Automated Selection of Materialized Views and Indexes for SQL Databases

Robert Rübner, 03. 12. 2003

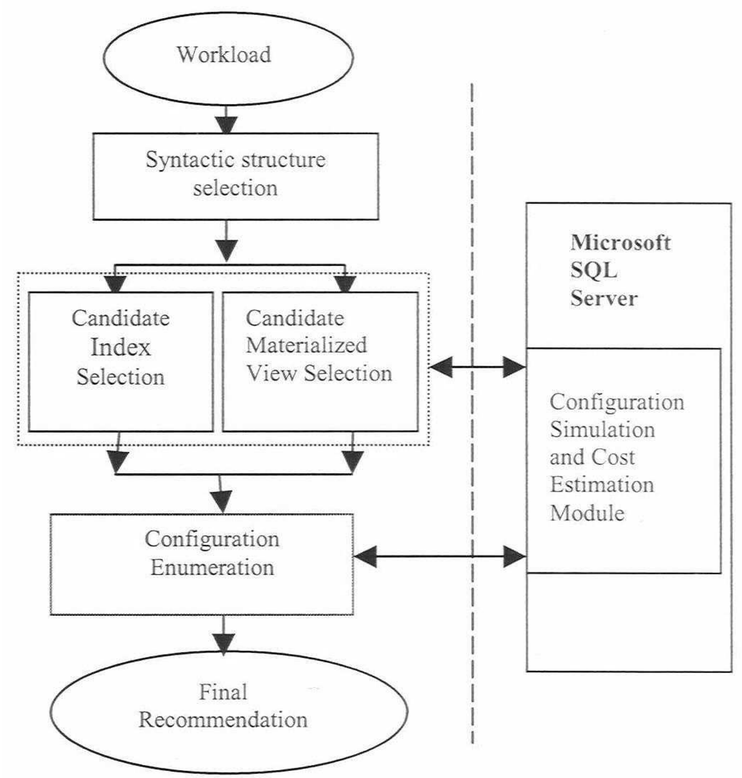
Structure

- Introduction
- Architecture for Index and Materialized View Selection
- Candidate Materialized View Selection
- Trading Choices of Indexes and Materialized Views
- Conclusion
- References

Introduction

- presence of the right materialized view improve performance
- to take into account the interaction between indexes and materialized views to optimise the physical design for the workload
- materialized view much richer in structure than an index
- two key techniques for an approach for candidate materialized view selection
- this work as part of the AutoAdmin research project at Microsoft

Architecture for Index and Materialized View Selection (I)



Architecture for Index and Materialized View Selection (II)

- first step to identify relevant indexes, materialized views and indexes on materialized views
- crucial to eliminate spurious indexes and materialized views from consideration early
- after chosen candidates find the ideal physical design, called **configuration**
- greedy algorithm for searching in the space
- an important characteristic that configuration enumeration is over the joint space of indexes and materialized views

Candidate Materialized View Selection

- goal to eliminate materialized views that not relevant for answering queries in configuration enumeration phase
- approach the task of candidate materialized view selection using three steps
  1) Finding interesting table-subsets
  2) Exploiting the query optimiser to prune relevant materialized views
  3) View merging

1) Finding interesting table-subsets

- table-subset interesting when reducing the cost of the workload, e.g., above a given threshold
- TS-Cost(T) = total cost of all queries in the workload where table-subset T occurs
- TS-Weight(T) = $\sum_i$ Cost(Q_i)*((sum of sizes of tables in T)/(sum of sizes of all tables in Q_i))
- TS-Cost(T) the property of “monotony” since for table subsets T_1, T_2, $T_1 \subseteq T_2 \Rightarrow$ TS-Cost(T_1) $\geq$ TS-Cost(T_2)

Algorithm for finding interesting table-subsets in the workload

1. Let S_1 = {T | T is a table-subset of size 1 satisfying *TS-Cost*(T) $\geq$ **C**}; i = 1
2. **While** i < MAX-TABLES and $|S_i| > 0$
3. $i = i + 1$; $S_i = \{\}$
4. Let G ={T | T is a table-subset of size i, and $\exists$ s $\in S_{i-1}$ such that s $\subset$ T}
5. **For** each T $\in$ G
   If *TS-Cost* (T) $\geq$ **C Then** $S_i = S_i \cup \{T\}$
6. **End For**
7. **End While**
8. $S = S_1 \cup S_2 \cup \ldots S_{MAX\text{-}TABLES}$
9. R = {T | T $\in$ S and *TS-Weight*(T) $\geq$ **C**}
10. **Return** R

2) Exploiting the query optimiser to prune relevant materialized views

- many of these materialized views, finding a step before, not relevant for answering any query
- because the decision is made by the query optimiser
- goal to prevent materialized views that are not used in answering any query from being considered during configuration enumeration

Cost-based pruning of syntactically relevant materialized views

1. M = {} /* M is the set of materialized views that is useful for at least one query in the workload W*/
2. **For** i = 1 to |W|
3. Let S_i = Set of materialized views proposed for query Q_i.
4. C = *Find-Best-Configuration* (Q_i, S_i)
5. M = M ∪ C;
6. **End For**
7. **Return** M

3) View merging (I)

- limited materialized views, get in step before, return maybe sub-optimal recommendations when storage is constrained
- set M good starting point for generating additional “merged” materialized views
- to explore the space by using a sequence of pair-wise merges
- addressing two key issues
  1) determining the criteria when and how to merge
  2) enumerating the space of possible merged views

3) View merging (II)

- MergeViewPair Algorithm
  - goal to create a new view with 2 properties
    1) new $view_{12}$ answering all queries which also can be answered using $view_1$ or $view_2$
    2) cost of $view_{12}$ not significantly higher than the cost of using views in M
- Algorithm for generating merged views
  - possible for a merged view to be merged again
  - set of returned merged views not depending on the exact sequence in which views are merged

Trading Choices of Indexes and Materialized Views

- indexes and materialized views interact with one another
- approach to consider joint enumeration of the space of candidate indexes and materialized views
- two alternatives to this approach

1) MVFIRST $\Rightarrow$ first select materialized views and then indexes
2) INDFIRST $\Rightarrow$ first select indexes and then materialized views

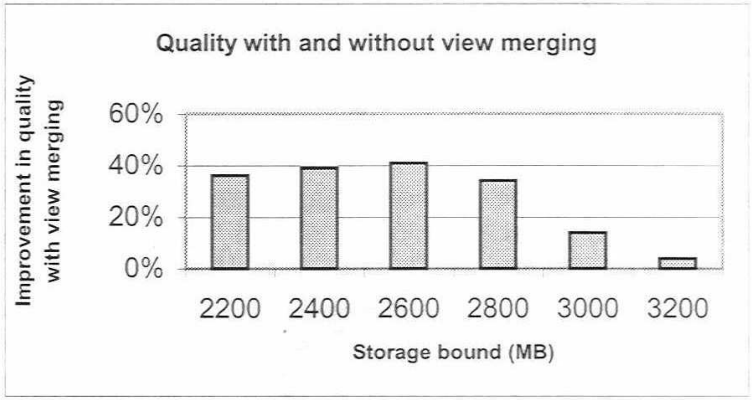
Selecting one feature set following by the other (MVFIRST, INDFIRST)

- for a global storage bound S and a fraction **f** ($0 \leq \mathbf{f} \leq 1$)
- determining **f** such that a storage constraint of **f***S to the first feature set
- using remaining storage for second feature set
- Problem: How to determine the fraction **f**?
  - depending on several attributes of the workload (e.g., complexity of queries)
- the optimal value of **f** changes from one workload to the next

Joint Enumeration (JOINTSEL)

- two attractions of joint enumeration of candidate indexes and materialized views
  1) a graceful adjustment to storage bounds
  2) considering interactions between candidate indexes and materialized views
- using the greedy algorithm for enumeration

Conclusion(I)



Quality vs. storage bound with and without view merging

Workload	Drop in quality of MVFIRST compared to JOINTSEL	Drop in quality of INDFIRST compared to JOINTSEL
TPCH-22	8%	0%
TPCH-UPD25	67%	11%

Comparison of alternative schemes without storage bound (e.g., storage = ∞)

Conclusion(II)

- architecture and algorithms are the foundation of a robust physical database design tool for Microsoft Server 2000 recommending both indexes and materialized views
- indexes and materialized views only a part of the physical design space
- to pursue the goal in the context of the AutoAdmin project of a complete physical design tool for SQL databases

References

- Paper from Agrawal S., Chaudhuri S., Narasayya V.
- AutoAdmin project, Microsoft Research
  http://www.research.microsoft.com/dmx/AutoAdmin
- Baralis E., Paraboschi S., Teniente E., Materialized View Selection in a Multidimensional Database, VLDB 1997