

# Recommending Materialized Views and Indexes with the IBM DB2 Design Advisor

Daniel C. Zilio et al

Proceedings of the International Conference on Automatic Computing (ICAC'04)

Rolando Blanco

CS848 - Spring 2005

# Overview

- Materialized Views
- DB2 Advisor extensions
- Other Systems
- Comment/Observations

# Materialized Views (MVs)

- Materialized partial query results
- Reason: Improve performance
- Drawbacks:
  - Introduce data redundancy
  - Maintenance:
    - \* Deferred
    - \* Immediate
- Good if they benefit many queries
- Design Issues:
  - What partial results to materialize
  - What to index (Require indices as any other tables)
  - How to refresh

Base tables may be partitioned.

# Materialized Views

- *Goal*: Minimize total cost of query evaluation and MV maintenance under limited resource (storage) constraint
- *Algorithm*:
  - *Input*: Workload, disk space constraint
  - *Output*: Optional set of MVs and indices for workload
  - *Implementation Options*:
    1. Choose MVs given storage constraint, then indices using remaining space
    2. (MVs, indices), (MVs, indices), ...
    3. (MV, indices) in one step

# DB2 Advisor

- MV and Index Recommendation:
  - a. Generation of MV candidates
  - b. Generation of index candidates
  - c. Stats estimation for MVs and indices
  - d. MV and index selection
  - e. Filtering

*Output:* (1) MVs, or (2) Indices on base tables, or (3) MVs, and indices on base tables and MVs.

*a, b, and c* by invoking optimizer under a new *EXPLAIN* mode.

# a. Generation of MV Candidates

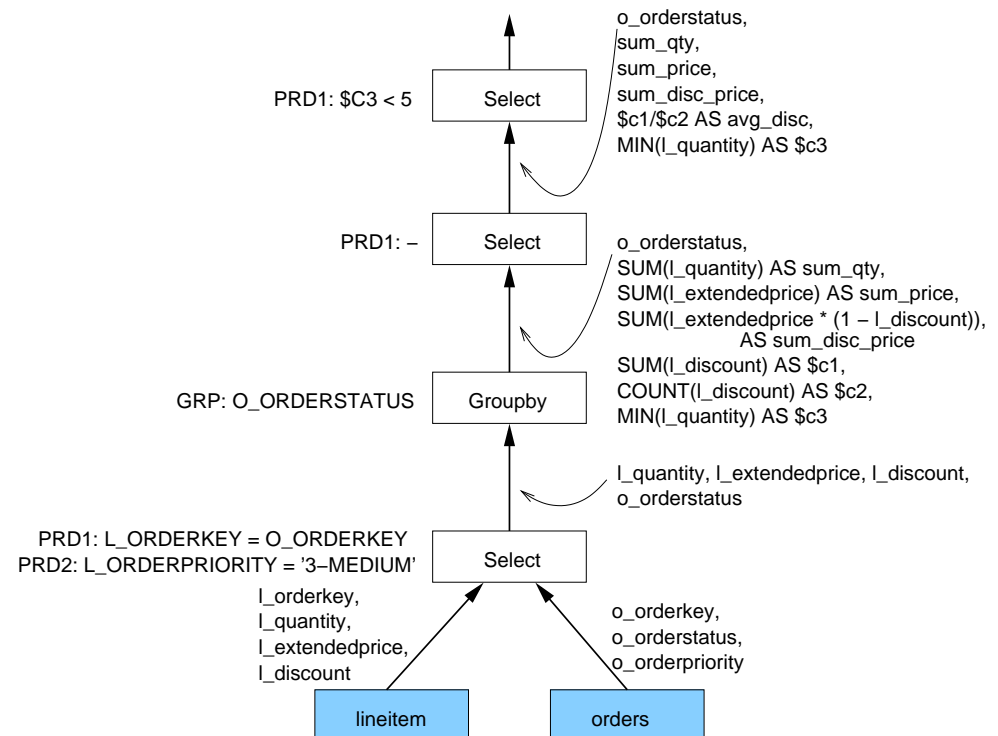
- Queries in workload (generalized)
  - select ... from ... where [group by]
- User defined views
- Multiple Query Optimization (MQO) to find common subexpressions among queries in workload

# a. Generation of MV Candidates

- MQO [LCPZ01]: Query represented as graph following DB2's query graph model (QGM)

```

SELECT o_orderstatus,
       SUM(l_quantity) AS sum_qty,
       AVG(l_discount) AS avg_disc,
       SUM(l_extendedprice)
           AS sum_base_price,
       SUM(l_extendedprice *
           (1 - l_discount))
           AS sum_disc_price
FROM lineitem, orders
WHERE l_orderkey = o_orderkey
      AND o_orderpriority = '3-MEDIUM'
GROUP BY o_orderstatus
HAVING MIN(l_quantity) > 5
    
```



# a. Generation of MV Candidates

- MQO
  - QGM traversed bottom up
  - Identification of common/similar subexpressions
  - Generalization of expressions if required
  - Compensation if data needs to be adjusted
    - \* Back-joins
    - \* Predicate adjustment
  - Rules:
    - \* Base table boxes refer to same data sources
    - \* Expressions must be derivable from generalized expression

## b. Generation of Index Candidates

- [VZZ<sup>+</sup>00]
  - Smart Column Enumeration for Index Scans (SAEFIS) – analysis of statement predicates and clauses
  - Brute Force and Ignorance (BFI) enumeration – enumeration of all possible indices with upper bound

*How are MVs considered?*

## c. Stats Estimation for MVs/Indices

- MVs:
  - Size: row width and cardinality for MVs
    - \* Estimation by optimizer, or
    - \* Data sampling; more reliable size estimates
- Indices:
  - Based on table and column(s) stats:
    - \* Number of unique values in index
    - \* B+-tree levels
    - \* Number of leaf and non-leaf pages

*Note indices on MVs will be estimated on estimates*

## d. MV/Index Selection

- Knapsack problem
- Obtain initial reasonable solution by relaxing integrality constraint
- $O(n \log n)$  by ordering by descending benefit/cost weight

$$Weight(A) = \frac{\sum_{q \in Q} f(q) * B(q)}{D(A)}$$

$$B(q) = E(q) - M(q)$$

$f(q)$ : query frequency

$B(q)$ : performance change

$D(A)$ : Disk space consumed by A

$E(q)$ : “Explain” plan cost with existent MVs and indices

$M(q)$ : “Explain” plan cost with existent and new MVs and indices

*Note  $B(q)$  not specific to A*

# d. MV/Index Selection

- Cost of maintaining MV:
  - Immediate refresh:
    - \* Option 1: Rows affected: Estimated number of rows for all MVs, estimated frequencies and number of updated rows for each update statement.

$|MV|/|BaseTable| * Cost\_Update(BaseTable)?$

$|Updates|$  calls to optimizer

- \* Option 2: Time to update:

```
foreach  $u \in Updates$  do
   $t1 =$  Estimate execution time for  $u$ 
  foreach  $v \in MVs$  do
     $t2 =$  Estimate execution time for  $u$  with  $v$ 
    Estimated time cost of  $v = t2 - t1$ 
```

Total calls to optimizer:  $|Updates| + |MVs| * |Updates|$

## d. MV/Index Selection

- Cost of maintaining MV:
  - Full refresh:

$$Weight(A) = \frac{\sum_{q \in Q} f(q) * B(q)}{D(A)} - g(A) * \frac{C(A)}{D(A)}$$

$g(A)$ : times  $A$  is materialized in the time interval, 1 by default (*Can it be deduced from workload?*)

$C(A)$ : cost of materializing  $A$

# d. MV/Index Selection

## ADD\_COMBINE

- Knapsack relaxed solution
  - Add MV if index selected
- Iterate random swapping
  - 8 times with no improvement, or
  - until time limit

*If MV swapped, its indices need to be swapped too; if index of MV swapped in, then MV must be added too*

Compensates for:

- $B(q)$  is the same for all MVs and indices used by query
- No way to compute  $Weight(A)$  given that  $B$  not selected
- Relaxation of integrality constraint

## e. Filtering

- Selection of MVs and indices that compete in some optimized query plan (Optimizer selects one or the other)
- Selections for other database systems in heterogeneous DBMS environments

# Experiments

- OLAP star schema with stats, no data (*no sampling*)
- 15 tables, 400 GBytes
- Workload: 12 OLAP queries, equal frequency
- Advisor limited to run 5 minutes

# Experiments

- Experiment 1:
  - Improvements from recommended MVs
  - Comparison of benefit from MQO candidates versus MQO + materializing workload
  - Result:
    - \* MQO: 30% improvement
    - \* MQO + Materialized queries: similar to MQO alone
- Experiment 2:
  - Improvement of MVs *and* indices
  - 89% improvement (versus 30% with MVs only)

# Summary/Contributions

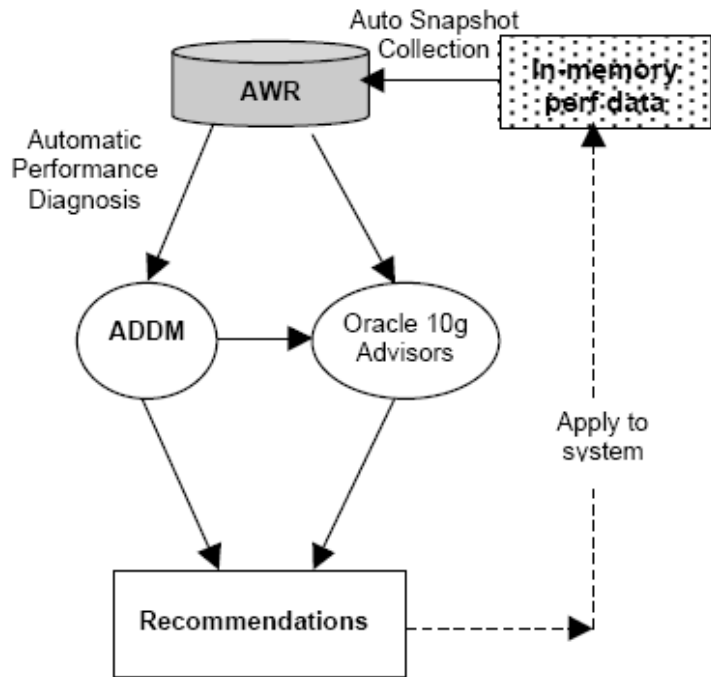
- Tool that uses optimizer itself to suggest and evaluate MVs and indices given a workload and disk space constraint
- Simple algorithm for finding “good” solution quickly

# Other Systems

- SQL Server [ACN00]
  - Generation of MV Candidates
    - \* MV candidates generated by syntactic structure
    - \* Single block MVs only, no back-joins. Hence MVs matching queries in workload not considered
    - \* MV for multiple queries via MV merging
  - Stats Estimation
    - \* No sampling
  - MV/Index Selection
    - \* Pruning based on benefit thresholds and *Greedy(m, k)* algorithm as with indices

# Other Systems

- Oracle 10g [DDD<sup>+</sup>04, DRS<sup>+</sup>05]



- AWR: Automatic Workload Repository
  - \* Performance stats and workload info
- ADDM: Automatic Database Diagnostic Engine.
  - \* Classification engine
  - \* Diagnosis every hour by default
- Advisors:
  - \* SQL Tuning Advisor. May recommend running SQL Access Advisor.
  - \* Segment Advisor
  - \* Memory Advisor

# Other Systems

- SQL Access Advisor:
  - Recommends what MVs and indices to create, drop, or retain.
  - PL/SQL interface accessible via PL/SQL, GUI or command line
    - \* Create task
    - \* Define workload
    - \* Generate recommendation
    - \* View and implement recommendations
  - SQL Profiling
    - \* Statistics about SQL statements
    - \* May require running query on sample data
    - \* Purpose is to reduce errors in various cost estimates, improving optimizer's operation and recommendations.

# Comments/Observations

- Matching and generalization of common subexpressions
- Experimentation:
  - Minimal
  - No comparison with optimal solution
  - No data sampling, just estimates (are improvements estimated, not corroborated?)
  - Cases where it does not work well (e.g. workloads with too many updates?)
- e. filtering: “useful in heterogeneous DBMS environments”
- d. MV/Index selection: “8 iterations”

# Comments/Observations

- Invalid SQL statements in subexpression examples / Why group by? MQT1: `SELECT D, E FROM S GROUP BY E`
- How are initial indices on MVs generated?
- No recommendations with regards to MV refresh interval
- Are incremental refreshes of MVs possible?
- Who determines if maintenance cost should consider number of rows or maintenance time?
- “Advisor is able to optimize large workloads of queries in reasonable amount of time”. Not supported (for example with experimentation). [ZRL<sup>+</sup>04]: “run time grows exponentially with linear increase of the workload size”. SAP: 30,000 database objects (includes tables and indices). Built-in workload compression.

# References

- [ACN00] Sanjay Agrawal, Surajit Chaudhuri, and Vivek R. Narasayya. Automated selection of materialized views and indexes in sql databases. In *VLDB '00: Proc's of the 26th International Conference on Very Large Data Bases*, pages 496–505, 2000.
- [DDD<sup>+</sup>04] Benoît Dageville, Dinesh Das, Karl Dias, Khaled Yagoub, Mohamed Zait, and Mohamed Ziauddin. Automatic sql tuning in oracle 10g. In *VLDB*, pages 1098–1109, 2004.
- [DRS<sup>+</sup>05] Karl Dias, Mark Ramacher, Uri Shaft, Venkateshwaran Venkataramani, and Graham Wood. Automatic performance diagnosis and tuning in oracle. In *CIDR*, pages 84–94, 2005.
- [LCPZ01] Wolfgang Lehner, Roberta Cochrane, Hamid Pirahesh, and Markos Zaharioudakis. fAST refresh using mass query optimization. In *Proc's of the 17th International Conference on Data Engineering*, pages 391–398, 2001.
- [VZZ<sup>+</sup>00] Gary Valentin, Michael Zuliani, Daniel C. Zilio, Guy Lohman, and Alan Skelley. Db2 advisor: An optimizer smart enough to recommend its own indexes. In *ICDE '00: Proc's of the 16th International Conference on Data Engineering*, page 101, 2000.
- [ZRL<sup>+</sup>04] Daniel C. Zilio, Jun Rao, Sam Lightstone, Guy M. Lohman, Adam Storm, Christian Garcia-Arellano, and Scott Fadden. Db2 design advisor: Integrated automatic physical database design. In *Proc's of the 30th VLDB Conference*, pages 1087–1097, 2004.