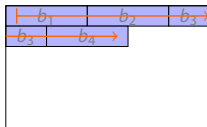


page 1

We could as well do that **column-wise**:



page 0



page 1

### Storage

Torsten Grust



### Magnetic Disks

Access Time

Sequential vs. Random Access

### I/O Parallelism

RAID Levels 1, 0, and 5

### Alternative Storage Techniques

Solid-State Disks

Network-Based Storage

### Managing Space

Free Space Management

### Buffer Manager

Pinning and Unpinning

Replacement Policies

### Databases vs. Operating Systems

### Files and Records

Heap Files

Free Space Management

Inside a Page

Alternative Page Layouts

### Recap

## Alternative Page Layouts

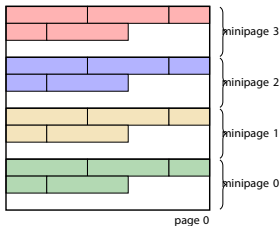
These two approaches are also known as **NSM (n-ary storage model)** and **DSM (decomposition storage model)**.<sup>1</sup>

- Tuning knob for certain workload types (e.g., OLAP)
- Suitable for narrow projections and in-memory database systems  
(↗ Database Systems and Modern CPU Architecture)
- Different behavior with respect to **compression**.

A hybrid approach is the **PAX (Partition Attributes Across)** layout:

- Divide each page into **minipages**.
- Group attributes into them.

↗ Ailamaki *et al.* Weaving Relations for Cache Performance. *VLDB 2001*.



<sup>1</sup>Recently, the terms **row-store** and **column-store** have become popular, too.

### Storage

Torsten Grust



### Magnetic Disks

Access Time

Sequential vs. Random Access

### I/O Parallelism

RAID Levels 1, 0, and 5

### Alternative Storage Techniques

Solid-State Disks

Network-Based Storage

### Managing Space

Free Space Management

### Buffer Manager

Pinning and Unpinning

Replacement Policies

### Databases vs. Operating Systems

### Files and Records

Heap Files

Free Space Management

Inside a Page

Alternative Page Layouts

### Recap

## Recap

### Magnetic Disks

Random access **orders of magnitude** slower than sequential.

### Disk Space Manager

Abstracts from hardware details and maps page number  $\mapsto$  physical location.

### Buffer Manager

Page **caching** in main memory; `pin ()/unpin ()` interface; **replacement policy** crucial for effectiveness.

### File Organization

Stable **record identifiers (rids)**; maintenance with fixed-sized records and variable-sized fields; NSM vs. DSM.

#### Storage

Torsten Grust



#### Magnetic Disks

Access Time

Sequential vs. Random Access

#### I/O Parallelism

RAID Levels 1, 0, and 5

#### Alternative Storage Techniques

Solid-State Disks

Network-Based Storage

#### Managing Space

Free Space Management

#### Buffer Manager

Pinning and Unpinning

Replacement Policies

#### Databases vs. Operating Systems

#### Files and Records

Heap Files

Free Space Management

Inside a Page

Alternative Page Layouts

#### Recap