LARGE SYNOPTIC SURVEY TELESCOPE

Large Synoptic Survey Telescope (LSST)

LDM-503-1 (WISE Data Loaded in PDAC) Test Report

Gregory P. Dubois-Felsmann, Xiuqin Wu

DMTR-52

Latest Revision: 2018-06-19

D R A F T

Abstract

This is the test report for LDM-503-1 (WISE Data Loaded in PDAC), an LSST DM level 2 milestone pertaining to the LSST Science Platform, with tests performed according to LSP-00, Portal and API Aspect Deployment of a Wide-Area Dataset.

Change Record

Version	Date	Description	Owner name
	2018-06-12	Draft of complete report	G. Dubois-Felsmann
1.0	2018-06-18	Approved in RFC-494	J. Swinbank

Document curator: Gregory P. Dubois-Felsmann
Document source location: `https://github.com/lsst-dm/DMTR-52`
Version from source repository: 73d21db

Contents

LDM-503-1 (WISE Data Loaded in PDAC) Test Report

1 Introduction

1.1 Objectives

The LDM-503-1 milestone calls for the principal WISE primary mission and NEOWISE first-year catalogs to be loaded into the LSP Integration Environment, also known as the Prototype Data Access Center. The LSP-00 test specification, defined in LDM-540, was used to verify that the data were loaded and that the data service provided meets a subset of requirements for functionality of the API Aspect and the Portal Aspect of the Science Platform.

1.2 Scope

The LSP-00 test specification describes a comprehensive set of tests exploring the functionality of the Science Platform as applied to catalog and image data. The following test cases were carried out as part of the verification of this milestone:

LSP-00-00 Verification of the presence of the expected WISE data

LSP-00-05 Demonstration of low-volume and/or indexed queries against the WISE data via API

LSP-00-10 Demonstration of table-scan queries against the WISE data via API

LSP-00-15 Execution of basic catalog queries in the Portal

LSP-00-20 Operation of the UI for interaction with tabular data results

LSP-00-35 Linkage of catalog query results to related catalog data

The LSP-00-25 and LSP-00-30 test cases are not included in this report, as they address image data functionality and the WISE data load to PDAC in the end did not include the image data. The decision to avoid loading the image data was taken when the time required to load the WISE catalog data turned out to be larger than originally expected, and also required unanticipated hardware upgrades. In addition to the time required for loading the WISE image data,

providing a useful image cutout service would have required developing an LSST-stack (`afw`) camera model for WISE, and this was judged to be an effort with very limited benefit to DM development. The key usefulness of the WISE data in the DM context is as a scientifically realistic scaling test for Qserv, requiring the handling of tens of billions of rows, so the priorities for the data loading were set accordingly.

The image-oriented test cases can be performed against the SDSS data previously loaded in the PDAC, and will be executed and documented separately.[1]

The WISE catalog data to be loaded included the following tables:

- AllWISE Source Catalog: this catalog is comparable to the projected LSST `Object` table, arising from source detection performed on the coaddition of all the data from the pre-hibernation year of WISE operations, including the primary 4-band data as well as shorter periods of 3-band (W1,W2,W3) and 2-band (W1,W2) data taken after the exhaustion of the stored cryogen.
- AllWISE Multi-Epoch Photometry (MEP) Table: roughly comparable to LSST `ForcedSource`, arising from forced photometry on every single-epoch image, based on the sources in the AllWISE Source Catalog. Differs from `ForcedSource` in that WISE takes all four band images at once, so every row in this table contains data from all four bands.
- Single-Epoch Photometry tables from four eras of the mission, comparable to LSST `Source`. Similarly to the MEP Table, the WISE imaging model produces measurements in each available band in each row. As the W4 and then W3 bands were shut down as the spacecraft warmed up, their columns were dropped from the table, leading to an evolution of the schema with time. The four eras and the associated official table names are:
  - All-Sky Single Exposure (L1b) Source Table, from the primary cold mission, with all four bands operating;
  - 3-Band Cryo Single Exposure (L1b) Source Table, from the brief period when the spacecraft was still cold enough to collect W3, but not W4, data;
  - Post-Cryo Single Exposure (L1b) Source Table, containing only W1 and W2 data, from the period following the primary mission when the cryogen was fully ex-

[1] As a user convenience, the Portal Aspect in PDAC does provide some limited access to WISE image data directly from IRSA, but as it does not use LSST interfaces to do so, it is not included in this test.

hausted but operations were continued, after which the spacecraft was put into hibernation until 2013;

- NEOWISE-R Year 1 Single Exposure Source Table, from the first year of operations after the reactivation of the spacecraft, and containing only W1 and W2 data.

The LSP-00 tests verified that these tables were present on the PDAC and exercised a variety of operations against them in the Portal and API Aspects. The rest of this report covers the six test cases into which this work was divided.

1.3 System Overview

The LSST Science Platform is the part of the LSST Data Management System which presents the LSST Data Products, a user computational environment, and a set of tools for interacting with them, to LSST's science users.

The system under test is the initial deployment of the LSST Science Platform in the Prototype Data Access Center (PDAC), an initial application of the NCSA-provided "integration environment". In the tested version of the PDAC, elements of the database systems (including a Qserv prototype), a partial deployment of the API Aspect data access Web services (DAX), and an early version of the Science User Interface and Tools (SUIT) Portal were operated on a hardware cluster managed by NCSA. The Notebook Aspect was not included in this deployment at all.

Software deployment was done essentially manually on the systems provided by NCSA, in some cases by traditional software installation and in some using Docker containers, but no Kubernetes layer was involved.

1.4 Applicable Documents

LDM-294	LSST DM Project Management Plan
LDM-503	DM Test Plan
LDM-540	LSST Science Platform Test Specification

2 Test Configuration

2.1 Documents

The v1.0 version of LDM-540 was used to guide the tests. In the process of performing the tests some editing errors were found in LDM-540 and corrections will be submitted. These mainly affected LSP-00-35, where, due to an editing error, the test was described as being on the SDSS Stripe 82 data, not the WISE data.

2.2 Hardware

The tests were carried out on the initial PDAC hardware configuration at NCSA with its late 2017 upgrade to a more capable server with solid state disk for the Qserv czar node.

2.3 Software

A variety of versions of Qserv and DAX software were used in the API Aspect as issues were discovered and corrected. The final versions, on which all of the tests reported here were repeated, were:

The Portal Aspect used the following software versions, installed in PDAC on January 24, 2018 and retained throughout the testing:

- The `firefly` back end package: SHA be6cc56; and
- the `suit` LSST-specific configuration package: SHA 4298ba0.

3 Personnel

The majority of the test cases were executed by Gregory Dubois-Felsmann (Caltech/IPAC). All of LSP-00-20 and portions of LSP-00-15 and LSP-00-35 were also executed by Xiuqin Wu (Caltech/IPAC), from whose work most of the screen shots used arise.

3.1 Summary Table

TEST CASE ID	PASS/FAIL	COMMENTS

LSP-00-00	Pass	Refer to §4.
LSP-00-05	Pass	Refer to §5.
LSP-00-10	Pass	Refer to §6.
LSP-00-15	Pass	Refer to §7.
LSP-00-20	Pass with reservations	Refer to §8.
LSP-00-35	Pass	Refer to §9.

3.2 Overall Assessment

The test cases executed passed with a variety of minor issues relating to the details of the data import and features not fully implemented in this early version of the Science Platform, as well as to some rather arbitrary numeric limits in the deployed portal. The latter affected primarily LSP-00-20 and are described in more detail there. Overall the test showed that it was possible to import a several $\times 10^8$ row dataset and access it usefully through Qserv and the Portal and API Aspects.

3.3 Impact of Test Environment

As noted above, the deployment of the database, API Aspect, and Portal Aspect components did not use Kubernetes and is not reflective of the planned final system. This affected not only the management of the components but substantive technical aspects of their behavior; in particular, in the Kubernetes deployment there will be ingress controllers and other components that affect the connections between components.

Furthermore, the external access control to the components was through a VPN rather than by means of LSP-level authentication and authorization, and therefore most likely not reflective of parameters such the latency and throughput to be expected from the final system.

No tests of authorization and access control behaviors were performed.

3.4 Recommended Improvements

1. The current cone-search query size limits in the Portal are unreasonably small and Qserv would clearly support larger ones. The Portal's polygon search capability already per-

mits the searching of much larger regions with good performance, so the limitation seems unnecessary.

2. The as-deployed Portal limits 2D scatterplots to a maximum of 5,000 points before switching to 2D histograms. This was motivated by some concerns about earlier versions of the underlying `Plotly` library slowing down visibly for denser scatterplots. This fallback behavior is not clearly documented or signaled by the UI behavior, so the user will have some difficulty anticipating it and working around it. In IRSA production this limit has recently been raised somewhat, while still preserving reasonable performance for larger scatterplots. This change should be deployed immediately in LSST production. In addition, a recently developed alternate Web visualization implementation for large scatterplots, based on WebGL, has been developed for `Plotly`. This implementation is said to scale to over a million points. If it can be used in Firefly without other significant impacts, we strongly encourage that. Note that the Firefly tabular data model supports multi-million row tables already.

3. The UI functionality related to 2D histograms, called “heat maps”, should be improved to support more quantitative exploitation. In particular, the UI should permit the upper and lower bin limits to be specified by the user, not just the number of bins. If this worked just as it does for 1D histograms, that would be fine. This would help the user in obtaining simpler bin boundaries. Also, both types of histograms should provide support for writing out the histogram contents as a table.

4. The UI procedure for filtering on manual row selections is not intuitive and requires some attentive exploration in order to discover. This may need to be rethought.

5. The interaction of column filters with row selections needs to be rethought; in particular, row selections should not be cleared by the application of column filters.

6. The UI for applying rectangular selections to 2D plots could be improved in two ways: display moving guide marks along the X and Y axes as the selection is drawn, facilitating by-eye measurement of the quantitative region boundaries; and once a drawn-rectangle filter has been applied, its specifications should be accessible in the filter dialogs to allow recording the exact values used.

7. The PDAC TAP service does not support standard ADQL and required the use of Qserv-specific functions in order to exploit its performance successfully. We look forward to the implementation of at least a basic subset of ADQL’s geometry constructs.

8. The PDAC TAP service should support the VOTable data format. The service should be tested using publically available validators.

9. All tables should be readily searchable by their primary key with good performance. In the case of the WISE tables there is the oddity that the primary keys appear in both integer and string forms. For public usability of the WISE service we recommend ensuring that both forms are indexed as needed for good performance. The data model and database teams should remain alert to the possibility of a similar situation arising in the LSST data model.

10. We strongly recommend that there be both programmatic and visual interfaces that aid users in determining whether their queries have triggered, or are likely to trigger, table scans. This information should also be available from user-oriented documentation about the materialization of release data in the LSP.

Package	Tag
antlr	2.7.7.lsst1
apr	1.5.2
apr_util	1.5.4
base	14.0-8-g7f6dd6b+6
boost	1.66.0
db	14.0+14
doxygen	1.8.13.lsst1
flake8	3.5.0+2
flask	0.12+1
jemalloc	5.0.1.lsst1
libcurl	7.53.1
libevent	2.1.8
log	14.0-2-ga5af9b6+11
log4cxx	0.10.0.lsst7
lua	5.1.4.lsst1-2-gda519a9
mariadb	10.1.21.lsst2
mariadbclient	10.1.21.lsst2
mysqlproxy	0.8.5.lsst1
numpy	1.13.1
partition	12.1-1-g721d0e5+41
pep8_naming	0.4.1+1
pex_exceptions	14.0-1-g13ef843+10
protobuf	2.6.1.lsst4-1-gdbc86f2+1
pybind11	2.1.1
pycodestyle	2.3.1+1
pyflakes	1.6.0
pytest	3.2.0.lsst2
pytest_flake8	0.9.1+2
pytest_forked	0.2.lsst2
pytest_session2file	0.1.9+3
pytest_xdist	1.20.1.lsst2
python	0.0.7
python_execnet	1.4.1.lsst2
python_future	0.16.0+1
python_mccabe	0.6.1+3
python_mysqlclient	1.3.7.lsst1+10
python_psutil	5.4.3
qserv	tickets.DM-11743-g57c820d962
requests	2.9.1.lsst1+2
scisql	0.3.8.lsst1+2
scons	3.0.0.lsst1+1
sconsUtils	14.0-15-g79b7e05
sphgeom	12.1-4-g110c6f4+22
sqlalchemy	1.0.8.lsst3+4
utils	14.0-9-g11010eb+1
xrootd	master-g344ca073bb

TABLE 1: Database and DAX package versions, last deployed

4 LSP-00-00: Verification of the presence of the expected WISE data

The tests were performed primarily from an Apple laptop computer running OS X 10.11.6, wirelessly connected to the Internet and using the NCSA VPN at `vpn.ncsa.illinois.edu`. API Aspect queries were performed using a combination of the Python `requests` package and the `curl` command. Notebooks were executed using Jupyter versions up through 5.4.1 and Python version 3.6.1. The Portal was accessed using Safari 11.0.2.

4.1 API Aspect tests

The API Aspect tests made use of the "v0" `dbserv` service at

`http://lsst-qserv-dax01.ncsa.illinois.edu:5000/db/v0/tap/sync`.

This service accepts SQL queries, forwards them to Qserv for execution, and returns the results as JSON structures. The open-source Python *Requests* package[2] was used to perform the HTTP GET and POST operations required, and to parse the returned JSON structures into Python dictionaries.

4.1.1 Table presence verification

API Aspect `dbserv` queries (`SHOW DATABASES` and `SHOW TABLES`) were used to establish the set of databases available, and within those databases, identify the WISE catalog tables of interest for the tests.

4.1.2 Table Schemas

Schemas for the tables were obtained from `DESCRIBE` queries on `dbserv`. The schemas were compared programmatically against those obtained from the appropriate IRSA program interface:

[2] `http://github.com/requests/requests`

WISE catalog	PDAC database	PDAC table
AllWISE Source Catalog	`wise_00`	`allwise_p3as_psd`
AllWISE Multi-Epoch Photometry Table	`wise_00`	`allwise_p3as_mep`
All-Sky Single Exposure (L1b) Source Table	`wise_4band_00`	`allsky_4band_p1bs_psd`
3-Band Cryo Single Exposure (L1b) Source Table	`wise_3band_00`	`allsky_3band_p1bs_psd`
Post-Cryo Single Exposure (L1b) Source Table	`wise_2band_00`	`allsky_2band_p1bs_psd`
NEOWISE-R Year 1 Single Exposure Source Table	`neowiser_yr1_00`	`neowiser_yr1_p1bs_psd`

TABLE 3: WISE tables as discovered through the PDAC DAX service

`https://irsa.ipac.caltech.edu/cgi-bin/Gator/nph-dd?mode=xml&catalog=`*table*.

The results were parsed with the Python standard library class `xml.etree.ElementTree`. Comparison of the schemas was carried out in Python, without regard to the ordering of columns. The following discrepancies were observed (details are preserved in the test notebook):

- The `dec` column in each table was renamed to `decl` to avoid a conflict with a reserved word in the Qserv implementation database.
- The Qserv implementation adds special columns `chunkId` and `subChunkId` to each partitioned table.
- The original column names in the `wise_2band_00.allsky_2band_p1bs_psd` table all appear in the PDAC version converted to uppercase.
- For some tables, The IRSA-published schemas via the program interface have additional columns compared to what is displayed in the IRSA Gator UI and, apparently, in addition to what was included in the export files used to load the PDAC Qserv. This was assessed to be irrelevant to the objectives of this test, and no serious effort was made to determine the reasons in time for this report, as this would have required IRSA staff time that was in short supply. IRSA support tickets will be filed to ensure that the discrepancies are ultimately understood and, if possible and desirable, corrected. The affected tables are:

- `allwise_p3as_psd`: 36 columns appear in the IRSA program interface schema that do not appear in the PDAC table. Most of these are associated with a proper motion model fit.
- `allsky_4band_p1bs_psd`: 20 columns appear in the IRSA program interface schema that do not appear in the PDAC table. 16 of these seem to be intermediate data products of the photometry pipeline, including zero-point corrected PSF and aperture magnitudes and the zero points themselves.
- `neowiser_yr1_p1bs_psd` (compared against the IRSA program interface for the "live" table `neowiser_p1bs_psd`): 24 columns appear in the IRSA program interface schema that do not appear in the PDAC table. These again appear to be a set of intermediate data products.

4.1.3 Row counts

Row counts were obtained from `COUNT(*)` queries against all six tables. These queries require accessing all chunks and subchunks in the Qserv partitioning and therefore resulted in table scans taking several minutes of wall clock time each. The row counts and representative query durations are shown in Table 4. The pattern of query times has not been fully understood; `allwise_p3as_psd` is a notable outlier. The row counts were compared to those advertised in the WISE documentation at IRSA. They agree exactly except for the AllWISE Multi-Epoch Photometry Table, `allwise_p3as_mep`, for which 42,759,337,365 rows are reported in the IRSA documentation, a difference of 761,630,509 or 1.8%. This discrepancy has not yet been understood.

PDAC WISE table	Rows	`COUNT(*)` duration
`allwise_p3as_psd`	747,634,026	798s
`allwise_p3as_mep`	41,997,706,856	1457s
`allsky_4band_p1bs_psd`	9,479,433,101	993s
`allsky_3band_p1bs_psd`	3,703,319,374	278s
`allsky_2band_p1bs_psd`	7,337,642,955	535s
`neowiser_yr1_p1bs_psd`	18,468,575,586	759s

TABLE 4: Row counts in PDAC WISE tables, with durations of the associated `COUNT(*)` queries

4.2 Portal Aspect

The Portal Aspect was accessed in order to verify the accessibility of the WISE catalog data.

This work covers step 6 of the LSP-00-00 test case procedure.

4.2.1 Step 6a

The front page of the Portal Aspect for this test was:

`https://lsst-sui-proxy01.ncsa.illinois.edu/suit/`

4.2.2 Step 6b

The front page presents an "LSST Data" screen which offers the catalog data available for access, associated with three "projects": SDSS, AllWISE, and the WISE and NEOWISE single-epoch photometry.

The radio buttons under the project selector show the tables available for selection. All six WISE tables covered by this test specification are available, as seen in the following screen shots.

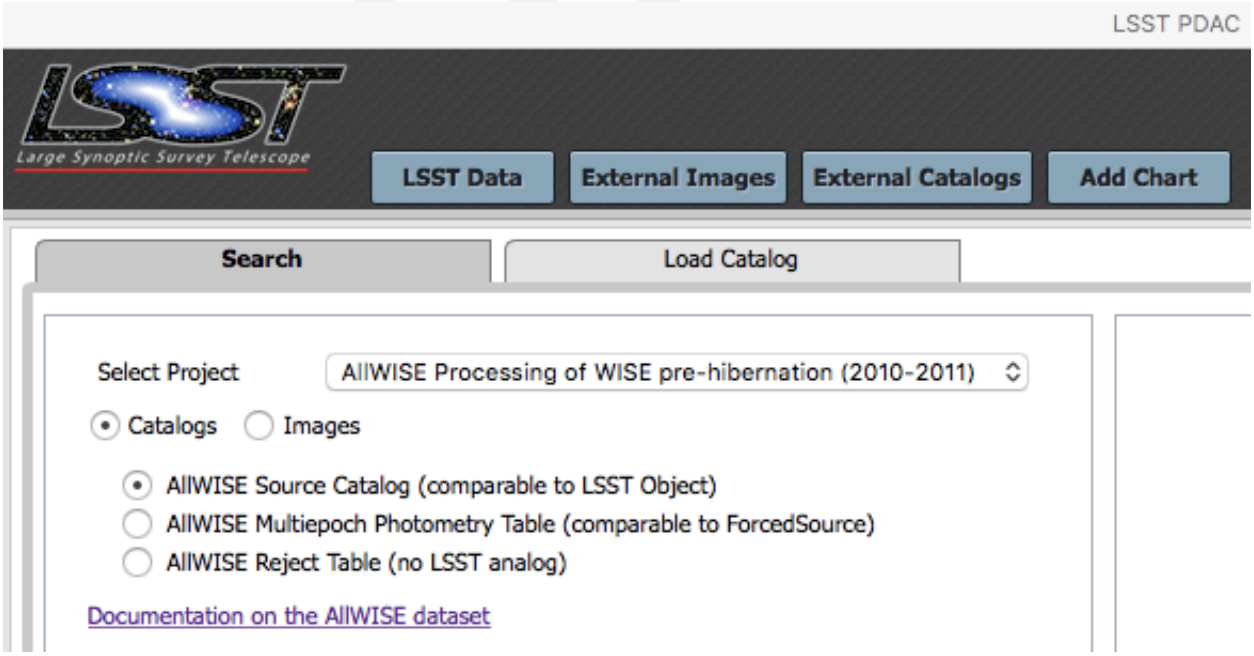

FIGURE 1: AllWISE tables available for query in the Portal

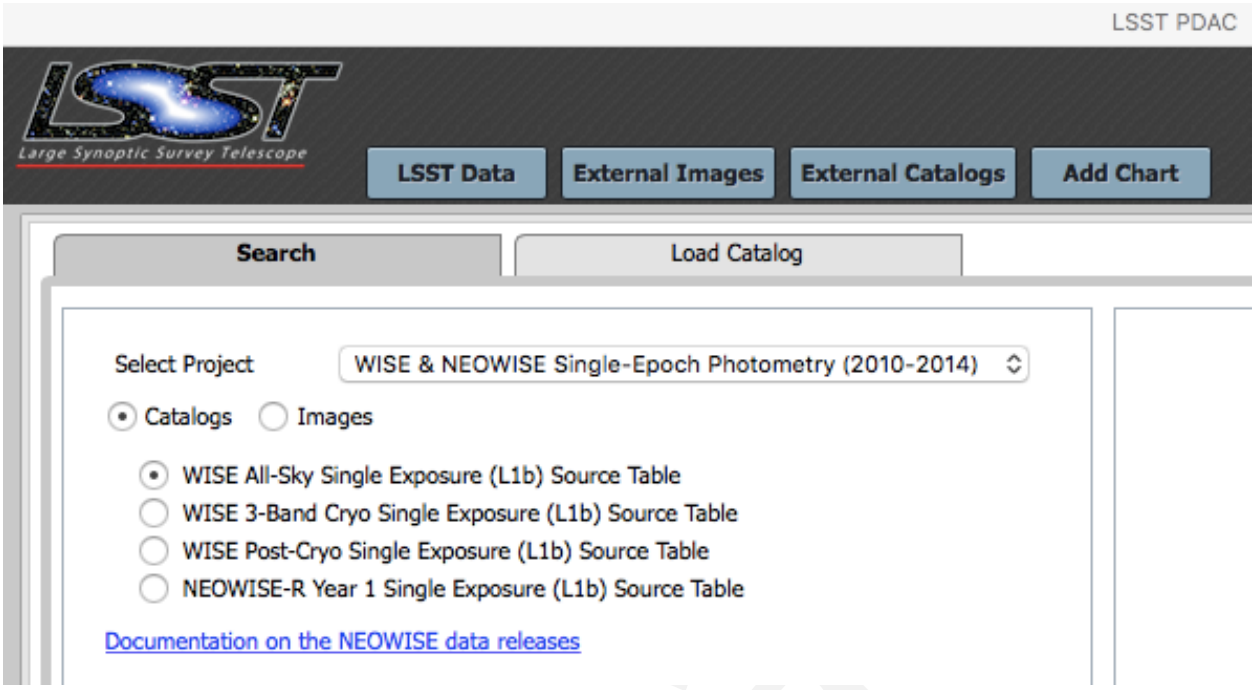

FIGURE 2: WISE and NEOWISE single-epoch photometry tables available for query in the Portal

Given a selected table, the UI presents a selector for the type of search to be performed, offering a variety of options for spatial region searches including cone, ellipse, box, and polygon, as well as a multi-object cone search and a free-form all-sky search. This UI element is shown in Figure 3 below.

4.2.3 Step 6c

The Portal also provides a UI element, shown in Figure 4 displaying the entire table schema in a scrolling region, with, among other features, the ability to apply restrictions to any column for application in the query to be performed. It was not a goal of this test to specifically validate the correct display of the schema, but some spot checks were performed and the results were as expected. For instance, the schema display was observed to follow the expected pattern of whether W3- and W4-related were available in the selected table(s). (The nature of the scrolling region provided for this selector inhibits easy capture of the entire content. It would be useful for the final Portal to provide a UI element facilitating download of the entire schema.) The schema anomalies reported in section 4.1.2 above were also observed here, e.g., the uppercase conversion of the original column names in the `allsky_2band_p1bs_psd` column.

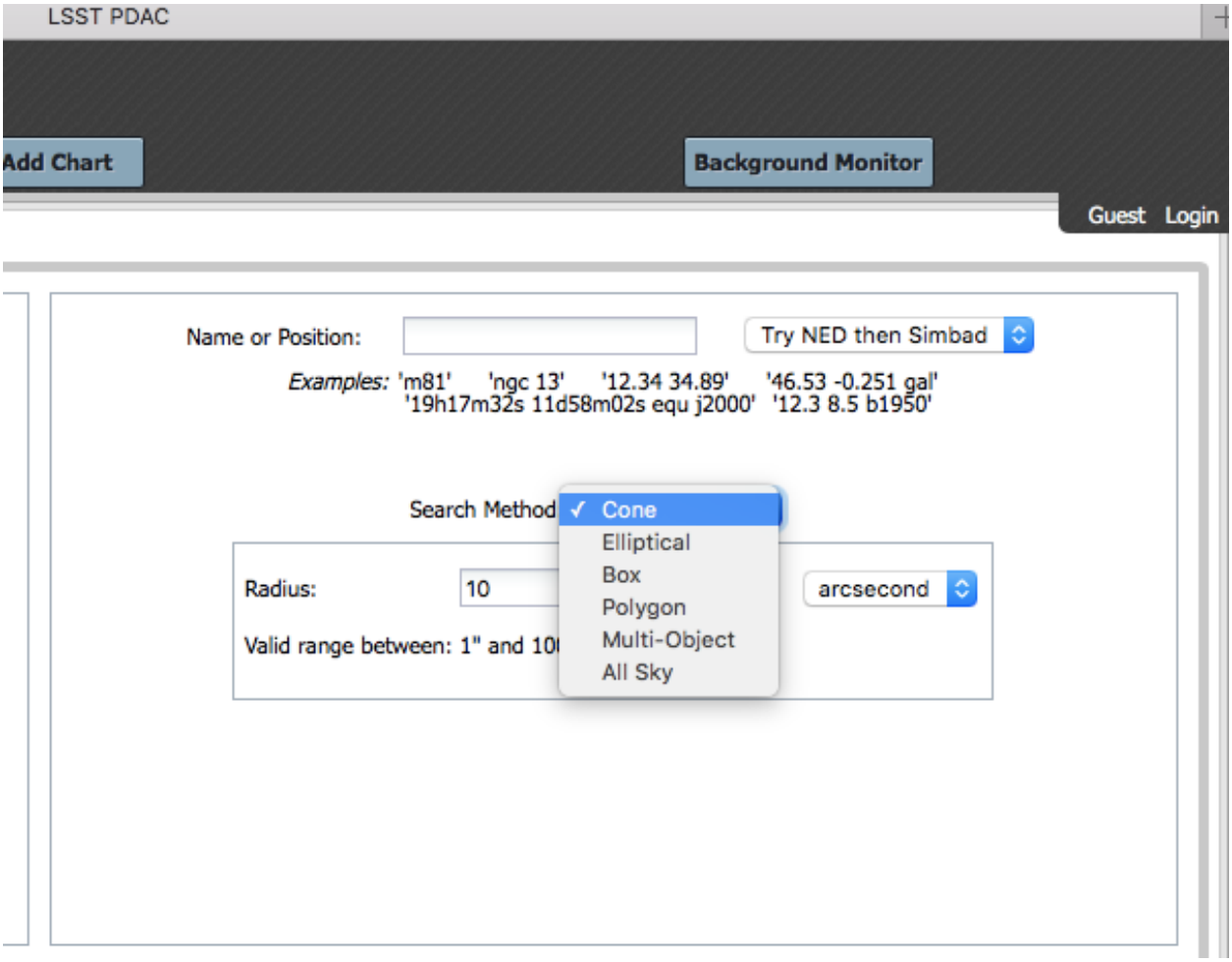

FIGURE 3: Search methods available for Portal queries on the WISE tables

Note that units and descriptions are not provided in the UI, though the layout anticipates them; this is a consequence of the absence of a full `metaserv` implementation at the time of the tests.

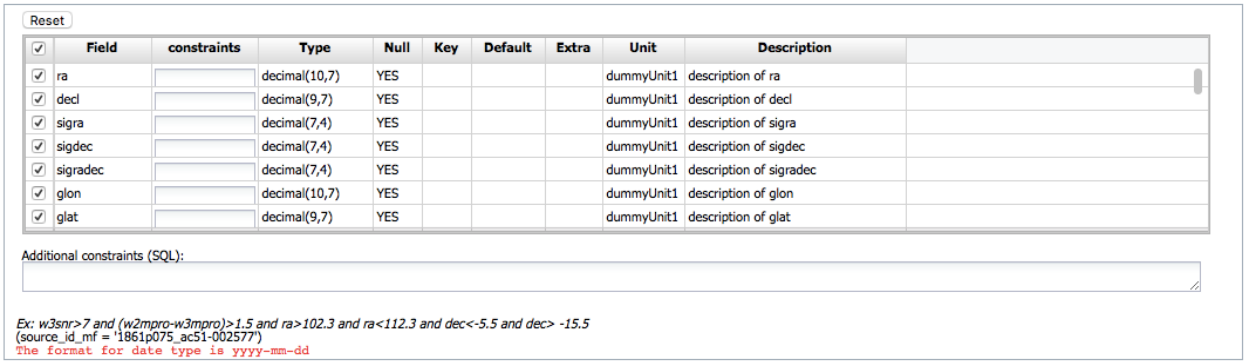

FIGURE 4: Schema browser and column-restriction API in the Portal, for WISE 3-band table

The test specification calls for doing spot queries around (ra=0, dec=0); it turns out to be more useful to do them away from ra=0 so that in the default plots in the Portal plotting tool the points are not split across values just below 360 and just above 0. (The values can be rescaled to (-180,180), of course, but this requires an extra step.)

100-arcsecond cone searches were successfully performed at (ra=1, dec=0) for all six tables. The `allsky_3band_p1bs_psd` search returned no rows, corresponding to this sky location not being included in the brief period of 3-band observation. Screen shots are provided for `allwise_p3as_psd` and `neowiser_yr1_p1bs_psd`, and tabular results for all five non-empty queries are stored in the DMTR-52 repository. Note the much larger number of sources for the NEOWISE-R single-epoch table query.

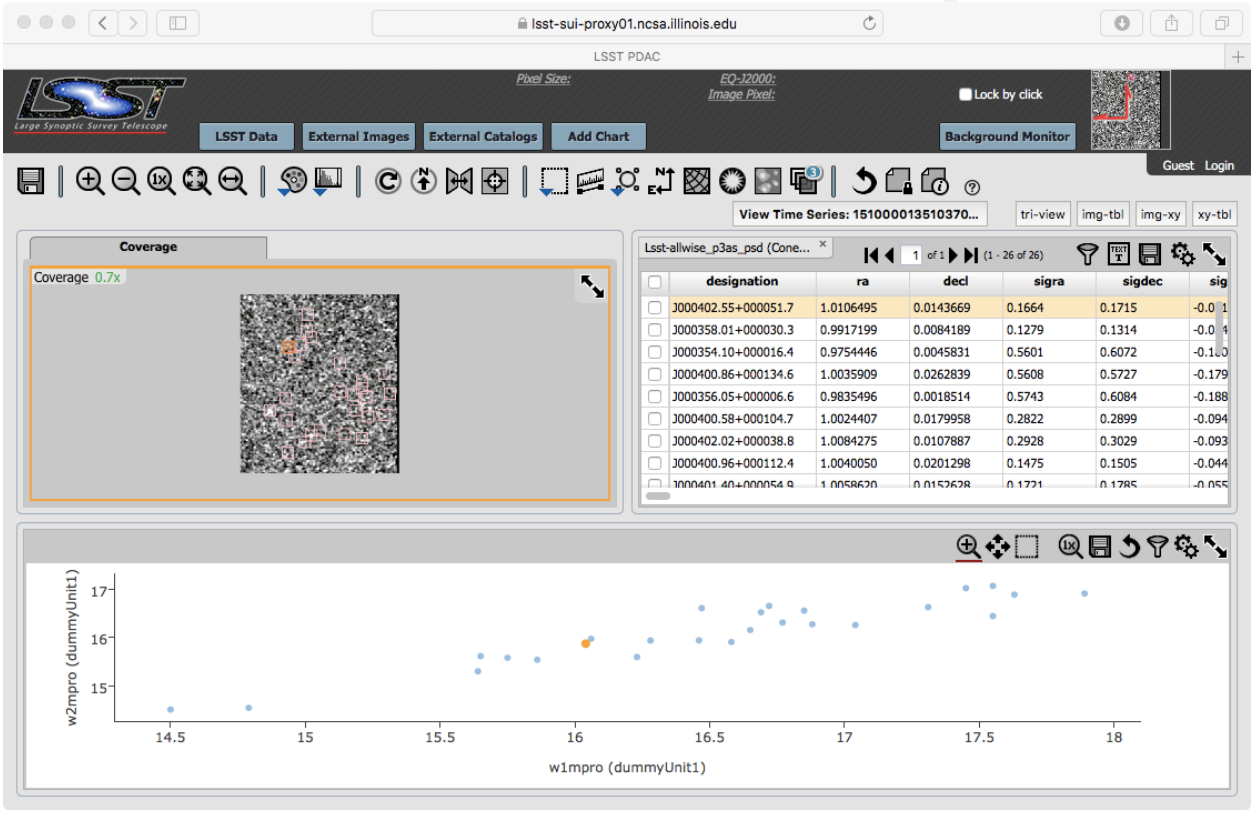

FIGURE 5: Query at ra=1, dec=0 on the AllWISE Source Catalog

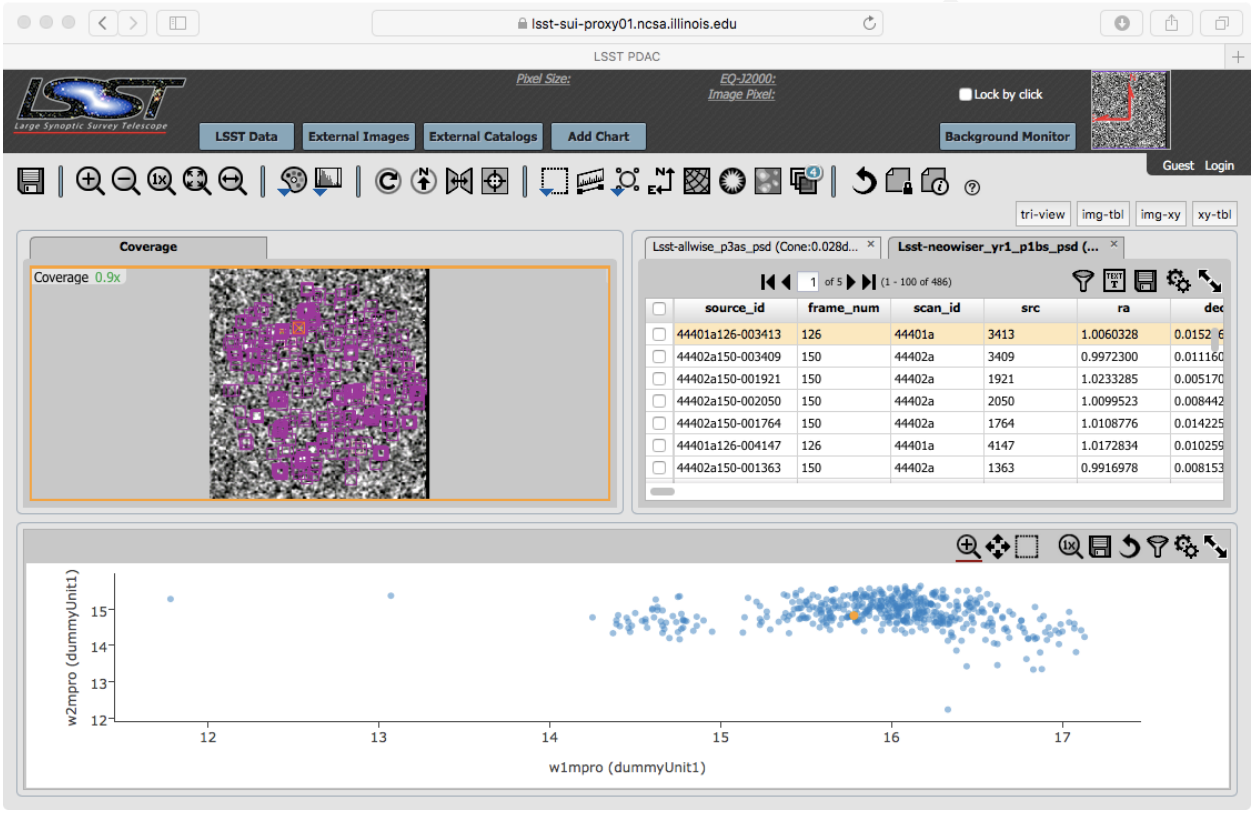

FIGURE 6: Query at ra=1, dec=0 on the NEOWISE-R Year 1 Single-Epoch Source Catalog

5 LSP-00-05: Demonstration of low-volume and/or indexed queries against the WISE data via API

The tests were performed primarily from an Apple laptop computer running OS X 10.11.6, wirelessly connected to the Internet and using the NCSA VPN at `vpn.ncsa.illinois.edu`. API Aspect queries were performed using a combination of the Python `requests` package and the `curl` command. Notebooks were executed using Jupyter versions up through 5.4.1 and Python version 3.6.1.

5.1 API Aspect tests

This test case consisted only of API Aspect tests.

The API Aspect tests made use of the "v0" `dbserv` service at

`http://lsst-qserv-dax01.ncsa.illinois.edu:5000/db/v0/tap/sync`,

as for LSP-00-00 above.

Comparison data was obtained from the IRSA TAP service at

`https://irsa.ipac.caltech.edu/TAP/sync`.

This service accepts SQL queries, forwards them to Qserv for execution, and returns the results as JSON structures. The open-source Python *Requests* package[3] was used to perform the HTTP GET and POST operations required, and to parse the returned JSON structures into Python dictionaries. The builtin Python `xml.etree.ElementTree` class was used to parse the VOTable data returned from the IRSA TAP service.

[3] `http://github.com/requests/requests`

5.1.1 Summary

The tests successfully verified that the PDAC and IRSA services return the same data for small-diameter cone searches, down to floating point roundoff issues associated with the return of data as ASCII strings.

The cone search queries were performed in the Galactic plane near Galactic coordinates (30, 0), expressed as (*ra* = 281.5, *dec* = -2.6), with a radius of 100 arcseconds. The radius was chosen to match the largest currently available radius for cone searches in the Portal UI (a relatively arbitrary number originally chosen based on IRSA experience with querying the MultiEpoch Photometry table).

In general the PDAC queries executed substantially more quickly than the IRSA ones. IRSA queries on the tables tested took several times longer. The IRSA times fluctuated more on repeated access, presumably because IRSA is an active community archive and has a fluctuating load of other queries. Some dependence on time of day was observed in the IRSA query performance, which would be consistent with performance depending on community load. Occasional large outliers were observed.

The PDAC DAX `dbserv` cannot yet be described as a true TAP service, though it is planned to develop into one. By way of comparison:

- The two services do not accept the same language at this time. DAX `dbserv` "v0" does not implement ADQL spatial query constructs, but does require the use of special Qserv UDFs to enable optimized spatial queries. As a result, the spatial `WHERE` clauses look quite different.
- The `dbserv` service does not yet support returning data in VOTable format. Currently JSON is the primary format for internal use in the PDAC and between SUIT and `dbserv`. It is still undecided how this will develop: e.g., whether we will develop a variant of VOTable with a more optimal bulk data representation but still use the VOTable XML "header" for table metadata.

A small fraction of the time (very roughly 2-3%) queries against the PDAC `dbserv` were observed to fail, apparently randomly, and always succeeded upon retry. The SUIT developers have also observed this during the implementation of the Portal interfaces to `dbserv`.

TAP Service	WHERE clause
PDAC dbserv	`qserv_areaspec_circle( 281.5, -2.6, 0.027777777777777776 )`
IRSA (ADQL)	`CONTAINS(POINT('J2000',ra,dec),CIRCLE('J2000',281.5,-2.6,0.027777777777777776))=1`

TABLE 5: Comparison of PDAC and IRSA cone search query syntax

5.1.2 Cone Search Queries

The cone-search queries returned the following results, with a range of elapsed times noted for repeating the query several times.

Table	Unique Object Keys	Rows	PDAC elapsed time	IRSA elapsed time
"Object" `wise_00.allwise_p3as_psd`	48	48	0.2 – 0.6s	1.4 – 3.2s
"ForcedSource" `wise_00.allwise_p3as_mep`	56	1377	0.8 – 1.8s	6.1 – 6.6s
"Source" `wise_4band_00.allsky_4band_p1bs_psd`	506	506	0.5 – 0.6s	1.6 – 3.3s

TABLE 6: Search results from small cone searches

All the sources found in the Object-like table were found in the ForcedSource-like table. Note that the WISE data analysis includes forced photometry on lower-quality objects, not just those in the main Object-like "AllWISE Source Catalog"; this is why there are more unique object keys in the ForcedSource-like table search result than in the Object-like search result.

All the IDs were tested for exact equality, with no failures. All queries retrieved `ra`, `decl`, and the WISE W1 and W1 magnitudes `w1mpro` and `w2mpro`. The time-dependent queries also retrieved the MJD mid-point of the corresponding observations. All these floating-point values were found to be equal to the limits of the round trip through the slightly different ASCII formats produced by the two services; in general discrepancies occurred when the IRSA TAP service returned additional non-significant digits, producing values like 4.3340000000000005.

5.1.3 Identifier Search Queries

One of the results from the cone search test query, WISE `source_id` "2813m031_ac51-041856", numeric ID (`cntr`) 2813003101351041856, was chosen as the target for tests of querying by a source identifier. This identifier was then used for queries against the Object-like and ForcedSource-like tables; in the ForcedSource-like table the identifier was used to query against `source_id_mf`, the ID of the parent object rather than of the individual forced photometry measurement, or against `cntr_mf`, the corresponding numeric ID.

Because no source association was done on the WISE single-epoch sources, the same identifier could not be used to test against the single-epoch Source-like table. A single identifier, "03245b121-000010", chosen from the cone search test against the Source-like table, was used to perform a comparable, though not particularly interesting, test on the Source-like table.

Pure identifier searches on the numeric ids, `cntr` and `cntr_mf`, were impractical on the PDAC tables because these triggered table scans (see §refsect:detail-lsp-00-10 below). This is because, currently:

1. No Qserv "secondary index" for this identifier was created, so Qserv must attempt to search every shard.
2. The PDAC tables were not indexed on this identifier in the chunks, so every shard search is itself necessarily a table scan.

This issue arose during PDAC development, and a workaround was used in the implementation of the initial Portal Aspect prototype for light curve retrieval. Given a target already located in a search on the Object-like table, in order to look up the light curve in the ForcedSource-like table, a search was performed on the combination of the numeric ID and a small spatial cone around the known location of the target. Because of a current Qserv implementation restriction, the spatial cone must appear first in the WHERE clause, i.e.: "WHERE qserv_areaspec_circle(281.4933836, -2.6186303, 0.005) AND cntr_mf=2813003101351041856".

This workaround is not a satisfactory solution for general ID lookup, of course. However, as a matter of interest, times for these spatially-restricted numeric identifier searches on PDAC are reported here and may be compared to the equivalent pure numeric-ID searches at IRSA, and to their table-scan equivalents, omitting the auxiliary spatial cone restriction, reported in the following section.

The ForcedSource-like searches returned the same light curves that were contained in the cone search results above, confirming the key relationship and its indexing were set up correctly.

Private discussion with IRSA engineering staff suggests that the main difference in performance arises from overhead in the IRSA TAP service, not the underlying database. If this can

Table and key	ID type	Rows	PDAC elapsed time	IRSA elapsed time
"Object" `source_id`	string	1	0.13 – 0.20s	1.4 – 5.3s
"Object" `cntr`	bigint(20)	1	0.18 – 0.19s[4]	1.7 – 10.5s
"ForcedSource" `source_id_mf`	string	25	0.18 – 0.20s	0.7 – 4.5s
"ForcedSource" `cntr_mf`	bigint(20)	25	0.11 – 0.41s[4]	0.7 – 30.0s[5]
"Source"	string	1	0.18 – 0.21s	1.5 – 5.2s

TABLE 7: Search results from queries by identifier

be remedied, this discussions suggest that the performance difference vs. Qserv for small indexed queries might go away.

The performance of Qserv at the scale deployed in PDAC, for small queries, appears quite suitable for responsive interactive work as long as the queries can be optimized by Qserv to select a subset of shards. What is not clear from the tests available in the current PDAC deployment is what the performance of Qserv will be when applied to indexed attributes that are not spatially correlated — i.e., where the indexing can only accelerate the per-shard queries but cannot yield any reduction in the number of shards on which the query must be executed.

6 LSP-00-10: Demonstration of table-scan queries against the WISE data via API

6.1 API Aspect tests

This test case exercises the LSP via the API Aspect only.

It verifies the ability to perform a variety of basic table scan queries.

Data for the test are primarily taken from the Object-like AllWISE (coadded) Source Catalog, and the ForcedSource-like Multi-Epoch Photometry table, except as noted below.

6.1.1 Step 2

The tests were performed primarily from an Apple Macbook Air computer running OS X 10.11.6, connected to the Internet wirelessly at IPAC and other locations. Queries requiring larger downloads were performed on an Apple iMac computing running OS X 10.11.6, connected to the IPAC wired institutional network.

The tests were performed using Jupyter 5.4.1 on Python 3.6.1, and the notebook was accessed using version 11.1 of the Safari browser.

6.1.2 Step 3

A connection was established to the PDAC network environment using the NCSA VPN at `vpn.ncsa.illinois.edu`.

The `dbserv` version 0 pseudo-TAP service was accessed at:

`http://lsst-qserv-dax01.ncsa.illinois.edu:5000/db/v0/tap/sync`

6.1.3 Step 4 — Very small queries

A target object was chosen near (ra=4, dec=0): `source_id` "`0045p000_ac51-034420`", `cntr` 45100001351034420, `ra` 3.9967364, `decl` 0.002037.

Since the `cntr` attribute is not indexed, a search for a row matching this unique ID in the Object-like database, or a search for rows matching this as the parent-object ID in the ForcedSource-like database, will result in a table scan. If combined with a spatial restriction, this table scan can be limited to a subset of the database shards maintained by Qserv.

Following the guidance in LDM-540, although these queries scan up to the full number of rows in these two tables (758 million and 42 billion, respectively), they return only a small amount of data in the end.

A series of increasing-diameter cone searches were done, using the special Qserv spatial-restriction function `qserv_areaspec_circle`, to trace out the behavior of Qserv as the fraction of shards scanned increased.

The query texts were: `SELECT cntr, ra, decl, w1mpro, w2mpro FROM wise_00.allwise_p3as_psd WHERE qserv_areaspec_circle(3.9967364,0.002037, r ) AND cntr=45100001351034420` for the Object-like table, and `SELECT cntr, cntr_mf, ra, decl, mjd, w1mpro_ep, w2mpro_ep FROM wise_00.allwise_p3as_mep WHERE qserv_areaspec_circle(3.9967364,0.002037, r ) AND cntr_mf=451000013510` for the ForcedSource-like table, where the radius r is in degrees.

The queries were carried out for radii in powers of two from $1/1024$ through 64, and for the half-sky radius 89.9. Finally, an all-sky version was carried out by omitting the `qserv_areaspec_circle` function.

All the queries returned the same results: a single row in the case of the Object-like queries, and 24 rows of forced photometry associated with that object for the ForcedSource-like queries. Successfully completing the half-sky and all-sky ForcedSource queries via `dbserv` required raising the `webserv` timeout from 30 minutes to two hours.

Radius (deg.)	Sky Area (sq. deg.)	Object elapsed time (s)	ForcedSource elapsed time (s)
0.0010	3.00×10^{-6}	0.54	0.83
0.0020	1.20×10^{-5}	0.22	0.33
0.0039	4.79×10^{-5}	0.22	0.35
0.0078	1.92×10^{-4}	0.22	0.37
0.0156	7.67×10^{-4}	0.21	0.40
0.0313	3.07×10^{-3}	0.22	0.41
0.0625	1.23×10^{-2}	0.21	1.03
0.125	4.91×10^{-2}	0.21	0.67
0.25	1.96×10^{-1}	0.22	0.46
0.5	7.85×10^{-1}	0.23	0.40
1.0	3.14×10^{0}	0.33	0.45
2.0	1.26×10^{1}	0.92	0.62
4.0	5.02×10^{1}	1.84	3.55
8.0	2.01×10^{2}	5.22	5.79
16.0	7.99×10^{2}	16.59	17.27
32.0	3.13×10^{3}	55.67	59.13
64.0	1.16×10^{4}	207.62	942.73
89.9	2.06×10^{4}	380.07	2365.89
all-sky	4.13×10^{4}	929.33	4506.66

TABLE 8: Table-scan results for small queries on Object- and ForcedSource-like tables

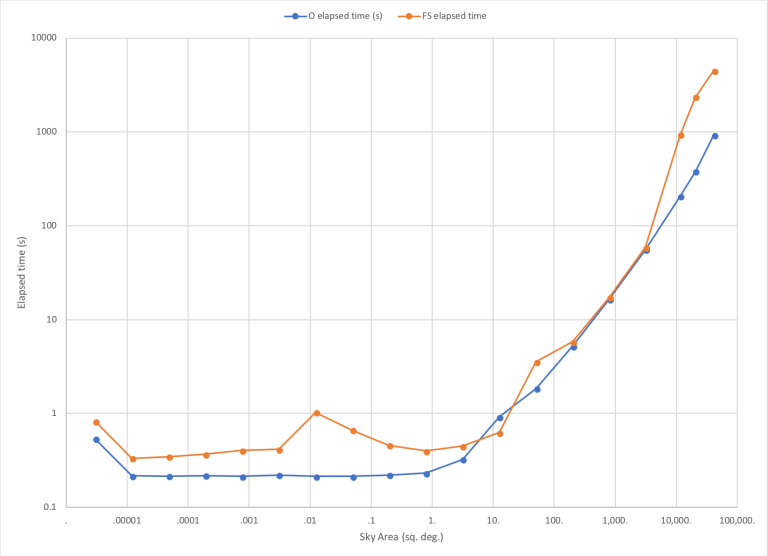

FIGURE 7: Query results for the Object-like AllWISE Source Catalog

6.1.4 Step 4 — Low volume queries

The "low volume" query specification is for queries which are intended to complete within an interactive-work-friendly time of 10 seconds. These were intended to be limited in scope

to a fraction of the Object database limited to no more than 10 million rows. In the logic of Qserv this has to be a spatial restriction, in order to permit the number of shards searched to be reduced. As a fraction of the eventually expected Object database, this is approximately 1/2,000. If this is expressed spatially, for the LSST survey it would be approximately 10 square degrees, or the size of a single focal plane field. This is also approximately the size expected for coadd "patches". Accordingly, in these tests we explored query sizes of approximately this spatial area, as well as of approximately the same number of rows:

- a 2-degree-radius cone search around the reference Object above, with area 12.6 square degrees, and including 170,503 rows; and
- a 16-degree-radius cone search around the reference Object above, with area 799 square degrees, and including 10,546,932 rows.

Note that for the latter query spatial-restriction "denominator" even the one-row identifier query above does currently take more than the specified 10 seconds (16.6s — see above); however, there was no expectation that the final Qserv performance would have been met by the current deployment.

For reference, as a fraction of the AllWISE Object-like database, the 1/2,000 ratio above corresponds to 379,000 rows, in between the two spatial-restriction choices explored.

The expected size of the result set supported for such a query is specified as 0.1 GB. Scaled by the ratio of the number of rows in the WISE dataset (758M) to the expected Object database size of approximately 20G rows, this would be about 4 MB. Nevertheless, queries approaching 100MB in result set size were performed.

To exercise the space, two types of queries were run, one returning a minimal set of columns (`cntr, ra, decl, w1mpro, w2mpro, w3mpro, w4mpro`) and the other returning all columns (`SELECT *`). For each spatial region and each query type, a series of queries was run with different restrictions on the value of `w1mpro` (which was not indexed) to scan the space of result set sizes.

The queries in this section were carried out via the `curl` command, executed from within a test notebook, because this permitted separately measuring the query execution time in the DAX service and the data transfer time from DAX to the client host. (The `curl` command has

an option for reporting the elapsed time until data begins to be returned from the request.) The notebook was running on the `lsst-lspdev.ncsa.illinois.edu/nb` JupyterLab service. The file transfer times recorded for the largest downloads (in the high 10s of MB) corresponded to intra-NCSA delivered bandwidths from the `dbserv` service to the Jupyter Notebook processes of 200-300 MB/s. It was not part of this test specification to perform tests of multiple simultaneous downloads.

Note that the performance of these queries was measured on an otherwise unloaded system. With the current Qserv configuration, all of these are believed to have been carried out as "shared scans", which could have been joined with any other pending shared scans and potentially taken much longer. The currently configured threshold for performing a shared scan, as opposed to immediate execution, is a query area larger than about 3 square degrees. On an unloaded system it is difficult to see any difference in performance, and there is no explicit feedback available from Qserv as to which mode of execution is being used for a particular query. The shared-scan-threshold configuration parameter can be revisited as operational experience is gained.

Approximately one in twenty of the queries performed failed randomly on the first attempt, but succeeded when immediately repeated. This is believed to be due to an issue in the communication protocol between `dbserv` and Qserv, rather than a problem in Qserv itself. There was no correlation with query size.

6.1.5 Step 4 — High volume queries

"High-volume" queries, by contrast, are expected to be ones that perform scans over all or almost all the shards in the Qserv system. The performance specification for these is that 20 simultaneous queries, with result sets no larger than 6GB, can be carried out simultaneously — clearly, as shared scans — while maintaining a one hour latency.

Once again, the purpose of the present test was not to verify the performance specification, but merely to exercise queries of this type.

All-sky queries were carried out for a selection of values of a cut on `w1mpro`, testing both the "wide" (`SELECT *`) and "narrow" query types above. Scaled by the number of Object-like rows in the AllWISE database compared to the expected LSST Object table, a result set target of 225MB was estimated. In practice result sets of up to 313 MB were exercised. All of these

	Narrow column selection				SELECT *		
w1mpro cut	rows	query time (s)	transfer time (s)	bytes	query time(s)	transfer time (s)	bytes
5	9	0.463	0.002	984	0.501	0.002	31,945
6	15	0.420	0.002	1,413	0.487	0.003	44,744
7	37	0.445	0.002	2,978	0.545	0.002	91,549
8	72	0.458	0.003	5,464	0.599	0.003	165,363
9	148	0.488	0.002	10,844	0.651	0.004	324,704
10	303	0.476	0.002	21,841	0.865	0.005	650,519
11	656	0.468	0.002	47,894	1.081	0.012	1,400,664
12	1,389	0.513	0.002	101,985	1.710	0.023	2,957,206
13	2,923	0.672	0.002	215,149	2.887	0.054	6,215,385
14	5,962	0.787	0.005	439,435	5.300	0.088	12,678,026
15	13,250	1.003	0.009	977,500	11.257	0.136	28,186,043
16	34,852	1.535	0.028	2,571,744	26.079	0.280	73,978,667
All	170,503	6.151	0.071	12,583,592			

TABLE 9: Results for "low-volume" table-scan queries on the AllWISE Object-like table; 2-degree cone

	Narrow column selection				SELECT *		
w1mpro cut	rows	query time (s)	transfer time (s)	bytes	query time(s)	transfer time (s)	bytes
5	424	20.074	0.002	31,242	17.469	0.005	922,819
6	907	19.682	0.002	66,353	18.786	0.014	1,953,476
7	2,157	16.992	0.003	157,386	18.756	0.030	4,612,767
8	4,224	19.767	0.003	307,831	22.555	0.052	8,975,541
9	8,634	19.411	0.006	629,012	29.801	0.097	18,229,195
10	18,786	19.488	0.012	1,369,350	34.722	0.161	39,591,673
11	41,462	17.496	0.021	3,086,630	51.364	0.302	87,834,386
12	90,133	18.112	0.040	6,775,570			
13	188,498	27.970	0.072	14,231,074			
14	382,298	35.801	0.138	28,917,025			
15	815,648	44.758	0.227	61,756,745			
All	10,546,932						

TABLE 10: Results for "low-volume" table-scan queries on the AllWISE Object-like table; 16-degree cone

tests completed in 15–20 minutes.

More complex WHERE clauses should be exercised in future tests, as well as the critical use case of spatial joins, which were not explored here.

		Narrow column selection			SELECT *		
w1mpro cut	rows	query time (s)	transfer time (s)	bytes	query time(s)	transfer time (s)	bytes
4.0	28,467	915.270	0.013	2,152,760	1030.500	0.247	61,605,670
4.5	50,097	918.895	0.024	3,784,044	1088.995	0.390	108,131,490
5.0	86,542	919.346	0.043	6,535,633	1129.759	0.579	186,377,582
5.5	145,792	939.140	0.061	11,012,287	1178.091	1.007	313,447,188
All	747,634,026						

TABLE 11: Results for "high-volume", all-sky table-scan queries on the AllWISE Object-like table

7 LSP-00-15: Execution of basic catalog queries in the Portal

7.1 Portal Aspect tests

This test case exercises the LSP via the Portal Aspect only, though it depends on the API Aspect for its operation.

It verifies the ability to perform a variety of basic catalog search types via the Portal.

Data for the test are primarily taken from the Object-like AllWISE (coadded) Source Catalog, except as noted below.

7.1.1 Step 1

The tests were performed primarily from an Apple iMac computer running OS X 10.11.6, connected to the Internet using a wired connection to the IPAC institutional network. The Portal was accessed using the Firefox browser, in various versions. Screenshots were taken from Firefox 59.

7.1.2 Step 2

A connection was established to the PDAC network environment using the NCSA VPN at `vpn.ncsa.illinois.edu`.

7.1.3 Step 3

Step 3a: The Portal was accessed at:

`https://lsst-sui-proxy01.ncsa.illinois.edu/suit/`

Step 3b: The test case instructs that cone searches of 300 arcsecond radius be performed. The Portal UI deployed on the PDAC (unnecessarily) limited cone searches to a 100 arcsecond radius, so that is what was used. See LSP-00-20 for additional discussion.

Cone searches of a 100 arcsecond radius around (*ra*, *dec*) = (0,0) were performed on the three tables mandated, with the NEOWISE Year 1 Single-Epoch Photometry table used as the Source-like table. All columns were requested (i.e., `SELECT *`). For the Object-like and Source-like tables the string `source_id` column was used for choosing the ID mandated in the test specification. For the Forced-Source-like table this column was not available and the numeric `cntr` column was used instead.

All searches took approximately 1-2 seconds to execute and display results.

PDAC WISE table	Rows	Selected ID	File
Object-like `allwise_p3as_psd`	37	`0000p000_ac51-032654`	`step3b-Object.tbl`
ForcedSource-like `allwise_p3as_mep`	1,313	`100010333930023`	`step3b-ForcedSource.tbl`
Source-like `neowiser_yr1_p1bs_psd`	394	`44373a126-005201`	`step3b-Source.tbl`

TABLE 12: Cone search results from selected PDAC WISE tables

The table-download feature of the UI was used to download the files noted in the table, preserved in the repository `DMTR-52/lsp-00-15`.

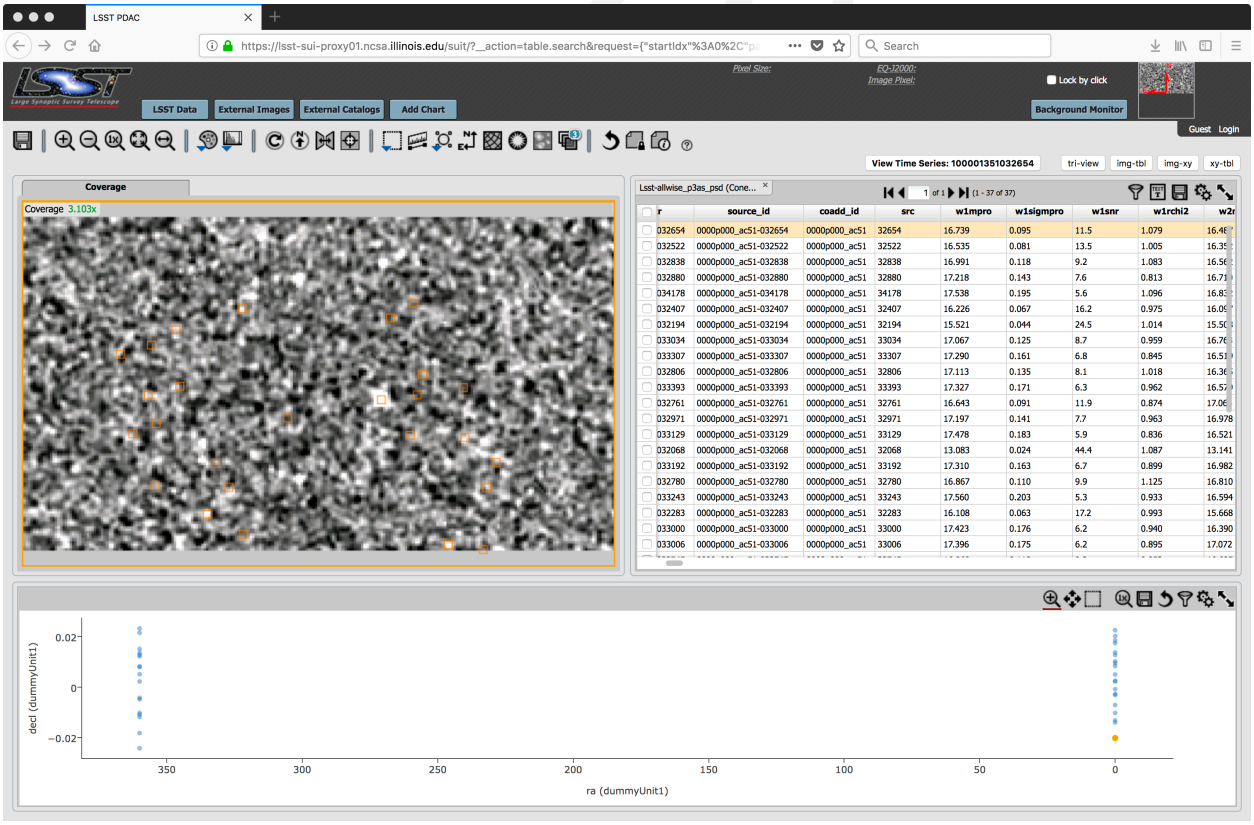

FIGURE 8: Query results for the Object-like AllWISE Source Catalog

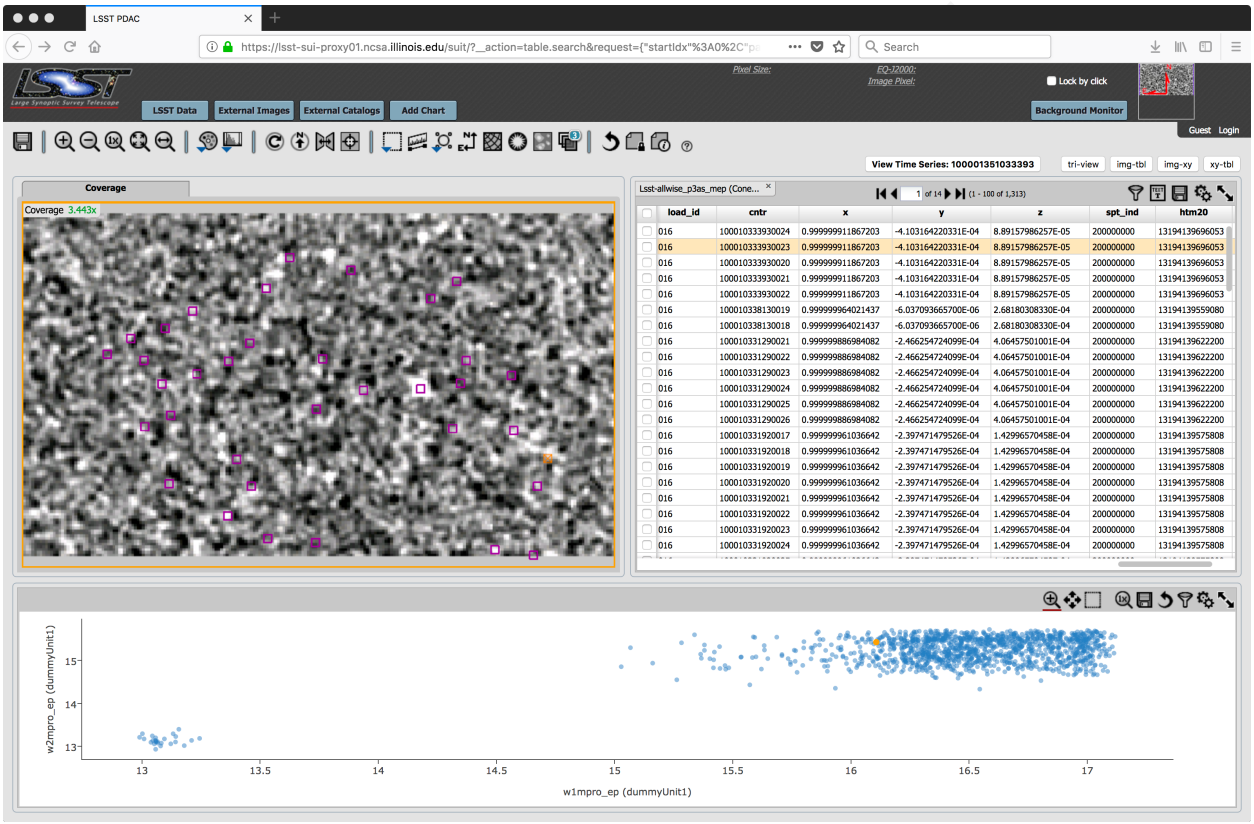

FIGURE 9: Query results for the ForcedSource-like AllWISE Multi-Epoch Photometry table

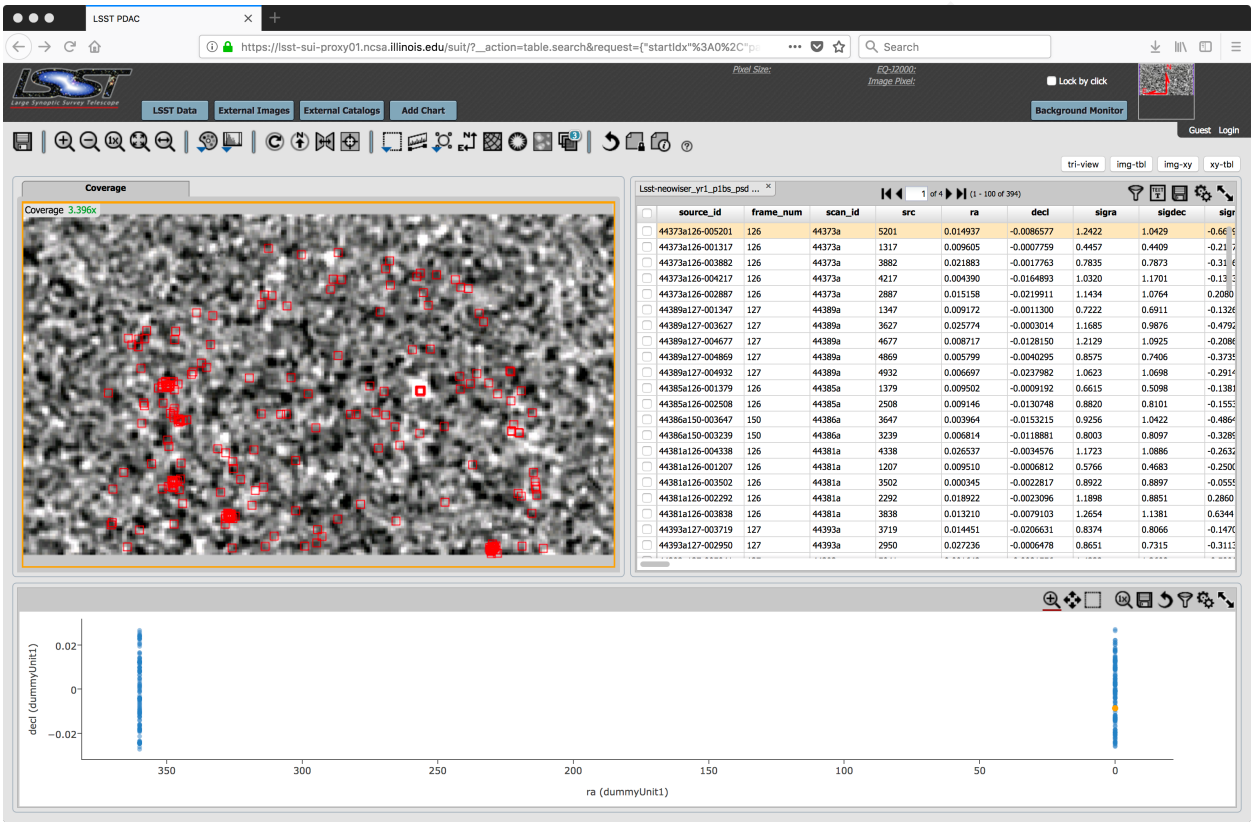

FIGURE 10: Query results for the Source-like NEOWISE Year 1 Single-Epoch Source Catalog

Step 3c: The multi-object search functionality of the UI was tested using a nominal 30 arcsecond search radius for each target in `LDM-540/test-scripts/lsp-00-15-coords.tbl`, which was uploaded to the UI. (The target list file had been generated using standard techniques for generating a uniform distribution of 100 random points on a sphere.) Note that the file name was changed from the originally suggested one in LDM-540 to help make clear that this file is in IPAC Table form, the only form currently accepted by the UI. The IPAC Table restriction should be reconsidered as part of the further development of the system. VOTable, at least, should be supported, and ideally also a more free-form input.

The test succeeded, finding multiple source for most targets, but requiring 14 seconds to perform a 100-object search. The test confirmed the ability to carry forward additional columns from the target table, beyond the `ra` and `dec`, to the result table, and associate them correctly with multiple search hits. A feature present in the equivalent IRSA capability was missing, however: the preservation of a row index from the original target table. This is a nice feature of the IRSA Viewer application's implementation. The result table was preserved as `DMTR-52/lsp-00-15/step3c-Objects.tbl`.

The normal implementation of multi-object searches is as a user table upload followed by a spatial join in a database. The current implementation of this feature in the Portal Aspect, however, required a workaround for the inability to upload user tables to the DAX system at this time. It therefore operates by iterating over searches one target at a time, inside the Portal Aspect server, and therefore performing many separate calls to the API Aspect. This feature will get reimplemented once the database systems provide the requisite underlying support. With the parallelism of Qserv, it can be expected to execute much more efficiently for substantially larger numbers of targets.

Step 3d: The "All Sky" search form was used to perform the identifier-based searches required. For the Object-Like and Source-like searches the search was performed on the `source_id` field, was successful and completed essentially instantaneously, consistent with the timings seen in LSP-00-05.

This did require typing the "constraint" in the search form in this format: "`='0000p000_ac51-032654'`", which is not natural. Typing just the string itself was not successful, even though this is an intuitive thing to attempt, and the resulting error message requires SQL knowledge to interpret. This should be improved.

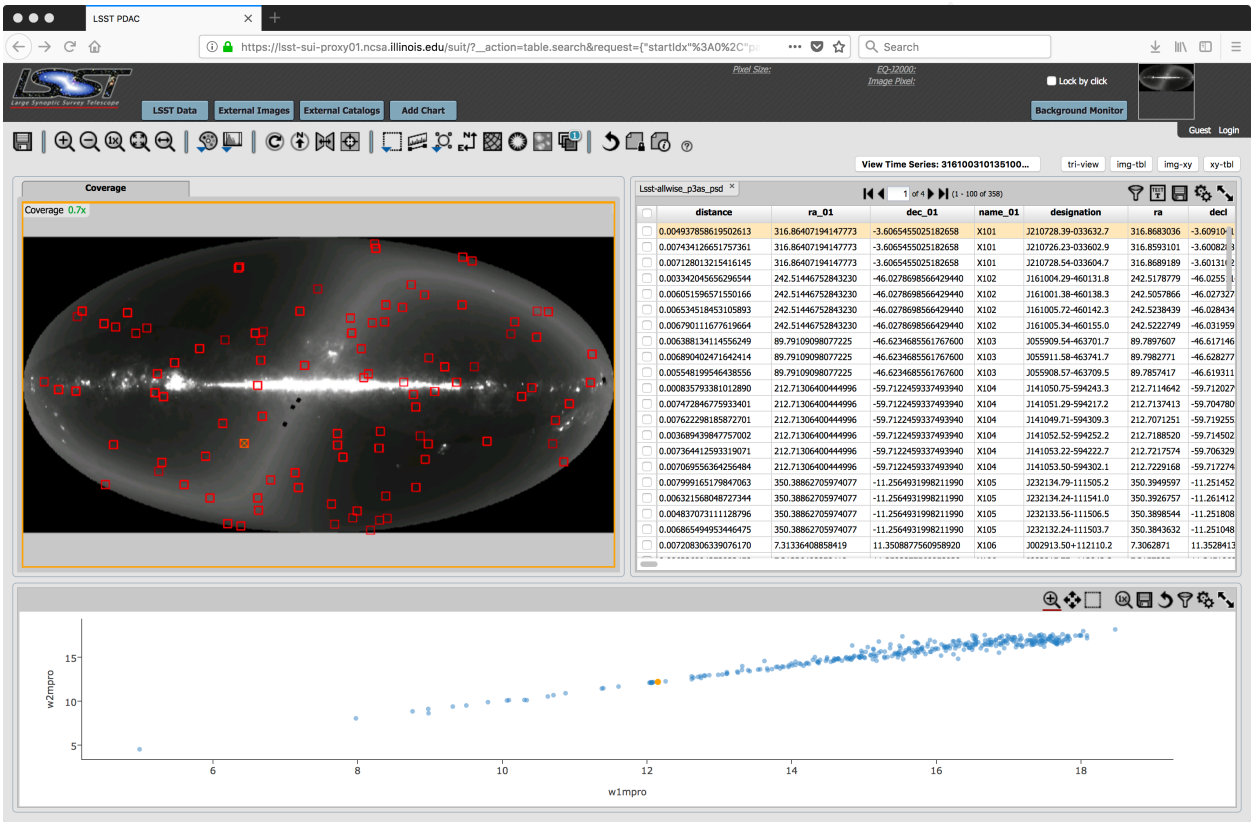

FIGURE 11: Query results for the 100-target search on the Object-like AllWISE Source Catalog

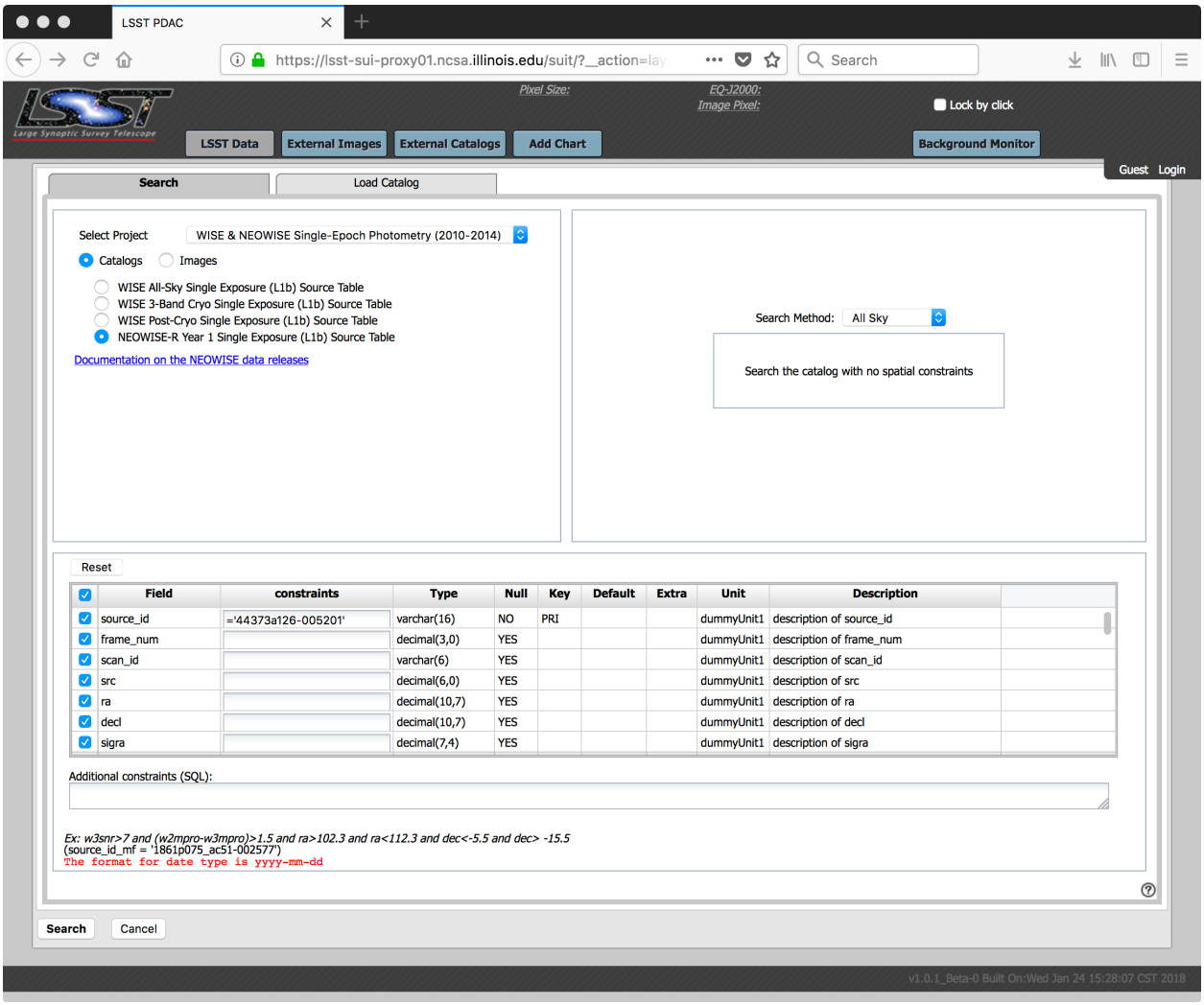

FIGURE 12: Search screen for a search-by-ID on the Source-like catalog

Search results were preserved as files `DMTR-52/lsp-00-15/step3d-Object.tbl` and `DMTR-52/lsp-00-15/step3d-`

For the ForcedSource-like table an attempt to search on the value of `cntr` appeared to trigger a table scan and was not feasible as an interactive query. Further investigation revealed that this table is not indexed on `cntr` or any other per-row unique key at this time.

It was, however, possible to manually search on the ForcedSource-like table with the ID from the Object table, `0000p000_ac51-032654`, and obtain a 27-row light curve table. (Note that a predefined workflow for this is also available in the UI and is exercised in LSP-00-35 below.)

Search results were preserved as files `DMTR-52/lsp-00-15/step3d-lightCurve.tbl`.

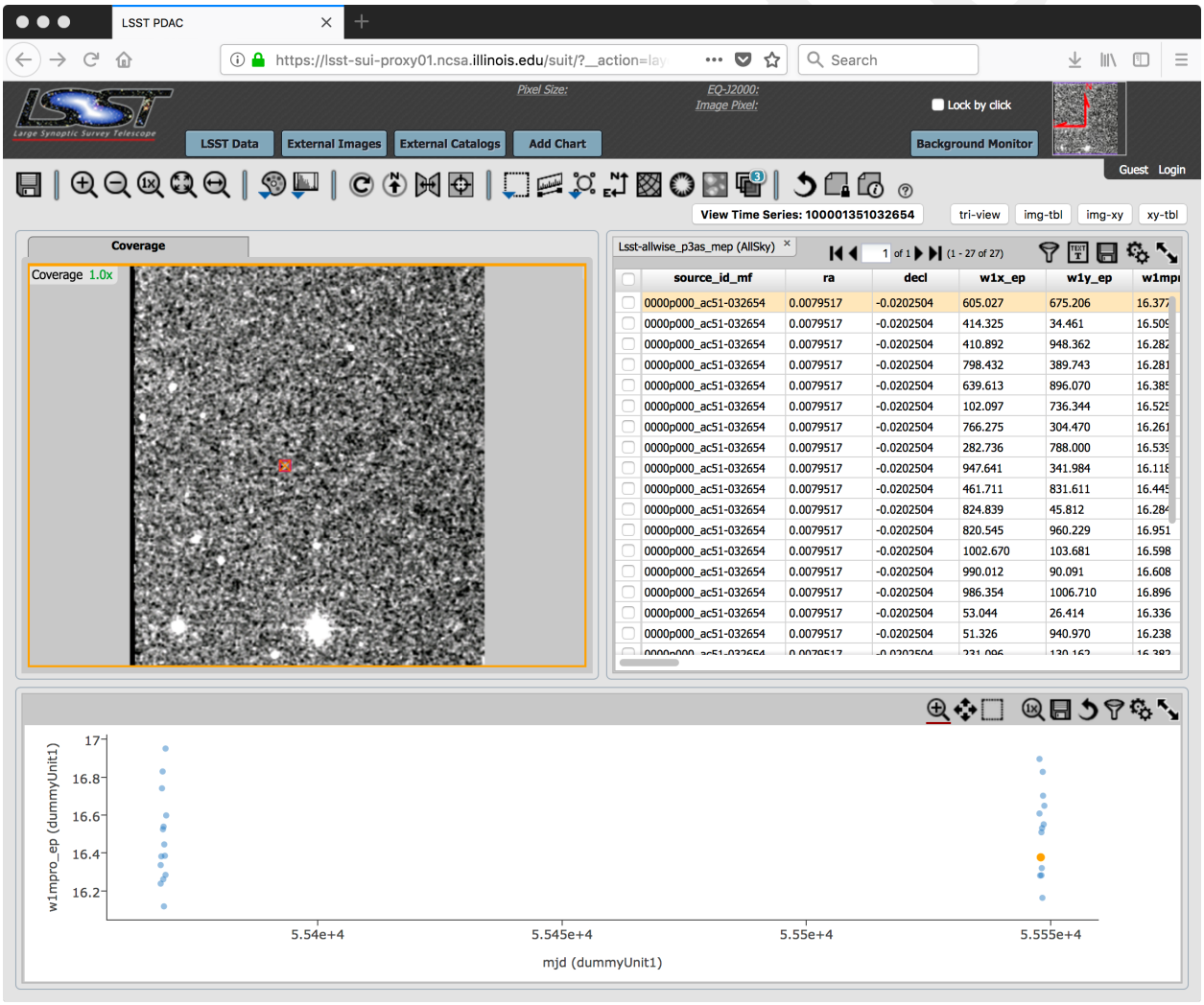

FIGURE 13: Results of a search-by-Object-ID on the ForcedSource-like catalog

8 LSP-00-20: Operation of the UI for interaction with tabular data results

8.1 Portal Aspect tests

This test case exercises the LSP via the Portal Aspect only, though it depends on the API Aspect for its operation.

It verifies the ability to perform a variety of display and exploratory data analysis operations in the Portal Aspect UI. Data for the test are taken from the Object-like AllWISE (coadded) Source Catalog.

8.1.1 Step 1

The tests were performed primarily from an Apple MacBook Pro laptop computer running OS X 10.12.6, connected to the Internet using a wired connection to the IPAC institutional network. The Portal was accessed using the Google Chrome browser, version 66.0.3359.139.

8.1.2 Step 2

A connection was established to the PDAC network environment using the NCSA VPN at `vpn.ncsa.illinois.edu`.

8.1.3 Step 3

Step 3a: The Portal was accessed at:

`https://lsst-sui-proxy01.ncsa.illinois.edu/suit/`

Step 3b: The test case instructs that a cone search returning at least 10,000 records be performed. In practice there are two problems with this:

1. The Portal interface limits cone searches to the rather tiny radius of 100 arcseconds, which in the AllWISE dataset cannot come close to yielding that many records.
2. The Portal limits scatterplots to a maximum of 5,000 points before falling back to representing the data as a 2D histogram or "heat map".

Neither of these limits is functionally essential. The Portal and the underlying DAX and Qserv interfaces can handle much larger search areas while preserving good performance. The polygon search interface exposed by the Portal can be used to query much larger areas while still yielding latencies compatible with interactive work, as noted in LSP-00-15 above, so there is no reason to limit the cone search so dramatically. For example, a 100 sq. deg. polygon search in the Object-like AllWISE Source Catalog executes in about two minutes, but then the resulting 1,318,805 row result table can be browsed and analyzed interactively with good performance, with the generation and display of 1D or 2D histograms, even on computed columns, taking only 2-4 seconds.

The limit on the scatterplot density is also overly conservative, though not by as large a factor. The current `Plotly` implementation only begins to slow down significantly when more than 20,000 points are displayed. Accordingly, the scatterplot limit has recently been raised in IRSA production, and there is no reason not to do this in LSST.

In addition, as described in the Summary section above, an alternate `Plotly` implementation, based on WebGL, is available which should scale to much larger plots.

Nevertheless, given the limits observed, the test case actually executed was modified as follows to permit all the remaining steps in the test case to function without further restrictions:

- a polygon search was used in place of a cone search, in order to permit searching a larger area; and
- the number of rows returned was adjusted to keep it below 5,000.

The resulting query, used as the input to the remaining steps in this section, was to select the polygonal region { (281.7, -2.4), (281.7, -2.8), (281.3, -2.8), (281.3, -2.4) } from the AllWISE Source Catalog, an area of approximately 0.16 square degrees. This search yielded 3,479 objects. The search was restricted via GUI to a subset of the available columns.

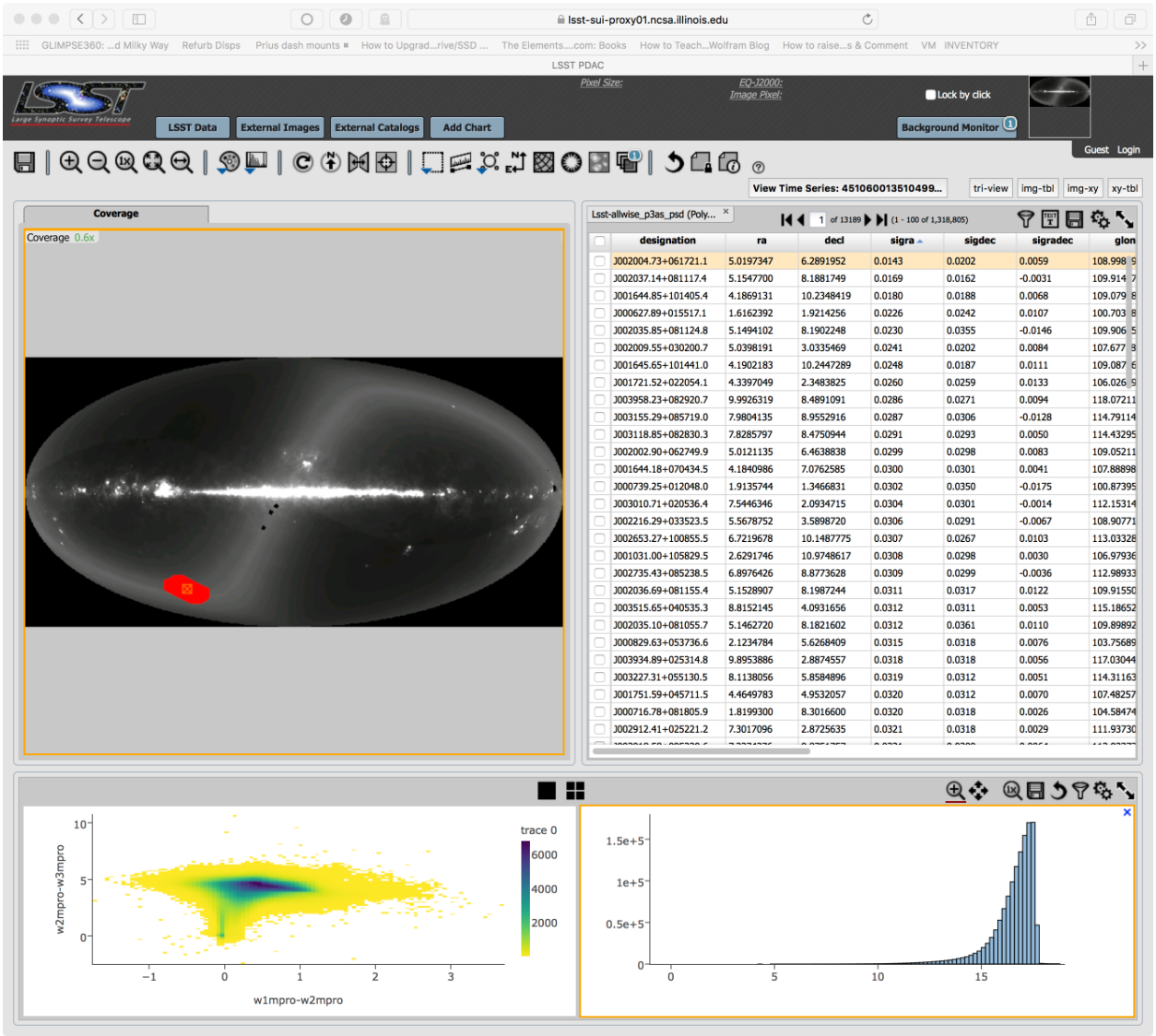

FIGURE 14: Result screen for a query with 1,318,805 result rows

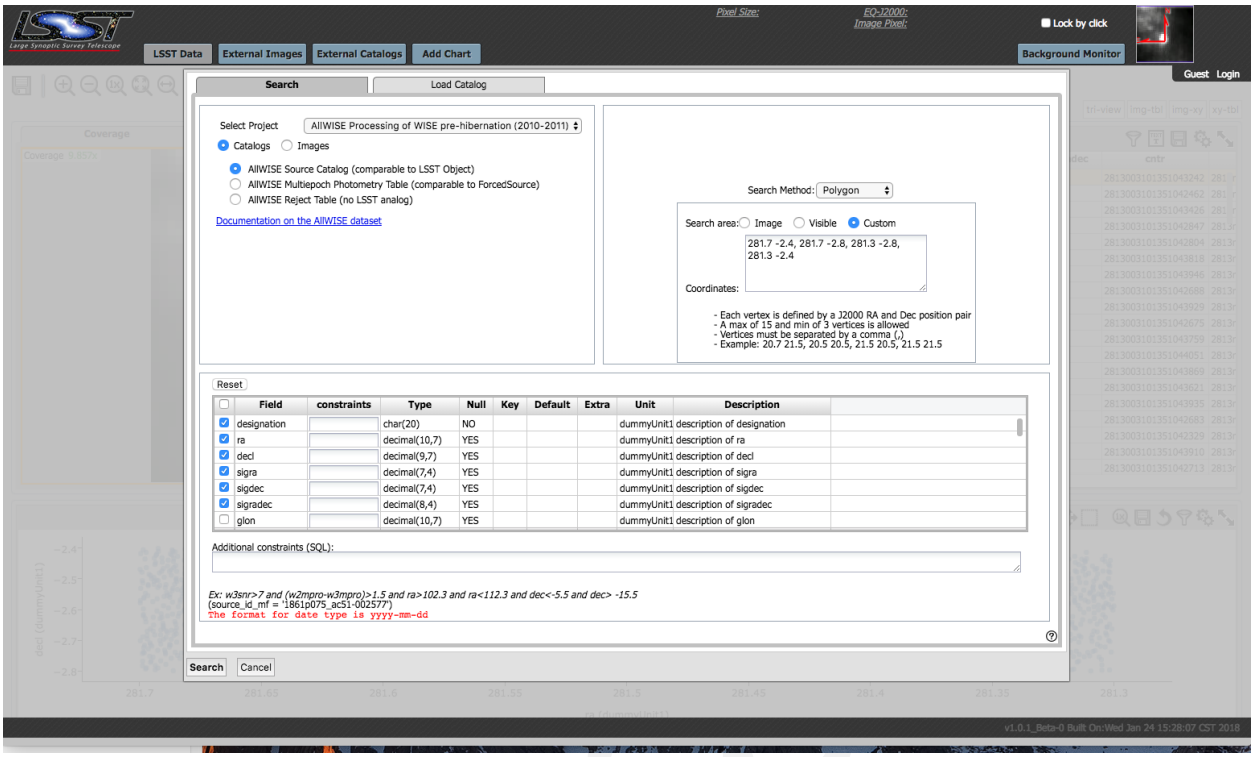

FIGURE 15: Query screen for the Object-like AllWISE Source Catalog

Step 3c: The table-download feature of the UI was used to produce the file preserved as `DMTR-52/lsp-00-20/step3-c.tbl`. No choice of file types was offered. The format is IPAC Table, with 17 rows of metadata and the expected 3,479 rows of data.

The table was confirmed by inspection of the first and last three rows in the GUI to be in the same basic order as displayed in the GUI.

The table was verified programmatically to contain `ra` and `decl` values well-distributed right up to the specified limits of the query, with the extrema being {281.3000661, 281.6998140} and {-2.7998067, -2.4000119}, respectively. No attempt was made to determine whether this was consistent with expected spherical-geometry deviations from the region boundaries, given the small size of the region.

Step 3d: The UI was used to sort the table based on the numeric column `ra`, and the result was visually inspected to be sorted as requested. Both sort directions were tested in the UI. The table, sorted in ascending `ra` order, was downloaded as `DMTR-52/lsp-00-20/step3-d.tbl`.

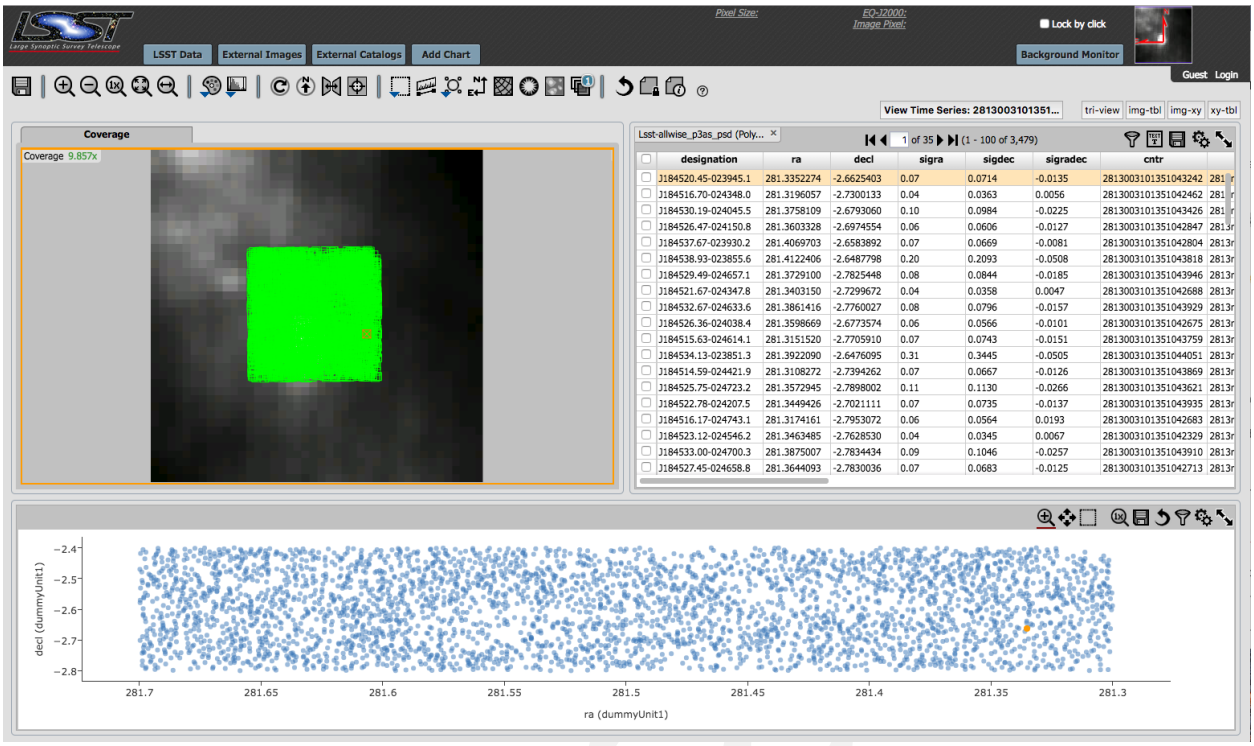

FIGURE 16: Result screen for the Object-like search near M81

The UI was then used to sort the table based on the string column `description`, with the table downloaded in ascending-`description` order as `DMTR-52/lsp-00-20/step3-d2.tbl`.

Step 3e: The sort order for the numeric `ra` column was confirmed by testing with the POSIX `sort -gs` utility on MacOS 10.11.6. The `-s` ("stable") option was required because of the presence of rows in which the `ra` values were identical to the 10 significant digits provided.

The sort order for the string column `description` was similarly verified with POSIX `sort` applied to the downloaded file.

It was noted that the downloaded table files contain an additional column `ROW_IDX` which contains a sequential index of the rows resulting from the initial query. These are preserved across sorting and filtering.

Step 3f: The UI was used to perform an interactive filtering operation on the resulting table, applying the filter `sigdec` > 0.4. The distribution of this variable is shown below; the filter selected only 28 of the 3,479 rows.

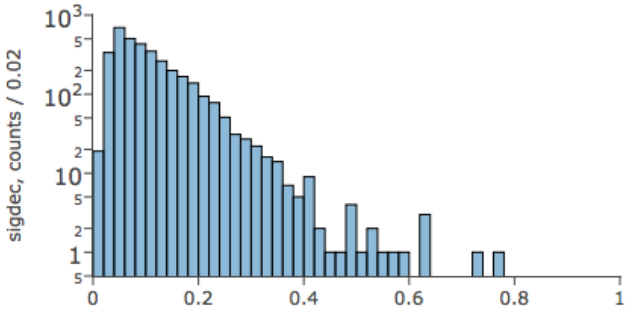

FIGURE 17: Histogram of `sigdec` before Step 3f filter

Visual inspection confirmed that the remaining rows all satisfied the filter. The filtered result was downloaded as `DMTR-52/lsp-00-20/step3-f.tbl`, and was confirmed to contain only the displayed rows. The POSIX `awk` utility was used to apply the same selection independently to the original download from Step 3c, and verify that the same rows were selected as by the filter in the UI.

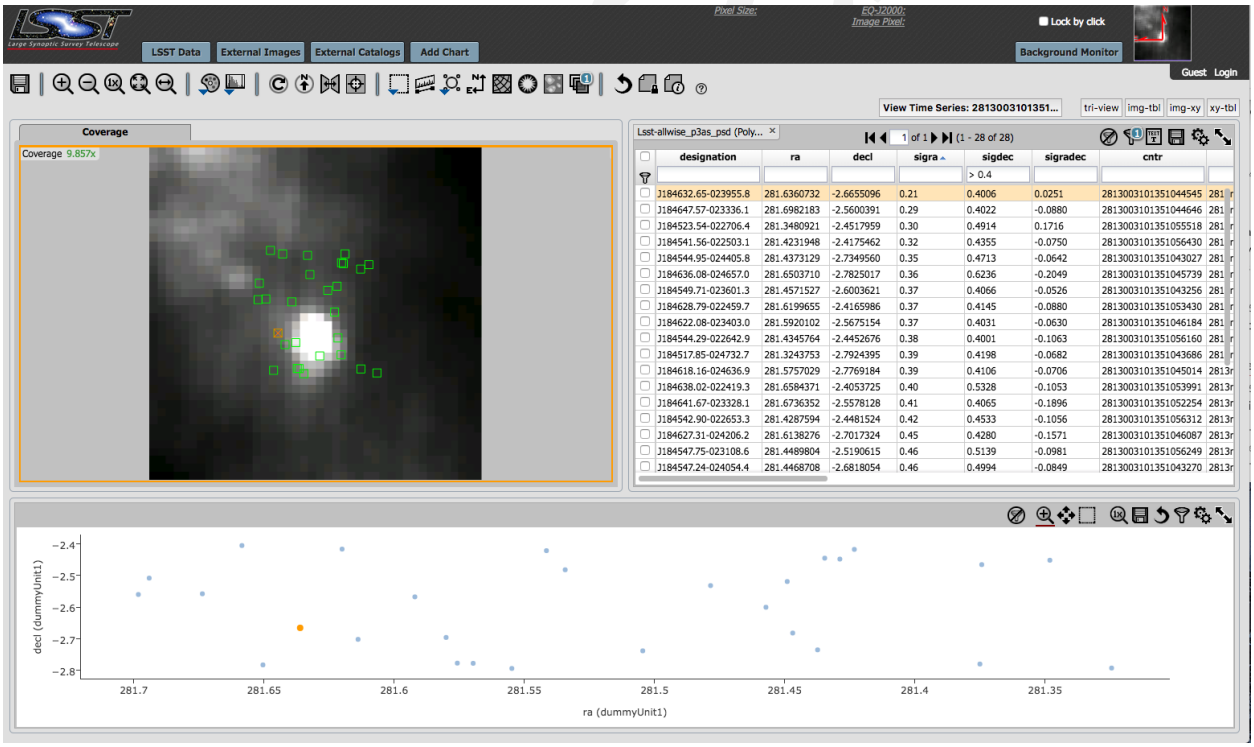

FIGURE 18: UI following application of filter on `sigdec`

Step 3g: Four rows were manually selected using the UI checkboxes, and then the table was filtered on these selections. The resulting four-row table was downloaded and visually confirmed to contain the same rows as displayed in the UI.

Step 3h: The UI was used to create a 1D histogram of `sigdec`, as displayed above in Figure 17. Hovering over selected bins, and noting the bin count reported, `awk` was used on the table from Step 3c above to verify the count. The limited number of significant figures reported for `sigdec` facilitated verifying that the bin definitions are inclusive on the low side and exclusive on the high side, as is conventional.

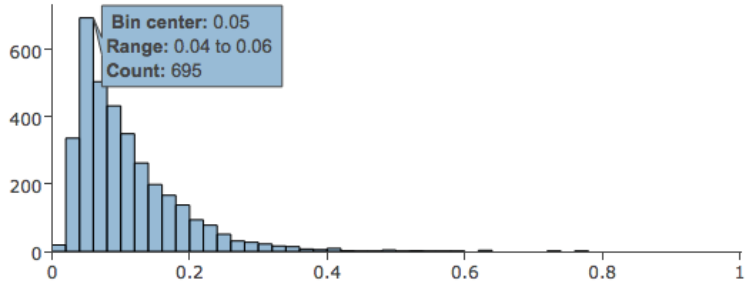

FIGURE 19: Histogram of `sigdec`, showing hover text for a bin

Step 3i: The UI was used to create a 2D histogram or "heat map" of `sigra` and `sigdec`. The spot checks in the test specification were performed to validate that outlying points in the distribution were displayed where expected. However, the combination of the hover text information displayed and the limited ability to control the bin boundaries meant that it was difficult to perform a quantitative verification of the binning itself. As noted in the summary above, while adequate to pass this early test, improvements are needed in order to support serious quantitative usage of this feature.

Step 3j: The UI was used to draw a rectangular selection on a scatterplot of `ra` versus `dec`, with the approximate limits, judged from the UI, of { {281.465, 281.58}, {-2.665, -2.595} }. The UI did not provide the expected means of reading off the exact boundaries once the region had been established.

The filtered result, containing 129 rows, was downloaded as `DMTR-52/lsp-00-20/step3-j.tbl`. POSIX utilities were used to determine the extrema of the values of `ra` and `dec` for the selected rows: {281.4646684, 281.5799433} and {-2.6611025, -2.5966385}, respectively. This appears consistent, within the ability to read off from the original selection, with the bounds chosen.

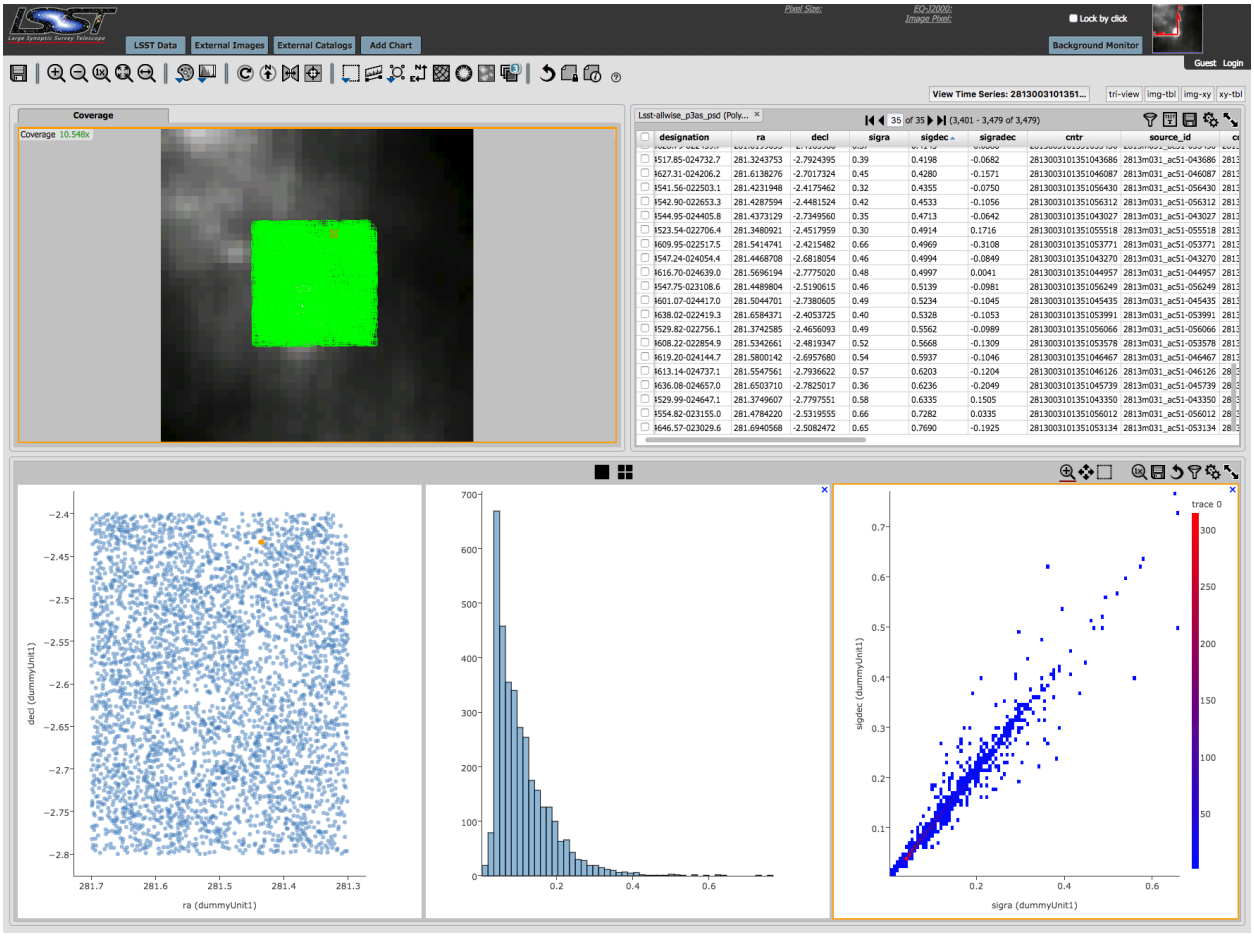

FIGURE 20: UI showing 1D and 2D histograms.

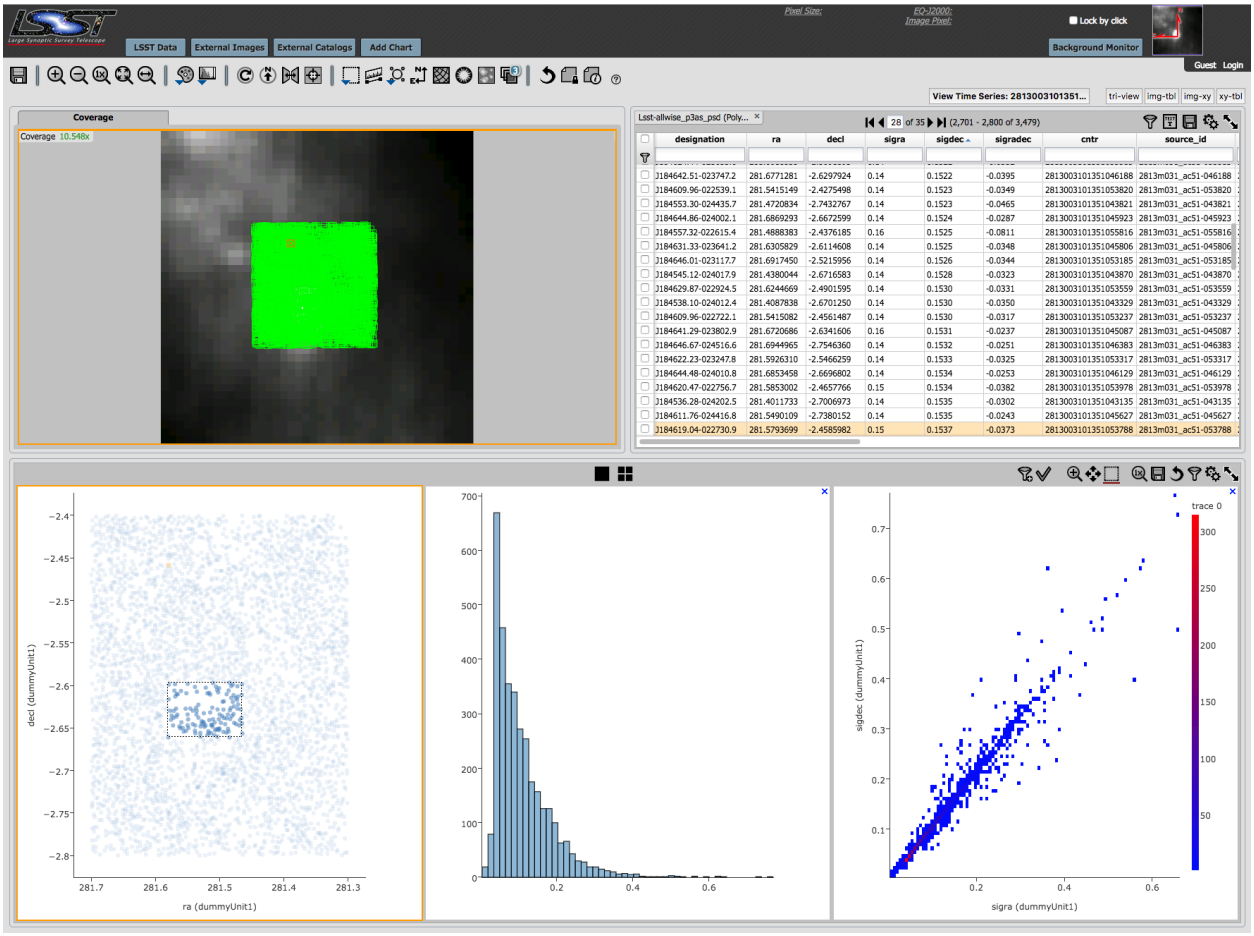

FIGURE 21: Step 3j selection region being drawn.

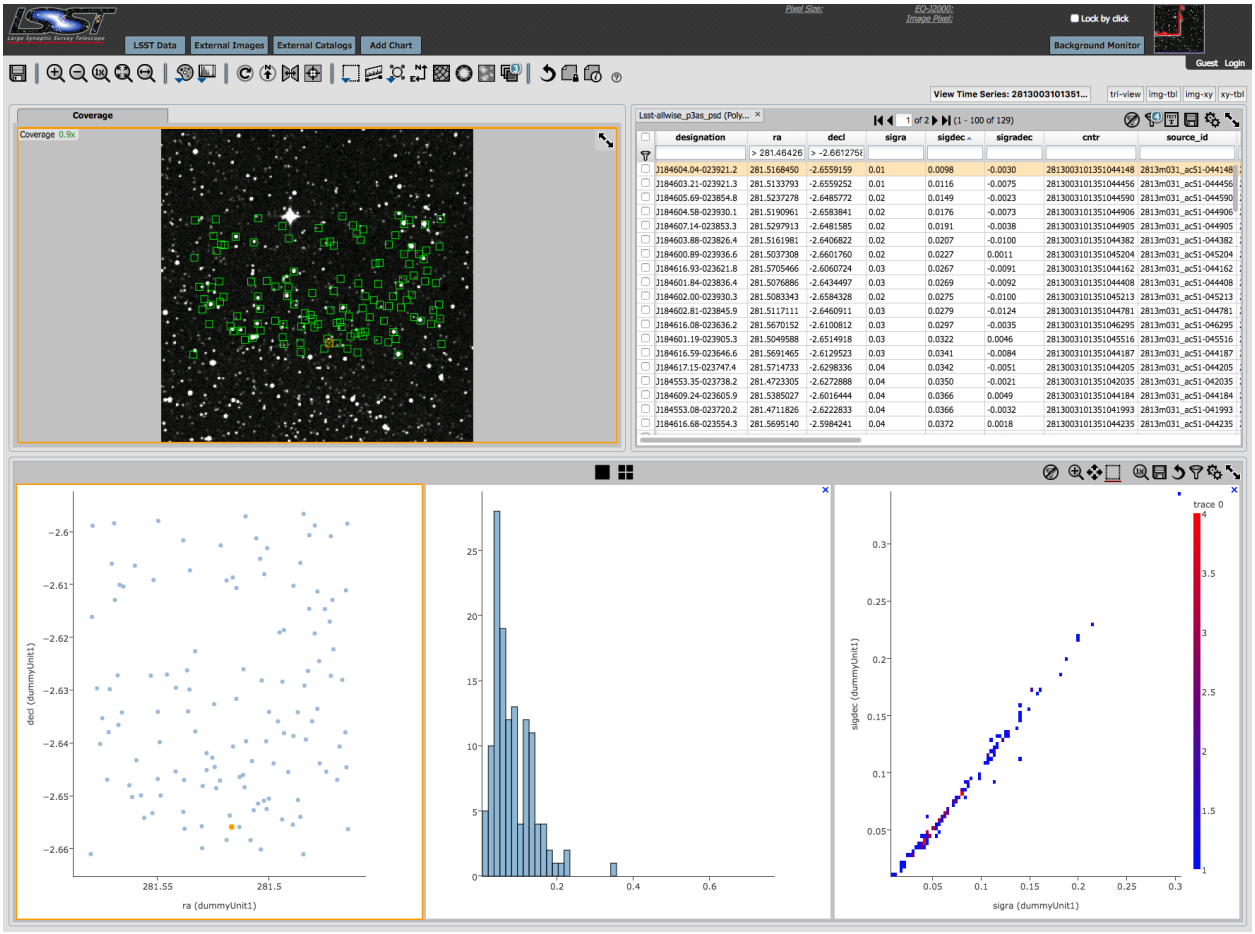

FIGURE 22: Display following application of Step3j 2D selection.

9 LSP-00-35: Linkage of catalog query results to related catalog data

9.1 Portal Aspect tests

This test case exercises only Portal Aspect functionality.

It verifies the ability to navigate from ann Object-like coadded source catalog entry to the associated ForcedSource-like forced photometry.

Note that, due to an editing error in LDM-540, at the time of the tests LDM-540's text called for this test to be performed against the SDSS data. The intent was that all these tests would be against the WISE data, since that was the point of the milestone associated with these tests, and the tests were, accordingly, carried out against the WISE tables.

9.1.1 Step 1

The tests were performed primarily from an Apple MacBook Pro laptop computer running OS X 10.12.6, connected to the Internet using a wired connection to the IPAC institutional network. The Portal was accessed using the Google Chrome browser, version 66.0.3359.139.

Additional tests, matching with LSP-00-05's cone searchs and Object-ID searches, were performed from an Apple MacBook Air laptop computing running OS X 10.11.6, wirelessly connected to the Internet. For these tests, the Portal was accessed, as an alternative to the browsers used in the other tests above, with Firefox 57.0.4.

9.1.2 Step 2

A connection was established to the PDAC network environment using the NCSA VPN at `vpn.ncsa.illinois.edu`.

9.1.3 Step 3

Step 3a: The Portal was accessed at:

`https://lsst-sui-proxy01.ncsa.illinois.edu/suit/`

Step 3b: A search was performed in the vicinity of M81 within the AllWISE Source Catalog (the coadded source table most comparable to LSST's Object). The search radius was set to the Portal's maximum cone search radius of 100 arcseconds.

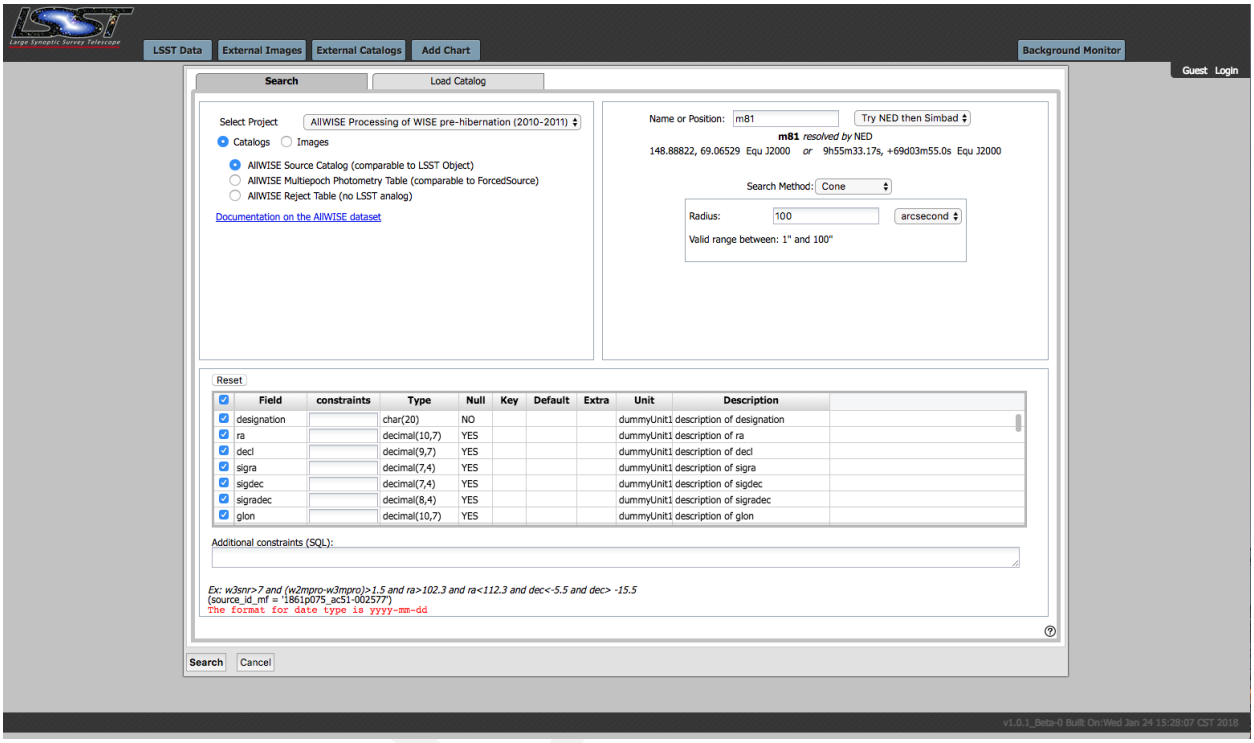

FIGURE 23: Query screen for the Object-like AllWISE Source Catalog near M81

Another search was performed on the reference area for cone searches from LSP-00-05 (§5 above), a 100 arcsecond radius region around (*ra, dec*) = (281.5, -2.6).

The search reproduced the set of 48 objects obtained via the API-based queries in LSP-00-05.

Step 3c: The "View Time Series" button in the Portal GUI was used to request the loading of the forced photometry time series data for a selected source. The current implementation of the button confirms the numeric ID of the object for which the time series is to be requested.

The figures below show the results of this action for an object contained in the cone search around M81, as well as for the reference object (`source_id` "2813m031_ac51-041856", `cntr`

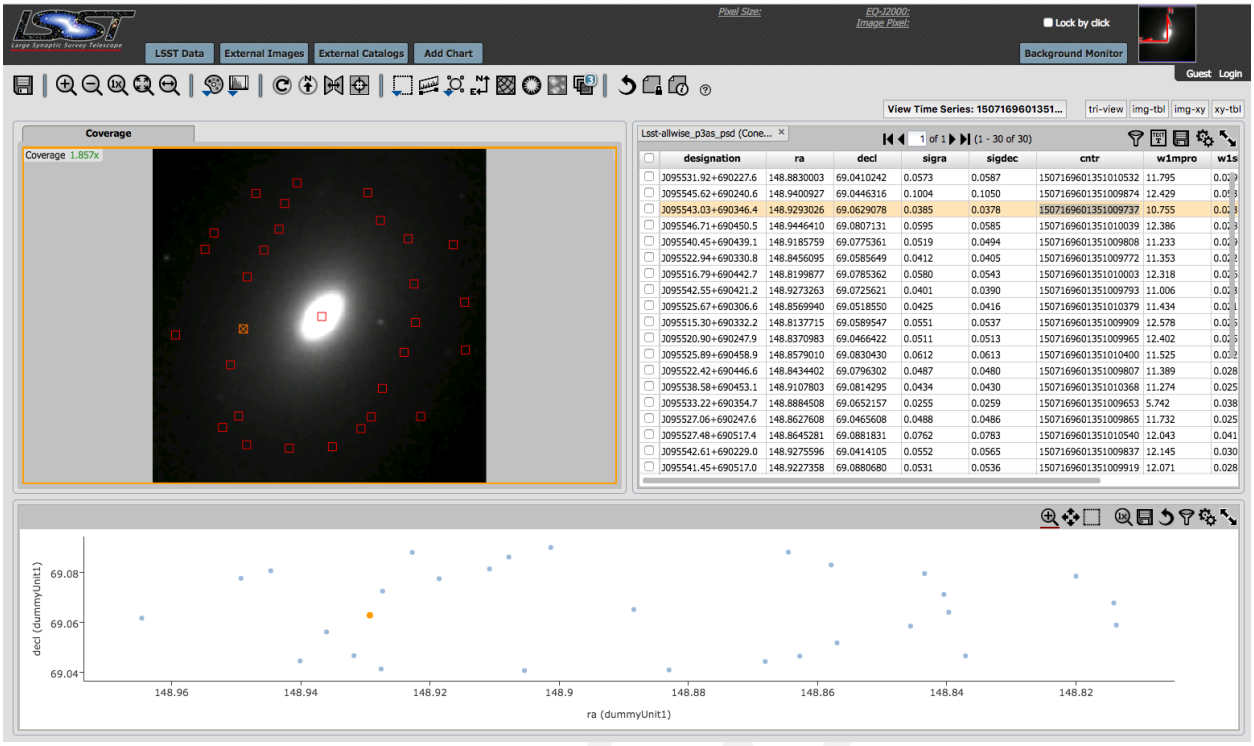

FIGURE 24: Result screen for the Object-like search near M81

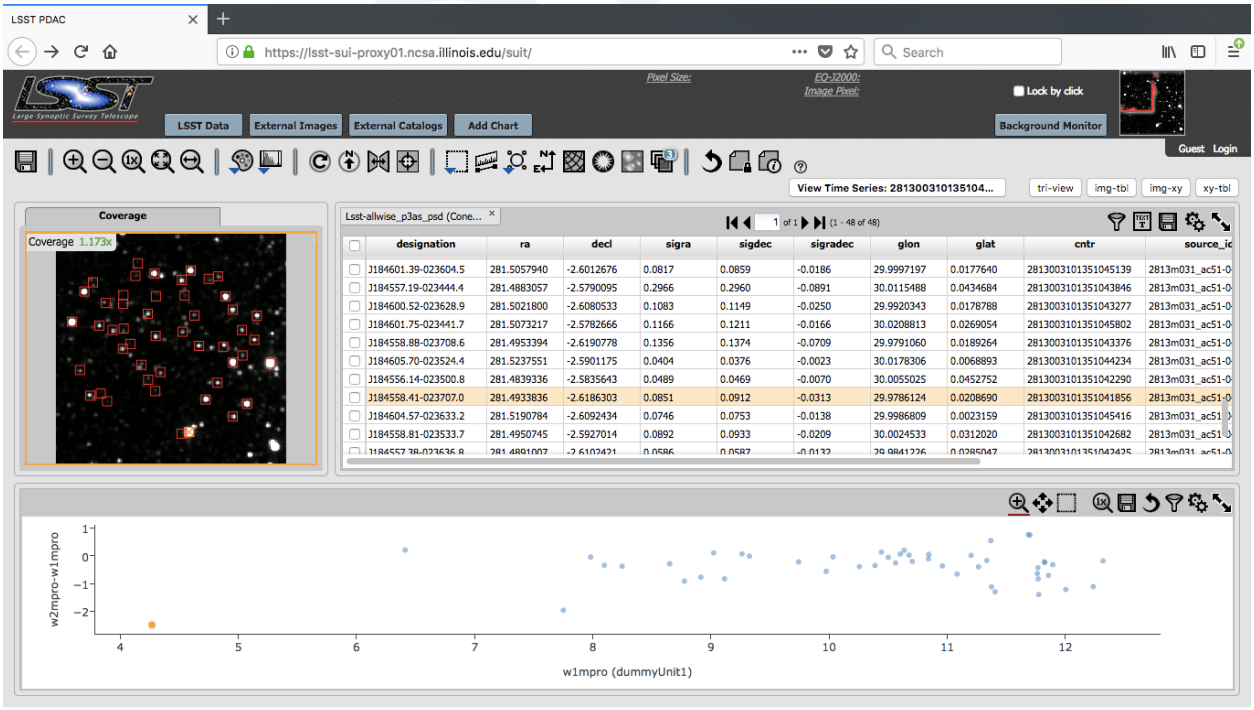

FIGURE 25: Result screen for the Object-like search near M81

2813003101351041856) selected for the comparable API Aspect query-by-ID tests in LSP-00-05.

Note that the time series displayed show the characteristic revisit time pattern arising from the WISE mission's orbit and observing strategy — an approximately 180 day recurrence, with a number of measurements made during the period during which the object is observable.

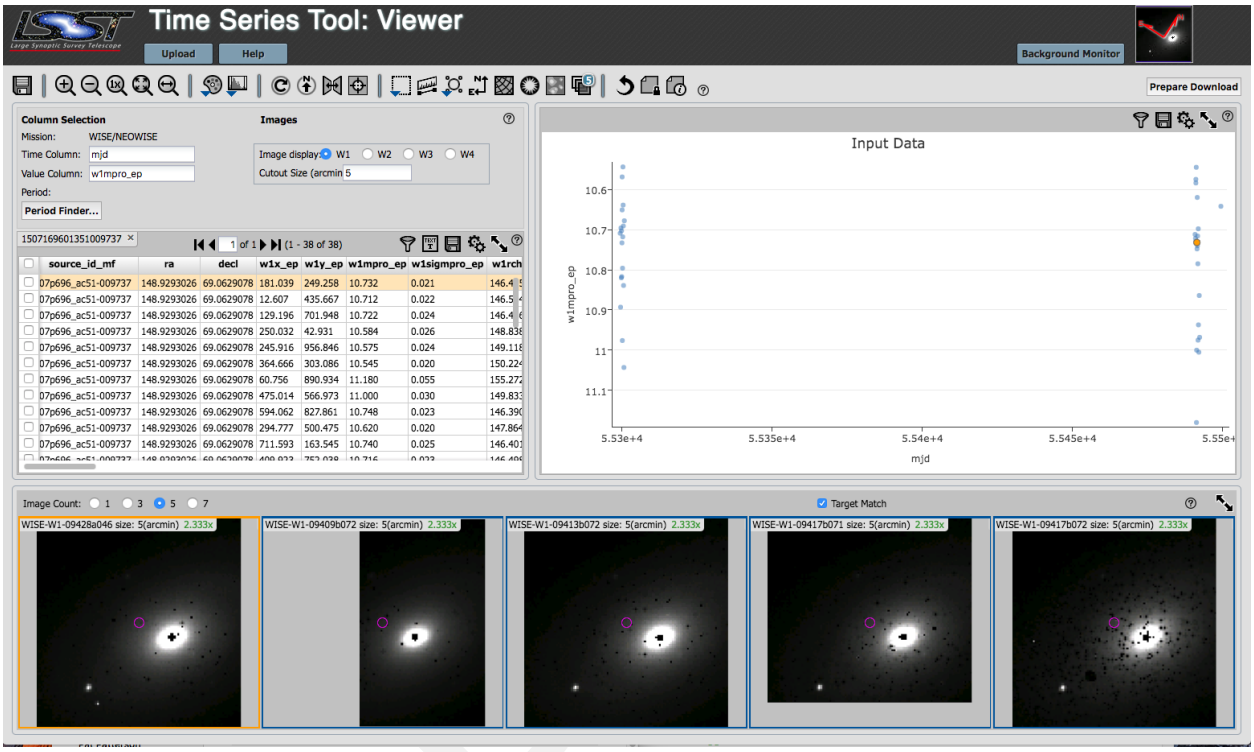

FIGURE 26: Result screen for a time series query near M81

Step 3d: Visual inspection of the `source_id` and `cntr_mf` values in the time series confirmed that the forced photometry data were in fact for the selected object. Visual inspection also confirmed that the time series returned for "2813m031_ac51-041856" was the same as the one retrieved in LSP-00-05. The time series for the latter object was downloaded from the Portal to the file `DMTR-52/lsp-00-35/2813003101351041856-lightcurve.tbl`, which is preserved in the repository for the present report.

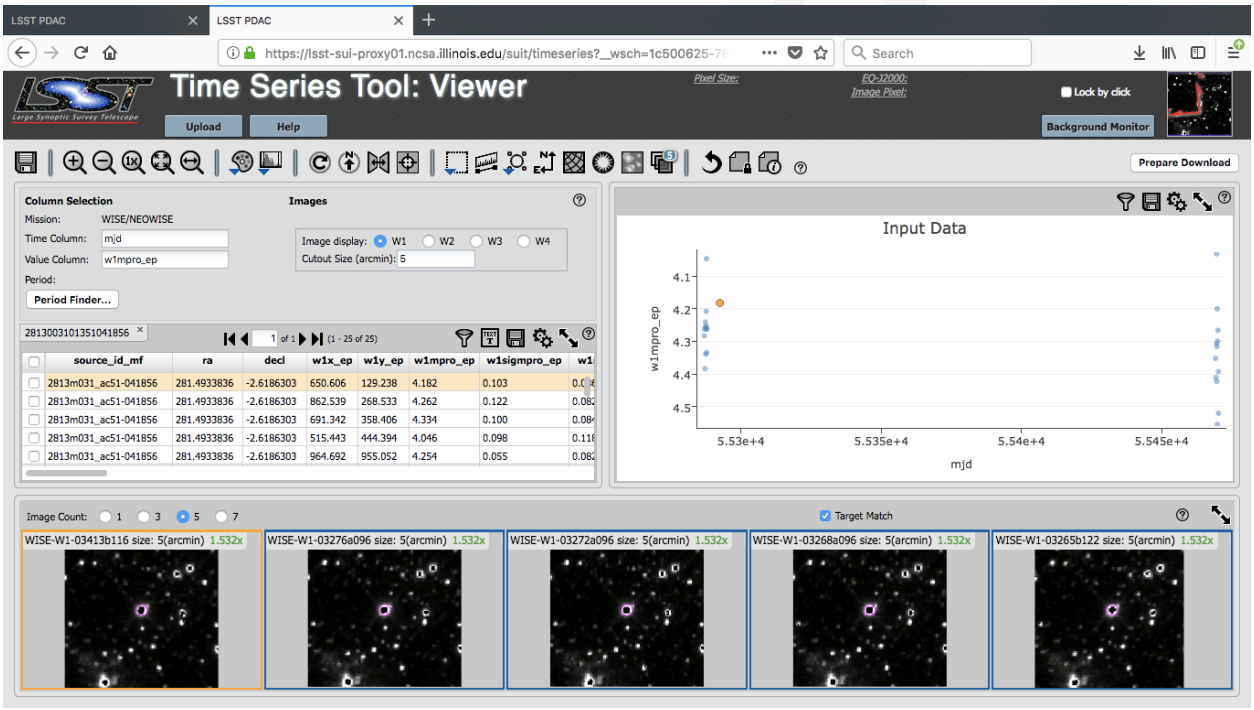

FIGURE 27: Result screen for a time series query for the reference object from LSP-00-05