[illegible]

A Note on Interfacing Object Warehouses and Mass Storage Systems for Data Mining Applications*

Robert L. Grossman

Magnify, Inc.
815 Garfield Street
Oak Park, IL 60304
Email: rlg@opr.com
Tel: 708-383-7002
Fax: 708-383-7084

University of Illinois at Chicago
Laboratory for Advanced Computing
851 South Morgan Street
Chicago, IL 60607
Email: grossman@uic.edu
Tel: 312-413-2176
Fax: 312-996-1491

Dave Northcutt
Magnify, Inc.
815 Garfield Street
Oak Park, IL 60304
Tel: 708-383-7002
Fax: 708-383-7084

Abstract

Data mining is the automatic discovery of patterns, associations, and anomalies in data sets. Data mining requires numerically and statistically intensive queries. Our assumption is that data mining requires a specialized data management infrastructure to support the aforementioned intensive queries, but because of the sizes of the data involved, this infrastructure is layered over a hierarchical storage system. In this paper, we discuss the architecture of a system which is layered for modularity, but exploits specialized lightweight services to maintain efficiency. Rather than use a full functioned database for example, we use light weight object services specialized for data mining. We propose using information repositories between layers so that components on either side of the layer can access information in the repositories to assist in making decisions about data layout, the caching and migration of data, the scheduling of queries, and related matters.

Introduction

Data mining is the automatic discovery of patterns, associations, and anomalies in data sets. The data mining of large data sets is a special challenge because the process requires numerically and statistically intensive queries on large amounts of data. Our assumption is that data mining requires a specialized data management infrastructure, but because of the sizes of the data involved, this infrastructure is layered over a hierarchical storage system. Our concern in this paper is an appropriate open, layered architecture to support this.

* This work was supported in part by the Massive Digital Data Systems Program.

A common layered architecture for this type of system is illustrated in Figure 1. There are three layers: the storage management layer, the data management layer, and the data mining and analysis layer. Unless these three layers coordinate how the data is physically laid out, how it is cached and migrated, and how it is prefetched, these layers can work at cross purposes and drastically impair the performance of the overall system.

The traditional approach forgoes the convenience and modularity of a layered approach for efficiency: with this approach, the data management system manages storage itself, while the data mining and data analysis applications manage the data themselves. In practice, this has meant that generally data mining applications simply work with flat data that fits into main memory. Of course, this data may be obtained by sampling large databases, but the point is that the data mining applications *themselves* work with small amounts of relatively simple data. This may be thought of as a sample-based approach to data mining.

In this paper, we are concerned with an alternative approach: the system is layered for modularity, but exploits specialized lightweight services to maintain efficiency. Rather than use a full functioned database for example, we use light weight object services specialized for data mining. With this approach, the data mining applications can work with large amounts of complex data. Another advantage of this approach is that the data management services can be used to manage the internal data structures required by the data mining algorithms. This may be thought of as a data-driven approach to data mining.

One of our specific concerns in this note is how the different layers can share information, especially in a heterogeneous environment. We propose using information repositories between layers so that components on either side of the layer can access information in the repositories to assist in making decisions about data layout, the caching and migration of data, the scheduling of queries, and related matters.

This proposal generalizes and extends the proposal in Brown et. al. [1] for providing a repository between a mass storage system and a relational database management system and is a refinement of the architecture described in Grossman [2] and [3] for a scaleable data mining system.

This work is preliminary. A fuller treatment is in preparation.

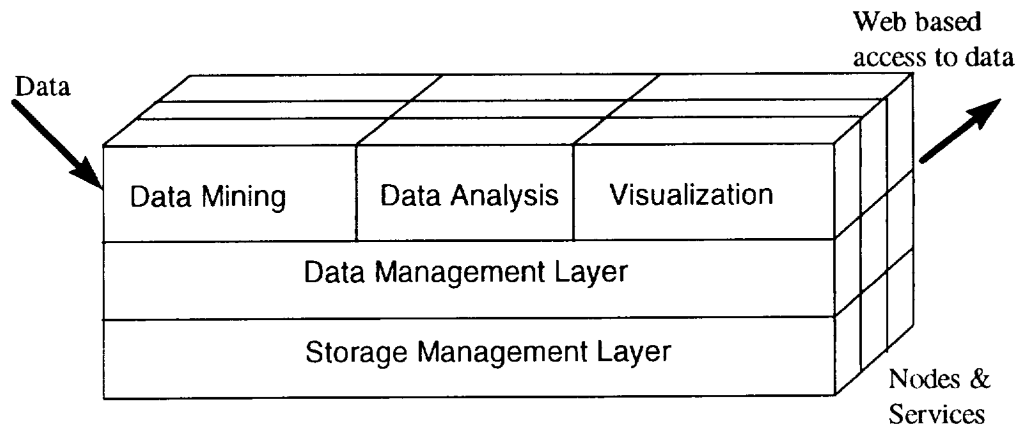

Figure 1. In a layered approach to data mining, rather than manage their own data, data mining applications use services from a data management layer, which in turn use storage services from a lower layer.

Background and Related Work

Broadly speaking, there are two relevant traditions: one system-based and one service-based. In the first, the essential question is how a database management system can interface to a storage management system. In the second, the essential question is what services are required so that data management, storage management, and application services can interoperate in an open network environment.

Relational database-mass storage system interfaces

Historically, databases have managed the storage of single disks; more recently, they have managed the storage of distributed disks. For some applications though much of the data is distributed on a storage hierarchy, including tape and other tertiary storage, which is managed by a mass storage system. One of the most important interfaces effecting performance is the interface between a relational database client and the mass storage system. A group at Lawrence Livermore National Laboratory has proposed an interface between a client of a relational database management system and a mass storage system Brown et. al. [1]. This interface which they call an Information Data Repository (IDR) would serve as the home for several relational tables, including: one for relational tables from the client database (called the bundle table), one for instances of the various components in the storage hierarchy (called the store table), one for mapping regular sub-components of bundles to stores (called the block table), and one for a list of pending requests for moving data between stores (called the movement table). In addition, the proposal [1] suggests using a standard relational database management system to manage the various tables in the IDR. The IDR would be external to both the database and the storage system and all interactions between the database and the mass storage system would be required to go through the IDR.

Light Weight Object Management Using Network Services

Another approach is to develop a data management system specifically designed for the mining and analysis of data. This type of system does not require the full functionality of a database, but instead is optimized to provide low overhead, high performance access to data which is read often, occasionally appended, but infrequently updated. In addition, data may be pre-computed and specialized indices may be provided. This can be thought of as providing specialized *lightweight* application specific data management services Grossman et. al. [4]; or alternatively, as providing an object warehouse specialized for data mining applications Grossman [3].

As usual with databases, with this approach there is a manager for physical collections of objects (called segments). In addition, to achieve scalability, physical collections of segments are themselves gathered into larger physical units called folios. There is also a folio manager which interacts with file and storage services, including mass storage systems. Just as the segment manager can query the folio manager, so can the mass storage system. The folio manager maintains a table of folios and their physical locations. In some sense, the folio manager can be viewed as the interface between a database and a (hierarchical) storage system. See [3] and [4] for more information about this approach.

Distributed Object Services

The Object Management Group's Common Object Request Broker Architecture (CORBA) is an industry standard for the development of distributed object-oriented applications across heterogeneous platforms. The OASIS environment developed at UCLA by Mesrobian et. al. [5] is an open environment for working with scientific information based upon CORBA. CORBA is optimized for working with relatively large-grained objects in heterogeneous environments in contrast to the use of lightweight data management and data warehousing described above. In some sense, CORBA is pessimistic about the physical layout of data and provides the infrastructure to support this in order to work in heterogeneous environments, while a lightweight approach is optimistic and only translates the physical format of data when necessary.

Requirements and Objectives

Our over all objective was to design an open system for data mining and data analysis which scales as the amount of data and the numerical complexity of the query increases. More specifically, we had the following requirements:

- *Large data sets.* Our most important goal was to support the mining and analysis of very large data sets, including data sets large enough to require multiple disks or tertiary storage.

- *Numerically intensive queries.* Our second most important goal was to provide very low overhead, high performance access to the data. In some sense databases are optimized to provide safe access to data which is expected to change; our goal was to provide high performance access to data which is relatively static.

- *Transparent access to data.* Because of the size of the data sets, much of the data is expected to be either on tertiary storage or on large arrays of disks. An important goal was to provide transparent access to the data, independent of its location or media type.

Architectural Description

Our architectural framework consists of a storage management layer, a data management layer, and an application layer consisting of clients of the data management services. We are primarily concerned with data mining and data analysis clients. Between each of the layers is a repository for information: a Storage Interface Repository (SIR) between the storage management and data management layers and a Data Interface Repository (DIR) between the data mining applications and the data management layer.

Data Interface Repository (DIR)

Traditionally, data mining has looked for patterns in small amounts of flat file-based data or sampled small amounts of data from relational databases using SQL queries. Data-driven data mining requires working with large amounts of complex data, much of which has to be warehoused because of performance considerations. The DIR has several roles, including:

- The data required for data mining and data analysis queries may be distributed in several data management systems, including data warehouses and operational and archival data management systems. The DIR provides a uniform interface for data mining and data analysis queries. The DIR maintains a list of logical data sets and the systems which are maintaining them.

- For performance reasons, some of the data for data mining applications may be warehoused, and specialized index and access structures may be provided. This requires periodically refreshing the data from the operational and archival databases. The DIR maintains the information required for this to take place.

- The DIR can also maintain the information for the optimization of data mining queries using information obtained from the results of previous queries.

Storage Interface Repository (SIR)

Data management systems by necessity divide the data they manage into regular sized extents. For example, access to file-based data is through blocks of equal length, while a common type of object-oriented database provides access to objects through extents of equal length called segments. These extents can then be managed by the data management systems themselves or by file or storage systems. In particular, they may be managed by hierarchical storage systems. The SIR has several roles, including:

- The demands upon extents imposed by the database management system are not necessarily those imposed by the hierarchical storage system. Not all extents are treated the same by the data management system: for example, some may contain directory or index information, which it would prefer remain pinned to secondary storage, even if infrequently accessed. The SIR provides a mechanism for a database and a hierarchical storage system to exchange information about desired movements of extents or sequences of extents.

- A database must be able to estimate the time to access data. If the physical management of the data is delegated to the hierarchical storage system, then the SIR must contain enough information so that the database can still make these estimates.

- To work with very large data sets, a hierarchy of extents, as described above, must be supported by the SIR. For example, for terabyte size data sets, there are simply too many segments to be managed directly by the database. Instead, it is important to group objects into segments, and segments into larger units.

The SIR discussed here is an extension of the IDR proposed in Brown et. al. [1].

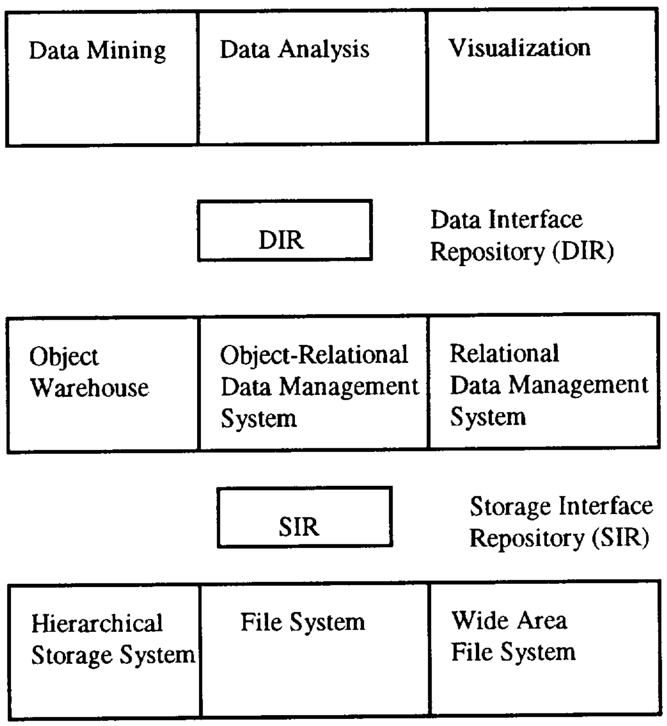
Data Mining
Data Analysis
Visualization
DIR
Data Interface Repository (DIR)
Object Warehouse
Object-Relational Data Management System
Relational Data Management System
SIR
Storage Interface Repository (SIR)
Hierarchical Storage System
File System
Wide Area File System

Figure 2. The role of the Data Interface Repository (DIR) and the Storage Interface Repository (SIR) is to maintain information so that services and applications in different layers can interoperate.

Discussion

In this section, we discuss some issues regarding the architecture.

- *Is the interface mandatory or advisory?* Systems can be built either way. If the interface is mandatory, then performance may suffer, since some of a component's essential services may have to be accessed externally. On the other hand, if the service is advisory, inefficiencies are likely and deadlocks are possible because different components accessing the service may make conflicting choices.

- *Is the interface part of one of the components or independent?* Traditionally, for example, the management of table information, block information, and the mapping from tables to blocks has been a component of the data management system. The role of the SIR is to provide this information through a separate service. Alternatively, the SIR could be incorporated into one of the layers and accessed from the other layer.

- *How should the DIR and SIR be implemented?* A variety of implementations are possible: The DIR and SIR could simply be implemented as a network service. Alternatively, a relational database can be used as proposed in Brown et. al. [1], or a CORBA Object Request Broker (ORB) could be used.

- *What is the granularity of access?* For this approach to succeed, it is important to be able to adjust the granularity of the objects referenced in the SIR and DIR so that performance is not adversely effected.

Status

This approach arose out of work with a system for data mining developed by Magnify, Inc. called PATTERN. PATTERN currently consists of beta versions of an object warehouse [3] and data mining modules for classification, prediction, and optimization Grossman et. al. [6]. A demonstration of the system mining and analyzing high energy physics data took place at Supercomputing 95. A performance evaluation of the system is currently being prepared and will appear elsewhere.

Currently, the SIR is part of the object warehouse and interfaces to the High Performance Storage System (HPPS) Teaff [7], while the functionality proposed by the DIR is currently shared between the different data mining modules.

Summary

In this paper, we propose a layered approach to a data mining system. Data mining applications exploit specialized data management services from a lower level, which in turn exploit specialized storage management services. We propose providing information repositories between each level so that services on either side can efficiently exchange information. To maintain performance, we use specialized lightweight data management services instead of a full functioned database, and adjust the granularity of the data passed between the layers to lower the cost of accessing the information repositories.

References

[1] P. Brown, D. Fisher, S. Louis, J. R. McGraw, R. Musick and R. Troy, "The Design of a DBMS/MSS Interface," Lawrence Livermore National Laboratory Technical Report, 1995.

[2] R. L. Grosman, H. Hulen, X. Qin, T. Tyler, W. Xu, "An Architecture for a Scalable, High Performance Digital Library," Proceedings of the 14th IEEE Computer Society Mass Storage Systems Symposium, S. Coleman, editor, IEEE, Los Alamites, CA, 1995, pages 89-98.

[3] R. L. Grossman, "Early Experience with a System for Mining, Estimating, and Optimizing Large Collections of Objects Managed Using an Object Warehouse ,"

Proceedings of the Workshop on Research Issues on Data Mining and Knowledge Discovery, Montreal, Canada, June 2, 1996.

[4] R. L. Grossman, S. Bailey, and D. Hanley, "Data Mining Using Light Weight Object Management in Clustered Computing Environments," Proceedings of the Seventh International Workshop on Persistent Object Systems, Morgan-Kauffmann, 1996.

[5] E. Mesrobian, R. Muntz, E. Shek, S. Nittel, M. LaRouche, and M. Krieger, "OASIS: An Open Architecture Scientific Information System," 6th International Workshop on Research Issues in Data Engineering, New Orleans, La. February, 1996.

[6] R. L. Grossman and H. V. Poor, "Optimization Driven Data Mining and Credit Scoring, Proceedings of the IEEE/IAFE 1996 Conference on Computational Intelligence for Financial Engineering (CIFEr), IEEE, Piscataway, 1996, pages 104-110.

[7] D. Teaff, R. W. Watson, and R. A. Coyne, "The Architecture of the High Performance Storage System (HPSS)," Proceedings of the Goddard Conference on Mass Storage and Technologies, College Park, MD, March, 1995.