CPSC 421
Database Management Systems

Lecture 15:
More on Join Algorithms

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

Agenda

• Quiz
• Alternative join algorithms
  – Sort-merge join + exercise
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)

• Assumptions:
  – $M = 1000$ pages in R
  – $P_R = 100$ tuples per page
  – $N = 500$ pages in S
  – $P_S = 80$ tuples per page
  – 100 I/Os per second
  – $B = 35$ available buffer pages
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

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
N-Way External Sorting

• Employ the “merge” step in the mergesort algorithm

• On the first pass:
  – Read pages of file until memory (buffers) full
  – Sort data in buffer pages on (search/sort) key
  – Write result back out to disk

  Result is a “sorted run” …
  A sorted run consists of a (sub-) set of small sorted files

N-Way External Sorting

• Employ the “merge” step in the mergesort algorithm

• Once we have a “sorted run”
  – Do an “N-way” merge
    … rather than a 2-way merge as in mergesort
  – $N = B - 1$ is the number of available buffers
  – One buffer reserved for output

• Results in a set of additional passes
• In each pass we create larger sorted sub-files
Example: 4 buffer pages

First Pass:
- load B = 4 pages, sort, and store as sorted sub-files

New File of 3 sorted sub-files

Example: 4 buffer pages

First Merge Pass:
- Merge B – 1 of the sorted sub files (sorted runs)
Example: 4 buffer pages

First Merge Pass:
– Merge B – 1 of the sorted sub files (sorted runs)

In this case 3-way merge
Example: 4 buffer pages

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>13</td>
<td>14</td>
<td>15</td>
</tr>
<tr>
<td>16</td>
<td>17</td>
<td>18</td>
</tr>
<tr>
<td>19</td>
<td>20</td>
<td>21</td>
</tr>
<tr>
<td>22</td>
<td>23</td>
<td>24</td>
</tr>
<tr>
<td>25</td>
<td>26</td>
<td>27</td>
</tr>
<tr>
<td>28</td>
<td>29</td>
<td>30</td>
</tr>
</tbody>
</table>

First Merge Pass:
- Merge B – 1 of the sorted sub files (sorted runs)

In this case 3-way merge

Example: 4 buffer pages

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td>13</td>
<td>14</td>
<td>15</td>
</tr>
<tr>
<td>16</td>
<td>17</td>
<td>18</td>
</tr>
<tr>
<td>19</td>
<td>20</td>
<td>21</td>
</tr>
<tr>
<td>22</td>
<td>23</td>
<td>24</td>
</tr>
<tr>
<td>25</td>
<td>26</td>
<td>27</td>
</tr>
<tr>
<td>28</td>
<td>29</td>
<td>30</td>
</tr>
</tbody>
</table>

First Merge Pass:
- Merge B – 1 of the sorted sub files (sorted runs)

In this case 3-way merge
Example: 4 buffer pages

First Merge Pass:
- Merge B – 1 of the sorted sub files (sorted runs)

In this case 3-way merge

Example: 4 buffer pages

First Merge Pass:
- Merge B – 1 of the sorted sub files (sorted runs)
Example: 4 buffer pages

First Merge Pass:
– Merge B – 1 of the sorted sub files (sorted runs)

In this case 3-way merge

Example: 4 buffer pages

First Merge Pass:
– Merge B – 1 of the sorted sub files (sorted runs)
Example: 4 buffer pages

First Merge Pass:
- Merge B – 1 of the sorted sub files (sorted runs)

In this case 3-way merge

Output 4th Page

Example: 4 buffer pages

First Merge Pass:
- Merge B – 1 of the sorted sub files (sorted runs)
Example: 4 buffer pages

First Merge Pass:
- Merge B - 1 of the sorted sub files (sorted runs)

In this case 3-way merge

Example: 4 buffer pages

First Merge Pass:
- Merge B - 1 of the sorted sub files (sorted runs)
### Example: 4 buffer pages

<table>
<thead>
<tr>
<th>1</th>
<th>6</th>
<th>13</th>
<th>20</th>
<th>4</th>
<th>9</th>
<th>16</th>
<th>24</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>10</td>
<td>14</td>
<td>21</td>
<td>5</td>
<td>12</td>
<td>17</td>
<td>26</td>
<td>18</td>
</tr>
<tr>
<td>3</td>
<td>11</td>
<td>19</td>
<td>25</td>
<td>7</td>
<td>15</td>
<td>22</td>
<td>27</td>
<td>23</td>
</tr>
</tbody>
</table>

**First Merge Pass:**

- Merge B – 1 of the sorted sub files (sorted runs)

---

<table>
<thead>
<tr>
<th>1</th>
<th>4</th>
<th>7</th>
<th>10</th>
<th>13</th>
<th>16</th>
<th>19</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>5</td>
<td>8</td>
<td>11</td>
<td>14</td>
<td>17</td>
<td>18</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>9</td>
<td>12</td>
<td>15</td>
<td>18</td>
<td></td>
</tr>
</tbody>
</table>

**First Merge Pass:**

- Merge B – 1 of the sorted sub files (sorted runs)
Example: 4 buffer pages

First Merge Pass:
- Merge B – 1 of the sorted sub files (sorted runs)

Output 7th Buffer

In this case 3-way merge

Example: 4 buffer pages

First Merge Pass:
- Merge B – 1 of the sorted sub files (sorted runs)
Example: 4 buffer pages

First Merge Pass:
– Merge B – 1 of the sorted sub files (sorted runs)

In this case 3-way merge

Output 8\textsuperscript{th} Buffer

Example: 4 buffer pages

First Merge Pass:
– Merge B – 1 of the sorted sub files (sorted runs)
Example: 4 buffer pages

First Merge Pass:
- Merge B – 1 of the sorted sub files (sorted runs)

N-Way External Sorting

- Merge may require multiple passes
- At each merge pass the number of sub-files is reduced by B – 1
N-Way External Sorting

- The cost:
  - Each pass does $2^*M$ I/Os (for $M$ pages in table)
  - We read and write the entire file (all pages) in each pass
  - So … how many passes?
- Number of passes depends on buffer space available
  - Passes $= \lceil \log_{B^{-1}} (M/B) \rceil$  ... Why $M/B$?
  - Can sort 100 million pages in 4 passes w/ 129 pages of memory
  - Can sort $M$ pages using $B$ memory pages in 2 passes if $\sqrt{M} < B$ (often true)

Sort-Merge Join

- Sort R on join attribute (if not already sorted)
- Sort S on join attribute (if not already sorted)
- Merge R and S
  - Scan of R until R-tuple $\geq$ current S-tuple
  - Then can S until S-tuple $\geq$ R-tuple
  - Repeat until R-tuple = S-tuple
  - At this point, we have a match, and output
  - Then resume scanning R and S
Sort-Merge Join

- Outer relation R is scanned once
  - Each time an R-tuple r matches first S-tuple
  - We form a “group” of S-tuples that match r
  - Each such group is scanned once per matching R tuple
  - Either:
    - This group fits into memory (and the scan is “free”)
    - Or we have extra page I/Os (to reread the group)

Sort-Merge Join

- Best case cost (all matches in memory):
  - Cost to sort R + Cost to sort S + (M+N)

- Worst case cost (all R and S have same value)
  - Matching group is the entire S relation
  - Cost to Sort R + Cost to sort S + M + M*N

- … note this is worse than page-oriented nested loops!
  (since you also have to sort R and S)
Sort-Merge Join

• For Reserves and Sailors:
  – Reserves has 1000 pages
  – Sailors has 500 pages
  – With 35 pages in the buffer, each sorted in 2 passes

• Best case cost is:
  – 4*1000 + 4*500 + 1000 + 500 = 7500 I/Os ≈ 1 minute

… multiply by 4 since it takes 2 passes and each pass reads and writes each page of file