CPSC 421
Database Management Systems

Lecture 14:
Join Algorithms

* Some material adapted from R. Ramakrishnan, L. Delcambre, and B. Ludaescher

Agenda

• Quiz
• Assignment 5 topics
• “Naive” (simple) join algorithm
• Start on alternative algorithms
Join Algorithms

• Consider this query:

```
SELECT *
FROM Reserves R, Sailors S
WHERE R.sid = S.sid
```

• Reserves and Sailors from text:

```
Sailors(sid, snam, rating, age)
Boats(bid, bname, color)
Reserves(sid, bid, day)
```

• \( R \bowtie S \) is very common …
  – \( R \times S \) followed by a selection is inefficient … why?
  – So we process joins (rather than cross product) when possible
  – Much effort in query processing invested in join algorithms
Join Algorithms

• Consider this query:

```
SELECT *
FROM Reserves R, Sailors S
WHERE R.sid = S.sid
```

• Some notation:
  - $M$ pages in R
  - $P_R$ tuples per page
  - $N$ pages in S
  - $P_S$ tuples per page

Join Algorithms

• Simple nested loops join

```
Join on i-th column of R and j-th column of S
1. foreach tuple r in R do
2.   foreach tuple s in S do
3.     if $r_i == s_j$ then add $<r, s>$ to result
```

• For $R \bowtie S$ ...

• We call $R$ the “outer” relation
• We call $S$ the “inner” relation
Join Algorithms

- Simple nested loops join
  Join on i-th column of R and j-th column of S
  1. foreach tuple r in R do
  2.   foreach tuple s in S do
  3.     if \( r_i = s_j \) then add \(<r, s>\) to result

For each tuple in the outer relation R, we scan the entire inner relation S tuple-by-tuple …
- If \( M = 1000 \) pages in R, \( P_R = 100 \) tuples/page
- If \( N = 500 \) Pages in S, \( P_S = 80 \) tuples/page
- If 100 I/Os per second
- Cost \( R \bowtie S = M + (P_R \times M) \times N = 1000 + 100*1000*500 \) I/Os
- 50,001,000 I/Os = 500,010 seconds = 6 days!

Simple Nested Loops Join

- This example highlights
  - Simple nested loop join isn’t very practical
  - We need algorithms that optimize joins

- There are also the other operations to consider …
  - But we’ll start with joins
Simple Nested Loops Join

Table 1 on Disk

\[
\begin{align*}
2, & \ldots \\
6, & \ldots \\
3, & \ldots \\
1, & \ldots \\
5, & \ldots \\
9, & \ldots 
\end{align*}
\]

Memory Buffers:

Table 2 on Disk

\[
\begin{align*}
\ldots, & 2 \\
\ldots, & 7 \\
\ldots, & 6 \\
\ldots, & 9 \\
\ldots, & 1 \\
\ldots, & 5 
\end{align*}
\]

Simple Nested Loops Join

Table 1 on Disk

\[
\begin{align*}
2, & \ldots \\
6, & \ldots \\
3, & \ldots \\
1, & \ldots \\
5, & \ldots \\
9, & \ldots 
\end{align*}
\]

Memory Buffers:

Table 2 on Disk

\[
\begin{align*}
\ldots, & 2 \\
\ldots, & 7 \\
\ldots, & 6 \\
\ldots, & 9 \\
\ldots, & 1 \\
\ldots, & 5 
\end{align*}
\]

- Load 1st page of Table 1 into memory
- Load 1st page of Table 2 into memory
- Start checking join condition (e.g., R.sid = S.sid)
Simple Nested Loops Join

Table 1 on Disk

2, ...
6, ...
3, ...
1, ...
5, ...
9, ...

Memory Buffers:

Table 2 on Disk

..., 2
..., 7
..., 6
..., 9
..., 1
..., 5

Query Answer:

..., 2

MATCH!

Simple Nested Loops Join

Table 1 on Disk

2, ...
6, ...
3, ...
1, ...
5, ...
9, ...

Memory Buffers:

Table 2 on Disk

..., 2
..., 7
..., 6
..., 9
..., 1
..., 5

Query Answer:

..., 2

NO MATCH
Discard!

Query Answer:

2, ..., 2
**Simple Nested Loops Join**

Table 1 on Disk

2, ...
6, ...
3, ...
1, ...
5, ...
9, ...

Memory Buffers:

Table 2 on Disk

..., 2
..., 7
..., 6
..., 9
..., 1
..., 5

Table 2 on Disk

..., 6
..., 9

Query Answer:

2, ...
...
2

• Load 2nd page of Table 2 into memory

---

**Simple Nested Loops Join**

Table 1 on Disk

2, ...
6, ...
3, ...
1, ...
5, ...
9, ...

Memory Buffers:

NO MATCH
Discard!

Query Answer:

2, ...
...
2

Table 2 on Disk

..., 2
..., 7
..., 6
..., 9
..., 1
..., 5
Simple Nested Loops Join

Table 1 on Disk

2, ...
6, ...
3, ...

1, ...
5, ...
9, ...

Memory Buffers:

2, ...
6, ...
3, ...

...6
...9

Table 2 on Disk

...2
...7

...6
...9

...1
...5

Table 1 on Disk

2, ...
6, ...
3, ...

1, ...
5, ...
9, ...

Memory Buffers:

2, ...
6, ...
3, ...

...1
...5

Table 2 on Disk

...2
...7

...6
...9

...1
...5

Query Answer:

2, ...
...2

• Load 3rd page of Table 2 into memory

NO MATCH
Discard!
Simple Nested Loops Join

Table 1 on Disk

\[
\begin{align*}
&2, \\
&6, \\
&3, \\
&1, \\
&5, \\
&9,
\end{align*}
\]

Memory Buffers:

\[
\begin{align*}
&2, \\
&6, \\
&3, \\
&\ldots, 2 \\
&\ldots, 7 \\
&\ldots, 6 \\
&\ldots, 9 \\
&\ldots, 1 \\
&\ldots, 5
\end{align*}
\]

Table 2 on Disk

Query Answer:

\[
2, \ldots, 2
\]

NO MATCH
Discard!

Simple Nested Loops Join

Table 1 on Disk

\[
\begin{align*}
&2, \\
&6, \\
&3, \\
&1, \\
&5, \\
&9,
\end{align*}
\]

Memory Buffers:

\[
\begin{align*}
&2, \\
&6, \\
&3, \\
&\ldots, 2 \\
&\ldots, 1 \\
&\ldots, 5
\end{align*}
\]

Table 2 on Disk

Query Answer:

\[
2, \ldots, 2
\]

NO MATCH
Discard!
Simple Nested Loops Join

Table 1 on Disk

| 2, ... |
| 6, ... |
| 3, ... |
| 1, ... |
| 5, ... |
| 9, ... |

Memory Buffers:

Table 2 on Disk

| ..., 2 |
| ..., 7 |
| ..., 6 |
| ..., 9 |
| ..., 1 |
| ..., 5 |

Query Answer:

- Go to next tuple in 1st page of Table 1
- (Re)Load 1st page of Table 2 into memory

Simple Nested Loops Join

Table 1 on Disk

| 2, ... |
| 6, ... |
| 3, ... |
| 1, ... |
| 5, ... |
| 9, ... |

Memory Buffers:

Table 2 on Disk

| ..., 2 |
| ..., 7 |
| ..., 6 |
| ..., 9 |
| ..., 1 |
| ..., 5 |

Query Answer:

- NO MATCH
- Discard!

Query Answer:

| 2, ... |
| ..., 2 |
Simple Nested Loops Join

Table 1 on Disk

\[
\begin{align*}
2, & \ldots \\
6, & \ldots \\
3, & \ldots \\
1, & \ldots \\
5, & \ldots \\
9, & \ldots
\end{align*}
\]

Memory Buffers:

\[
\begin{align*}
2, & \ldots \\
6, & \ldots \\
3, & \ldots \\
& \ldots
\end{align*}
\]

Table 2 on Disk

\[
\begin{align*}
& \ldots \\
& \ldots \\
& \ldots
\end{align*}
\]

\[
\begin{align*}
& \ldots \\
& \ldots \\
& \ldots
\end{align*}
\]

\[
\begin{align*}
& \ldots \\
& \ldots \\
& \ldots
\end{align*}
\]

\[
\begin{align*}
& \ldots \\
& \ldots \\
& \ldots
\end{align*}
\]

\[
\begin{align*}
& \ldots \\
& \ldots \\
& \ldots
\end{align*}
\]

NO MATCH
Discard!

Query Answer:

\[
2, \ldots \ldots, 2
\]

Simple Nested Loops Join

Table 1 on Disk

\[
\begin{align*}
2, & \ldots \\
6, & \ldots \\
3, & \ldots \\
1, & \ldots \\
5, & \ldots \\
9, & \ldots
\end{align*}
\]

Memory Buffers:

\[
\begin{align*}
2, & \ldots \\
6, & \ldots \\
3, & \ldots \\
& \ldots
\end{align*}
\]

Table 2 on Disk

\[
\begin{align*}
& \ldots \\
& \ldots \\
& \ldots
\end{align*}
\]

\[
\begin{align*}
& \ldots \\
& \ldots \\
& \ldots
\end{align*}
\]

\[
\begin{align*}
& \ldots \\
& \ldots \\
& \ldots
\end{align*}
\]

\[
\begin{align*}
& \ldots \\
& \ldots \\
& \ldots
\end{align*}
\]

Query Answer:

\[
2, \ldots \ldots, 2
\]

• (Re)Load 2nd page of Table 2 into memory
Simple Nested Loops Join

Table 1 on Disk:

```
2, ...
6, ...
3, ...
1, ...
5, ...
9, ...
```

Table 2 on Disk:

```
..., 2
..., 7
..., 6
..., 9
..., 1
..., 5
```

Query Answer:

```
2, ..., 2
6, ..., 6
```

And so forth…

- Does this algorithm work for \( R.sid < S.sid \)?
- Does it work for cross product?
**Cost of Simple Nested Loops Join (Revisited)**

- Simple nested loops join
  - Join on i-th column of R and j-th column of S
  1. foreach tuple r in R do
  2.    foreach tuple s in S do
  3.       if \( r_i = s_j \) then add <r, s> to result

For each tuple in the outer relation R, we scan the entire inner relation S tuple-by-tuple ...

- If \( M = 1000 \) pages in R, \( P_R = 100 \) tuples/page
- If \( N = 500 \) Pages in S, \( P_S = 80 \) tuples/page
- If 100 I/Os per second
- Cost \( R \bowtie S = M + (P_R \times M) \times N = 1000 + 100 \times 1000 \times 500 \) I/Os
- 50,001,000 I/Os \( \approx 500,010 \) seconds \( \approx 6 \) days!

---

**An Alternative Algorithm**

- "Page-oriented" nested loops join
  - Join on i-th column of R and j-th column of S
  1. foreach page of tuples in R do
  2.    foreach page of tuples in S do
  3.       foreach record r and s in memory
  4.         if \( r_i = s_j \) then add <r, s> to result

For each page in R, get each page in S ...

- If \( M = 1000 \) pages in R, \( N = 500 \) Pages in S, and 100 I/Os per sec.
- Cost \( R \bowtie S = M + M \times N = 1000 + 100 \times 500 = 501,000 \) I/Os
- Cost \( S \bowtie R = N + N \times M = 500 + 500 \times 1000 = 500,500 \) I/Os
- Thus, we typically use smaller relation as outer relation
- 500,500 I/Os \( = 1.4 \) hours
Page-Oriented Nested Loops Join

Table 1 on Disk

\[
\begin{array}{c}
2, \ldots \\
6, \ldots \\
3, \ldots \\
1, \ldots \\
5, \ldots \\
9, \ldots \\
\end{array}
\]

Table 2 on Disk

\[
\begin{array}{c}
\ldots, 2 \\
\ldots, 7 \\
\ldots, 6 \\
\ldots, 9 \\
\ldots, 1 \\
\ldots, 5 \\
\end{array}
\]

Memory Buffers:

- Load 1st page of Table 1 into memory
- Load 1st page of Table 2 into memory
- Check every combination of records in buffers
Page-Oriented Nested Loops Join

Table 1 on Disk

2, ...
6, ...
3, ...

1, ...
5, ...
9, ...

Memory Buffers:

Table 2 on Disk

..., 2
..., 7

..., 6
..., 9

..., 1
..., 5

• Do the same thing for 2nd page in Table 2

Page-Oriented Nested Loops Join

Table 1 on Disk

2, ...
6, ...
3, ...

1, ...
5, ...
9, ...

Memory Buffers:

Table 2 on Disk

..., 2
..., 7

..., 6
..., 9

..., 1
..., 5

• Do the same thing for 3rd page in Table 2
Page-Oriented Nested Loops Join

Table 1 on Disk

\[ 2, \ldots, 6, \ldots, 3, \ldots, 1, \ldots, 5, \ldots, 9, \ldots \] 

Table 2 on Disk

\[ \ldots, 2, \ldots, 7, \ldots, 6, \ldots, 9, \ldots, 1, \ldots, 5, \ldots \] 

Memory Buffers:

- Repeat for the 2nd page of Table 1
- And so on

Another Alternative Algorithm: Use Buffer

- “Block” nested loops join
  
  Join on i-th column of R and j-th column of S
  
  1. Assume B pages of memory in buffer
  2. Assign one page of memory in buffer to output
  3. Load B-2 pages of tuples from R
  4. Load 1 page of tuples from S
  5. foreach record \( r \) and \( s \) in memory
  6. if \( r_i = s_j \) then add \( <r, s> \) to result

For multiple pages in R, get each page in S … check all pairs and output
- If \( M = 1000 \) pages in R, \( N = 500 \) Pages in S, \( B = 35 \), and 100 I/Os per sec.
- Cost \( R \bowtie S = M + (M / (B - 2)) \cdot N = 1000 + (1000 / 33) \cdot 500 \approx 16,000 \) I/Os
- Cost \( S \bowtie R = N + N \cdot M = 500 + (500 / 33) \cdot 1000 \approx 15,500 \) I/Os
- 15,500 I/Os \( \approx 3 \) minutes
Block Nested Loops Join

Table 1 on Disk

2, ...
6, ...
3, ...

1, ...
5, ...
9, ...

Memory Buffers:

2, ...
6, ...
3, ...

1, ...
5, ...
9, ...

Table 2 on Disk

..., 2
..., 7

..., 6
..., 9

..., 1
..., 5

• Load 1st page of Table 2 into Memory
• Load as many pages of Table 1 into memory as possible
• Check every combination of records in buffers

Yet Another Alternative Algorithm: Use Index

• “Index” nested loops join
  
  Join on i-th column of R and j-th column of S
  1. Assuming there is an index on the j-th column of S
  2. foreach tuple r in R do
  3.   find tuples s in S with matching search key r_i
  4.   for each such s, add <r, s> to result

For records in R, use search key to obtain matching S records
  
  - If M = 1000 pages in R, P_R = 100 tuples/page, and 100 I/Os per sec.
  - Cost R $\bowtie$ S = M + (M*P_R) * cost of finding matching S tuples
    = 1000 + (1000*100) * 3 ≈ 300,100 I/Os ≈ 1 hour
  - Cost S $\bowtie$ R = 500 + (500*80)*4 ≈ 160,500 I/Os ≈ 30 minutes
  - If probing R is 2 I/Os, then ≈ 15 minutes
Index Nested Loops Join

Table 1 on Disk

2, ...
6, ...
3, ...

1, ...
5, ...
9, ...

Memory Buffers:

2, ...
6, ...
3, ...

Table 2 on Disk

..., 2
..., 7

..., 6
..., 9

..., 1
..., 5

Table 2 Index

• Load 1st page of Table 1 into memory
• For each tuple in page, probe index of Table 2
• Output resulting tuples

And Another Alternative Algorithm: Sort

• If each relation is sorted on the join attributes …

• Cost of joining R and S can be reduced to $M + N$ !!!

• Compare 1st in R and 1st in S
• If match output $r, s$
• Otherwise discard smallest and repeat

• But what if R and S are not sorted?
  – We need to sort them
  – The Challenge: The tables do not fit into memory!
  – The Solution: External Sorting (… more later)
  – Note that other relational operator algorithms also require sorting
Comparison of (approximate) costs

<table>
<thead>
<tr>
<th>Join Algorithm</th>
<th>I/Os</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple Nested Loops Join</td>
<td>50,000,000</td>
<td>6 days</td>
</tr>
<tr>
<td>Page Nested Loops Join</td>
<td>500,000</td>
<td>1.4 hours</td>
</tr>
<tr>
<td>Block Nested Loops Join</td>
<td>16,000</td>
<td>3 minutes</td>
</tr>
<tr>
<td>Index Nested Loops Join</td>
<td>160,500</td>
<td>30 minutes</td>
</tr>
<tr>
<td>Sort-Merge Join</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hash Join</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Assuming:
- R has 1000 pages, 100 tuples/page
- S has 500 pages, 80 tuples/page
- 35 buffer pages
- 100 I/Os per second