Vol. 10, No. 1
January/February 2008

CasJobs and MyDB
A Batch Query Workbench

Catalog Archive Server Jobs (CasJobs) is an asynchronous query workbench service that lets users run unrestricted SQL queries against scientific catalog archives. After running queries in batch mode, users can save their results to a personal database called MyDB before downloading them, letting users manage their query workloads, results, and histories without causing network overloads.

The SkyServer Web interface[1,2] (http://skyserver.sdss.org) performed extremely well for the Early Data Release (EDR) and Data Release 1 (DR1), even outperforming the object-oriented database that we initially offered as the heavy-duty query engine[3] for power users. However, as the data volume increased by a factor of four (from a few hundred gigabytes to more than a terabyte) from the EDR to DR2, and as the number of objects in the Sloan Digital Sky Survey's Science Archive began to approach 100 million, queries that requested large amounts of data—more than a few gigabytes—became common.

Such queries were beyond the capabilities of the browser-based, synchronous SkyServer Web interface. Specifically, Web browsers cannot render large outputs without disabling the Web server for other users. It was also impractical to maintain a browser session for the length of time required to transport gigabytes of data over the network. The interface's technical difficulties associated with synchronous query execution and high-bandwidth queries over the network motivated us to seek an asynchronous alternative. At the same time, we were also struggling to meet users' needs with regard to querying the huge (and rapidly growing) SDSS data set (see the "Querying a Terabyte Database" sidebar for more details).

To address the SkyServer's limitations and the challenges of querying a terabyte data set, we developed Catalog Archive Server Jobs (CasJobs), an asynchronous query workbench service[4] that provides users of catalog archives the ability to run virtually unrestricted SQL queries against the archive and save results to a personal database. In developing this service, our primary goals were

- to segregate short from long queries to better utilize limited resources and
- to maximize processing local to the database by minimizing data movement over the network (bringing the program to the data, not vice versa).

CasJobs, which is available for free download at http://skyserver.org/casjobs, has proven to be extremely useful and reliable in handling complex and wide-ranging query workloads. Since its initial release in late 2002, CasJobs has seen steadily increasing use and has handled an increasingly complex query workload.

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

NOLAN LI AND ANI R. THAKAR
Johns Hopkins University

Querying a Terabyte Database

We had to address some inadequacies of the Catalog Archive Server (CAS) system as the size of the catalog data began to approach a terabyte because a significant fraction of our user community was expressing dissatisfaction with the available tools and query performance.

Load Balancing

Our analysis of the usage logs for the SkyServer Web interface[1] over the first year of its release showed a power-law CPU-usage pattern (see Figure A). Although the vast majority of queries finished in under a minute, a small number of them ran for a long time and slowed down the database server for all users. More than a couple of such long, intensive queries running concurrently on a given server caused serious performance degradation. Clearly, we needed to segregate the short and long queries to avoid having a few users running intensive, long queries significantly affect the majority of users running short ones.

Increasing SQL Usage

SQL usage steadily increased as users became more familiar with it (see Figure B). They wanted to do more complex queries and postprocessing of the results as they graduated beyond the simple Web-form-based and restricted SQL interfaces in the SkyServer. The most sophisticated CAS users wanted access to SQL Server 2000's Query Analyzer so that they could submit complex queries directly to the database server and write their own Transact-SQL (T-SQL) functions and stored procedures to postprocess the results of their queries. The SkyServer SQL Query page didn't support sessions or query batches, so it couldn't accept variables or other procedural elements of T-SQL. Power users also wanted to cache their intermediate query results on the server so they could download them when they were satisfied with them. The only way to meet this demand was to give users access to the full SQL functionality offered by SQL Server and to let them store results on the server.

Minimizing Data Movement

As the frequency of queries that returned gigabytes of data increased, it became imperative to minimize unnecessary data movement between the query output destination and the database server—for example, due to the unsuccessful or incorrect data requests that are common when users initially formulate and refine their queries. Often, users realized that they didn't need a good fraction of the columns or rows only after they had downloaded all the query results. This type of unnecessary network traffic could be minimized only by saving output close to the database server for verification and postprocessing.

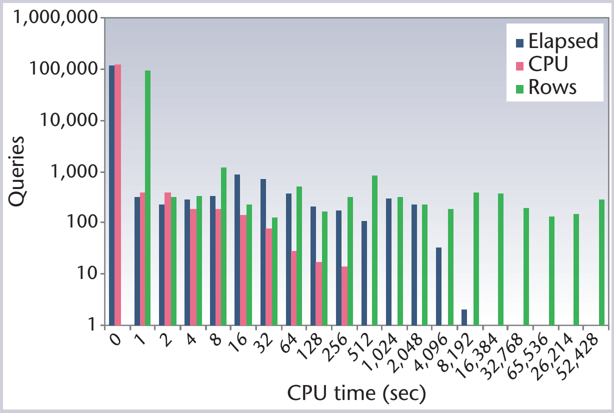


Figure A. Number of queries as a function of elapsed and CPU time (in seconds) vs. the number of rows. The vast majority of queries finish in under a minute.

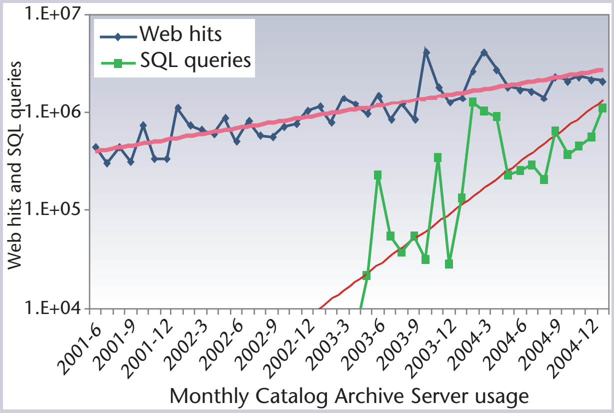


Figure B. Number of Web hits and SQL queries as a function of time. Both have been steadily increasing, although the increase in SQL usage is more dramatic.

Reference

1. A. Szalay et al., "The SDSS SkyServer - Public Access to the Sloan Digital Sky Server Data," *Proc. 2002 ACM SIGMOD Int'l Conf. Management of Data*, ACM Press, 2002, pp. 570–581.

CasJobs Functionality

CasJobs is a set of .NET Web applications and services implemented with ASP.NET and C# that provides an asynchronous batch query workbench interface to the SDSS Catalog Archive Server (CAS). We can best describe CasJobs's

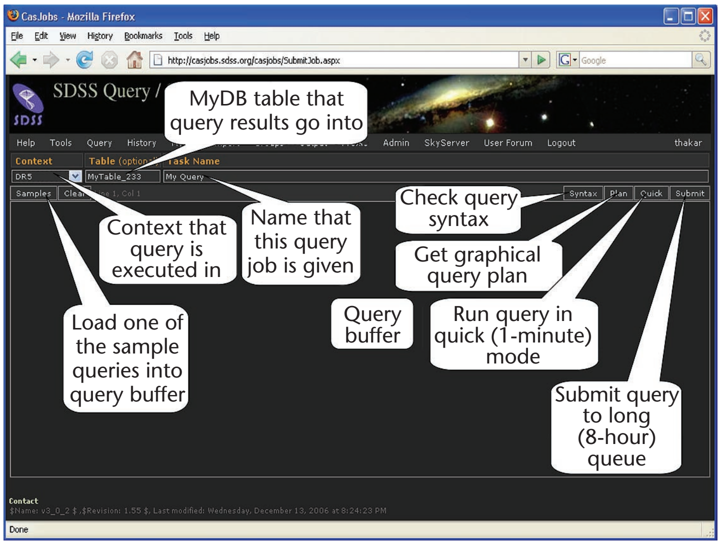

MyDB table that query results go into
Check query syntax
Context that query is executed in
Name that this query job is given
Get graphical query plan
Load one of the sample queries into query buffer
Query buffer
Run query in quick (1-minute) mode
Submit query to long (8-hour) queue
SDSS Query /
Context
Table (optional)
Task Name
DR5
MyTable_233
My Query
Samples
Clear
Syntax
Plan
Quick
Submit

Figure 1. CasJobs query pane showing various controls. Each query is submitted to a particular context, and results are sent by default to a MyDB table if users click the Submit button. If users submit their queries with the Quick button, the results are displayed in the browser under the query pane.

full functionality by highlighting its primary features:

- *MyDB*. Each user has a personal 500-Mbyte SQL Server database that query results are sent to for batch queries.
- *Contexts*. CasJobs introduces the concept of a context as an abstraction of the database layer. Contexts let users select from multiple mirrors of the same data set as well as run queries and joins between multiple different data sets.
- *Two modes of query execution*. Users can submit queries in one of two modes on the query page: a synchronous mode for quick queries and an asynchronous batch mode for long queries.
- *Job history and management*. A detailed Job History page is available to users for viewing all the jobs they've ever submitted along with their original queries (which they can resubmit with or without modification). This system maintains a history of all queries a user has run, as well as execution information about queries, such as the time they took to run, and whether they succeeded or failed.
- *Table import*. Users can import their own data as tables in MyDB via the CasJobs Import page. Although we limit the imported table's size, this is still a convenient and powerful way for users to import data into CasJobs and run cross-match or other queries on it. For example, users can import the entire Abell Galaxy Cluster into CasJobs and cross match it with SDSS data.
- *Shared data with groups*. Users can create or join one or more groups to which they can publish their MyDB tables. This makes the tables visible to all members of that group in their MyDBs. Group members can also see the query that created the table. Members can query shared tables like any other table using their fully qualified names.
- *User login and account management*. Users must log in to run batch queries, but they can run quick queries without logging in.
- *Data extraction*. Users can download copies of MyDB tables to their home machines. The expectation is that users typically download their tables only after they've refined their queries and results to their satisfaction, thus minimizing network traffic.
- *Annotation*. Users can leave comments on any database object in any visible context. Such comments are then visible to all users who can view that object. This lets people leave descriptions and usage details that might benefit other users and collaborators.
- *Access control*. It's possible to restrict access to contexts to a certain group so that only group members can query that context. In this case, the group's administrator controls the group's member list. Restricted-access contexts might be used to give access to data that isn't yet public or to provide additional resources for a restricted group by having such contexts map to more powerful servers or queues with larger time limits.
- *Programmatic access*. The Web services APIs (http://cas.sdss.org/casjobs/casjobs.asmx) facilitate the development of third-party access tools for programmatic access to CasJobs. A Java command-line access tool that we developed using these APIs is currently available from the CasJobs Tools page, and an Emacs script is available for running CasJobs queries.

All these features provide a versatile and convenient workbench environment for users to perform complex query and analysis tasks on the catalog data.

Batch Queries

Figure 1 shows an example of the CasJobs query page, which users see when they log in.

For database and data-set abstraction, load balancing, and access to multiple databases, CasJobs uses contexts to run queries on particular data sets. Each context corresponds to a particular instance of a database (that is, a database and server combination)—for example, there's a DR5 context corresponding to the BestDR5 database, a DR4 context for the BestDR4 database, and so on—in addition to the MyDB context. We reference database objects with a simple `context.objname` syntax (such as `DR6.PhotoObj`), replacing the traditional Transact-SQL (T-SQL) `database.owner.objname` syntax (such as `BestDR6.dbo.PhotoObj`). Contexts let users select from multiple mirrors of the same data set, allowing a crude form of load balancing. Contexts also let users run queries and joins between multiple data sets.

Each query is submitted to a particular context. The user can change the context by selecting from the dropdown context menu. The list of contexts visible to a user is determined by that user's privileges; the order displayed in the context menu is determined by the rank field for the context in the BatchAdmin table that holds the entry for each context. Each context name is unique, but a given database can have more than one context (on different servers) for load balancing.

Quick and Batch Modes and Output

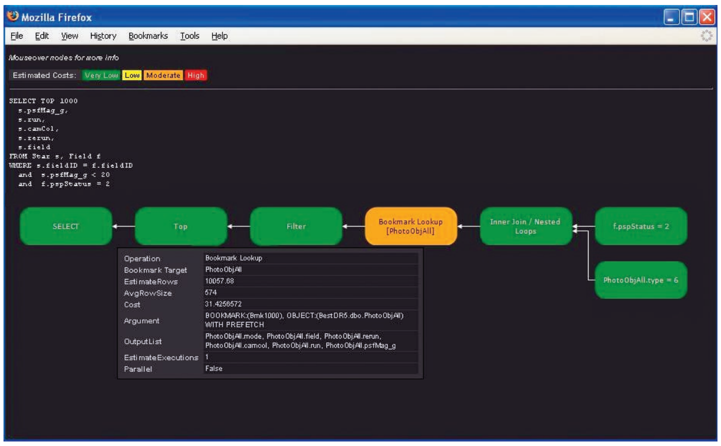
CasJobs's users can submit queries in one of two modes on the query page:

- A query started with the Quick button is executed synchronously, with a low maximum execution time and a limit on the amount of data returned. This mode is intended for low-overhead queries, such as object creation or deletion, data exploration, and so on. CasJobs imposes no limit on the number of concurrent quick queries.
- Queries started with the Submit button are executed asynchronously with a high maximum execution time and no data limit (other than the MyDB size, which is 500 Mbytes by default). The results of these queries can only be sent to the user's MyDB, not to the client browser. We limit the number of concurrent queries per context, so the job scheduler queues submitted queries until the number of executions in that context drops below the context limit. This mode is intended for queries with an expected high execution time or output size.

Each context accepts query submission by either of these methods. We can route each type of query to a different server, thus segregating the quick queries from the long ones. The Quick button in Figure 1 will execute the query immediately (synchronously) but will limit the query execution time to one minute (configurable) and output buffer to a modest size. If the query doesn't complete within one minute or the output is too large, it is canceled and an error message is displayed. If the query finishes within a minute, the results are displayed in the browser in a scrollable preview window.

If users select the Submit button, then the query executes in batch mode and is queued for execution in the queue corresponding to the chosen context. Each context has two queues associated with it, a quick (one-minute) queue and a long (usually eight-hour) queue.

The default output destination for synchronous quick queries is the client browser, so the "Table (optional)" field in Figure 1 is ignored unless explicitly specified in the query using the INTO clause. However, each batch query's output always goes to a MyDB table. If the user does not enter a table name in the optional field, CasJobs generates a table name of the form "MyTable_<*n*>," where *n* is the next unused table number in that user's MyDB.


Mozilla Firefox
File Edit View History Bookmarks Tools Help
Mouseover nodes for more info
Estimated Costs: Very Low Low Moderate High
SELECT
Top
Filter
Bookmark Lookup [PhotoObjAll]
Inner Join / Nested Loops
f.pspStatus = 2
PhotoObjAll.type = 6

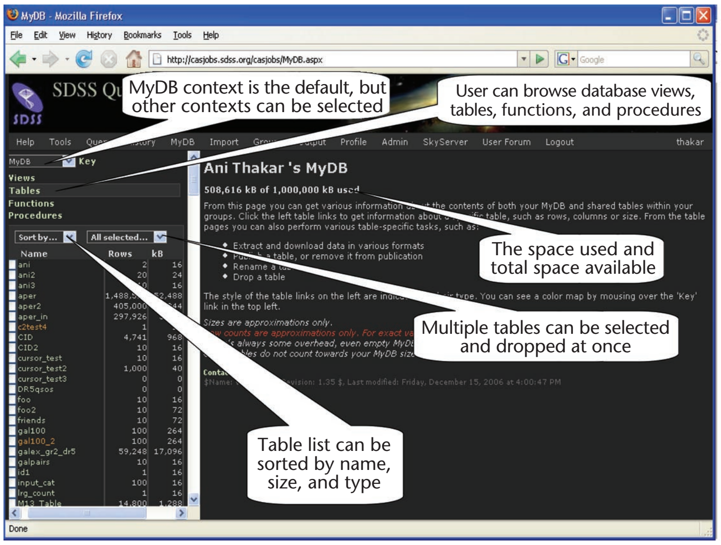
```
SELECT TOP 1000
  s.psfMag_g,
  s.run,
  s.camCol,
  s.rerun,
  s.field
FROM Star s, Field f
WHERE s.fieldID = f.fieldID
  and  s.psfMag_g < 20
  and  f.pspStatus = 2
```

Operation	Bookmark Lookup
Bookmark Target	PhotoObjAll
EstimateRows	10057.68
AvgRowSize	574
Cost	31.4256572
Argument	BOOKMARK:(Bmk1000), OBJECT:(BestDR5.dbo.PhotoObjAll) WITH PREFETCH
OutputList	PhotoObjAll.mode, PhotoObjAll.field, PhotoObjAll.rerun, PhotoObjAll.camcol, PhotoObjAll.run, PhotoObjAll.psfMag_g
EstimateExecutions	1
Parallel	False

Figure 2. Graphical query plan showing color-coded steps and mouse-over details. This is a graphical version of the query execution plan that the SQL Server generates, showing how expensive each part of the query is in relative terms (percentage of the whole execution time).

Query Plan, Syntax Check, and Samples

CasJobs provides a color-coded version of the SQL Server graphical query plan when users press the Plan button (see Figure 1) and the query buffer holds a valid query. Figure 2 shows an example of a query plan display; execution steps are color coded by their estimated cost. Details of each step are propagated from the SQL Server and displayed as mouse-over tool tips.


MyDB - Mozilla Firefox
http://casjobs.sdss.org/casjobs/MyDB.aspx
MyDB context is the default, but other contexts can be selected
User can browse database views, tables, functions, and procedures
Help Tools Query History MyDB Import Groups Output Profile Admin SkyServer User Forum Logout thakar
MyDB Key
Views
Tables
Functions
Procedures
Sort by... All selected...
The space used and total space available
Multiple tables can be selected and dropped at once
Table list can be sorted by name, size, and type
Done

Ani Thakar 's MyDB

508,616 kB of 1,000,000 kB used

From this page you can get various information about the contents of both your MyDB and shared tables within your groups. Click the left table links to get information about a specific table, such as rows, columns or size. From the table pages you can also perform various table-specific tasks, such as:

- Extract and download data in various formats
- Publish a table, or remove it from publication
- Rename a table
- Drop a table

The style of the table links on the left are indicative of their type. You can see a color map by mousing over the 'Key' link in the top left.

Sizes are approximations only.
Row counts are approximations only. For exact va[illegible]
[illegible]'s always some overhead, even empty MyD[illegible]
[illegible]bles do not count towards your MyDB size

Conta[illegible]
$Name: [illegible] Revision: 1.35 $, Last modified: Friday, December 15, 2006 at 4:00:47 PM

Name	Rows	kB
ani	2	16
ani2	20	24
ani3	[illegible]	16
aper	[illegible]	[illegible]
aper2	405,000	[illegible]
aper_in	297,926	[illegible]
c2test4	1	[illegible]
CID	4,741	968
CID2	10	16
cursor_test	10	16
cursor_test2	1,000	40
cursor_test3	0	0
DR5qsos	0	0
foo	10	16
foo2	10	72
friends	10	72
gal100	100	264
gal100_2	100	264
galex_gr2_dr5	59,248	17,096
galpairs	10	16
id1	1	16
input_cat	100	16
lrg_count	1	16
M13_Table	14,800	1,288

Figure 3. MyDB front page. This view shows the object browser (showing tables) and initial overview in the selected object pane.

The Syntax button in Figure 1 lets users perform a syntax check before submitting a query. This simply passes the query on to the SQL Server for syntax checking. CasJobs displays a "Syntax OK" message in green if the query is valid; otherwise, the error message from the SQL Server is propagated and displayed in red.

In addition, the user can load any of more than 50 sample queries with one click, via the Samples button in Figure 1. These sample queries are also available on the SkyServer site, ranging from simple `SELECT-FROM-WHERE` queries to complex queries with multiple joins and nestings. The dropdown menu for the sample queries has mouse-over descriptions for each query.

Query Buffer

The CasJobs query buffer accepts free-form SQL queries, with the query window providing basic editing capabilities. Queries can't be saved, but a submitted query is always available in the history page so users can edit and resubmit it at a later time.

Due to CasJobs's distributed implementation, a few small differences still exist between the standard SQL Server T-SQL and the version accepted by the CasJobs parser. The most prominent difference is the way a table name is specified (`context.table` rather than `database.owner.table`). Other differences (which we explain in the Help and FAQ pages) are related to how function calls and multiple queries in a single buffer are handled.

MyDB

When users create a CasJobs account, they get control over a 500-Mbyte SQL Server database, but they don't have permission to expand or delete it. The initial size is intentionally small because inexperienced users are likely to inadvertently run queries that generate very large output tables. Users can request more space for their MyDBs from the SDSS help desk, and all reasonable requests are usually granted, including temporary requests for large amounts of space for specific queries. It's not uncommon for users to have 1- to 5-Gbyte MyDBs, with some collaboration users (temporarily) having MyDBs as large as 40 Gbytes. MyDBs can be physically located on multiple servers dedicated for MyDB use for better performance and load balancing.

Users can create and drop tables, views, functions, and stored procedures in their MyDBs using various buttons on the MyDB page or running them as T-SQL queries on the query page.

The MyDB page in CasJobs presents the user with an object browser in the gutter frame and information about the selected object in the main frame. The object browser lists objects sorted either alphabetically (default) or by another key (such as size or type) that the user chooses. The displayed objects include tables (default), views, functions, and procedures.

When the user initially clicks on the MyDB menu link, the selected object pane gives an overview of MyDB operations instead of information about a selected object (see Figure 3). The top of the pane displays the current space used and total space available in the database. Once the user clicks on a particular object from the list, this pane displays information about that object.

Tables and Views

The browser and selected object panes for tables and views are similar. Views are virtual tables that select a horizontal or vertical partition of a database table.

When tables (the default) are selected, the approximate number of rows and kilobytes that each table occupies in the database are shown next to the table's name. (This information isn't available for views.) Users can sort table information by type or size, but several table types are denoted by color coding:

- *Normal* tables have the default coloring.
- *Published* tables are the ones that the user owns that have been published to the user's group(s).

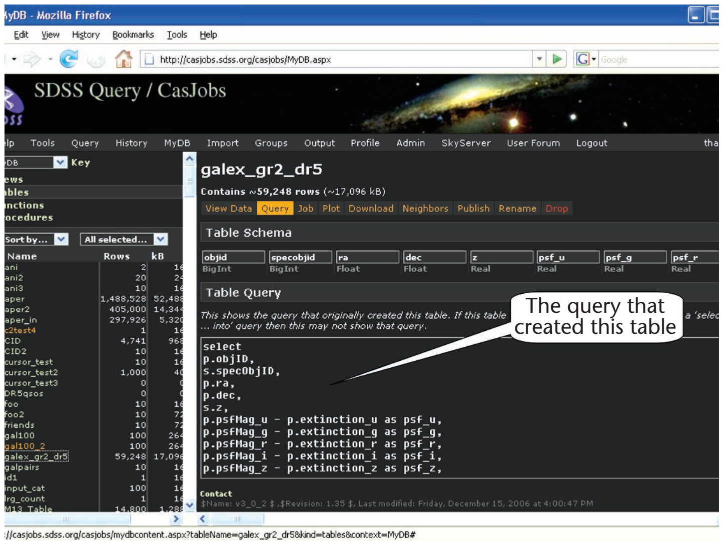
- *Group* tables are published by other users to the current user's groups.
- *Pending* tables are still being created by a query that hasn't finished yet. For obvious reasons, these tables can't be dropped until their state changes.
- *Failed* tables are the results of failed queries (hence they are usually incomplete).

The colors change when users mouse over the key next to the context dropdown menu.

The selected object pane in Figure 4 contains a mouse-over menu with selections for various functions, depending on the type of object selected. Tables have functions as well:

- *View data* displays the selected number of rows (default 100) from the selected table or view.
- *Query* displays the query that created the table or view.
- *Job* shows information about the job that created this table.
- *Plot* plots two columns of the table against each other using a Java plotting tool. This is a simple plotting capability that lets users create and download quick plots.
- *Download* copies the table data out to a file of the selected format such as ASCII comma-separated value (CSV); Flexible Image Transport System (FITS, http://fits.gsfc.nasa.gov/); or Virtual Observatory Table (VOTable, www.ivoa.net/twiki/bin/view/IVOA/IvoaVOTable), a standard that provides an XML schema for FITS tables. Users can then download the file from the output page.
- *Neighbors* searches for spatial neighbors of each object in the table within a specified search radius. This is a quick way for astronomers to do a proximity search on an imported table of objects (from another data set) or points of interest. The table must have right ascension (RA) and declination (dec) columns containing the coordinates on which neighbor searches are to be centered. The Help page under "Advanced CasJobs Queries" provides examples of more advanced neighbors searches.
- *Publish* shares this table or view with the user's groups.
- *Rename* renames the table or view.
- *Drop* deletes the table or view from the database.

The table information also includes a horizontal listing of the columns in the table and their respective data types under the Table Schema heading. Users can rename individual columns in the table schema by directly editing the column names shown on the page. There's currently no capability to add or delete columns, although users can do this in the query window using the appropriate T-SQL commands if they know how.


MyDB - Mozilla Firefox
Edit View History Bookmarks Tools Help
http://casjobs.sdss.org/casjobs/MyDB.aspx
SDSS Query / CasJobs
Tools Query History MyDB Import Groups Output Profile Admin SkyServer User Forum Logout
Key
Sort by... All selected...
View Data Query Job Plot Download Neighbors Publish Rename Drop
The query that created this table

galex_gr2_dr5

Contains ~59,248 rows (~17,096 kB)

Table Schema

objid	specobjid	ra	dec	z	psf_u	psf_g	psf_r
BigInt	BigInt	Float	Float	Real	Real	Real	Real

Table Query

This shows the query that originally created this table. If this table ... into' query then this may not show that query.

```
select
p.objID,
s.specObjID,
p.ra,
p.dec,
s.z,
p.psfMag_u - p.extinction_u as psf_u,
p.psfMag_g - p.extinction_g as psf_g,
p.psfMag_r - p.extinction_r as psf_r,
p.psfMag_i - p.extinction_i as psf_i,
p.psfMag_z - p.extinction_z as psf_z,
```

Contact
$Name: v3_0_2 $,$Revision: 1.35 $, Last modified: Friday, December 15, 2006 at 4:00:47 PM

Name	Rows	kB
ani	2	1
ani2	20	2
ani3	10	1
aper	1,488,528	52,488
aper2	405,000	14,344
aper_in	297,926	5,320
c2test4	1	1
CID	4,741	968
CID2	10	1
cursor_test	10	1
cursor_test2	1,000	4
cursor_test3	0	
DR5qsos	0	
foo	10	1
foo2	10	7
friends	10	7
gal100	100	26
gal100_2	100	26
galex_gr2_dr5	59,248	17,096
galpairs	10	1
id1	1	1
input_cat	100	1
lrg_count	1	1
M13_Table	14,800	1,288

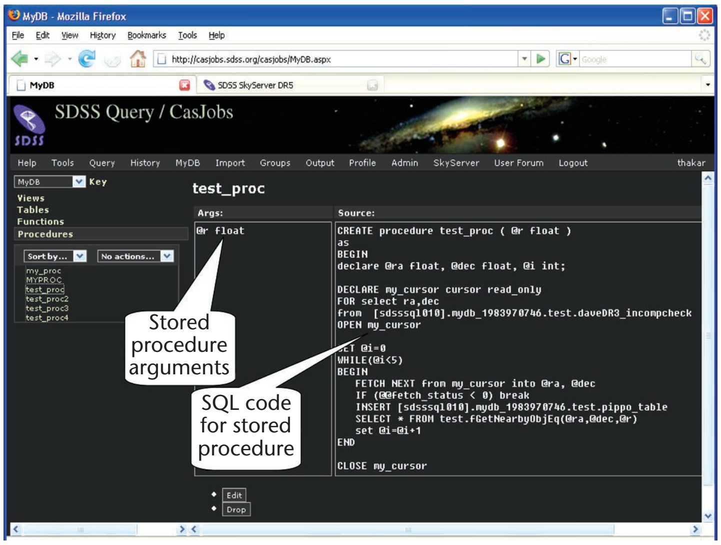
://casjobs.sdss.org/casjobs/mydbcontent.aspx?tableName=galex_gr2_dr5&kind=tables&context=MyDB#

Figure 4. MyDB page showing a selected table. The query function is highlighted, showing the query that created the table (if available).

Functions and Procedures

The object browser lets users browse user-defined functions and stored procedures in the selected context (the default is MyDB) on the MyDB page by selecting the appropriate object type (functions or procedures) in the object browser and then selecting a particular function or procedure to view information about it. Figure 5 gives an example of a stored procedure on the MyDB page; this image displays the arguments and the SQL code for the function or procedure. For a MyDB function or procedure, the information pane also has buttons that let users edit or drop that object. The other databases are read-only, so these buttons aren't available, and the user can browse only the arguments and SQL code.

The ability to view the SQL code for functions and procedures lets users clone constructs from SDSS procedures in their MyDBs. Some restrictions on using SDSS functions and procedures are due to the fact that CasJobs queries are executed in distributed mode—that is, the function or procedure must be invoked remotely from a different server, which isn't allowed in the SQL Server. The user can get around this restriction


MyDB - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://casjobs.sdss.org/casjobs/MyDB.aspx
MyDB
SDSS SkyServer DR5
SDSS Query / CasJobs
Help Tools Query History MyDB Import Groups Output Profile Admin SkyServer User Forum Logout thakar
MyDB Key
Views
Tables
Functions
Procedures
Sort by... No actions...
my_proc
MYPROC
test_proc
test_proc2
test_proc3
test_proc4
Stored procedure arguments
SQL code for stored procedure
Edit
Drop

test_proc

Args:

```
@r float
```

Source:

```
CREATE procedure test_proc ( @r float )
as
BEGIN
declare @ra float, @dec float, @i int;

DECLARE my_cursor cursor read_only
FOR select ra,dec
from  [sdsssql010].mydb_1983970746.test.daveDR3_incompcheck
OPEN my_cursor

SET @i=0
WHILE(@i<5)
BEGIN
   FETCH NEXT from my_cursor into @ra, @dec
   IF (@@fetch_status < 0) break
   INSERT [sdsssql010].mydb_1983970746.test.pippo_table
   SELECT * FROM test.fGetNearbyObjEq(@ra,@dec,@r)
   set @i=@i+1
END

CLOSE my_cursor
```

Figure 5. MyDB page showing a stored procedure listing. The procedure arguments and the T-SQL code for it appear here, along with buttons to edit or drop the procedure.

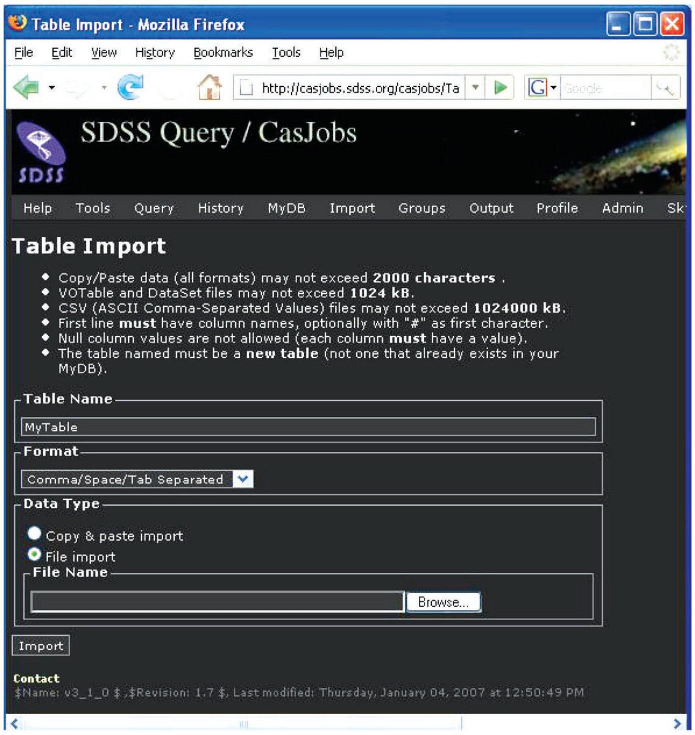


Table Import - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://casjobs.sdss.org/casjobs/Ta
SDSS Query / CasJobs
Help Tools Query History MyDB Import Groups Output Profile Admin Sk
Browse...
Import

Table Import

- Copy/Paste data (all formats) may not exceed **2000 characters** .
- VOTable and DataSet files may not exceed **1024 kB**.
- CSV (ASCII Comma-Separated Values) files may not exceed **1024000 kB**.
- First line **must** have column names, optionally with "#" as first character.
- Null column values are not allowed (each column **must** have a value).
- The table named must be a **new table** (not one that already exists in your MyDB).

Table Name: MyTable

Format: Comma/Space/Tab Separated

Data Type: Copy & paste import [✓ File import]

File Name

Contact
$Name: v3_1_0 $,$Revision: 1.7 $, Last modified: Thursday, January 04, 2007 at 12:50:49 PM

Figure 6. Table import page. Comma-separated values or white-space-separated files are accepted as well as specialized formats such as XML. Users can import small files using copy and paste. The current uploaded file size limit is 100 Mbytes.

by cloning the function in his or her MyDB and invoking it locally.

Data Import and Export

When users download data from their MyDB table, we queue the table output as a job just as with queries, although a separate job scheduler can handle output jobs (see the "Jobs Service" section for details). Downloads of large tables can take a considerable amount of time, so they must be done in batch mode as well.

Users can import their own data into CasJobs using the Table Import page. Accepted formats for imported data also include ASCII CSV, FITS, and VOTable. Because the import is via an HTTP upload, the uploaded file's size is (currently) limited to 100 Mbytes by the `Web.config` (the default configuration file used by .NET Web services and applications) parameter `MAX_HTTP_REQUEST_LENGTH`. Setting the limit too high causes too much memory to be used for the upload. Another consideration is how long it takes to upload the file: the limit must be small enough that the upload can be completed in a few minutes. Figure 6 shows the Table Import page.

Sharing Data with Groups

The groups functionality lets users share their data and queries with other users. The Groups page gives users the ability to create, subscribe, and unsubscribe to user groups.

Creating a Group

A user can create a group with the Create Group button; the user's current list of groups is listed on the left, and pending invitations to subscribe to groups created by other users are listed at the top. The user can accept or decline an invitation to join a group. Once the invitation is accepted, the member shows up in the group.

When users click on a group that they've created, a Manage Users button lets them invite other users to join the group. This brings up the list of users who have their group visibility flag turned on in their personal profile (set in the Profile page). The group creator can choose one or more users from the list and invite them to join the group.

Publishing Tables to a Group

The MyDB page lets users publish individual tables to a particular group. Once published, these tables become visible to all users who have accepted the invitation to join that group and are color coded in their MyDBs in a standard `GroupName.UserId.TableName` name format (such as `QuasarWorkingGroup.jdoe.HiZQsos`). Thereafter, group members can use these tables in their queries using a qualified name—for example, “`SELECT TOP 10 * FROM QuasarWorking-Group.jdoe.HiZQsos`”.

Users can turn on or off lists of group tables in the MyDB page on a per-group basis. A user might choose to display all tables or only local tables in the MyDB or include tables from specific groups.

Jobs Service

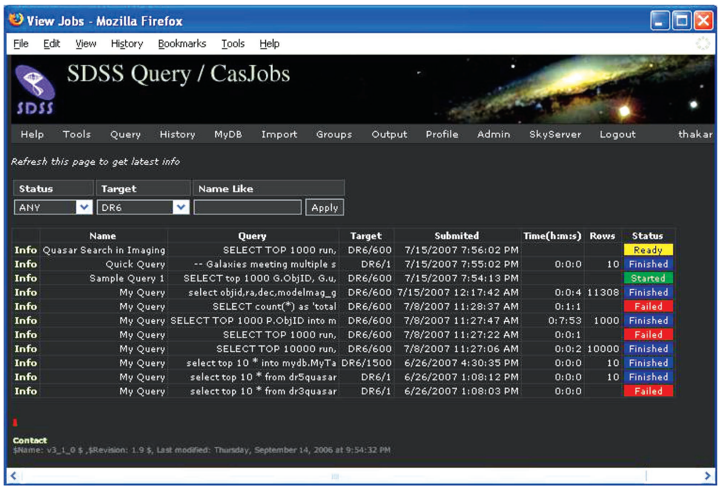
The jobs service is a stand-alone application that schedules query and output jobs. For better performance and load balancing, CasJobs administrators can run separate instances of the service for query and output job handling.

Job Scheduling

The jobs service schedules jobs in the order of their submission time, but users aren’t allowed to run successive jobs if other users are waiting. This is to ensure that users submitting multiple, machine-gun queries don’t hog a given queue. At the end of each job, the service will give preference to a job submitted in the meantime from a different user before picking up the next job from the same user. For example, in Table 1, if job 1001 completes sometime after 11:30:00, then job 1004 will be picked up before job 1002. This prevents user Pumba from running successive jobs and gives user Timone a fair chance to run his job.

Table 1. Example job scheduling log in the jobs service application.

Target	Queue	User ID	Time submit	Job ID
DR5	500	Pumba	11:10:00	1001
DR5	500	Pumba	11:11:00	1002
DR5	500	Pumba	11:15:00	1003
DR5	500	Timone	11:30:00	1004


View Jobs - Mozilla Firefox
File Edit View History Bookmarks Tools Help
SDSS Query / CasJobs
SDSS
Help Tools Query History MyDB Import Groups Output Profile Admin SkyServer Logout thakar

Refresh this page to get latest info

Status	Target	Name Like	
ANY	DR6		Apply

	Name	Query	Target	Submited	Time(h:m:s)	Rows	Status
Info	Quasar Search in Imaging	SELECT TOP 1000 run,	DR6/600	7/15/2007 7:56:02 PM			Ready
Info	Quick Query	-- Galaxies meeting multiple s	DR6/1	7/15/2007 7:55:02 PM	0:0:0	10	Finished
Info	Sample Query 1	SELECT top 1000 G.ObjID, G.u,	DR6/600	7/15/2007 7:54:13 PM			Started
Info	My Query	select objid,ra,dec,modelmag_g	DR6/600	7/15/2007 12:17:42 AM	0:0:4	11308	Finished
Info	My Query	SELECT count(*) as 'total	DR6/600	7/8/2007 11:28:37 AM	0:1:1		Failed
Info	My Query	SELECT TOP 1000 P.ObjID into m	DR6/600	7/8/2007 11:27:47 AM	0:7:53	1000	Finished
Info	My Query	SELECT TOP 1000 run,	DR6/600	7/8/2007 11:27:22 AM	0:0:1		Failed
Info	My Query	SELECT TOP 10000 run,	DR6/600	7/8/2007 11:27:06 AM	0:0:2	10000	Finished
Info	My Query	select top 10 * into mydb.MyTa	DR6/1500	6/26/2007 4:30:35 PM	0:0:0	10	Finished
Info	My Query	select top 10 * from dr5quasar	DR6/1	6/26/2007 1:08:12 PM	0:0:0	10	Finished
Info	My Query	select top 10 * from dr3quasar	DR6/1	6/26/2007 1:08:03 PM	0:0:0		Failed

Contact
$Name: v3_1_0 $,$Revision: 1.9 $, Last modified: Thursday, September 14, 2006 at 9:54:32 PM

Figure 7. Job History page showing all jobs submitted to target DR6. Each job listing shows the job parameters and outcome and includes an Info link to view details and resubmit the job if necessary.

Canceling Jobs

The jobs service also ensures that jobs don’t exceed the time allotted for each queue by canceling the query when the time limit is reached. A user (or administrator) might also cancel a query at any time by pressing the Cancel button on the Job Info page.

If a job takes longer than the time limit for that queue, it’s marked for cancellation as soon as the limit is reached. Thus, quick queries return a timeout message at close to 60 seconds, but it can sometimes take a while to cancel a job. For example, a user might run a query for two seconds, then cancel it and find that the cancel itself takes 10 to 20 seconds. This usually happens because cancellation requires rolling back a transaction. Jobs stay in the canceling state for up to five minutes, after which the process is abandoned.

Canceling a job takes up a queue slot in the jobs queue, so a job that takes a while to cancel will show up in the jobs queue in canceling state. Other than this, canceling a job doesn’t affect the user’s ability to run other jobs or access tables in their MyDBs.

Once the job is actually canceled, it is marked as canceled. Jobs that time out are canceled, while jobs that failed for other reasons are marked as failed.

Job History and Info

The Job History page (which is accessible from the History link in the menu in Figure 7) provides a job history for all the jobs a user ever submitted. The job history is chronologically ordered in reverse order (latest first) and is searchable so that users can look for a job with a particular name or pattern.

A particular job’s full history is available by clicking the Info link in that job’s history listing. Users can resubmit a job with or without modification from the history page. This history provides a permanent record of queries a given user submits; it’s retrievable at any time, even after users delete the table the job created (and hence the query associated with it).

CasJobs Administration

Administration tasks for CasJobs include managing user accounts and privileges, MyDB databases, user groups, and archiving query and output

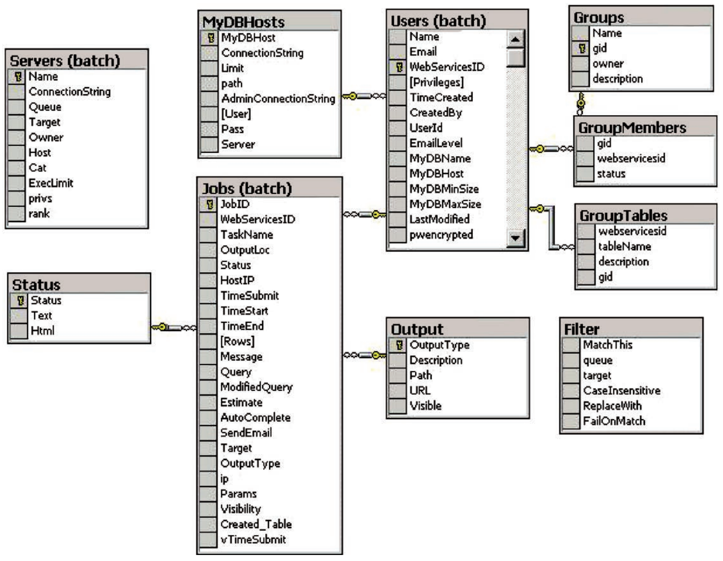


Figure 8. BatchAdmin database schema. The tables in this database contain the information to manage CasJobs sessions, contexts, users, jobs, and MyDBs.

jobs. Most of this is automated and managed by a special administration database that we call BatchAdmin.

BatchAdmin Database

The BatchAdmin database contains all the information necessary for CasJobs's day-to-day system administration. Each of the following is a single table in the BatchAdmin database; unless otherwise indicated, the default owner in SQL Server, dbo, owns the tables.

1. The *Servers* table describes the contexts available to CasJobs users. This is where we maintain each context's privilege level, indicating which group of users can access it. The Servers table entry provides the mapping between a context name and its associated database (catalog) in the Cat column and the server that hosts that catalog in the Host column; the connection string to the database is in the ConnectionString column. The string in the Privs column specifies the privilege level necessary for this context; the groups that users belong to must include this privilege in the Groups table for users to see this context in their context menus. The server table is owned by the *batch user*, an SQL Server user who has administrator-level privileges to create user (MyDB) databases and run jobs on all the servers.
2. The *Users* table stores user profiles and is owned by the batch user. In addition to the user's name, email, Web services ID (WSID), and user ID (all of which must be unique), this table contains privileges and MyDB information such as the server on which the MyDB is hosted and MyDB size limits (MyDBHost, MyDBMinSize, and MyDBMaxSize columns). The pwencrypted column stores the user's password in encrypted form to protect user privacy. Other profile information kept here includes the email notification level, which controls the messages sent to the user in response to changes in job status or other events. Users can opt to turn off job notification by selecting the admin-only notification level or select whether they want to be notified upon job completion, failure, and so on.
3. The *Jobs* table (also owned by the batch user) contains an entry for each job submitted in the CasJobs system. This includes query and output jobs, regardless of whether they're successful. The *Status* table defines the meaning of each numerical job status code.
4. The *Groups*, *GroupMembers*, and *GroupTables* tables are linked to the Users table and manage the functionality that lets users share tables with other users. Each group has an entry in the Groups table, each user who is part of a group has an entry in the GroupMembers table, and each shared table has an entry in the GroupTables table.
5. The *MyDBHosts* table contains information on all servers currently hosting MyDBs. The MyDBHost column in the Users table points to an entry in the MyDBHosts table.
6. The *Filter* table lists the filters to apply to user-submitted SQL queries, mainly to remove any dangerous SQL commands (such as "drop table") from the queries for non-MyDB targets.

Figure 8 shows each table in the BatchAdmin schema and the relationships between them.

Administering the Jobs Service

The jobs service is a Windows service that must be installed using the Windows `installutil.exe` application. The administrator can name the service during installation and must set the login account for the service to a local or domain administrative account. The jobs service can run in one of three modes:

• *jobs*, which handles query jobs;

- *output*, which handles output jobs; or
- *both*, which handles both query and output jobs.

The service mode is set in the configuration file, `Jobsservice.exe.config`, via the service mode parameter, which has three values: 0 = jobs, 1 = output, and 2 = both.

The service mode lets the administrator separate the query and output jobs services and run them on different servers for better load balancing. It also allows query jobs to run independently from output jobs, which is preferable so that user queries don't have to be killed if the output queue gets stuck.

The connection string to the BatchAdmin server is also one of the configuration parameters. Only one jobs service instance of each service mode can point to a given BatchAdmin server—that is, only one jobs service or one output service can point to a given BatchAdmin server at any given time. Hence, multiple jobs services can coexist on a single Web server as long as each is either pointing to a different BatchAdmin server or is in a different service mode.

Managing MyDBs and User Accounts

Most of the day-to-day administration of MyDBs and users is done via the CasJobs Admin page, visible only to administrative-level users (determined by the "batch" privilege in the Users table). This page shows the currently active queries and output jobs, analogous to the Queries page for normal users. It also includes a search window to search for users by user ID and name. The administrator can load a particular user's profile into the page using the Load User button and then update it as necessary to include MyDB size and password.

Each MyDB host must have an entry in the MyDBHosts table that provides a mapping between the logical and physical MyDB server names and their connection parameters.

CasJobs under the Hood

Now let's look briefly at the CasJobs implementation—in particular, the Web application and the jobs service—and how we achieve load balancing through the distributed server configuration. The CasJobs interface we've illustrated in this article is the SDSS version (http://cas.sdss.org/casjobs), but it can be customized for any archive. A Windows application—the jobs service—manages the CasJobs queues. The Web application and jobs service interface with several different CasJobs Web services:

- user services (http://casjobs.sdss.org/casjobs/CasUsers.asmx);
- job services (http://cas.sdss.org/casjobs/CasJobs.asmx), not to be confused with the jobs service application that we describe in this section;
- extraction services (http://casjobs.sdss.org/casjobs/TableJobs.asmx); and
- groups services (http://casjobs.sdss.org/casjobs/CasGroups.asmx).

Anyone can use the Web service descriptions for these services (such as http://casjobs.sdss.org/casjobs/casjobs.asmx?WSDL) to build their own application that interfaces with the CasJobs Web services.

Web Application

The CasJobs Web application's back end interfaces with SQL database servers to execute user queries in a distributed manner. The CasJobs Web site is based on a set of SOAP services, so any user can access these services directly using a SOAP toolkit in his or her preferred programming language. At the Johns Hopkins University, we've successfully used Python, Java (Axis), and C# clients for Web services, but others have written Perl clients as well. More information is available at the International Virtual Observatory Alliance's (IVOA's) Web site (www.ivoa.net).

We have developed a command-line package for accessing CasJobs in Java that serves as an example of how to programmatically access the services provided by CasJobs. It's also a useful tool for demonstrating the interoperation between .NET and Apache Axis.

CasJobs Web services are stateless. We don't use any Web service or grid technology to track state in SOAP calls: each one could be answered by any server hosting the CasJobs software that has access to the CasJobs database. CasJobs also has a database of jobs belonging to a particular user, so it can manage privileges and authorization on an individual basis. We choose, however, to follow the template of most e-commerce sites and implement state within our system using database technology, much as e-commerce shopping carts are generally implemented. In each call, the user's WSID is passed as a parameter; internally, we use it to set the context of the message and find the state in the SQL database. Hence, the service is "stateful" but without any library overhead.

Jobs Service

Apart from short jobs, everything in the CasJobs system is asynchronous and requires job tracking. Each job has an entry in the BatchAdmin Jobs

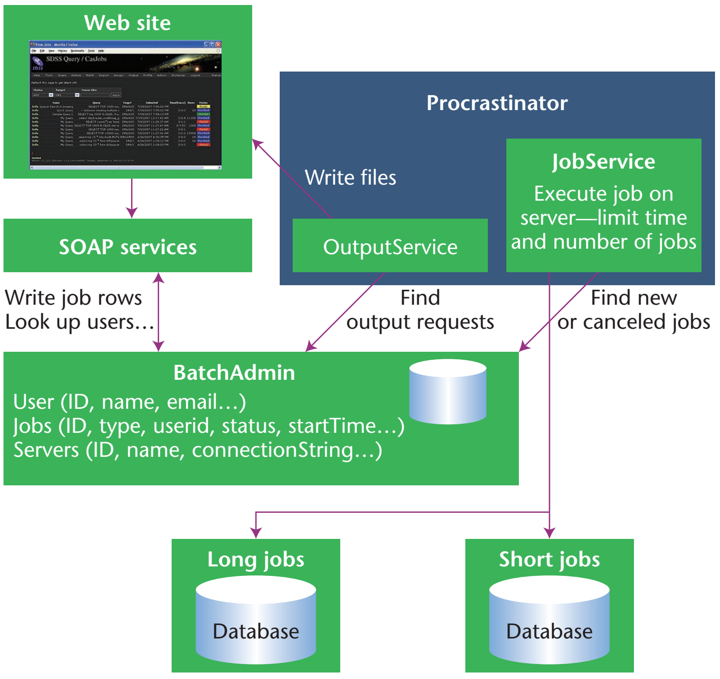


Figure 9. Jobs service implementation using the Windows procrastinator service. The jobs service runs continuously and executes user query and output jobs.

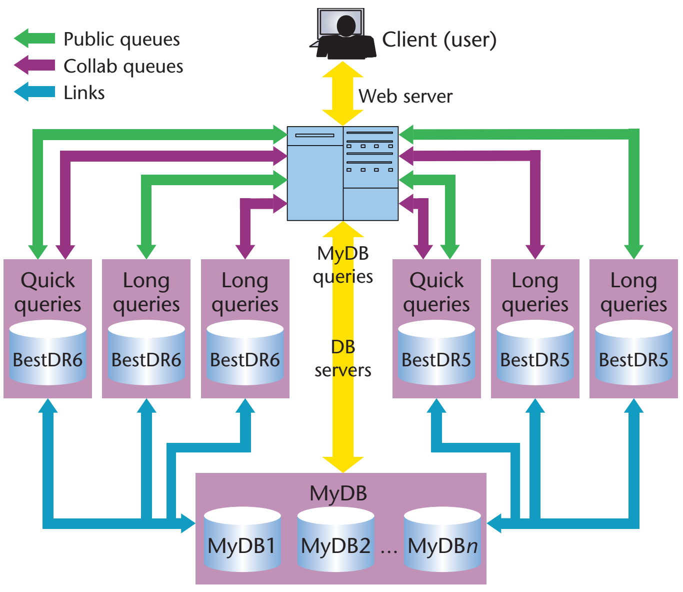


Figure 10. Distributed query execution and load balancing in CasJobs with linked servers. Queries are divided between multiple copies of data sets hosted on different servers by creating more than one context for each data set. Quick-mode queries and Submit-mode (batch) queries are sent to different servers in order to maintain adequate performance.

table—simply submitting a job creates this entry.

A Windows service (the procrastinator) runs a thread that wakes up periodically and scans the Jobs table for new jobs for each target specified in the Servers table (see Figure 9). If a server isn't running its allowed number of jobs, a thread spins off for the job; the thread includes a timer that cancels the query and closes the connection if the job runs longer than the specified queue length. The entry in the Jobs table is updated to show the job has started and will be updated once more to show whether it completed or failed. At this point, the service will also email users concerning the job if they've selected that option in their profile. This thread also checks for jobs marked canceled in the Jobs table and stops them.

The procrastinator runs a separate thread for file export (output) jobs. It creates output files (sequentially), updates job entries, and scans the HTTP directory in which files are written and removes files older than a configurable time (currently one week).

Load Balancing with Distributed Query Execution

CasJobs uses the SQL-Server-linked server mechanism to execute queries remotely on different database servers. We link all servers bidirectionally, to allow a query running on one server to read or write results to a different server (for example, execute a query on a SDSS database server and write results to a table on the MyDB server). The query is executed on the server hosting the context to which the query was submitted.

This also lets us achieve a crude but functional form of parallelism and load balancing by providing multiple copies of a given database on different servers. Each copy is a different context in the query page, so users can submit queries to multiple contexts for a given data set, thereby distributing the load. This can be scaled out as necessary by simply adding servers with more copies and creating contexts for them.

Figure 10 shows how this works. For a given SDSS data set—here, DR6—we achieve load balancing by making copies of the BestDR6 database on three servers and providing each copy as a separate context in the CasJobs query page. Users can select one of the DR6 contexts to which to submit their query. Generally, each data set has at least one public and one collaboration-restricted context. Different data sets are usually on different servers, so queries directed at the DR5 data set, for instance, won't contend with queries against the DR6 data set; the MyDBs are on their own server(s). In particular, we use this feature to separate short (quick) and long (submit) queries. Multiple quick query queues might point to a given server, but usually only one long query queue points to one server. Otherwise, the query performance can degrade significantly.

CasJobs has become the mainstay and workhorse of the SDSS data-access system. It's in use at SDSS mirror sites worldwide and has also been adapted for non-SDSS astronomical archives, such as Galaxy Evolution Explorer (GALEX), Palomar Quest, and the upcoming Panoramic Survey Telescope and Rapid Response System (Pan-STARRS) PS1 archive, as well as for non-astronomy applications that contain hydrology sensor data such as AmeriFlux. The technologies that CasJobs employs—asynchronous query execution and personal server-side database access—are universally applicable and inescapable for query management in very large online databases.

References

1. A. Szalay et al., "The SDSS SkyServer - Public Access to the Sloan Digital Sky Server Data," *Proc. 2002 ACM SIGMOD Int'l Conf. Management of Data*, ACM Press, 2002, pp. 570–581.
2. *The SDSS SkyServer - Public Access to the Sloan Digital Sky Server Data*, tech. report MSR-TR-2001-104, 2001; Microsoft Research, http://research.microsoft.com/research/pubs/view.aspx?msr_tr_id=MSR-TR-2001-104.
3. A.R. Thakar et al., "The Sloan Digital Sky Survey Science Archive: Migrating a Multi-Terabyte Astronomical Archive from Object to Relational DBMS," *Computing in Science & Eng.*, vol. 5, no. 5, 2003, pp. 16–29.
4. W. O'Mullane et al., *Proc. IEEE Int'l Conf. Web Services* (ICWS), vol. 1, IEEE Press, 2005, pp. 33–40.

Nolan Li *is a doctoral student in the departments of computer science and physics and astronomy at the Johns Hopkins University. His work and research interests focus on Web services and innovative solutions for efficient access to large scientific databases. Li has a BS in computer science from the Johns Hopkins University. Contact him at nli@pha.jhu.edu.*

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

The magazine of computational tools and methods for 21st century science

ALSO Building Better Search Engines, p. 7
Why Fortran? p. 68
Designing a Cluster, p. 72
July/August 2007
Computing in SCIENCE & ENGINEERING
Computing in Science & Engineering is a peer-reviewed, joint publication of the IEEE Computer Society and the American Institute of Physics
COMPUTING CLIMATE CHANGE

Interdisciplinary

Communicates to those at the intersection of science, engineering, computing, and mathematics

Emphasizes real-world applications and modern problem-solving

Top-flight departments in each issue!

- Book Reviews
- Computer Simulations
- Education
- News
- Scientific Programming
- Technologies
- Views and Opinions
- Visualization Corner

Peer-reviewed topics

2007		2008	
Jan/Feb	Anatomic Rendering	Jan/Feb	SSDS Science Archive
Mar/Apr	Stochastic Modeling	Mar/Apr	Combinatorics in Computing
May/Jun	Python: Batteries Included	May/Jun	Computational Provenance
Jul/Aug	Climate Modeling	Jul/Aug	High-Performance Computing in Education
Sep/Oct	Computational Wizardries	Sep/Oct	Novel Architectures
Nov/Dec	High-Performance Computing Defense Applications	Nov/Dec	Computational Astronomy

MEMBERS
$45/year
for print and online

Subscribe to CiSE online at http://cise.aip.org and www.computer.org/cise

IEEE IEEE computer society AMERICAN INSTITUTE OF PHYSICS