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

```
PUT(1002, value)  GET(1005)
```
STORAGE HIERARCHY

Volatile
Random Access
Byte-Addressable

Non-Volatile
Sequential Access
Block-Addressable

CPU Registers

CPU Caches

DRAM

SSD

HDD

Network Storage

Faster
Smaller
Expensive

Slower
Larger
Cheaper

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

CPU
- CPU Registers
- CPU Caches

Memory
- DRAM

Disk
- SSD
- HDD

Network Storage

Faster
Smaller
Expensive

Slower
Larger
Cheaper

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

Latency Numbers Every Programmer Should Know

1 ns L1 Cache Ref
4 ns L2 Cache Ref
100 ns DRAM
16,000 ns SSD
2,000,000 ns HDD
~50,000,000 ns Network Storage
1,000,000,000 ns Tape Archives

Source: Colin Scott

Credit: https://15445.courses.cs.cmu.edu/fall2023/
## ACCESS TIMES

*Latency Numbers Every Programmer Should Know*

<table>
<thead>
<tr>
<th>Access Time</th>
<th>Description</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 ns</td>
<td>L1 Cache Ref</td>
<td>1 sec</td>
</tr>
<tr>
<td>4 ns</td>
<td>L2 Cache Ref</td>
<td>4 sec</td>
</tr>
<tr>
<td>100 ns</td>
<td>DRAM</td>
<td>100 sec</td>
</tr>
<tr>
<td>16,000 ns</td>
<td>SSD</td>
<td>4.4 hours</td>
</tr>
<tr>
<td>2,000,000 ns</td>
<td>HDD</td>
<td>3.3 weeks</td>
</tr>
<tr>
<td>~50,000,000 ns</td>
<td>Network Storage</td>
<td>1.5 years</td>
</tr>
<tr>
<td>1,000,000,000 ns</td>
<td>Tape Archives</td>
<td>31.7 years</td>
</tr>
</tbody>
</table>

Source: Colin Scott

Credit: [https://15445.courses.cs.cmu.edu/fall2023/](https://15445.courses.cs.cmu.edu/fall2023/)
Disk

- Stores data persistently
- 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

[Diagram of sequential access pattern]
Sequential/Random Access

Sequential

[Diagram showing sequential access]
Sequential/Random Access

Sequential
Sequential/Random Access

Sequential

Random
Sequential/Random Access

Sequential

Random

Faster
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

Will discuss next
## Row-Oriented Storage

**Logica schema**

### Product

<table>
<thead>
<tr>
<th>Name</th>
<th>Price</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>iPhone</td>
<td>599</td>
<td>gray</td>
</tr>
<tr>
<td>Jacket</td>
<td>129</td>
<td>blue</td>
</tr>
<tr>
<td>Pants</td>
<td>89</td>
<td>black</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Bicycle</td>
<td>599</td>
<td>Red</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
### Physical layout

#### Page 0

<table>
<thead>
<tr>
<th>Product</th>
<th>Price</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>iPhone</td>
<td>599</td>
<td>gray</td>
</tr>
<tr>
<td>Jacket</td>
<td>129</td>
<td>blue</td>
</tr>
<tr>
<td>Pants</td>
<td>89</td>
<td>black</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Page 1

<table>
<thead>
<tr>
<th>Product</th>
<th>Price</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bicycle</td>
<td>599</td>
<td>Red</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Logica schema

#### Product

<table>
<thead>
<tr>
<th>Name</th>
<th>Price</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>iPhone</td>
<td>599</td>
<td>gray</td>
</tr>
<tr>
<td>Jacket</td>
<td>129</td>
<td>blue</td>
</tr>
<tr>
<td>Pants</td>
<td>89</td>
<td>black</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

...
Row-Oriented Storage

<table>
<thead>
<tr>
<th>Product</th>
<th>Price</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>iPhone</td>
<td>599</td>
<td>gray</td>
</tr>
<tr>
<td>Jacket</td>
<td>129</td>
<td>blue</td>
</tr>
<tr>
<td>Pants</td>
<td>89</td>
<td>black</td>
</tr>
</tbody>
</table>

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)

<table>
<thead>
<tr>
<th>Slot₁</th>
<th>Slot₂</th>
<th>Slotₙ</th>
<th>Slotₙ+₁</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Free Sp.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>N</td>
</tr>
</tbody>
</table>

How do we insert a new record?

Number of records
### 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)

<table>
<thead>
<tr>
<th>Slot₁</th>
<th>Slot₂</th>
<th>Slotₙ</th>
<th>Slotₙ₊₁</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Free Sp.</strong></td>
</tr>
</tbody>
</table>

#### Number of records

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td></td>
</tr>
</tbody>
</table>

**How do we insert a new record?**

**How do we delete a record?**
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

Free space
Page Format Approach 2

Slot directory

Free space

4 F
Page Format Approach 2

Free space

Slot directory

Why at the end of the page?
Page Format Approach 2

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

Why at the end of the page?

Slot directory
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)

| Field 1 | Field 2 | . . . | . . . | Field K |

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:

Disk → Main memory → Cache* → CPU

*aka CPU cache; several! L3, L2, L1 cache
### Current Scheme: Slotted Pages

**Formal name: NSM (N-ary Storage Model)**

Records are stored sequentially

- Offsets to start of each record at end of page

<table>
<thead>
<tr>
<th>RID</th>
<th>SSN</th>
<th>Name</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1237</td>
<td>Jane</td>
<td>30</td>
</tr>
<tr>
<td>2</td>
<td>4322</td>
<td>John</td>
<td>45</td>
</tr>
<tr>
<td>3</td>
<td>1563</td>
<td>Jim</td>
<td>20</td>
</tr>
<tr>
<td>4</td>
<td>7658</td>
<td>Susan</td>
<td>52</td>
</tr>
<tr>
<td>5</td>
<td>2534</td>
<td>Leon</td>
<td>43</td>
</tr>
<tr>
<td>6</td>
<td>8791</td>
<td>Dan</td>
<td>37</td>
</tr>
</tbody>
</table>

PAGE HEADER

- RH1: 1237
- RH2: 4322
- RH3: 1563
- RH4: 7658

Formal name: NSM (N-ary Storage Model)

Current Scheme: Slotted Pages

Ailamaki VLDB’01 [http://research.cs.wisc.edu/multifacet/papers/vldb01_pax_talk.ppt](http://research.cs.wisc.edu/multifacet/papers/vldb01_pax_talk.ppt)
## Predicate Evaluation using NSM

### Query:
```sql
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](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](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
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](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
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)

### NSM PAGE

<table>
<thead>
<tr>
<th>PAGE HEADER</th>
<th>RH1</th>
<th>1237</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jane</td>
<td>30</td>
<td>RH2</td>
</tr>
<tr>
<td>45</td>
<td>1563</td>
<td>Jim</td>
</tr>
<tr>
<td>7658</td>
<td>Susan</td>
<td>52</td>
</tr>
</tbody>
</table>

### PAX PAGE

<table>
<thead>
<tr>
<th>PAGE HEADER</th>
<th>1237</th>
<th>4322</th>
</tr>
</thead>
<tbody>
<tr>
<td>1563</td>
<td>7658</td>
<td></td>
</tr>
<tr>
<td>Jane</td>
<td>John</td>
<td>Jim</td>
</tr>
<tr>
<td>30</td>
<td>52</td>
<td>45</td>
</tr>
</tbody>
</table>

Partition data **within** the page for spatial locality

Ailamaki VLDB’01 [http://research.cs.wisc.edu/multifacet/papers/vldb01_pax_talk.ppt](http://research.cs.wisc.edu/multifacet/papers/vldb01_pax_talk.ppt)
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](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](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

FIRST PAGE HEADER

PAGE HEADER 1237 4322
1563 7658

JANE  JOHN  JIM  SUZAN

30 52 45 20

MAIN MEMORY

CACHE

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

 Fewer cache misses, low reconstruction cost

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

null bitmap, record length, etc

offsets to variable-length fields

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, ...
Column-oriented Storage

Row-based (4 pages)

Page

A 1
A 2
A 2
A 2
B 2
B 4
C 4
C 4

Column-based (4 pages)

Page

C-Store also avoids large tuple headers

B 2
B 4
C 4
C 4

A 1
A 2
A 2
A 2
1 2
2 2
2 2
2 4
4 4
4 4
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 record, 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

Page Requests from Higher Levels

- 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