CS522 Advanced Database Systems
Query Processing

Chengyu Sun
California State University, Los Angeles

SQL Query
<>SFW< syntax
SELECT <SelList> FROM <FromList> WHERE <Condition>

SQL Query Example

<table>
<thead>
<tr>
<th>R</th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>1</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>1</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>2</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>d</td>
<td>2</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>e</td>
<td>3</td>
<td>15</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>S</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>x</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>y</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>x</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>y</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

select B, D from R, S
where R.A = c and S.E = 2 and R.C = S.C;

From Query to Parse Tree

<table>
<thead>
<tr>
<th>R.B</th>
<th>R.A</th>
<th>S.D</th>
<th>S.E</th>
<th>R.C</th>
<th>S.C</th>
</tr>
</thead>
</table>

Relational Algebra Notations

<table>
<thead>
<tr>
<th>Operation</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNION</td>
<td>UNION</td>
</tr>
<tr>
<td>INTERSECTION</td>
<td>INTERSECTION</td>
</tr>
<tr>
<td>DIFFERENCE</td>
<td>DIFFERENCE</td>
</tr>
<tr>
<td>SELECT</td>
<td>SELECT</td>
</tr>
<tr>
<td>PROL</td>
<td>PROL</td>
</tr>
<tr>
<td>OUTERJOIN</td>
<td>OUTERJOIN</td>
</tr>
<tr>
<td>LEFTJOIN</td>
<td>LEFTJOIN</td>
</tr>
<tr>
<td>RIGHTJOIN</td>
<td>RIGHTJOIN</td>
</tr>
<tr>
<td>RENAME</td>
<td>RENAME</td>
</tr>
</tbody>
</table>

Review Chapter 5
Sets and Bags

- Bag (multi-set) semantics
  - Allow duplicates
  - default for SQL
- Set semantics
  - No duplicates
  - More expensive to implement in general

From Parse Tree to Logical Query Plan

- With no subquery
  - Start with a product of all relations in the <FromList>
  - Add \( \sigma \) for each condition C
  - Add \( \pi \) for all attributes in the <SelList> L
- With subquery
  - see Query Optimization

From LQP to Best LQP

- Plan rewriting
  - Algebraic laws
  - Heuristics
- Choose the best plan
  - Cost estimation – result size

From Logical Query Plan to Physical Query Plan

- Plan enumeration
- Choose the right physical operator
  - Estimate algorithmic cost
- Other considerations

Cost Model for Physical Operators

- B data pages
- M buffer pages
- T tuples
- \( V(a) \) or \( V(a_1, a_2, \ldots, a_n) \) distinct values

- Simplifications
  - In-memory operations are free
  - Computation and storage
  - Output cost does not count

Algorithms for Physical Operators

- One-pass algorithms
- Two-pass algorithms
- Multi-pass algorithms
- Index-based algorithms
One-pass Algorithms

- Simple
- Efficient
- May depends on the size of memory buffer
  - e.g. apply only to relations with $B < M$
  - Potential performance degradation if $M$ is not estimated correctly

Selection and Projection

- One tuple at a time
- Restriction on relation size: none

Duplicate Elimination and Grouping

- Duplicate elimination
  - In-memory data structure?
  - Relation size?

- Grouping
  - MIN, MAX, COUNT, SUM, AVG
  - Relation size?

Union, Intersection, and Difference

- with Bag semantics
  - Union: trivial
  - Intersection: $\min(|R|,|S|) < M$
  - Difference: ??

- with Set semantics
  - Union: ??
  - Intersection: ??
  - Difference: ??

Product and Join

- Product
- Nested loop join
  - One relation fits in memory buffer
  - Neither relations fits in memory buffer – one and a half pass

Summary of One-pass Algorithms

<table>
<thead>
<tr>
<th>Operators</th>
<th>M Required</th>
<th>Disk I/O</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sigma_{p}$</td>
<td>1</td>
<td>B</td>
</tr>
<tr>
<td>$\pi_{q}$</td>
<td>B</td>
<td>B</td>
</tr>
<tr>
<td>$\cup_{x} \cap_{y}$</td>
<td>$\min(B(R),B(S))$</td>
<td>$B(R)+B(S)$</td>
</tr>
<tr>
<td>$\times_{x} \bowtie_{y}$</td>
<td>$\min(B(R),B(S))$</td>
<td>$B(R)+B(S)$</td>
</tr>
<tr>
<td>$\times_{x} \bowtie_{y}$</td>
<td>$M \geq 2$</td>
<td>$B(R)B(S)/M$</td>
</tr>
</tbody>
</table>
Two-pass Algorithms
- Can handle very large relations
- Sort-based algorithms
- Hash-based algorithms

Two Phase Multiway Merge Sort (TPMMS)
- Phase 1: fill memory buffer, in-memory sort
- Phase 2: merge sorted sublists, one block from each

TPMMS Merge Phase
- input buffer, one for each sorted sublist
- output buffer
- M Required
- Disk I/O

Duplicate Elimination and Grouping
- Example
  - R: \{2, 5, 2, 1, 2, 2, 4, 5, 4, 3, 4, 2, 1, 5, 2, 1, 3\}
  - 2 tuples per page
  - 3 buffer pages

Union, Intersection, and Difference
- Example
  - R: \{2, 5, 2, 1, 2, 2, 4, 5, 4, 3, 4, 2\}
  - S: \{1, 5, 2, 1, 3\}
  - 2 tuples per page
  - 3 buffer pages
  - Bag or Set??

Simple Sort Join
- Example
  - Sort R, sort S, output matching tuples
  - Example
    - B(R) = 1000
    - B(S) = 500
    - M = 101
    - vs. Nested Loop Join??
Sort Merge Join

- Join in the merge phase
- vs. Simple Sort Join??

Summary of Two-pass Algorithms (Sorting)

<table>
<thead>
<tr>
<th>Operators</th>
<th>M Required</th>
<th>Disk I/O</th>
</tr>
</thead>
<tbody>
<tr>
<td>γ, δ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>∪, ∩, −</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&gt;= (simple sort)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&gt;= (merge sort)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Hashing

Hash-based Algorithms

- Duplicate eliminations
- Grouping and aggregation
- Union, intersection, and difference
- Join

Hybrid Hash Join

Hybrid Hash Join Parameters

- The smaller k, the better
- k cannot be too small

- one buffer page for a "small" bucket
- multiple buffer pages for a "big" bucket s.t. all tuples in the bucket can be kept in memory
Summary of Two-pass Algorithms (Hashing)

<table>
<thead>
<tr>
<th>Operators</th>
<th>M Required</th>
<th>Disk I/O</th>
</tr>
</thead>
<tbody>
<tr>
<td>γ, δ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>∪, ∩, −</td>
<td></td>
<td></td>
</tr>
<tr>
<td>≥ ≤ (hash)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>≥ ≤ (hybrid hash)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Sorting vs. Hashing

- Similar complexity and limitations
- Hashing looks better on paper
  - What’s the catch??
- Sorting allows more optimization

Multi-pass Algorithms

- kPMMS
  - What’s the largest relation we can sort in k passes??
  - What’s the I/O complexity of kPMMS??

Index-based Algorithms

- Clustered index vs. Un-clustered index

Indexed Join

- Indexed Nested Loop Join
- Zig-zag Join

Reading

- Stanford book: Chapter 15
- [Join survey paper]