Oracle Query Processing and Optimization for OLTP and OLAP

Håkan Jakobsson
Outline

• Some basic topics
  • Query transformations
  • Search space
  • Uncertainty

• Examples of “advanced” issues
  • Plan stability
  • Ease of use
  • SQL tune
  • Optimization criteria
  • Computer architecture evolution
Query Transformations

- Query optimization is a mixture of techniques at different levels
  - Query transformations easier early
  - Access path selection easier later

- Query transformations convert SQL statements into semantically equivalent statements for which access path selection is easier
  - Example: Subquery-to-join transformations

- Oracle includes query transformations as part of the cost-based search space
  - Mainly software-engineering challenges
Search Space

- Large space of possible access paths, join methods and orderings for large queries
- Global effects of local decisions
  - Choice of join method in the middle of join order may affect whether you can eliminate an ORDER BY sort after all tables have been joined
  - Hence, just picking the locally least expensive join method may not give the lowest global cost
- Bushy join trees
  - Bushy trees can be optimal, but considering them increases the search space and complexity of the optimizer compared to considering only left-deep trees
- Substantial research results in this area
Search Space in Oracle

- Left-deep trees unless query inherently bushy or plan has hash joins
- Exhaustive search of join orderings for small joins
- Initial ordering heuristics and cut-off based on best plan so far
  - Does a very good job of keeping search space problems in check
- Avoid Cartesian products for large joins
- Separate passes for global effects like row ordering for ORDER BY
- Query transformations and materialized view rewrites
Uncertainty about Cardinality

- Problem of computing accurate estimates for intermediate result sets
- Can lead to bad join orders, bad join methods, bad access paths
- Information about tables based on statistics (Oracle uses histograms for column statistics)
  - Correlation between columns unknown
    
    ```sql
    SELECT * FROM EMP WHERE TITLE = 'MANAGER' AND SAL < 40000
    ```
  - Hard to know properties of join results => huge uncertainty for large joins
  - Difficult predicates
    
    ```sql
    SELECT * FROM EMP WHERE ENAME LIKE '%SMITH%'
    ```
  - Bind variables
    
    ```sql
    SELECT * FROM EMP WHERE SAL < :1
    ```
Relevant Oracle Features

- Bind peeking for bind variables
- Dynamic sampling for correlation and difficult predicates
  - Sample a small subset of the data and extrapolate properties
- SQL tune (more later)
Motivating Example: Oracle E-Business Suite

<table>
<thead>
<tr>
<th>Object Type</th>
<th>Count:</th>
</tr>
</thead>
<tbody>
<tr>
<td>MATERIALIZED VIEW</td>
<td>402</td>
</tr>
<tr>
<td>TYPE</td>
<td>584</td>
</tr>
<tr>
<td>JAVA CLASS</td>
<td>891</td>
</tr>
<tr>
<td>TABLE</td>
<td>21,980</td>
</tr>
<tr>
<td>INDEX</td>
<td>40,078</td>
</tr>
<tr>
<td>TRIGGER</td>
<td>3,343</td>
</tr>
<tr>
<td>VIEW</td>
<td>28,281</td>
</tr>
<tr>
<td>PACKAGE BODY</td>
<td>40,950</td>
</tr>
<tr>
<td>PACKAGE</td>
<td>41,939</td>
</tr>
</tbody>
</table>

**Queries** 515,000

Average table references per query: 4
Max table references per query: 232
Plan Stability

• Problem of improving the optimizer without risking that some queries deteriorate
• Customers don’t want something that works to be broken in a new release
• One query that deteriorates may outweigh improvements in all other queries
• Solutions:
  • Store plans (in “query outlines”)
  • Versioning of optimizer behavior
Ease of Use

- Obviously a good thing
- Problem: Features and tuning knobs geared towards power users are not always trivial to reconcile with ease of use
- Need to automate difficult tasks flawlessly
- Major industry push towards manageability, self tuning, etc.
- Major theme for Oracle10g
- Example: Statistics collection
  - Automated collection
  - Automated object selection based on degree of change
  - Automated sample size
  - Automated histogram determination based on usage and data skew
Automatic Tuning Optimizer

- New feature in Oracle10g
- Intended for recurrent workloads
- Makes it easy to tune high-load SQL
- Based on the notion of correcting optimizer cardinalities
- Spends time finding actual cardinalities using dynamic sampling and partial evaluation of the query
- Store correction factors for the optimizer to use when query is optimized in the future
- Allows queries to be tuned without changing the text of the query
- Also gives advice about index creation, statistics gathering, query restructuring, etc.
Optimization Criteria

- What are we optimizing for, throughput or response time?
- Problem very apparent for parallelism
- Parallelizing a query is good for response time but doesn’t use less resources
- How do you determine the right degree of parallelism when many users?
  - Adaptive degree of parallelism in Oracle
  - Also, adaptive memory management
- More than an optimizer problem -- Need a model for allocating resources to queries that may include tools for DBA, education of end users, etc.
Partitioning

- Oracle supports range, hash, and list and certain combinations
- Benefits manageability
- Benefits availability -- faster backup and recovery
- Benefits performance
  - Partitionwise join
  - Partition pruning
  - All partition maintenance operations
Computer Architecture Evolution

• Changing relationship between speed of disk and memory access, caching considerations, larger memory sizes, multiple cores, etc.

• Affects fundamental query processing algorithms

• Affects the optimizer’s cost model

• Affects the relative cost of different components of large database systems