SQL Server Query Optimizer
&
(Some Recent Research on Query Optimization)

Ravi Ramamurthy

DMX Group
Microsoft Research
Outline

• **SQL Server optimizer**
  - Enumeration architecture
  - Search space: flexibility/extensibility
  - Cost and statistics

• Critique of modern optimizers

• Empirical analysis of optimizers

• Conclusion
Running Example (TPC-H Database)
Sample query: Obtain information about certain ordered lineitems that are filtered by suppliers and parts.

```sql
SELECT l_orderkey, l_linenumber, o_orderstatus
FROM lineitem JOIN orders ON l_orderkey = o_orderkey
WHERE l_suppkey < 2000 AND l_partkey < 2000
```
Based on Cascades Framework

- Transformation-based, top-down approach
- Optimization = Tasks + Memo
  ( Programs = Algorithms + Data Structures )

Fully cost-based

Flexible and Extensible

- Search space easy to change
- New operators and rules easy to add
Operators and Operator Trees

- Logical: Specify data transformations without algorithms.
- Physical: Algorithms that implement relational queries.

```
Project (lineitem.l_orderkey, ...)
Select
  Join
    Get (lineitem)
    Get (orders)
    Const (true)
Logical (and)
  Comp (=)
    Identifier (l_orderkey)
    Identifier (o_orderkey)
  Comp (<)
    Identifier (l_suppkey)
    Const (2000)
  Comp (<)
    Identifier (l_partkey)
    Const (2000)
```

```
MergeJoin
  Range (orders_0) ASC
  Sort (l_orderkey)
  Apply lookup (lineitem_0)
  HashJoin
    Range (lineitem_2)
    Comp (<)
      Identifier (l_partkey)
      Const (2000)
    Range (lineitem)
    Comp (<)
      Identifier (l_suppkey)
      Const (2000)
    Range (lineitem_1)
    Comp (=)
      Identifier (l_orderkey)
      Identifier (l_orderkey)
  Comp (=)
    Identifier (o_orderkey)
    Identifier (l_orderkey)
```
Search Space Memory
- Compactly stores all explored alternatives (AND-OR graph)
- Groups together equivalent operator trees and their plans
- Provides memoization, duplicate detection, property and cost management, etc.

Groups

Expressions

1) SELECT (b > 20,)
b > 20
S1) SELECT (a < 10,)
a < 10
R
2) JOIN (x = y, , )
1) SELECT (a < 10, )
a < 10
b > 20
R S
1) GET (S)
S1) GET (R)
R
Memo and Properties

- Logical Properties
  - Valid for whole group
  - E.g.: cardinality, output columns, column equivalences, distinctness
  - Flow bottom-up

- Physical Properties
  - Valid for specific expression
  - E.g.: sort columns, halloween protection, cursors
  - Flow in both directions (derived vs. required)

5/25/2011
Stanford Talk
Optimization Tasks

Optimize Group
- Explore Group
- Obtain enforcers and implementation rules for all expressions
- Apply Rules by Promise

Optimize Inputs
- Obtain optimization context
- Optimize children groups
- Calculate costs, pick best

Apply Rule
- Pruning checks
- Generate bindings, substitutes
- Restrict rule set optimizing?Opt Inputs:Explore

Explore Group
- For each expression
- Get Guidance
- Explore Expression

Explore Expression
- Apply-always rules
- Explore Inputs
- Rest of Apply-always Rules
- Other Rules by Promise

Stanford Talk
Example (1/5)

OptimizeGroup 12: ExploreGroup 12, OptimizeGroup 12
   ExploreGroup 12: ExploreExpr 12.0
      ExploreExpr 12.0: ExploreGroup 7-8, applyRule [Comm, Assoc, ViewMatch, …]
         ExploreGroup 7: ExploreExpr 7.0
            ExploreExpr 7.0: ExploreGroup 5, applyRule [Sel2->Sel, Sel->LASJ,…]
               ExploreGroup 5: ExploreExpr 5.0
                  ExploreExpr 5.0: applyRule [ViewMatch, …]
         ExploreGroup 8: ExploreExpr 8.0
            ExploreExpr 8.0: applyRule [ViewMatch,…]
   ApplyRule [Join-Commutativity]
Rule Result -> Group 12.1 and (ExploreExpr 12.1)
OptimizeGroup 12: applyRule[JN->INL, JN->SM, ...]

ApplyRule (JN->INL)

Rule Result -> 12.2 and (optInputs 12.2)

INL (<7>, SelectIdx(
  GetIdx orders_CL,
  o_orderkey=l_orderkey)
)
Example (3/5)

Group 14: 0) SelectIdx 13 11
Group 13: 0) GetIdx orders_0
Root Group 12: 0) Join 7 8 11
  1) Join 8 7 11
  2) INLJoin 7 14
Group 11: 0) Comp (=) 9 10
Group 10: 0) Identifier (l_orderkey)
Group 9: 0) Identifier (o_orderkey)
Group 8: 0) Get (orders)
Group 7: 0) Select 5 6
Group 6: 0) Logical (and) 2 4
Group 5: 0) Get (lineitem)
Group 4: 0) Comp (<) 3 1
...

  OptGroup 14: ExploreGroup 14, OptGroup 14
    ExploreGroup 14: ExploreExpr 14.0
      ExploreExpr 14.0: ExploreGroup 13, no rules pushed
      ...
      OptGroup 14: applyRule[SelIdx->Rng]
        ApplyRule SelIdx->Rng:
          Rule Result -> 14.1 and (optInputs 14)
          ...
      OptGroup 7: applyRule [Sel->Filter, Sel->IDX,...]
        ApplyRule (Sel -> IndexExpression)
          Rule Result -> 7.1 and (OptInputs 7.1)
          ...

5/25/2011 Stanford Talk
ApplyRule Jn->SM (fourth implementation rule to root group 12)
Rule Result 12.3 and (OptInputs 12.3)

OptimizeInputs 12.3

OptGroup 7 (with sort required for column o_orderkey)

...
ApplyRule Jn->SM (fourth implementation rule to root group 12)
Rule Result 12.3 and (OptInputs 12.3)

OptimizeInputs 12.3
OptGroup 7 (with sort required for column o_orderkey)

...
Operators derive own I/O and CPU costs
- Based on input cardinalities, constraints, runtime parameters, etc.
- Combination of top-down (own cost for pruning) and bottom-up (tree cost)

Cardinality Estimation
- Uses variation of MaxDiff histograms, MCDs, constraints, and magic numbers
  - MCDs used for group by, MC joins, etc.
- Histograms are propagated through plan
- Small number of histogram buckets (~200)
Cardinality and Resulting Plans

```
SELECT l_tax, o_totalprice
FROM lineitem JOIN orders ON l_orderkey = o_orderkey
WHERE l_suppkey < 2000 AND l_partkey < 2000
```

```
SELECT l_tax, o_totalprice
FROM lineitem JOIN orders ON l_orderkey = o_orderkey
WHERE l_suppkey < 20000 AND l_partkey < 20000
```
Filtered Statistics

- CREATE STATISTICS on Column where <predicate>
- Filtered statistics provide opportunity to reduce two sources of errors in cardinality estimation
  - Independence assumption
    - E.g., Salary | (Age < 30) vs. Salary | (Age > 50)
  - Interpolation within a histogram bucket
    - Can be important for very large tables
- Workloads are complex
  - DBAs need help in deciding which filtered statistics are important
SQL Server Optimizer: Flexibility

[Diagram of SQL Server Optimizer process]

- Trivial Plan Optimizer
  - Found plan?
    - Yes
      - SMP and greater than parallelism threshold?
        - Yes
          - Full Optimization for Parallel Execution
        - No
          - Full Optimization for Serial Execution
    - No
      - Cost Based Optimizer: Phase 2 - QuickPlan
        - Found a cheap plan?
          - Yes
            - Output Plan
          - No
            - No, one phase remains
              - Re-run process

- Simplification and Statistics Loading
  - Cost Based Optimizer: Phase 1 - Transaction Processing
    - Found a cheap plan?
      - Yes
        - Output Plan
      - No
        - Re-run process
Extensibility: Nested Subqueries

```
select *
from customer
where 100,000 <
  (select sum(o_totalprice)
   from orders
   where o_custkey = c_custkey)
```

- Subqueries are expensive to execute
- Represented as relational operator trees
  - Not SQL-Block-focused
  - Apply Operator Abstraction
The Apply operator

- **R Apply E(r)**
  - For each row r of R, execute function E on r
  - Return union: \{r_1\} \times E(r_1) \cup \{r_2\} \times E(r_2) \cup \ldots
  - Abstracts “for each” and relational function invocation

- **Subquery removal**: Transform tree to remove relational operators from under scalar operators
  - The crux of efficient processing
  - “Unnesting”, “Decorrelation”
  - Get joins, outerjoins, semijoins ... as a result
Example: Apply Removal

```
select * from customer where
100,000 <
(select sum(o_totalprice) from orders where o_custkey = c_custkey)

SELECT(1000000<X)
  APPLY(bind:C_CUSTKEY)
    CUSTOMER SGb(X=SUM(O_TOTALPRICE))
      SELECT(O_CUSTKEY=C_CUSTKEY)
        ORDERS
```
Example: Apply Removal

SELECT(1000000<X)

Gb[C_CUSTKEY] X = SUM(O_TOTALPRICE)

LEFT OUTER JOIN (O_CUSTKEY=C_CUSTKEY)

CUSTOMER  ORDERS
SQL Server Optimizer: Summary

- Transformation-based, top-down approach
  No need for bottom-up interesting orders
- Fully Cost-based
  No separation into phases (heuristics + cost)
- Flexible and Extensible
  New operators, rules, and strategies are simple to add
- Adaptive
  - Automatic statistics create and refresh
  - Automatic optimization levels
Outline

- SQL Server optimizer
- **Critique of modern optimizers**
  - Limitations of Cardinality Estimation
  - Limitations of Search Space Exploration
  - Limitations of Cost Estimation
- Empirical analysis of optimizers
Limitations of Cardinality Estimation

- Single table expressions
  - Single column histograms are reasonable
  - No easy answers to modeling correlations
    - Multi-dimensional histograms (does not scale with #dimensions)
    - Sampling (need to execute queries over sample)
- Join expressions
  - Independence, containment assumptions lead to serious error propagation
  - Sampling over results of expression requires pre-computing join
- Is execution feedback the answer?
  - Not really, at least not yet
Query Execution Feedback

- Leverage accurate cardinalities obtained from query execution
  - Self-Tuning Histograms, LEO project, Dynamic Re-optimization
Example 1

\[
\text{SELECT SUM(Amount) FROM Orders WHERE State = 'WA' and Year > '2007' and Segment = 'HOME'} \]

- Crucial cardinalities not available via execution feedback

5/25/2011

Stanford Talk
• Collecting cardinalities only at the output of operators can be restrictive
  ▪ Can be “stuck” with a poor plan despite using execution feedback
• Using execution feedback information can incur non-trivial overheads during query optimization
Limitations of Search Algorithms

- Several ad-hoc elements in search algorithm of all modern optimizers
  - Introduced for pragmatics (think 50-way join!)
  - Unpredictable plan quality
- Trade-off between optimization time and plan quality not exposed
- Optimizer makes unnecessarily (?) fine-grained choices
  - Plan Diagrams
Limitations of Cost Estimation

- Knowledge of current system state not exploited
  - E.g. Buffer pool contents not considered in estimating I/O cost
  - Can be particularly serious for ad-hoc queries

- Modeling all important parameters is hard
  - E.g. Two tables R, S on same disk
    - ⇒ In Merge Join (R, S), sequential I/O not so sequential!

- Even parameters that are modeled may be inaccurate as H/W changes
Outline

- SQL Server optimizer
  - Enumeration architecture
  - Search space: flexibility/extensibility
  - Cost and statistics
- Critique of modern optimizers
- Empirical analysis of optimizers
Need for Empirical Information

- Effectiveness of query optimizers poorly understood
  - Which transformation rules are important?
  - How important are accurate cardinalities for plan quality?
  - How does time-outs and optimizer levels impact plan quality?
  - ...
- Better (empirical) understanding of limitations can help drive next generation query optimizers
  - Rule Profiling
  - Exact Cardinality Query Optimization
Rule Profiling

• Profiling is an integral part of software development
  • Crucial for understanding usage and effectiveness
• Apply “profiling” to query optimizers to better understand its effectiveness
  • Specifically to transformation rules
• Rule profiling can help answer questions such as:
  • Which rules are necessary for obtaining the optimal plan?
  • How does a rule (or set of rules) affect query performance?
Set of all rules – R

**Essential set of rules** – E

- $\text{Plan}(Q, E) = \text{Plan}(Q, R)$
- *This property is not true for any proper subset of $E*

May not be unique
Set of all rules – R

**Relevant rule for a query**
- Optimizing the query with the rule turned off changes the plan
  - Plan(Q, R) \(!=\) Plan(Q, R-{r})

**Relevance Ratio** of a rule r for a workload W
- Fraction of queries for which rule is relevant
Workloads

- **TPC-H**
  - Original DSS benchmark, 22 queries
- **TPC-DS**
  - New DSS benchmark, 100 complex DSS queries
- **SkyServer**
  - Astronomy database (publicly available)
  - User defined functions
- **InventoryDB**
  - Real world database of an online book retailer
  - Stored procedure with 8 table joins, many aggregates
- **SalesDB**
  - Microsoft sales database
  - 6-8 table joins
Essential set sizes are small, typically around 10% of the total number of rules.
Small number of rules with high relevance
- For TPCH, only 15 rules with relevance ratio > 50%

Stanford Talk
Impact on Plan Diagram Complexity

- Simplify plan diagrams using rule relevance information
- Use only the top 20 rules with highest *RelevanceRatio*
- Examine how plan diagram changes
  - 14 distinct plans (instead of 54)
  - Average Degradation is 6%
Substantial reduction in optimization time possible if essential set is known

- Can reduce optimization time through rule hints
- Revisit design of optimization levels through rule subsets
Conclusion

• Query optimizers play crucial role in success of DBMS for decision support queries
• Important abstractions, data structures and architectures have evolved
  ▪ Transformation rules
  ▪ Memoization
  ▪ Components: Search algorithm, cardinality estimation, cost model
• Despite success, several technical challenges remain
  ▪ No simple fixes likely
DMX Group at Microsoft Research

- [http://research.microsoft.com/dmx](http://research.microsoft.com/dmx)
- Managed by Surajit Chaudhuri
- Projects
  - AutoAdmin
  - Data Cleaning
  - Data Exploration
  - Data Security/Compliance
Related Bibliography

Questions?
Comments?