Graphic Era University

From the SelectedWorks of vijay singh

Winter February 10, 2009

Novel Materialized View Selection in a Multidimensional Database

vijay singh

Available at: https://works.bepress.com/vijaysingh/5/
**Abstract:** A multidimensional database is a data repository that supports the efficient execution of complex business decision queries. Query response can be significantly improved by storing an appropriate set of materialized views. These views are selected from the multidimensional lattice whose elements represent the solution space of the problem. Data analysis applications typically aggregate data across many dimensions looking for anomalies or unusual patterns. The SQL aggregate functions and the GROUP BY operator produce zero-dimensional or one-dimensional aggregates. Multidimensional databases store data in multidimensional structure on which analytical operations are performed. A challenge for these systems is how to handle large data sets in a large number of dimensions. Several techniques have been proposed in the past to perform the selection of materialized views for databases with a reduced number of dimensions. When the number and complexity of dimensions increase, the proposed techniques do not scale well. The technique we are proposing not only will reduce the solution space by considering only the relevant elements of the multidimensional lattice and further will support multiple query execution thereby reducing the time of result generation and is scalable.

**Keywords:** MDDB, fact table, OLAP, data cube, data mining, aggregation, summarization, database, analysis, query

**1. Introduction**

A multidimensional database (MDDB) is a data repository that provides an integrated environment for decision support queries that require complex aggregations on huge amounts of historical data. It is a form of database that is designed to make the best use of storing and utilizing data. Usually structured in order to optimize online analytical processing (OLAP) and data warehouse applications, the multidimensional database can receive data from a variety of relational databases and structure the information into categories and sections that can be accessed in a number of different ways. The multidimensional database - or a multidimensional database management system (MDDDBMS) - implies the ability to rapidly process the data in the database so that answers can be generated quickly. Conceptually, a multidimensional database uses the idea of a data cube to represent the dimensions of data available to a user.

the information is organized following the so-called star-model [11]. MDDB’s basic structure may be represented with the simple ER diagram depicted in Figure 1, in which all the Di entities represent the dimensions of the MDDB, while the connecting relationship F is the fact table.

![Figure 1: Entity-Relationship representation of an MDDB](image)

Each dimension table Di contains all the information that is specific only to the dimension itself, while the fact table F correlates all dimensions and contains information on the attributes of interest for the intersection of all the dimensions. A data-cube operator, has been proposed to perform the computation, on a single relation (the fact table), of one or more aggregate functions for all possible combinations of grouping attributes (which are the elements of the data-cube). [7] Since the computation of the elements of the cube is rather time-consuming it may be pre computed for satisfactory query response time. On the other hand the materialization of the complete cube may be unfeasible, because of size and time required to update the fact table.

The technique we are proposing works well with medium size database and can be scaled for increased complexity of actual operational MDDB’s. Solution space can be reduced significantly if a set of user-specified queries us available. We have seen total number of elements in data cube is considerably high with respect to number of representative queries. The indication of the relevant queries is exploited to drive the selection of the candidate views, if these views are materialized, it may yield reduction in cost. These candidate views can be reduced further by use of heuristic based estimation of the size of the candidate views. These views are discarded when their aggregation granularity increases, as their materialization would not yield a substantial improvement.
1.1 Related Work

Multidimensional data processing for relational data warehouses has raised considerable interest both in the scientific community and in the industrial community, where several products have appeared. [7, 8, 9, 10, 13, 14]. In particular, [10] considers an MDDB including only the fact table and proposes a greedy algorithm for the selection of an appropriate subset of the views of the complete data-cube to materialize. Work in [8] extends the previous results to the selection of both materialized views and indexes. Both works do not consider the cost of maintaining the materialized views in the model. A more general query and update model is proposed in [9], where a theoretical framework for the view-selection problem is presented.

[12] First gave a formal description of the multiple view maintenance problems. They present a framework for improving query performances by storing an additional set of materialized views and consider several heuristics for optimization. The cost model they propose, which includes both query and maintenance costs, uses an estimate of the number of disk accesses, by making hypotheses on the physical design of the database, while we selected a more abstract metric.

2. MDDB Modeling

Definition 2.1 A Multidimensional Database is a collection of relations D1, . . . , Dn, F, where

- Each Di is a dimension table, i.e., a relation characterized by an identifier di that uniquely identifies each tuple (di is the primary key of Di).
- F is a fact table, i.e., a relation connecting all tables D1, . . . , Dn; the identifier of F is given by the foreign keys dl, . . . , dn of all the dimension tables it connects; the schema of F contains a set of additional attributes V (representing the values on which the aggregate functions are applied).

Definition 2.2 Let D be a dimension table with identifier d. An attribute hierarchy on D is a set of functional dependencies FD = {fd0, fd1, . . . , fdn}, where each fd is characterized by two sets of attributes A_l ⊆ Attr(D) and A_r ⊆ Attr(D) (respectively called left side and right side of the dependency); the dependency is represented as fd : A_l + A_r

Definition 2.3 An attribute hierarchy is tree-like if disregarding all transitive dependencies (i.e., dependencies that can be deduced from other dependencies), all the attributes appear at most once in the right side of a functional dependency and the left side always contains a single attribute. The transitive closure of a tree-like hierarchy is a tree-like complete (FDTC) attribute hierarchy.

In this paper we consider only tree-like attribute hierarchies. In the algorithms we will often use complete hierarchies, in order to simplify the algorithm description.

3. A Practical Example

Consider a multidimensional database MDDB = {Product, Store, Time, Promotion, F}, where each dimension table has its corresponding attributes and with fact table F = {p, s, d, r, f}, having p, s, d (time dimension is represented with the granularity of day) and r as foreign keys of the dimension tables and f representing the amount of sales. The following queries can be requested on the MDDB:

Q1: total sales of product in store 2.
Q2: total sales per day in store 3.
Q3: promotions scheme in store 3.
Q4: total sales of product, per day in store 2.
Q5: promotion per (period) in store 3.
Q6: total sales of product with particular promotion, per (period) in store 3.

The queries we consider are select-join-group by queries, with some restrictions on the allowed selection and join predicates.

Figure 2: MDDB fact table diagram, fact tables are connected by a relation which is unique store id and is shared between all dimensions.

In this figure, we consider that four fact tables are connected with dimensions of different stores all having unique store id (s_id) example first F.T will contain store id from 1 to 5 other will store data of store id 6 - 10 and hence forth. All these fact tables
are connected by a relation, which will break the
single subsequent query in such a way that it will be
delivered to different fact tables depending upon
the store id and combining the results of broken up
queries will generate overall view of all the stores.
Since queries will run simultaneously thereby
reducing the query processing time.

Example- query: total sales in all stores.
Assume we have 20 stores and four fact tables are
used to store the information of these stores.
Hence query is divided into four parts each getting
the sales information from four groups of stores. As
these queries are not dependent on each other, they
will run simultaneously there by reducing processing
time.

4. Data Cube

A fundamental characteristic of MDDB queries is
that it is often possible to reuse the results of queries
to answer other queries. In our example, we can use
the result of query q2, q3 to answer query q5, adding
the promotion, period for store 3 to get the result. The
reuse of queries is strictly related to a new operator,
the data-cube [GBLP96]. This operator, receiving as
input a table T, a set of aggregating attributes A and a
function f, computes the union of the results of the
queries evaluating f, having as grouping attributes all
possible combinations of attributes in A.

Definition 2.4 Given an MDDB = {D1, . . . , Dn, F},
the data-cube lattice Cube-lattice of MDDB is the
lattice of the set of all possible grouping queries that
can be defined on the foreign keys of F. This lattice is
characterized by the following elements:
- an ordering relation defined as the
comparison between the sets of grouping
attributes (i.e., q^A1 ≤ q^A2 ⇔ A1 ⊆ A2);
- meet operator as union of the grouping
attributes;
- join operator as intersection of the grouping
attributes;
- The query grouping on all the foreign keys
as top element;
- The query computing the aggregate function
on all the tuples of F as bottom element
(empty set of grouping attributes).

5. The Multidimensional Lattice

The presence of dimensions makes the problem more
complex. The first aspect is the increase in the
number of potential grouping attributes, which
exponentially increases the number of elements of the
lattice. The second aspect is the presence of
hierarchies, which permit to remove some elements
from the lattice. In fact, consider a query grouping on
a dimension key di and also on an attribute aj of the
same dimension Di. Since there exists a functional
dependency from di to aj, a query grouping on {di, aj}
must produce the same result of the query grouping
on {di}. This observation is the basis for the
following generalization.

Definition 2.5 Let q^Az and q^Ay be two queries and
FD_DB the MDDB attribute hierarchy. The operator
ancestor (represented by the symbol ⊕) is defined by
the following algorithm:
Algorithm 5.1 Ancestor of two queries.
operator ⊕: q^Ax ⊕ q^Ay = q^Az
Az := A_x ∪ A_y;
for each fd_i ∈ FD_DB
for each a_j ∈ A_i if
{a_i} ∪ a_j ⊆ A_z
A_z := A_z - a_j;
return q^Az;
Algorithm 5.1 operates by building the union of
the attributes characterizing the queries and eliminating
all the elements for which there exists a functional
dependency in FD_DB. The result of applying the
operator $ to queries q_x and q_y is the “smallest”
query that contains all the information necessary for
answering q_z as well as q_Y. If applied in a reflexive
way (i.e., q^Ax ⊕ q^Ax), it eliminates all redundant
attributes.

Figure 3: Data Cube lattice for fig2 connecting 3
stores
Similarly, looking in the above fact table diagram and the corresponding data cube lattice diagram we can see the queries Q1, Q2, Q3, Q4, Q5, Q6 will generate the result in much lesser time.

Further we can see that result from

- Q1 and TS_ID2 will generate Q4.
- Q2 and Q3 will generate Q5.
- PTS_ID3 and Q5 will generate Q6.

We can use such type of query breaking and join mechanism to reduce the query response or execution time by executing some of candidate queries simultaneously.

5. Conclusion and Future Work

This improved mechanism of combining the fact tables based on different candidate elements in MDDB can highly increase the overall efficiency and reduce the query response time.

Also this type of combination is highly scalable and can be applied to enormous applications where the response time of query execution is of importance.

Future work can be done on deciding amongst the elements in MDDB for candidate elements and breaking and joining of candidate queries, as they can highly effect the response time.

7. References

[6] Lakshmanan, J. Pei, and J. Han. Quotient cube: How to summarize the semantics of a data cube. In Proc. 2002 Int. Conf. Very Large Data Bases (VLDB’02), Hong Kong, China, Aug. 2002.

8. Authors

*Namit Khanduja
Senior Lecturer, namitkhanduja@gmail.com

*Vijay Singh
Senior Lecturer, vijaysingh_agra@hotmail.com

*Navin Garg
Asst. Professor, naveengarg.geu@gmail.com

*Devesh Pratap Singh,
Senior Lecturer, devesh.geu@gmail.com

*Graphic Era University, Dehradun, Uttarakhand