Lecture 2: Storage Management

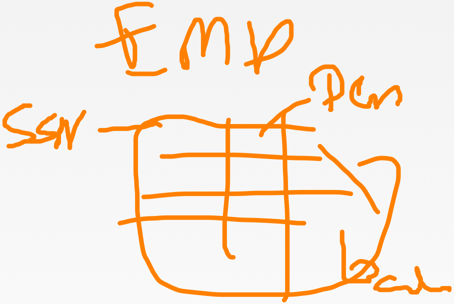
CREATING THE NEXT®

Today's Agenda

Recap

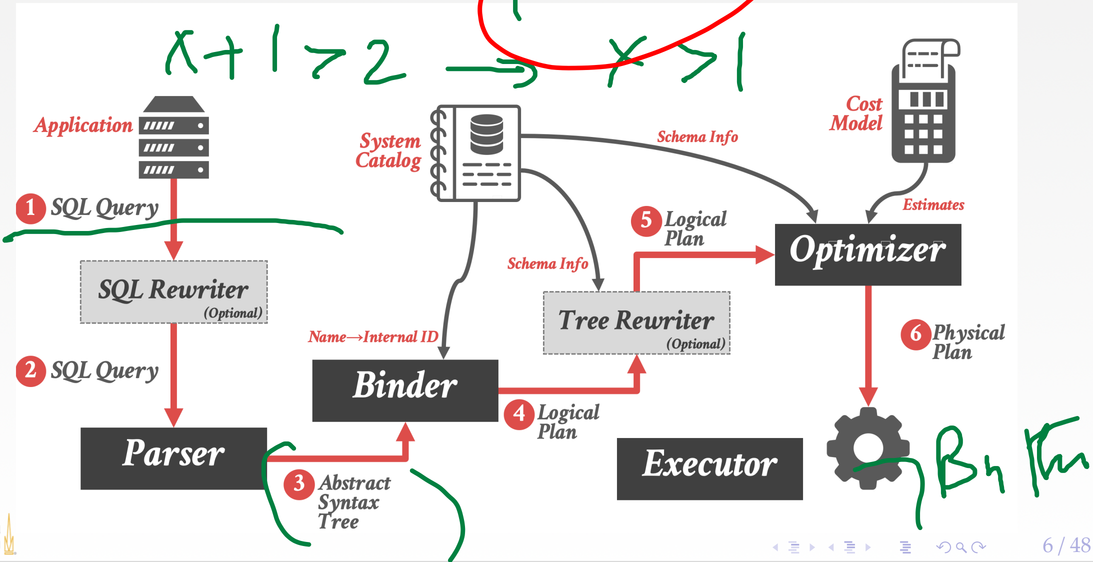
History of Database Systems

- **1960s**: Hierarchical data model (Tree)
- **1970s**: Network data model (Graph)
- **1980s**: Relational data model (Relation)
- **1990s**: Object-Oriented Data Model
- **2000s**: Data Warehouses – OLAP workload
- **2000s**: NoSQL systems
- **2010s**: NewSQL systems
- **2010s**: Hybrid systems – OLTP + OLAP workload
- **2010s**: Cloud systems
- **2020s**: Specialized systems (*e.g.*, Time Series DBMSs, GPU-based DBMSs)



Anatomy of a DBMS

Anatomy of a Database System [Monologue]



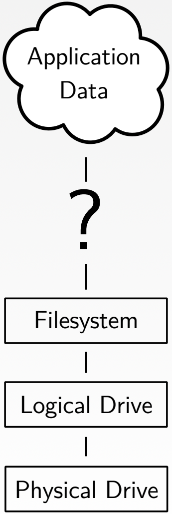
Anatomy of a Database System [Monologue]

- Process Manager
  - Manages client connections
- Query Processor
  - Parse, plan and execute queries on top of storage manager
- Transactional Storage Manager
  - Knits together buffer management, concurrency control, logging and recovery
- Shared Utilities
  - Manage hardware resources across threads

Anatomy of a Database System [Monologue]

- Process Manager
  - Connection Manager + Admission Control
- Query Processor
  - Query Parser
  - Query Optimizer (*a.k.a.*, Query Planner)
  - Query Executor
- Transactional Storage Manager
  - Lock Manager
  - Access Methods (*a.k.a.*, Indexes)
  - Buffer Pool Manager
  - Log Manager
- Shared Utilities
  - Memory, Disk, and Networking Manager

The Problem



Custom OS

- HW
- Only SW

Requirements

There are different classes of requirements:

- Data Independence
  - application logic must be shielded from physical storage implementation details
  - physical storage can be reorganized
  - hardware can be changed
- Scalability
  - must scale to (nearly) arbitrary data size
  - efficiently access to individual tuples
  - efficiently update an arbitrary subset of tuples
- Reliability
  - data must never be lost
  - must cope with hardware and software failures
- ...

Layered Architecture

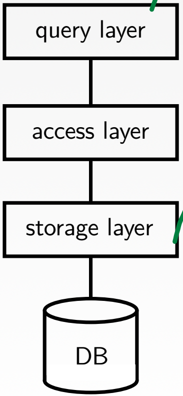
- implementing all these requirements on “bare metal” is hard
- and not desirable
- a DBMS must be maintainable and extensible

Instead: use a **layered architecture**

- the DBMS logic is split into levels of functionality
- each level is implemented by a specific layer
- each layer interacts only with the next lower layer
- simplifies and modularizes the code

A Simple Layered Architecture

Purpose		Access Granularity
		declarative queries sets of records
query translation and optimization	query layer	
managing records and access paths	access layer	records
DB buffer and hardware interface	storage layer	page



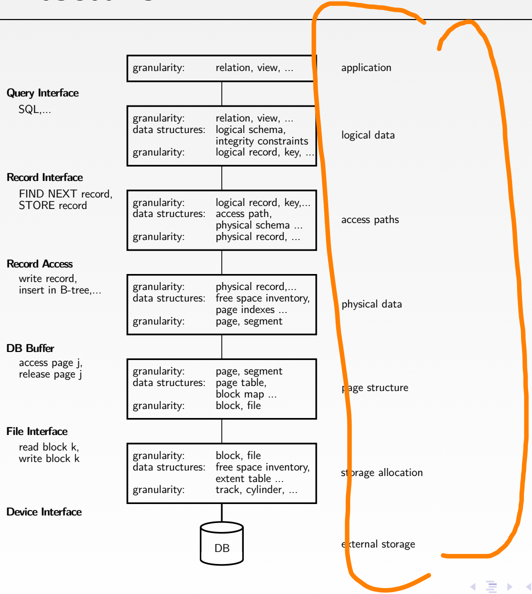
index

A Simple Layered Architecture (2)

- layers can be characterized by the data items they manipulate
- lower layer offers functionality for the next higher level
- keeps the complexity of individual layers reasonable
- rough structure: physical $\rightarrow$ low level $\rightarrow$ high level

This is a reasonable architecture, but simplified.
A more detailed architecture is needed for a complete DBMS.

A More Detailed Architecture



Hardware Properties

Impact of Hardware

Must take hardware properties into account when designing a storage system.

For a long time dominated by **Moore's Law**:

The number of transistors on a chip doubles every 18 month.

Indirectly drove a number of other parameters:

- main memory size
- CPU speed
  - no longer true!
- HDD capacity
  - start getting problematic, too. density is very high
  - only capacity, not access time

Memory Hierarchy

capacity
latency

bytes
1ns

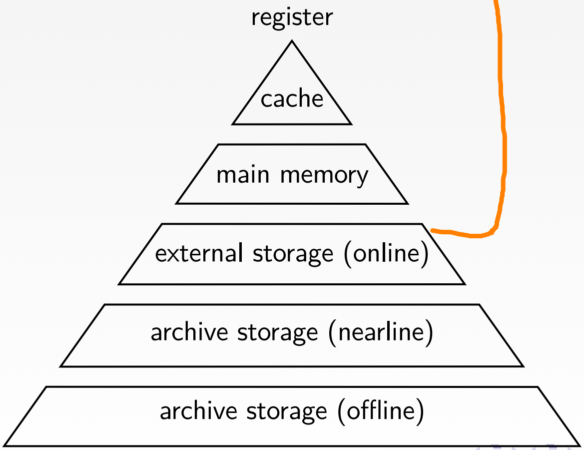
K-M bytes
$<$10ns

G bytes
$<$100ns

T bytes
ms

T bytes
sec

T-P bytes
sec-min



Georgia Tech

Memory Hierarchy (2)

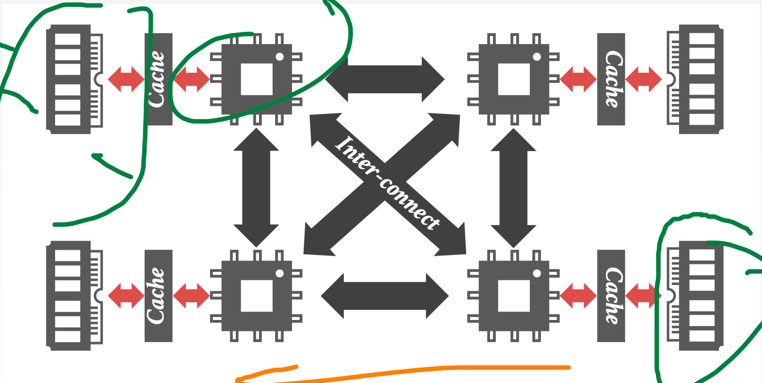
There are huge gaps between hierarchy levels

- traditionally, main memory vs. disk is most important
- but memory vs. cache etc. also relevant

The DBMS must aim to maximize locality.

Georgia Tech

Non-Uniform Memory Access



Hard Disk Access

Hard Disks are still the dominant external storage:

- rotating platters, mechanical effects
- transfer rate: ca. 150MB/s
- seek time ca. 3ms
- huge imbalance in random vs. sequential I/O!

Hard Disk Access (2)

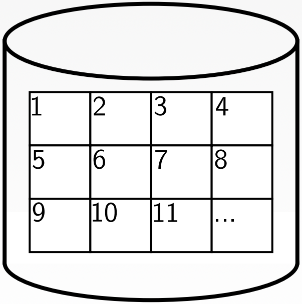
The DBMS must take these effects into account

- sequential access is much more efficient
- traditional DBMSs are designed to maximize sequential access
- gap is growing instead of shrinking
- even SSDs are slightly asymmetric (and have other problems)
- DBMSs try to reduce number of writes to random pages by organizing data in **contiguous blocks**.
- Allocating multiple pages at the same time is called a **segment**

Hard Disk Access (3)

Techniques to speed up disk access:

- do not move the head for every single tuple
- instead, load larger chunks. typical granularity: one **page**
- page size varies. traditionally 4KB, nowadays often 16K and more (**trade-off**)



Hard Disk Access (4)

The page structure is very prominent within the DBMS

- granularity of I/O
- granularity of buffering/memory management
- granularity of recovery

Page is still too small to hide random I/O though

- sequential page access is important
- DBMSs use read-ahead techniques
- asynchronous write-back

Database System Architectures

Storage Management

Disk-Centric Database System

- The DBMS assumes that the primary storage location of the database is HDD.

Memory-Centric Database System (**MMDB**)

- The DBMS assumes that the primary storage location of the database is DRAM.

2010s

Buffer Management

The DBMS's components manage the movement of data between non-volatile and volatile storage.

Access Times

Access Time	Hardware	Scaled Time
0.5 ns	L1 Cache	0.5 sec
7 ns	L2 Cache	7 sec
100 ns	DRAM	100 sec
350 ns	NVM	6 min
150 us	SSD	1.7 days
10 ms	HDD	16.5 weeks
30 ms	Network Storage	11.4 months
1 s	Tape Archives	31.7 years

Source: Latency numbers every programmer should know

Storage Management

Storage Manager

- The **storage manager** is responsible for maintaining a database's files.
  - Some do their own **scheduling** of I/O operations to improve spatial and temporal locality of pages.
- It organizes the files as a collection of pages.
  - Tracks data being read from and written to pages.
  - Tracks the available free space.

Database Pages

- A **page** is a fixed-size block of data.
  - It can contain tuples, meta-data, indexes, log records . . .
  - Most systems do not mix page types.
  - Some systems require a page to be self-contained. Why?
- Each page is given a unique identifier.
  - The DBMS uses an **indirection layer** to map page ids to physical locations.
  - This is implemented as a page directory table.

 Storage Management

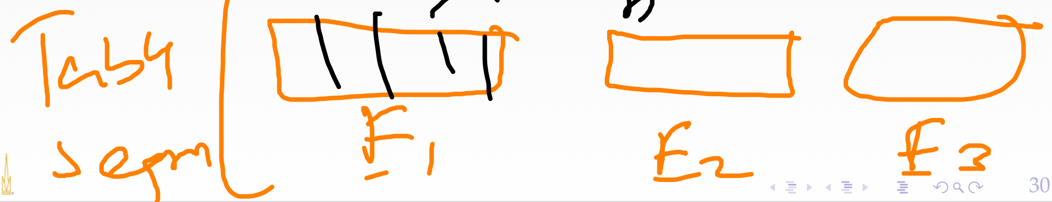
Database Pages

- There are three different notions of "pages" in a DBMS:
  - Hardware Page (usually 4 KB)
  - OS Page (usually 4 KB)
  - Database Page (512 B – 16 KB)
- By hardware page, we mean at what level the device can guarantee a "failsafe write".

Disk Block Mapping

The units of database space allocation are **disk blocks, extents, and segments**.

- A disk block is the smallest unit of data used by a database.
- An extent is a logical unit of database storage space allocation made up of a number of **contiguous** disk blocks.
- A segment is made up of one or more extents (and is hence not always **contiguous** on disk).



Georgia Tech

System Catalog

- A DBMS stores **meta-data** about databases in its internal catalog.
  - List of tables, columns, indexes, views
  - List of users, permissions
  - Internal statistics (*e.g.*, disk reads, storage space allocation)
- Almost every DBMS stores their catalog as a **private database**.
  - Wrap object abstraction around tuples.
  - Specialized code for "bootstrapping" catalog tables. Why?

Data Representation

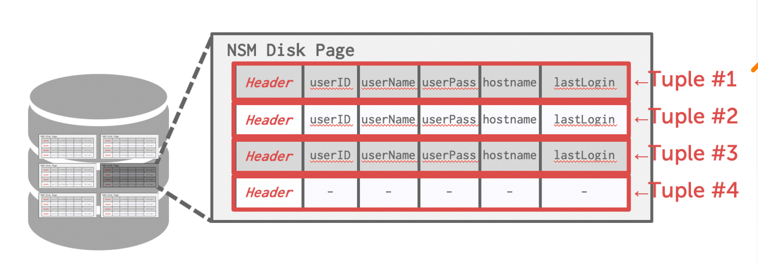
- INTEGER/BIGINT/SMALLINT/TINYINT
  - C/C++ Representation
- FLOAT/REAL vs. NUMERIC/DECIMAL
  - IEEE-754 Standard / Fixed-point Decimals
- VARCHAR/VARBINARY/TEXT/BLOB
  - Header with length, followed by data bytes.
- TIME/DATE/TIMESTAMP
  - 32/64-bit integer of (micro)seconds since Unix epoch

 Storage Management

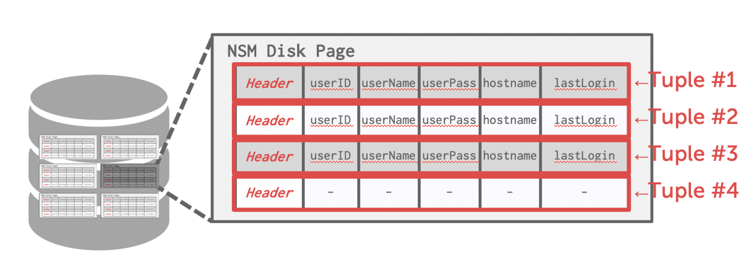
N-ary Storage Model (NSM)

row store

- The DBMS stores all attributes for a single tuple contiguously in a page.



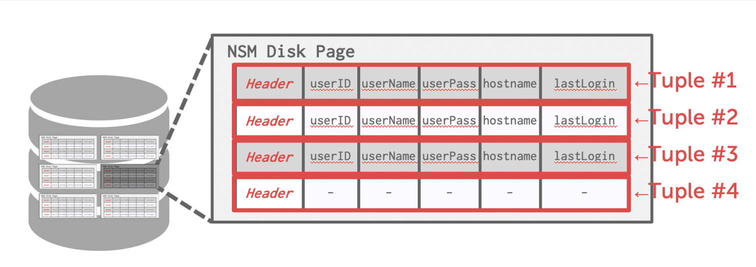
N-ary Storage Model (NSM)




```
SELECT * FROM useracct
WHERE userName = ? AND userPass = ?
```

Use index to access the particular user's tuple.

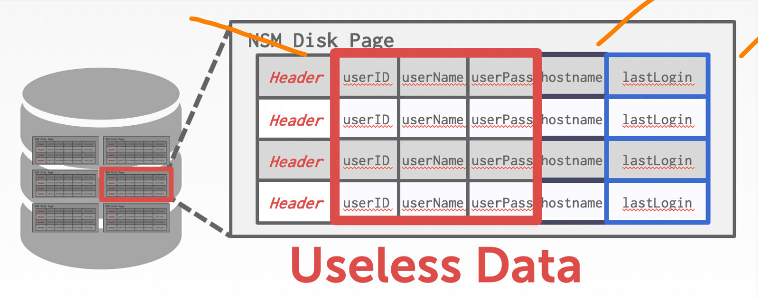
N-ary Storage Model (NSM)




```
INSERT INTO useracct VALUES (?,?,...?)
```

Add the user's tuple using `std::memcpy`.

N-ary Storage Model (NSM)




```
SELECT COUNT(U.lastLogin), EXTRACT(month FROM U.lastLogin) AS month
FROM useracct AS U
WHERE U.hostname LIKE '%.gov'
GROUP BY EXTRACT(month FROM U.lastLogin)
```

Useless data accessed for this query.

N-ary Storage Model (NSM)

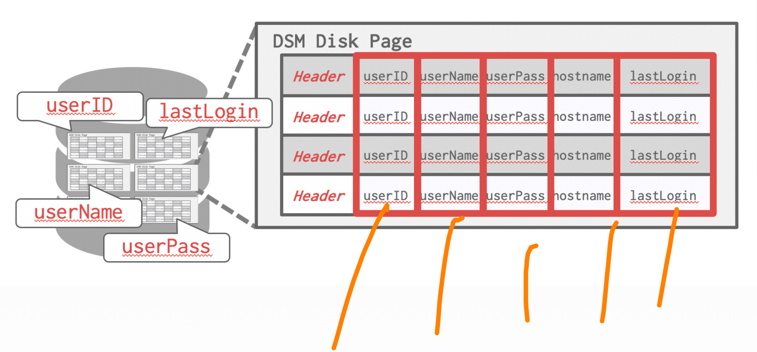
- Advantages
  - Fast inserts, updates, and deletes.
  - Good for queries that need the entire tuple.
- Disadvantages
  - Not good for scanning large portions of the table and/or a subset of the attributes.

Decomposition Storage Model (DSM)

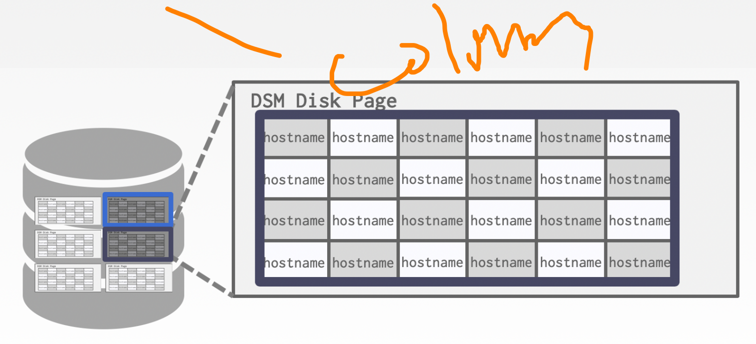
- The DBMS stores the values of a **single attribute** for all tuples contiguously in a page.
  - Also known as a "column store".
- Ideal for OLAP workloads where read-only queries perform large scans over a subset of the table's attributes.

Decomposition Storage Model (DSM)

- The DBMS stores the values of a single attribute for all tuples contiguously in a page.
  - Also known as a "column store".



Decomposition Storage Model (DSM)




```sql
SELECT COUNT(U.lastLogin), EXTRACT(month FROM U.lastLogin) AS month
FROM useracct AS U
WHERE U.hostname LIKE '%.gov'
GROUP BY EXTRACT(month FROM U.lastLogin)
```

Workload Characterization

- On-Line Transaction Processing (OLTP)
  - Fast operations that only read/update a small amount of data each time.
  - OLTP Data Silos
- On-Line Analytical Processing (OLAP)
  - Complex queries that read a lot of data to compute aggregates.
  - OLAP Data Warehouse
- Hybrid Transaction + Analytical Processing
  - OLTP + OLAP together on the same database instance

Workload Characterization

Workload	Operation Complexity	Workload Focus
OLTP	Simple	Writes
OLAP	Complex	Reads
HTAP	Medium	Mixture

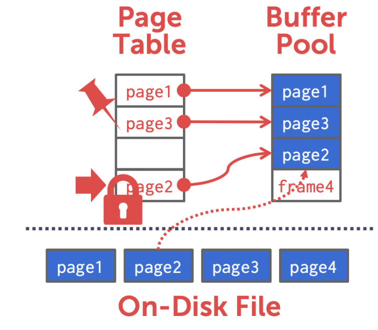
Source

Storage Models

- A DBMS encodes and decodes the tuple's bytes into a set of attributes based on its schema.
- It is important to choose the right storage model for the target workload
  - OLTP $\longrightarrow$ Row-Store
  - OLAP $\longrightarrow$ Column-Store

Buffer Pool Meta-Data

- The **page table** keeps track of pages that are currently in memory.
- Also maintains additional meta-data per page:
  - **Dirty Flag**
  - **Pin/Reference Counter**



 Storage Management

Buffer Replacement Policies

- When the DBMS needs to free up a frame to make room for a new page, it must decide which page to evict from the buffer pool.
- Policies:
  - FIFO
  - LFU
  - LRU
  - CLOCK
  - LRU-k
  - 2Q

Scan-resistant

Conclusion

Parting Thoughts

- Database systems have a layered architecture.
- Design of database system components affected by hardware properties.
- Database is physically organized as a collection of pages on disk.
- The units of database space allocation are disk blocks, extents, and segments
- The DBMS can manage that sweet, sweet memory better than the OS.
- Leverage the semantics about the query plan to make better decisions.
- It is important to choose the right storage model for the target workload

Georgia Tech

Next Class

- Recap of access methods
- Submit exercise sheet #1 6 via Gradescope.