Computing in SCIENCE & ENGINEERING

www.computer.org/cise

Vol. 10, No. 1
January/February 2008

The Catalog Archive Server Database Management System

The multiterabyte Sloan Digital Sky Survey's (SDSS's) catalog data is stored in a commercial relational database management system with SQL query access and a built-in query optimizer. The SDSS Catalog Archive Server adds advanced data mining features to the DBMS to provide fast online access to the data.

The Sloan Digital Sky Survey's (SDSS's) Science Archive catalog data is large and complex enough to warrant a commercial database management system (DBMS) to ensure its long-term integrity and to provide the storage, organization, distribution, and data mining capabilities that it warrants. A traditional hierarchical file-based system, even with specialized tools built on top of it, would have fallen far short of meeting the kind of scientific use that this data deserves and demands.

The SDSS produces both raw and catalog data; here, we focus on the catalog data, which flows into a commercial relational DBMS and is served up to the outside world by the SDSS Catalog Archive Server (CAS). The data-loading pipeline used for this task is the sqlLoader; the online Web interface that provides access to the CAS is the SkyServer (http://skyserver.sdss.org/ or http://cas.sdss.org/).

Although the DBMS we adopted for the CAS provides built-in, advanced data mining and query optimization capabilities, we have additionally built a multidimensional spatial indexing scheme—called the Hierarchical Triangular Mesh (HTM)[1,2]—right into the DBMS to enable fast $O(\log N)$ spatial searches. In this article, we'll look briefly at the DBMS features and schema, and the additional functionality that we've built into it.

1521-9615/08/$25.00 © 2008 IEEE
Copublished by the IEEE CS and the AIP

ANI R. THAKAR, ALEX SZALAY, AND GEORGE FEKETE
Johns Hopkins University
JIM GRAY
Microsoft Research

A Relational DBMS

We chose Microsoft's SQL Server, a Windows-based relational DBMS, as the CAS's data repository. We started with an object-oriented DBMS, with the idea that an object data model would provide a much better conceptual match to the SDSS data as well as superior performance compared to the commercial relational DBMS technology available at the time (early-to-mid 1990s). However, even though the object DBMS's performance was initially satisfactory, it soon became insufficient for the SDSS project's data mining needs. Over the same period, relational DBMS technology advanced to the point where it overtook object DBMSs in terms of ease of use, performance, and reliability features.

An earlier *CiSE* article describes the difficulties we encountered with the object-based system, details about migrating from an object to a relational system, and the features and advantages we gained by adopting SQL Server.[3] The decision to select SQL Server was mostly pragmatic and based on our immediate needs and resources, rather than a comparative evaluation of relational DBMS products. Nevertheless, SQL Server is highly rated for its ease of use and administrative features, and has one of the best query optimizers in the business.[4] We've been quite happy with our choice so far.

The unit of data storage in a relational DBMS is a two-dimensional table of rows and columns. As such, all the CAS data goes into tables in the DBMS. Let's look more closely at the data model.

The CAS Data Model

Figure 1 shows the SDSS CAS schema, which is divided into four functional groups of tables: photometric data tables (Photo group), spectroscopic data tables (Spectro group), tables that contain information about various types of regions in SDSS space (Region group), and metadata tables that contain documentation and other schema information.

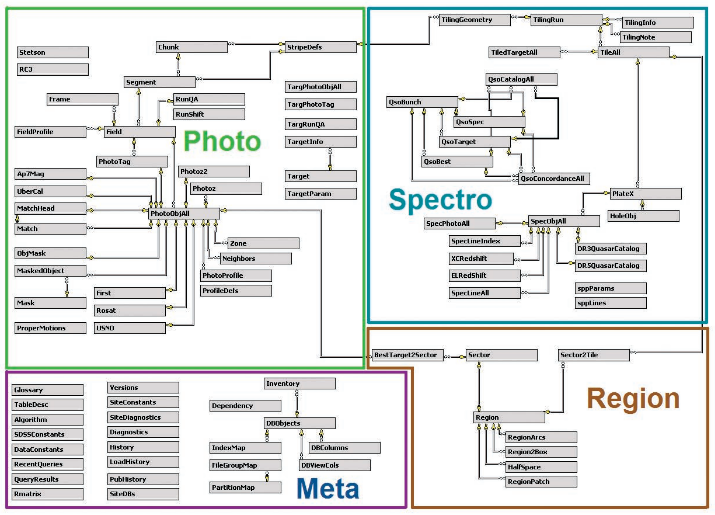

Figure 1. The Catalog Archive Server schema. Note the various database tables and the relationships (foreign keys) between them; tables are grouped by the kind of data they contain.

Photo Tables and Views

The Photo group of tables holds the SDSS photometric pipeline's outputs—that is, the parameters computed from data taken with the SDSS imaging camera. The main table in this group is the PhotoObjAll table, which contains the photometric parameters for each astronomical object that the photometric pipeline identifies. This superset of all observations recorded by the SDSS camera also includes repeat observations of objects. For objects that have multiple recorded observations, the best one is marked as *primary* in the SDSS photometric pipeline; other observations are marked *secondary* if they're good enough for science or *family* for anything else.

Primary and secondary objects in the PhotoObjAll table are listed in the PhotoObj *view* (a view is a virtual table defined by an SQL query), which is the most frequently used subset of the PhotoObjAll data by the majority of users. Expert users and SDSS collaboration members are more likely to go directly to the PhotoObjAll table to look beyond primary and secondary observations. The PhotoObjAll table is by far the largest table in the SDSS data and contains 80 percent of the data (by volume) in the database. We don't cache any of the database views; rather, we let the DBMS handle caching instead. More specialized views of PhotoObj facilitate searching for users who are interested in certain types of objects:

- *PhotoPrimary* for the primary objects in the PhotoObj view;
- *Star* for primary objects classified as stars by the SDSS pipelines;
- *Galaxy* for primary objects classified as galaxies;
- *Sky* for primary sky (sampling) observations;
- *Unknown* for primary observations that the pipeline can't classify as anything else;
- *PhotoSecondary* for the secondary objects in PhotoObj; and
- *PhotoFamily* for objects that can't be marked as either primary or secondary.

We've also made a vertical partition of the PhotoObjAll table called PhotoTag that contains the most frequently accessed columns. It's significantly faster to scan the PhotoTag table because it's a lot thinner than PhotoObjAll and hence makes better use of the cache. PhotoTag also has analogous views to PhotoObjAll called StarTag and GalaxyTag, which return primary star and galaxy objects, respectively.

Other tables in the Photo group contain observing parameters and provenance metadata. The Chunk, Segment, and Field tables, for example, contain information about various units of data export: a chunk is a completely observed part of an SDSS stripe (the SDSS divides the sky into strips using a special coordinate system; a stripe is two strips), a segment is a single camera column (out of six total) within a chunk, and a field is a field of view within a camera column.

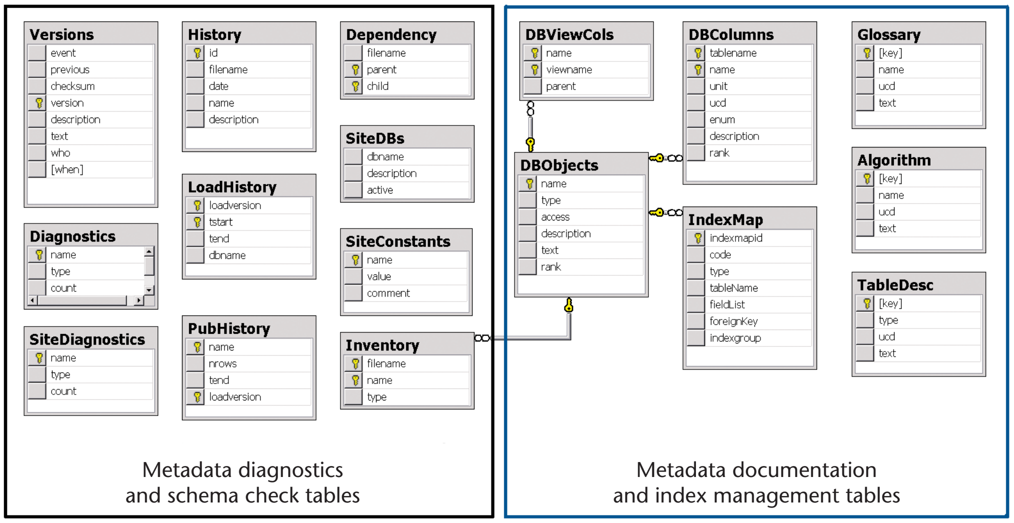

Figure 2. Metadata tables used for diagnostics and integrity checks on the schema. Everything except the Versions, SiteDBs, and SiteConstants tables is autogenerated.

The Photo group also has some derived and science data tables. The US Naval Observatory Survey (USNO), Faint Images of the Radio Sky at Twenty-one centimeters (FIRST), and the Roentgen Satellite (ROSAT) tables, for example, contain matches between the SDSS and the USNO, FIRST, and ROSAT surveys. These tables are indexed by PhotoObj.objid as a foreign key, which is a unique column that links two tables in a relational database. The Match and MatchHead tables contain information about multiple observations of objects typically used for variability and other time-domain science.

Each object's neighbors in the PhotoObjAll table within a predetermined radius (usually 30 arc seconds) are precomputed and kept in the Neighbors table for fast proximity and cross-ID searches. These types of searches are crucial for multiwavelength astronomy because they let astronomers compare data on the same objects in different surveys with different wavelength coverage.

Spectro Tables and Views

The outputs from the one-dimensional Spectro pipeline, spectroscopic targeting, and tiling information are contained in the Spectro group of tables. The main table in this group is the SpecObjAll table, which is analogous to the PhotoObjAll table in the Photo group but with spectra instead of images. Also analogous to PhotoObj is the SpecObj view of the SpecObjAll table, which contains unique spectra determined by the SDSS to be fit for science.

Region Tables

The Region and Sector tables contain information about the survey geometry—in particular, the various types of regions and boundaries pertaining to photometric, spectroscopic, and plate tiling data. They're populated by region and sector functions.

Metadata Tables

The Metadata group includes the contents of the documentation pages in the CAS Web sites (SkyServer and CasJobs), which we can broadly divide into the Help and Schema Browser pages. Together, the metadata tables provide an application-independent description of a DBMS that we can use as a template for other scientific archives. Indeed, the Schema Browser group of tables is already used in the virtual observatory community as a template for SkyNodes in the Open SkyQuery federated query system (www.openskyquery.net). Let's look closer at the tables used for metadata.

Schema description tables. Several tables describe the schema used in the SkyServer, specifically

- *DBObjects*. The description of each object in the schema is kept in the DBObjects table, which describes four types of objects—tables, views, functions, and stored procedures. This table also stores the access type for each object (for example, whether it's accessible to all users or just administrators).
- *DBColumns*. The one-line description of each column in each table is kept in the DBColumns

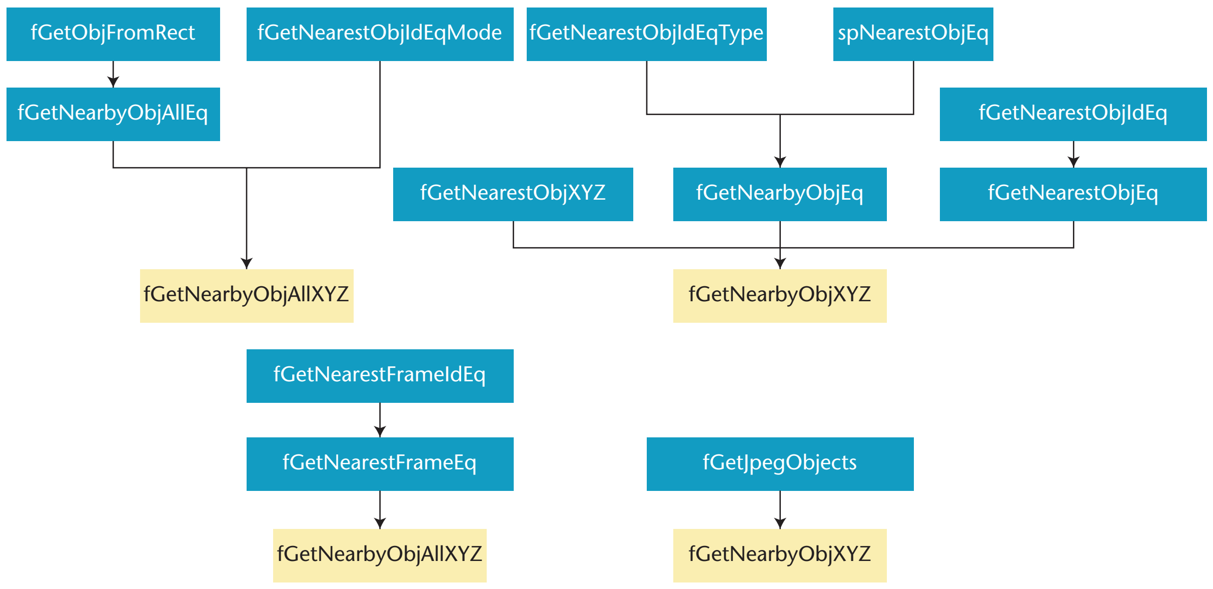

Figure 3. Nearby function dependency chart. These functions provide fast spatial search capabilities using the Hierarchical Triangular Mesh (HTM) spatial index. The light (yellow) functions are the primitives that make the actual HTM library calls. All other functions call these primitives.

table, which also holds other information such as the column's data type, units (if applicable), and a unified content descriptor (UCD) name that identifies the kind of astronomical quantity a column represents. UCDs are now an official International Virtual Observatory Alliance recommendation meant to enable comparison of data between diverse archives without having to know archives' individual schemas (www.ivoa.net/Documents/latest/UCD.html).

- *DBViewCols*. Definitions of columns for views are kept in the DBViewCols table, which is the counterpart of the DBColumns table. A special feature of the DBViewCols is that it allows "*" as shorthand for a view that includes all the columns from the parent table. Thus, column names need not be listed if they are not different from those of the parent table.

Several functions and stored procedures dynamically load the contents of these tables into the Schema Browser as requested by the user. Documentation about possible values for enumerated data columns appears in the DataConstants table and is also linked to the names of the columns in the table schema listing. Other documentation tables include detailed descriptions of SDSS data-processing algorithms, a glossary that explains all the SDSS jargon, and TableDesc, which contains a short description of each table in the database.

The contents of these tables are also dynamically loaded into the SkyServer Help pages, which lets a given SkyServer site show different content depending on the data set it's connected to. Changes to the content for a given data release are thus automatically propagated to all Web sites serving that release.

Diagnostics and schema check. Figure 2 shows the other metadata tables that provide diagnostics and integrity checks on the schema:

- *Inventory* lists every object in the schema on disk, along with its source file and type of object. We use it to cross-check the database schema against what's on disk.
- *Dependency* lists function and stored procedure dependencies.
- *History* shows the modification history of each schema source file for quick searching.
- *LoadHistory* lists all the loading pipeline steps during the load stage.
- *PubHistory* collects all the steps in the data loading process during the publish/merge and finish stages.
- *Diagnostics* lists each database object, its type, and the number of rows in it if it's a table.

Stored procedures associated with the diagnostics tables actually perform the diagnostics checks.

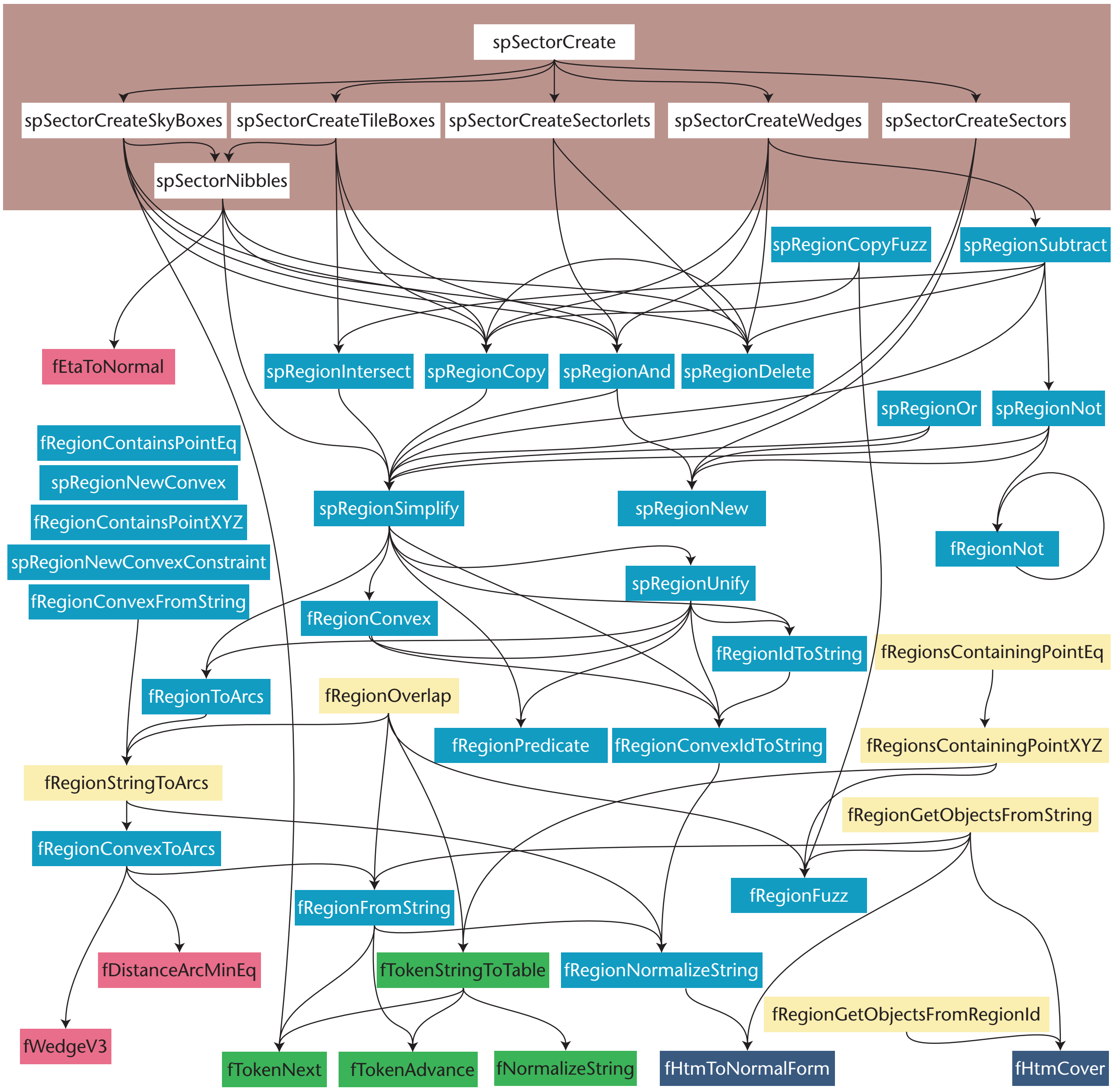

Figure 4. Dependency chart for region and sector functions and stored procedures. These functions compute the spectroscopic plate tiling geometry required for large-scale structure studies. The shading represents the functional group that each function belongs to; the chart illustrates the complex programming achieved with SQL functions and stored procedures inside the database management system.

Functions and Stored Procedures

One of the most powerful features of a DBMS is the ability to write user-defined functions and stored procedures that operate on the data in the database tables. Users perform complex operations, including statistical analysis, right in the database and avoid expensive movement of data as much as possible. This feature is critical for large data sets such as the SDSS.

In the SDSS's databases, many functions and stored procedures written in SQL perform various postprocessing tasks on the data. They're broadly divided into two classes—*admin* and *user*—depending on whether they're available to all users of the database or to database administrators only. There are currently 135 stored procedures (107

Table 1. The IndexMap schema.

Column	Type	Description
indexmapid	Int	Unique primary key for the IndexMap table
code	varchar(2)	One char designator of index category for lookup ('K'|'F'|'I')
type	varchar(32)	Index type, one of ('primary key'|'foreign key'|'index'|'unique index')
tableName	varchar(128)	Name of the table on which the index is built
fieldList	varchar(1000)	List of columns to be included in the index
foreignKey	varchar(1000)	Definition of the foreign key, if any
indexgroup	varchar(128)	The group ID, one of ('PHOTO'|'TAG'|'SPECTRO'|'META'|'TILES'|'FINISH')

admin and 28 user) and 188 functions (6 admin and 182 user).

We can further subdivide user functions and stored procedures into the following groups:

- *Nearest-neighbor searches* are functions and procedures that let users perform proximity searches. Figure 3 shows a dependency chart for the "nearby functions" group.
- *Coordinate conversions* provide conversion between various coordinate systems.
- *Astronomical functions* are specific analysis tasks frequently required by astronomers.
- *Utility functions* are the functions and procedures required to display data and metadata, especially via the Schema Browser and Help pages.

Users can browse the available functions and stored procedures in the SkyServer and CasJobs schema browsers. The CasJobs browser also shows the SQL code for each function or procedure, which is useful for users wanting to learn how to write their own functions.

The administrator class of functions includes the following groups:

- *Region and sector functions* compute the tiling geometry for use in large-scale structure studies. Figure 4 shows the large number of region functions and the complex interrelationships between them. It also shows the kind of heavy-duty computational tasks that we can perform inside the database, where all the various data tables are available in one place and can be searched quickly.
- *HTM functions* apply specifically to the HTM's spatial library (we discuss them more fully in the "Hierarchical Triangular Mesh Spatial Index" section).
- *Web support functions* provide the interface layer between the SkyServer Web pages and the database server. They include procedures to filter user-submitted SQL and log queries. The main procedure in this group of functions, spExecuteSQL, executes user-submitted SQL queries, filters out any dangerous commands (such as "drop table"), and ensures that the query is logged into the usage tracking system. This function also adds the SQL code necessary to limit query results so that queries aren't allowed to return millions of rows and bog down the SkyServer Web interface. The SkyServer also limits query execution time by imposing timeouts on Web pages. Queries that require large CPU and disk resources are handled instead by the CasJobs batch query interface.
- *Metadata and documentation functions* provide the Schema Browser and Help page support.
- *Diagnostics functions* are the procedures for generating diagnostics and checking schema integrity.
- *Loader support functions* include the functions and procedures that the sqlLoader pipeline uses to load and validate data and build auxiliary tables.

The diagnostics functions deserve special mention because they help us maintain the integrity and versioning of the data in the DBMS:

- spMakeDiagnostics checks every table, view, function, and stored procedure into the Diagnostics table and counts the number of rows in each table and view.
- fGetDiagChecksum generates a new checksum from the diagnostics table.
- spCompareChecksum compares the checksum in the SiteConstants table with the checksum generated from diagnostics (using fGetDiagChecksum).
- spUpdateStatistics updates the statistics in every user table.

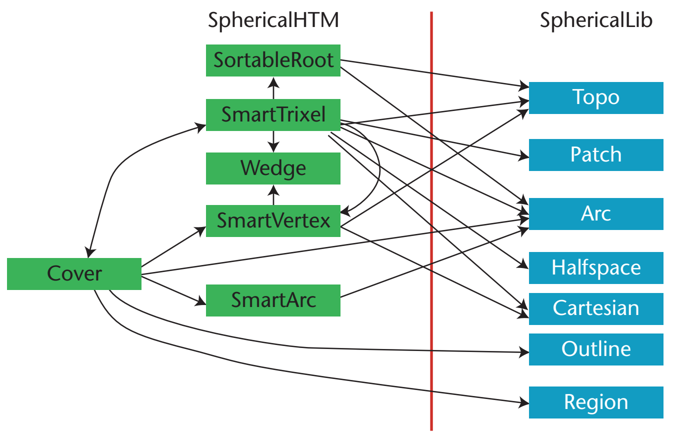

Figure 5. Hierarchical Triangular Mesh (HTM) functions. The cover functions provide the entry interface that calls functions in the spherical libraries. SphericalHTM is an HTM layer on top of SphericalLIB.

- spCheckDBObjects compares the objects in the database with the objects in the schema loaded from disk.
- spCheckDBColumns compares the columns in the database with the columns in the schema loaded from disk.
- spCheckDBIndexes compares the indices in the database (such as those listed in the sysindexes and sysobjects tables) with the indices in the disk schema (in this example, in the IndexMap table).

The diagnostics are typically run at the end of the data loading and publishing process, just before the data is released to the user community.

Database Indices and the IndexMap Table

The SQL Server indices on the CAS databases are created by the sqlLoader data-loading pipeline during the final loading stage; they're managed by the IndexMap table and associated stored procedures. The IndexMap table contains an entry for each index defined in the CAS tables. The indices are subdivided into groups according to the tables they belong to and when they're to be created in the loading process. Some indices are created during the loading stage, when all the data is loaded into the destination tables, and others are created during the publish and finish part of the loading workflow. Table 1 shows the IndexMap's schema.

The IndexMap table's contents are available to users for browsing in the SkyServer Schema Browser, and the names and types of indices on each CAS table are dynamically loaded into the data model description in the SkyServer Help pages.

Hierarchical Triangular Mesh Spatial Index

The size of a data set as big as the SDSS (3 Tbytes) necessitates a spatial indexing technique in addition to database indices to enable fast searches. Without a spatial index, searches based on coordinate cuts would still take too long, particularly for large regions. Proximity searches are also greatly facilitated by such an index. Accordingly, we built a multidimensional *k-d* tree-based spatial index (the HTM). Each object in the SDSS database contains a 64-bit *htmID*, which is the HTM *k-d* tree index key for that object.

We included several functions in the database to implement fast ($O(\log n)$) searching using the HTM index. The nearest-neighbor functions described earlier call the HTM functions to perform spatial searches. The spherical HTM functions implemented in C# comprise an assembly (dynamically linked library) that extends SQL Server 2005 with new scalar and table-valued functions. Figure 5 shows an overview of the HTM library function groups. For details and an HTM primer, see http://skyserver.org/HTM.

Porting the SDSS

We've received (and continue to receive) requests for all the SDSS CAS data from parties wishing to port it to other DBMS platforms such as Oracle, DB2, and MySQL. However, we aren't aware of the SDSS data being successfully deployed on any other DBMS besides Microsoft SQL Server to date. We can think of several reasons why:

- The sheer size of the SDSS data set (several terabytes) makes it quite a daunting task to port it to another DBMS.
- The number of functions and stored procedures in the schema isn't easy to port to other platforms. The SDSS schema has roughly 30,000 lines of SQL code, which required about 10 to 12 person-months of development and testing effort.
- We currently don't have the resources to make the SDSS data available in a portable data format. Obviously, writing several terabytes of database tables to comma-separated value files would take a very long time and isn't straightforward—we would have to split large tables into smaller files, for example. The best we can do at the moment is to make the SQL Server database files available for anyone who wants to

take a stab at porting them to another DBMS.
- The HTM library must also be ported to the other platform and linked to the DBMS.

Although these issues make it difficult for us (and others) to port the CAS to a different DBMS platform, we're actively working on ways to make the various components more generic and portable. We're also working on making the CAS metadata framework generic and extensible to other non-SDSS and non-astronomy archives.

The Microsoft SQL Server relational DBMS gives us a reliable, versatile, and high-performance data repository for the SDSS catalog data. Along with the spatial indexing library and the metadata and diagnostics framework that we've added to it, the ability to formulate efficient SQL queries and write complex code right in the database gives us a very powerful and scalable data mining platform. We continue to enhance and improve the CAS's performance, and in the near future, we plan to implement a distributed, partitioned version of the CAS DBMS on a cluster of database servers using SQL Server's Distributed Partitioned Views technology (see www.databasejournal.com/features/mssql/article.php/3319481). This will let us scale the CAS out to the much larger data sets that will succeed the SDSS in the upcoming years.

References

1. P.Z. Kunszt et al., "The Hierarchical Triangular Mesh," *Mining the Sky: Proc. MPA/ESO/MPE Workshop*, A.J. Banday, S. Zaroubi, and M. Bartelmann, eds., Springer-Verlag, 2001, pp. 631–637.
2. A.S. Szalay et al., *Indexing the Sphere with the Hierarchical Triangular Mesh*, tech. report MSR-TR-2005-123, Microsoft Research, 2005.
3. A.R. Thakar et al., "Migrating a Multiterabyte Astronomical Archive from Object to Relational Databases," *Computing in Science & Eng.*, vol. 5, no. 5, 2003, pp. 16–29.
4. T. Dyck, "Clash of the Titans," *PC Magazine*, vol. 21, no. 6, 2002 pp. 122–138.

Ani R. Thakar *is a research scientist in the Center for Astrophysical Sciences at the Johns Hopkins University. His research interests include science with large databases, data mining, and simulations of interacting galaxies. Thakar has a PhD in astronomy from the Ohio State University. Contact him at thakar@jhu.edu.*

Alex Szalay *is Alumni Centennial Professor of Physics and Astronomy at the Johns Hopkins University. His research interests include cosmology, large-scale structure of the universe, data mining, and science with large databases. Szalay has a PhD from Eotvos University, Hungary. Contact him at szalay@jhu.edu.*

George Fekete *is a research scientist in the Center for Astrophysical Sciences at the Johns Hopkins University. His research interests include graphics, visualization, and spatial data management. Fekete has a PhD in computer science from the University of Maryland, College Park. Contact him at gfekete@pha.jhu.edu.*

Jim Gray*, prior to his disappearance in February 2007, was the Turing Award-winning distinguished engineer, researcher, and manager of Microsoft Research's eScience Group in San Francisco. His primary research interests were in databases and transaction-processing systems, with a particular focus on using computers to make scientists more productive.*