CS4432: Database Systems II

Lecture #16
Join Processing Algorithms

Professor Elke A. Rundensteiner
Join: $R_1 \bowtie R_2, R_2 \bowtie R_1$

- Iteration Join
- Sort-Merge Join
- Index-Join
- Hash Join
Factors that affect performance

(1) Tuples of relation stored physically together?

(2) Buffer-aware data movement

(3) Relations sorted by join attribute?

(3) Indexes exist?
Example  \( R1 \bowtie R2 \) over common attribute \( C \)

\[
\begin{align*}
T(R1) &= 10,000 \\
T(R2) &= 5,000 \\
S(R1) &= S(R2) = 1/10 \text{ block} \\
\text{Memory available} &= 101 \text{ blocks}
\end{align*}
\]

→ Metric: \# of IOs

   (ignoring writing of result)
Join: $R_1 \bowtie R_2, R_2 \bowtie R_1$

- Iteration (nested loops)
- Merge join
- Join with index
- Hash join
• **Iteration join** (nested-loop join)

  for each \( r \in R1 \) do
  for each \( s \in R2 \) do
    if \( r.C = s.C \) then output \( r,s \) pair
Example 1(a) Iteration Join $R_1 \bowtie R_2$

- Relations not contiguous
- Recall $T(R_1) = 10,000 \quad T(R_2) = 5,000$
  
  $S(R_1) = S(R_2) = \frac{1}{10} \text{ block}$
  
  $\text{MEM} = 101 \text{ blocks}$

Cost: for each $R_1$ tuple:

\[ \text{[Read 1 R1 tuple + Read R2]} \]

Total $= 10,000 \times [1 + 5000] = 50,010,000 \text{ IOs}$
• Can we do better?

Use our memory

(1) Read 100 blocks of R1
(2) Read all of R2 (using 1 block) + join
(3) Repeat until done
**Cost:** for each R1 chunk:

- Read chunk: 1000 IOs
- Read R2: 5000 IOs

Total = \[rac{10,000 \times (1000 + 5000)}{1,000}\] = 60,000 IOs
Does Reverse Join Order Matter?

Reverse join order: $R2 \bowtie R1$

Total $= \frac{5000}{1000} \times (1000 + 10,000) = \frac{5 \times 11,000}{1000}$

$5 \times 11,000 = 55,000$ IOs
Example Iteration Join $R2 \bowtie R1$

- Relations contiguous

Cost
For each $R2$ chunk:
  - Read chunk: 100 IOs
  - Read $R1$: 1000 IOs

Total = 5 chunks x 1,100 = 5,500 IOs
So far

- Not contiguous
  - Iterate R2 $\bowtie$ R1 \hspace{1cm} 50,010,000

- Be memory access aware:
  - Iterate R2 $\bowtie$ R1 \hspace{1cm} 55,000

- Contiguous
  - Blocked-access:
    - Iterate R2 $\bowtie$ R1 \hspace{1cm} 5,500
Join:  \( R_1 \bowtie R_2, \ R_2 \bowtie R_1 \)

- Iteration (nested loops)
- Sort-Merge join
- Join with index
- Hash join
Merge join (conceptually)

(1) if R1 and R2 not sorted, sort them
(2) i ← 1; j ← 1;
    While (i ≤ T(R1)) ∧ (j ≤ T(R2)) do
        if R1{ i }.C = R2{ j }.C then outputTuples
        else if R1{ i }.C > R2{ j }.C then j ← j+1
        else if R1{ i }.C < R2{ j }.C then i ← i+1
Procedure Output-Tuples

While (R1{ i }.C = R2{ j }.C) ∧ (i ≤ T(R1)) do
  [jj ← j;
  while (R1{ i }.C = R2{ jj }.C) ∧ (jj ≤ T(R2)) do
    [output pair R1{ i }, R2{ jj };
      jj ← jj+1 ];
  ]

i ← i+1 ]
**Example**

<table>
<thead>
<tr>
<th>i</th>
<th>R1{i}.C</th>
<th>R2{j}.C</th>
<th>j</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td>20</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td>20</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>30</td>
<td>30</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>40</td>
<td>30</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>50</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>52</td>
<td>7</td>
</tr>
</tbody>
</table>
Example: Merge Join

- Both R1, R2 ordered by C; relations contiguous

```
Memory

R1

R2

R1

......

R2

......
```

Total cost: Read R1 cost + read R2 cost

\[= 1000 + 500 = 1,500 \text{ IOs}\]
Example  Merge Join

- R1, R2 not ordered, but contiguous

--> Need to sort R1, R2 first.... HOW?
One way to sort: Merge Sort

(i) For each 100 blk chunk of R:
   - Read chunk
   - Sort in memory
   - Write to disk
(ii) Read all chunks + merge + write out
Cost: Sort

Each tuple is read, written, read, written

so...

Sort cost R1: $4 \times 1,000 = 4,000$

Sort cost R2: $4 \times 500 = 2,000$
Example Merge Join

R1, R2 contiguous, but unordered

Total cost = sort cost + join cost
            \[= 6,000 + 1,500 = 7,500\] IOs
Comparison?

Iteration cost = 5,500 IOs
Merge cost = 7,500 IOs

Conclusion: so merge join does not pay off?

True sometimes? or always ??
Next a case

Where sort-merge join beats the iteration join.
For: $R1 = 10,000$ blocks contiguous
$R2 = 5,000$ blocks not ordered

Iterate: $\frac{5000}{100} \times (100 + 10,000) = 50 \times 10,100$
$= 505,000$ IOs

Merge join: $5(10,000 + 5,000) = 75,000$ IOs

Merge Join (with sort) WINS!