Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

CHAI
Humanities-Centered AI

Prof. Dr. Ralf Möller

Databases Architecture

Overview

1. **Introduction**
   - Models and abstraction
   - Database systems
   - Applications and data independence
2. **Entity-relationship model (ER model)**
3. **Relational data model (RDM)**
   - From ER to RDM
4. **Relational schema design theory**
   - Relational algebra as a query language
   - Functional dependencies
   - Normal forms
5. **Structured Query Language (SQL)**
   - Data definition
   - Query specification
   - Data manipulation
6. **Query processing**
   - Architecture
   - Indexing
   - Query plans, optimization
7. **Transactions**
   - Transaction processing, schedules, locks
   - Recovery
8. **Other types of databases**
   - Vector databases
   - Data stream query languages

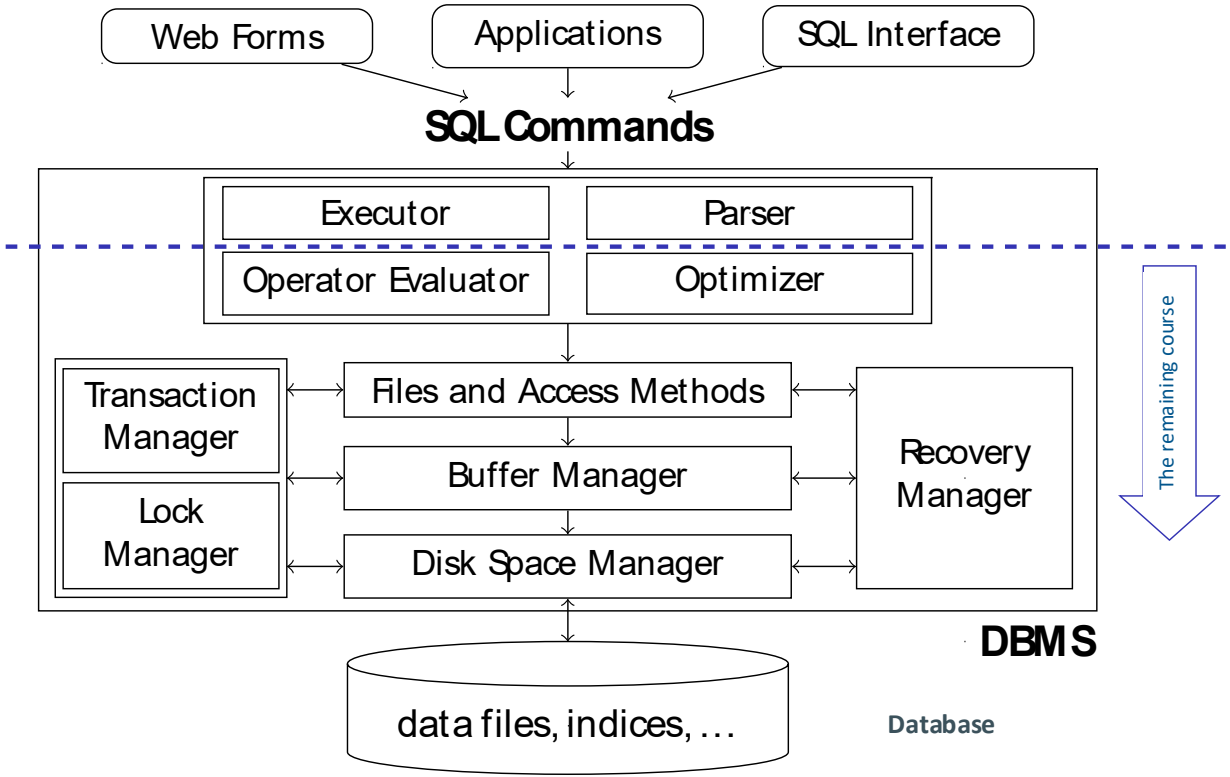
Acknowledgements

- The following parts of the lecture are inspired by the course:

  „Architecture and Implementation of Database Systems“
  by Jens Teubner at ETH Zürich

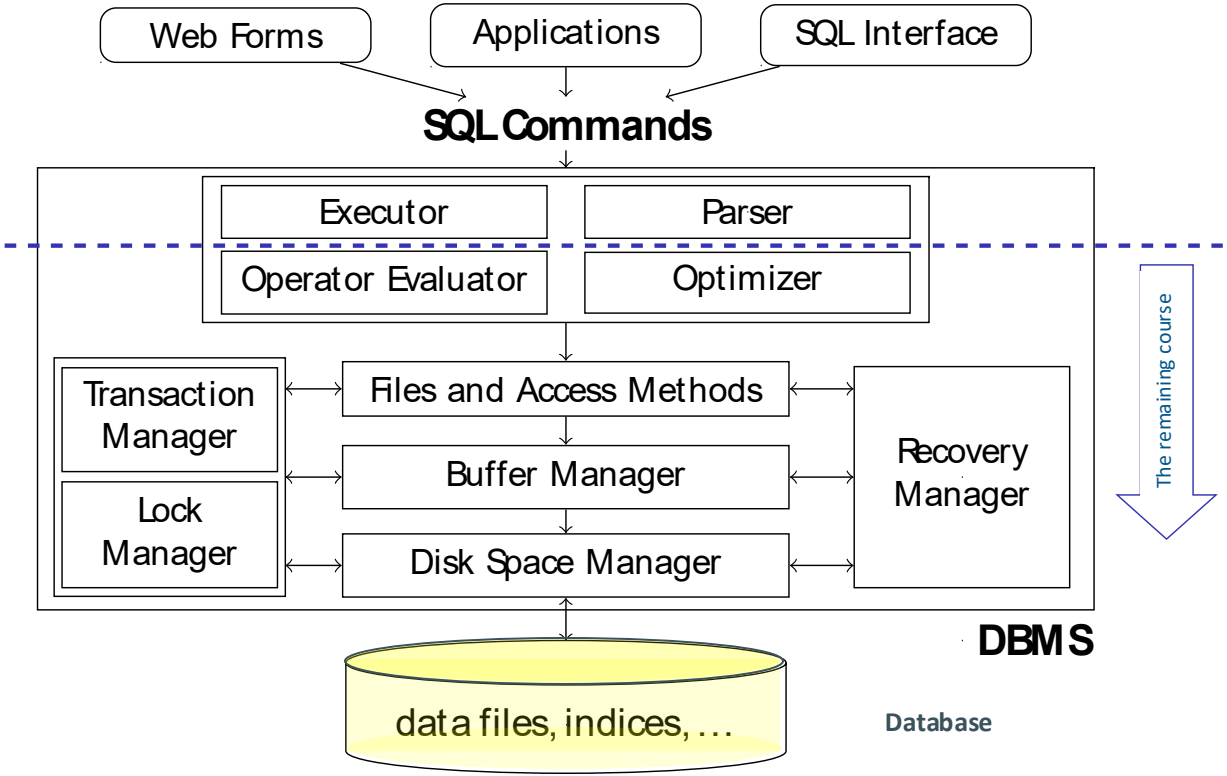
- Some graphics were taken from this course with the author's permission

Architecture of a DBMS




Databases
Architecture

Secondary memory: Disks and files

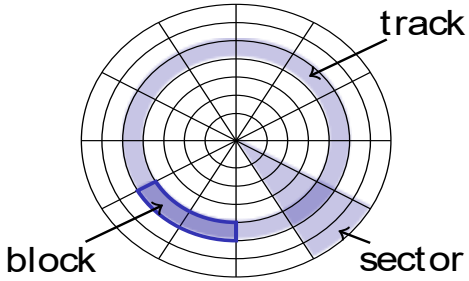


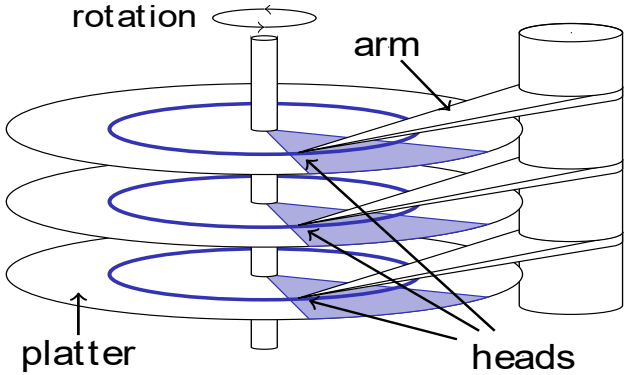
Memory hierarchy

	Capacity	Latency
▪ CPU (with registers)	Bytes	< 1 ns
▪ Cache memory	Kilo-/Mega-Bytes	< 10 ns
▪ Main memory	Giga-Bytes	20-100 ns
▪ Flash memory	Giga/Tera-Bytes	30-250 µs
▪ Hard disk	Tera-Bytes	3-10 ms
▪ Tape machine	Peta-Bytes	varying

- Toward CPU: Fast but small
- Toward Periphery: Slow but large
- Cache memory to reduce latency

Magnetic disks / hard disk drives





- Step motor positions arms on certain tracks
- Magnetic disks (platters) rotate permanently
- Organization into blocks
- Transfer takes place in blocks (reading and writing)

Photo: http://www.metallurgy.utah.edu/

Access time

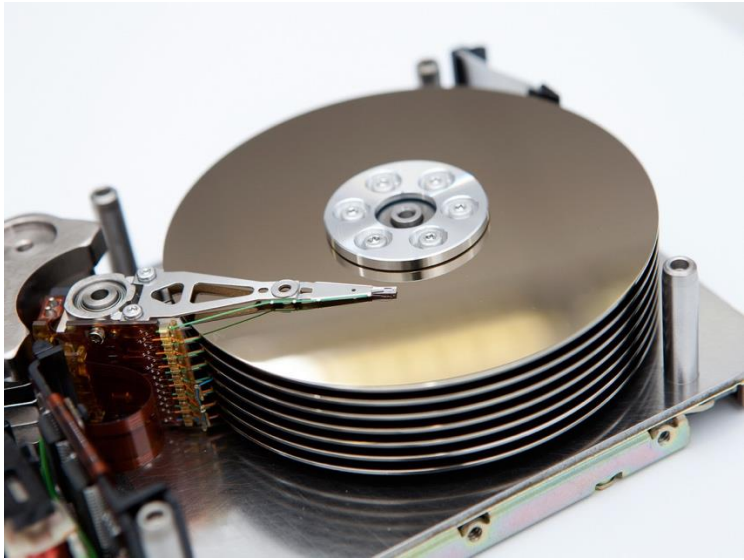
The design of the platters affects the access time (read and write) to a block

Movement of the arms to the desired track t_s

Waiting time for the desired block until it is under the arm (rotation delay t_r)

Read time or write time (transfer time t_{tr})

Access time: $t = t_s + t_r + t_{tr}$



In practice for a concrete model Travelstar 7k200: 14,33 ms

Sequential vs. random access

Example: Read 1000 blocks of 8 KB each

Random access:

- $t_{rnd} = 1000 \cdot 14.33$ ms

Sequential access:

- Travelstar 7k200 has 63 sectors per track, with a track-to-track search time $t_{s,track\text{-}to\text{-}track}$ of 1 ms
- An 8 KB block requires 16 sectors
- $t_{seq} = t_s + t_r + 1000 \cdot t_{tr} + 1000 \cdot 16/63 \cdot t_{s,track\text{-}to\text{-}track} = 10 \text{ ms} + 4.17 \text{ ms} + 160 \text{ ms} + 254 \text{ ms} \approx 428 \text{ ms}$

Insight: Sequential access is much faster than random access: Avoid random I/O if possible

Observation: If few parts of an 8MB file is required only, you might as well read the entire file

Solid state drives

- Latency: SSDs → nanoseconds; HDDs → milliseconds (rotational + seek delay).
- Random I/O: SSDs handle many simultaneous reads/writes with little slowdown; HDDs slow dramatically because the head must move to each sector.
- Sequential I/O: SSDs still faster, but the gap is smaller; HDDs rely on continuous rotation for decent sequential throughput.
- Consistency: SSD performance stays uniform; HDD performance varies with data location (inner vs. outer tracks).

Result: SSDs give virtually instant, uniform access; HDDs are limited by mechanical movement, making random access much slower.

Solid state drives as an alternative to hard disk drives?

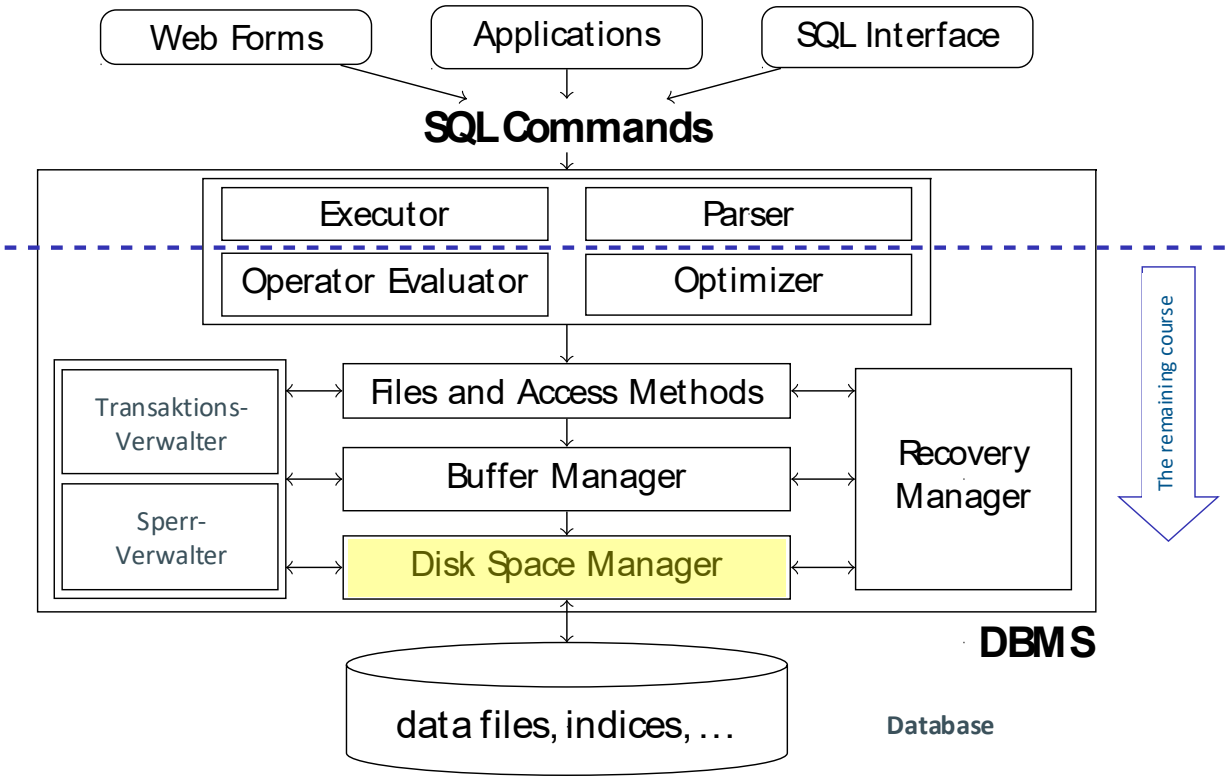
- SSDs have replaced HDDs for most active, performance-critical storage, ...
- ...but they have not (and likely will not soon) replace HDDs for inexpensive, very large, long-term persistent archives

The two technologies will continue to coexist, with smarter tiering layers that move data between them as usage patterns change

Networks are no bottleneck

- Hard disk or SSD throughput: >500 MB/s
- SDRAM (Synchronous Dynamic Random-Access Memory): 50 Gbit/s (latency: ~ ns)
- Ethernet
  - >800 Gbits/s today (latency: ~ ms)
- So why not reference database storage via the network?
  - Network-attached storage (NAS)

Architecture of a DBMS



Management of external memory

- Abstraction of technical details of storage media
- Page concepts with typically 4-64KB as storage units for the remaining components
- Directory for mapping Page number → Physical storage location
  where the physical storage location can be:
  - an operating system file including offset,
  - a specification of the head sector track of a hard disk, or
  - a specification for tape device and number including offset

Management of empty pages

Techniques used:

List of free pages

- Addition to this list if page is no longer used

Bitmap with one bit for each page

- Flipping the bit k when page k is (de)allocated

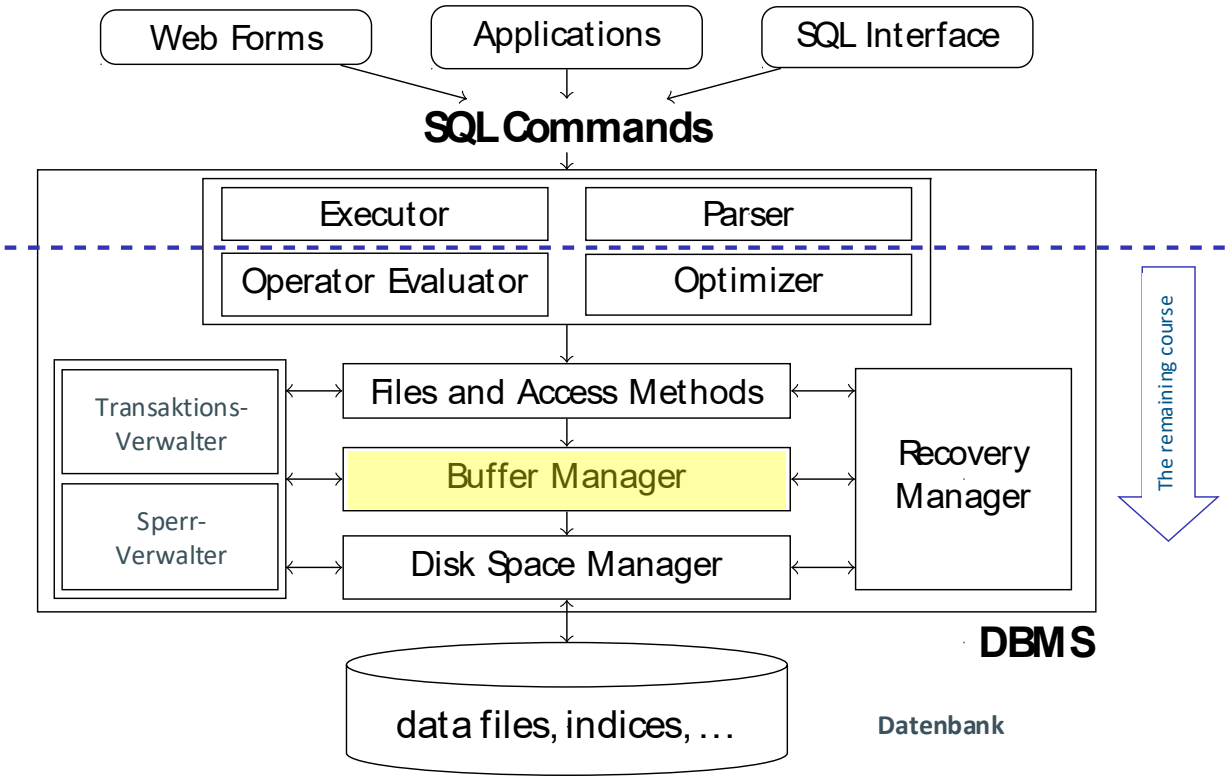
Excercise

Management of empty pages
- List of free pages
- Bitmap with one bit for each page

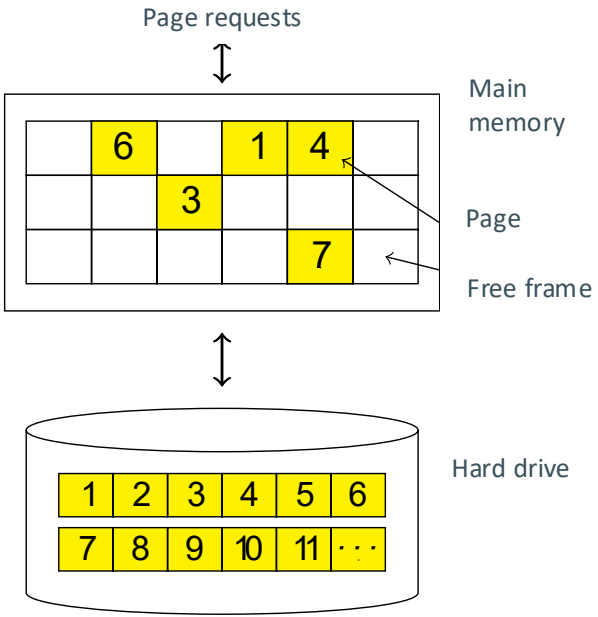
To increase sequential access, consecutive pages should be used.

Which technique, 1 or 2, would you choose to support this?

Architecture of a DBMS



Buffer manager



- Mediates between external and internal memory (main memory)
- Manages a special area in the main memory for this purpose, the buffer pool
- Loads external pages within the buffer pool
- Displacement strategy if the buffer pool is full

Interface of buffer manager

Function pin for requests for pages and unpin for releasing pages after use

pin(pageno)

- Request for page number pageno
- Load page into main memory if necessary
- Return a reference to pageno

unpin(pageno, dirty)

- Release page pageno for possible swapping out
- dirty := true if the page has been modified

Implementation of pin()

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

Implementation of unpin()

```
1 Function: unpin(pageno, dirty)
2 pinCount(pageno)      pinCount(pageno) − 1;
3 if dirty then
4     dirty(pageno)     dirty;
```

Why are pages
not immediately stored
to disk when unpinned?

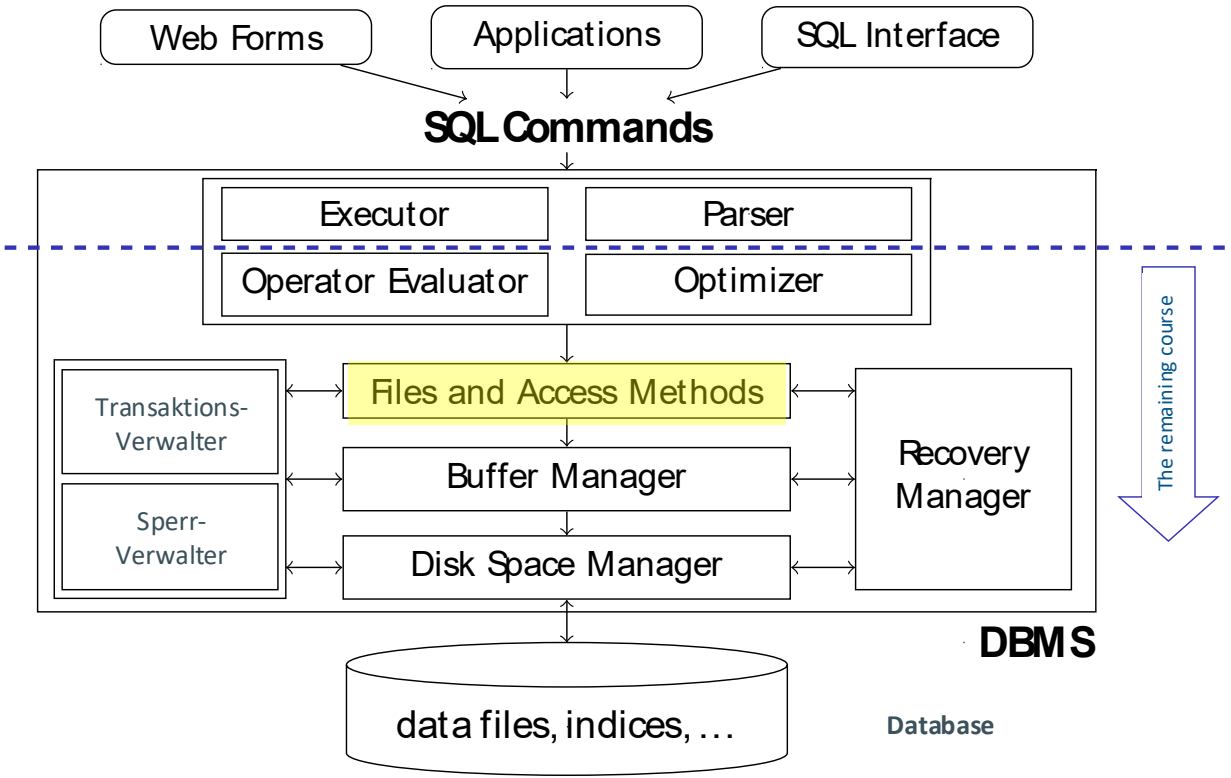
Replacement strategies

The effectiveness of the buffer manager depends on the chosen replacement strategy, e.g.:

- Least Recently Used (LRU)
  - Replacement of the page with the longest unpin()
- LRU-k
  - Like LRU, but k-most recent unpin(), not most recent
- Most Recently Used (MRU)
  - Replacement of the page with the most recent unpin()
- Random
  - Replacement of any page

When should we use which strategy?

Architecture of a DBMS

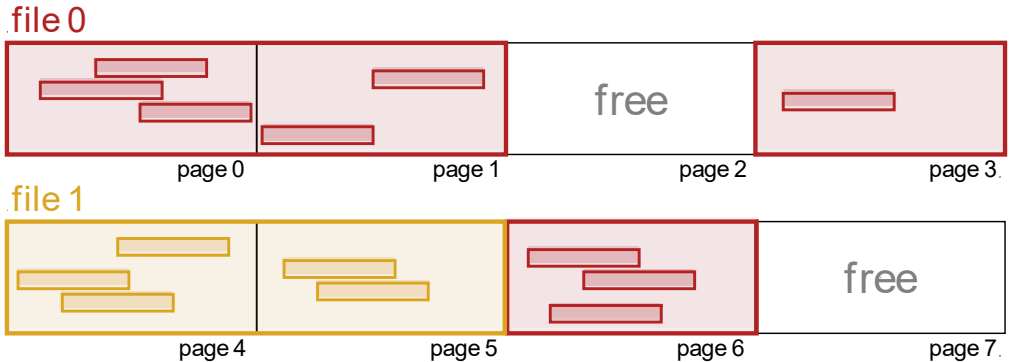


Datenbanken

Architecture

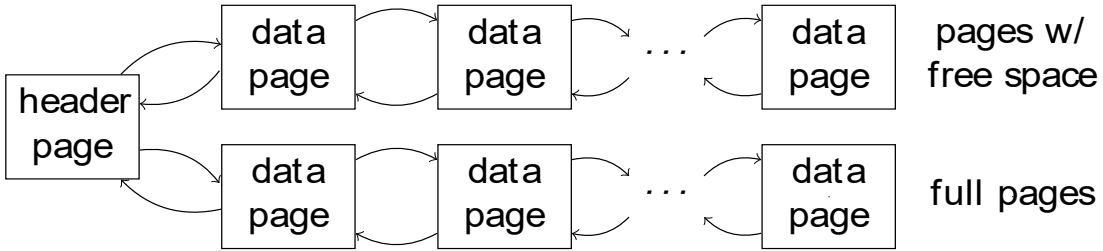
Database files

- Page management unaffected by content
  - DBMS manages tables of tuples, index structures, etc.
- Tables are files of data records
  - A file consists of one or more pages
  - Each page stores one or more data records
  - Each data record corresponds to a tuple



Heap files

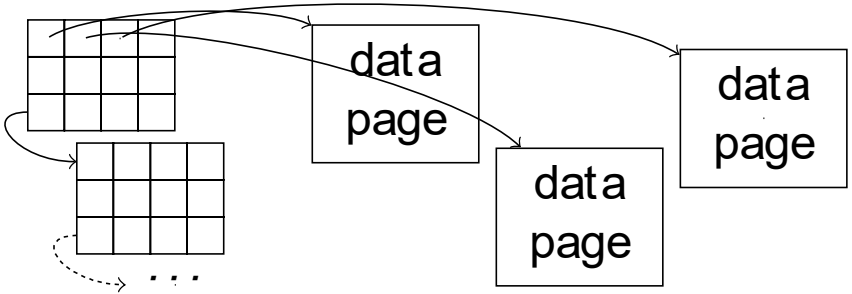
- Most important file type: Storage of data records in arbitrary order (compliant with SQL)
- Linked list of pages



- Easy to implement
- Many pages on the list of free pages

Heap files

- Directory of pages



- Use as an illustration with information about free spaces (granularity is a matter of consideration)
- Efficient search for free spaces
- Additional effort for directory storage

Free memory directory

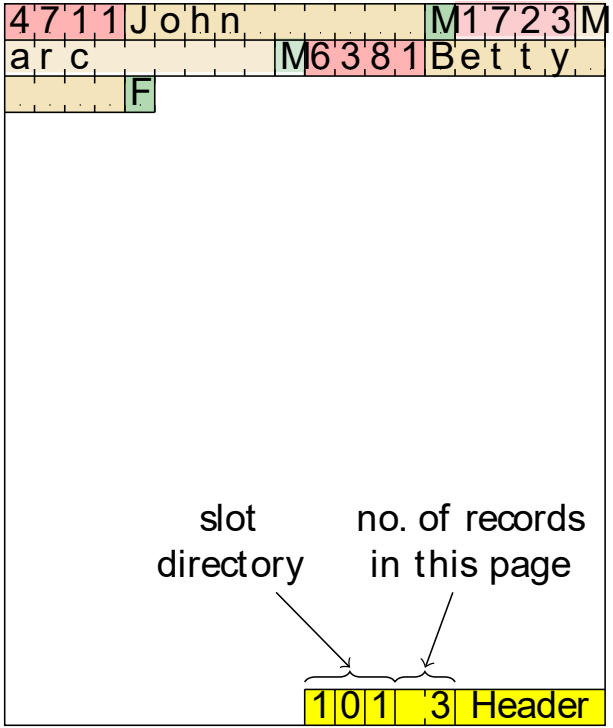
Which page should be selected for new data records?

- Append Only
  - Always append to last page, otherwise request new page
- Best-Fit
  - All pages must be considered, reduction of fragmentation
- First-Fit
  - Search from the beginning, take the first page with enough space
  - First pages fill up quickly, are viewed again and again
- Next-Fit
  - Manage pointers and continue search where it ended last time

Page content

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

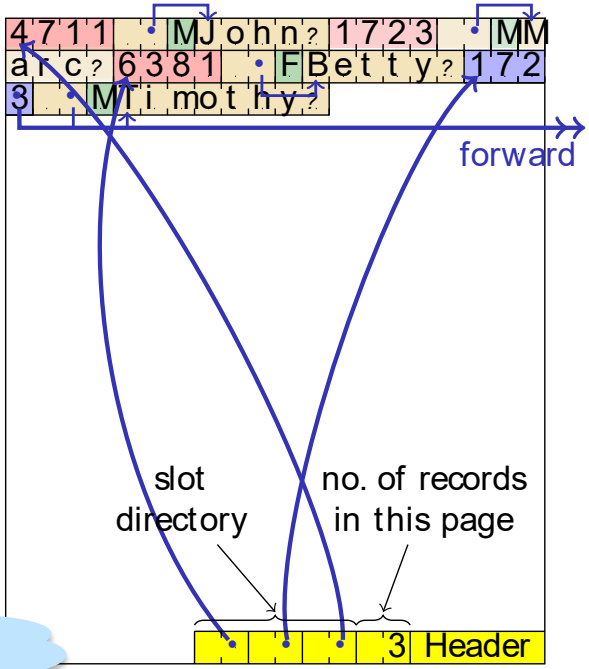
- Data record id (record identifier, rid)
- Data record position (offset on page) Slotno x Bytes per Slot
- Data record deleted?
  - Rid shoud not change
  - Slot directory (bitmap)



Page content: Fields of variable length

- Variable-length fields moved to the end
  - Placeholder points to position
- Slot directory points to start of a field
  - Fields can be moved on the page
  - (e.g., if field size changes)
- Introduction of a forward reference if field does not fit on the page

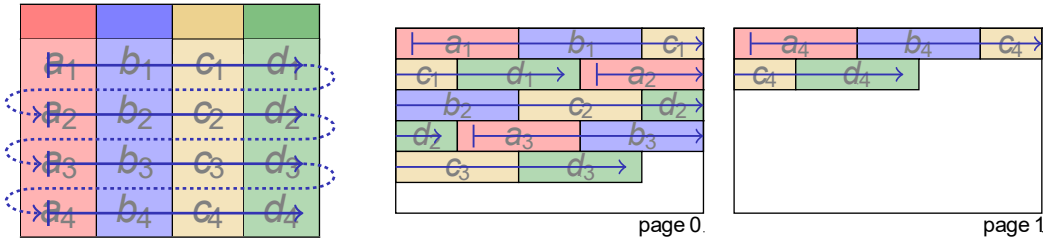
Why?



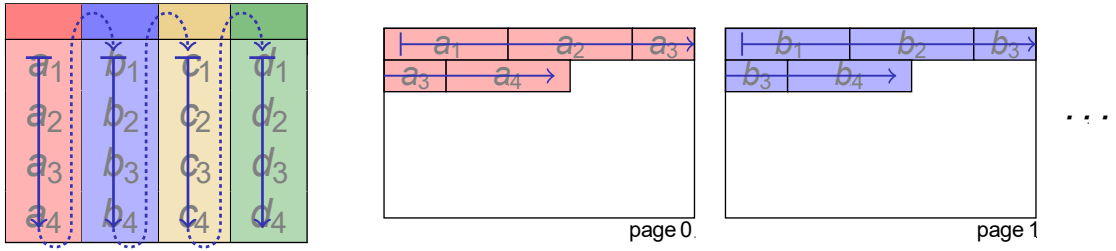
What happens with updates?

Page layout

- Row-wise:

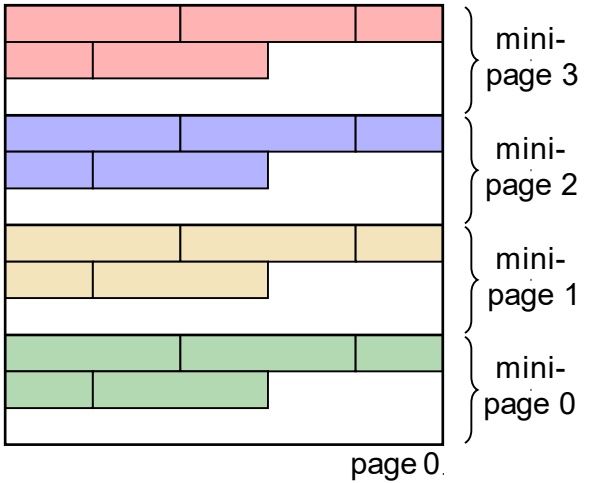


- Column-wise:



Page layout

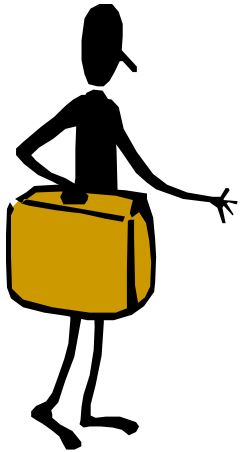
- Featured schemas are also called:
  - Row store
  - Column store
- Applications for different load types and application contexts (e.g., OLAP)
  - Different compression options
- Combination possible:
  - Subdivision of a page into mini-pages with corresponding division



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

Summary

- Characteristics of storage media
  - Slow random access (I/O complexity)
- External storage manager
  - Abstraction of hardware details
  - Page number ⧈ Physical storage location
- Buffer manager
  - Page caching in main memory
  - Replacement strategy
- File organization
  - Stable record identifiers (RIDs)
  - Management of static and dynamic fields



Overview

1. Introduction
   - Models and abstraction
   - Database systems
   - Applications and data independence
2. Entity-relationship model (ER model)
3. Relational data model (RDM)
   - From ER to RDM
4. Relational schema design theory
   - Relational algebra as a query language
   - Functional dependencies
   - Normal forms
5. Structured Query Language (SQL)
   - Data definition
   - Query specification
   - Data manipulation
6. **Query processing**
   - Architecture
   - Indexing
   - Query plans, optimization
7. **Transactions**
   - Transaction processing, schedules, locks
   - Recovery
8. **Other types of databases**
   - Vector databases
   - Data stream query languages