Advanced Implementation Techniques for Scientific Data Warehouses

Torben Bach Pedersen Christian S. Jensen

Department of Computer Science, Aalborg University
Fredrik Bajers Vej 7E, DK–9220 Aalborg Ø, Denmark,
{tbp,csj}@cs.auc.dk

Abstract

Data warehouses using a *multidimensional* view of data have become very popular in both business and science in recent years. Data warehouses for scientific purposes such as medicine and bio-chemistry[1] pose several great challenges to existing data warehouse technology. Data warehouses usually use pre-aggregated data to ensure fast query response. However, pre-aggregation cannot be used in practice if the dimension structures or the relationships between facts and dimensions are irregular. A technique for overcoming this limitation and some experimental results are presented. Queries over scientific data warehouses often need to reference data that is external to the data warehouse, e.g., data that is too complex to be handled by current data warehouse technology, data that is "owned" by other organizations, or data that is updated frequently. An example of this are the public genome databases such as Swissprot. This paper presents a federation architecture that allows the integration of multidimensional warehouse data with complex external data.

1 Introduction

Data Warehousing (DW) and On-Line Analytical Processing (OLAP) systems based on a dimensional view of data are being used increasingly in traditional business applications as well as in applications such as health care and bio-chemistry for the purpose of analyzing very large amounts of data. The use of DW and OLAP systems for scientific purposes raises several new challenges to the traditional technology [9]. This paper describes two of these challenges, both of which are concerned with implementation aspects. The first is the optimal use of pre-aggregated data for improved query performance even when the data structures are irregular, while the second is the integration of multidimensional OLAP databases with complex external data. Other challenges are related to the conceptual and logical design of scientific data warehouses, including modeling and querying complex multidimensional data [12, 15] and handling imprecise data [10, 15]. However, these challenges are beyond the scope of this paper.

In order to improve query performance, modern OLAP systems [19] use a technique known as *practical pre-aggregation*, where combinations of aggregate queries are materialized *selectively* and re-used when computing other aggregates; full pre-aggregation, where all combinations of aggregates are materialized, is infeasible, as it typically causes a blowup in storage requirements of 200–500 times the size of the raw data [8, 18]. Normally, practical pre-aggregation requires the dimension hierarchies to be regular, i.e., to be balanced trees, but this is quite often not the case in

[1] In this preliminary version of the paper, we will use medical cases for examples, but we hope to learn enough about biochemical data at the EML workshop to be able to revise the paper to use biochemical examples and thus show that our DW techniques are relevant for biochemical data management.

real-world systems. The technique presented here enables practical pre-aggregation even for irregular hierarchies. The details of the technique can be found elsewhere [11, 14]. We show how to achieve practical pre-aggregation through transformations of the dimensions and how the transformations can be accomplished transparently to the user. The technique enables the achievement of fast query response time while saving huge amounts of storage compared to current OLAP systems and techniques. The prototype implementation of TreeScape [14] demonstrates that these benefits may be achieved with standard technology.

OLAP systems are very popular for analyzing large amounts of data, as they provide both ease-of-use and good query performance for queries that aggregate large amounts of data. However, OLAP databases do not handle complex relationships in the data well, and it is hard to integrate OLAP data with external data. One approach to integration is physical integration of the data in one database, i.e., (physical) data warehousing [4], but physical integration is often not feasible, in which case a federated approach is desirable. In a previous paper [13], we have defined the theoretical framework for handling federations of OLAP and object databases. A prototype federation system, OLAP++, has been implemented [3]. In this paper we present the concepts of OLAP-object federations and discuss advantages of using the federation approach compared to physical integration of the data, based on our experiences with the prototype system. The federation approach allows users to easily pose OLAP queries that reference data residing in external databases, enabling flexible and fast integration of external data in OLAP systems without the need for prior physical integration.

2 Practical Pre-Aggregation for Irregular Hierarchies

This section describes a technique for using practical pre-aggregation even when the dimension hierarchies are irregular.

2.1 Normalizing Hierarchies

We use a small case study concerning patients and their diagnoses for illustrating the workings of the system. Diagnoses have three different levels of precision, depending on how accurate a patient's condition can be described. The most precise diagnoses are *low-level diagnoses*, which are grouped into *diagnosis families*, which, in turn, are grouped into *diagnosis groups*. The example data consists of 9 diagnoses and their hierarchical relationship, along with patient counts. The data can be seen in Table 1 and to the left in Figure 1.

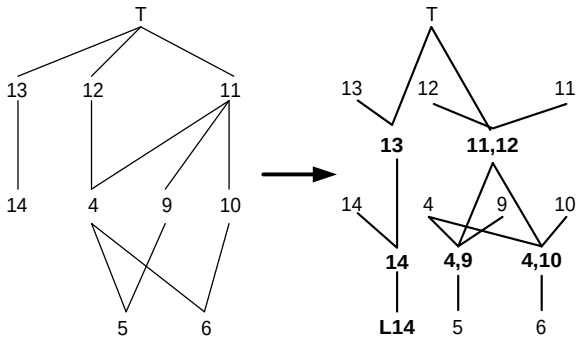

Figure 1: Dimension Transformations

ID	Text	Type
4	Diabetes during pregnancy	Family
5	Insulin dependent diabetes during pregnancy	Low-Level
6	Non insulin dependent diabetes during pregnancy	Low-Level
9	Insulin dependent diabetes	Family
10	Non insulin dependent diabetes	Family
11	Diabetes	Group
12	Pregnancy related	Group
13	Cancer	Group
14	Lung cancer	Family

Diagnosis

ParentID	ChildID
4	5
4	6
9	5
10	6
11	9
11	10
12	4
13	14

Grouping

DiagID	Count
5	1

Patient

Table 1: Case Study Tables

The hierarchy is irregular. It is unbalanced because the diagnosis "Lung cancer" (14) has no low-level diagnoses associated with it and non-strict because, e.g., diagnosis 4 ("Diabetes during pregnancy") has several parents.

With this data, problems occur when pre-aggregated data at lower levels is used to compute new values at higher levels. For example, if we pre-aggregate the counts of patients at the low-level diagnosis level and want to aggregate to the diagnosis family level, we cannot deduce what the value should be for "Lung Cancer" (14). If we pre-aggregate at the diagnosis family level, patients with diagnoses 5 or 6 will be counted for both of the diagnoses 4 and 9, and 4 and 10, respectively, leading to wrong results when we aggregate to the diagnosis group level.

Our solution to the problems with reusing aggregates is to render the hierarchies well-behaved by *normalizing* them. Informally, the normalization process introduces new *placeholder* values where the hierarchy is unbalanced, and introduces *fused* values that represent *sets of* parent values when child values have multiple parents. The result of normalizing the hierarchy in our example is given to the right in Figure 1. For example, value "**L14**" representing "Lung Cancer" at the low-level diagnosis level, and value "**4,9**" representing the set of diagnoses $\{4, 9\}$ are introduced by the normalization. In the figure, all values and links in boldface have been added by the normalization process, which is described in detail elsewhere [11].

The normalized hierarchy supports practical pre-aggregation. For example, it is possible to store counts of patients at the low-level diagnosis level, and then re-use these to compute the counts for diagnosis families and diagnosis groups. With the example data (one patient with diagnosis 5), this will only require the storage of the one value versus six values being required for *full* pre-aggregation (one value for low-level diagnosis 5, two values for diagnosis families 4 and 9, two values for diagnosis groups 11 and 12, and one value for $\top$, which represents the total for all diagnoses).

The example is somewhat indicative of the storage savings achieved within a single dimension. When several dimensions are combined, the total space saved (with respect to full pre-aggregation) is the product of the savings in each dimension, resulting in savings factors of 100 or more in practice. The savings occur because of *multidimensional sparseness* [8, 18], the phenomenon of the multidimensional space being very sparse for the lower levels in the dimensions, while quickly becoming more dense at higher levels. The query response time using the normalization approach will not be quite as fast as using full pre-aggregation, but will most likely be comparable, i.e.,

within an order of magnitude. This is much faster than computing the results from the base data, as would be required with *no* pre-aggregation. A study of the benefits of normalization are presented in Section 2.3.

2.2 TreeScape System Architecture

While the hierarchy transformations enable practical pre-aggregation, they also have the undesired side-effect of introducing new values into the hierarchies that are of little meaning to the users. Thus, the transformations should remain invisible to the users. This is achieved by working with two versions of each user-specified hierarchy and by using a query rewrite mechanism, as described in detail elsewhere [11, 14]. The overall system architecture is seen in Figure 2.

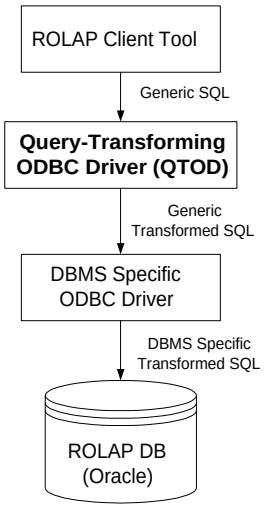

Figure 2: TreeScape System Architecture

The ROLAP client tool makes SQL requests to the ROLAP database, in this case the Oracle8 RDBMS, using the ODBC standard. We have implemented a special, query-transforming ODBC driver (QTOD) that, based on case-specific metadata, transforms the SQL requests into requests that hide the transformations from the users, returning the query results that the user would expect based on the original hierarchies. A transformed request is submitted to the OLAP DB using an RDBMS-specific ODBC driver. The QTOD component is common to all RDBMSs, so Oracle8 may be replaced by another RDBMS such as IBM DB2, Informix, or MS SQL Server. Any ROLAP tool may be used, making the solution quite general and flexible.

The prototype is based on an RDBMS (Oracle8) since RDBMSs are the most commonly used platform for Data Warehouse and OLAP applications. Additionally, the major RDBMSs now, like dedicated multidimensional DBMSes (MDDBs), use pre-aggregated data for faster query responses [20]. However, the approach could also be implemented using multidimensional technology, e.g., the Microsoft OLE DB for OLAP technology [7].

The transformation algorithms are implemented in Oracle's PL/SQL programming language. The transformations are relatively fast, taking at most a few minutes, even for large dimensions. Once the dimension hierarchies have been transformed, the QTOD transforms queries and results between the original and transformed hierarchies. The QTOD is a thin layer and adds very little overhead to queries. It is implemented using GNU Flex++/Bison++ scanner/parser generators and the MS Visual C++ compiler.

2.3 TreeScape Experimental Results

In this section, we briefly describe the results of comparing our technique to the two alternatives, namely *no* pre-aggregation, which gives very long query response times, and *full* pre-aggregation, which requires unrealistically large amounts of storage for pre-aggregated data. We assume that an answer can be fetched using 1 I/O in the optimal case, and that 1 I/O takes 10ms to perform.

The comparison has been done analytically using a combination of real and synthetic data. The dimension data is based on the British "Read Codes" diagnosis classification. The initial dimension hierarchy has 22570 values (diagnoses), while the transformed hierarchy has 51695 nodes, both have eight levels. The fact data has been synthetically generated to have 10% density per dimension, i.e., .01 density for two dimensions, .001 density for three dimensions, etc., which corresponds to real-world cases where the multidimensional space gets very sparse when the number of dimensions increase. For the our technique, materialized aggregates where chosen so that the average-case performance was at most ten times worse than the optimal performance obtained with full pre-aggregation.

We found that the benefit of our technique increased dramatically with the number of dimensions. For four dimensions, the average response time with no pre-aggregation is *57 years*, which clearly makes this alternative unsuable. Even with a speed increase of a factor of 1000 due to the use of parallel and sequential I/O, we still get an average response time of *20 days*. Full pre-aggregation, on the other hand, requires *41 times* the size of the base data, equal to 1.1 *petabytes* of storage for four dimensions, which puts it far beyond the capacity of current disk systems. We note that the problems with these techniques will only get worse for more dimensions. In comparison, our technique achieves an average response time of *98 ms* using only 80% more storage than the base data, making it very attractive.

3 Federations of OLAP and Object Databases

In this section, we present the concepts used in OLAP-object federations and argue why federations are often superior to physical integration of the data.

3.1 Federation Concepts

OLAP systems use a multidimensional view of data that typically categorize data as being measurable facts (measures) and dimensions, which are mostly textual and characterize the facts. Dimensions are structured using categories (levels) that correspond to the required levels of detail. Object systems use the familiar concepts of classes, attributes, and relationships between classes. A federation between an OLAP and an object DB is defined by specifying a link between a category in the OLAP DB and a class in the object DB.

The case study concerns data in three different databases, each managed by a separate organization. Each databases serves a different purpose, but the databases contain related data. A graphical illustration of the databases using UML notation [16] is seen in Figure 3.

The three databases were built and are used separately, which explains the differences in their information contents. We want to use them together, to exploit information from the demographic and epidemiology databases in OLAP queries against the admissions database.

The measured fact in the OLAP DB is the total number of admissions The facts are characterized by a Hospital dimension and a Diagnosis dimension. The Hospital dimension has Hospital and State categories. The Diagnosis dimension has the categories Diagnosis and Diagnosis Group. The demographic object DB has State, City, and Mayor classes, while the epidemiology object

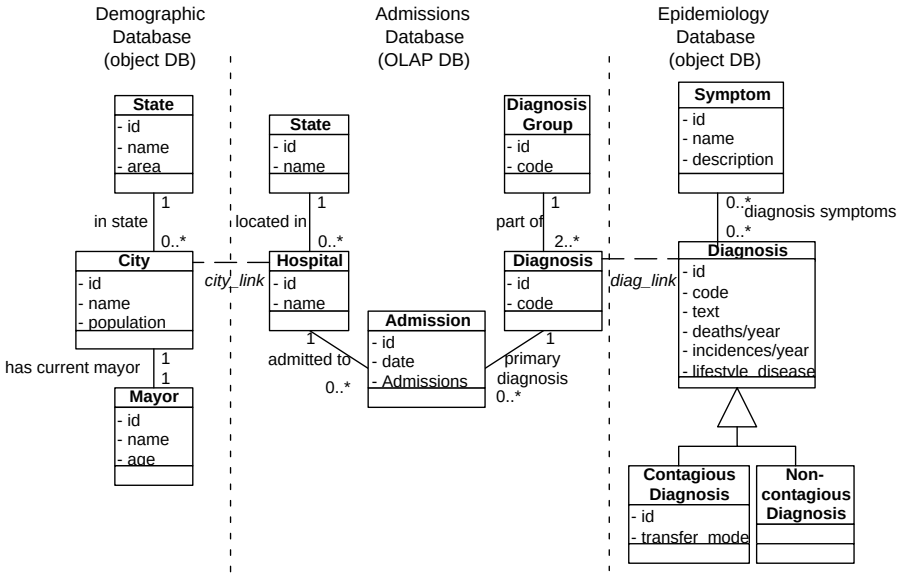

Figure 3: UML Schema of Federation Case Study

DB has Symptom, Diagnosis, Contagious Diagnosis, and Non-Contagious Diagnosis classes. The "diag_link" link links the Diagnosis category in the OLAP part to the Diagnosis class in the object part as indicated by the dotted lines. The "city_link" link links the Hospital category to the City class. Below is an example query for the schema in the SumQL++ federation language [13].

```
SELECT Admissions BY_CATEGORY Hospital,Diagnosis
FROM Admissions
WHERE Hospital.city_link.population > 1,000,000
        AND Diagnosis.diag_link.symptoms.name = "Cough"
```

The above query gets the total number of admissions for a 2-dimensional cross product of hospital and diagnosis where the cities in which the hospitals are located have a population of more than 1 million and the diagnoses have cough as a symptom. This query uses the links "diag_link" and "city_link" to go from the OLAP schema to the object schema.

3.2 OLAP++ System Architecture

The overall architecture of the OLAP++ system is seen in Figure 4. The object part of the system is based on the OPM tools [2] that implements the Object Data Management Group (ODMG) object data model [1] and the Object Query Language (OQL) [1] on top of a relational DBMS, in this case the ORACLE RDBMS. The OLAP part of the system is based on Microsoft's SQL Server OLAP Services using the Multi-Dimensional eXpressions (MDX) [7] query language.

When a SumQL++ query is received by the Federation Coordinator (FC), it is first parsed to identify the measures, categories, links, classes and attributes referenced in the query. Based on this, the FC then queries the metadata to get information about which databases the object data and the OLAP data reside in and which categories are linked to which classes. Based on the object parts of the query, the FC then sends OQL queries to the object databases to retrieve the data for which the particular conditions holds true. This data is then put into a "pure" SumQL statement

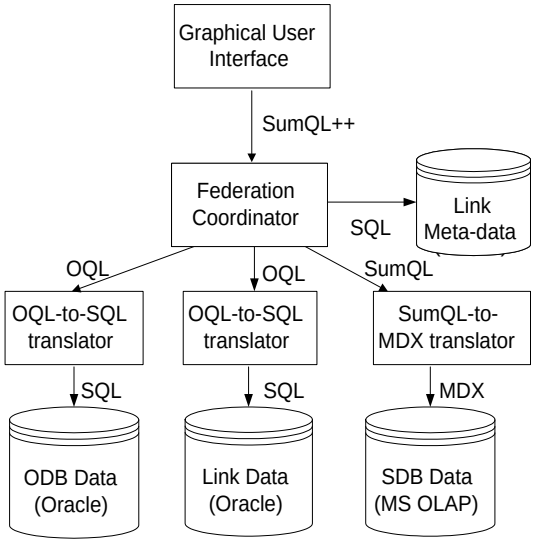

Figure 4: OLAP++ Architecture

(i.e. without object references) as a list of category values. This SumQL statement is then sent to the OLAP database layer to retrieve the desired measures, grouped by the requested categories. The SumQL statement is translated into MDX by a separate layer, the ”SumQL-to-MDX translator”, and the data returned from OLAP Services is returned to the FC. The reason for using the intermediate SumQL statements is to isolate the implementation of the OLAP data from the FC. As an another alternative, we have also implemented a translator into SQL statements against a ”star schema” relational database design [5]. The system is able to support a good query performance even for large databases while making it possible to integrate existing OLAP data with external data in object databases in a flexible way that can adapt quickly to changing query needs.

3.3 Advantages of Federation

Many reasons exist for preferring federating OLAP DBs with external DBs, as opposed to physically integrating these. The generic arguments for federation include leveraging existing technology, accessing the most current information, and allowing the autonomous existence of the systems being federated. These arguments also apply here, so we concentrate on the advantages specific to OLAP and object databases.

In many situations, OLAP DBs only contain abstract summary data and not the base data from which the summary data is derived. For example, summary databases provided by the Ministry of Health do not permit access to base data, e.g., diagnosis information, which is considered *too sensitive* for general disclosure. The same situation arises in census databases, where only highlevel information is disclosed publicly. In these cases, the federation approach allows superusers to access the base data to answer certain queries without having to put sensitive data into the OLAP DB itself.

Federating OLAP and object DBs enables a *simple* and *special-purpose OLAP system*. An OLAP DB needs not contain all objects, attributes, and relationships in the base database, but only the elements relevant to summary querying. This is attractive, as capturing all information in the OLAP DB unnecessarily impedes casual use of the OLAP system. Indeed, most OLAP systems do not have the necessary facilities to support this extra information. The federated approach allows the OLAP DB to remain simple, while still allowing access to relevant external data. When OLAP data

resides in a special-purpose OLAP system, we cannot use existing database middleware to access it, leading to a need for technology that enables federations of OLAP DBs and external DBs.

It is possible to obtain *better performance* when performing aggregation querying in an OLAP-type system rather than in a general-purpose DBMS. OLAP systems typically employ specialized, performance enhancing techniques, such as multidimensional storage [19] and pre-aggregation [4, 19]. This performance gain can often outweigh the performance loss due to the fact that the data is not physically integrated, meaning that a federated system can have comparable (or even better) performance without the limitations incurred by physical integration. Our experiments have shown that the federation approach performs *up to 40 times faster* than executing the queries on a relational database where the data has been physically integrated. Next, it is *easier to formulate aggregation queries* in an OLAP system than in a general (relational or object) DBMS. This is because an OLAP query language is designed exclusively for expressing aggregation queries over categories, taking advantage of, e.g., the automatic aggregation implied by the OLAP DB semantics. Even when extending an OLAP language to access object data, it is easier to pose aggregation queries in the extended language than in a general database query language such as OQL or SQL.

An OLAP system may support the formulation of aggregation queries that return *correct, or meaningful, query results*. When building an OLAP DB, the data may be shaped in order to satisfy summarizability conditions [6]. Briefly, an aggregation query satisfies summarizability conditions if the query result is correct w.r.t. the real world. For example, summarizing the populations over cities to get summaries for states will produce incorrect results if the populations in towns and farms outside cities are not accounted for. As another example, if patients have several diseases, and we summarize over all diseases to get the total number of sick people, we will get the wrong result as some patients are counted more than once. We may enrich an OLAP system with information that enables the system to ensure correctness. For example, we may specify that inventory levels should not be added across time [6] or that patient counts for diseases should not be added. In a general-purpose DBMS, no mechanisms for ensuring correct aggregation results are available.

The federated approach offers additional *flexibility* when query requirements change. OLAP DBs may be huge, and therefore rebuilding them may be time consuming. Updates to an OLAP DB, e.g., adding new types of information, may require a total or partial rebuild of the database. Because of the rebuild time, a rebuild of the OLAP DB will most likely be refused totally or postponed to the next scheduled rebuild, e.g., once a week or once a month. In contrast, a new link can be added in a matter of minutes, yielding much faster access to newly required information. This allows rapid prototyping of OLAP systems. The above reasoning suggests that in many cases, it is advantageous to logically federate OLAP and object databases instead of performing physical integration.

4 Conclusion

Data warehouses using a *multidimensional* view of data are increasingly used for business as well as scientific purposes such as medicine and bio-chemistry. However, the application of DW technology to the scientific domain poses several great challenges to existing DW technology. This paper presented techniques that aimed to solve two such challenges, namely the optimal use of pre-computed aggregates over irregular hierarchies and the integration of multidimensional data with complex external data.

The first technique employed the process of *normalizing* irregular dimension hierarchies in order to enable practical pre-aggregation. When the dimension hierarchies are irregular, we showed that this technique is far superior to the two alternatives, namely *no* or *full* pre-aggregation.

The second technique used a *federation* approach to logically combine multidimensional OLAP

data with complex external data stored in object databases. The paper showed that the federation approach was often superior, in terms of flexibility, correct query results, and performance, to physical integration of the data.

References

[1] R. G. G. Cattell et al. (Eds). *The Object Database Standard: ODMG 2.0*. Morgan Kaufmann, 1997.

[2] I-M. A. Chen and V. M. Markowitz. An Overview of the Object-Protocol Model (OPM) and OPM Data Management Tools. *Information Systems* 20(5): 393–418 (1995).

[3] J. Gu, T. B. Pedersen, and A. Shoshani. OLAP++: Powerful and Easy-to-Use Federations of OLAP and Object Databases. In *Proceedings of the Twentysixth International Conference on Very Large data Bases*, demo track, 2000.

[4] W. H. Inmon. *Building The Data Warehouse*, 2nd Edition. Wiley, 1996

[5] R. Kimball. *The Data Warehouse Toolkit*. Wiley Computer Publishing, 1996.

[6] H. Lenz and A. Shoshani. Summarizability in OLAP and Statistical Databases. In *Proceedings of the Ninth International Conference on Scientific and Statistical Databases*, pp. 39–48, 1997.

[7] Microsoft Corporation. OLE DB for OLAP Version 1.0 Specification. Microsoft Technical Document, 1998.

[8] The OLAP Report. *Database Explosion*. <www.olapreport.com/DatabaseExplosion.htm>. Current as of February 18, 2000.

[9] T. B. Pedersen and C. S. Jensen. Research Issues in Clinical Data Warehousing. In *Proceedings of the Tenth International Conference on Statistical and Scientific Database Management*, pp. 43–52, 1998.

[10] T. B. Pedersen, C. S. Jensen, and C. E. Dyreson. Supporting Imprecision in Multidimensional Databases Using Granularities. In *Proceedings of the Eleventh International Conference on Statistical and Scientific Database Management*, pp. 90–101, 1999.

[11] T. B. Pedersen, C. S. Jensen, and C. E. Dyreson. Extending PractiPre-Aggregation in On-Line Analytical Processing. In *Proceedings of the Twentyfifth International Conference on Very Large Data Bases*, pp. 663–674, 1999. Extended version available as TR R-99-5004, Dept. of Comp. Sci. Aalborg University, <www.cs.auc.dk/∼tbp/articles/R995004.ps>, 1999.

[12] T. B. Pedersen and C. S. Jensen. Multidimensional Data Modeling for Complex Data. In *Proceedings of the Fifteenth International Conference on Data Engineering*,1999. Extended version available as TimeCenter Technical Report TR-37, <www.cs.auc.dk/TimeCenter>, 1998.

[13] T. B.Pedersen, A. Shoshani, J. Gu, and C. S. Jensen. Extending OLAP Querying to External Object Databases. In *Proceedings of the Ninth International Conference on Information and Knowledge Management*, 2000.

[14] T. B. Pedersen, C. S. Jensen, and C. E. Dyreson. The TreeScape System: Reuse of Pre-Computed Aggregates over Irregular OLAP Hierarchies. In *Proceedings of the Twenty-Sixth International Conference on Very Large Databases*, demo track, 2000.

[15] T. B. Pedersen, C. S. Jensen, and C. E. Dyreson. A Foundation for Capturing and Querying Complex Multidimensional Data. To appear in *Information Systems* - Special Issue: *Data Warehousing*, 2001.

[16] Rational Corporation. UML 1.1 Notation Guide. URL: <www.rational.com/uml/resources/documentation/notation/index.hmpl>. Current as of Sep 7, 2000.

[17] National Health Service (NHS). *Read Codes version 3*. NHS, September 1999.

[18] A. Shukla et al. Storage Estimation for Multidimensional Aggregates in the Presence of Hierarchies. In *Proceedings of the Twenty-Second International Conference on Very Large Databases*, pp. 522–531, 1996.

[19] E. Thomsen. *OLAP Solutions*. Wiley, 1997.

[20] R. Winter. Databases: Back in the OLAP game. *Intelligent Enterprise Magazine*, 1(4):60–64, 1998.