Astronomical Data Analysis Software and Systems XIII
ASP Conference Series, Vol. 314, 2004
F. Ochsenbein, M. Allen, and D. Egret, eds.

From FITS to SQL - Loading and Publishing the SDSS Data

Aniruddha R. Thakar and Alexander S. Szalay

Center for Astrophysical Sciences, The Johns Hopkins University, Baltimore, MD 21218, Email: thakar@pha.jhu.edu

Jim Gray

Microsoft Research

Abstract. For large astronomical databases like the SDSS Science Archive, data loading is potentially the most time-consuming and labor-intensive part of archive operations, and it is also the most critical: it is the last chance to examine the data before it is published. We attempted to automate this job as much as possible, and to make it easy to diagnose data and loading errors. We describe the **sqlLoader** — a distributed workflow system of modules that check, load, validate and publish the data to the databases. The workflow is described by a directed acyclic graph whose nodes are the processing modules. It is designed for parallel loading and is controlled from a web interface (Load Monitor). The validation stage represents a systematic and thorough scrubbing of the data. Finally, the different data products are merged into a set of linked tables that can be efficiently searched with specialized indices and pre-computed joins.

1. Introduction

The Sloan Digital Sky Survey Data Release 1 (DR1) contains a Terabyte of catalog data published online at http://skyserver.sdss.org/dr1. The total size of the catalog data including backup and archive copies is about 3-4 TB, and presents special challenges in terms of its storage and organization (Thakar et al. 2003) as well as loading and publishing it. The load/publish process is very time-consuming for Terabyte databases (several days to several weeks at current disk speeds), and data once publicly distributed cannot be taken offline or changed after it has been used for scientific publications.

Loading the data is therefore a critical step whose efficiency and success are paramount to the timely availability and correctness of the published data. However, in reality it is often the least automated, least budgeted and planned-for step in archive operations, and almost always takes far longer than anticipated.

Unfortunately, loading is not a one-time thing. Data errors cause reloads. Errors in the software pipeline cause reloads. Sometimes, the fastest way to redesign and reorganize the database is to simply reload it. So it is very important to have a fully automated load process that can reload the entire database in a few days with minimal supervision. Accordingly, we have developed an au-

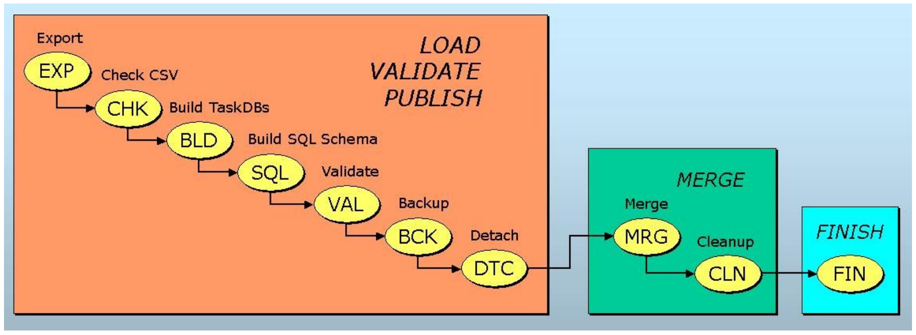


Figure 1. sqlLoader steps represented as a Directed Acyclic Graph.

tomated loading and publishing pipeline called **sqlLoader** that takes the data exported in FITS format by the SDSS Operational DB, converts it to CSV (comma-separated values) format, then validates and publishes it to the Microsoft SQL Server DBMS that contains our catalog data. The heart of the sqlLoader is a distributed workflow system of modules described by a directed acyclic graph (DAG) (Figure 1).

2. Loading Process

The basic processing entity is a *task* A task is started when a data *chunk* is exported by the OpDB. Exported chunks are converted to CSV format, and are contained in a single directory. There are several different export types — TARGET, BEST, RUNS, PLATE and TILING — each of which is subjected to different tests and destined for a different database. Each task comes with a taskID that is unique within its category.

The loading process consists of *steps.* The first step is to **load** each chunk of data into a separate task DB, containing only a thin set of indices. Then we **validate** the data. This includes verifying that there are no primary key collisions and all foreign keys point to a valid record. As we find problems downstream we add more and more tests for data ranges, sanity checks, and cross-checks on cardinalities. We build several ancillary tables for spatial searches (HTM, Neighbors, etc.) After the validation step we **publish** the data: we perform a DB-to-DB copy, where the target is the final production database. After publishing, we make a **backup** of the task DB. At the very end, all the different datasets (databases) are **merged** into the final (published) database in the finish step, and indices are created for efficient data mining.

3. Validation

Validation is perhaps the most important step in the loading process. The speed, integrity and convenience that databases offer come at a price: data once published cannot be retracted or changed. The data must always be available once science has been done with it, and it must be correct because publications are based on it. The validate step in sqlLoader represents a systematic scrubbing and sanity-check of the data, from a scientific as well as data integrity point

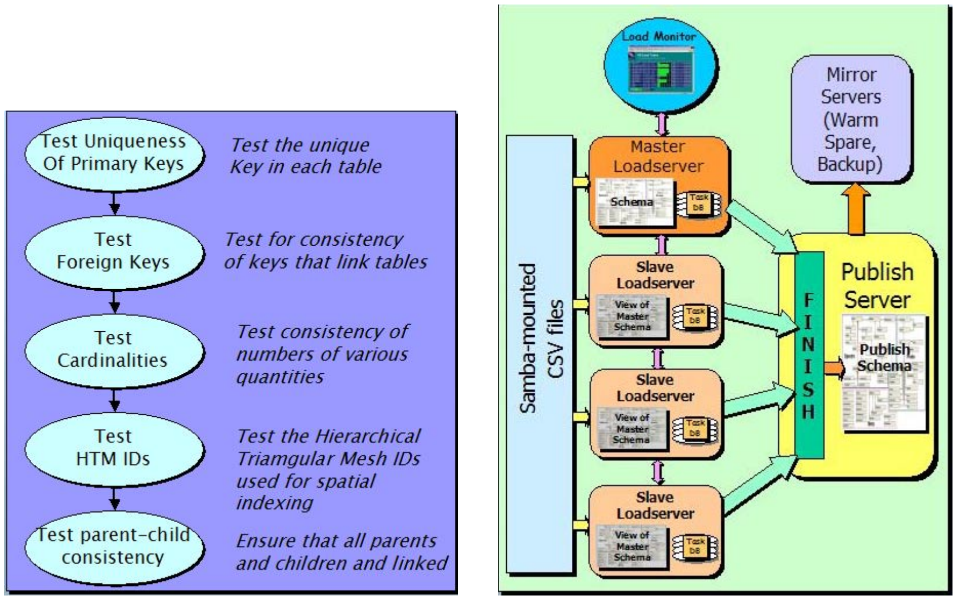

Test Uniqueness Of Primary Keys
Test the unique Key in each table
Test Foreign Keys
Test for consistency of keys that link tables
Test Cardinalities
Test consistency of numbers of various quantities
Test HTM IDs
Test the Hierarchical Triamgular Mesh IDs used for spatial indexing
Test parent-child consistency
Ensure that all parents and children and linked
Load Monitor
Mirror Servers (Warm Spare, Backup)
Samba-mounted CSV files
Master Loadserver
Schema
Task DB
Slave Loadserver
View of Master Schema
FINISH
Publish Server
Publish Schema

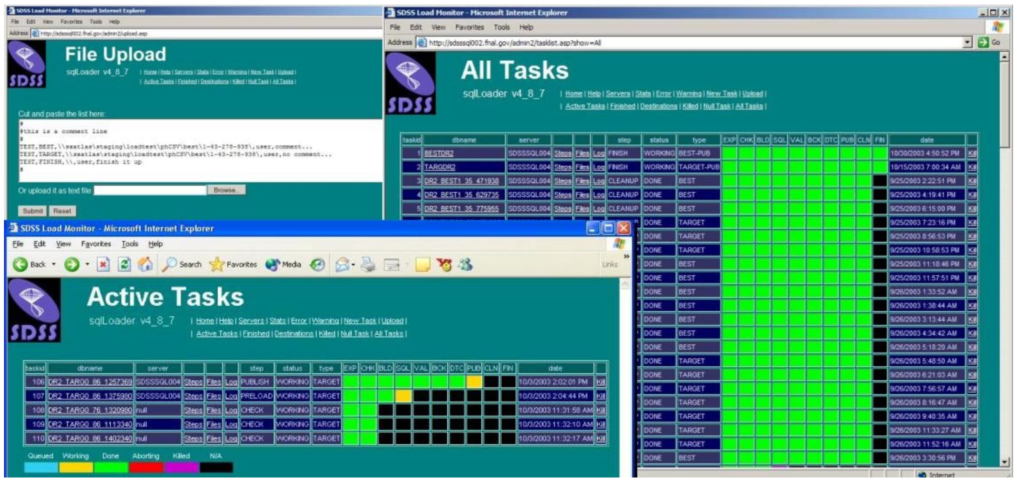
Figure 2. (a) Data validation steps and (b) Distributed Loading.

of view. Figure 2(a) shows the various operations that are performed on the data. The primary and foreign key tests are run on all the tables. The photo (imaging) and spectro tables are tested for HTM IDs, which are 64-bit IDs used by the Hierarchical Triangular Mesh indexing scheme. The referential integrity of parent-child relationships for deblended image objects is also checked. Finally, the consistency of counts of various quantities is checked.

This validation process has proven invaluable in finding numerous inconsistencies and errors in the data and catching them early, during the testing of DR1 rather than after the data is published. If a chunk fails somewhere in the load process, the operator can kill the operation and restart it. Every step of the pipeline generates a 3-level detailed log that records the operation, how long it took and how many errors were detected. This allows the operators to quickly identify data quality problems, performance problems, and “stuck” tasks.

4. Distributed Loading

Loading a Terabyte or more of data is a time-consuming process even with fast disks, and parallelization of the loading steps is a big help, especially as we get into the multi-TB data volumes of future SDSS releases. The load, validate and publish steps in the sqlLoader (leftmost rectangle in Fig. 1) are fully parallelizable and can be executed on a distributed cluster of load-servers. Figure 2(b) shows the setup for parallel distributed loading. The master schema is on the loadadmin (master) server, with each loadsupport (slave) having a remote *view* of the schema. Distributed loading makes use of advanced DBMS features like linked servers with distributed views and distributed transactions. After loading, validating and publishing is done in parallel, the merging of the parallel data streams and the finish step are performed sequentially on the publish server.


SDSS Load Monitor - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://sdsssql002.fnal.gov/admin2/upload.asp
File Upload
sqlLoader v4_8_7
| Home | Help | Servers | Stats | Error | Warning | New Task | Upload |
| Active Tasks | Finished | Destinations | Killed | Null Task | All Tasks |
Cut and paste the list here:

```
#
#this is a comment line
#
TEST,BEST,\\sxatlas\staging\loadtest\phCSV\best\1-43-278-938\,user,comment...
TEST,TARGET,\\sxatlas\staging\loadtest\phCSV\best\1-43-278-938\,user,no comment...
TEST,FINISH,\\,user,finish it up
#
```

Or upload it as text file
Browse...
Submit
Reset
SDSS Load Monitor - Microsoft Internet Explorer
Address http://sdsssql002.fnal.gov/admin2/tasklist.asp?show=All
Go
All Tasks
sqlLoader v4_8_7
| Home | Help | Servers | Stats | Error | Warning | New Task | Upload |
| Active Tasks | Finished | Destinations | Killed | Null Task | All Tasks |

taskid	dbname	server				step	status	type	EXP	CHK	BLD	SQL	VAL	BCK	DTC	PUB	CLN	FIN	date	
1	BESTDR2	SDSSSQL004	Steps	Files	Log	FINISH	WORKING	BEST-PUB	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	10/30/2003 4:50:52 PM	Kill
2	TARGDR2	SDSSSQL004	Steps	Files	Log	FINISH	WORKING	TARGET-PUB	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	10/15/2003 7:00:34 AM	Kill
3	DR2_BEST1_35_471938	SDSSSQL004	Steps	Files	Log	CLEANUP	DONE	BEST	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	⬛	9/25/2003 2:22:51 PM	Kill
4	DR2_BEST1_35_629735	SDSSSQL004	Steps	Files	Log	CLEANUP	DONE	BEST	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	⬛	9/25/2003 4:19:41 PM	Kill
5	DR2_BEST1_35_775955	SDSSSQL004	Steps	Files	Log	CLEANUP	DONE	BEST	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	⬛	9/25/2003 6:15:00 PM	Kill
							DONE	TARGET	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	⬛	9/25/2003 7:23:16 PM	Kill
							DONE	TARGET	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	⬛	9/25/2003 8:56:53 PM	Kill
							DONE	TARGET	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	⬛	9/25/2003 10:58:53 PM	Kill
							DONE	BEST	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	⬛	9/25/2003 11:18:46 PM	Kill
							DONE	BEST	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	⬛	9/25/2003 11:57:51 PM	Kill
							DONE	BEST	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	⬛	9/26/2003 1:33:52 AM	Kill
							DONE	BEST	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	⬛	9/26/2003 1:38:44 AM	Kill
							DONE	BEST	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	⬛	9/26/2003 3:13:44 AM	Kill
							DONE	BEST	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	⬛	9/26/2003 4:34:42 AM	Kill
							DONE	BEST	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	⬛	9/26/2003 5:18:20 AM	Kill
							DONE	TARGET	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	⬛	9/26/2003 5:48:50 AM	Kill
							DONE	TARGET	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	⬛	9/26/2003 6:21:03 AM	Kill
							DONE	TARGET	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	⬛	9/26/2003 7:56:57 AM	Kill
							DONE	TARGET	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	⬛	9/26/2003 8:16:47 AM	Kill
							DONE	TARGET	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	⬛	9/26/2003 9:40:35 AM	Kill
							DONE	TARGET	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	⬛	9/26/2003 11:33:27 AM	Kill
							DONE	TARGET	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	⬛	9/26/2003 11:52:16 AM	Kill
							DONE	BEST	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	⬛	9/26/2003 3:30:56 PM	Kill

Internet
SDSS Load Monitor - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Media
Links
Active Tasks
sqlLoader v4_8_7
| Home | Help | Servers | Stats | Error | Warning | New Task | Upload |
| Active Tasks | Finished | Destinations | Killed | Null Task | All Tasks |

taskid	dbname	server				step	status	type	EXP	CHK	BLD	SQL	VAL	BCK	DTC	PUB	CLN	FIN	date	
106	DR2_TARG0_86_1257369	SDSSSQL004	Steps	Files	Log	PUBLISH	WORKING	TARGET	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟨	⬛	⬛	10/3/2003 2:02:01 PM	Kill
107	DR2_TARG0_86_1375980	SDSSSQL004	Steps	Files	Log	PRELOAD	WORKING	TARGET	🟩	🟩	🟩	🟨	⬛	⬛	⬛	⬛	⬛	⬛	10/3/2003 2:04:44 PM	Kill
108	DR2_TARG0_76_1320980	null	Steps	Files	Log	CHECK	WORKING	TARGET	🟩	🟩	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	10/3/2003 11:31:58 AM	Kill
109	DR2_TARG0_86_1113340	null	Steps	Files	Log	CHECK	WORKING	TARGET	🟩	🟩	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	10/3/2003 11:32:10 AM	Kill
110	DR2_TARG0_86_1402340	null	Steps	Files	Log	CHECK	WORKING	TARGET	🟩	🟩	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	10/3/2003 11:32:17 AM	Kill

Queued	Working	Done	Aborting	Killed	N/A
🟦	🟨	🟩	🟥	🟪	⬛

Figure 3. Loader Monitor screens showing All Tasks, Active Tasks and File Upload pages.

5. The Load Monitor

The Load Monitor is the admin web interface to the sqlLoader. It enables job submission, control and tracking via a user-friendly GUI. Loading jobs (tasks) can be submitted either a single chunk at a time or bulk-uploaded with the File Upload feature. Tasks can be monitored at several levels, and information is available on the status of the individual files being loaded, detailed logs of each step in the task, and separate listings of errors and warnings encountered. Sample Load Monitor screens are shown in Figure 3 to illustrate the features that are available. The Active Tasks listing shows all the tasks that are currently executing. The task status is displayed as a color code for each step (export, check, build, validate, backup, detach, merge, cleanup and finish). Amber means the step is in progress, green means it is done, red means it failed and purple means that the task was killed in that step. The Servers page shows status of each server and enables individual servers to be started and stopped.

6. Conclusions

The sqlLoader pipeline is a distributed workflow system that automates the loading and publishing of the SDSS data. It has enabled the loading for DR1 to be completed largely as a turnkey operation with very little human intervention.

References

Thakar, A.R., Szalay, A.S., vandenBerg, J.V. & Gray, J. 2003, in ASP Conf. Ser., Vol. 295, ADASS XII, ed. H. E. Payne, R. I. Jedrzejewski, & R. N. Hook (San Francisco: ASP), 217.