Data Warehouse and OLAP. Methodologies, Algorithms, Trends

Radu LOVIN
IT Consultant – Tata Consultancy Services
GE European Equipment Finance - Data Warehouse Project
radu.lovin@ge.com

On-Line Analytical Processing (OLAP) is a component of Decision Support System (DSS). OLAP queries are complex aggregated computation queries over large data set. Speed is primary goal of OLAP applications. Computation of multiple related group-bys is one of the core operations on OLAP. Gray [8] proposed "CUBE" operator to support processing of group-by queries. Currently, there are two popular technologies for OLAP implementation: Relational OLAP (ROLAP) and Multidimensional OLAP (MOLAP). Some efficient algorithms have been developed for ROLAP, but even though multidimensional OLAP is more natural way to represent multidimensional data, not many algorithms have been developed for MOLAP. MOLAP systems, though may suffer from sparsity of data, are generally more efficient than ROLAP when sparsity of data cube is removed or when data set varies from small to medium size. In this article I just want to give you an overall picture about OLAP research activity in the last years, along with some basic information about Data Warehousing and Analytical Processing.

Keywords: *data warehouse, algorithms, trends, methodologies.*

1 Introduction

During the past decade the demand of decision support systems has emerged at a rapid rate. The separation of the enterprise information architecture into two separate environments has quickly become popular. Alongside with traditional On Line Transaction Processing (OLTP) environment a new On Line Analytical Processing (OLAP) component has been introduced, dedicated to decision oriented analysis of historical data. The central element of new environment is Data Warehouse, a read only repository of data that is used for analysis and assist in decision making of enterprise policy by the top executives.

Data warehouses and related OLAP technologies [4, 5] continue to receive strong interest from research community as well as from industry. A warehouse contains data from a number of independent sources, integrated and cleansed to support clients who wish to analyse the data for trends and anomalies. The decision support is provided by OLAP tools, which present their users with a multidimensional perspective of the data warehouse and facilitate the writing of reports involving aggregation along the various dimensions of the data set [4]. Because all these queries are often complex and the data warehouse database is often very large, processing the data queries quickly is an important prerequisite for building efficient Decision Support Systems (DSS). Before we introduce our approach to answering complex OLAP queries, we first review some important background information.

2. What is the Data Warehouse?

Maybe the first definition of what Data Warehouse means is dating from the early 1990s: the data warehouse is the queryable presentation resource for an enterprise's data and this presentation resource must not be organized around an entity-relation model due to lack of understandability and the performance. W. H. Imon, a leading architect of data warehouse concept, defined a data warehouse as a "subject-oriented, integrated, time variant and non-volatile collection of data in support of management's decision marketing process" [13]

3. Dimensional Model

The fundamental idea of dimensional modelling is that nearly every type of business data can be represented as a kind of cube data, where the cells of the cube contain measured values and the edges of the cube define the natural dimensions of the data. Of course, the model allows more than three dimensions in the design, so we technically should call the cube a hypercube, although the term cube and data cube are used by almost everyone. Any point inside the cube represents the "place" where the measurement of the business for that combination of dimensions is stored.

Facts – A fact is "an observation in the market place". It contains the measurement of the business. The most useful facts are numeric and additive.

Dimensions – A dimension is a collection of text-like attributes that are highly correlated with each other. The dimensions are syntactical categories that all allow us to specify multiple ways to look at business information, according to natural perspectives under which its analysis can be performed. Each dimension can be organized into a hierarchy of levels, corresponding to data domains at different granularity. A level can have description associated with it. Within a dimension, values of different levels are related through a family of rollup functions.

Variable – Variables are typical numerical measures like Sales, Cost, Profits, Expenses and so forth.

Hierarchy – A hierarchy is path of aggregation of dimensions. A dimension may have multiple levels of granularity, which have parent-child relationship. A hierarchy defines how these levels are related.

Member – A member is a data item in the dimensions. Typically, you create a caption or describe a measure of database using members.

Datacube – Datacubes are sets of precomputed views or special data structure (e.g. multidimensional arrays) [5] of selected data that are formed by aggregating values across attributes combinations (a group by in the database terminology). A generated cube on d attribute values can either be complete, that is it contains all 2^d possible views formed by attribute combinations or partial, that is contains only subset of 2^d possible views.

4. OLAP

OLAP stands for "On-Line Analytical Processing" and describes a class of technologies that are design for ad-hoc data access and analysis.

The general activity of querying and presenting text and numbers data from data warehouses, as well as a specifically dimensional style of querying and presenting that is exemplified by a number of "OLAP vendors". The OLAP vendors' technology is non-relational and is almost always based on an explicit multidimensional cube of data. OLAP databases are also known as multidimensional databases, or MDDBs. While relational databases are good at retrieving a small number of records quickly, they are not good at retrieving large number of records and summarizing them on the fly [14]. Slow response time and inordinate use of system resource are common characteristics of decision support built exclusively on top of relational database technology.

4.1. FASMI

The standard characteristic of OLAP is defined as Fast Analysis of shared Multidimensional Information (FASMI) [4].

- ✓ FAST – The system is targeted to deliver most of the response within seconds.
- ✓ ANALYSIS – The system can cope with any business logic and statistical analysis that is relevant for the application and the user. It is certainly necessary to allow the user to define ad-hoc calculations as part of the analysis and to report on data in any desired way without having to program.
- ✓ SHARED – The system implements all security requirements for confidentiality and, if multiple write accesses are required, concurrent updates and locking at an appropriate level have to be possible. But this is an area of weakness in many OLAP products, which tends to assume that all OLAP applications will be read only, with simple security control.
- ✓ MULTIDIMENSIONAL – The system must provide a multidimensional conceptual view of data including full support for hierarchies, as is certainly most logical way to ana-

lyse business and organization.

✓ INFORMATION – The system has to contain all the required information, as much as is relevant for application.

4.2. OLAP Operations

✓ ROLL-UP. Roll-up creates a subtotal at any level of aggregation needed from the most detailed up to the grant total. This function is also called consolidation [4].

✓ DRILL-DOWN. Drill-down breaks the subtotal at any level of granularity to lower level of granularity to lower level of granularity in hierarchy, means it gives details of relationship at the lower level.

✓ SLICING AND DICING – Selecting a subsection of datacube based on the constants in one or few dimensions. If one dimension is fixed, the operation is called slice and if more than one dimension are fixed, the operation is called dice.

✓ PIVOT. Pivoting is swapping of columns and rows. This allow user to look at data from different view. This is also commonly known as rotation [13].

4.3 OLAP Implementations

Currently there are two technologies for implementation of OLAP servers, namely Relational OLAP (ROLAP) and Multidimensional OLAP (MOLAP)

✓ OLAP might be implemented on standard or extended relation DBMS, called Relational OLAP (ROLAP) server (ex MetaCube by IBM). This server support extension to SQL and special access and implementation method to efficiently implement the multidimensional data model and operations. It is assumed that data is stored in relation databases either in the form of star or snowflake schema [5] or in the form of materialized views.

✓ Multidimensional OLAP (MOLAP) servers directly store multidimensional data in some special data structures (such as arrays) and implement the OLAP operations over these specials data structures [5]. Oracle's Oracle Express and Essbase from Hyperion come under this category [13].

✓ Hybrid online analytical processing (HOLAP) is a combination of relational OLAP (ROLAP) and multidimensional OLAP (MOLAP). HOLAP was developed to combine the greater data capacity of ROLAP with the superior processing capability of MOLAP. HOLAP can use varying combinations of ROLAP and OLAP technology. Typically it stores data in a both a relational database (RDB) and a multidimensional database (MDDB) and uses whichever one is best suited to the type of processing desired. The databases are used to store data in the most functional way. For heavy data processing, the data is more efficiently stored in a RDB, while for speculative processing, the data is more effectively stored in an MDDB. HOLAP users can choose to store the results of queries to the MDDB to save the effort of looking for the same data over and over which saves time. Although this technique - called "materializing cells" - improves performance, it takes a toll on storage. The user has to strike a balance between performance and storage demand to get the most out of HOLAP. Nevertheless, because it offers the best features of both OLAP and ROLAP, HOLAP is increasingly preferred. DbMiner is a product that adopts HOLAP methodology.

✓ Desktop OLAP (DOLAP) is a term used to denote single-tier, desktop-based business intelligence software. Its most distinguishing feature is its ability to download a relatively small hypercube from a central point (usually a data mart or data warehouse).

4.4. Related Works

Framework design for OLAP and data cube computation has drawn considerable amount of attention of database research community in the last decade. There is a number of literatures available on data computation [1, 10, 12, 15, 16] and modelling multidimensional database [2, 6, 7, 9, 11]. The algorithms proposed in [1, 12] are for ROLAP and aims to support CUBE operator as SQL extension in existing relational database systems rather than providing stand-alone (source data independent) OLAP applications. Though MOLAP is more natural way to implement OLAP operations only few [10, 15] algorithms have been developed for MOLAP. Moreover, the most of the algorithms proposed so far are designed with the consideration that we have very small memory at our disposal. There are few algorithms, so-called "in memory", based on a big amount of RAM. They are not implemented yet but they tend to be preferred in the near future due to rapidly decreasing price of memory. In the following lines I will try to present an overview of related work in the field of data warehousing and On Line Analytical Processing. Research related to this work falls under these categories: OLAP servers, including ROLAP and MOLAP, data cube computation, view materialization in data warehouse and developing framework for querying data cube.

4.4.1. ROLAP Server

ROLAP servers store the data in the relational tables using star or snowflake schema design [5]. In the star schema, there is a fact table plus one or more dimensions tables. The snowflake schema is a generalization of the star schema where the core dimensions have aggregation levels of different granularities. In the ROLAP approach, cube queries are translated into relational queries against the underlying star or snowflake schema using the standard relational operators such as selection, projection, relational join, group by, etc. However, directly executing translated SQL can be very inefficient. Therefore many commercial ROLAP servers extend SQL to support important OLAP operations directly. Examples of ROLAP servers are IBM's Metacube, Microstrategy, Seagate.

4.4.1.1. Computation and optimizing the computation of data cube

The first published work on methods of optimising the computation of data cube is referring to storing collections of group-bys [1]. Computing the cube stored on some relational tables requires generalization of standard relational aggregation operator. Two basic methods have been studied for computing single group-bys:

✓ Sort Based Method PipeSort

✓ Hash Base Method PipeHash

These two methods have been adapted to compute multiple group-bys as well by incorporating the following optimisations:

1. Smallest-parent – This optimisation, first proposed in [18], aims at computing a group-by from the smallest previously computed group by. In general, each group-by can be computed from a number of other group-bys. Figure 1 shows a four-attribute cube (ABCD) and the options for computing a group-by from a group-by having one more attribute called its parent. For instance, AB can be computed from ABC, ABD or ABCD. ABC and ABD are clearly better choices for computing AB. In addition, even between ABC and ABD, there can often be big difference in size making it critical to consider size in selecting a parent for computing AB.
2. Amortize-scans: This optimisation aims at amortising disk reads by computing as many group-bys as possible, together in memory. For instance, if the group-by ABCD is stored on disk, we could reduce disk read costs if all of ABC, ACD, ABD and BCD were computed in one scan of ABCD.
3. Cache-results: This optimisation aims at caching (in memory) the results of a group-by from which other group-bys are computed to reduce disk I/O. For instance, for the cube in Figure 1, having computed ABC, we compute AB from it while ABC is still in memory.
4. Share-sorts: This optimisation is specific to the sort based algorithms and aims at sharing sorting cost across multiple group-bys. For instance, if we sort the raw data on attribute order ABCD, then we can compute groups-bye ABCD, ABC, AB and A without additional sorts. However, this decision could conflict with the optimisation smallest-parent. For instance, the smallest parent of AB might be BDA although by generating AB from ABC we are able to share the sorting cost. So it's necessary to do a global planning to decide what group-by is computed from what and the attribute order in which it is computed. Here comes into picture the PipeSort algorithm.
5. Share-partitions: This optimisation is specific to the hash-based algorithms. When the hash-table is too large to fit in memory, data is partitioned and aggregation is done

for each partition that fits in memory. We can save on partitioning cost by sharing this cost across multiple group-bys.

For OLAP databases, the size of the data to be aggregated is usually much larger than the available main memory. Under such constraints, the above optimisations are often contradictory. For computing B, for instance, the first optimisation will favour BC over AB if BC is smaller but the second optimisation will favour AB if AB is in memory and BC is on disk.

The PipeSort Method

As I mentioned above, the PipeSort method combines the optimisations share-sorts and smallest-parent to get the minimum total cost. We can use a number of statistical procedures [19] for this purpose. The input to the algorithm is the search lattice, which is a graph where a vertex represents a group-by of the cube. A directed edge connects group-by i to group-by j whenever j can be generated from i and j has exactly one attribute less than i (i is called the parent of j). Thus, the out-degree of any node with k attributes is k. **Figure 1** is an example of search lattice. Level k of the search lattice denotes all group-bys that contain exactly k attributes. The keyword all is used to denote the empty group-by (Level 0). Each edge in the search lattice e_{ij} is labelled with two costs, as we will see in the algorithm presentation as well. The first cost $S(e_{ij})$ is the cost of computing j from i when i is not already sorted. The second cost $A(e_{ij})$ is the cost of computing j from i when i is already sorted.

The output, O, of the algorithm is a subgraph of the search lattice where each group-by is connected to a single parent group-by from which it will be computed and is associated with an attribute order in which it will be sorted. If the attribute order of a group-by j is a prefix of the order of its parent i, then j can be computed from i without sorting i and in O, e_{ij} is marked A and incurs cost $A(e_{ij})$. Otherwise, i has to be sorted to compute j and in O, e_{ij} is marked S and incurs cost S_{ij}. Clearly, for any output O, there can be at most one out-edge marked A from any group-by i, since there can be only one prefix of i in adjacent level. However, there can be multiple out-edges marked S from i. The objective of the algorithm is to find an output O that has minimum sum of edge costs.

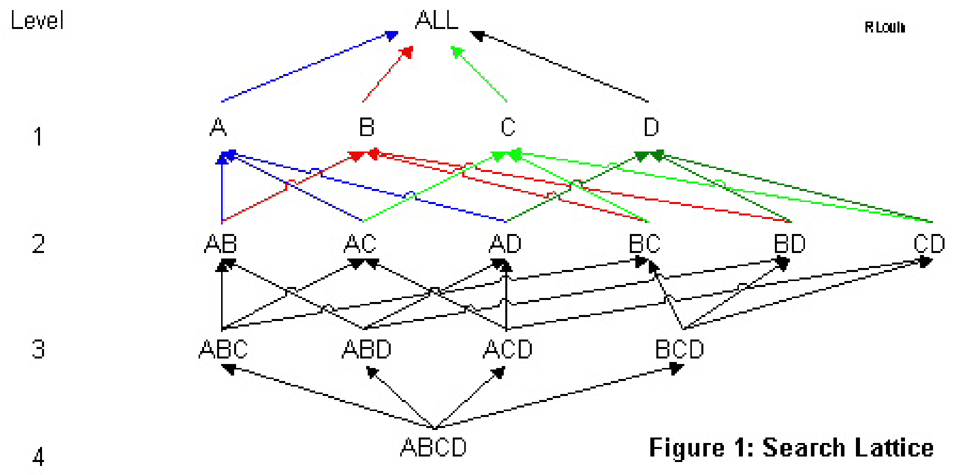

Figure 1: Search Lattice

Algorithm. The algorithm proceeds level-by-level, starting from level k-0 to level k=N-1, where N is the total number of attributes. For each level k, it finds the best way of computing level k from level k+1 by reducing the problem to a weighted bipartite matching problem [20] as follows.

We first transform level k+1 of the original search lattice by making k additional copies of each group-by in that level. Thus each level k+1 group-by has k+1 vertices whish is the same as the number of children or out-edges of that group-by. Each replicated vertex is connected to the same set of vertices as the original vertex in the search lattice. The cost on an edge e_{ij} from the original vertex i to a level k vertex j is set to $A(e_{ij})$ whereas all replicated vertices of i have edge cost set to $S(e_{ij})$. We then find the minimum 3 cost matching in the bipartite graph included by this transformed graph. In the matching so found, each vertex h in level k will be matched to some vertex g in level k+1. If h is connected to g by an A() edge, then h determines the attribute order in which g will be sorted during its computation. On the other hand, if h is connected by an S() edge, g will be re-sorted for computing h.

For illustration, we show how level 1 group-bys are generated from level 2 group-bys for a three-attribute search lattice. As shown in **Figure 2**, we first make one additional copy of each level 2 group-by. Solid edges represent the A() edges whereas dashed edges indicate the S() edges. The number underneath each vertex is the cost of all out-edges from this vertex.

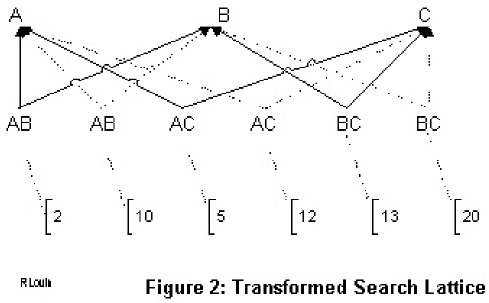

Figure 2: Transformed Search Lattice

In the minimum cost matching (**Figure 3**), A is connected to AB with an S() edge and B is connected to AB with an A() edge . Thus at level 2, group-by AB will be computed in the attribute order BA so that B is generated from it without sorting and A is generated by resorting BA. Similarly, since C is connected to AC by an A() edge, AC will be generated in the attribute order CA. Since BC is not matched to any level-1 group-by, BC can be computed in any order.

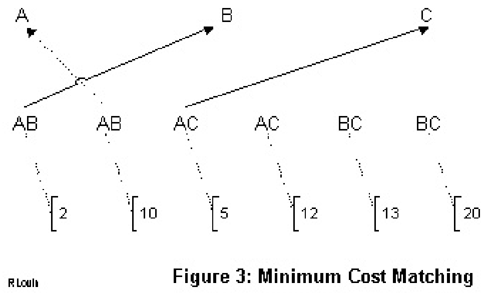

Figure 3: Minimum Cost Matching

The PipeHash Method

For hash based methods, the new challenge is careful memory allocations of multiple hash-tables for incorporating optimizations like cache-results and amortize-scans. For instance, if the hash tables for AB and AC fit in memory then the two group-bys could be computed on one scan of ABC. After AB is computed one can compute A and B while AB is still in memory and thus avoid the disk scan for AB. If memory were not a limitation, we could include all optimizations stated at 5.4.1.1.

However, the data to be aggregated is usually too large for the hash-tables to fit in memory. The standard way to deal with limited memory when constructing hash tables is to partition the data on one or more attributes. When data is partitioned on some attribute, say A, then all groups-bys that contain A can be computed by independently grouping on each partition – the results across multiple partitions need not be combined. We can share the cost of data partitioning across all group-bys that contain the partitioning attribute, leading to the optimization share-partitions. The algorithm PipeHash, which incorporates this optimization and also includes the optimizations cache-results, amortize-scans and smallest-parent, has as an input the search lattice described in the previous session. The algorithm fist chooses for each group by, the parent group by with the smallest estimated total size. The outcome is the minimum spanning tree (MTS) where each vertex is a group-by and an edge from group-by a to b shows that a is the smallest parent of b. In general, the available memory will not be sufficient to compute all the group-bys in the MST together, hence the next step is to decide what group-bys to compute together, when to allocate and de-allocate memory for different hash-tables, and what attribute for partitioning data.

Optimizations cache-results and amortize-scans are favoured by choosing as large a subtree of the MST as possible so that we can use the method above to compute together the group-bys in the subtree. However, when data needs to be partitioned based on some attribute, the partition attribute limits the subtree to only include group-bys containing the partitioning attribute.

4.4.1.2. Data Cube as set of materialized views.

In ROLAP systems data cube can also be maintained as set of materialized views. An algorithm to implement data cube as set of materialized views was proposed in [21]. In this approach each cell of data cube is considered as view of aggregation of interest like total sales. The values of many data cube cells are dependent on the values of other cells in the data cube. The most common optimization is materializing all or some of these cells of the data cube cells. The algorithm proposed in [21] is a polynomial time greedy algorithm that works with a lattice and determines good set of views to be materialized such that a good trade-off between the space used and the average time to answer a query is maintained.

4.4.2. MOLAP Server

MOLAP servers use multi-dimensional arrays as the underlying data structure. MOLAP is often several times faster than ROLAP alternative when the dimensionality and domain size are relatively small compared to available memory [7, 10, 14, 15]. However, when the number if dimensions and their domain size increases, the data become very sparse, resulting many empty cells in the array structure. A popular technique to deal with the sparse data is chunking [15]. The full array is chunked into small pieces called cuboids. For non-empty cells, a pair is stored.

MOLAP system has different sort of challenges in computing data cube than ROLAP systems. The fundamental difference is the different data structures used to store the data. For MOLAP systems where cube is stored in the form of multidimensional arrays, we can use some rules of thumb to compute data cube efficiently (eg shortest parent) described in [8]. Unfortunately, none of the techniques developed for ROLAP cube computation can apply. The main reason is that there is no equivalent of "reordering to bring together related tuples" [15] based upon their dimensions values.

Array based algorithm for simultaneous multidimensional aggregates

A Multi-Way Array cubing algorithm was introduced in [15]. The idea behind this approach is that the cells of the cube are visited in the right order so that a cell does not have to be revisited for each sub-aggregate. The goal is to overlap the computation of all these group-bys and finish the cube in one scan of array with the requirement of memory minimized. In case the data cube is too large so that it can't fit into memory, the array is split into chunks each of which small enough to fit into memory. Zhao [15] also introduced the concept of optimal dimension ordering and Minimum Memory Spanning Tree (MMST). MMST is similar to Minimum Spanning Tree (MST). A MMST, for a given order, is minim in terms of the total memory requirement for that dimension order. The optimal dimension order is that dimension for which MMST requires least amount of memory. For this method, the authors [15] have given stress on the fact that the related group by can be computed when the raw data is being scanned. They made the assumption that they would have enough memory to allocate the required memory for related group-by nodes in MMST.

CubiST: A new approach to speed up OLAP queries

In [10] an in memory algorithm called CubiST (Cubing with Statistical Trees), for evaluating OLAP queries on the top of a relational data warehouse is processed. CubiST can be considered as MOLAP approach in spite of the fact that CubiSt does not use multidimensional arrays directly [10]. They introduced a new data structures, called Statistical Tree (ST) to speed up computation of data cube. A statistical tree is a multi-way tree in which an internal node contains references to the next-level nodes which are directly used in the query evaluation. Leaf nodes hold the statistics or histogram of data and are linked together to facilitate scanning, similar to B-Tree data structures. Each root-to-leaf path in a statistical tree represents a particular sub-cube of underlying data set. In order to use a ST to answer cube queries over a particular data set, one must first pre-compute the aggregation on all sub-cubes by scanning detailed data set. CubiST encodes all possible aggregate views in the leaves of the underlying ST during one time scan of the detailed data. The algorithm requires only one scan over detailed data set. CubiST is focused on classes of queries which return aggregated values only. There is no information given in this method regarding modelling dimension hierarchy and about ROLL-UP, DRILL-DOWN and SLICE AND DICE operations. However, CubiST performs well in case of dense datacube and specially when cardinality of dimension is low.

Conclusions

The data warehousing concepts are meant to enable the business to win in the marketplace everyday, with every old or newly acquired customer, with every new purchase. They are able to determine clients' wishes, habits, dreams and to offer them the right product or service that, sometimes, they were not even conscious that really need it. The data warehouse and OLAP research is one of the most important activities in the universities across the world. Huge amounts of money are invested yearly to find and develop new technologies and algorithms. And I have no doubts saying that transformations of thousands of terabytes of data will be soon a matter of seconds and couple of mouse clicks.

Bibilografy

[1] S. Agrawal, R. Agrawal, P. M. Deshpande, A. Gupta, J. F. Naughton, R. Ramakrishnan, S. Sarawagi. *On the computation of multidimensional aggregates*. 1996 – International Conference Very Large Databases, Bombay, India.
[2] R. Agrawal, A. Gupta, S. Sarawagi. *Modeling multidimensional databases*. In Int. Conf. on Data Engineering (ICDE), IEEE Press, pages 232-243, April 1997
[3] G. Colliat. *OLAP, relational and multidimensional databases systems*. SIGMOND Record 25, pages 64-69, 1996
[4] E. F. Codd, S.B. Codd and C. T. Salley. *Providing OLAP (on-line analytical processing) to user analysis: An IT mandate.*In E.F. Codd & Associates, available on www.essbase.com/resource_library/white_papers, 1993
[5] S. Chaudhuri and U. Dayal. An overview of data warehousing and OLAP technology. ACM SIGMOND Record 26, pages 65-74, 1997.
[6] L. Cabbibo and R. Torlone. A logical approach to multidimensional databases. In Advances in Database Technology – EDBT'98, Number 1377 in LNCS, Springer, 1998.
[7] D. W. Cheung, B. Zhou, B. Kao, K. Hu, S. D. Lee. DROLAP: Dense Region based OLAP. In "Data and Knowledge Engineering", Volume 36, Number 1, pages 1-27, January 2001
[8] J. Gray, A. Bosworth, A. Layman, H. Pirahesh. Datacube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. In Proceedings of IEEE Conference on Data Engineering, pages 152-159. IEEE Computer Society, 1996.
[9] W. Leher. Modeling Large Scale OLAP Scenarios. In Advances in Database Technology – EDBT'98, Number 1377 in LNCS, Springer, 1998.
[10] L. Fu. And Joachim Hammer. CUBIST: A New Approach to Speeding Up OLAP Queries in Data Cubes. In Proceedings of the ACM Third International Workshop on Data Warehousing and OLAP, Washington, DC, November 2000.
[11] C. Sapia, M. Blaschka, G. Hofling. An Overview of Multidimensional Data Models for OLAP, FORWISS Technical Report 1999-001, available from www.forwiss.tumuenchen.de/~system42/publications, February 1999
[12] K. Ross and D. Srivastav. Fast computation of sparse datacube. In Proc. 23rd Int. Conf. Very Large Databases, pages 116-125, Athens, Greece, August 1997
[CM89]
[13] M. Corry, M. Abbey, I. Abramson, b. Taub. *Oracle 8i Data Warehousing*. Tata McGraw Hill Publishing Co. Ltd, 2001.
[14] S. Samtani, M. Mohania, K. Vijay, and Y. Kambayashi. *Recent Advances and Research Directions in Data Warehousing Technology*. Australian Journal of Information Systems, 2001
[15] Y. Zhao, P. Deshpande and J. Naughton. *An array-based algorithm for simultaneous multidimensional aggregates.* In Precedings of 2001 ACM SIGMOND Conference on Management od data, pages 159-170, Tucson, Arizona, May 2001
[16] R. S. Craig, J. A. Vivona, D. Berocovitch – *Microsoft Data Warehousing Building Distributed Decision Support Systems.* John Willy & Sons, Inc, New York, 2000
[17] Jim Gray, Adam Bosworth, Andrew Layman and Hamid Pirahesh. Data Cube: A relational Operator Generalizing Group-By, Cross-Tab and Sub-Totals. Proc. Of the 12th Int. Conf. on Data Engineering, pp 152-159, 1996
[18] C. T. Ho, R. Agrawal, N. Megiddo. *Techniques for Speeding up Range-Max Queries in OLAP Data Cubes.* In Proceedings of ACM SIGMOND Conference on Management of Data, pages 73-88, May 1997.
[19] P.J. Haas, J.F. Naughton, S. Seshadri, L. Stokes. *Sampling based estimation of the number of distinct values of an attribute.*In Proceedings of the Eight International Conference on Very Large Databases, pages 311-322, Zurich, Switzerland, September 1995.
[20] C. H. Papadimitriou and K. Steiglitz. Combinatorial Optimization: Algorithms and Complexity, chapter 11, pages 247-254. Englewood Cliffs, N.J., Prentice Hall, 1982
[21] J.D. Ullman, V. Harynaraian, A. Rajaraman. *Implementing Datacube Efficiently*. SIGMOND record (ACM Special Interest Group on Management of Data), pages 205-216, 1996.
[22] - http://www.dmreview.com