CSE544
Data Management

Lectures 4-6
Storage Manager

Announcements

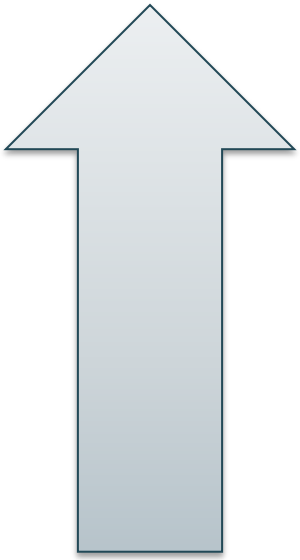
- Project teams tonight
- HW1 extended until Wednesday

Database Management System

- Stores data persistently
- Inserts/deletes/updates tuples
- Creates/updates indexes
- Executes Queries
- Transactions (won't discuss in 544)

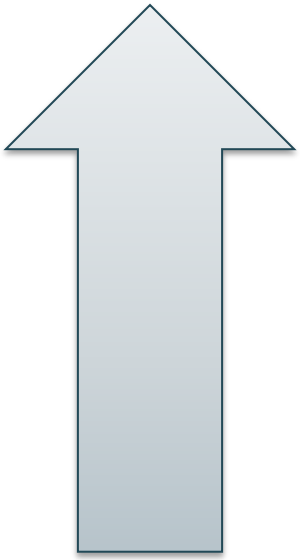
Database Management System

- Query optimizer
- Operator execution
- Access method
- Buffer pool manager
- Disk manager



Database Management System

- Query optimizer
- Operator execution
- Access method
- Buffer pool manager
- Disk manager

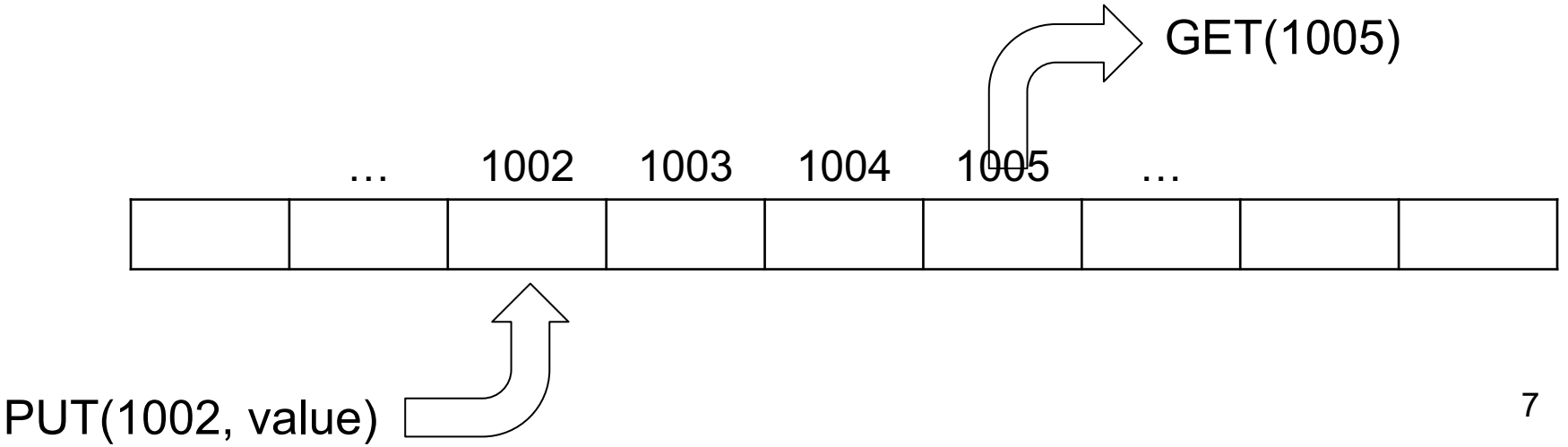


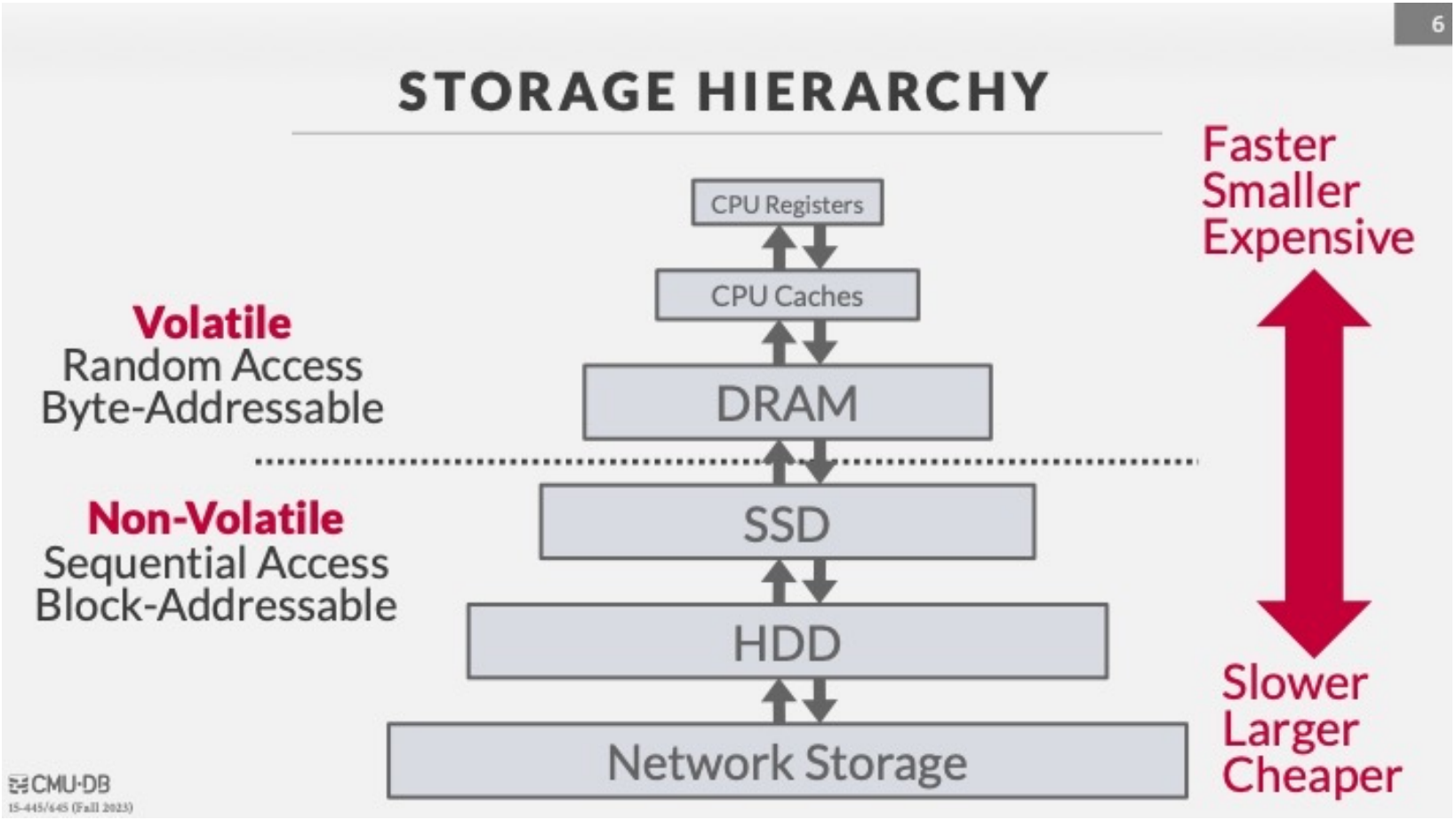
Today: storage manager and (briefly) buffer manager

Storage Manager

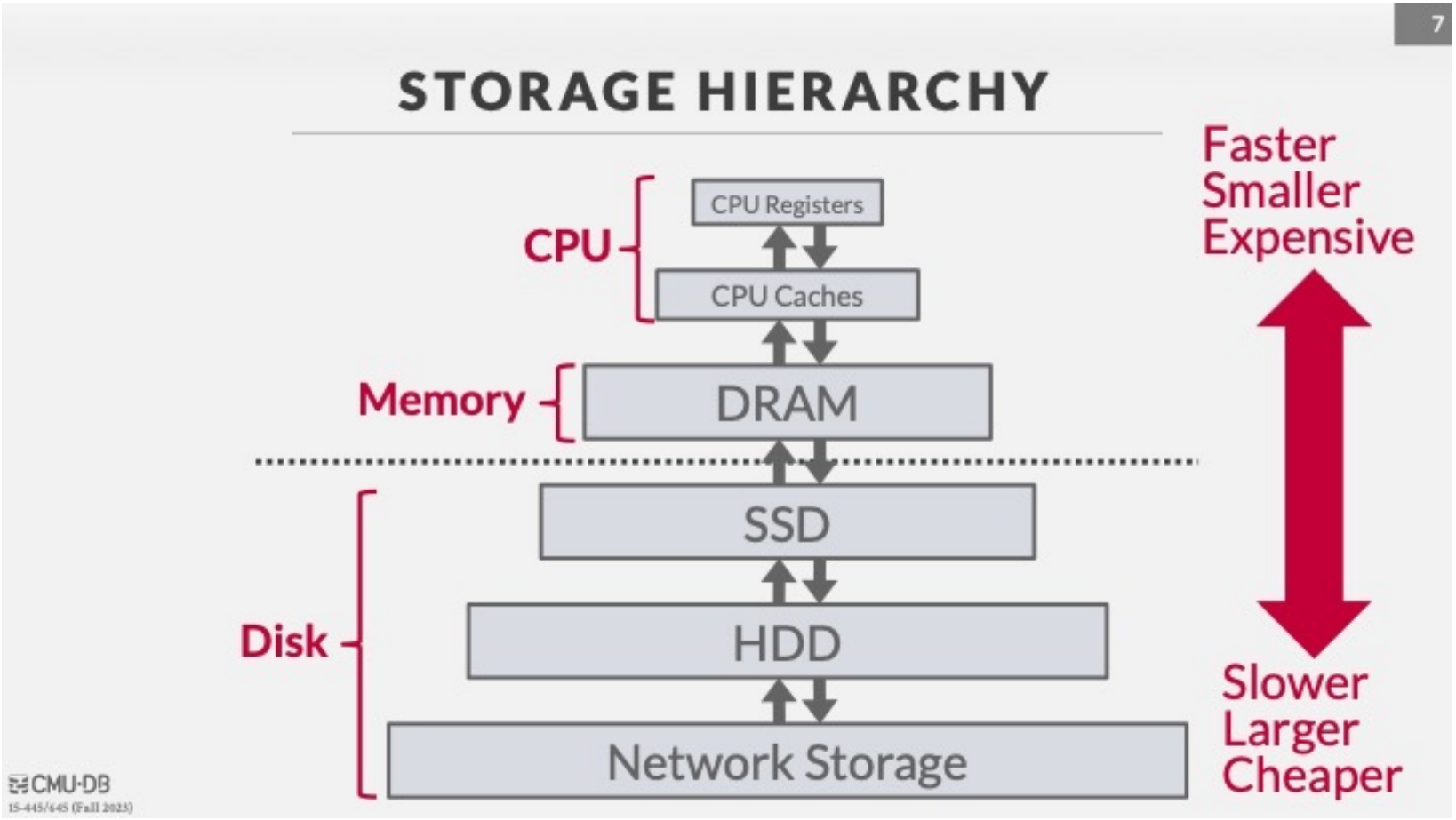
Basics of Storage Methods

- Multiple storage layers: disk, main memory, cache, registers
- Each layer: an array of locations
- Location: word or page/block

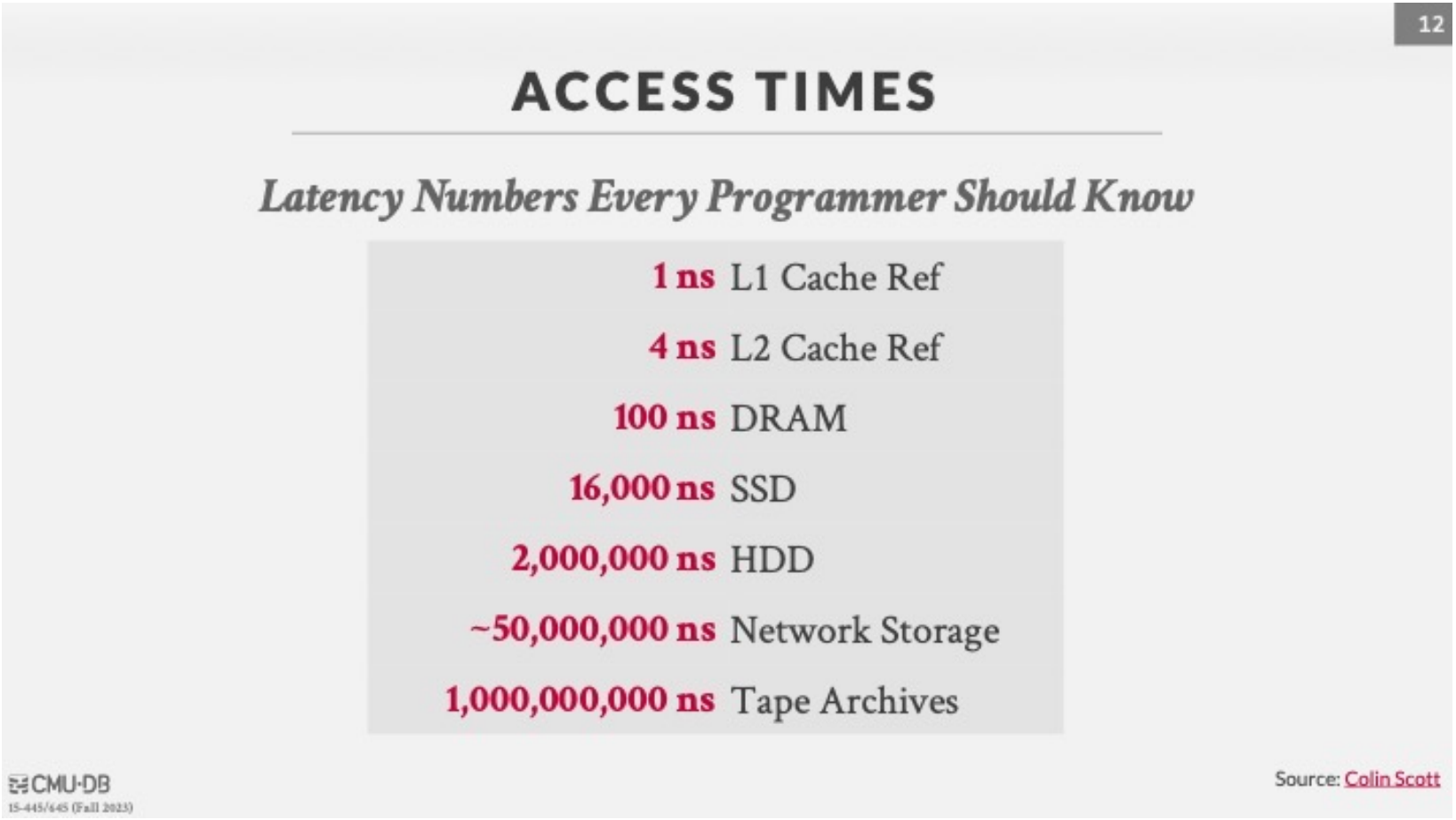




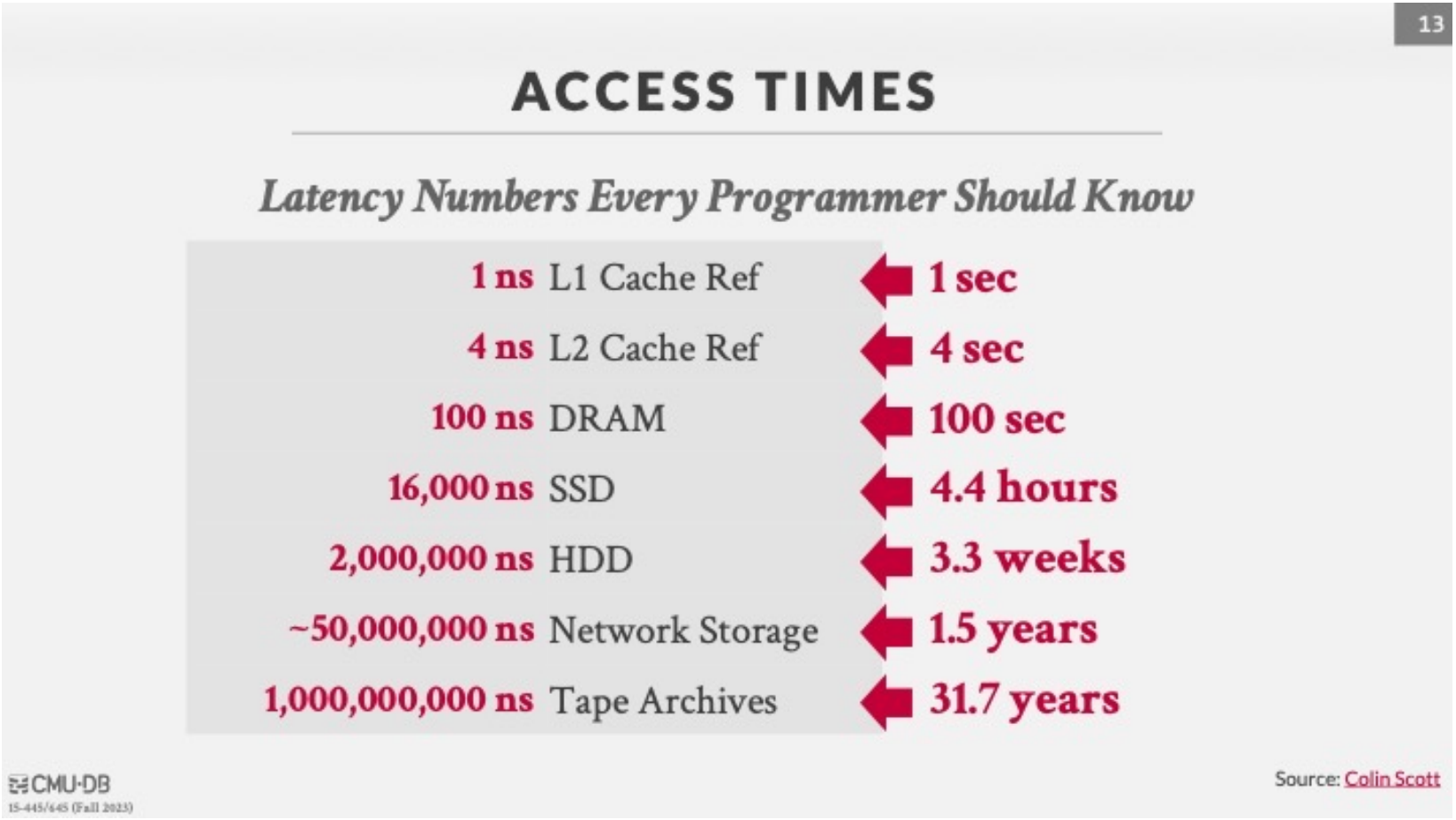
Credit: https://15445.courses.cs.cmu.edu/fall2023/



Credit: https://15445.courses.cs.cmu.edu/fall2023/



Credit: https://15445.courses.cs.cmu.edu/fall2023/



Credit: https://15445.courses.cs.cmu.edu/fall2023/

Disk

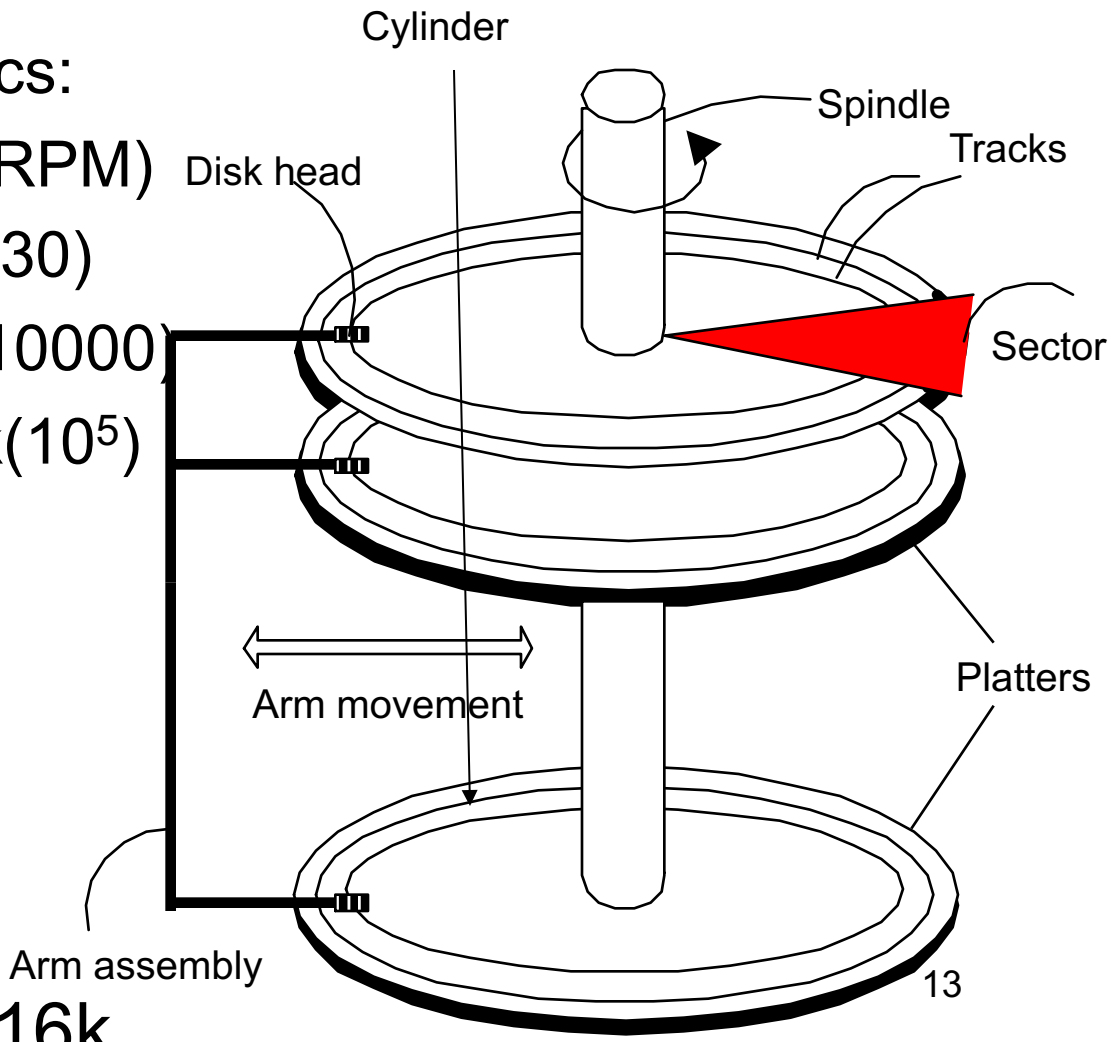
- Stores data persistenly
- Different technologies:
  - Tapes: long-term archives
  - HDD: most today's data is stored here
  - SDD: your laptop, or cache for HDD
- Unit of Read/Write operation:

  1 block = 0.5k .. 32k

Hard Drive Disk (HDD)

Mechanical characteristics:

- Rotation speed (5400RPM)
- Number of platters (1-30)
- Number of tracks (<=10000)
- Number of bytes/track(10^5)



Unit of read or write:

disk block

Once in memory:

page

Typically: 4k or 8k or 16k

Disk Access Characteristics

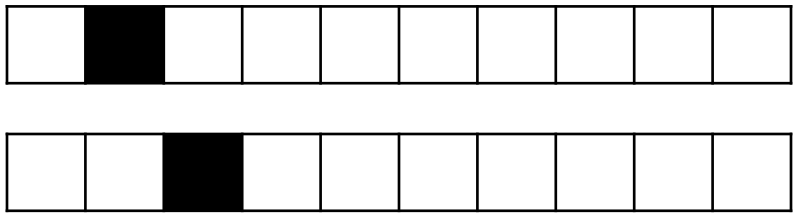
- Disk latency
  - Time between request and when data is in memory
    = seek time + rotational latency
- Seek time = time for the head to reach cylinder
  - 10ms – 40ms
- Rotational latency = time for sector to rotate
  - Rotation time = 10ms
  - Average latency = 10ms / 2
- Transfer time = typically 40-80MB/s

Sequential/Random Access

Sequential

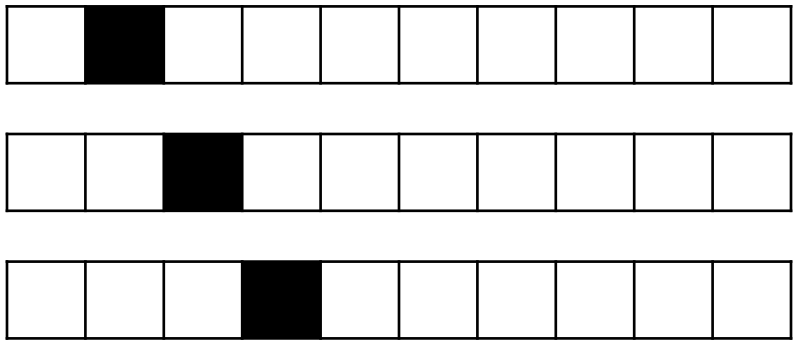
Sequential/Random Access

Sequential



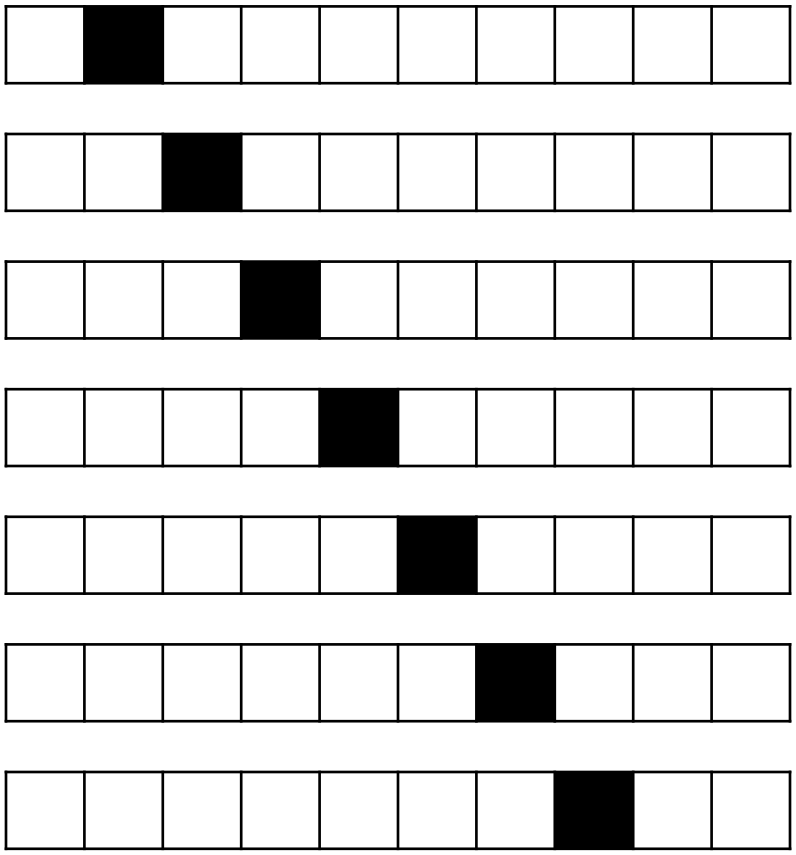
Sequential/Random Access

Sequential



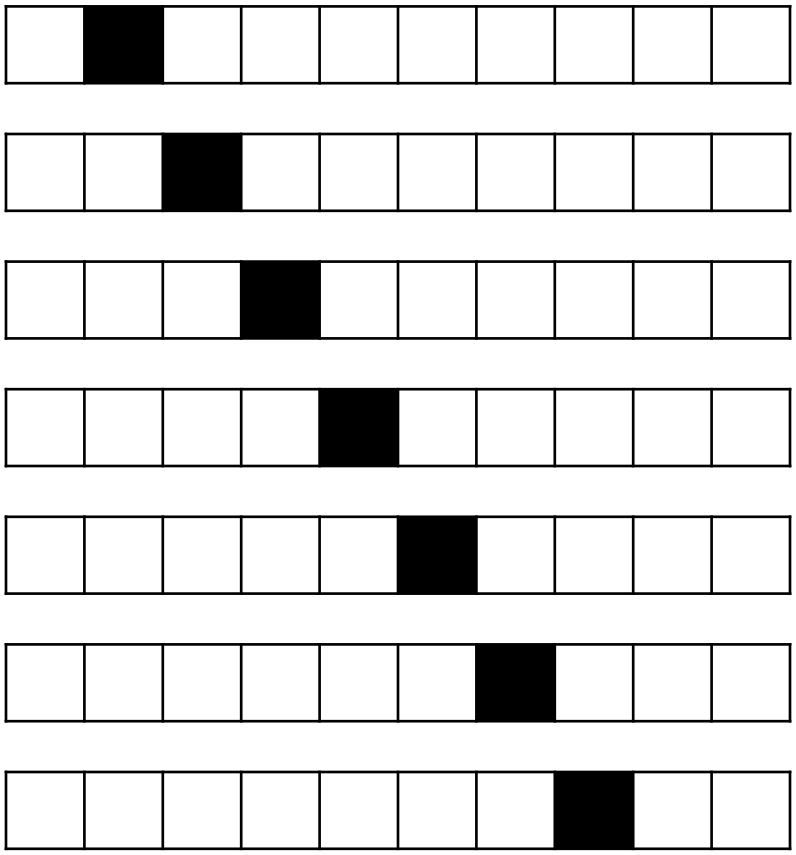
Sequential/Random Access

Sequential

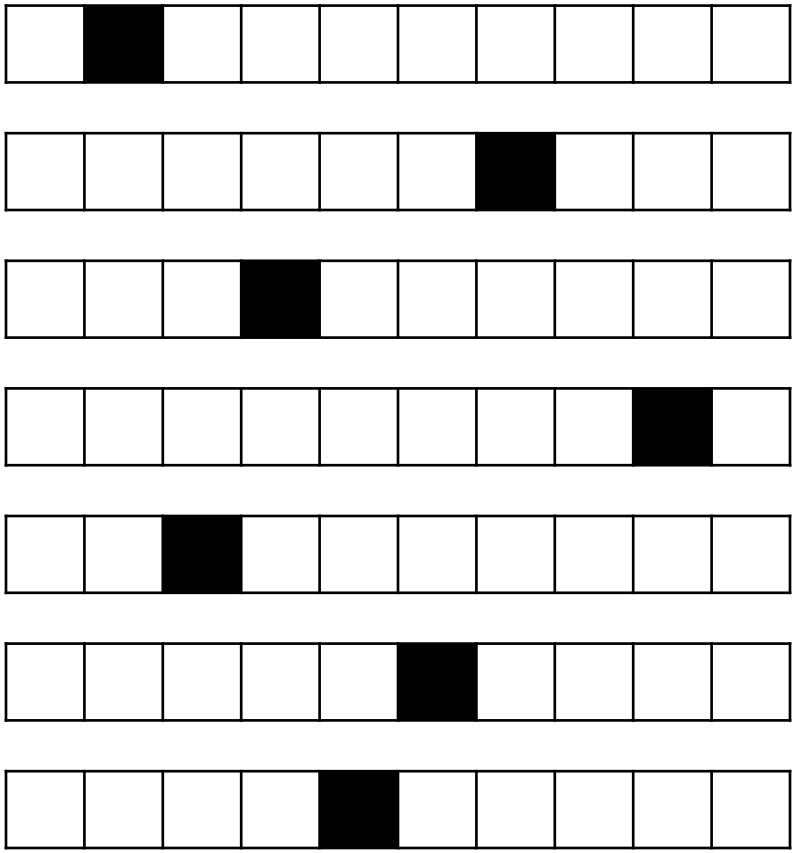


Sequential/Random Access

Sequential



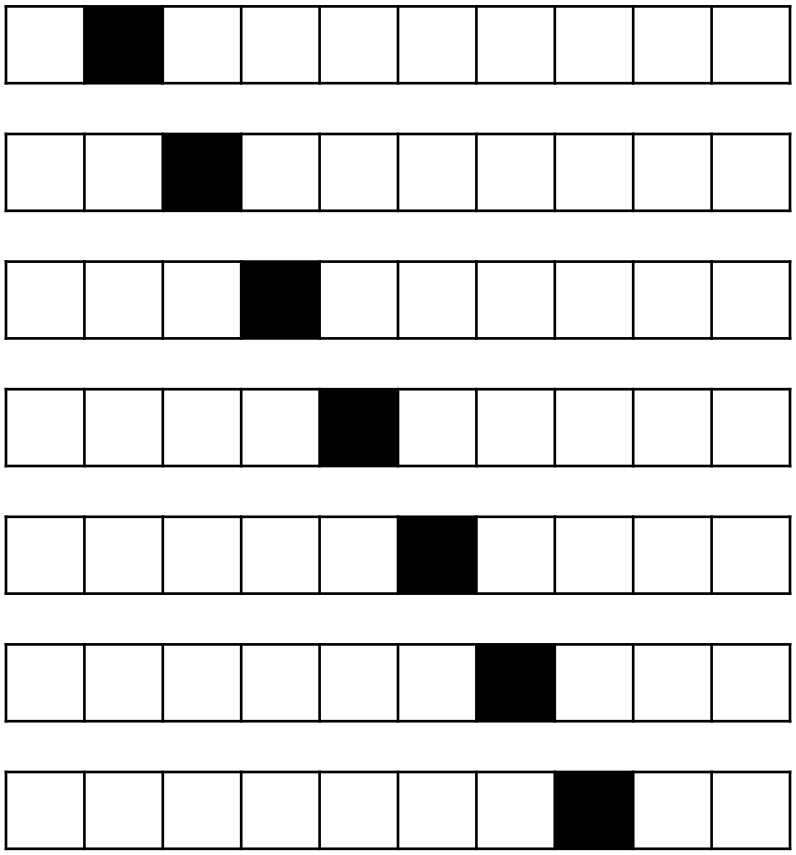
Random



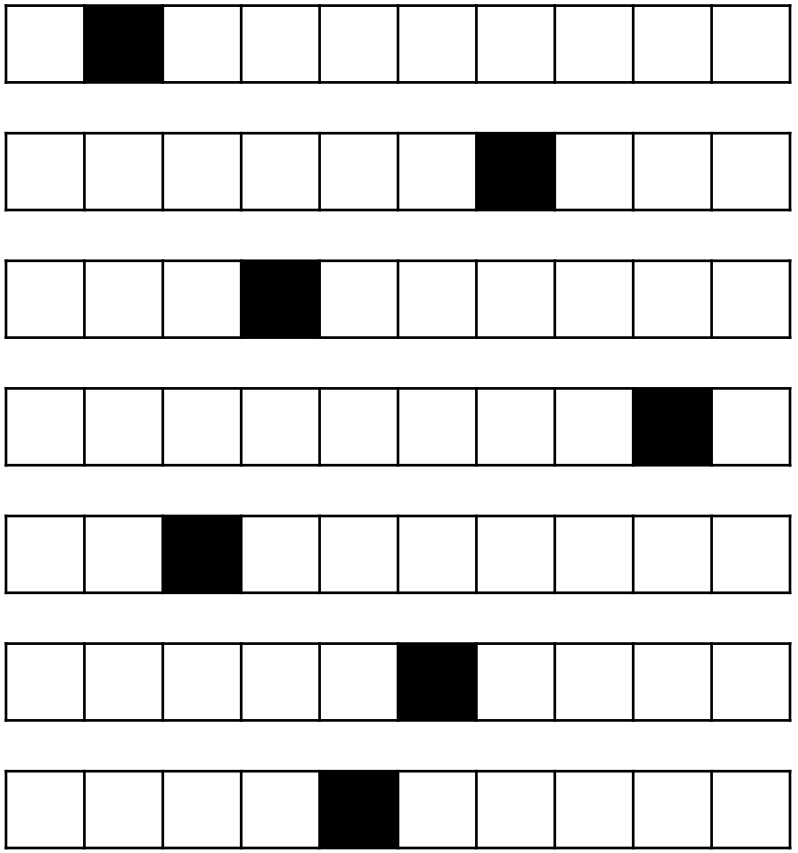
Sequential/Random Access

Sequential

Faster



Random



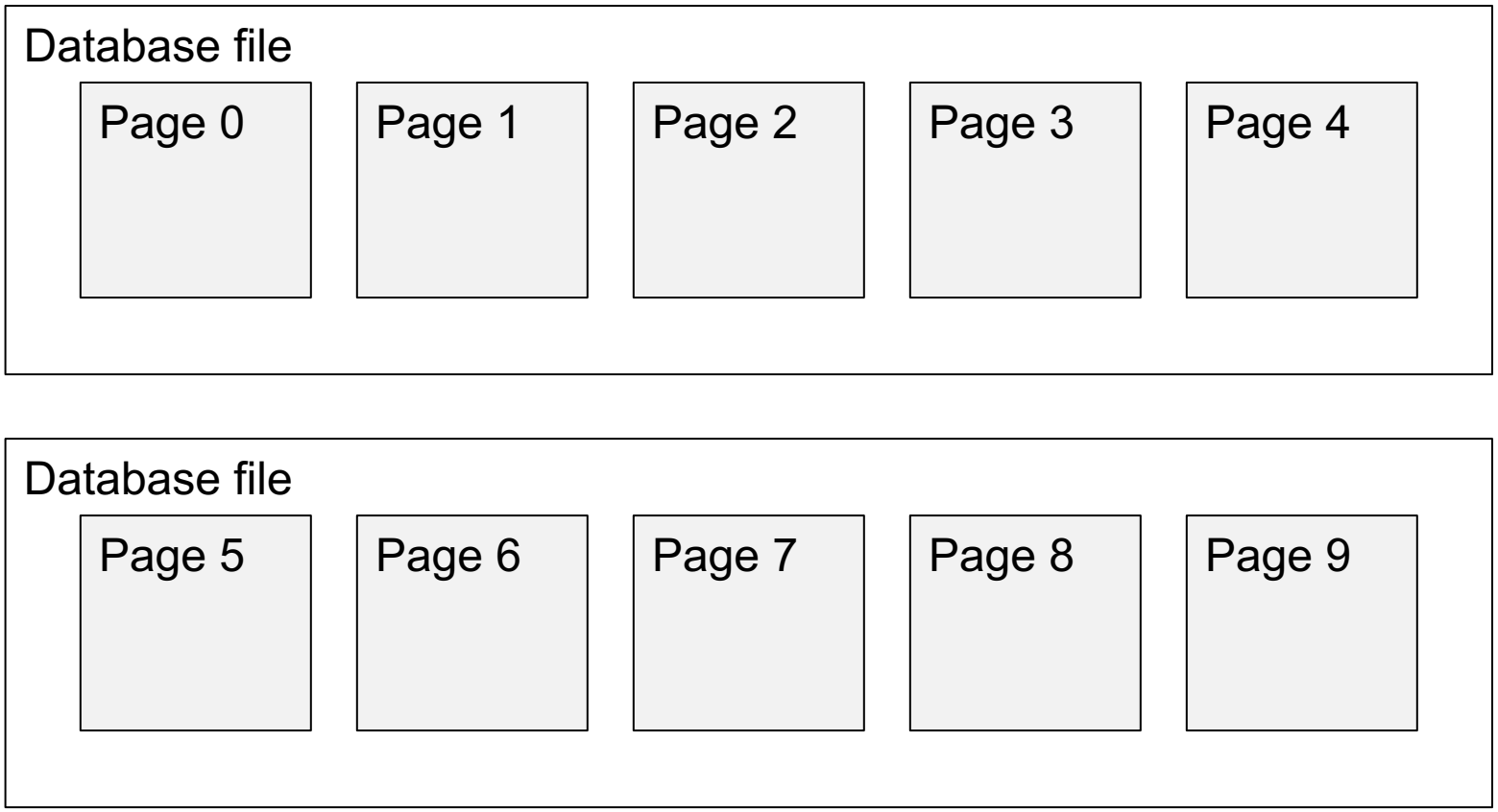
Storage Manager

Older DBMS manage the raw disk

Modern DBMS use OS to create files

- 1 file = multiple (continuous?) blocks
- 1 block = multiple records, free space
- Storage manager keeps track of the files, their content, free space

Storing Pages on Disk



Catalog of metadata: files, pages, records, free space

Page Format

- Row-oriented Format
  - A.k.a. N-ary storage model NSM
  - Each page contains several records
  - Records are laid out in the attribute order

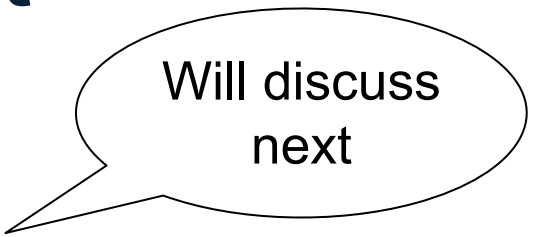
Page Format

- Row-oriented Format
  - A.k.a. N-ary storage model NSM
  - Each page contains several records
  - Records are laid out in the attribute order
- PAX
  - Each page contains several records, but grouped by their attribute

Page Format

- Row-oriented Format
  - A.k.a. N-ary storage model NSM
  - Each page contains several records
  - Records are laid out in the attribute order
- PAX
  - Each page contains several records, but grouped by their attribute
- Column-oriented Format
  - A.k.a. C-store
  - Each page contains values from only one attribute

Page Format



- Row-oriented Format
  - A.k.a. N-ary storage model NSM
  - Each page contains several records
  - Records are laid out in the attribute order
- PAX
  - Each page contains several records, but grouped by their attribute
- Column-oriented Format
  - A.k.a. C-store
  - Each page contains values from only one attribute

Row-Oriented Storage

Product

Name	Price	Color
iPhone	599	gray
Jacket	129	blue
Pants	89	black
…		
…		
Bicycle	599	Red
…		

Row-Oriented Storage

Physical layout

Logica schema

Page 0

iPhone	599	gray
Jacket	129	blue
Pants	89	black
…		

Page 1

Bicycle	599	Red
…		
…		

…

Product

Name	Price	Color
iPhone	599	gray
Jacket	129	blue
Pants	89	black
…		
…		
Bicycle	599	Red
…		

Row-Oriented Storage

Physical layout

Logica schema

Page 0

iPhone	599	gray
Jacket	129	blue
Pants	89	black
…		

Page 1

Bicycle	599	Red
…		
…		

…

Product

Name	Price	Color
iPhone	599	gray
Jacket	129	blue
Pants	89	black
…		
…		
Bicycle	599	Red
…		

The schema stored separately, in the *database catalog*

Row-Oriented Storage

Sequential file: unordered collection of records

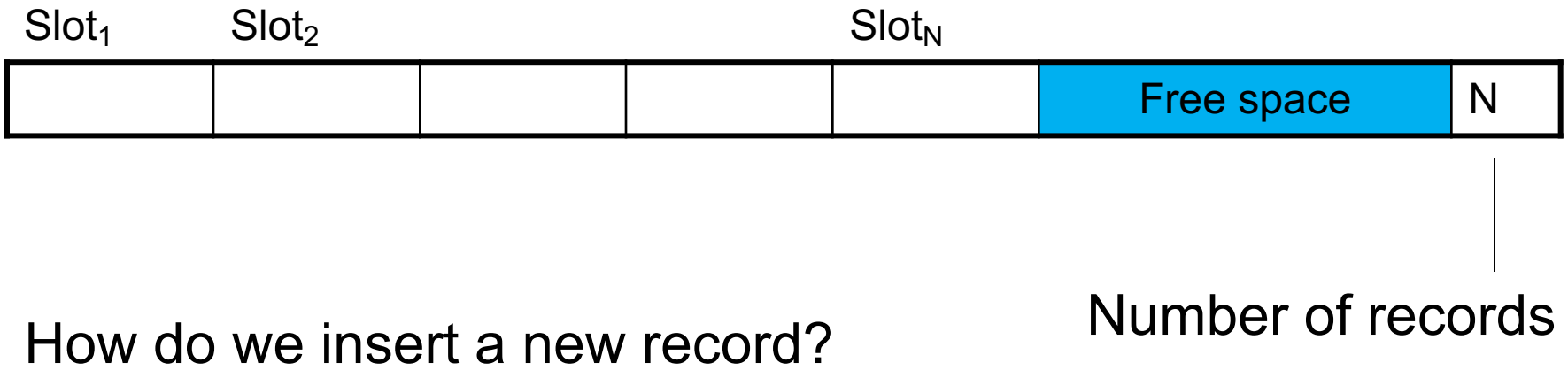
- Advantage:
  - Can insert a new record at the end of the file, or in any page that has free space
- Disadvantage
  - Sequential search for a record (→indexes)
  - Overwrite entire block on update (→LSM)

Page Format

- One page contains several records
- How exactly do we store these records inside the page?

Page Format Approach 1

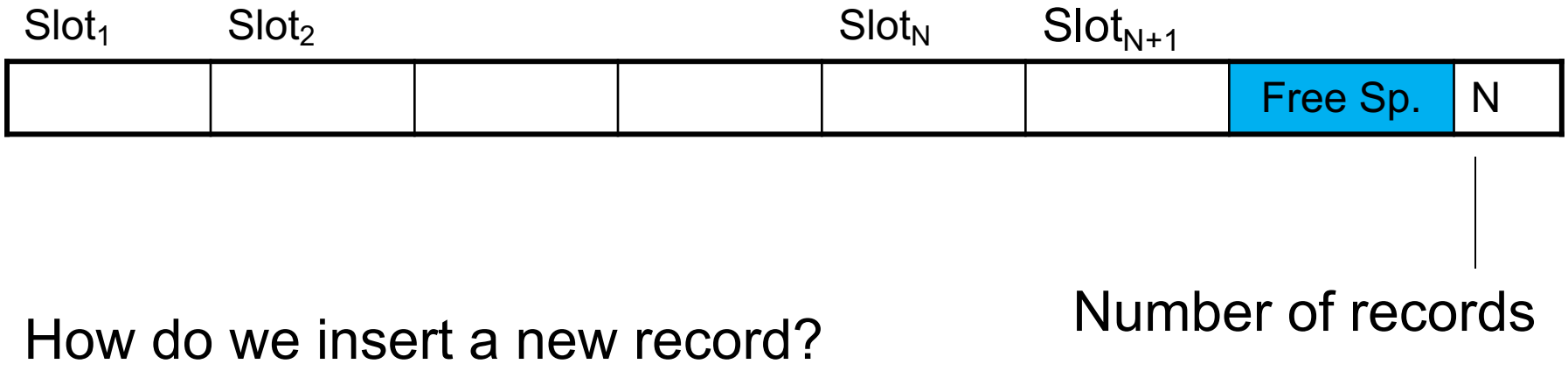
Fixed-length records: packed representation
Divide page into **slots**. Each slot can hold one tuple
Record ID (RID) for each tuple is (PageID,SlotNb)



How do we insert a new record?

Page Format Approach 1

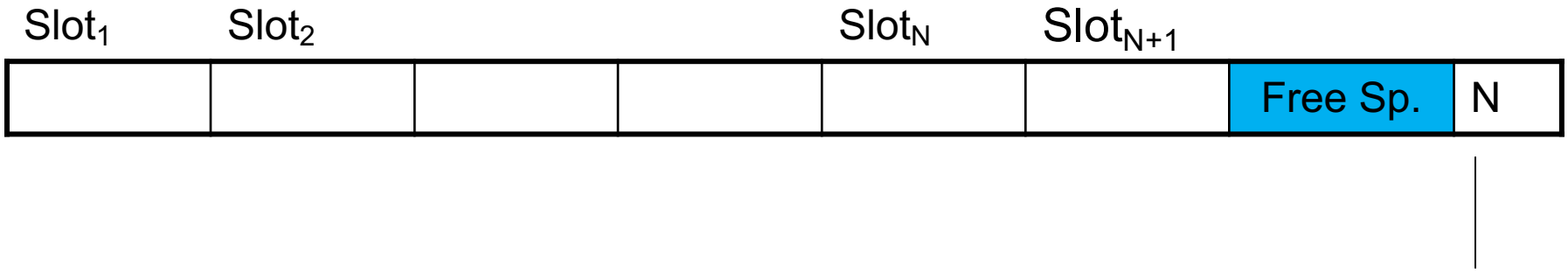
Fixed-length records: packed representation
Divide page into **slots**. Each slot can hold one tuple
Record ID (RID) for each tuple is (PageID,SlotNb)



How do we insert a new record?

Page Format Approach 1

Fixed-length records: packed representation
Divide page into **slots**. Each slot can hold one tuple
Record ID (RID) for each tuple is (PageID,SlotNb)

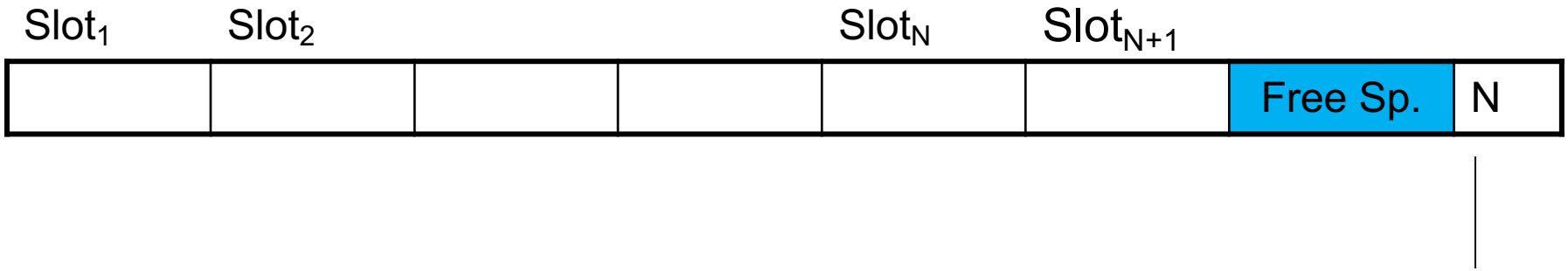


How do we insert a new record?

How do we delete a record?

Page Format Approach 1

Fixed-length records: packed representation
Divide page into **slots**. Each slot can hold one tuple
Record ID (RID) for each tuple is (PageID,SlotNb)



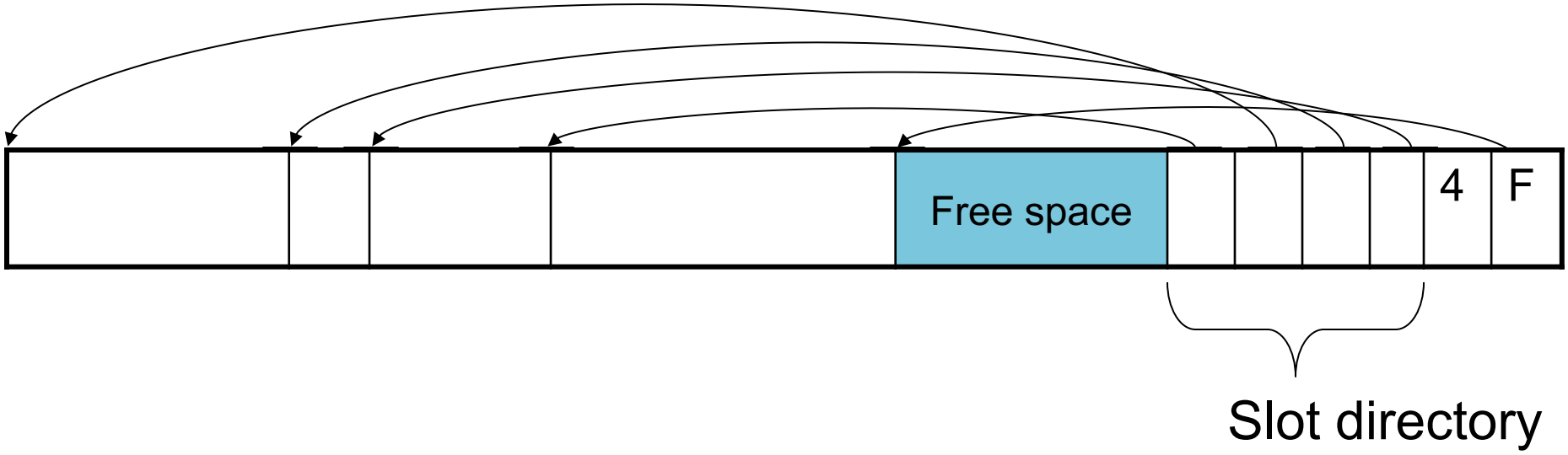
How do we insert a new record?

How do we delete a record? Cannot remove record (why?)

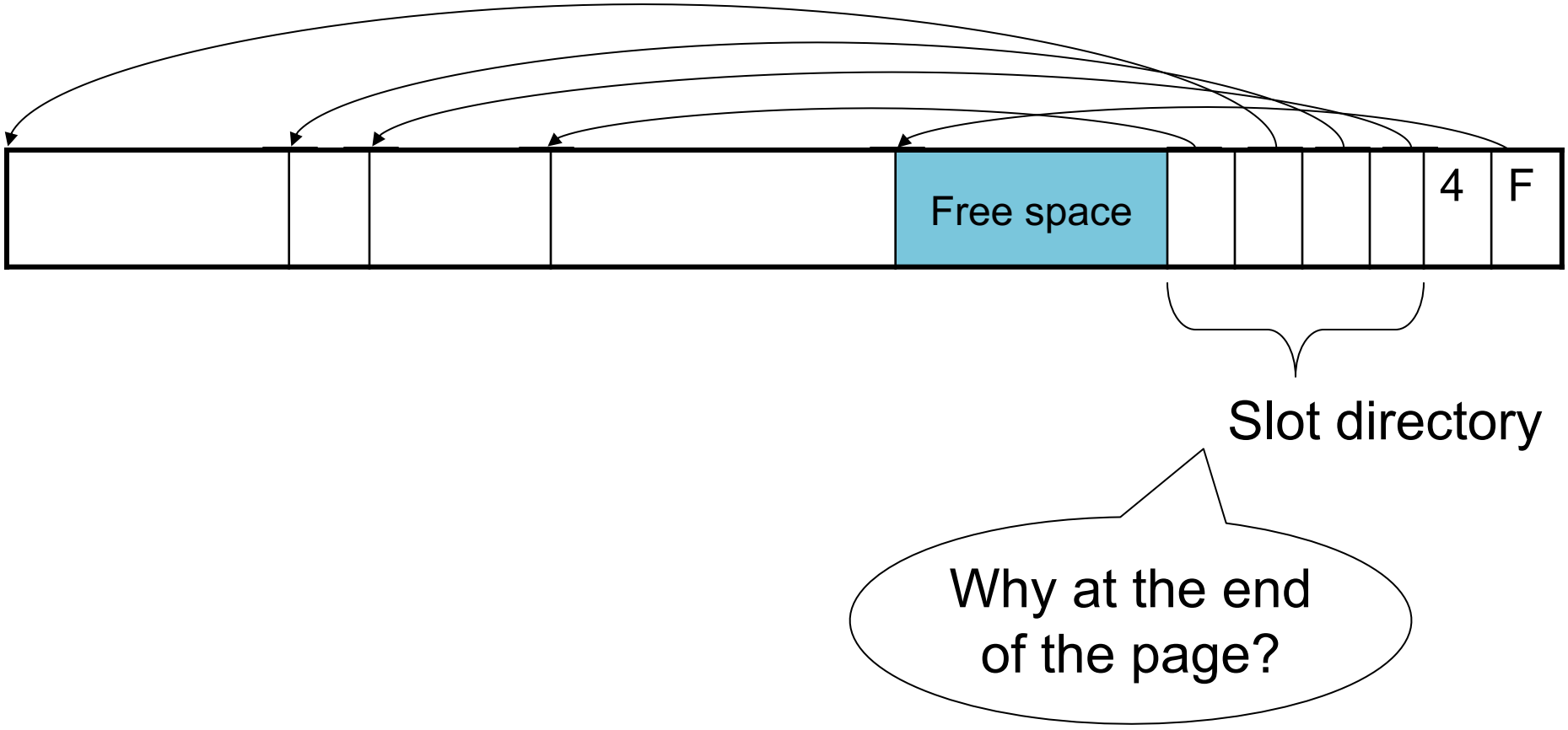
How do we handle variable-length records?

Page Format Approach 2

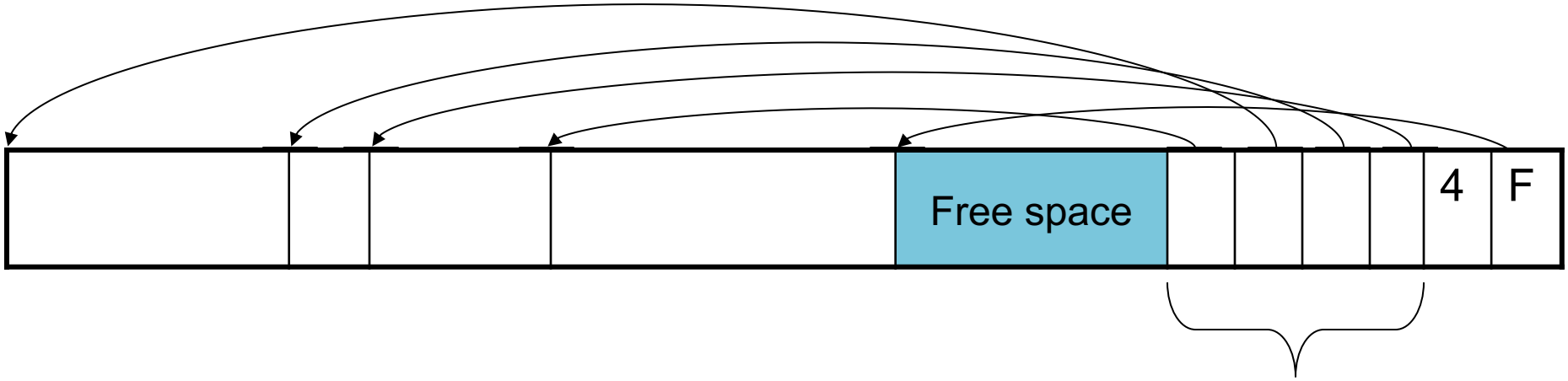
Page Format Approach 2



Page Format Approach 2



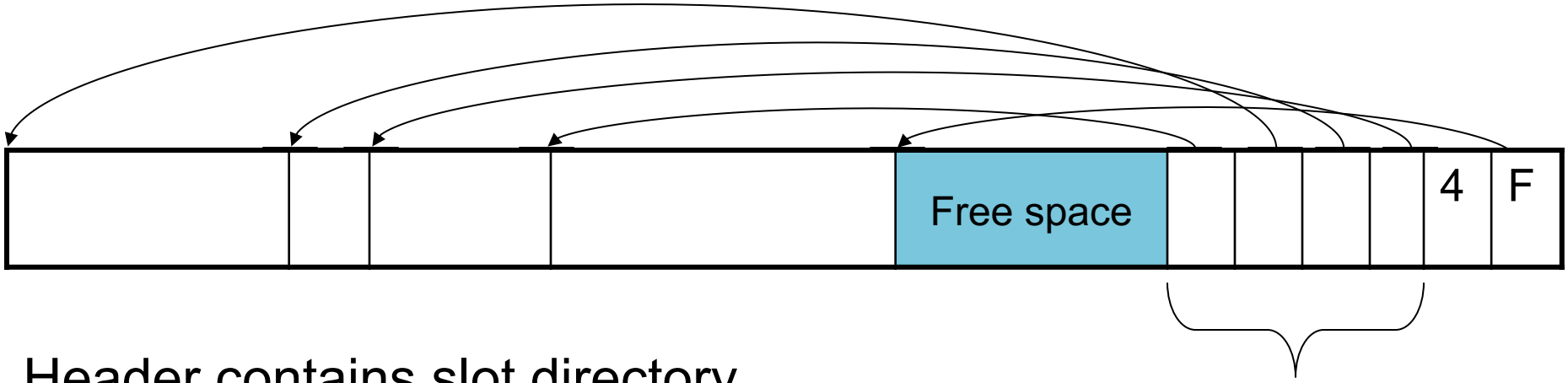
Page Format Approach 2



We can add a new slot when we insert a new record

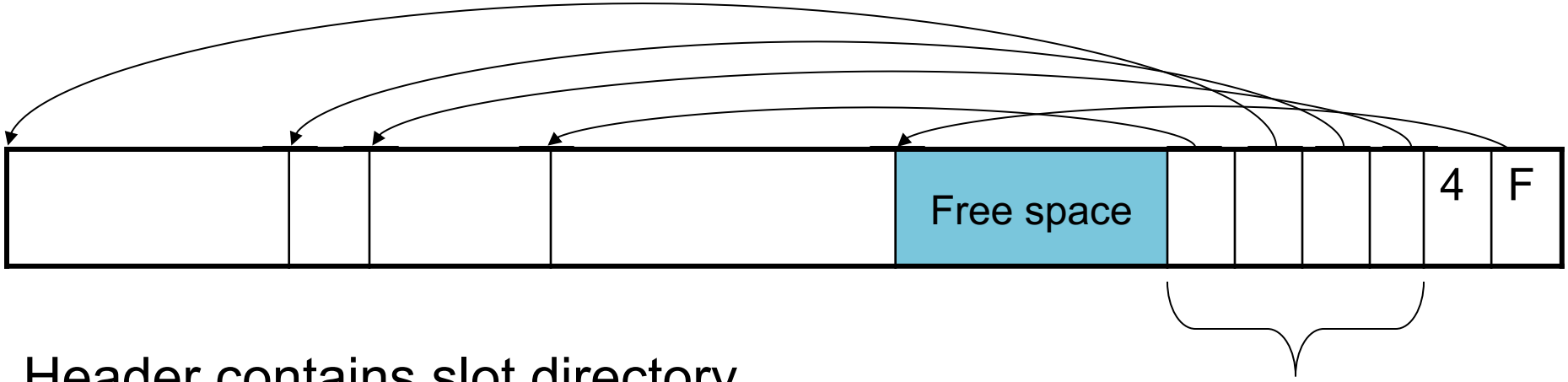
Why at the end of the page?

Page Format Approach 2



Header contains slot directory
+ Need to keep track of nb of slots
+ Also need to keep track of free space (F)

Page Format Approach 2



Header contains slot directory
+ Need to keep track of nb of slots
+ Also need to keep track of free space (F)

RID is (PageID, SlotID) combination
Variable-length records OK
Moving tuples inside page OK

Record Format

- One record contains several attributes
- How exactly do we store these attributes inside the record?

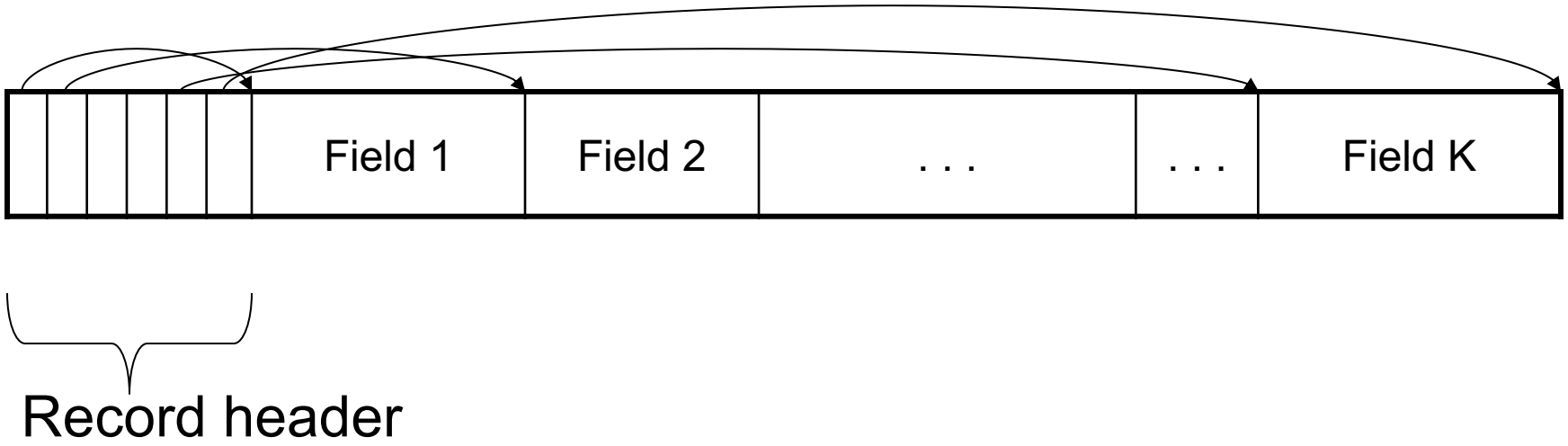
Record Formats

Fixed-length records => Each field has a fixed length (i.e., it has the same length in all the records)

Information about field lengths and types is in the catalog

Record Formats

Variable length records



Remark: NULLS require no space at all (why ?)

Row-Oriented: Summary

- Sequential file: records stored in arbitrary order
- One page contains a set of records
  - Records cannot exceed block boundary
- One record contains a sequence of attributes

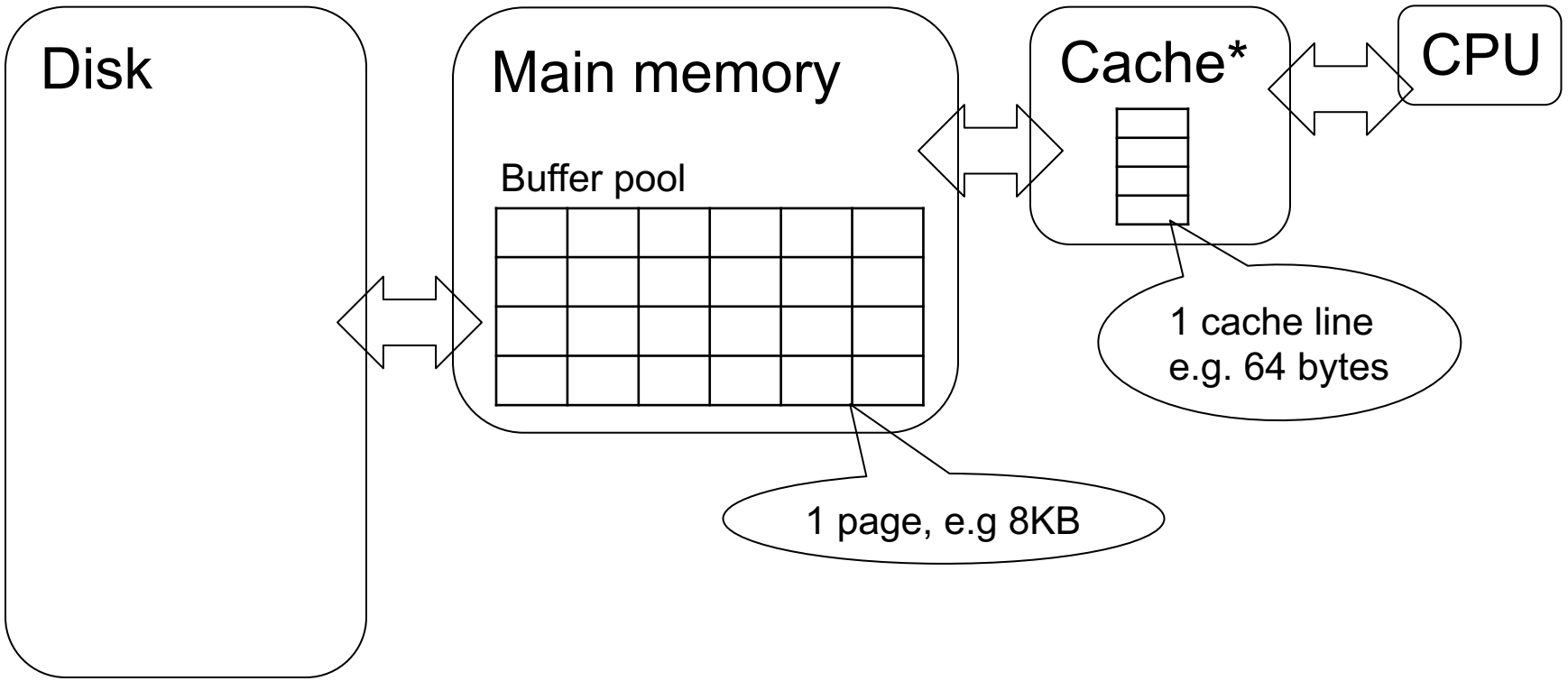
Next: PAX (improves L2-locality)

PAX

- As before, each page has a complete set of records
- However, the records are not stored sequentially, but their values are grouped by the attribute
- This improves the L2-cache locality, as we will see next
- I'm using (w/ permission) the slides from the original presentation at VLDB 2001

Review: the Cache

Memory hierarchies:



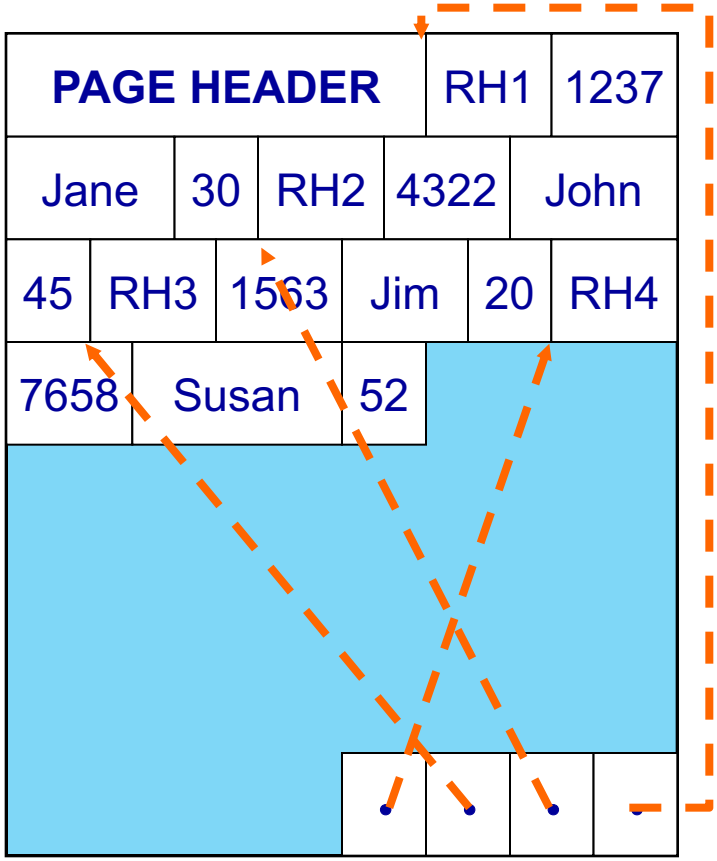
*aka CPU cache; several! L3, L2, L1 cache

Current Scheme: Slotted Pages

Formal name: NSM (N-ary Storage Model)

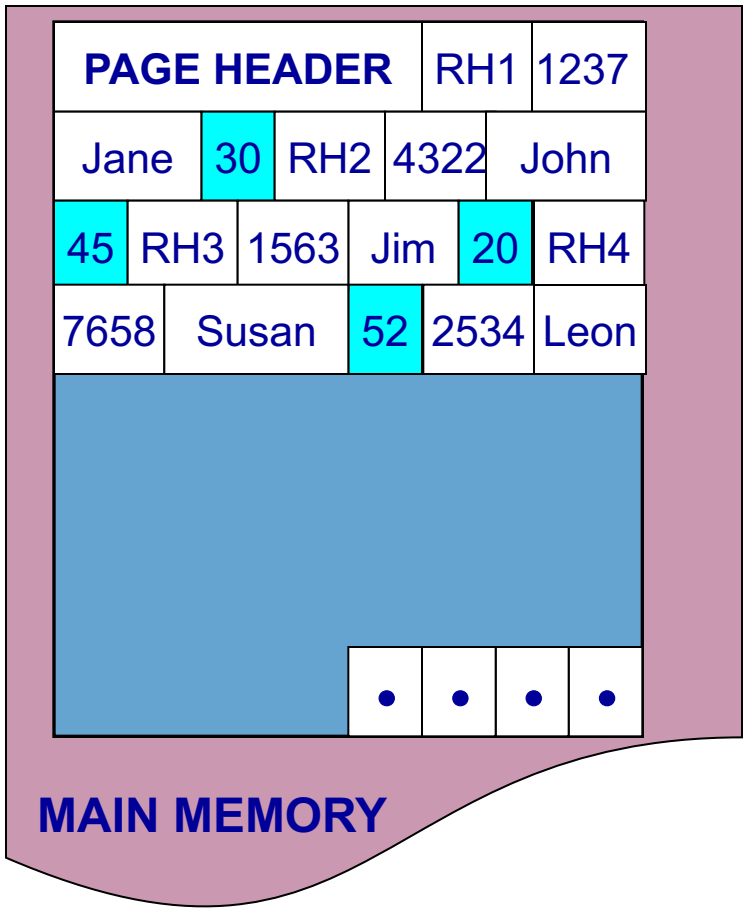
R

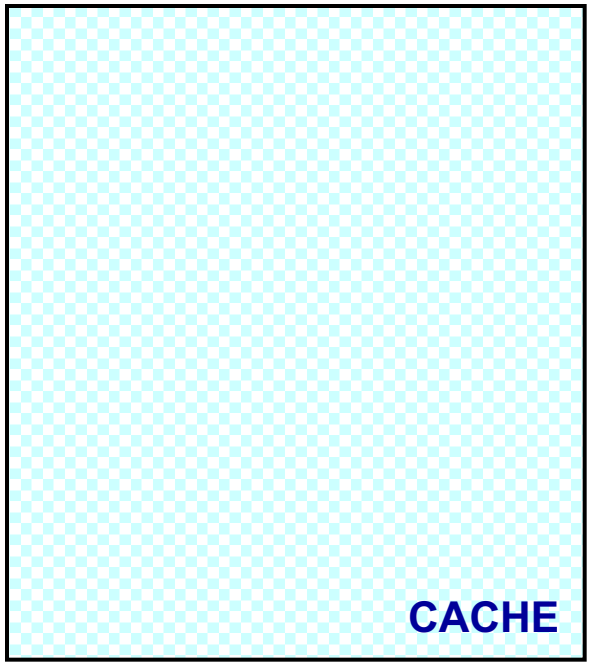
RID	SSN	Name	Age
1	1237	Jane	30
2	4322	John	45
3	1563	Jim	20
4	7658	Susan	52
5	2534	Leon	43
6	8791	Dan	37



- Records are stored sequentially
- Offsets to start of each record at end of page

Predicate Evaluation using NSM



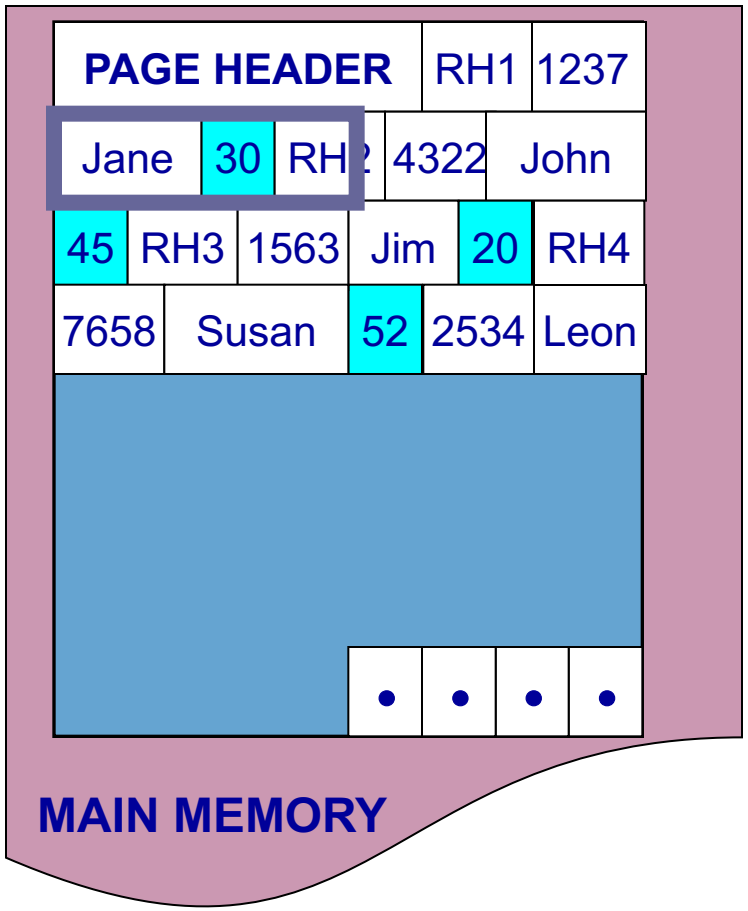


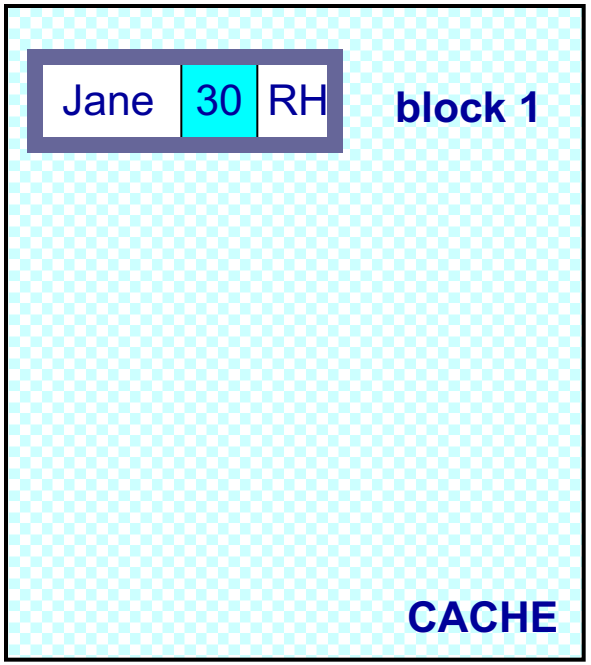

```
select name
from R
where age > 50
```

NSM pushes non-referenced data to the cache

Ailamaki VLDB'01 http://research.cs.wisc.edu/multifacet/papers/vldb01_pax_talk.ppt

Predicate Evaluation using NSM



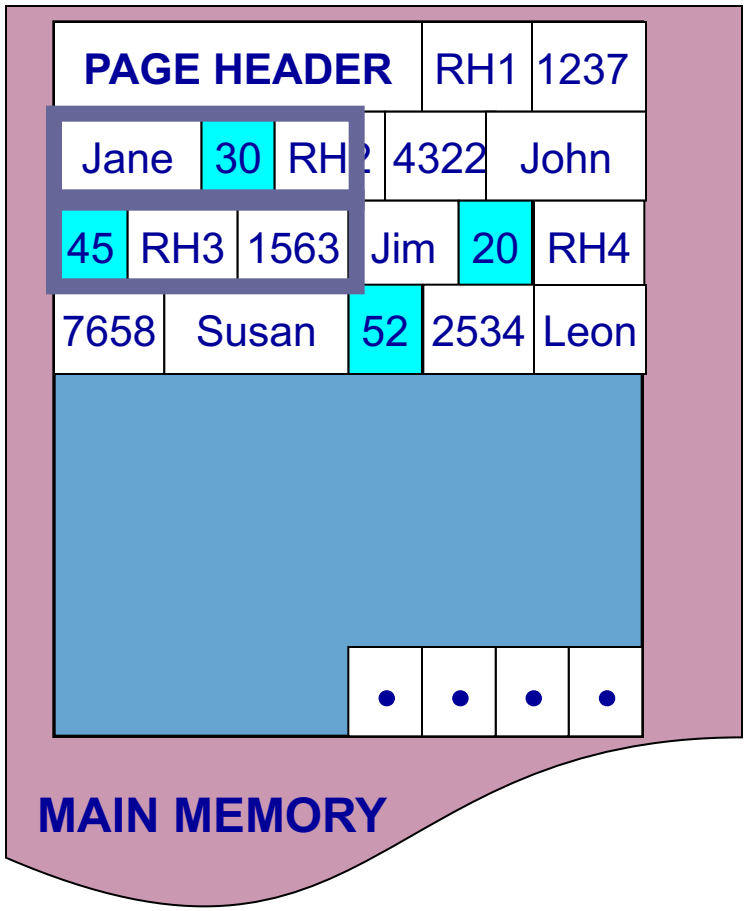


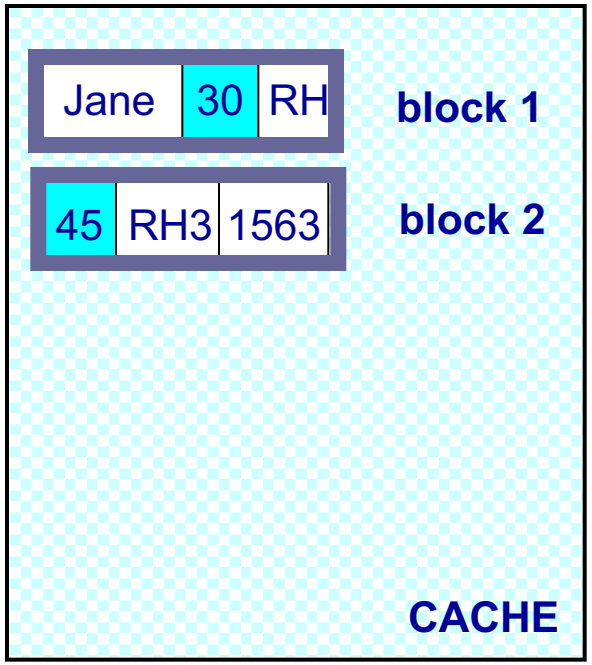

```
select name
from R
where age > 50
```

NSM pushes non-referenced data to the cache

Ailamaki VLDB'01 http://research.cs.wisc.edu/multifacet/papers/vldb01_pax_talk.ppt

Predicate Evaluation using NSM



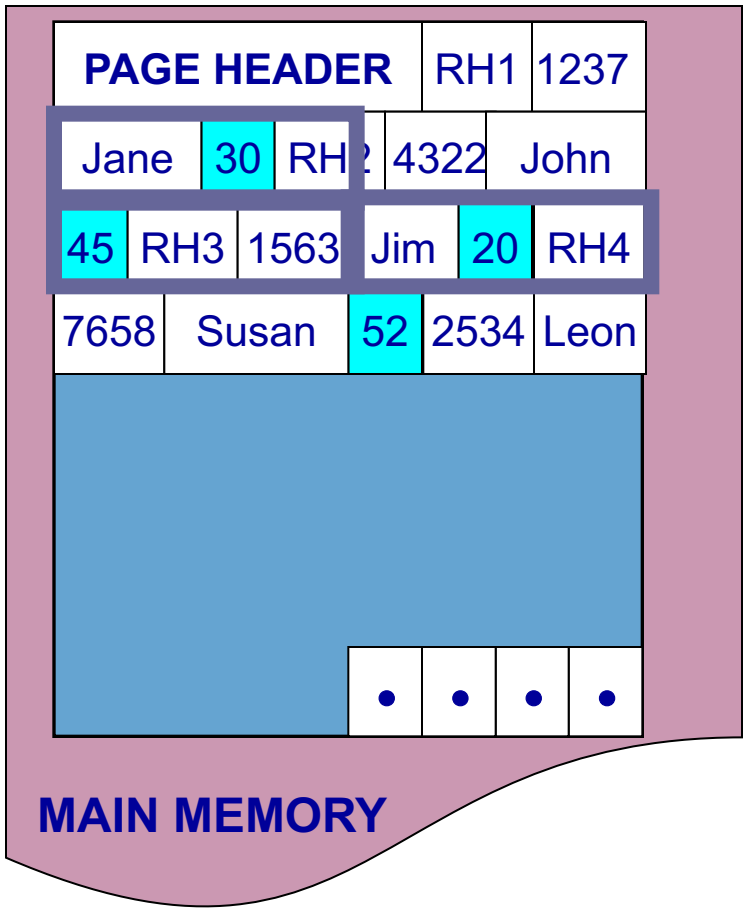


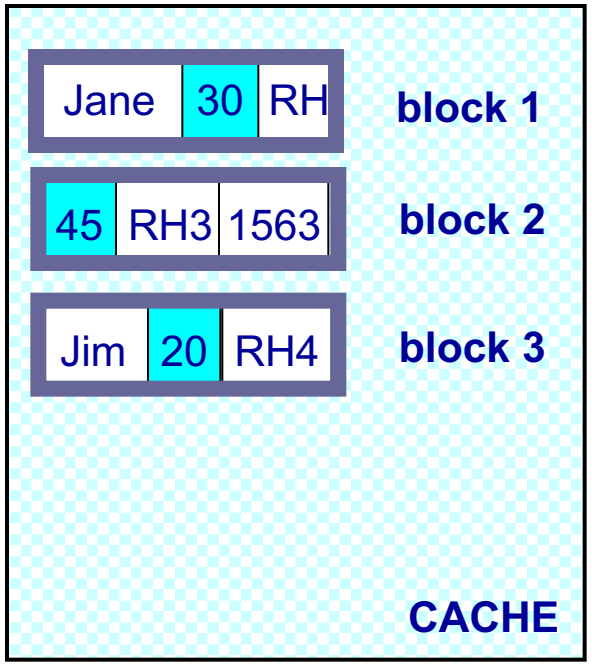

```
select name
from R
where age > 50
```

NSM pushes non-referenced data to the cache

Ailamaki VLDB'01 http://research.cs.wisc.edu/multifacet/papers/vldb01_pax_talk.ppt

Predicate Evaluation using NSM



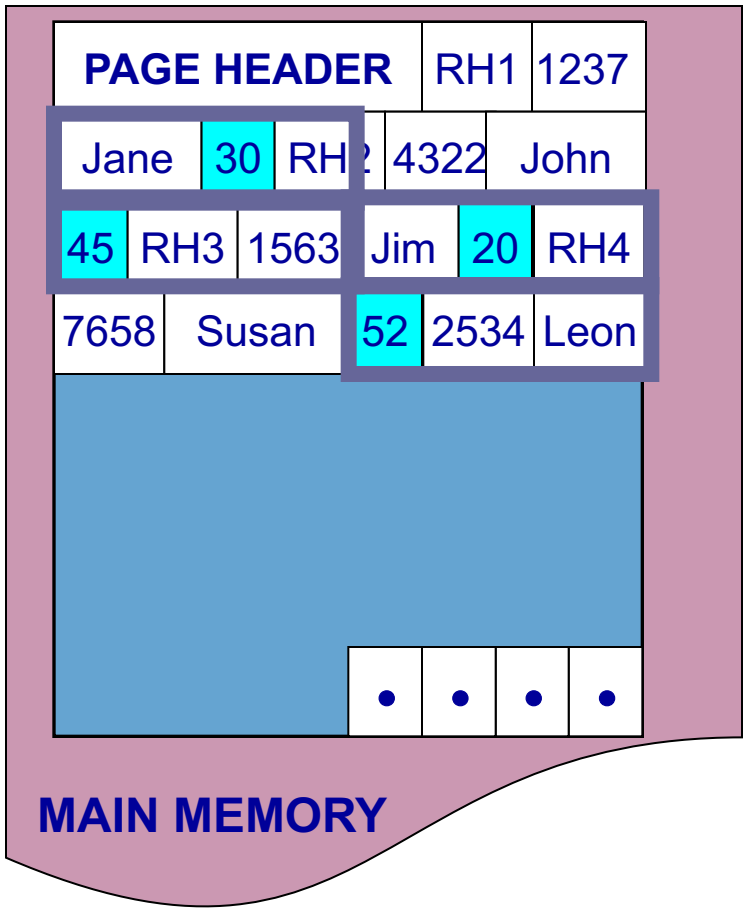


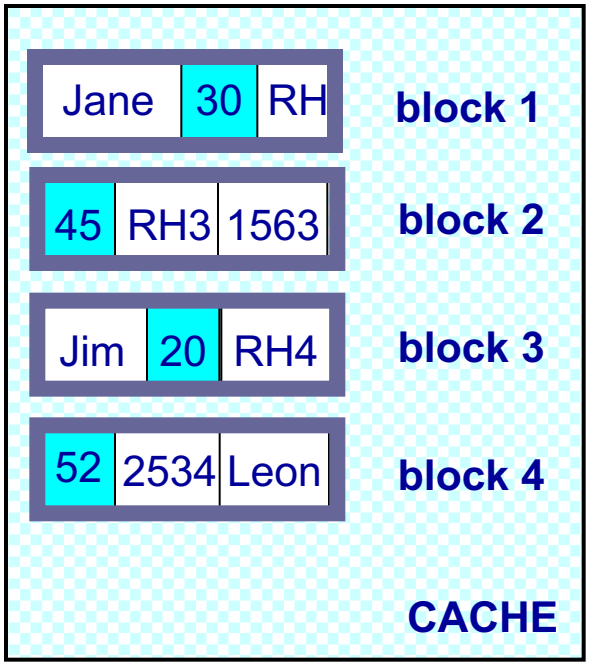

```
select name
from R
where age > 50
```

NSM pushes non-referenced data to the cache

Ailamaki VLDB'01 http://research.cs.wisc.edu/multifacet/papers/vldb01_pax_talk.ppt

Predicate Evaluation using NSM






```
select name
from R
where age > 50
```

NSM pushes non-referenced data to the cache

Ailamaki VLDB'01 http://research.cs.wisc.edu/multifacet/papers/vldb01_pax_talk.ppt

Need New Data Page Layout

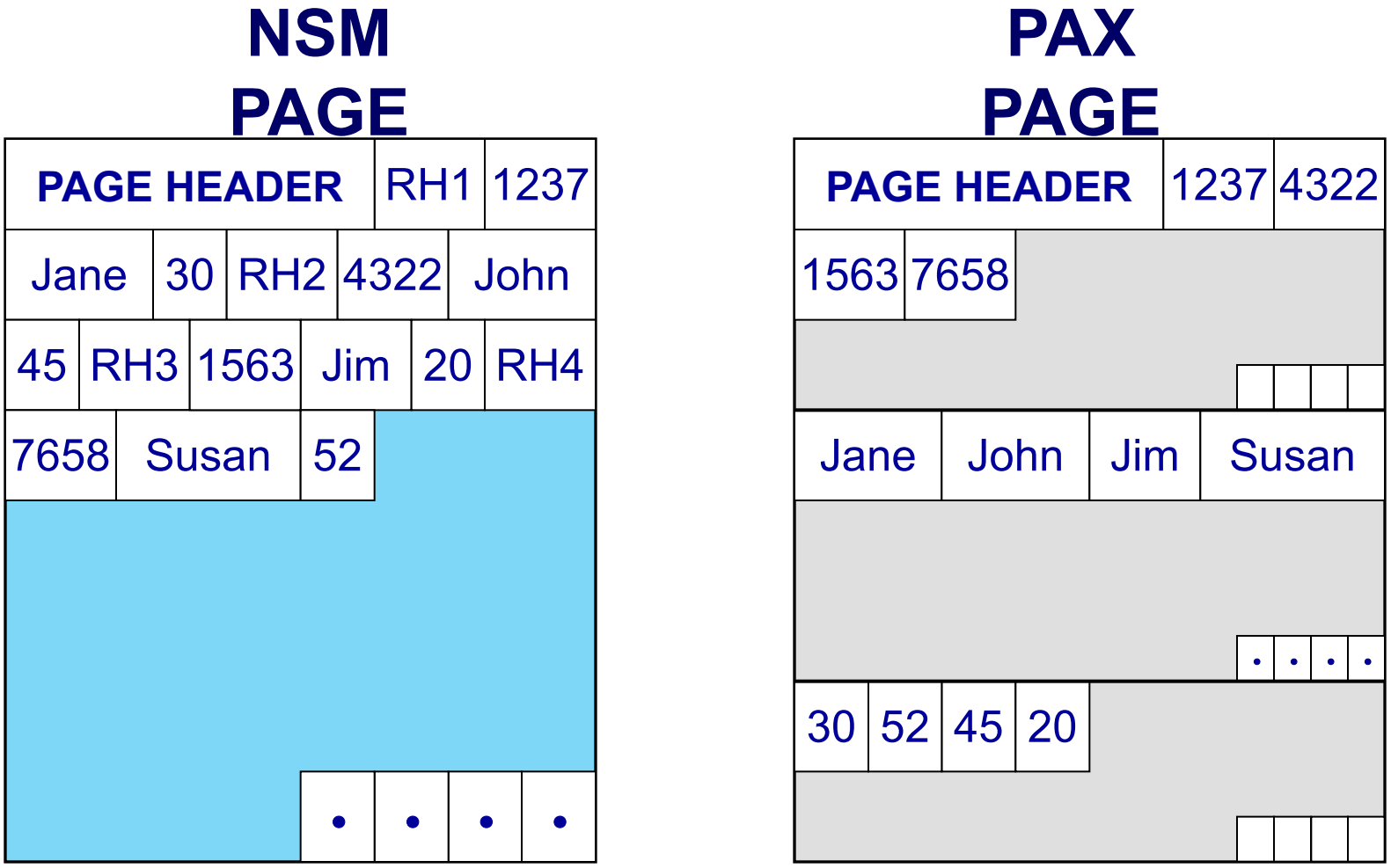
- Eliminates unnecessary memory accesses
- Improves inter-record locality
- Keeps a record's fields together
- Does not affect I/O performance

and, most importantly, is…

low-implementation-cost, high-impact

Ailamaki VLDB'01 http://research.cs.wisc.edu/multifacet/papers/vldb01_pax_talk.ppt

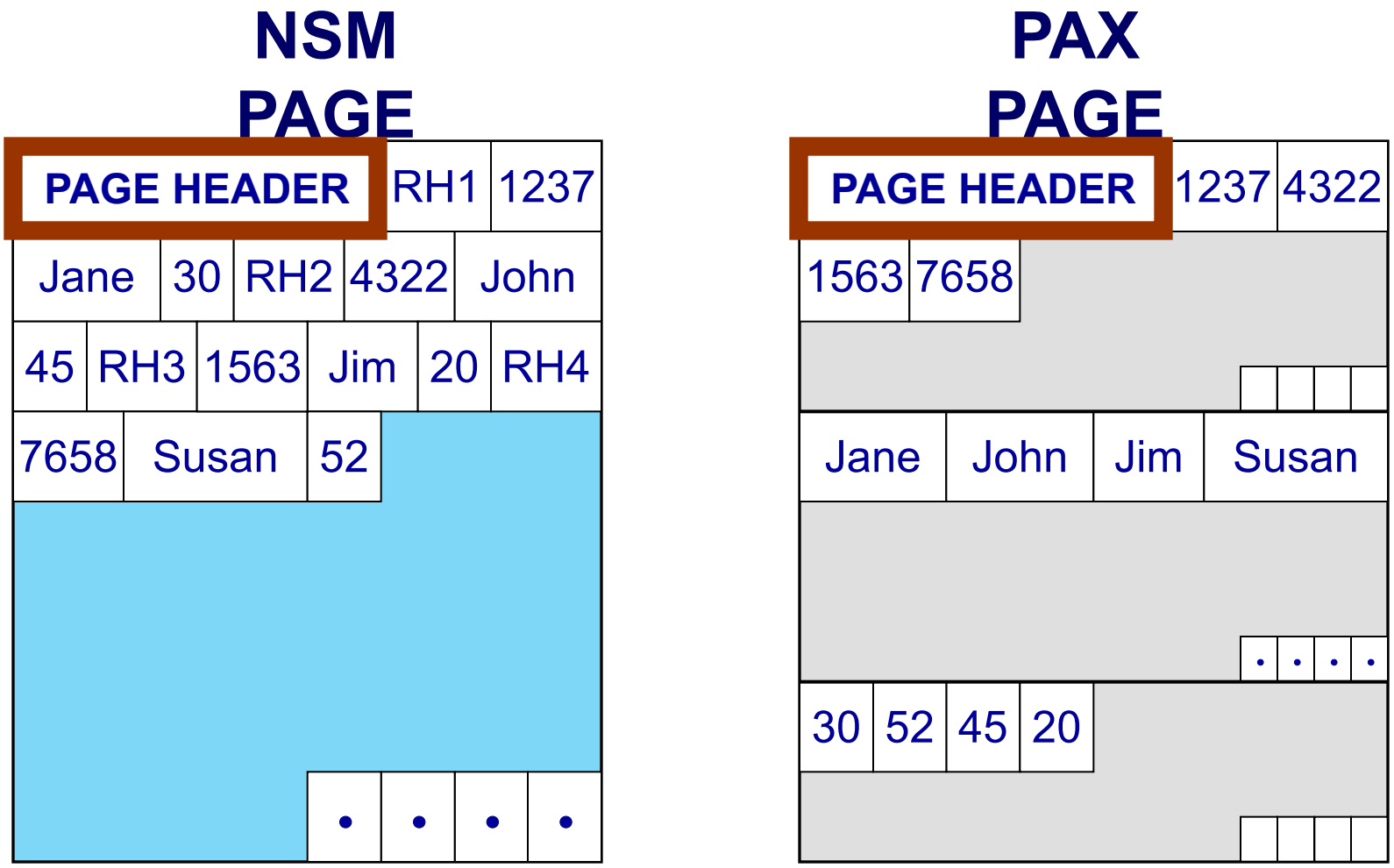
Partition Attributes Across (PAX)



Partition data *within* the page for spatial locality

Ailamaki VLDB'01 http://research.cs.wisc.edu/multifacet/papers/vldb01_pax_talk.ppt

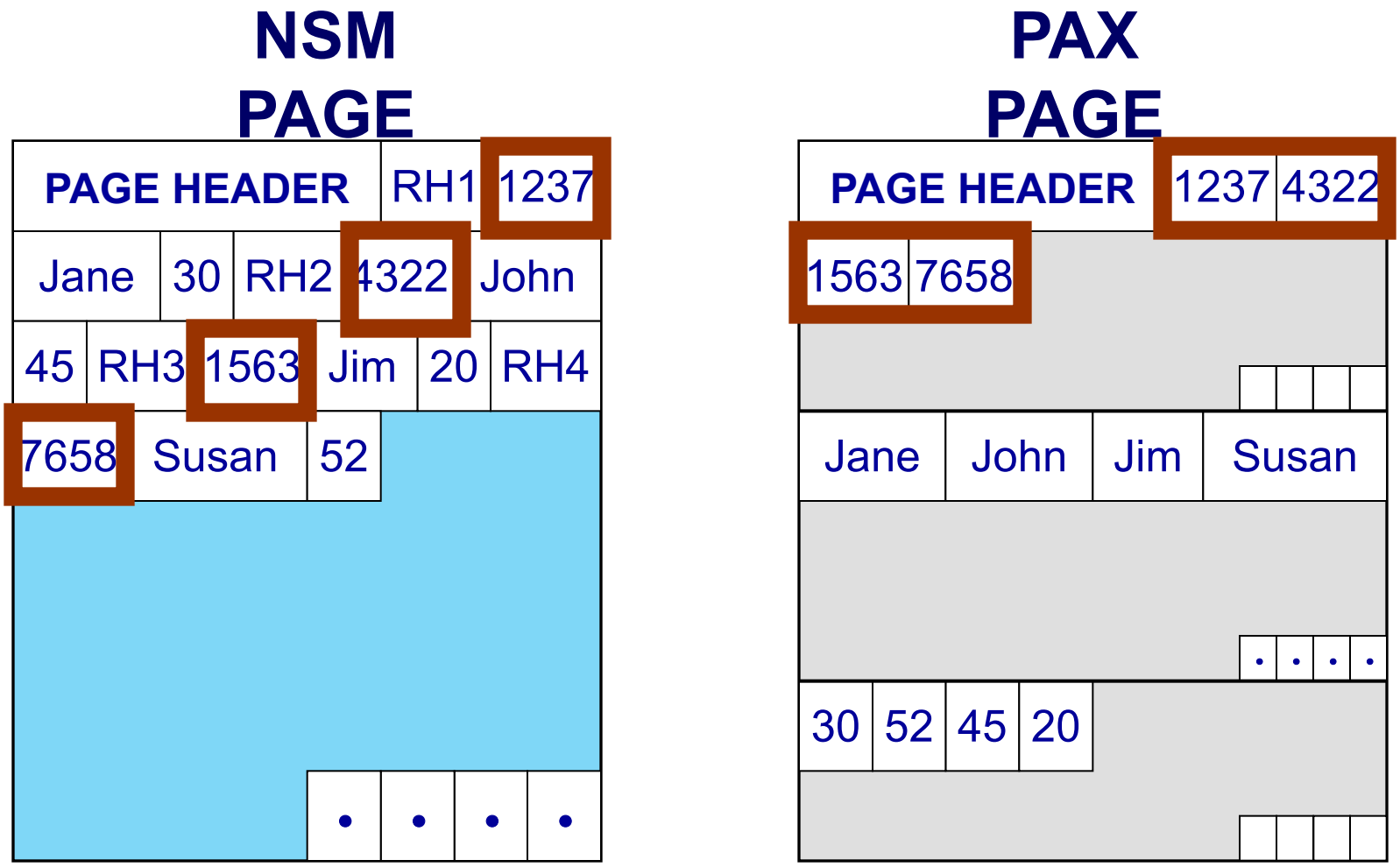
Partition Attributes Across (PAX)



Partition data *within* the page for spatial locality

Ailamaki VLDB'01 http://research.cs.wisc.edu/multifacet/papers/vldb01_pax_talk.ppt

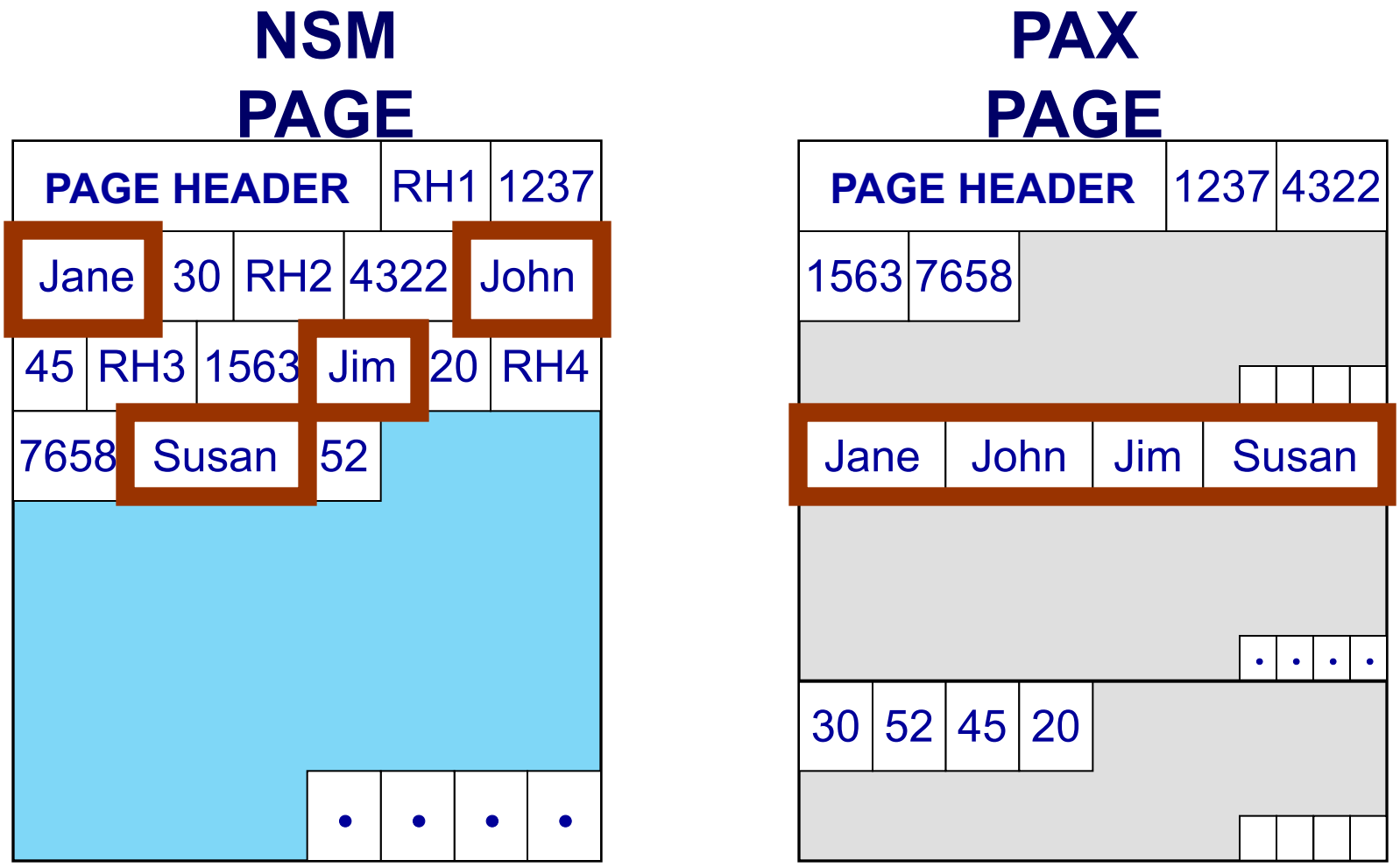
Partition Attributes Across (PAX)



Partition data *within* the page for spatial locality

Ailamaki VLDB'01 http://research.cs.wisc.edu/multifacet/papers/vldb01_pax_talk.ppt

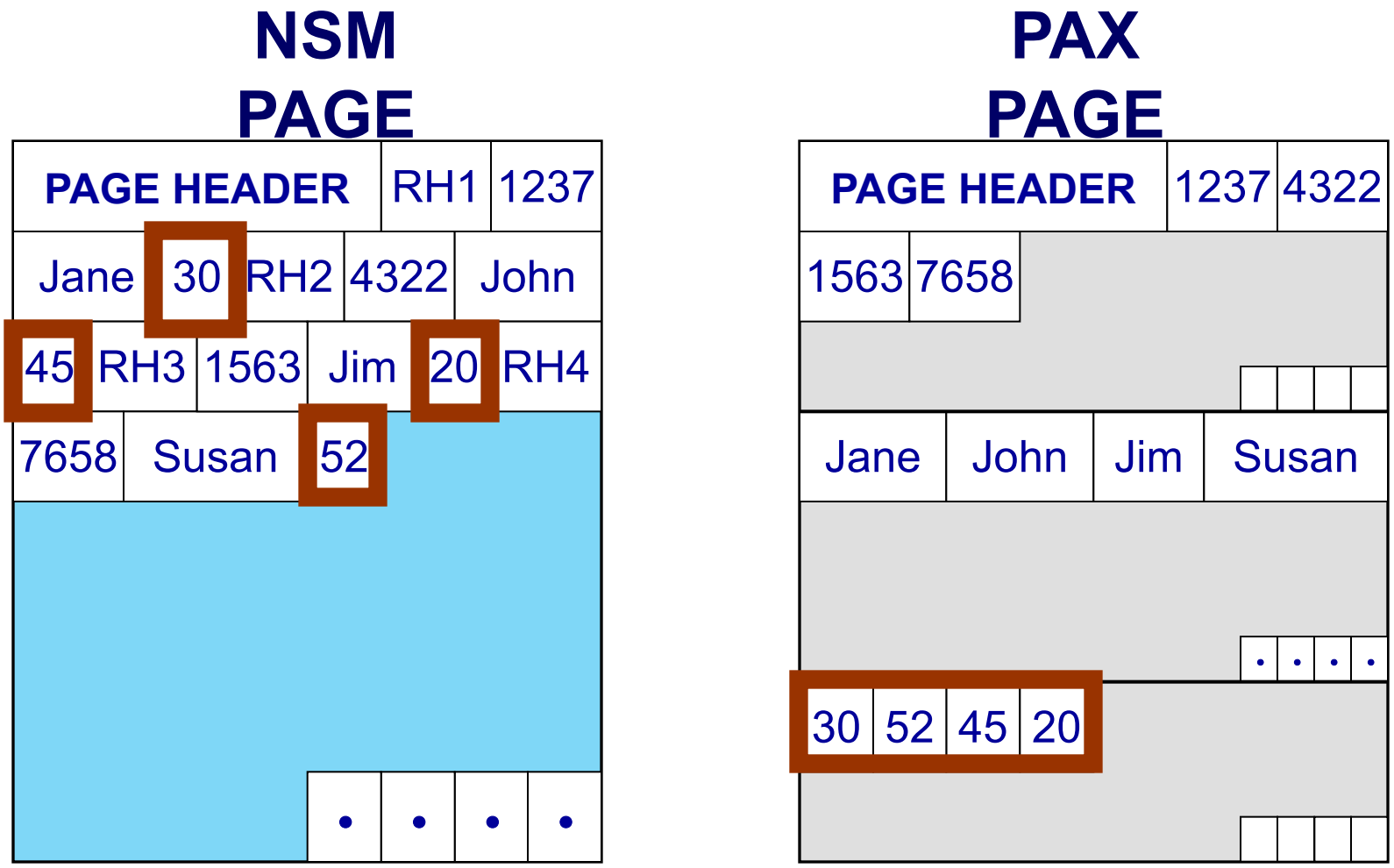
Partition Attributes Across (PAX)



Partition data *within* the page for spatial locality

Ailamaki VLDB'01 http://research.cs.wisc.edu/multifacet/papers/vldb01_pax_talk.ppt

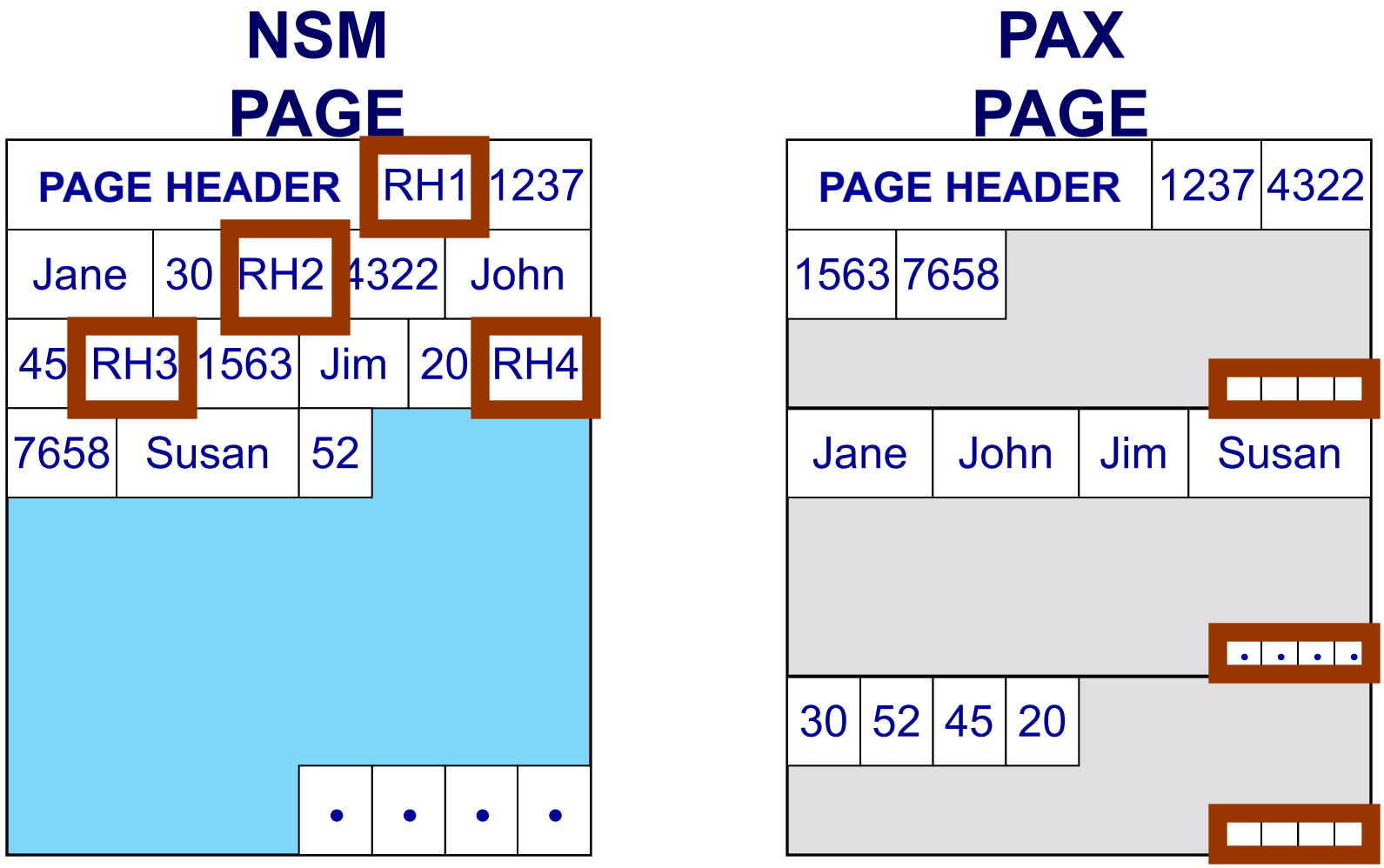
Partition Attributes Across (PAX)



Partition data *within* the page for spatial locality

Ailamaki VLDB'01 http://research.cs.wisc.edu/multifacet/papers/vldb01_pax_talk.ppt

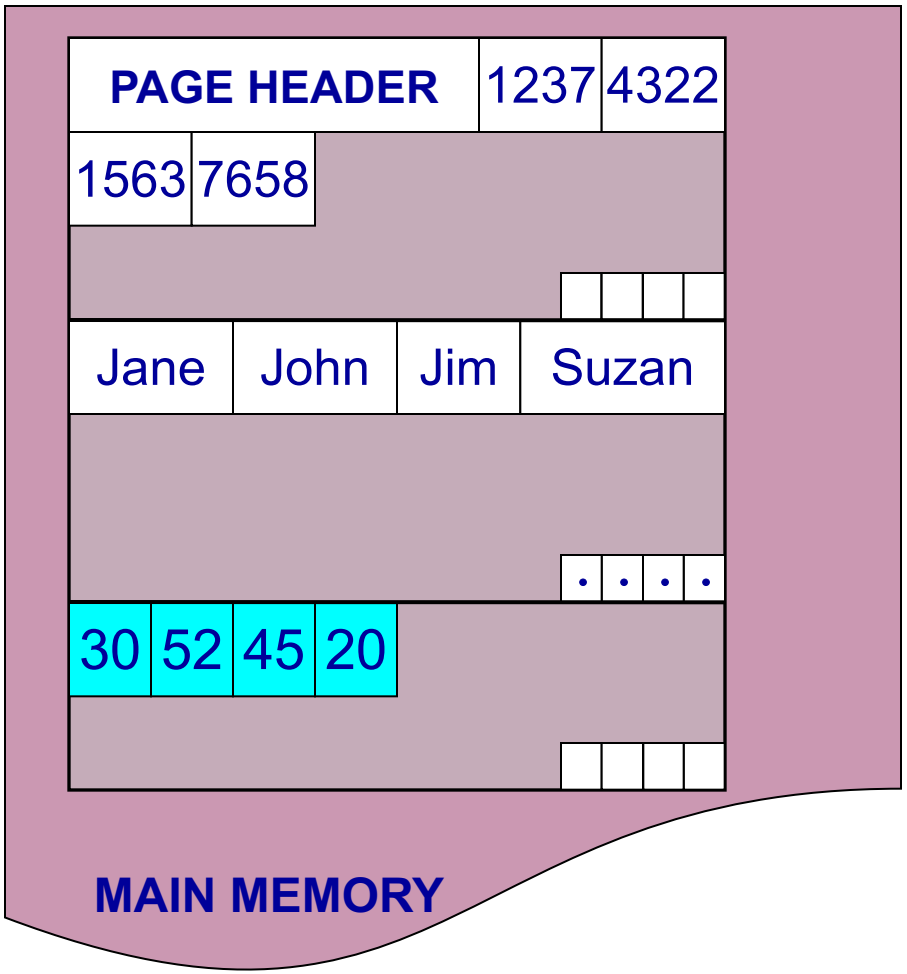
Partition Attributes Across (PAX)

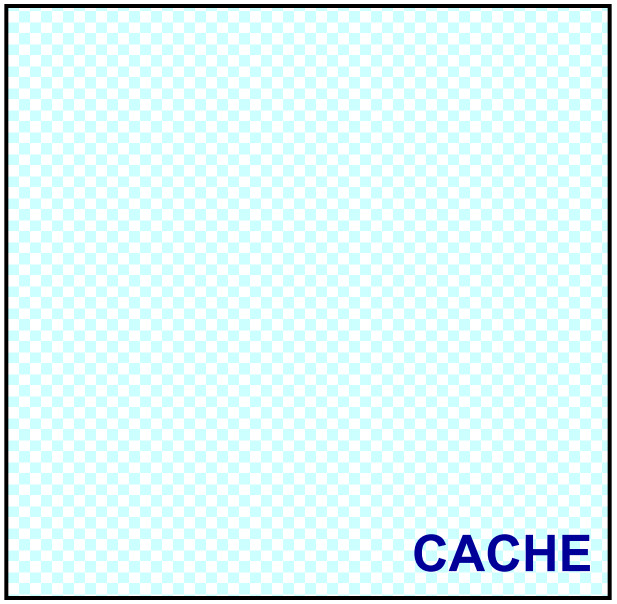


Partition data *within* the page for spatial locality

Ailamaki VLDB'01 http://research.cs.wisc.edu/multifacet/papers/vldb01_pax_talk.ppt

Predicate Evaluation using PAX



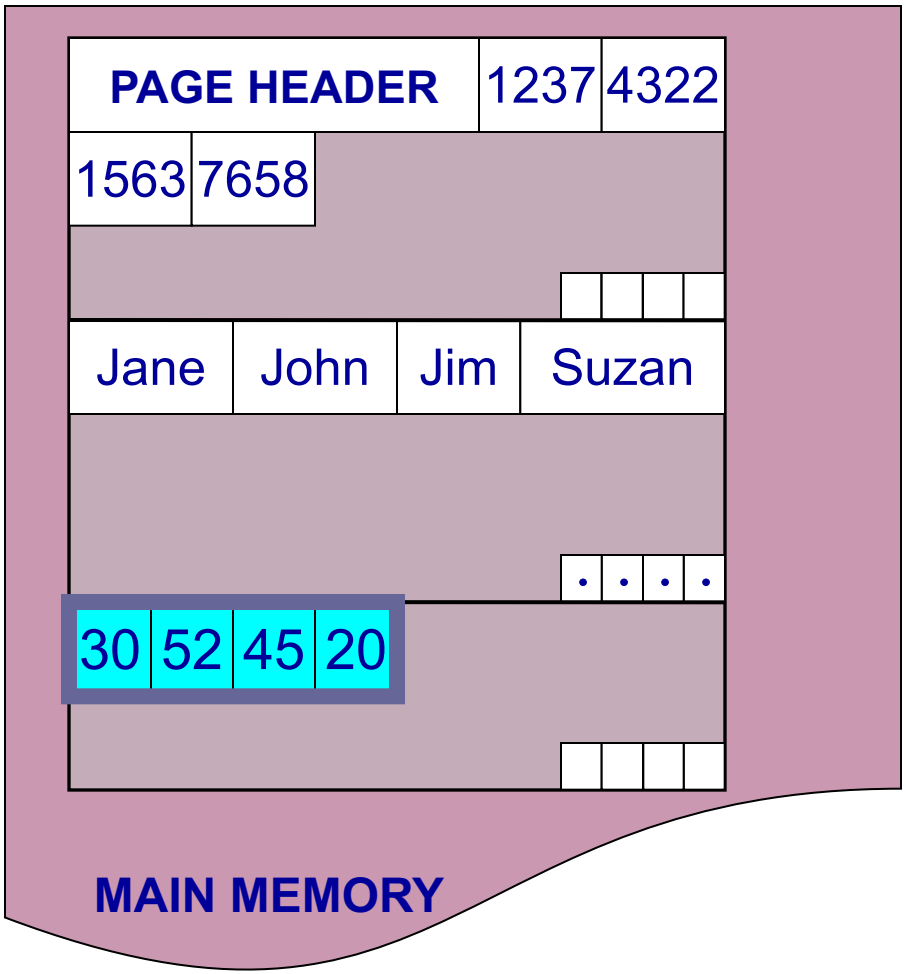


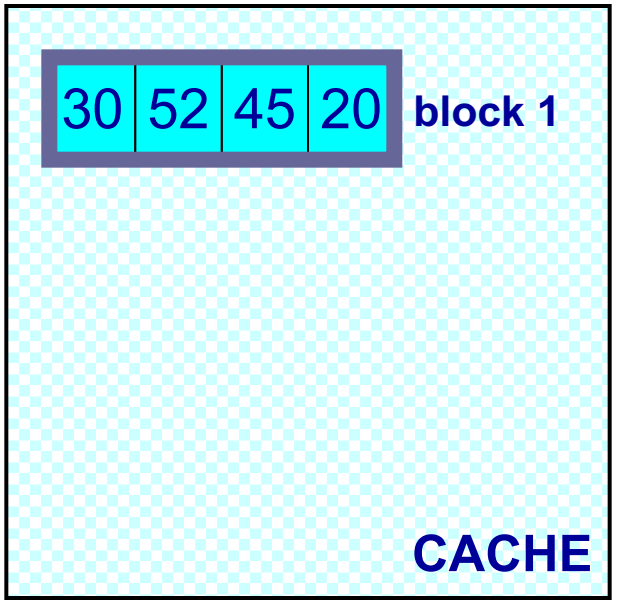

```
select name
from R
where age > 50
```

Fewer cache misses, low reconstruction cost

Ailamaki VLDB'01 http://research.cs.wisc.edu/multifacet/papers/vldb01_pax_talk.ppt

Predicate Evaluation using PAX



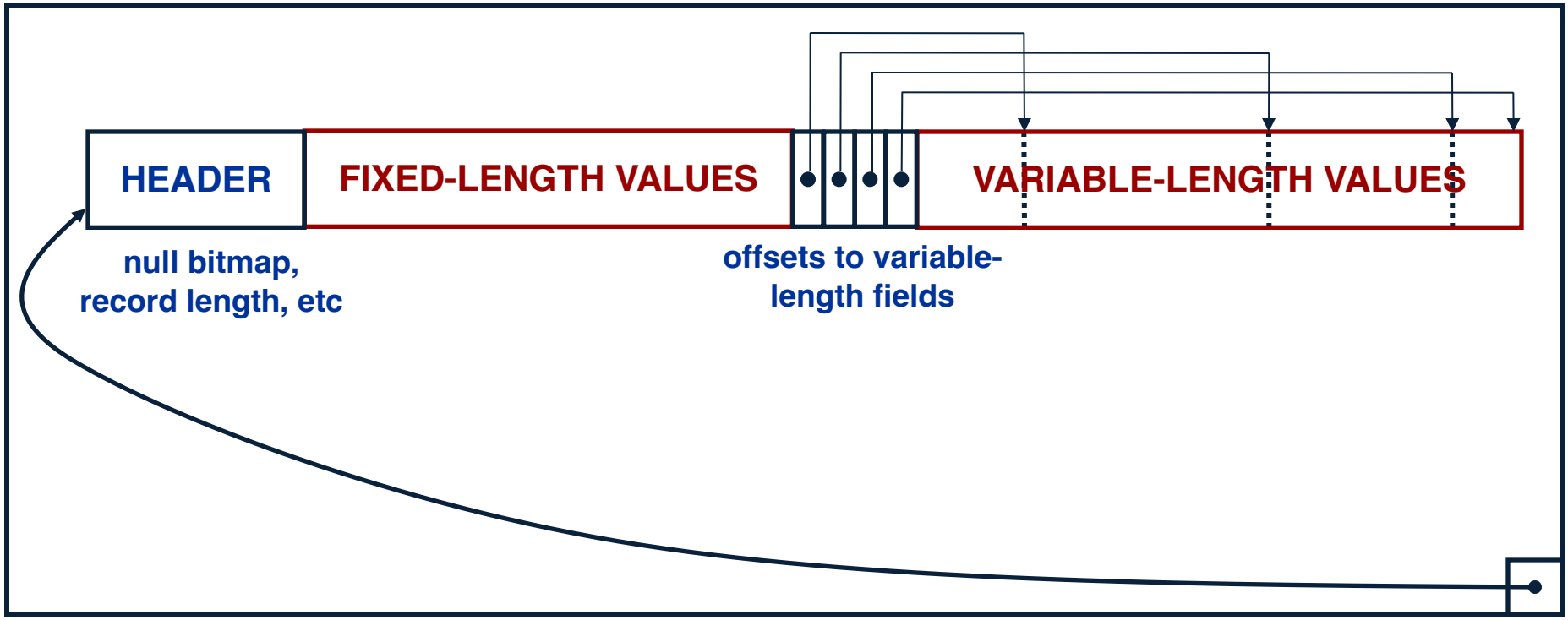



```
select name
from R
where age > 50
```

Fewer cache misses, low reconstruction cost

Ailamaki VLDB'01 http://research.cs.wisc.edu/multifacet/papers/vldb01_pax_talk.ppt

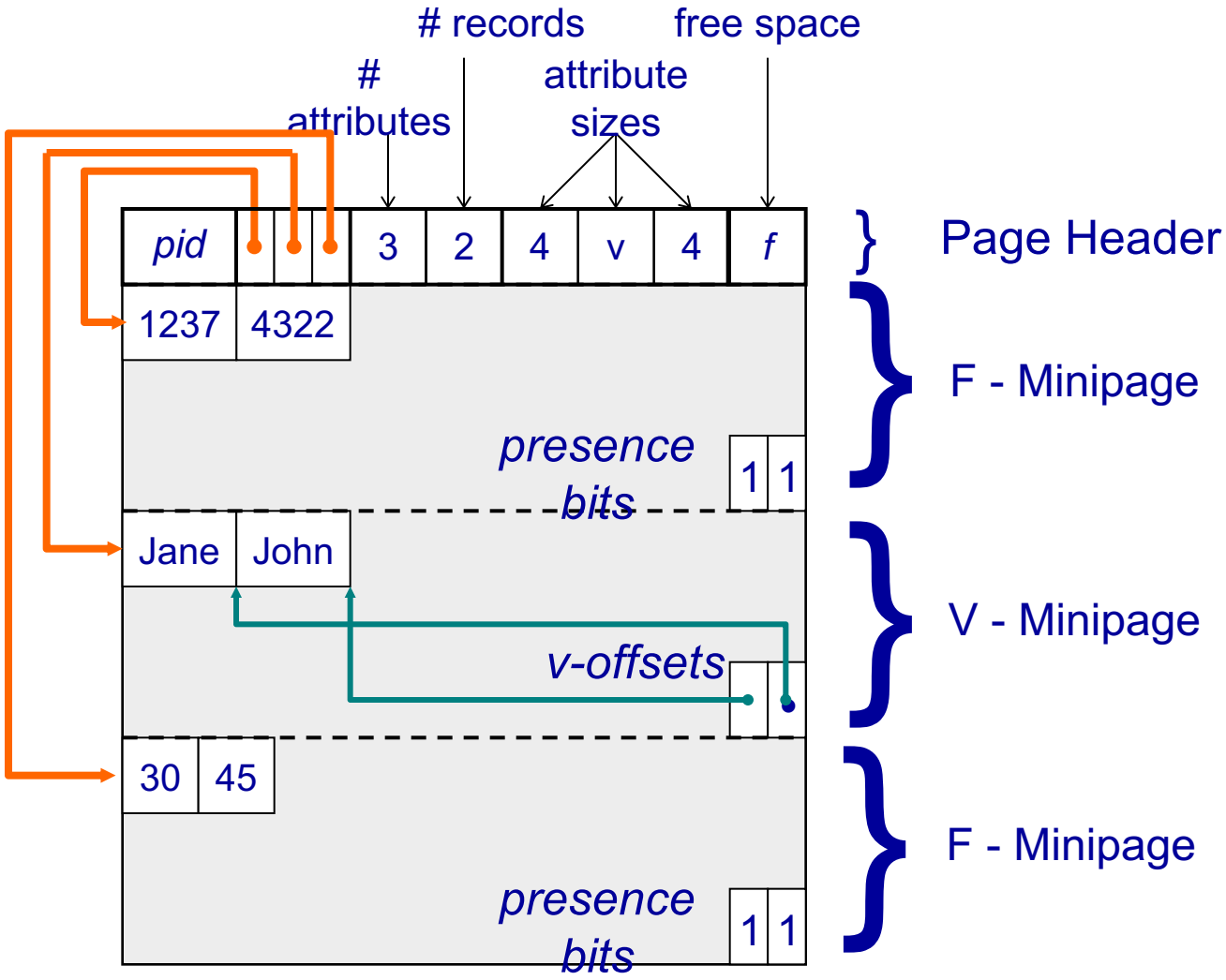
A Real NSM Record



NSM: All fields of record stored together + slots

Ailamaki VLDB'01 http://research.cs.wisc.edu/multifacet/papers/vldb01_pax_talk.ppt

PAX: Detailed Design



PAX: Group fields + amortizes record headers

Ailamaki VLDB'01 http://research.cs.wisc.edu/multifacet/papers/vldb01_pax_talk.ppt

PAX - Summary

- Improves processor cache locality
- Does not affect I/O behavior
  - Same disk accesses for NSM or PAX storage
  - No need to change the buffer manager

- Today:
  - Most (all?) commercial engines use a PAX layout of the disk
  - Beyond disk: Snowflake partitions tables horizontally into files, then uses column-store inside each file (hence, PAX)

Column-oriented Storage

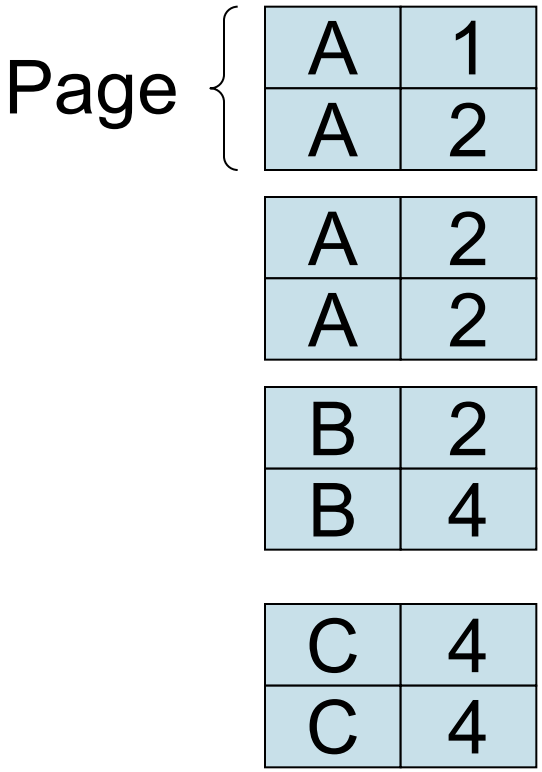
- Store each attribute in a different file

- Column 1: file0, file1, …
- Column 2: file10, file11, …

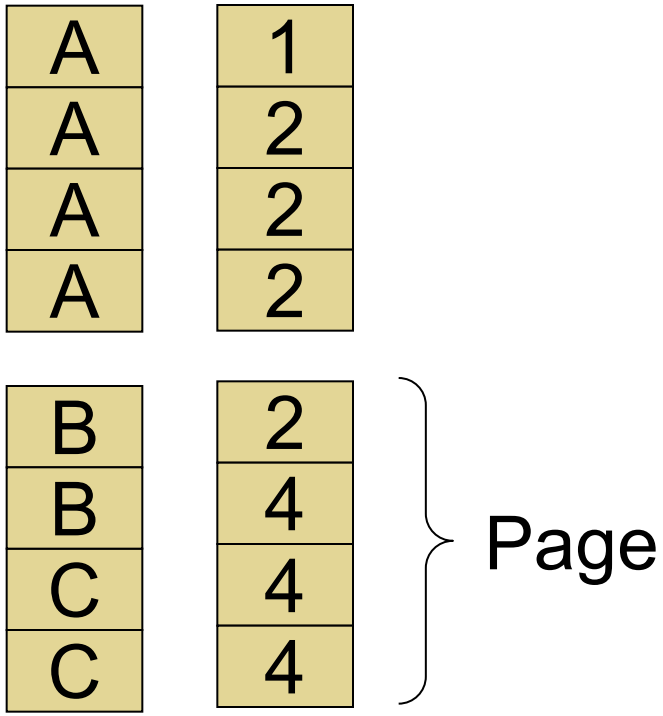
The Design and Implementation of Modern Column-Oriented Database Systems, Abadi et al., 2013

Column-oriented Storage

Row-based
(4 pages)



Column-based
(4 pages)



C-Store also avoids large tuple headers

From Row to Column Storage (Modern Designs)

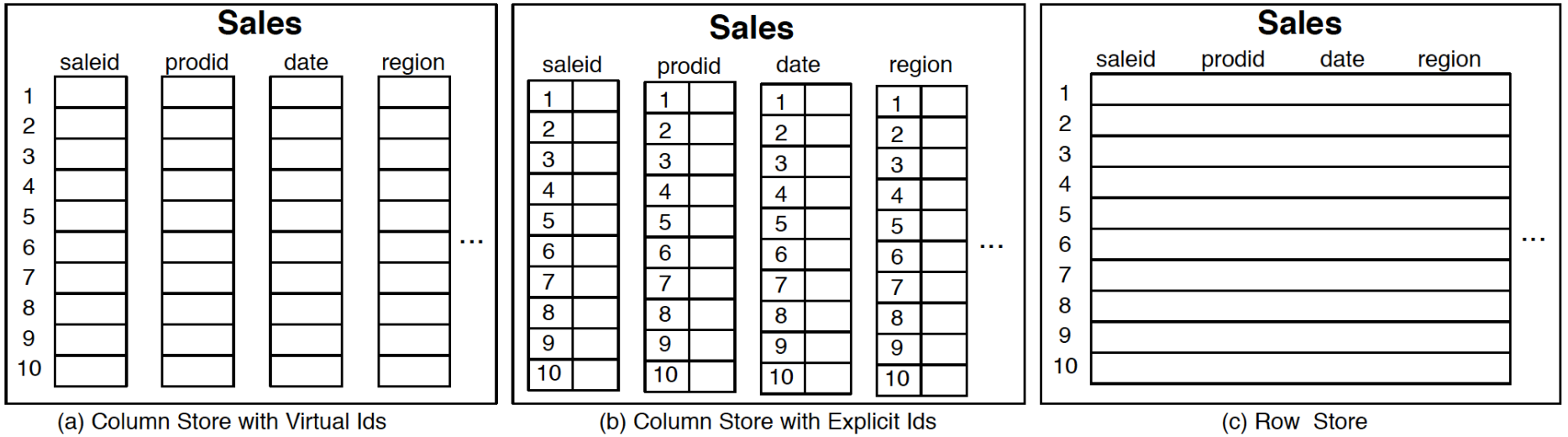


Figure 1.1: Physical layout of column-oriented vs row-oriented databases.

Basic tradeoffs:

- Reading all attributes of one records, v.s.
- Reading some attributes of many records

Column-oriented Storage

- Main idea:
  - Physical storage: complete vertical partition; each column stored separately: R.A, R.B, R.A
  - Logical schema: remains the same R(A,B,C)

- Main advantage:
  - Improved transfer rate: disk to memory, memory to CPU, better cache locality

Trade-Offs

- Row stores
  - Quick to update entire tuple (1 page IO)
  - Quick to access a single tuple

- Column stores
  - Avoid reading unnecessary columns
  - Better compression

Problem: needs an entire redesign of the DBMS

Storage Manager: Summary

- Maps between the logical view of the data and the physical storage on disk
- Storing pages on disk
- Storing data on page

- HW3!

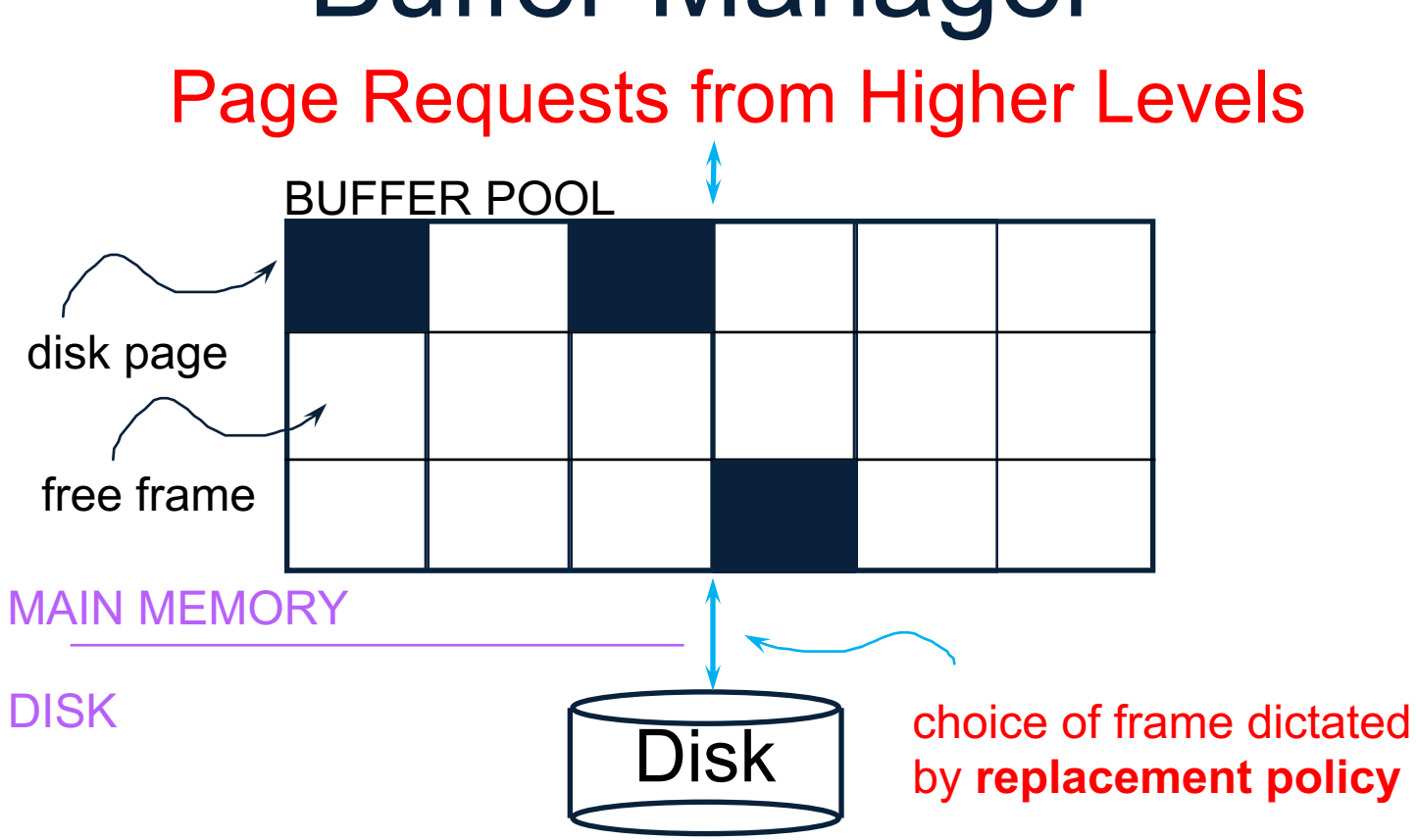
Next: the buffer pool

The Buffer Pool

The Buffer Pool

- Fixed chunk of main memory
- When a page is read from disk, it is brought to the buffer pool
- If same page is requested later, it's found in the buffer pool: saves disk I/O
- If a new page is read, but no room in the buffer pool, one page is evicted

Buffer Manager



- Data must be in RAM for DBMS to operate on it!
- Table of <frame#, pageid> pairs is maintained

Buffer Manager

Needs to decide on page replacement policy

- LRU: Least Recently Used (in class)
- Clock algorithm (on your own)

Both work well in OS, but not always in DB, mostly because of frequent sequential accesses

Summary

- Storage manager and buffer manager are a significant component of DBMS
- Key for good performance
- They also need to handle transactions, which we will not cover in 544