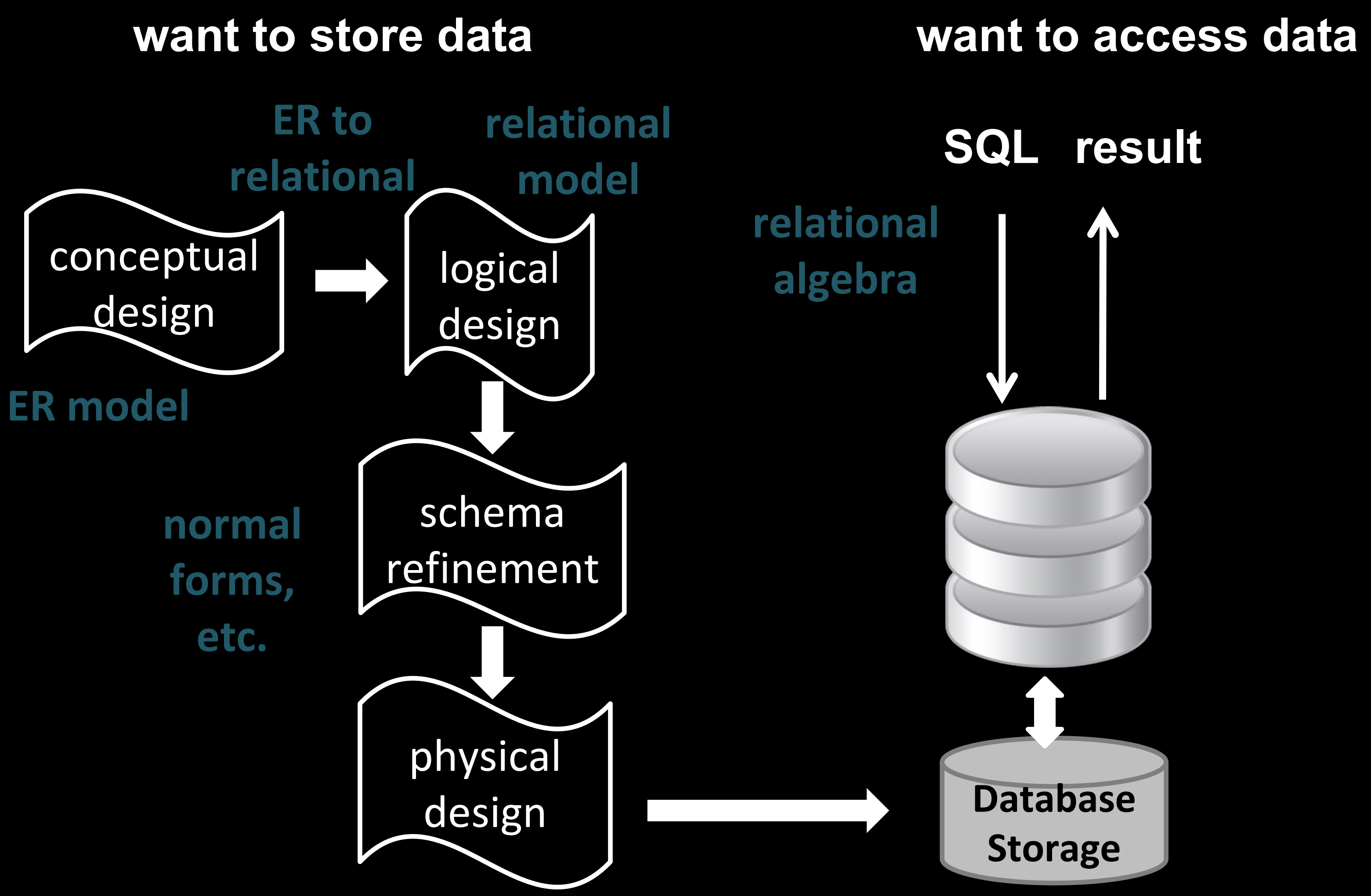
Introduction to Database Systems IDBS – Spring 2024

- Week 9:
- Storage Models
- Architecture of a DBMS
- Main-Memory DBMSs

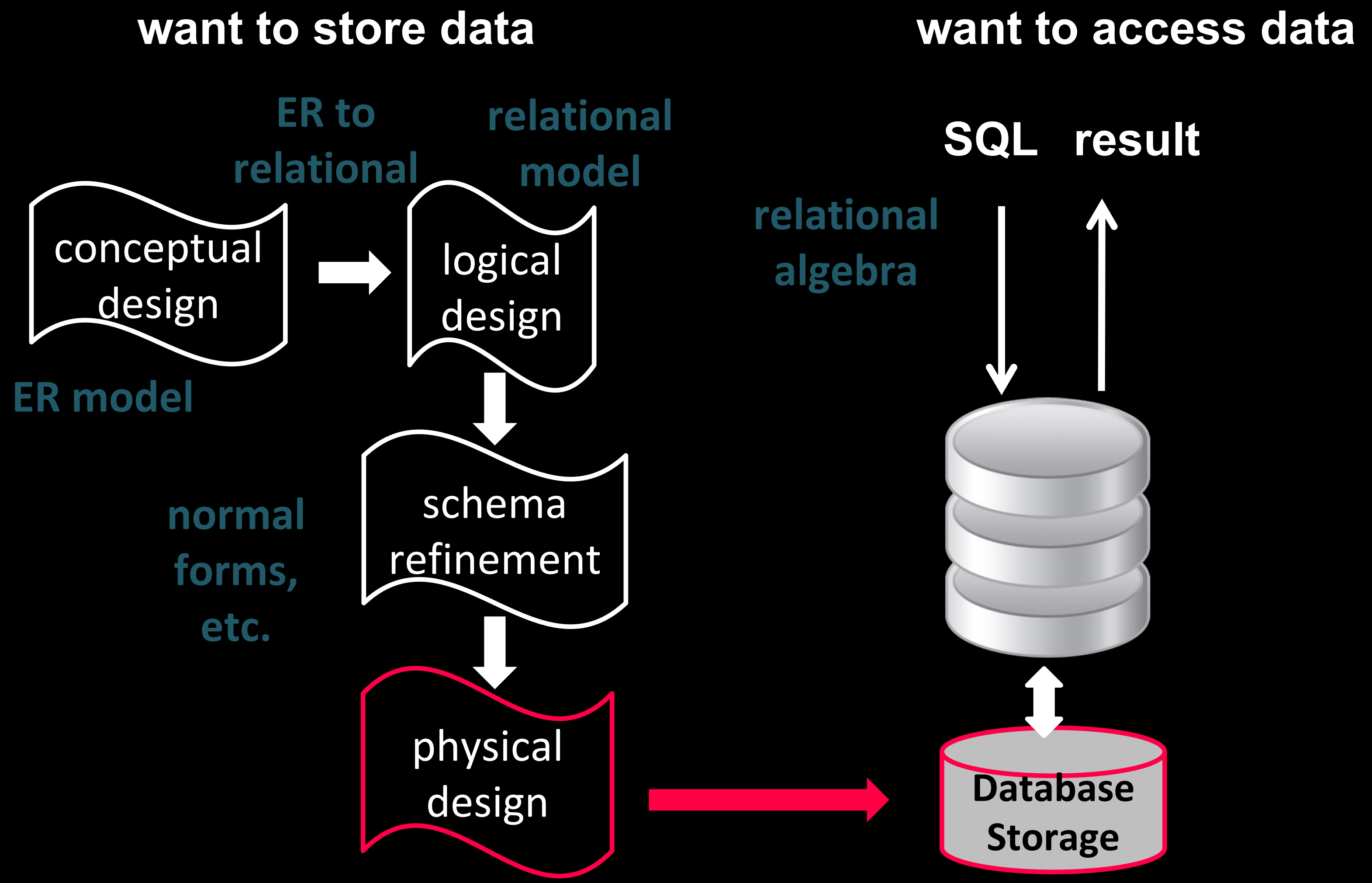
Eleni Tzirita Zacharatou

Readings:
PDBM 2 + more material on LearnIT

The Bigger Picture



The Bigger Picture



Today we will talk about the lowest physical representation of data in a database

Storage Models

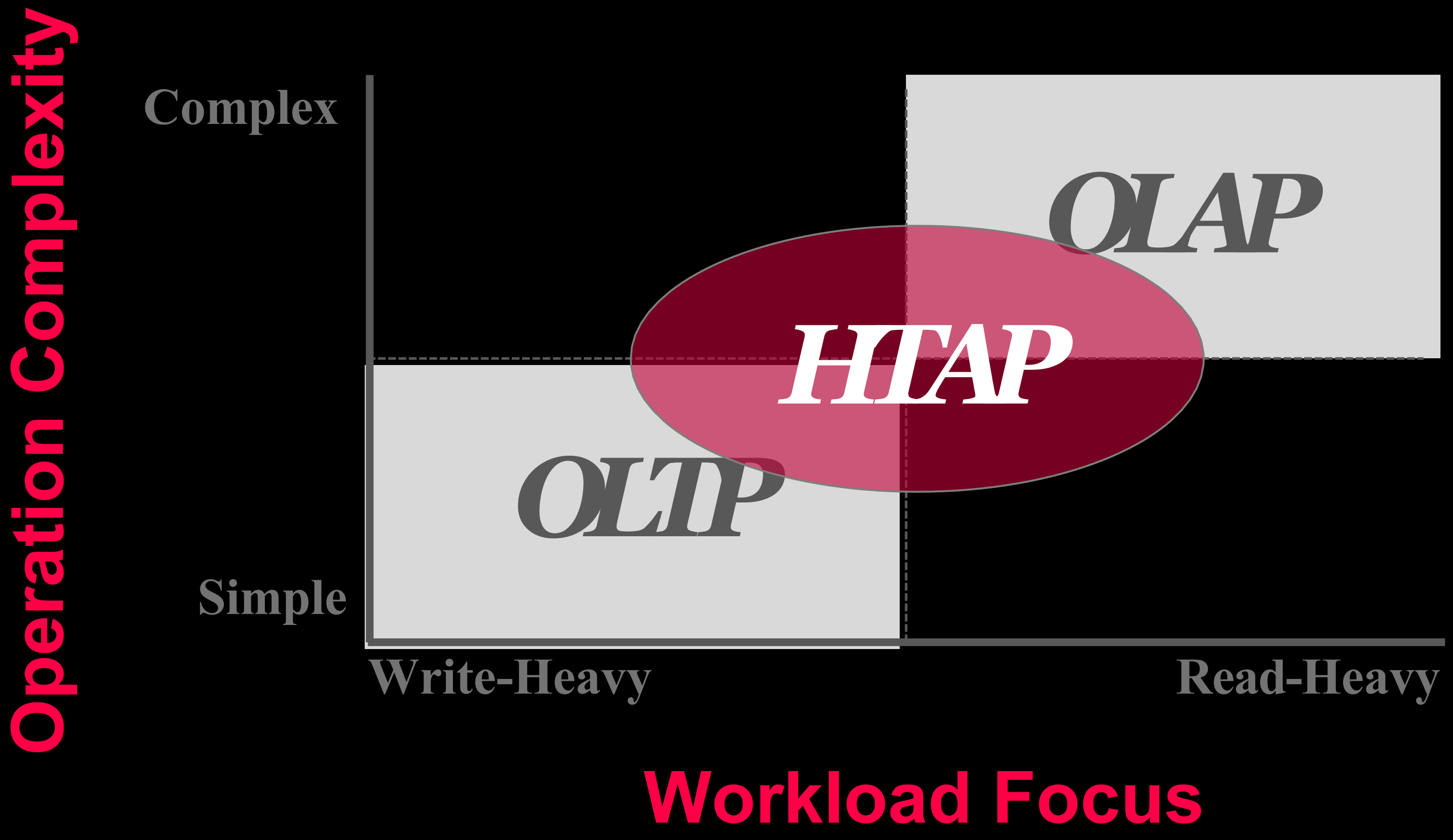
Database Workloads

On-Line Transaction Processing (OLTP)
→ Fast operations that only read/update a small amount of data each time

On-Line Analytical Processing (OLAP)
→ Complex queries that read a lot of data to compute aggregates

Hybrid Transaction + Analytical Processing (HTAP)
→ OLTP + OLAP together on the same database instance

Database Workloads



Source: Mike Stonebraker

OLTP

On-Line Transaction Processing:
→ Simple queries that read/update a small amount of data that is related to a single entity in the database.

This is usually the kind of application that people build first.

```
SELECT P.*, R.*
  FROM pages AS P
 INNER JOIN revisions AS R
    ON P.latest = R.revID
 WHERE P.pageID = ?
```

```
UPDATE useracct
   SET lastLogin = NOW(),
       hostname = ?
 WHERE userID = ?
```

```
INSERT INTO revisions
VALUES(?,?…,?)
```

OLAP

On-Line Analytical Processing:
→ Complex queries that read large portions of the database spanning multiple entities.

You execute these workloads on the data you have collected from your OLTP application(s).

```
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)
```

Exercise

Consider the following relation:

- Sales (ProductID, ProductName, Date, Quantity, Revenue)

Query A

```
SELECT ProductID, SUM(Quantity)
AS TotalQuantitySold
FROM Sales
WHERE Date BETWEEN '2023-01-01'
AND '2023-12-31'
GROUP BY ProductID;
```

Query B

```
UPDATE Sales
SET Revenue = Revenue * 1.1
WHERE ProductID = 12345 AND
Date = '2023-03-10';
```

Observation

The relational model *does not* specify that the DBMS must store all a tuple's attributes together on a single page.

This *may not* be the best layout for some workloads.

Storage Models

A DBMS's *storage model* specifies how it physically organizes tuples on disk and in memory.

→ Can have different performance characteristics based on the target workload (OLTP vs. OLAP).
→ Influences the design choices of the rest of the DBMS.

Choice #1: N-ary Storage Model (NSM)
Choice #2: Decomposition Storage Model (DSM)
Choice #3: Hybrid Storage Model (PAX)

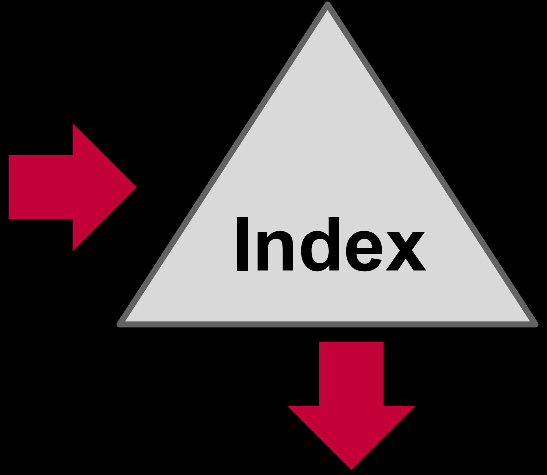
N-ary Storage Model (NSM)

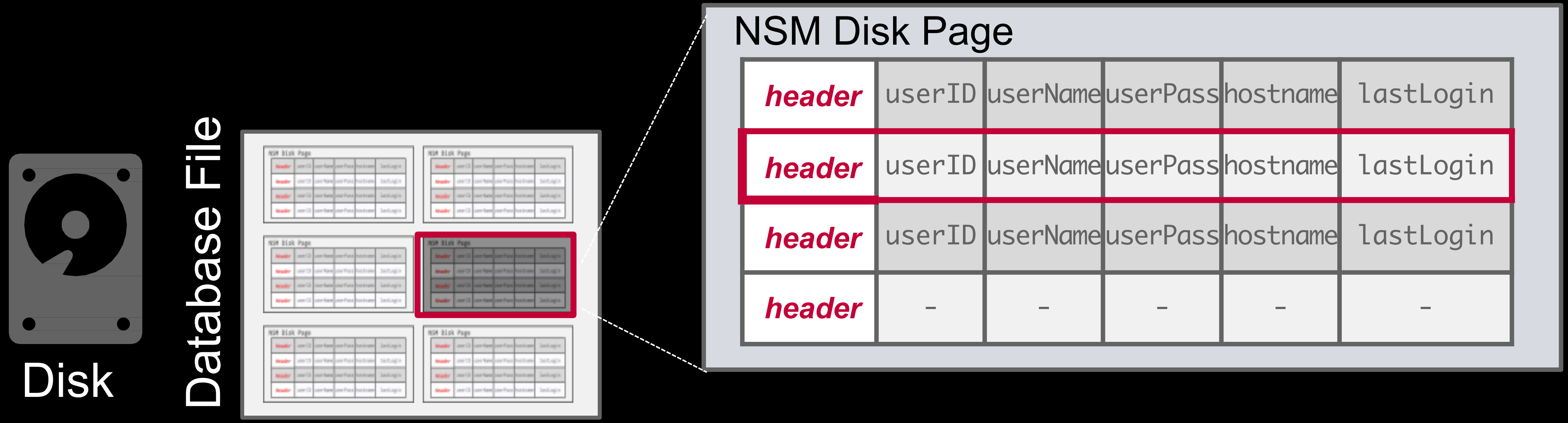
The DBMS stores (almost) all attributes for a single tuple contiguously on a single page.
→ Also known as a "row store"

Ideal for OLTP workloads where queries are more likely to access individual entities and execute write-heavy workloads.

NSM: OLTP Example

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

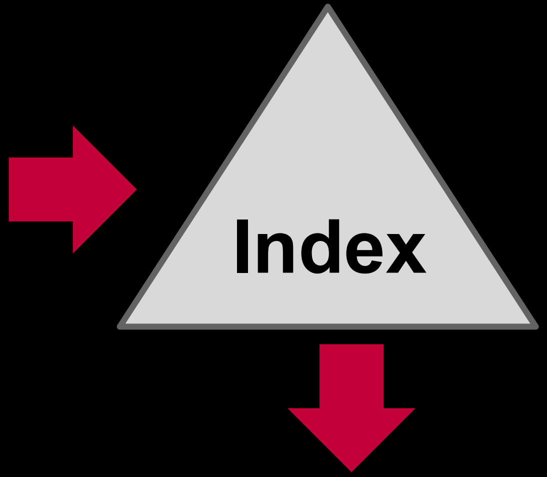


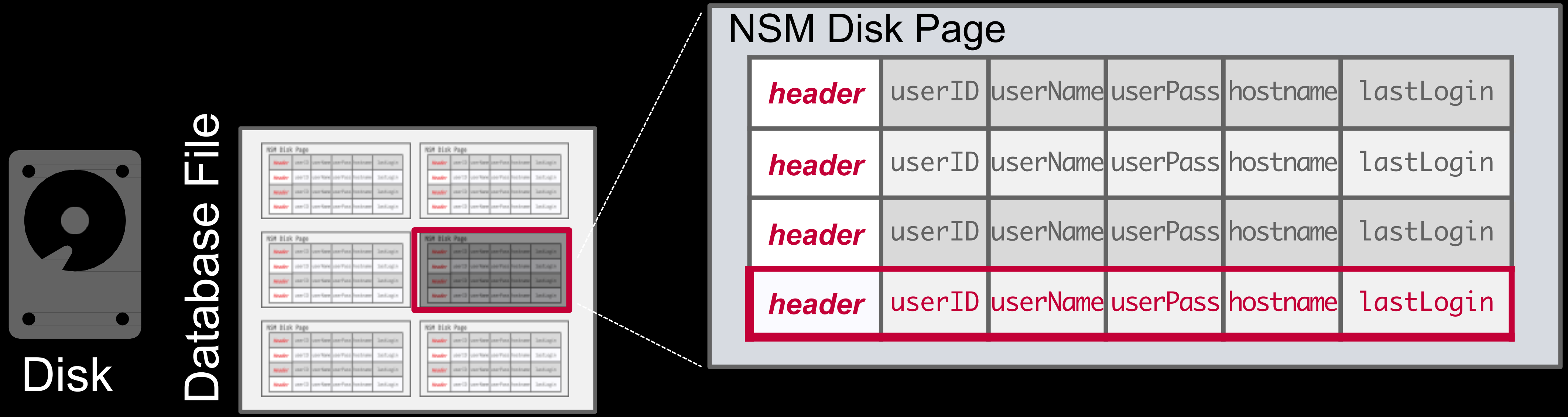


NSM: OLTP Example

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

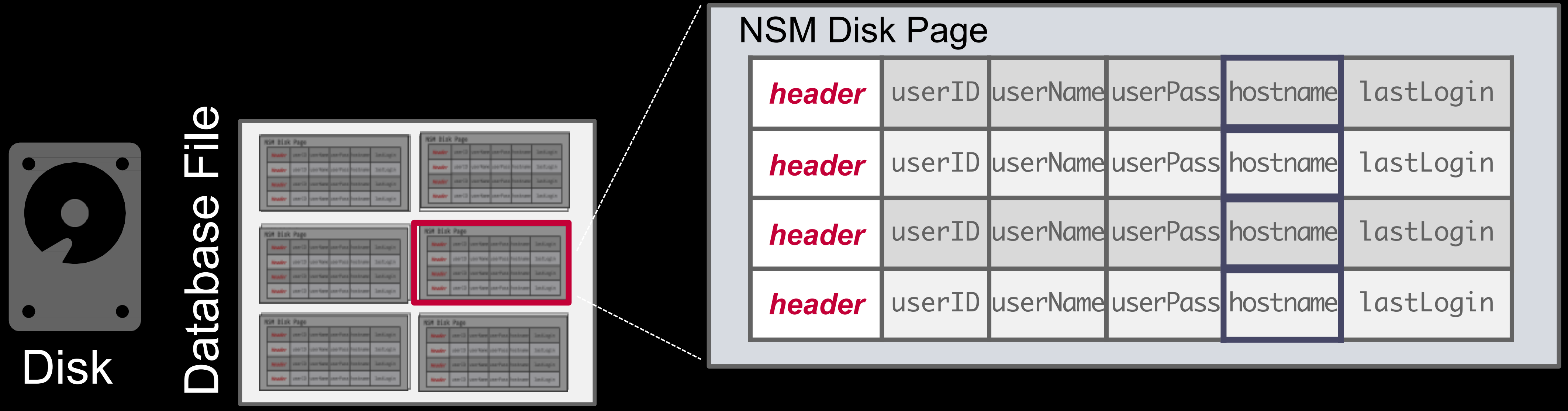
```sql
INSERT INTO useracct
VALUES (?,?,…?)
```





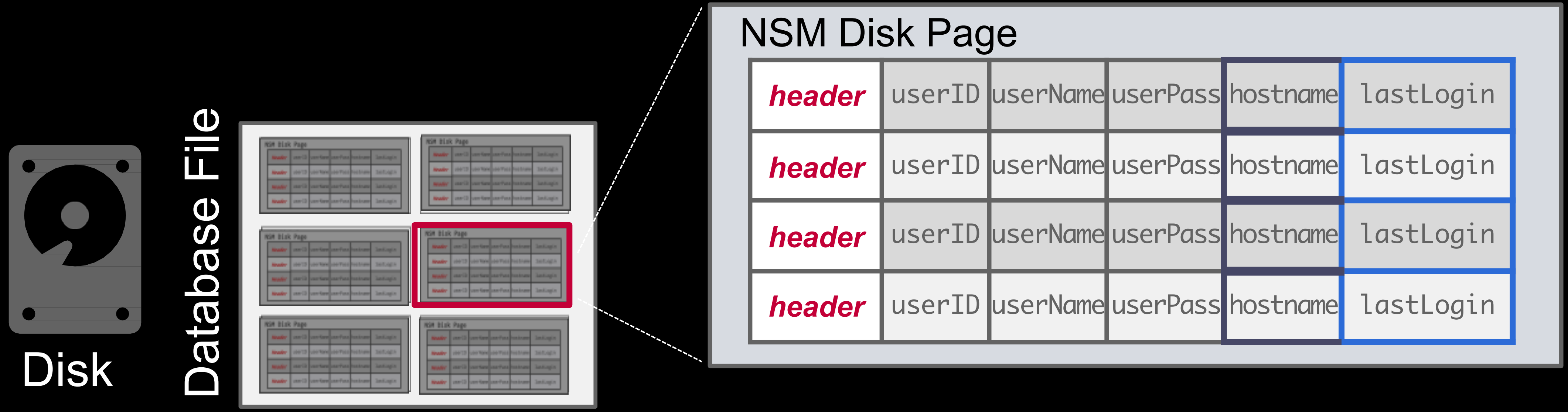
NSM: OLAP Example

```
 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)
```



NSM: OLAP Example

```
    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)
```

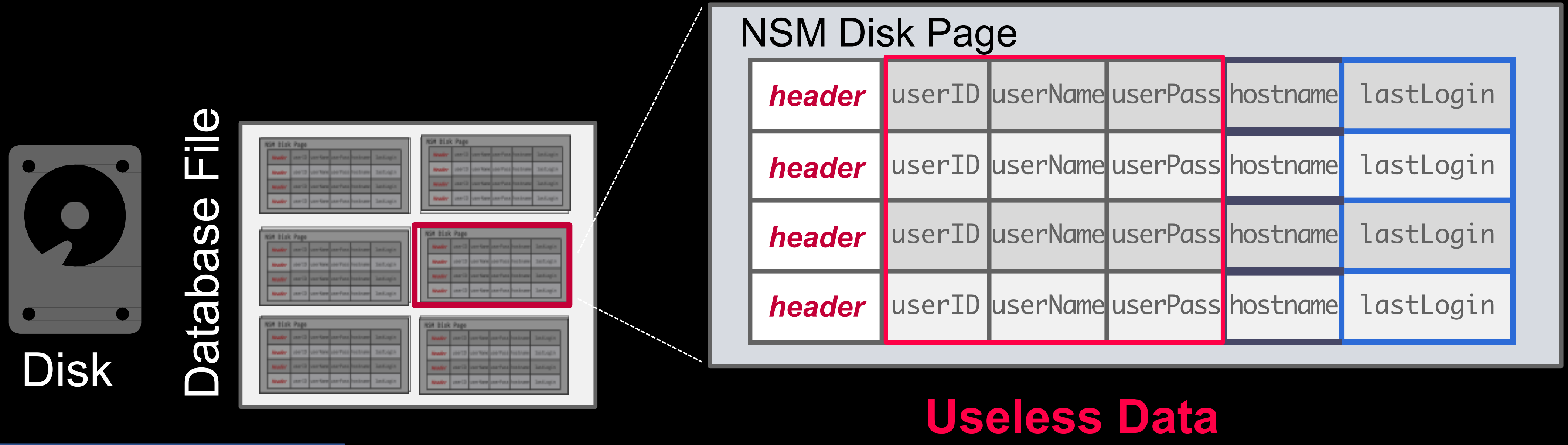


Reflection

- Do you see a problem in the previous OLAP example?
- Is NSM a good choice for OLAP queries?
- Can you think of some drawbacks of NSM?

NSM: OLAP Example

```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)
```



NSM: Summary

Advantages

→ Fast inserts, updates, and deletes.
→ Good for queries that need the entire tuple (OLTP).
→ Can use index-oriented physical storage for clustering.

Disadvantages

→ Not good for scanning large portions of the table and/or a subset of the attributes.
→ Terrible memory locality in access patterns.
→ Not ideal for compression because of multiple value domains within a single page.

Decomposition Storage Model (DSM)

The DBMS stores a single attribute for all tuples contiguously in a block of data.
→ 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.

DBMS is responsible for combining/splitting a tuple's attributes when reading/writing.

DSM: Database Example

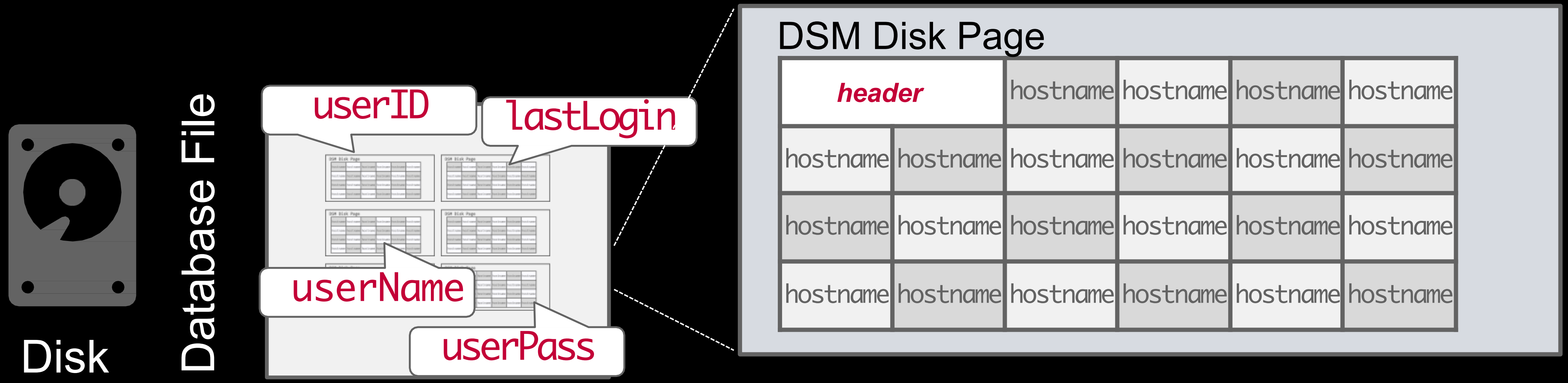
The DBMS stores the values of a single attribute across multiple tuples contiguously in a page.

→ Also known as a "column store".

header	userID	userName	userPass	hostname	lastLogin
header	userID	userName	userPass	hostname	lastLogin
header	userID	userName	userPass	hostname	lastLogin
header	userID	userName	userPass	hostname	lastLogin

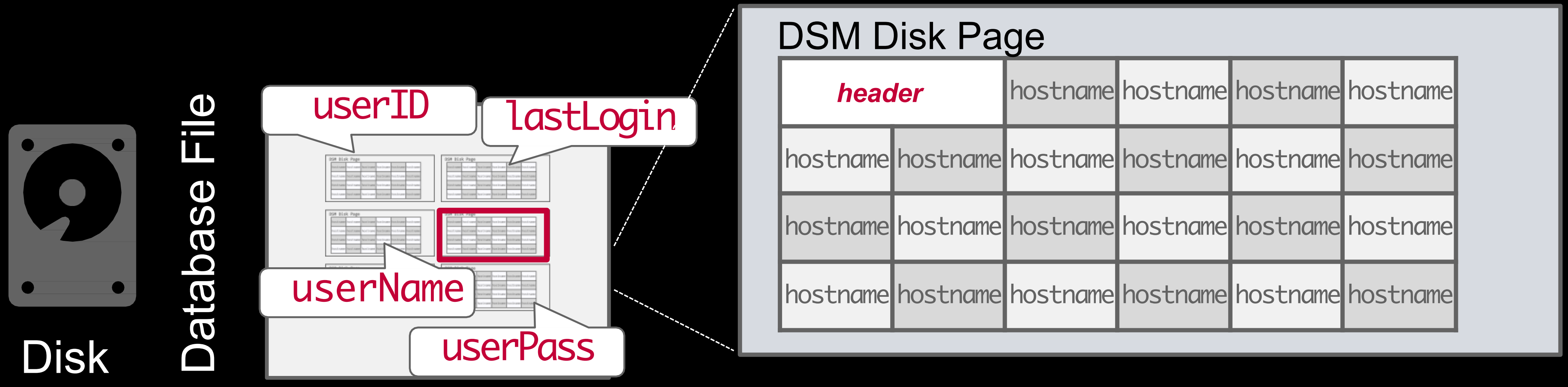
DSM: Database Example

The DBMS stores the values of a single attribute across multiple tuples contiguously in a page.
→ Also known as a "column store".



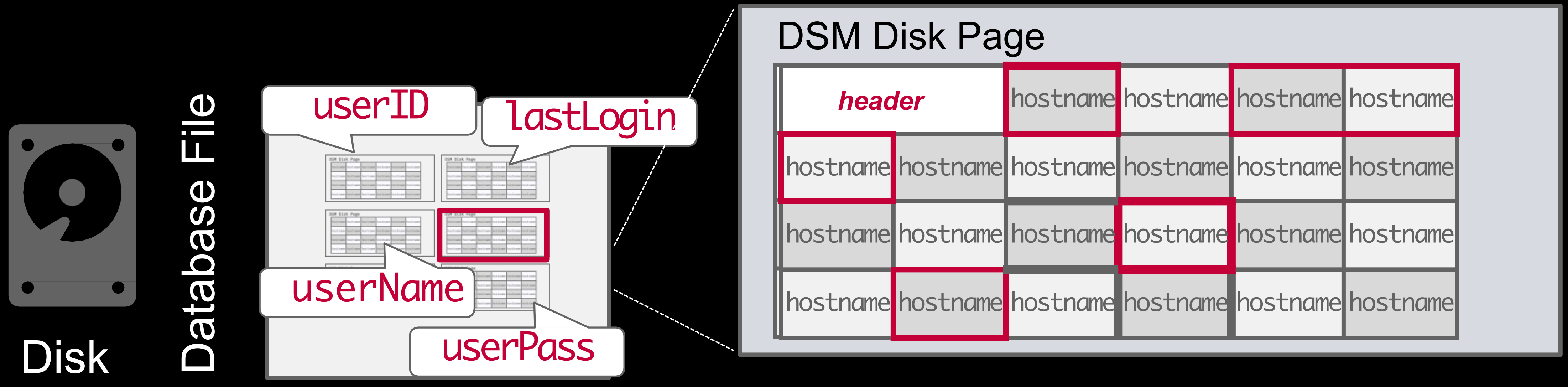
DSM: OLAP Example

```
    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)
```



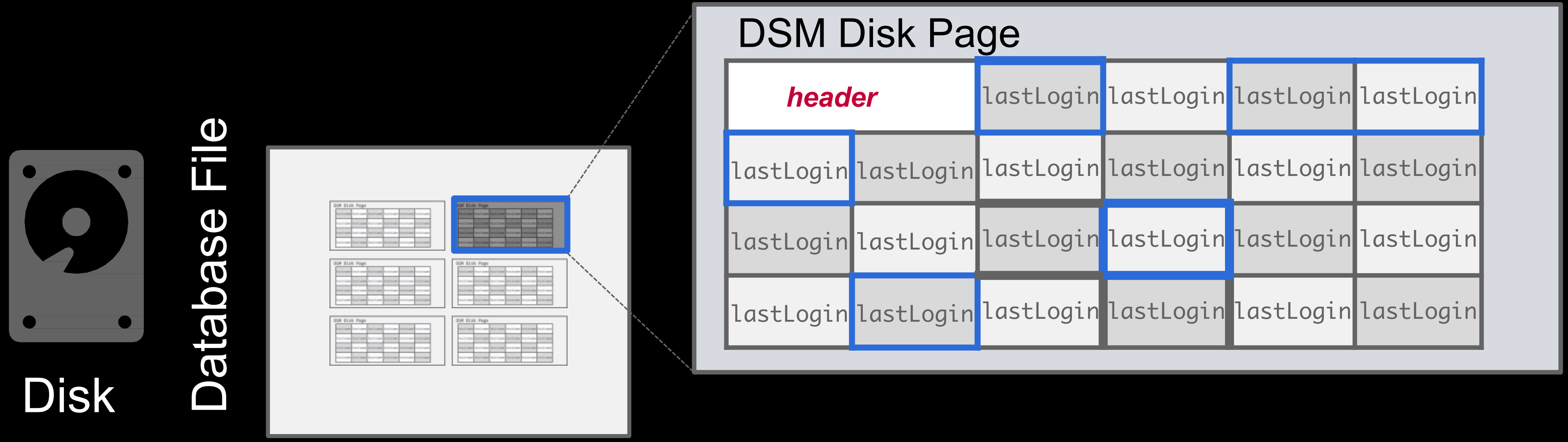
DSM: OLAP Example

```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)
```



DSM: OLAP Example

```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)
```



Reflection

- Is DSM a good choice for OLTP queries?

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

- Can you think of some drawbacks of DSM?

DSM: Summary

Advantages

→ Reduces the amount wasted I/O per query because the DBMS only reads the data that it needs.

→ Faster query processing because of increased locality and cached data reuse.

→ Better data compression.

Disadvantages

→ Slow for point queries, inserts, updates, and deletes because of tuple splitting/stitching/reorganization.

Optimal Layout

NSM	DSM
easy to implement	have tuple reconstruction cost
good if you are accessing the whole tuple or most columns of a tuple	good if you are accessing one or a few columns of a tuple

Choice depends on the workload!

Systems in the Wild

row stores

column stores

big vendors with both row & column store offerings

DBMS Architecture

Main-Memory DBMSs

DBMS Architecture

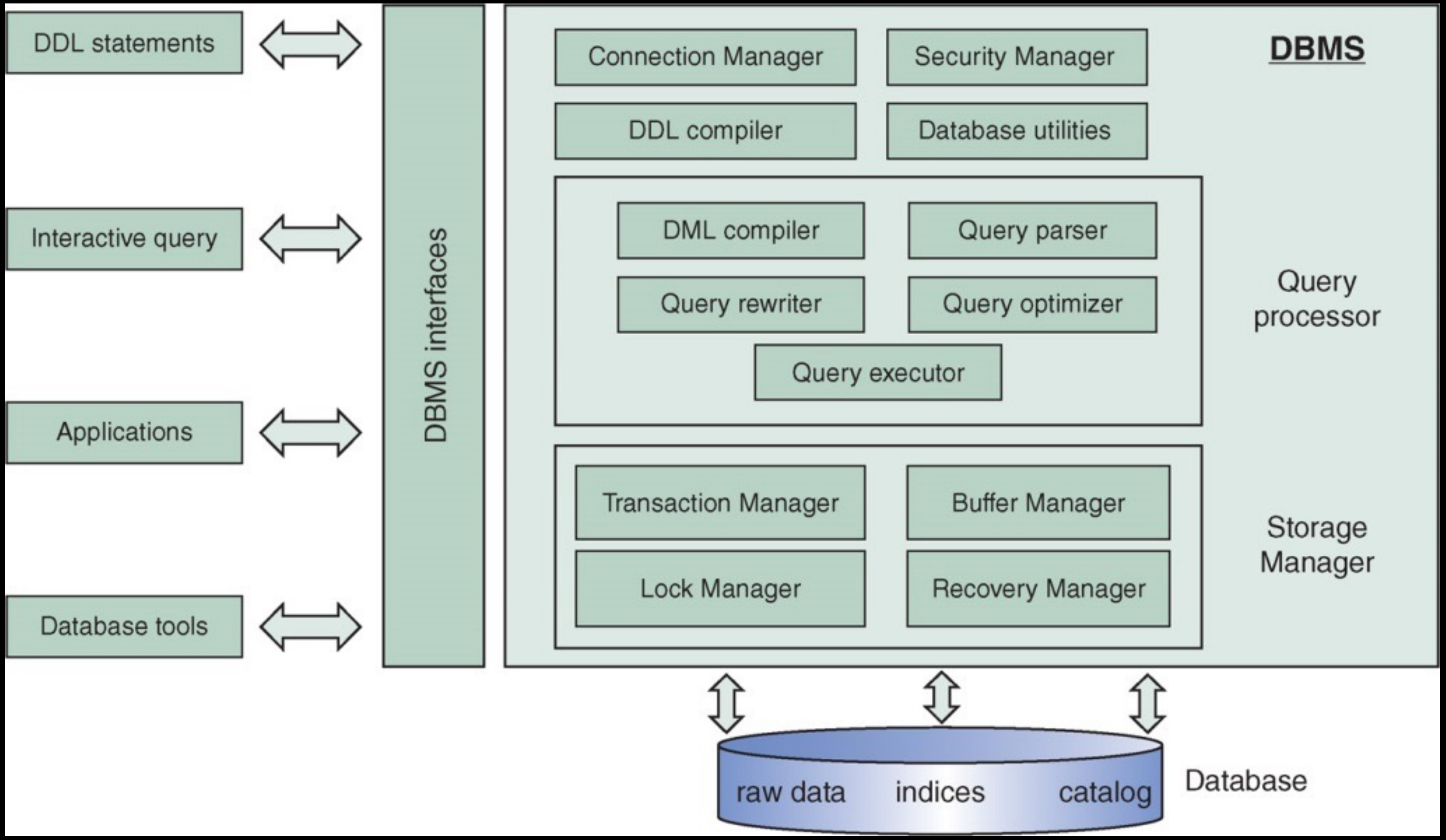
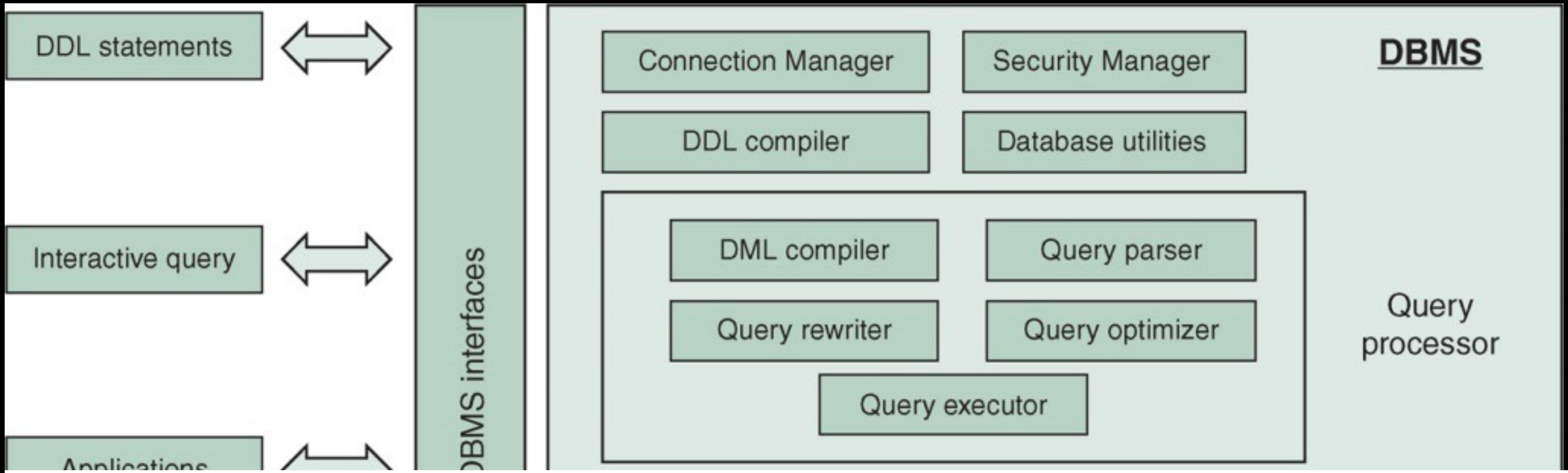


Figure 2.1

Query Processor

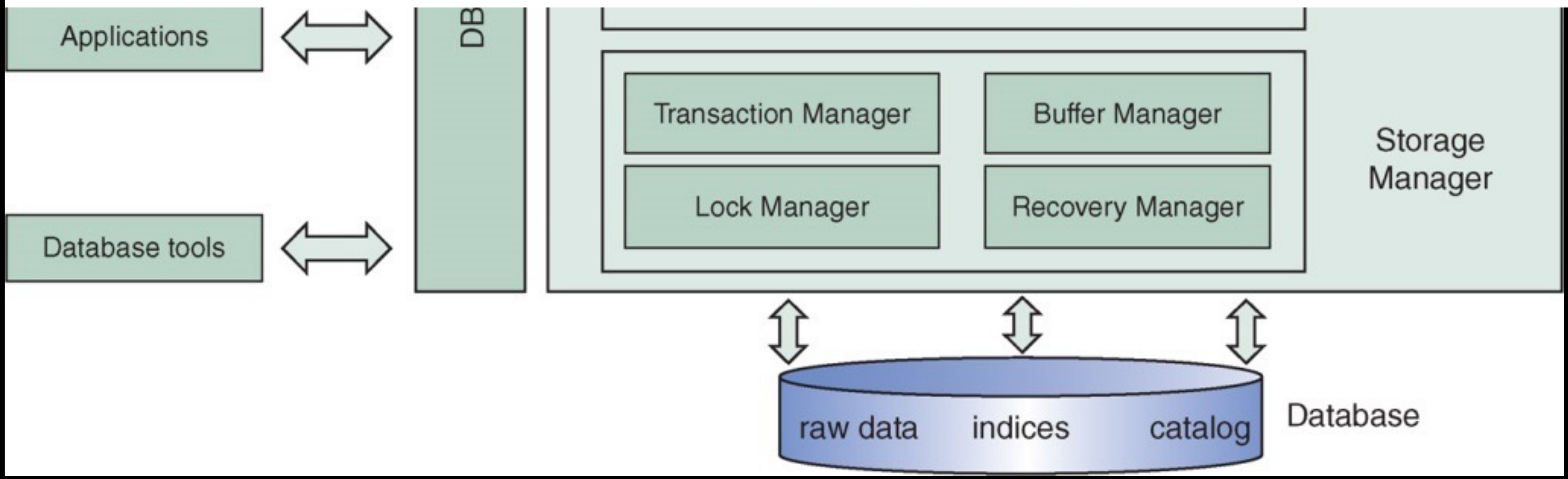


parsing an SQL query into a plan
optimizing that plan
executing the optimized plan using relational operators (e.g., join, groupby, etc.)

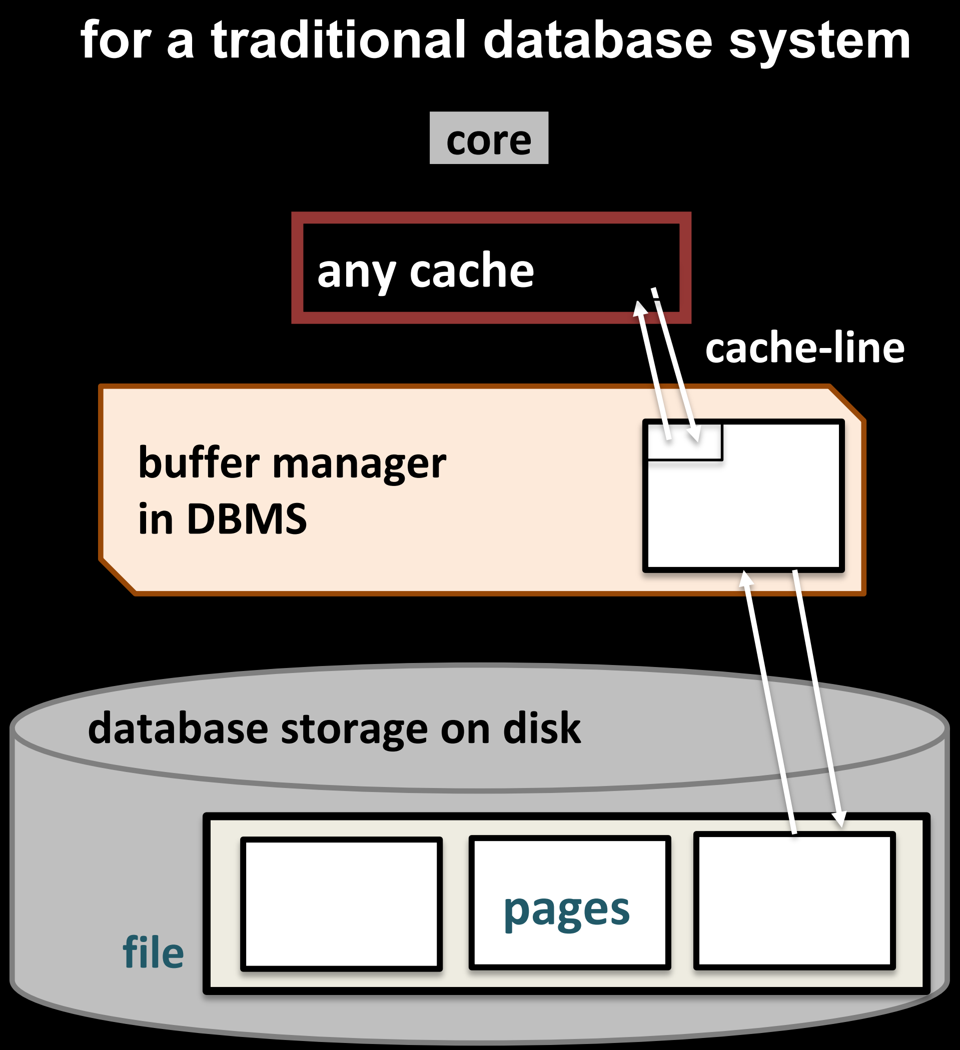
Storage Manager

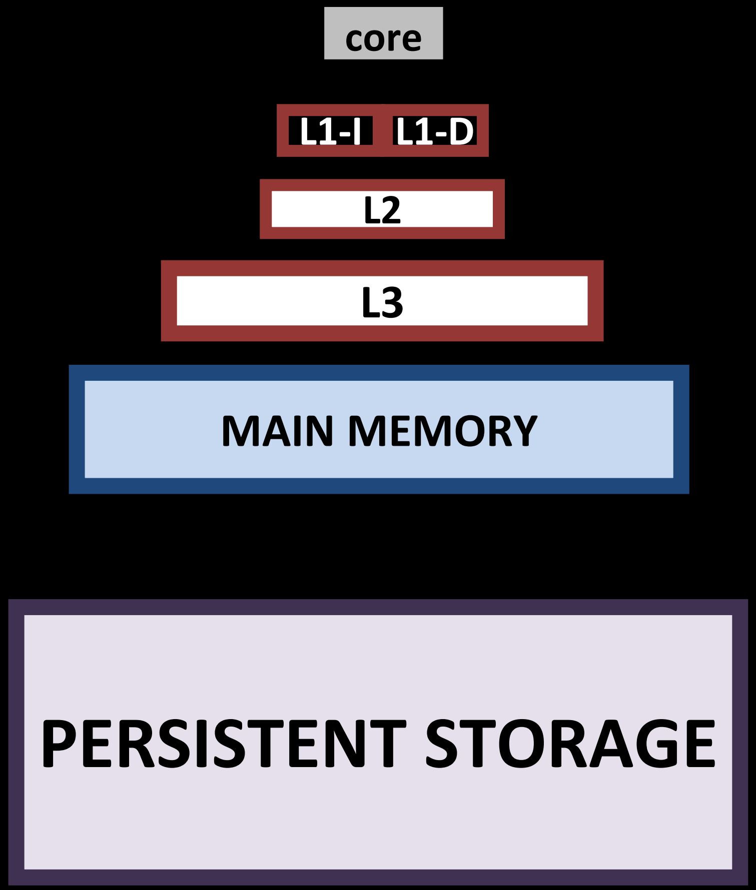
manages & keeps track of the data read from persistent storage into main memory:

- better optimizations because of application knowledge, e.g., replacement policy, prefetching
- you may want to force-flush some data to disk (e.g., log)
- you may not want to ever flush some data (e.g., aborted requests)



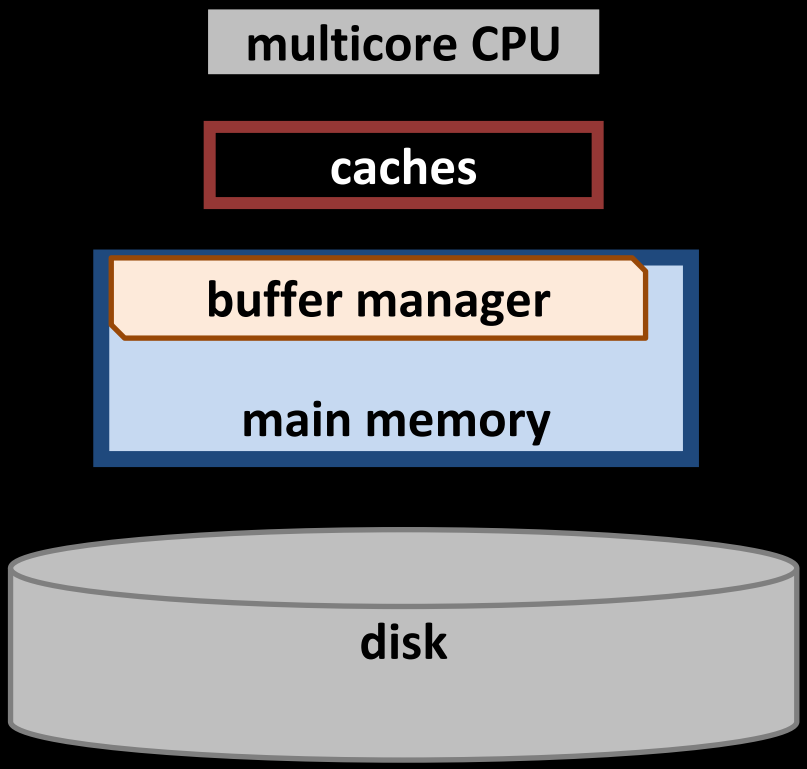
Data Movement through Storage Hierarchy





What if data fits in-memory?

Traditional System



Main-Memory System

- query compilation that generates more efficient code

- no buffer manager
- data organized for better cache utilization/accesses

- no disk use during query execution
- lightweight logging for recovery

Main-Memory DBMSs

- **What if all data is in RAM?**
  - No disk while executing tasks → never need to wait for disk
  - *Note:* There is still disk for persistent storage!

- **Don’t have to optimize data accesses for disk**

→ no buffer manager
  - → no need for tuples in slotted-pages in-memory, have direct access to tuples
  - → indexes aren’t kept in pages either & usually aren’t persisted

→ aim better cache utilization/accesses
  - → row-store for OLTP, column-store for OLAP
  - → cache-conscious index structures
  - → query compilation that generates cache-conscious code

also valid for disk

Main-Memory DBMSs: Downsides

- **Startups/restarts are expensive**
  - need to load everything in-memory
  - indexes has to be rebuilt
  - recovery takes longer if not done carefully

- **There comes a day where your data doesn’t fit in memory anymore**
  - most main-memory systems added support for efficiently accessing cold/old data from disk later
    → because customers ask for it

Take Away

- It is important to choose the right storage model for the target workload:
  - OLTP = Row Store
  - OLAP = Column Store
- Goal: maximize sequential access, minimize unnecessary data reads.

- Main-memory DBMS
  - No blocking for disk I/O
  - No buffer management/buffer pool
  - Optimizing for cache accesses
  - Non-blocking concurrency, lightweight logging