Weaving Relations for Cache Performance

Anastassia Ailamaki, David J. DeWitt, Mark D. Hill, Marios Skounakis

Presented by, Nosayba El-Sayed
Why?

- DBMSs → Compute and memory bound
"DBMSs on a modern processor: Where does time go?" (VLDB'99)
Improve Data Placement

- Improve data placement
  - Avoid loading cache with useless data
  - Reduce cache misses

- How is data placed?

<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>
</tbody>
</table>
# Slotted Pages

<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>
</tbody>
</table>

- Records are stored sequentially
- Offsets to start of each record at end of page
- NSM (N-ary Storage Model)

Records are stored sequentially.

Offsets to start of each record at end of page.

NSM (N-ary Storage Model)
Slotted Pages: Problem?

select name
from R
where age > 50

NSM pushes non-referenced data to the cache!
Vertical Partitioning

<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>
</tbody>
</table>

- N attributes → N sub-relations
- DSM (Decomposition Storage Model)
Vertical Partitioning: Problem?

Select name from R where age > 50

Sub-relation R3

Cache
Vertical Partitioning: Problem?

Sub-relation R3

Sub-relation R1

JOIN
Slotted vs Vertical

**Slotted (NSM)**
- Poor cache performance
- Low record reconstruction cost

**Vertical (DSM)**
- Inter-record spatial locality
- High record reconstruction cost

→ Solution? **Partition Attributes Across (PAX)**
Partition Attributes Across (PAX)

Partition data \textit{within} the 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>RH3</td>
<td>1563</td>
<td>Jim</td>
</tr>
<tr>
<td>Susan</td>
<td>52</td>
<td></td>
</tr>
<tr>
<td>7658</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<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>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Jane</th>
<th>John</th>
<th>Jim</th>
<th>Susan</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>52</td>
<td>45</td>
<td>20</td>
</tr>
</tbody>
</table>
PAX: Cache Behavior

<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>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Jane</th>
<th>John</th>
<th>Jim</th>
<th>Suzan</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>52</td>
<td>45</td>
<td>20</td>
<td>...</td>
</tr>
</tbody>
</table>

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

Fewer cache misses, low reconstruction cost
PAX: Detailed Design

Page Header

F - Minipage

V - Minipage

F - Minipage
Evaluation (NSM-DSM-PAX)

Experimental Setup
• Windows NT4.0
• Pentium II Xeon/MT
• 16KB L1-I, 16KB L1-D, 512 KB L2, 512 MB RAM
• Implemented on top of Shore Storage Manager

Query?
```sql
select avg (ai) 
from R 
where aj >= Lo and aj <= Hi
```
- PAX saves 70% of NSM’s data cache penalty
- PAX reduces cache misses at both L1 and L2
Sensitivity Analysis

Performance of the two techniques converges!
DSS Benchmarks

- Queries
  - Bulk-loading
  - Range Selections
  - TPC-H Q1 and Q6
    → sequential scans
    → aggregates (sum, avg, count)
    → grouping/ordering of results
  - TPC-H Q11 and Q14
    → (Adaptive Hybrid) Hash Join

- 100M, 200M, and 500M TPC-H DBs
Bulk-loading/Insertions

- PAX Implementation
  → Reorganize minipage boundaries as records are inserted
  → Adjust average field sizes

- PAX loads a TPC-H database in 2-10% longer than NSM
TPC-H Queries

PAX/NSM Speedup on PII/NT

PAX improves performance up to 42% even with I/O
Updates

- Policy: Update in-place
- Variable-length: Shift when needed
- PAX only needs shift minipage data
Updates with various selectivities

PAX/NSM Speedup

Updates: Speedup

Number of updated attributes

PAX always speeds queries up (10-16%)
Summary

✓ PAX: a low-cost, high-impact DP technique

✓ Improves Cache Performance
  - Eliminates unnecessary memory references
  - High utilization of cache space/bandwidth

✓ Faster than NSM (does not affect I/O)

✓ Usability: Orthogonal to other storage decisions

✓ “Easy” to implement in large existing DBMSs
Discussion

- What about TPC-C (OLTP)? Any data layout policy should exhibit relatively good performance in both the two types of workloads.

- What happens after deletions? Overhead for page re-organization.

- Experimental setup very convenient for PAX? Fixed-line queried attributes, exactly 4 times smaller than cache block size.

- Lacks elaboration on changes made to the "page level data manipulation code"?