CS 457: Database Management Systems

Lecture 13
Query Execution (cont’d)
Logistics

• HW2
  – .exit is back
    • Talk to me/TA if you found it hard to implement
  – Extended to after Spring week?
    • Pro: you’ll have more time
    • Con: more of your vacation time

• Midterm exam preview
  – This Thursday!
Operator Algorithms
Operator Algorithms

Design criteria

• Cost: IO, CPU, Network

• Memory utilization

• Load balance (for parallel operators)
Cost Parameters

• **Cost = total number of I/Os**
  - This is a simplification that ignores CPU, network

• **Parameters:**
  - \( B(R) \) = # of blocks (i.e., pages) for relation R
  - \( T(R) \) = # of tuples in relation R
  - \( V(R, a) \) = # of distinct values of attribute a
    - When \( a \) is a key, \( V(R,a) = T(R) \)
    - When \( a \) is not a key, \( V(R,a) \) can be anything < \( T(R) \)
Convention

- Cost = the cost of *reading* operands from disk

- Cost of *writing* the result to disk is *not included*; need to count it separately when applicable
Outline

• Join operator algorithms
  – One-pass algorithms
  – Index-based algorithms
  – Two-pass algorithms

• Note about readings:
  – In class, we discuss only algorithms for joins
  – Other operators are easier: read the book
Join Algorithms

• Hash join
• Nested loop join
• Sort-merge join
Hash Join

Hash join: $R \bowtie S$

- Scan $R$, build buckets in main memory
- Then scan $S$ and join
- Cost: $B(R) + B(S)$

- One-pass algorithm when $B(R) \leq M$
Hash Join Example

Patient(pid, name, address)
Insurance(pid, provider, policy_nb)

Patient \( \Join \) Insurance

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>'Bob'</td>
<td>'Ela'</td>
<td>'Jill'</td>
<td>'Joe'</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>'Seattle'</td>
<td>'Everett'</td>
<td>'Kent'</td>
<td>'Seattle'</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>2</th>
<th>4</th>
<th>4</th>
<th>3</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>'Blue'</td>
<td>'Prem'</td>
<td>'Prem'</td>
<td>'GrpH'</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>123</td>
<td>432</td>
<td>343</td>
<td>554</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Hash Join Example

Patient ▷ Insurance

<table>
<thead>
<tr>
<th>Patient</th>
<th>Insurance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>9</td>
<td>6</td>
</tr>
<tr>
<td>8</td>
<td>5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>2</th>
<th>4</th>
<th>6</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>3</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>9</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Disk

Memory M = 21 pages

Some large-enough nb

Showing pid only

This is one page with two tuples
Hash Join Example

Step 1: Scan Patient and build hash table in memory

Memory M = 21 pages
Hash h: pid % 5

Disk
Patient | Insurance
---|---
1 | 2
3 | 4
9 | 6
8 | 5

Input buffer
Hash Join Example

Step 2: Scan Insurance and probe into hash table

Memory M = 21 pages
Hash h: pid % 5

Disk

Patient  Insurance
1 2 3 4 5 6 7 8 9
2 4 6 6
4 3 1 3
9 6
2 8
8 9

Input buffer

Output buffer

Write to disk or pass to next operator
Hash Join Example

Step 2: Scan Insurance and probe into hash table

Memory M = 21 pages
Hash h: pid % 5

Disk

<table>
<thead>
<tr>
<th>Patient</th>
<th>Insurance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2</td>
<td>2 4</td>
</tr>
<tr>
<td>3 4</td>
<td>4 3</td>
</tr>
<tr>
<td>9 6</td>
<td>2 8</td>
</tr>
<tr>
<td>8 5</td>
<td>8 9</td>
</tr>
</tbody>
</table>

Input buffer

Output buffer

CS 457 - Spring 2018
Hash Join Example

Step 2: Scan Insurance and probe into hash table

Memory $M = 21$ pages

Input buffer

Output buffer

Patient | Insurance
--- | ---
1 2 | 2 4 6 6
3 4 | 4 3 1 3
9 6 | 2 8
8 5 | 8 9

Hash $h: \text{pid} \% 5$

Keep going until read all of Insurance

Cost: $B(R) + B(S)$
Nested Loop Joins

• Tuple-based nested loop $R \bowtie S$
• $R$ is the outer relation, $S$ is the inner relation

```
for each tuple $t_1$ in $R$ do
    for each tuple $t_2$ in $S$ do
        if $t_1$ and $t_2$ join then output $(t_1,t_2)$
```

What is the Cost?
Nested Loop Joins

- Tuple-based nested loop $R \bowtie S$
- $R$ is the outer relation, $S$ is the inner relation

```
for each tuple $t_1$ in $R$ do
  for each tuple $t_2$ in $S$ do
    if $t_1$ and $t_2$ join then output $(t_1, t_2)$
```

- **Cost:** $B(R) + T(R) B(S)$
- Multiple-pass since $S$ is read many times

What is the Cost?
for each page of tuples r in R do
    for each page of tuples s in S do
        for all pairs of tuples $t_1$ in r, $t_2$ in s
            if $t_1$ and $t_2$ join then output $(t_1,t_2)$

What is the Cost?
Page-at-a-time Refinement

\[
\begin{align*}
&\text{for each page of tuples } r \text{ in } R \text{ do} \\
&\quad \text{for each page of tuples } s \text{ in } S \text{ do} \\
&\quad \quad \text{for all pairs of tuples } t_1 \text{ in } r, t_2 \text{ in } s \\
&\quad \quad \quad \text{if } t_1 \text{ and } t_2 \text{ join then output } (t_1, t_2)
\end{align*}
\]

• Cost: \( B(R) + B(R)B(S) \)
Page-at-a-time Refinement

Disk

Patient  Insurance

1 2
3 4
9 6
8 5

Input buffer for Patient

2 4
4 3
1 3

Input buffer for Insurance

1 2
2 4

Output buffer

1 2
8 9
Page-at-a-time Refinement

Disk

Patient

1 2
3 4
9 6
8 5

Insurance

2 4
6 6
4 3
1 3
2 8
8 9

Input buffer for Patient

Input buffer for Insurance

Output buffer

CS 457 - Spring 2018
Page-at-a-time Refinement

Cost: $B(R) + B(R)B(S)$
Block-Nested-Loop Refinement

for each group of M-1 pages r in R do
  for each page of tuples s in S do
    for all pairs of tuples t₁ in r, t₂ in s
      if t₁ and t₂ join then output (t₁, t₂)

What is the Cost?
Block-Nested-Loop Refinement

\[
\text{for each group of } M-1 \text{ pages } r \text{ in } R \text{ do} \\
\quad \text{for each page of tuples } s \text{ in } S \text{ do} \\
\quad \quad \text{for all pairs of tuples } t_1 \text{ in } r, t_2 \text{ in } s \\
\quad \quad \quad \text{if } t_1 \text{ and } t_2 \text{ join then output } (t_1,t_2)
\]

- Cost: \( B(R) + \frac{B(R)B(S)}{(M-1)} \)
Sort-Merge Join

Sort-merge join: $R \bowtie S$

• Scan $R$ and sort in main memory
• Scan $S$ and sort in main memory
• Merge $R$ and $S$
• Cost: $B(R) + B(S)$
• One pass algorithm when $B(S) + B(R) \leq M$
• Typically, this is NOT a one pass algorithm
  – Think about it.. (vs. Hash Join)
Sort-Merge Join Example

Step 1: Scan Patient and sort in memory

Disk

<table>
<thead>
<tr>
<th>Patient</th>
<th>Insurance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2</td>
<td>2 4</td>
</tr>
<tr>
<td>3 4</td>
<td>4 3</td>
</tr>
<tr>
<td>9 6</td>
<td>2 8</td>
</tr>
<tr>
<td>8 5</td>
<td>8 9</td>
</tr>
</tbody>
</table>

Memory M = 21 pages

[1 2 3 4 5 6 8 9]
Sort-Merge Join Example

Step 2: Scan Insurance and sort in memory

Disk

<table>
<thead>
<tr>
<th>Patient</th>
<th>Insurance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1  2</td>
<td>2  4</td>
</tr>
<tr>
<td>3  4</td>
<td>4  3</td>
</tr>
<tr>
<td>9  6</td>
<td>2  8</td>
</tr>
<tr>
<td>8  5</td>
<td>8  9</td>
</tr>
</tbody>
</table>

Memory M = 21 pages

1  2  3  4  5  6  8  9
1  2  2  3  3  4  4  6
6  8  8  9
Sort-Merge Join Example

Step 3: Merge Patient and Insurance

Disk

<table>
<thead>
<tr>
<th>Patient</th>
<th>Insurance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2</td>
<td>2 4</td>
</tr>
<tr>
<td>3 4</td>
<td>4 3</td>
</tr>
<tr>
<td>9 6</td>
<td>2 8</td>
</tr>
<tr>
<td>8 5</td>
<td>8 9</td>
</tr>
</tbody>
</table>

Memory M = 21 pages

Output buffer
Sort-Merge Join Example

Step 3: **Merge** Patient and Insurance

Memory \( M = 21 \) pages

Keep going until end of first relation
Outline

• Join operator algorithms
  – One-pass algorithms
  – Index-based algorithms
  – Two-pass algorithms
Index Based Selection

Selection on equality: $\sigma_{a=v}(R)$

- $B(R)$ = size of $R$ in blocks
- $T(R)$ = number of tuples in $R$
- $V(R, a)$ = # of distinct values of attribute $a$
Index Based Selection

Selection on equality: $\sigma_{a=v}(R)$

- $B(R) =$ size of $R$ in blocks
- $T(R) =$ number of tuples in $R$
- $V(R, a) =$ # of distinct values of attribute $a$

What is the cost in each case?

- Clustered index on $a$:
- Unclustered index on $a$:
Index Based Selection

Selection on equality: $\sigma_{a=v}(R)$
- $B(R) =$ size of $R$ in blocks
- $T(R) =$ number of tuples in $R$
- $V(R, a) =$ # of distinct values of attribute $a$

What is the cost in each case?
- Clustered index on $a$: $B(R)/V(R,a)$
- Unclustered index on $a$: $T(R)/V(R,a)$
Index Based Selection

Selection on equality: $\sigma_{a=v}(R)$
- $B(R) =$ size of $R$ in blocks
- $T(R) =$ number of tuples in $R$
- $V(R, a) =$ # of distinct values of attribute $a$

What is the cost in each case?
- Clustered index on $a$: $B(R)/V(R,a)$
- Unclustered index on $a$: $T(R)/V(R,a)$

Note: we ignore I/O cost for index pages
Index Based Selection

• Example:
  
  \[
  \begin{align*}
  B(R) &= 2000 \\
  T(R) &= 100,000 \\
  V(R, a) &= 20
  \end{align*}
  \]

  cost of \( \sigma_{a=v}(R) = ? \)

• Table scan:

• Index based selection:
Index Based Selection

- Example:
  - Table scan: $B(R) = 2,000$ I/Os
  - Index based selection:
    - $B(R) = 2000$
    - $T(R) = 100,000$
    - $V(R, a) = 20$
    - cost of $\sigma_{a=v}(R) = ?$

- Table scan: $B(R) = 2,000$ I/Os
- Index based selection:
**Index Based Selection**

- **Example:**
  
  \[
  \begin{array}{|c|c|}
  \hline
  B(R) & 2000 \\
  T(R) & 100,000 \\
  V(R, a) & 20 \\
  \hline
  \end{array}
  \]

  \[
  \text{cost of } \sigma_{a=v}(R) = ?
  \]

- **Table scan:** \(B(R) = 2,000\) I/Os

- **Index based selection:**
  - If index is clustered:
  - If index is unclustered:
## Index Based Selection

- **Example:**
  
  \[
  \begin{array}{|c|c|}
  \hline
  B(R) & 2000 \\
  T(R) & 100,000 \\
  V(R, a) & 20 \\
  \hline
  \end{array}
  \]

- Table scan: \( B(R) = 2,000 \) I/Os
- Index based selection:
  - If index is clustered: \( B(R)/V(R,a) = 100 \) I/Os
  - If index is unclustered:

Cost of \( \sigma_{a=v}(R) = ? \)
Index Based Selection

- Example:
  - Table scan: $B(R) = 2000$ I/Os
  - Index based selection:
    - If index is clustered: $B(R)/V(R,a) = 100$ I/Os
    - If index is unclustered: $T(R)/V(R,a) = 5,000$ I/Os

<table>
<thead>
<tr>
<th>$B(R)$</th>
<th>$T(R)$</th>
<th>$V(R, a)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>100,000</td>
<td>20</td>
</tr>
</tbody>
</table>

Cost of $\sigma_{a=v}(R) = ?$
Index Based Selection

• Example:
  - B(R) = 2000
  - T(R) = 100,000
  - V(R, a) = 20

• Table scan: B(R) = 2,000 I/Os

• Index based selection:
  - If index is clustered: B(R)/V(R,a) = 100 I/Os
  - If index is unclustered: T(R)/V(R,a) = 5,000 I/Os

Lesson: Don’t build unclustered indexes when V(R,a) is small!
Index Nested Loop Join

\( R \bowtie S \)

- Assume \( S \) has an index on the join attribute
- Iterate over \( R \), for each tuple fetch corresponding tuple(s) from \( S \)

- **Cost:**
  - If index on \( S \) is clustered: \( B(R) + T(R)B(S)/V(S,a) \)
  - If index on \( S \) is unclustered: \( B(R) + T(R)T(S)/V(S,a) \)