Jens Albrecht, Wolfgang Lehner

On-line Analytical Processing in Distributed Data Warehouses

Erstveröffentlichung in / First published in:

International Symposium on Database Engineering and Applications (IDEAS'98). Cardiff, 08.-10.07.1998. IEEE, S. 78–85. ISBN 0-8186-8307-4

DOI: https://doi.org/10.1109/IDEAS.1998.694361

Final edited form was published in "International Symposium on Database Engineering and Applications (IDEAS'98). Cardiff, 1998". ISBN 0-8186-8307-4.
https://doi.org/10.1109/IDEAS.1998.694361

ON-LINE ANALYTICAL PROCESSING IN DISTRIBUTED DATA WAREHOUSES

Jens Albrecht, Wolfgang Lehner
University of Erlangen-Nuremberg, Germany
{jalbrecht, lehner}@informatik.uni-erlangen.de

Abstract

The concepts of 'Data Warehousing' and 'On-line Analytical Processing' have seen a growing interest in the research and commercial product community. Today, the trend moves away from complex centralized data warehouses to distributed data marts integrated in a common conceptual schema. However, as the first part of this paper demonstrates, there are many problems and little solutions for large distributed decision support systems in worldwide operating corporations. After showing the benefits and problems of the distributed approach, this paper outlines possibilities for achieving performance in distributed on-line analytical processing. Finally, the architectural framework of the prototypical distributed OLAP system CUBESTAR is outlined.

1 Introduction

Today's global economy has placed a premium on information, because in a dynamic market environment with many competitors it is crucial for an enterprise to have on-line information about its general business figures as well as detailed information on specific topics to be able to make the right decisions at the right time. That's why today almost all big companies are trying to build data warehouses. In contrast to former management information systems based mostly on operational data, data warehouses contain integrated and non-volatile data and provide therefore a consistent basis for organizational decision making. In addition to classical reporting, the main application of data warehouses is On-line Analytical Processing (OLAP, [CoCS93]), i.e. the interactive exploration of the data. The data warehouse promise is getting accurate business information fast.

But the ultimate goal of data warehousing and OLAP goes even further, the vision is to "put a crystal ball on every desktop". Today that ideal is far from being reality. Because first, to build a single centralized data warehouse serving many different user groups takes a long time. Setup and maintenance are very expensive. All this contributes to a relatively inflexible architecture. Second, local access behavior is considered in the data warehouse design neither on the conceptual nor on the internal but only on the external layer.

Therefore, many companies today decide to start with smaller, flexible data marts dedicated to specific business areas. In order to get the possibilities for cross-functional analysis, there are two possibilities. The first one is to create again a centralized data warehouse for only cross-functional summary data. The other one is to integrate the data marts into a common conceptual schema and therefore create a distributed data warehouse.

We will concentrate on the second alternative which allows more flexible querying. To the user a distributed data warehouse should behave exactly like a centralized data warehouse. For transparent and efficient OLAP on a distributed data warehouse several problems need to be solved. The intention of this article is not to present final solutions but to show the potential of the distributed approach and the challenges for researchers and software vendors as well.

In the following section we will characterize different data warehouse architectures and their inherent drawbacks. After showing the benefits of a distributed approach to data warehousing we show the problems resulting for distributed OLAP systems. Performance as the main problem is the issue of section 3. Finally, in section 4 we give an overview over the prototypical distributed OLAP system CUBESTAR.

2 Data Warehouse Architectures

According to [Inmo92], a data warehouse is a "subject-oriented, integrated, time-varying, non-volatile collection of corporate data". Since that definition is very generic, we will now give an overview of three types of data warehouse architectures, partly based on [WeCa96].

2.1 Enterprise Data Warehouses

What most people think of when they use the term data warehouse is the enterprise or corporate data warehouse, where all business data is stored in a single database with a single corporate data model (figure 1). Enterprise data warehouses are created to provide knowledge workers with consistent and integrated data from all major business areas. They offer the possibility to correlate information across the enterprise.

Final edited form was published in "International Symposium on Database Engineering and Applications (IDEAS'98). Cardiff, 1998". ISBN 0-8186-8307-4.
https://doi.org/10.1109/IDEAS.1998.694361

But, because of the huge subject area (the corporation) and the vast amount of different operational sources that need to be integrated, enterprise data warehouses are very difficult and time-consuming to implement. Too many different data sources and too many different user requirements are likely reasons for the failure of the whole data warehouse project.

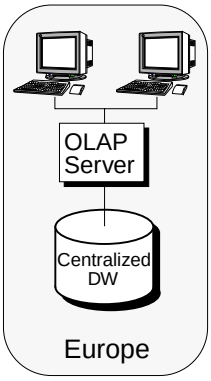


Fig. 1: Enterprise Data Warehouse

Since the data warehousing setup strategy from the analysts point of view is, as Inmon puts it, "give me what I want and I will tell you what I really want" [Inmo92], an incremental approach has proven to be the right choice.

2.2 Data Marts

The term data mart stands for the idea of a highly-focused version of a data warehouse (figure 2). There is a need to distinguish data marts which are local extracts from an enterprise data warehouse and data marts which are serving as the data warehouse for a small business group with well defined needs. In the latter case, which is the definition we will use in this paper, the limitation in size and scope of the data warehouse, e.g. only sales or marketing, dramatically reduces setup costs. Data marts can be deployed in weeks or months. That is why most of the current data warehousing projects include data marts.

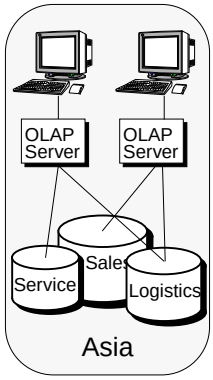


Fig. 2: Data Marts

2.3 Distributed Enterprise Data Warehouses

Although the data mart idea is very attractive and potentially offers the opportunity to build a corporate wide data warehouse from the bottom up, the benefits of data marts can easily be outweighed if there is no corporate-wide data warehouse strategy. If data mart design is not done carefully, independent islands of information are created and cross-functional analysis, as in the enterprise data warehouse, is not possible.

Therefore, the pragmatic approach to combine the virtues of the former is to integrate the departmental data marts into a common schema. The result is a distributed enterprise data warehouse (figure 3), also called federated data warehouse [WeCa96] or enterprise data mart [Info97].

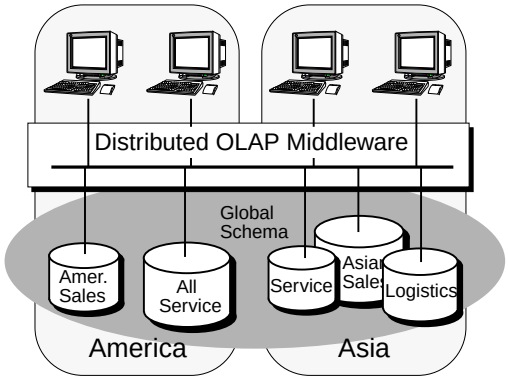


Fig. 3: Distributed Enterprise Data Warehouse

The benefits of this divide-and-conquer approach are clear. In the distributed enterprise data warehouse the individual data marts can be created and managed flexibly.

The flexibility for the individual data marts is traded for additional complexity in the management of the global schema and query processing across several data marts. Table 2.1 compares some characteristics to the centralized data warehouse.

Characteristics	Centralized DW	Distributed DW	
		Local Area	Wide Area
Conceptual Flexibility	low	low	moderate
Data Distribution	none	moderate	high
Query Processing	easy	moderate	complex
Communication Costs	low	moderate	high
Security Requirements.	none	low	low
Local Maintenance	hard	easy	easy
Global Maintenance	n/a	easy	moderate

Tab. 2.1 Characteristics of Centralized and Distributed Data Warehouses

There are basically two scenarios for a distributed enterprise data warehouse. The local area scenario (e.g. the data marts of only the Asian division) characterizes what first comes into mind. The extension of this approach to a wide area scenario (the global setting depicted in figure 3) is possible, but the high degree of data distribution results in much more complexity for query processing since communication costs become a major factor. The more global the distributed data warehouse is, the more important is increasing local access by replication. On the contrary, the more local the scenario is, the more likely techniques for load balancing in a shared-nothing server cluster can be applied.

Final edited form was published in "International Symposium on Database Engineering and Applications (IDEAS'98). Cardiff, 1998". ISBN 0-8186-8307-4.
https://doi.org/10.1109/IDEAS.1998.694361

2.4 Discussion

The current trend in the creation of data warehouses follows the bottom up approach where many data marts are created with a later integration into a distributed enterprise data warehouse in mind. The Ovum report [WeCa96] predicts that centralized data warehouses will become increasingly unpopular due to their inherent drawbacks and thus be replaced by a set of interoperating data marts.

The commercial product community seems to confirm the prediction. A first step towards easy integration was recently done by the Informatica Corp. with the release of its tools for enterprise data mart support. These tools are proposed to easily allow the creation and maintenance of distributed data marts and a seamless integration into a common schema. Moreover, query tools can directly access the metadata by providing a metadata exchange API. OLAP tool vendors like Microstrategy Inc. already announced to support this feature. For querying, “it means that all data marts are working off the same sets of data definitions and business rules allowing users to query aggregate data that may be distributed across multiple disparate data marts” [Ecke97]. Companies like Cognos [Cogn97] and Hewlett Packard [HP97] also offer already rudimental support for distributed data warehouses.

But to our knowledge, no query tool today is able to *transparently and performantly* generate and execute queries in distributed setting. An intelligent middleware layer as the basis for traffic routing, aggregate caching and load balancing is needed. This is the problem we will focus on in the remaining sections.

3 Achieving Performance in Distributed OLAP Systems

In order to deserve the attribute on-line, queries should never run much longer than a minute. In this section we will outline some specific methods and necessary prerequisites to achieve performance in large distributed OLAP environments. Note, that some of these also apply in the non-distributed case but become essential in the distributed case.

3.1 The Selection and Use of Aggregates

Researches and vendors agree that the main approach to speed up OLAP queries is the use of aggregates or materialized views [LeRT96]. In the last two years many articles on the selection of views to materialize in a data warehouse were published. A good overview over techniques on the selection of aggregates can be found in [ThSe97].

Dynamic Management of Aggregates. None of the recent articles considers the dynamics of an on-line environment with dozens of completely different queries to execute in every minute. If user behavior is considered at all, only a fix number of queries is taken as the basis for the aggregation algorithms ([GHRU97], [Gupt97], [BaPT97]). On the commercial product side there are OLAP tools like Microstrategy’s DSS Tools or Informix’ Metacube which are able to give the DBA hints for the creation of aggregates based on monitored user behavior. However, if the DBA decides to create aggregates, these are static and a change in the user behavior is not reflected.

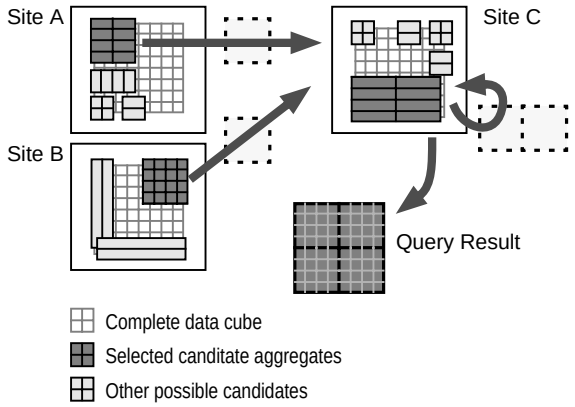


Fig. 4: Patchworking algorithm

Today, aggregates are more or less seen as another multidimensional access method, similar to an index, that must be managed by the DBA. But only a self-maintaining, dynamic aggregate management working like a (distributed) system buffer or cache for aggregates offers the needed flexibility and the potential for high query response times and low human maintenance efforts. The selection and the usage of aggregates should be completely transparent to the user. The DBA should only be able to setup parameters for the creation of but not the content of aggregates.

Partial Aggregates. The algorithms for the selection of aggregates explicitly supporting the multidimensional model compute preaggregates that are based on the full scope of the data cube. The query result in figure 4, if materialized, would be such a *complete* aggregate. Relationally spoken, only different attribute combinations in the group-by clause are considered. The where-clause in the materialized view definition is always empty. The benefit of this approach is that query processing is relatively simple. Therefore, this is the only kind of aggregates commercial products support today.

However, materialized views created in that manner may contain much data that is never needed and therefore storage space is wasted. Even more important, in a distributed setting with inherent data fragmentation the handling of *partial aggregates* (the candidates in figure 4) becomes a necessary prerequisite. A data mart server has only parts of the whole data cube stored locally. In fact, that is the fun-

Final edited form was published in "International Symposium on Database Engineering and Applications (IDEAS'98). Cardiff, 1998". ISBN 0-8186-8307-4.
https://doi.org/10.1109/IDEAS.1998.694361

damental idea of having data marts in a distributed enterprise data warehouse. Thus, at least at the lower aggregation levels it does not even make sense to create aggregates with full, enterprise-wide scope.

The big drawback is clearly that query optimization in the presence of partial aggregates becomes more complex. A patchworking algorithm finding the least expensive aggregate partitions must be invoked (figure 4).

Data Allocation and Load Balancing. Directly related to the question of selecting aggregates in a distributed environment is the allocation problem. The system should not only manage the dynamic creation of partial aggregates, but also decide where to place them on a tactical basis. Moreover, the traditional method to increase local access performance in distributed environments by the use of replicas can and should also be applied in distributed OLAP.

Hence, redundancy in the distributed data warehouse can be distinguished into vertical redundancy involving the creation of summary data with a coarser granularity than the detailed raw data, and horizontal redundancy being introduced by the replication of aggregates (figure 5).

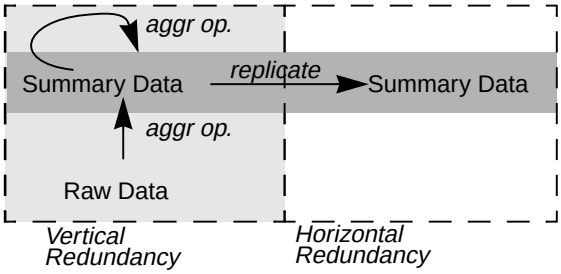


Fig. 5: Horizontal and Vertical Redundancy

In order to allow the system to adapt to changing user requirements, there should be no fixed locations for any kind of redundant data fragments. Aggregate data should be close to sites where they are needed in order to minimize the communication costs. In order to increase parallel query execution, data fragments should be distributed on several data mart servers, especially if communication cost is low. Redundant fragments must dynamically be created and dropped according to the changing user behavior.

Thus, a flexible adapting migration and replication policy is essential for a data based load balancing at the data mart servers. Involved in the process of choosing and allocating aggregates are not only access characteristics but also communication cost, maintenance cost for redundant data and additional storage cost.

Differences to traditional distributed DBMS. Although the fragmentation and allocation problematic is well known from traditional distributed database systems, there are several specifics for distributed data warehouses.

The fragmentation of the raw data is already fix due to the definition of the data mart. Usually, there is not much of a choice here. The problems as well as the performance potential of distributed data warehouses arise from the mentioned techniques of dynamic aggregate management and dynamic data allocation. These techniques were not previously applicable because redundancy in transaction-oriented systems always requires expensive synchronization mechanisms.

The query optimizer of a distributed OLAP system first needs to be aware of the aggregations which would be helpful in order to speed up query processing. If this information is known it must invoke a patchworking algorithm finding the least expensive aggregate partitions (figure 4). The choice how to assemble the query result may vary from query to query because the underlying redundant fragments may be dropped for the selection of new ones by the resource management and the system and network load may change.

An important point is that the determination of candidates as well as the patchworking itself, which includes the comparison of selection predicates, is only feasible in the context of a multidimensional data model. In contrast to the simple relational data model, selections (slices) can be defined by classification nodes (see section 4.2). Predicates defined in that manner contain much more semantics as a simple relational selection predicate. If one would allow arbitrary predicates for the definition of fragments, patchworking would not be possible.

3.2 Quality of Service Specification

Another point greatly distinguishes the distributed OLAP scenario from traditional distributed databases. In the analytical context general figures are of interest and not individual entities. It is often the case that approximate numbers are sufficient if they can be delivered faster. Thus, it is feasible to trade correctness for performance. Some factors characterizing correctness are accuracy, actuality and consistency.

Accuracy. One approach to relaxing the accuracy of the reports is offering summary data that is either aggregated at a higher level than requested or offering only parts of the requested information. For example, if a query requesting sales figures for each city in Germany would take hours, an answer containing sales figures for each county or even only for South Germany might be enough for the moment. An algorithm dealing with these issues is given in [Dyre96].

Actuality. Absolute actuality in a worldwide distributed enterprise data warehouse is an expensive requirement. If each data mart is updated once a day for example, it means for the distributed data warehouse that it is updated every

Final edited form was published in "International Symposium on Database Engineering and Applications (IDEAS'98). Cardiff, 1998". ISBN 0-8186-8307-4.
https://doi.org/10.1109/IDEAS.1998.694361

few hours. Therefore, the maintenance cost for summary data can increases dramatically. But getting the most actual numbers may not be necessary for the analyst. Hence, a certain degree of staleness can be acceptable allowing redundant data to be updated or dropped gradually [Lenz96].

Consistency. The consistency of the reports generated in a session should be another adjustable factor. In general, it is desirable that a sequence of drill-down and roll-up operations relies on the same base figures. However, this requirement might not be crucial, if the analyst just wants to get an overview over some figures. Hence, this condition as well might be relaxed according to the analysts needs. Note, that generating consistent reports can be very expensive, because all reports in a session must have the same actuality.

Especially in a large distributed environment where communication costs can not be neglected, these factors are critical for reasonable overall performance. A little inaccurate, a little stale or a little inconsistent data may be sufficient for the analyst, if he just gets the figures in a few seconds rather than minutes.

4 The Distributed OLAP system CUBESTAR

This section focuses on the architecture and the principles for query processing and resource management of the prototypical distributed OLAP system CUBESTAR. In order to deal with the complexity of the issues we borrowed many ideas from the Mariposa system ([SDK+94]) and applied them to our special application domain.

4.1 The Architecture

The general architecture of CUBESTAR consists of a three tier architecture as shown in figure 6. At the *client layer* are the end user tools with reporting facilities. Queries are issued in the Cube-Query-Language (CQL, [BaLe97], figure 7), specifically supporting the multidimensional data model.

The heart of the architecture is the *middleware layer* hiding the details of data distribution. Its main task is to translate a user query and generate an optimized distributed query execution plan. For distributed query processing and aggregate management we apply a microeconomic paradigm. A site tries to maximize revenues earned for taking part in query processing. In order to compete, the site has to maintain a set of redundant data according to the current market needs. The middleware layer also implements a global classification information service providing a globally consistent view of the multidimensional data model.

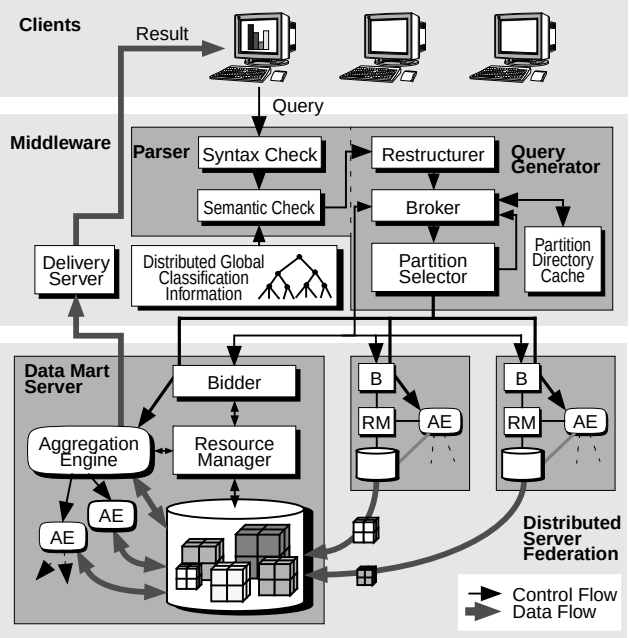


Fig. 6: Architecture of CUBESTAR

The third layer (*server layer*) consists of a federation of data mart servers. Each data mart server controls the locally available set of data partitions and the parallel query execution at its machine.

4.2 The Data Model

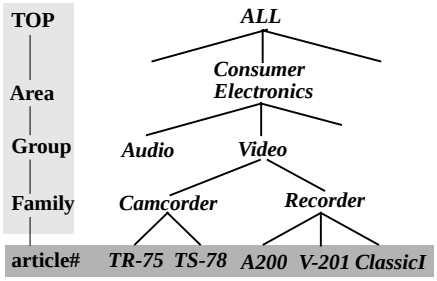
CUBESTAR consequently applies the multidimensional data model at all levels of query processing and aggregate management. Only issues of physical data access are managed by the underlying relational database engines.

Since the main operation in OLAP is the application of aggregation operations, classification hierarchies can be used to define groups for the application of aggregation operations (figure 7). For example, the typical OLAP-query of figure 7 specified in CQL asks for the sales figures at the granularity of product families subsumed by video equipment (= camcorders and recorders).

Multidimensional Objects. In analogy to the 'cubette'-approach of [Dyre96], a query may be seen as a multidimensional subcube, called *Multidimensional Object* (MO), basically described by three components [Lehn98]:

- A specification of the original cell and the applied aggregation operator (SUM(Sales))
- A multidimensional scope specification consisting of a tuple of classification nodes (<P.Group = 'Video', G.Country = 'Germany', T.Year = '1997'>)
- A data granularity specification consisting of a tuple of categories (<P.Family, G.Top, T.Top>)

Final edited form was published in "International Symposium on Database Engineering and Applications (IDEAS'98). Cardiff, 1998". ISBN 0-8186-8307-4.
https://doi.org/10.1109/IDEAS.1998.694361




```
SELECT SUM(SALES)
FROM Product P, Geography G, Time T
WHERE P.Group = 'Video',
        G.Country = 'Germany',
        T.Year = '1997'
UPTO P.Family
```

Fig. 7: A Sample Classification Hierarchy and a Sample Query in CQL

Within the distributed multidimensional OLAP environment, multidimensional objects reflect the basic units for distributed storage management and query processing. Therefore, different types of MOs with regard to their generation in the distributed environment are considered.

The raw or base data partitions at each data mart are called *Base Multidimensional Objects* (BMO). Thus, BMOs represent data of the finest granularity. BMOs are not allowed to migrate but are assigned a unique home site, where they are uploaded by the data mart loading tools. Consequently, they are per definition up-to-date.

From the BMOs any number of MOs can be derived. In a relational sense these derived MOs are materialized views. However, because of the definition of MOs, the questions whether one MO is contained in, intersected by or can be derived from another MO is easily decidable. According to the requirements of the system MOs are allowed to migrate or be replicated freely.

4.3 Microeconomics

An elegant solution to cope with the complexity of dynamic aggregate management and load balancing is to put all issues related to shared resources into a microeconomic framework ([SDK+94]). In an economy where every site tries to selfishly maximize its utility there is no need for a central coordinator; the decision process is inherently decentralized. Moreover, the competition for orders and profits allows the system to dynamically adapt to market demands, i.e. query behavior and system load as well.

In contrast to Mariposa where demand and supply regulate the prices, for the sake of simplicity a *uniform billing scheme* is applied to each executed query having fixed rates for the aggregation efforts. The value of a query result is computed by a weighted combination of the following factors:

- *Actual size of the raw data MOs required for the computation* (number of tuples).
  Taking the size of the raw data is reasonable, since requests for aggregates of huge fact tables should be more expensive than those of smaller ones. Moreover, this decision favors the storage of higher aggregates, since they need little space and yield high profits.
- *Actual size of the query result* (number of tuples).
  This term simply values the additional overhead for storing large aggregates.
- *Aggregation time*.
  This is the time spent from issuing the query to delivering the result, i.e. the local query processing time. Higher aggregation time results in a lower price, thus favoring fast data mart servers.
- *Transfer time*.
  The computation may require the shipping of intermediate MOs from one data mart server to another.

4.4 Query Processing

A query in CUBESTAR is basically a multidimensional object specified by a CQL query like the one in figure 7. Affiliated with each query is a user defined quality of service specification consisting of a limitation to the query processing time and an indicator for the requested accuracy of the data. Thus, the user has the choice to trade speed for actuality.

The query is issued to a broker which tries to perform the query in the best way possible on behalf of the user. Since all users are treated equally and prices are regulated, a query is not assigned a budget. The brokers objective is simply to get the best service available for the users request, i.e. to minimize query execution time under the quality of service constraints. After query processing, the broker pays the price according to the billing scheme to the participating data mart servers. Note, that in order to support user priorities it would make sense to limit the users budget resulting in a second objective, price, to query optimization.

Query processing basically proceeds in the three steps query preparation, query optimization, and query execution.

In the query preparation phase, the query string is handed over to the parser, where it is checked for syntactical and semantic correctness. The result of this step is a tree-like representation of the query with the queried MO at the top, intermediate results at the inner nodes, and raw data at the leaves.

Final edited form was published in "International Symposium on Database Engineering and Applications (IDEAS'98). Cardiff, 1998". ISBN 0-8186-8307-4.
https://doi.org/10.1109/IDEAS.1998.694361

For the generation of a distributed query execution plan the current distribution of the aggregates must be taken into account. Note, that due to the dynamic aggregation management the existence and location of aggregates and replicates changes frequently. Therefore, the broker may issue a broadcast request to the data servers in order to obtain the necessary information. This kind of meta data is cached in the partition directory cache in order to avoid too many broadcasts. Using this method, it is possible that invalid meta data are used. In this case, there are two possibilities, either force a data mart server to the recomputation of dropped aggregates or submit a new broadcast.

At the data mart server side, the bidder component receiving the request checks for those locally available MOs which can contribute to the query. Each bid from a data mart server includes the following data

- the identifier of the data mart server,
- a set of tuples { (M_i, T_i) } each of which contains the description of an offered MO in the repository and the time the site expects to aggregate that MO to the granularity of the query M,
- an indicator for the available processing power the site is willing to reserve for the query depending on the system load,
- the free space the site is willing to reserve for intermediate results possibly from other data mart servers.

The query optimizer's task is now to select from the set of multidimensional objects offered by the bidding sites those, which require the least aggregation and transportation efforts using a patchworking algorithm. To estimate communication costs, average network throughputs between the sites are used.

The result of this step is a distributed query execution plan, which is basically a tree with physically existing MOs at the leaves, the target MO at the top, and intermediate results at the inner nodes. Attached to each node is an operation, an estimated processing time for the operation, and the location for the execution of the operation denoting the bidders that won the competition.

In the query execution step, the bidders are informed whether their offer was accepted or not. The notification includes the complete query execution plan allowing the sites to understand the decision of the broker and draw conclusions for their future resource management.

If all MOs necessary for processing have arrived at the site, the local query execution is initiated by an aggregation engine. If subqueries can locally be executed in parallel, the aggregation engine may spawn new aggregation engines running in parallel threads. After the aggregation engine completed its task, in the last step the resulting MO is sent to the target location.

4.5 Storage Management

So far, the existence of many redundant multidimensional objects was assumed. This subsection focuses on when, how, and where these MOs are generated. In order to deal with the complexity of the issues of section 3.1, this problem is handled by the application of microeconomics.

The resource manager at a data server aims towards maximal usage of its local resources in terms of revenues. Therefore, a site always tries to expand its market share by cleverly creating and dropping aggregates and replicas. In order to estimate current demands the resource manager maintains a query history list containing those requests the site was rejected to answer, either because it did not have any suitable MOs or other bidders were preferred. The list entries include for each query the actually selected MOs with their locations and revenues. Based on this information the resource manager can decide to obtain new MOs in order to underbid its competitors in the future. If the new MO is to be acquired from a remote location, the resource manager acts as a client itself, and therefore also has to pay for the MO from its own account. In this case, it is doing an investment.

The other way to use the storage space is to offer it for query processing. If MOs from different locations must be coalesced, one site will receive the other parts. Note, that it is very lucrative to receive other pieces of MOs fitting to locally stored ones, because next time a larger MO can be offered.

In fact, using the intermediate results of the query processing is the most convenient way to get new MOs also in the local case. Each of these MOs is checked by the resource manager for usefulness. Basically, storing MOs that were already needed is working like a cache, with a microeconomic heuristics for replacement.

5 Summary and Future Work

In the first part of this paper we made clear the drawbacks of the centralized data warehouse and the non-integrated data mart approach. In order to circumvent these drawbacks and combine the benefits, integrated distributed enterprise data warehouses were presented as a possible solution. The on-line exploration of distributed data warehouses or distributed OLAP is related to many complex issues like distributed query optimization, meta data management and security policy handling, each of which demanding innovative solutions. We characterized some possibilities for achieving query performance by dynamic and partial aggregation, data based load balancing and the specification of the quality of service. Finally, we discussed the architectural framework of the prototypical distributed OLAP system CUBESTAR.

Final edited form was published in "International Symposium on Database Engineering and Applications (IDEAS'98). Cardiff, 1998". ISBN 0-8186-8307-4.
https://doi.org/10.1109/IDEAS.1998.694361

We consider sophisticated aggregation management a major research issue. This covers algorithms for using distributed partial aggregations and deciding which aggregate combinations yield the highest benefit and are the best to be materialized. A reasonable solution offering the needed flexibility is the application of a microeconomic framework as in the Mariposa system ([SDK+94]). We admit, that this last statement needs to be proven by the implementation.

References

BaLe97 Bauer, A.; Lehner, W.: The Cube-Query-Language for Multidimensional Statistical and Scientific Database Systems, in: *5th International Conference on Database Systems For Advanced Applications* (DASFAA'97, Melbourne, Australia, April 1-4, 1997), pp. 263-272

BaPT97 Baralis, E.; Paraboschi, S.; Teniente, E.: Materialized View Selection in a Multidimensional Database, in: *23rd International Conference on Very Large Data Bases* (VLDB'97, Athens, Greece, 1997)

CoCS93 Codd, E.F.; Codd, S.B.; Salley, C.T.: *Providing OLAP (On-line Analytical Processing) to User Analysts: An IT Mandate*, White Paper, Arbor Software Corporation, 1993

Cogn97 Cognos Corporation: Distributed OLAP, White Paper, http://www.cognos.com, 1997

Cube97 The CUBESTAR Project: http://www6.informatik.uni-erlangen.de/research/cubestar.html

Dyre96 Dyreson, C.: Information Retrieval from an Incomplete Data Cube, in: *22th International Conference on Very Large Data Bases*, (VLDB'96, Mumbai, India, 1996)

Ecke97 Eckerson, W.: Building and Managing Data Marts, Patricia Seybold Group Report for Informatica, http://www.informatica.com, 1997

GHRU97 Gupta, H.; Harinarayan, V.; Rajaraman, A.; Ullman, J.D.: Index Selection for OLAP, in: *13th International Conference on Data Engineering*, (ICDE'97, Birmingham, UK, April 7-11, 1997)

Gupt97 Gupta, H.: Selection of Views to Materialize in a Data Warehouse, in: *6th International Conference on Database Theory* (ICDT'97, Delphi, Greece, Jan 8-10, 1997), pp. 98-112

HaRU96 Harinarayan, V.; Rajaraman, A.; Ullman, J.D.: Implementing Data Cubes Efficiently, in: *25th International Conference on Management of Data*, (SIGMOD96, Montreal, Quebec, Canada, June 4-6, 1996), pp. 205-216

HP97 N.N.: *HP Intelligent Warehouse*, White Paper, Hewlett Packard, http://www.hp.com, 1997

Inmo92 Inmon, W.H.: *Building the Data Warehouse*, John Wiley, 1992

Info97 N.N.: *Enterprise-Scalable Data Marts: A New Strategy for Building and Deploying Fast, Scalable Data Warehousing Systems*, White Paper, Informatica, http://www.informatica.com, 1997

Lehn98 Lehner, W.: Modeling Large Scale OLAP Scenarios, in: *6th International Conference on Extending Database Technology*, (EDBT'98, Valencia, Spain, March 23-27, 1998)

Lenz96 Lenz, R.: Adaptive Distributed Data Management with Weak Consistent Replicated Data. In: *Proceedings of the 11th annual Symposium on Applied Computing (SAC'96)*, Philadelphia, 1996

LeRT96 Lehner, W.; Ruf, T.; Teschke, M.: Improving Query Response Time in Scientific Databases Using Data Aggregation, in: *7th International Conference and Workshop on Database and Expert Systems Applications* (DEXA'96, Zurich, Switzerland, Sept. 9-10, 1996)

SDK+94 Stonebraker, M.; Devine, R.; Kornacker, M.; Litwin, W.; Pfeffer, A.; Sah, A.; Staelin, C.: An Economic Paradigm for Query Processing and Data Migration in Mariposa, in: *Proceedings of 3rd International Conference on Parallel and Distributed Information Systems* (PDIS'94, Austin, TX., Sept. 28-30), 1994, pp. 58-67

ThSe97 Theodoratos, D.; Sellis, T.: Data Warehouse Configuration, in: *23th International Conference on Very Large Data Bases*, (VLDB'97, Athens, Greece, 1997)

WeCa96 Wells, D.; Carnelley, P.: Ovum evaluates: The Data Warehouse, Ovum Ltd., London, 1996