Grid Collector: An Event Catalog With Automated File Management

Kesheng Wu*, Wei-Ming Zhang†, Alexander Sim*, Junmin Gu* and Arie Shoshani*
*Lawrence Berkeley National Lab, Berkeley, CA 94720, Email: {KWu, ASim, JGu, Arie}@lbl.gov
†Kent State University, Kent, Ohio 44242, Email: zhang@hpacq.kent.edu

***Abstract*— High Energy Nuclear Physics (HENP) experiments such as STAR at BNL and ATLAS at CERN produce large amounts of data that are stored as files on mass storage systems in computer centers. In these files, the basic unit of data is an event. Analysis is typically performed on a selected set of events. The files containing these events have to be located, copied from mass storage systems to disks before analysis, and removed when no longer needed. These file management tasks are tedious and time consuming. Typically, all events contained in the files are read into memory before a selection is made. Since the time to read the events dominate the overall execution time, reading the unwanted event needlessly increases the analysis time. The Grid Collector is a set of software modules that works together to address these two issues. It automates the file management tasks and provides "direct" access to the selected events for analyses. It is currently integrated with the STAR analysis framework. The users can select events based on tags, such as, "production date between March 10 and 20, and the number of charged tracks > 100." The Grid Collector locates the files containing relevant events, transfers the files across the Grid if necessary, and delivers the events to the analysis code through the familiar iterators. There has been some research efforts to address the file management issues, the Grid Collector is unique in that it addresses the event access issue together with the file management issues. This makes it more useful to a large varieties of users.**

I. Introduction

A number of High-Energy Nuclear Physics (HENP) experiments currently produce many petabytes (10^{15} bytes) of data a year and future experiments are expected to produce data even faster. In these data sets, information regarding one collision is called an event. Most user analyses are conducted on subsets of events [1], [2]. Typically, the events are organized into files and the files are stored on tapes in some mass storage systems such as HPSS. To perform an analysis, file management functions such as finding what files contain the wanted events, locating the files, transferring the files to a convenient location for analyses and removing the files afterwards, are some of the most tedious and time consuming tasks. There are two general strategies to deal with this problem, automatically caching needed files or try to place commonly used files on a large disk system. The first approach require significant amount of software infrastructures and the essential pieces of which are just become mature enough for practical use. Our Grid Collector is an example of this approach. It addresses a number of shortcomings of the alternative approach. Next we briefly describe these shortcomings.

In order for the end users not to deal with file management issues, current experiments all establish some large computer centers to house commonly used files on large disk systems, for example the STAR experiment has a large computer center at Brookhaven National Laboratory[1]. In these cases, committees have setup policies to decide what to place on disks. Many users can easily conduct their analyses because the files are on disk. However, there are a number of serious limitations. Because the disk space is only sufficient to store a small part of data, only a small amount of data about each event is on disk. This is generally designed to make "common" analyses convenient and is typically sufficient immediately following the collection of the data. After the initial set of "common" analyses, users may typically want to perform more detailed or more "exotic" analyses that require information than contained in the files on disk. For example, a follow-on study of the anti-particle production in STAR [3] wants to perform a more detailed analysis on about 350 (out of 600,000) events that might contain anti-helium-3 (${}^3\bar{H}e$). Since this type of analysis may be unique for every user, the user has to perform all the necessary file management tasks. Clearly, automating the file management tasks would benefit users in this situation.

Most experiments are large collaborations. They typically have many users who are far away from the computer centers. They typically have their own computing facilities. Automated file management software would also make it much easier to make effective use of these computing facilities.

In experiments like STAR, the analysis framework only allows the user to read all events in a file one after the other. In order to select the wanted events a user has applied a filter after the event data has been read into the memory. This is an inefficient use of the computer resources since the time to read the events typically dominates the overall execution of the analyses. The large disk arrays are typically hosted on file servers, these unnecessary file accesses increases the network traffic and reduces the overall system throughput. There are a number tools such as condor that makes it more convenient to distribute the files over a cluster to reduce the network congestion, however, they can not provide a comprehensive solution to the event access issue.

The Grid Collector is a set of software designed to provide file-transparent event access for analysis programs. Using this

[1]More information available at http://www.star.bnl.gov.

software, users specify their requests for events as sets of conditions on physically meaningful attributes, such as triggers, production versions and other tags. The Grid Collector resolves the given conditions into a list of relevant events and a list of files containing the events. It is able to locate the files and transfer the files as needed. Currently, the Grid Collector is integrated into the STAR analysis framework to provide familiar access functions to the user analysis code. This means that the users can take advantage of its functions with a minimal amount of changes to their analysis programs.

The Grid Collector is designed to work in a stream mode. It attempts to maintain only a small number of active files on disk. This is especially useful for analysis jobs that require a large number of files that can not be stored on disks at the same time. On busy centralized systems, like the one at Brookhaven, it allows different analysis jobs to share files already retrieved without user intervention. This should improve the overall throughput of the analysis jobs. Another benefit of the stream mode is that the analysis program requires less computer memory since a smaller amount of information about files and events are active at any given time. This may improve the overall CPU utilization.

Overall, the Grid Collector has a number of unique features that complements current data analysis approaches. This document describes the main components behind the scene, gives examples of how to use it in the STAR analysis framework, and explains how it may be extended into other experiments.

II. Related Work

Large HENP experiments like STAR, ATLAS[2] and CMS[3] are usually large collaborative projects involving hundreds of researchers from many institutions. Participants have a large varieties of computing and storage resources. The need for easy access to data is generally recognized and there are a number of activities to address various aspects of this issue. Some of the large projects, including the Particle Physics Data Grid (PPDG)[4] and The Grid Physic Network (GriPhyN)[5], are studying various techniques and developing a number of different tools. A general approach of these projects is to leverage the momentum of Open Grid Service Architecture[6] to enable analysis of large distributed datasets.

One of the early projects following this general approach is the Grid Data Management Pilot (GDMP) [4], [5]. It produced a prototype data replication tool using Grid middleware for authentication and data moving. Some of the later projects, such as the Chimera virtual data system [6], produce software that handle additional issues of querying and tracking derivation of datasets from other datasets. The Grid Collector needs to retrieve files over the Grid and cache them. For this purpose, we use a tool called the Storage Resource Manager[7] [7].

The work that most directly influences the design of the Grid Collector is the Storage Access Coordination System (STACS) [8], [9], [10], [11]. Most known approaches treat a file as a basic unit of data [12], [13], but STACS provides file-transparent event access. STACS was able to select events efficiently by using bitmap indices [10]. Another important ingredient of STACS is that it was designed to interact with mass storage systems [9]. These are two important features that are not present in other tools. STACS software was designed for STAR experiment before it came on-line. However, immediately after STAR come on-line, the data most users care about fit on the disks in the computer center. STACS was left unattended while the rest of the STAR analysis framework evolved. For example, STAR has decided to use the ROOT system while STACS and its associated programs are only able to deal with Objectivity files.

STAR experiment has been on-line for a couple of years and the data production rate has been increasing steadily. It is anticipated that the data produced in 2004 will have to be distributed for reconstruction because the computer resources at Brookhaven will not be able perform the task in time. This would make it necessary for the STAR analysis framework to deal with multiple storage sites. The upcoming experiments like ATLAS and CMS are expected to have more than one storage site as well. To demonstrate that file-transparent event access is still efficient, we developed the Grid Collector software. In addition to utilizing Grid middleware to access multiple remote store sites, Grid Collector also implements a more efficient version of the bitmap indexing scheme [14] and a more flexible file caching scheme [7]. It is designed to utilize multiple disk caches distributed on different machines. This makes it suitable for use on clusters of workstations without centralized file server.

III. Architecture

In this section, we describe the overall designed and the main components of the Grid Collector. Overall, the Grid Collector adopts a client-server model. The server portion performs the bulk of the functions and the client portion is currently integrated into the STAR analysis framework. The most unique feature of the Grid Collector is that it provides direct access to event level data. This feature is implemented with the Query Interpreter that holds a set of bitmap indices about the events. It enables the user to specify criteria for selecting events based on known attributes. The information about the selected events in each file is passed to the Event Iterator so that only the selected events are given to the user analysis code. The other modules in the server, including the File Scheduler, the File Catalog, and the Storage Resource Manager[7], are for file management tasks. To save space, we only describe the Query Interpreter and the Event Iterator. The information about the Storage Resource Manager and its Disk Resource Manager (DRM) has been

[2]More information about ATLAS is available at http://atlas.web.cern.ch.
[3]More information about CMS is available at http://cmsinfo.cern.ch.
[4]Information about PPDG is available at http://www.ppdg.net.
[5]Information about GriPhyN is available at http://www.griphyn.org.
[6]Information about OGSA is available at http://www.globus.org/ogsa.

[7]Information about SRM also available at http://sdm.lbl.gov/projectindividual.php?ProjectID=SRM.

OID	$\mathbf{X}$	=0	=1	=2	=3
		bitmap index			
1	0	1	0	0	0
2	1	0	1	0	0
3	3	0	0	0	1
4	2	0	0	1	0
5	3	0	0	0	1
6	3	0	0	0	1
7	1	0	1	0	0
8	3	0	0	0	1
		b_0	b_1	b_2	b_3

Fig. 2. A sample bitmap index.

published [7]. Other modules performs functions that are either relatively straightforward or has been described elsewhere.

A. *Query Interpreter*

The core of the Query Interpreter module is the bitmap index, which is generally used to index the write-once read-many (WORM) data [15], [16], [17]. It is particularly useful in applications like data warehousing to support On-Line Analytical Processing (OLAP). All major database system vendors now have some versions of the bitmap index in their products. Since most data produced in a HENP experiment are only read by analysts, bitmap indexing techniques should be effective for selecting events. In STAR, there are about 500 attributes that can be used to select events. In database research community, this is known as high-dimensional data since each attribute can be viewed a dimension of the data. The selection criteria typically are range conditions on a handful of different attributes. This type of query on high-dimensional data is known as the partial range query because it does not involve all dimensions of the data. Many indexing schemes are available in the database literatures, however most of them are not efficient for data with hundreds of dimensions [18], [19]. To answer partial range queries, they usually take longer than the sequential scan that does not use any index at all [18], [19]. In our tests, the bitmap index is able significantly outperform the sequential scan [14], [20].

Figure 2 shows a bitmap index example. In this case, a bitmap is a set of columns of 0s and 1s. Bitmaps $b_0, \ldots, b_3$ each represent whether $\mathbf{X} = 0, 1, 2,$ or 3. In this case, the number of bitmaps is equal to the number of distinct values of the attribute, also known as the attribute cardinality. When the number of bitmaps is large, they may take up too much space to be used efficiently. This is especially a concern for HENP data since many of the attributes have floating-point values which make the attribute cardinalities very high. We address this with a combination of binning [10] and compression [14].

The current implementation of the Query Interpreter stores the compressed bitmaps in files, not in a database system. We came to this arrangement after a number of experiments show that overhead of using a database system was very high in many cases [20]. The bitmaps are read into memory when needed. They are removed from memory when the space is required for other operations.

The attributes indexed include all attributes in the tag database [21]. The physics working groups of STAR call them tags and have requested about 500 or so to be included in the database. The data production programs generate these tags and store them in ROOT files. Currently, a separate module called the Index Feeder is used to digest the ROOT files before generating bitmap indices. In actual operations, we have noticed some inefficiencies in this arrangement. Eventually we are going to integrate the Index Feeder into the Bitmap Index module.

In addition to tags, we also build indices for attributes such as production version, trigger and magnetic filed configuration. These are common attributes that analysts use to select the data files to perform analyses on. For example, a user might select all events from production "P03ia" with the number of primary tracks greater than 500.

This module implements a number of different indexing strategies. The user may change the default scheme through either a graphical tool or by directly editing the control files.

The Query Interpreter is implemented as two layers, a core for handling bitmap indices plus an interprocessor communication layer. The communication layer currently uses CORBA and can relatively easily changed to use another scheme, say, web service.

B. *Event Iterator*

The Event Iterator is the key that glues the STAR analysis framework to the Grid Collector. To a user, it appears as a low level file access class called `StGridCollector`. It provides the same interface as an existing class named `StFile`. Its constructor takes a string argument. This string is a simplified version of the select statement of SQL [22] or a `GC` command. Both form can be used to specify the conditions on the events to be selected. The Query Interpreter translates the selection criteria into a list of files and events in the files. The Event Iterator response to user analysis program by retrieving files and pass the selected events to the analysis code. Since these interactions are provided through other classes of the STAR analysis framework, such as `StIOMaker`, the user is not required to change how they work in order to take advantage of the Grid Collector.

The Event Iterator has a number of features that are important to the overall function of the Grid Collector. For example, it releases the pin on a file immediately after it receives the function call to retrieve the next file. This allows DRM to reclaim the disk space as necessary. This is also used as a cue to the File Scheduler for it to schedule more prefetches if there are more files to be retrieved from remote storage sites. Normally, a job using the STAR analysis framework holds information about all files it uses. The Event Iterator allows one to use only a small set of files at a time. It is also responsible for generating the necessary key for the underlying ROOT system

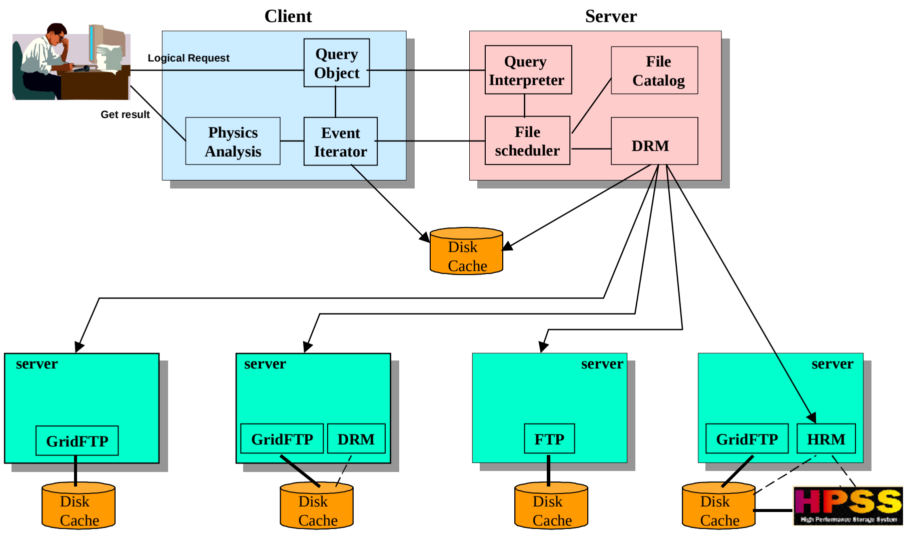

Fig. 1. A scatch of the Grid Collector.

to directly read the selected events instead of looping through all events in the files.

IV. Using the Grid Collector

In a typical analysis program, an object of class `StFile` is constructed to hold the list of files to be analyzed. This object is passed on to another class `StIOMaker` to perform the actual analysis operations. With the Grid Collector, this object is replaced with an object of class `StGridCollector`. As mentioned in previous section, the two objects implements the same set of functions and are thus interchangeable for the remaining portion of the analysis code. Both type of objects also take a string as the argument to their constructors. The difference is in the syntax of the string. The constructor of `StFile` takes a list of file names, the constructor of `StGrid-Collector` takes a select statement or a `GC` command.

The select statement mimics the SQL select statement [22], but much simpler. It supports only three keywords, SELECT, FROM and WHERE. Each keyword can be followed by a clause. In many cases, keywords SELECT and FROM can be omitted. Keyword WHERE is used to lead a set of conditions joined by the usual logical operators, such as “production=P03ia AND numberOfPrimaryTracks>100.” The keyword FROM may be used to specify the data sets to be searched. If it is omitted, it is assumed to be “FROM *.” In STAR, most data sets are identified by attributes such as production version, trigger type, and magnetic field configuration. A user can use these attributes to select events in the WHERE clause instead of specifying the name of data sets in the FROM clause. The keyword SELECT may be used to specify the type of files to be used. In STAR, the types of files supported are `event`, `MuDst`, `hist`, `runco` and `tags`. If keyword SELECT is omitted, `event` files are assumed.

The `GC` command is a string started with “GC ” followed by a set of options. It can be used to carry the same information specified in a select statement. Here we only describe one useful option that can not be easily expressed in a select statement. An event typically has an identifier. A user may wish to specify a list of identifiers and request the Grid Collector to retrieve them. The example given in the 1introduction about a follow-on study of anti-helium 3 acutally has a list of event identifiers available from previous analyses. In this case, the user may place the identifies in a file, say “he3bar,” and use the following `GC` command, “GC -i he3bar.”

Most of the existing analysis scripts in STAR take a list of file names as input to create an object of class `StFile`. To use the Grid Collector, the user replaces the list of file names with a select statement (or a `GC` command) and pass the string to the function `StGridCollector::Create()`. This function creates an object of type `StGridCollector`. Either an object of type `StFile` or `StGridCollector` can be used to create a `StIOMaker` object and the remaining portion of the analysis script does not require any change.

One of the analysis scripts that have gone through this change is `doEvents.C`. Using the old interface, if the user knows that location of some files, the following command line may be used to analyze the first 10 events of the files.

```
.x doEvents.C(10, "/star/data10/*.root")
```

Using the Grid Collector, the following command line analyzes the first 10 events that from production P02gc with more than 300 primary tracks,

```
.x doEvents.C(10, "where production=P02gc
 and numberOfPrimaryTracks>300")
```

The next command can be used to extract the events listed in the file “he3bar.” The output files follow the usual convention in STAR.

```
.x doEvents.C(-1, "GC -i he3bar",
                "evout")
```

V. Summary and Future Plans

The Grid Collector implements an efficient bitmap index to make it practical to catalog millions of events from the STAR experiment. Most user analysis programs make some selections on events. Currently, this is accomplished by first reading the events into memory. Since reading the events into memory takes much longer than the analysis computations, avoiding unnecessary read operations can significantly improve the efficiency of these analysis programs. Many of these selection criteria can be efficiently realized using the bitmap indices in the Grid Collector.

The Grid Collector is designed to perform its own file management functions without user intervention. This can greatly improve the user productivity. The Grid Collector is able to build personalized subsets and users are no longer restricted to work with "official" data sets. This is especially useful for users who perform exotic analyses and those who wish to use their own computer facilities rather than the centrally managed ones.

While developing and testing the current version of the Grid Collector, we have identified a number of issues that needed to be addressed. We will mention two of them here. The first one is related to the use of CORBA. In general, the client-server programming model is being replaced by web services and grid services. In the free software community, there is the lack of support for free CORBA. The Index Feeder uses CORBA to pass data to the Query Interpreter, this causes extra delays because CORBA requires the data to be marshaled and unmarshaled. The second issue is related to how STAR analysis framework handles different types of data. Currently, the two main types of data files, `event` files and `MuDst` files are handled differently. Currently the Grid Collector can process `event` files correctly, but additional work is needed to deal with `MuDst` files.

VI. Acknowledgments

The authors gratefully acknowledge the helps received from the STAR software team, particularly, Jerome Lauret, Victor Perevoztchikov, and Frank Laue. We would also like to thank our early users, especially, Lee Barnby and Aihong Tang, for their patience and willingness to work with a new tool.

References

[1] J. Beringer, N. Brook, P. Buncic, F. Carminati, R. Cavanaugh, P. Cerello, F. Donno, D. Foster, C. Grandi, F. Harris, L. Perini, A. Pfeiffer, R. Pordes, D. Quarrie, A. Sciaba, O. Smirnova, J. Templon, A. Tsaregorodtsev, and C. Tull, "Common use cases for A HEP common application layer for analysis – HEPCAL II," http://www.cern.ch/fca/HEPCAL-II.doc, 2003.

[2] F. Carminati, P. Cerello, C. Grandi, E. V. Herwijnen, O. Smirnova, and J. Templon, "Common user cases for A HEP common application layer – HEPCAL," http://www.cern.ch/fca/HEPCAL-prime.doc, 2003.

[3] C. A. et al (STAR collaboration), "$\bar{d}$ and ${}^3\bar{H}e$ production in $\sqrt{s}_{NN}$ = 130 GeV Au + Au collisions," *Phys. Rev. Lett.*, no. 26, p. 262301, 2001, available on-line at http://link.aps.org/abstract/PRL/v87/e262301.

[4] A. Samar and H. Stockinger, "Grid data management pilot (GDMP): A tool for wide area replication in high-energy physics," in *Proc. of IASTED International Conference on Applied Informatics (AI 2001), Innsbruck, Austria, February 2001*, 2001.

[5] H. Stockinger, A. Samar, K. Holtman, B. Allcock, I. T. Foster, and B. Tierney, "File and object replication in data grids," *Cluster Computing*, vol. 5, no. 3, pp. 305–314, 2002.

[6] I. Foster, J. Vöckler, M. Wilde, and Y. Zhao, "Chimera: A virtual data system for representing, querying, and automating data derivation," in *Fourteenth International Conference on Scientific and Statistical Database management, Proceedings, July 24-26, 2002, Edinburgh, Scotland*, J. Kennedy, Ed. IEEE Computer Society, 2002, pp. 37–46.

[7] A. Shoshani, A. Sim, and J. Gu, "Storage resource managers: Middleware components for grid storage," in *Proceedings of Nineteenth IEEE Symposium on Mass Storage Systems, 2002 (MSS '02)*. IEEE, 2002, paper available on the web at http://romulus.gsfc.nasa.gov/msst/conf2002/papers/d02ap-ash.pdf.

[8] L. Bernardo, H. Nordberg, D. Olson, A. Shoshani, A. Sim, A. Vaniachine, D. Zimmerman, B. Gibbard, R. Porter, T. Wenaus, and D. Malon, "New capabilities in the HENP grand challenge storage access system and its application at RHIC," *Computer Physics Communications*, vol. 140, pp. 179–188, Oct. 2001.

[9] L. M. Bernardo, A. Shoshani, A. Sim, and H. Nordberg, "Access coordination of tertiary storage for high energy physics applications," in *IEEE Symposium on Mass Storage Systems*, 2000, pp. 105–118, document available at http://esdis-it.gsfc.nasa.gov/MSST/conf2000/PAPERS/B05PA.PDF.

[10] A. Shoshani, L. M. Bernardo, H. Nordberg, D. Rotem, and A. Sim, "Multidimensional indexing and query coordination for tertiary storage management," in *11th International Conference on Scientific and Statistical Database Management, Proceedings, Cleveland, Ohio, USA, 28-30 July, 1999*. IEEE Computer Society, 1999, pp. 214–225.

[11] A. Shoshani, A. Sim, L. M. Bernardo, and H. Nordberg, "Coordinating simultaneous caching of file bundles from tertiary storage," in *Proceedings of the 12th International Conference on Scientific and Statistical Database Management, July 26-28, 2000, Berlin, Germany*, O. Günther and H.-J. Lenz, Eds. IEEE Computer Society, 2000, pp. 196–208.

[12] A. S. Johnson, "Java analysis studio," Stanford Linear Accelerator Center, Stanford University, Tech. Rep. SLAC-PUB-7963, 1998, document available at http://www-sldnt.slac.stanford.edu/jas/documentation/Chep98/Chep98.htm and the last software is available at http://jas.freehep.org.

[13] H. B. Newman, J. Bunn, and S. Iqbal, "Distributed heterogeneous data warehouse for grid analysis," http://pcbunn.cacr.caltech.edu/GAE/workshop/SaimaGAEworkshop.ppt, 2003.

[14] K. Wu, E. J. Otoo, and A. Shoshani, "Compressing bitmap indexes for faster search operations," in *Proceedings of SSDBM'02*, 2002, pp. 99–108, lBNL-49627.

[15] P. O'Neil, "Model 204 architecture and performance," in *2nd International Workshop in High Performance Transaction Systems, Asilomar, CA*, ser. Lecture Notes in Computer Science, vol. 359, Sept. 1987, pp. 40–59.

[16] C.-Y. Chan and Y. E. Ioannidis, "Bitmap index design and evaluation," in *Proceedings of the 1998 ACM SIGMOD: International Conference on Management of Data*. ACM press, 1998.

[17] M.-C. Wu and A. P. Buchmann, "Encoded bitmap indexing for data warehouses," in *Fourteenth International Conference on Data Engineering, February 23-27, 1998, Orlando, Florida, USA*. IEEE Computer Society, 1998, pp. 220–230.

[18] V. Markl, M. Zirkel, and R. Bayer, "Processing operations with restrictions in RDBMS without external sorting: The tetris algorithm," in *Proceedings of the 15th International Conference on Data Engineering, 23-26 March 1999, Sydney, Austrialia*. IEEE Computer Society, 1999, pp. 562–571.

[19] R. Weber, H.-J. Schek, and S. Blott, "A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces," in *VLDB'98, Proceedings of 24th International Conference on Very Large Data Bases, August 24-27, 1998, New York City, New York, USA*, J. Widom, A. Gupta, and O. Shmueli, Eds. Morgan Kaufmann, 1998, pp. 194–205.

[20] K. Stockinger, K. Wu, and A. Shoshani, "Strategies for processing ad hoc queries on large data warehouses," in *Proceedings of DOLAP'02*, 2002.

[21] D. Zimmerman, "The design and implementation of the star tag data base," 1998, slides of presentation given at CHEP 98 is available at http://wwwasd.web.cern.ch/wwwasd/cernlib/rd45/chep98/star-tag.ppt.

[22] J. R. Groff and P. N. Weinberg, *SQL: The Complete Reference*, 2nd ed. McGraw-Hill Osborne Media, 2002.