

ORACLE®

**Issues in Real-World Query Optimization and
Processing**

Håkan Jakobsson

October 30, 2002

Outline

- **Oracle and query processing**
- **Issues in query optimization**
- **Summary**

Oracle and Query Processing

- **Many new features in Oracle9i**
 - Release 2 is current release
- **Large set of processing techniques in one engine**
 - Many different join and index methods, parallel query, etc. E.g., nested loops, sort-merge, and hash joins, anti-joins, semi-joins, B-tree, bitmap, reverse, functional, and domain indexes, clusters and hash clusters, index-organized tables, nested tables, materialized views, partitioning, join indexes, index joins, index skip-scan, etc.
 - No single technique is best for everything. Oracle provides many different ones, and the optimizer determines which is the best for each individual query.
- **A single general-purpose engine that can do everything gives leverage of the same features for different applications, but makes query optimization more complex**

Query Processing in Oracle

- **Parsing**
- **Query transformations**
- **Optimization of query blocks**
 - Cost-based optimizer
 - Rule-based optimizer uses heuristic rules
- **Repeat transformations and query block optimization for certain types of transformations and rewrites**
- **Generation of execution tree of “row sources”**
- **Execution**

Issues in Query Optimization

- **SQL is a big language**
- **Mixture of techniques at different levels**
- **Size of search space**
- **Uncertainty**
- **Plan stability**
- **Ease of use**
- **Optimization criteria**

SQL Is a Big Language

- **Many language constructs in basic language**
- **Plus vendor extensions, stored procedures, etc.**
- **An RDBMS is very complex and many little things may affect optimization**
 - **Example: Multi-byte character sets and index use**
- **These factors contribute to the complexity of the optimizer**

Mixture of Techniques at Different Levels

- **Query transformations easier early**
- **Access path selection easier later**
- **Tricky to do it all based on cost**
 - **Mainly software-engineering problems**
- **Many special-case optimizations possible**
 - **Some come from really stupid queries generated by tools**

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 heuristic and cut-off based on best plan so far**
- **Avoid Cartesian products for large joins**
- **Separate passes for global effects like row ordering for ORDER BY**
- **Query transformations and materialized view rewrites**

Uncertainty

- **Problem of computing accurate cost estimates**
- **Cost of execution may depend on run-time environment, e.g., how much caching**
- **Information about tables based on statistics (Oracle uses histograms for column statistics)**
 - **Correlation between columns unknown**

```
SELECT * FROM EMP WHERE TITLE = 'MANAGER' AND SAL < 40000
```
 - **Hard to know properties of join results => huge uncertainty for large joins**
 - **Difficult predicates**

```
SELECT * FROM EMP WHERE ENAME LIKE '%SMITH%'
```
 - **Bind variables**

```
SELECT * FROM EMP WHERE SAL < :1
```
- **Remedies in Oracle include**
 - **Bind peeking for bind values**
 - **Dynamic sampling for correlation and difficult predicates**

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**
- **Centralized vs. distributed databases**

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.**

Summary

- **Optimizer designers have to combat problems in areas such as traversal of large search spaces, uncertainty about validity of cost estimates, and various software engineering problems, plus pay a lot of attention to detail**
- **Oracle believes in having a large collection of techniques for query processing since no single technique is best for everything**
- **Hence, importance of the optimizer increases as new techniques are added**