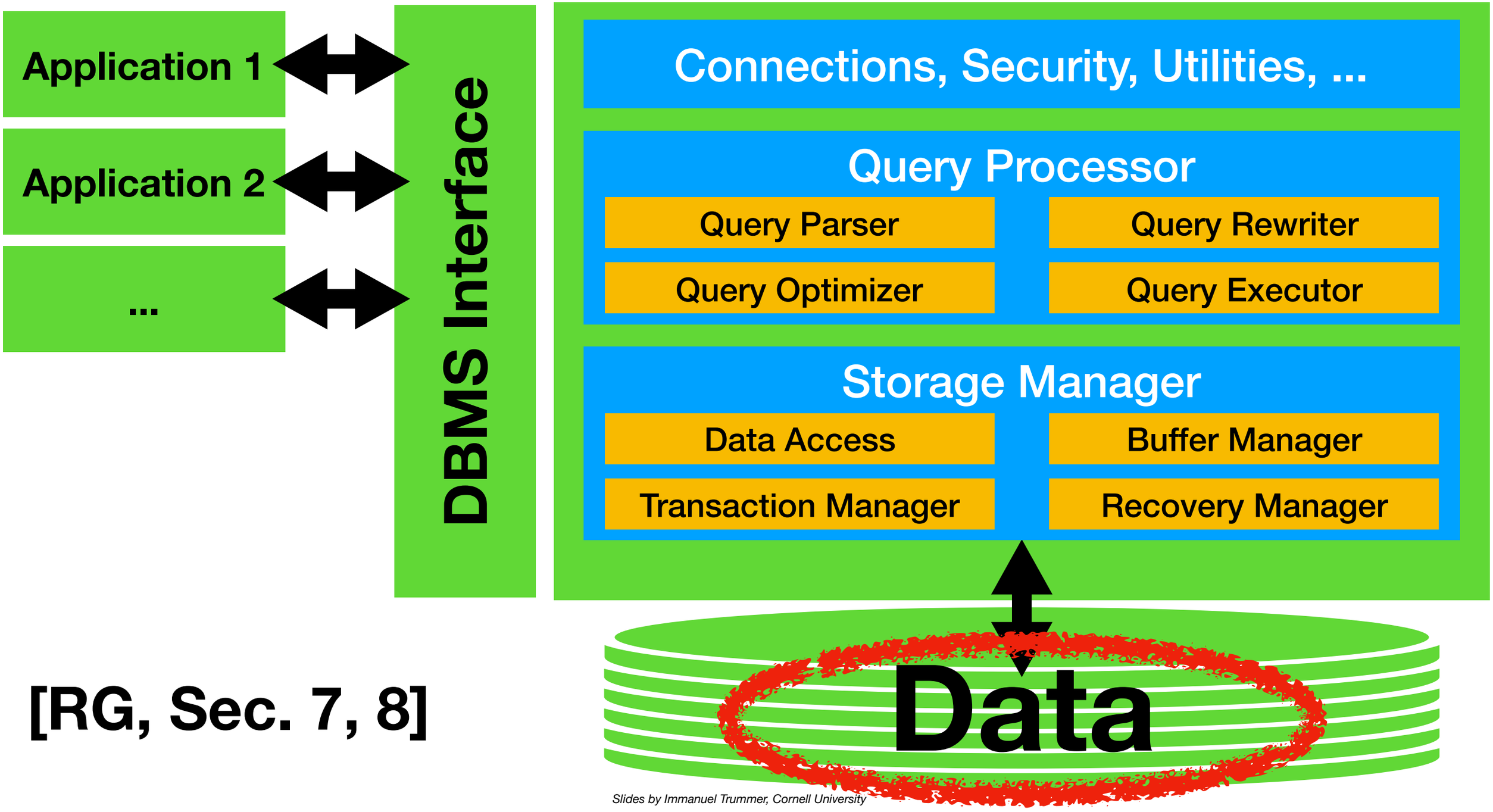
Data Storage

Immanuel Trummer
itrummer@cornell.edu
www.itrummer.org

Database Management Systems (DBMS)



[RG, Sec. 7, 8]

Slides by Immanuel Trummer, Cornell University

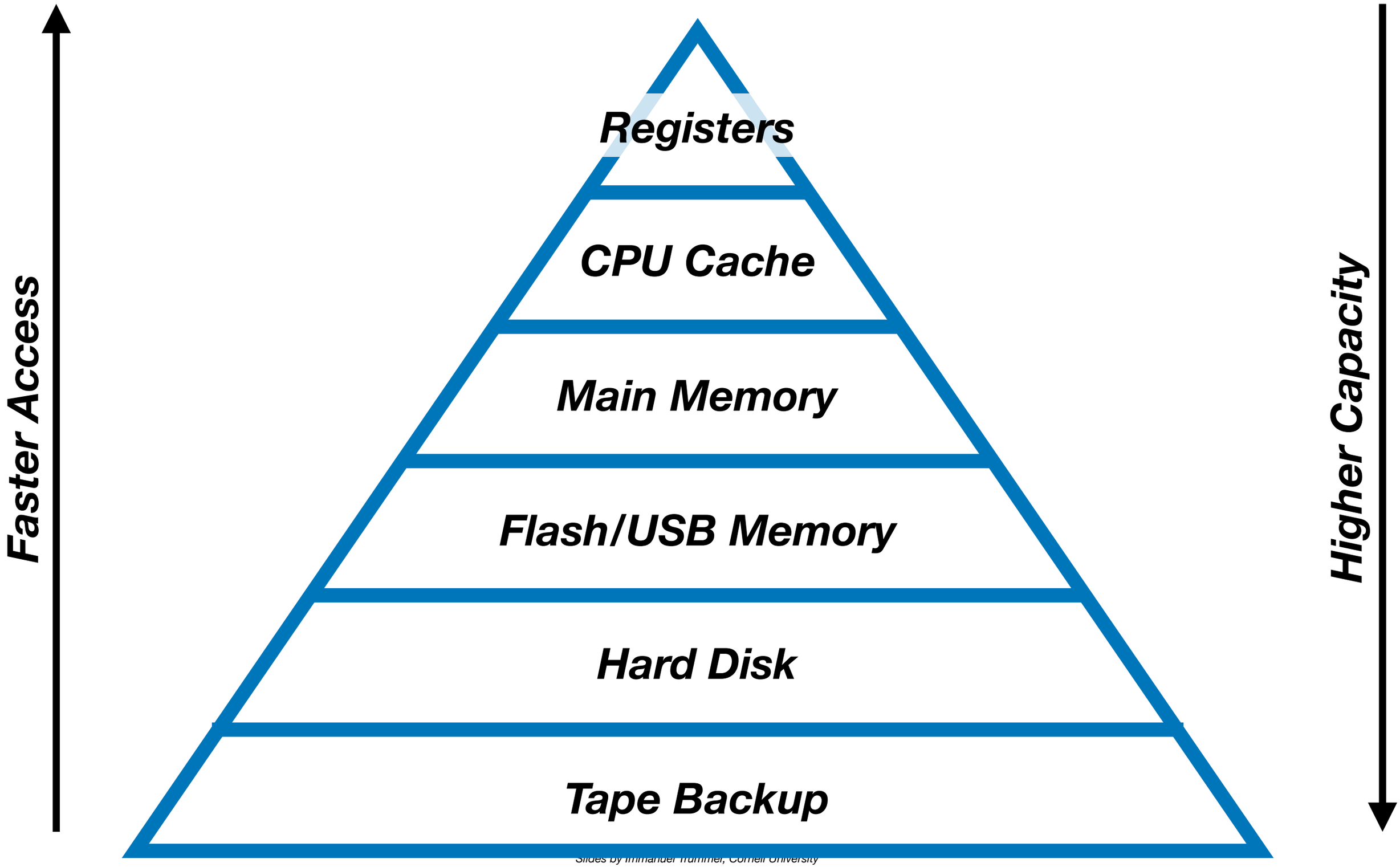
Outline

- Data storage **hardware**
  - What devices to store data on?
- Data storage **format**
  - How to represent relations?

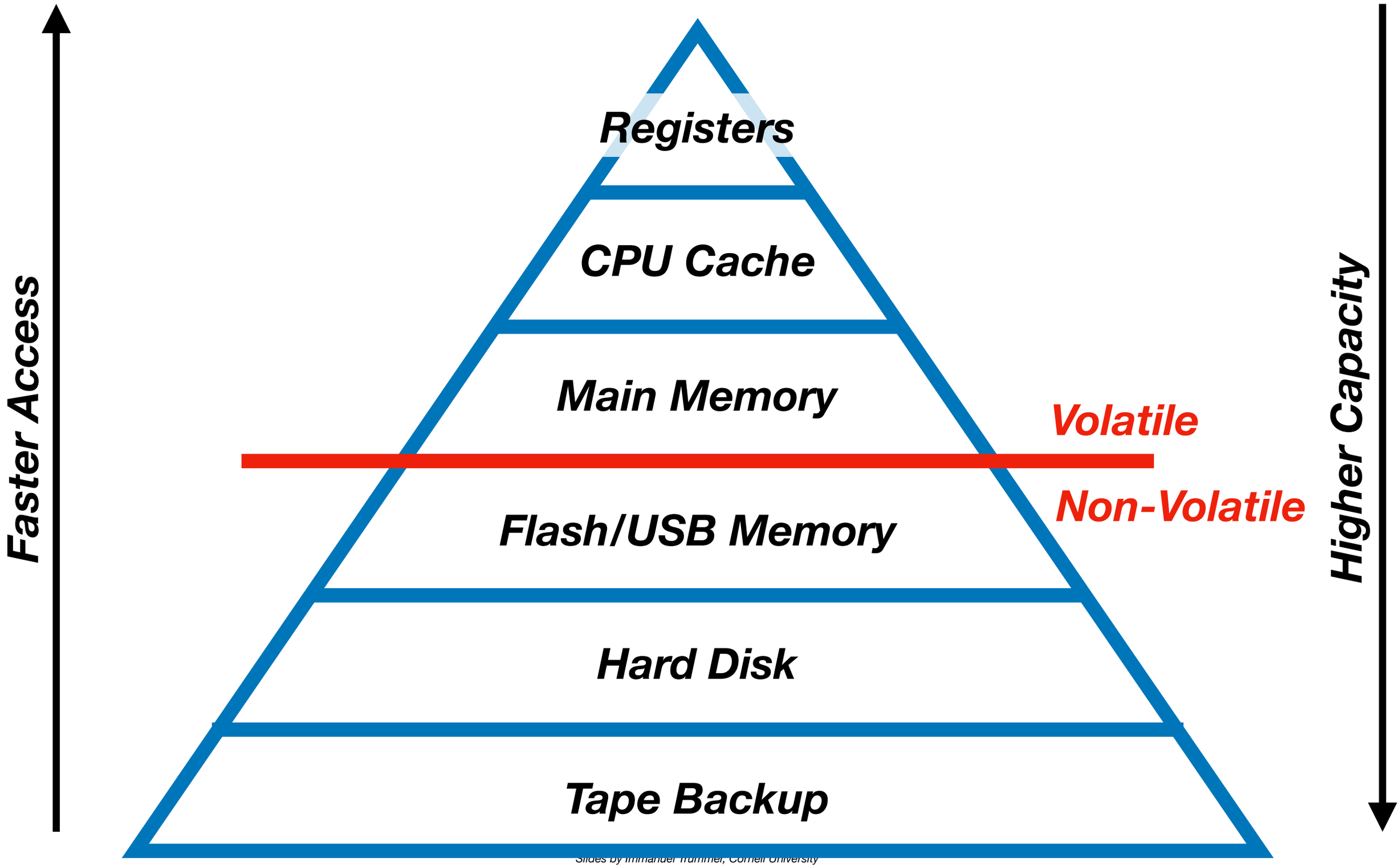
Outline

- Data storage **hardware**
  - What devices to store data on?
- Data storage **format**
  - How to represent relations?

The Memory Hierarchy



The Memory Hierarchy



Tape Storage

Photo: Victor Prado

- Bits as **magnetic** information on tape
- **Very slow access** (10s of seconds)
- **Moderate read speed** (up to 300 MB/second)
- **Very cheap** (around $0.02 per Gigabyte [source])
- Used for long-term **archival** (e.g., by Google)
- More info: Why the future of data storage is (still) magnetic tape, IEEE Spectrum, 2018.

Hard Disk

Photo: Wikimedia Commons

- Bits as **magnetic** information on platter
- Patters **spin** under read/write heads
- **Slow access** (10s of milliseconds access time)
- **Moderate read speed** (around 200 MB/second)
- **Cheap** (around $0.035 per Gigabyte)
- Used for **less frequently accessed** data

Solid State Drives

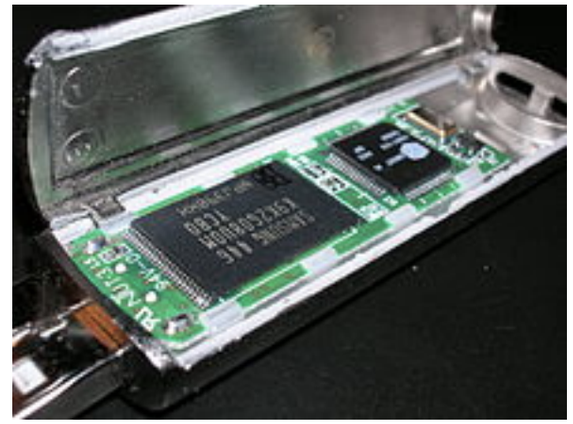


Photo: Wikimedia Commons

- Bits as small **electric charges**
- **Elevated price** (around $0.25 per Gigabyte)
- **Fast access** (around 1 millisecond)
- **Elevated speed** (around 500 MB/second)
- Limited number of write cycles (**memory wear**)

Main Memory

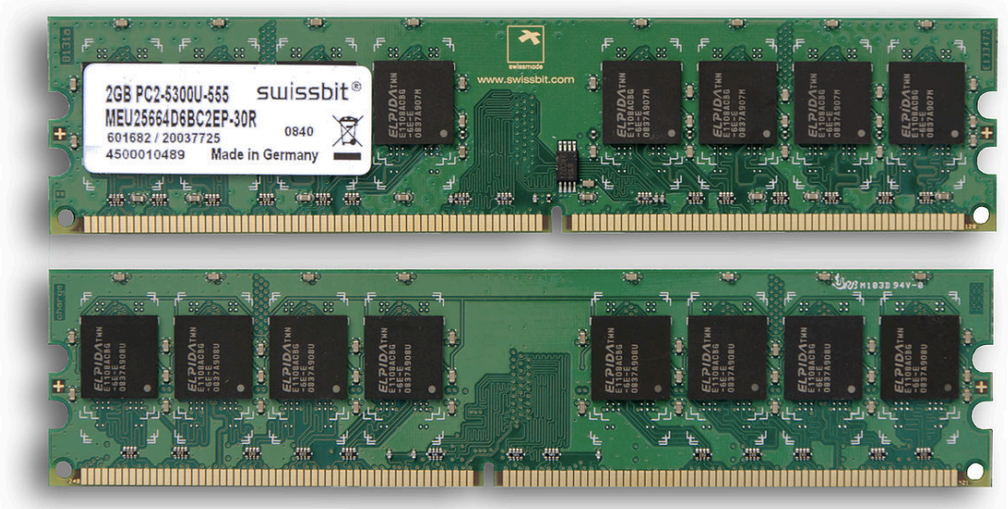


Photo: Wikimedia Commons

- Bits as small **electric** charges
- **Expensive** (several dollars per Gigabyte)
- **Very fast access** (order of nanoseconds)
- **High bandwidth** (Gigabytes per second)
- Used to access **hot** data - **all** if economically feasible!

Caches

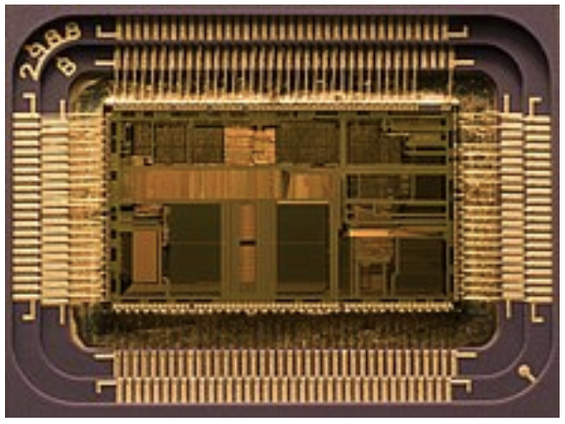


Photo: Wikimedia Commons

- Bits as small **electric** charges
- Typically organized as cache **hierarchy**
- **Very expensive** (hundreds of dollars per Gigabyte)
- **Near-instantaneous** access (few nanoseconds)
- **Very high bandwidth** (tens of Gigabytes per second)
- Used to store **immediately relevant** data

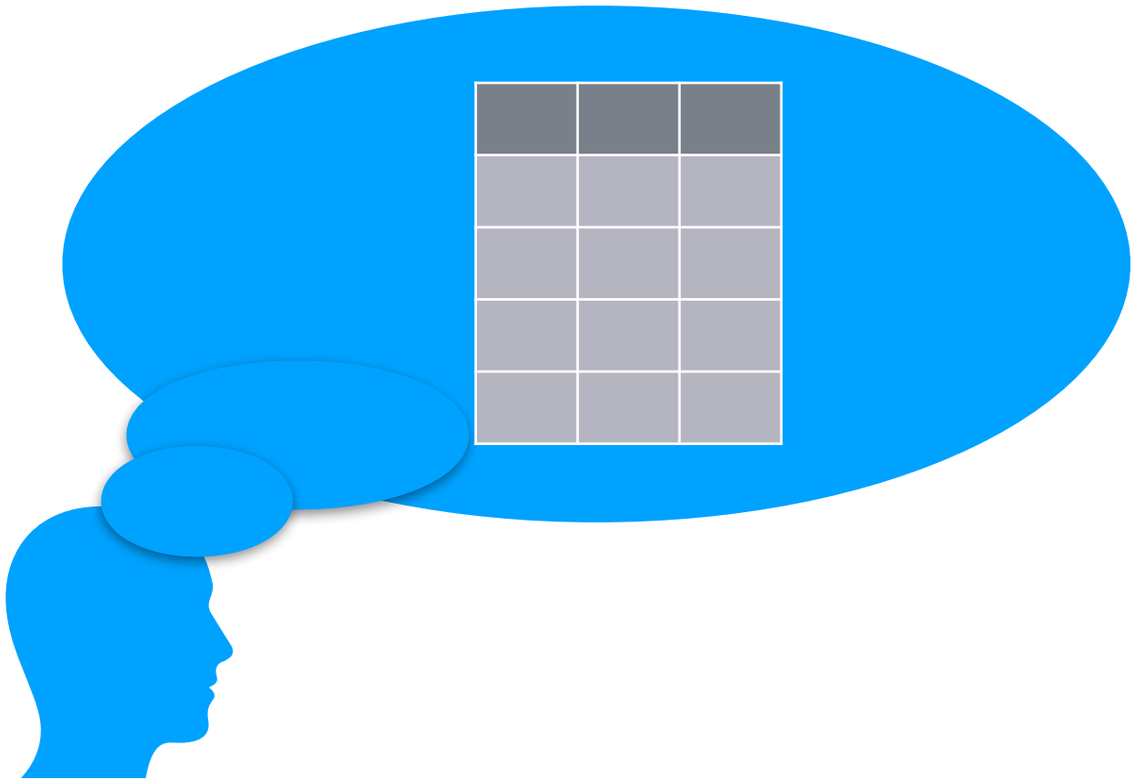
Relevance for DBMS

- **Capacity limits** force data to lower parts of hierarchy
- Data **access speed** may become bottleneck
  - Design algorithms to **minimize data movements**
- **Random** data access is expensive
  - Read data in larger **chunks ("pages")**
  - Keep related data **close** together
- Take into account volatility for **recovery** considerations

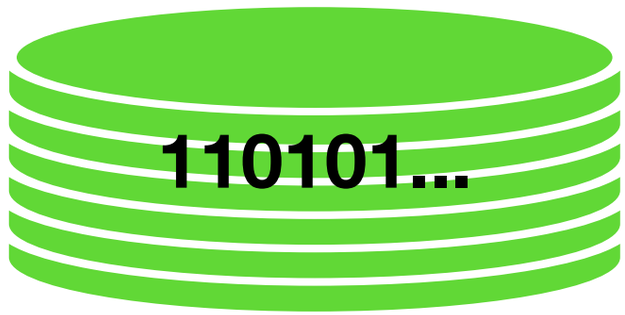
Outline

- Data storage **hardware**
  - What devices to store data on?
- Data storage **format**
  - How to represent relations?

Logical Perspective vs. Physical Storage



User

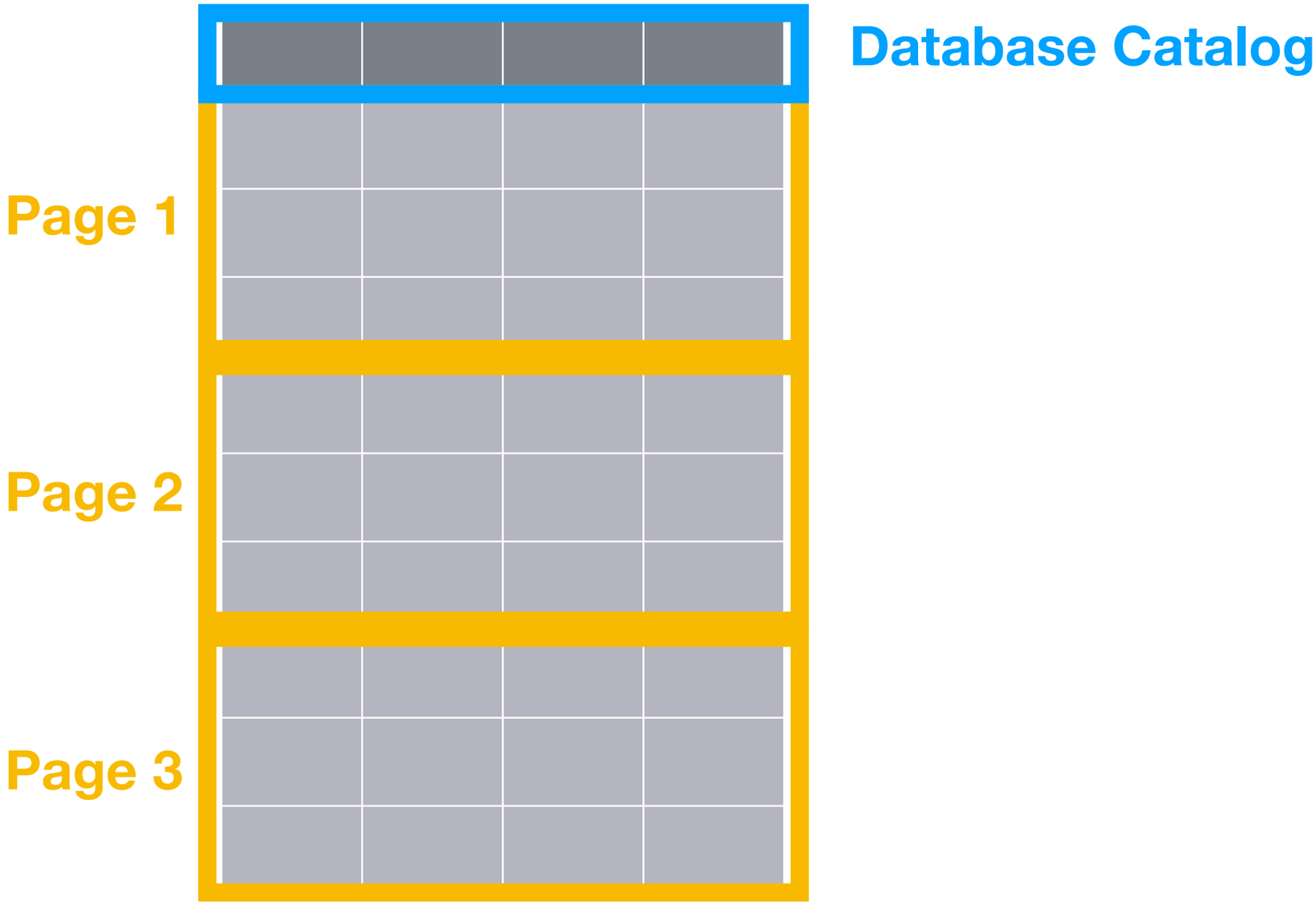


DBMS

Tables as Files

- Table schema information is stored in **database catalog**
- Table content is stored as collection of pages ("**file**")
- Each page typically stores a **few KB** of data
- Enough to store **multiple rows** but not entire table

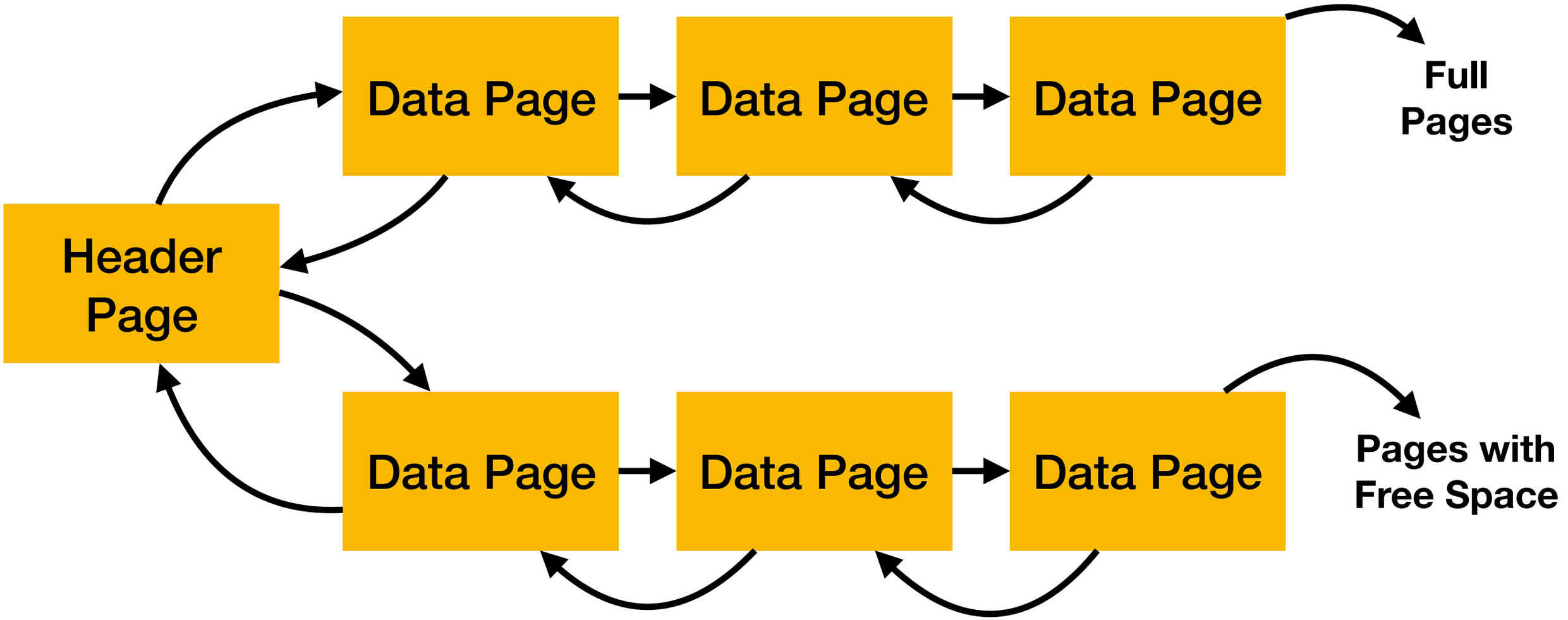
Illustration: Table Storage



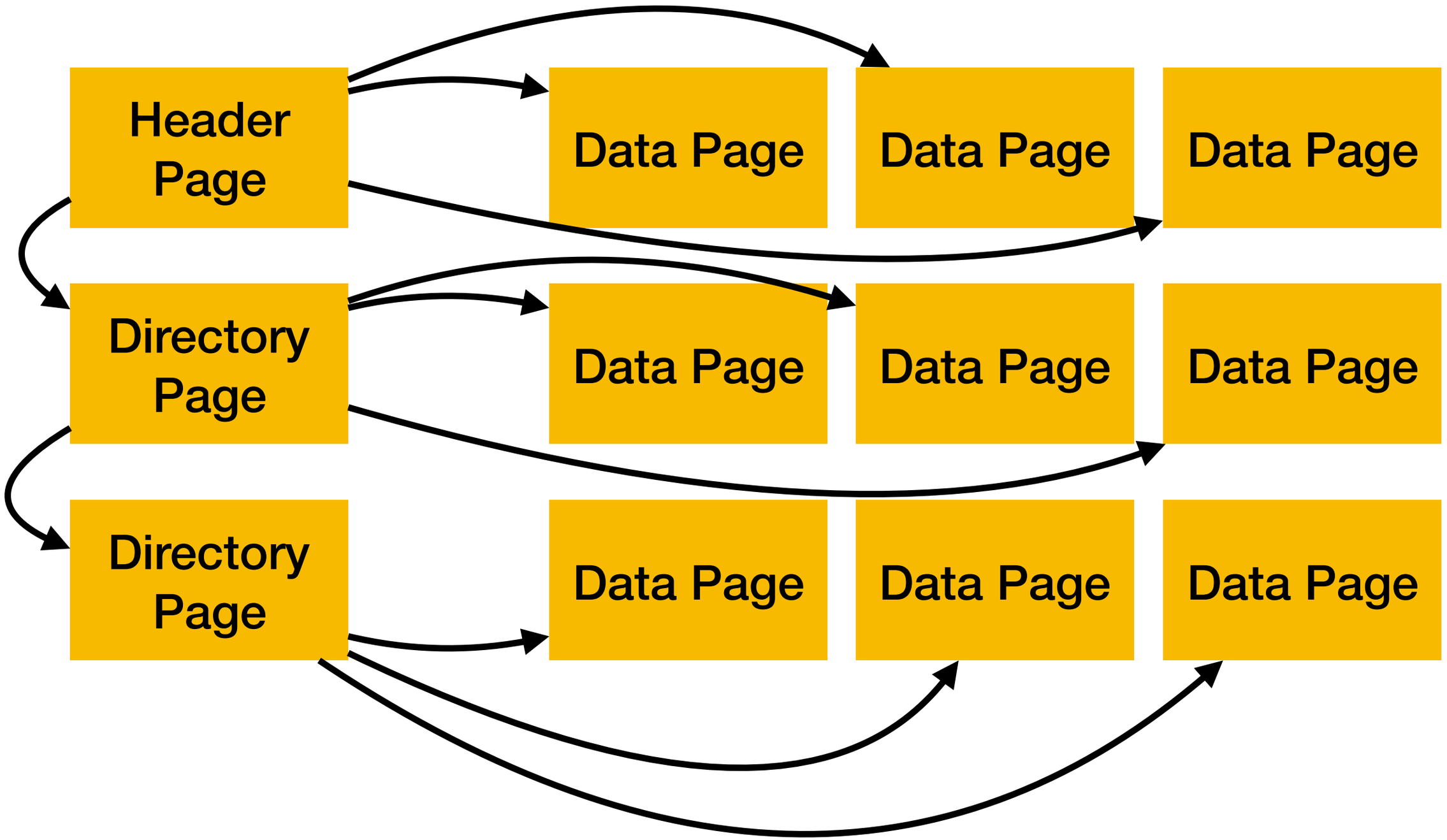
From Files to Pages

- Possibility 1: store pages as **(doubly) linked list**
  - Each page contains **pointers** to next/prior page
  - Can use separate lists for **full/partially empty** pages
  - Reference to **header** page stored in DB catalog
- Possibility 2: **directory** with pointers to pages
  - Directory pages reference **data pages with meta-data**

Files as Linked Lists



Files via Directories



From Pages to Slots

- Pages are divided into **slots**
- Each slot stores one **record** (i.e., table row)
- Can refer to records via **(pageID, slotID)**
- Multiple ways to **divide** pages into slots
- **Fixed**-length vs. **variable**-length records

Fixed-Length Records

- Number of bytes per slot is determined **a-priori**
- Need to keep track of which slots are **used** (insertions ...)
- **Packed** representation uses consecutive slots
  - Only keep track of **number** of slots used
- **Unpacked** representation allows unused slots in-between
  - Need **bitmap** to keep track of used slots

Packed Representation

USED
USED
USED
FREE
FREE
FREE
FREE
FREE
3

What's the Problem with Deletions ... ?

Unpacked Representation

USED
FREE
USED
FREE
USED
FREE
USED
FREE
101010010

Variable-Length Records

- E.g., records with variable-length **text** fields
- Number of bytes per slot is **not fixed** a-priori
- Each page maintains **directory** about used slots
  - Store **first byte** and length of slots
- **Flexibility** to move around records on page
  - Can use that for regular **compaction**

From Slots to Fields

- Must divide each slots into **fields**
- **Fixed** length vs. **variable** length fields
- Fixed length: store field sizes in DB **catalog**
- Variable length: store field sizes on **page**
  - Option 1: use special **delimiter symbol** between fields
  - Option 2: store "**field directory**" at beginning of record

Summary: Files

- Decomposing
  tables into pages,
  pages into slots,
  slots into fields

- **Variable** length versus **fixed** length content

- Variable length content can be handled via **directories**

Row Stores vs. Column Stores

- So far: have seen how to store data **"row-wise"**
  - I.e., data for **same row** is close together
  - This is done by **traditional** DBMS like Postgres
- Can also store data **"column-wise"**
  - I.e., data for **same column** is close together
  - Can help if queries access only **few columns**
  - Will see corresponding **systems** later ...