D2.2.3: Ingestion software for ancillary surveys

WP2.2: Data Ingestion and Publication

Project Acronym	LUSC-B
Project Title	UK Involvement in the Legacy Survey of Space and Time
Document Number	LUSC-B-25

Submission date	21/JUN/2022
Version	1.0
Status	Final
Author(s) inc. institutional affiliation	Mike Read (U. of Edinburgh); George Beckett (U. of Edinburgh); Craig Manzi (U. of Edinburgh)
Reviewer(s)	Aaron Watkins (Herts); Tom Wilson (Exeter)

Dissemination level	
Public	

Version History

Version	Date	Comments, Changes, Status	Authors, contributors, reviewers
0.1	27/JAN/22	Document skeleton	MGB
0.2	27/MAY/22	Mature draft	MAR
0.3	2/JUN/22	Internal checking	MGB
0.4	17/JUN/22	Revisions considering internal checking	MAR and MGB
1.0	1/AUG/22	Updated in light of reviewers comments	MAR and MGB

Table of Contents

Index of Figures

1 Executive Summary

The UK-DAC has been tasked with hosting and serving datasets from external surveys and those produced by LSST:UK work packages. This document describes the initial investigations into ingesting survey catalogues and details the workflows and software required. Two tests of preparing and ingesting data are discussed together with the priorities and next steps for this work.

Ancillary catalogues that need to be hosted alongside the LSST Data Releases will typically be hosted in Qserv, the bespoke, distributed, relational database being developed by the Rubin Observatory. While some archives provide a bulk-download facility, this is not universally the case. The DAC team is likely to need to collaborate with a small number of archive providers to obtain a copy of the ancillary catalogues they host.

There are likely to be differences between the form and format of ancillary catalogues from different archives (for example, related to treatment of null values) and an understanding of these differences is required to successfully ingest and publish the data.

Catalogue ingestion is a resource-intensive and time-consuming task, thus it is worthwhile to invest effort to carefully prepare each catalogue to reduce the risk of problems being encountered late on in the process. The Qserv administrator needs to make decisions regarding the partitioning of data and indexing of the catalogue which will have long-term implications on the utility of the data. At the time of writing, a strategy for making such decisions is still being developed, and we expect this to be informed by the Rubin Data Management team as they scale up their use of Qserv to LSST-scale datasets.

At the time of writing, a catalogue needs to be available as (or converted to) a CSV/ TSV format, to be ingested, even though the CSV/TSV format is inefficient for transferring and handling substantial catalogues. This is related to a limitation in the underlying Qserv database architecture (MySQL).

The ingestion process has the potential to consume twice as much storage as the catalogue itself (during ingestion), though tactics exist for reducing the storage requirement.

The Rubin Data Management team is developing a suite of tools to help with ingestion of catalogues. These can simplify and automate the process. The UK DAC team has supplemented this suite with scripts to automate the preparation of ancillary datasets.

The UK DAC team plans to ingest and publish a number of survey catalogues, over the coming months, including ancillary surveys and previous, user-generated products from DEV activities. This work will provide the prerequisite experience for managing LSST-scale catalogues during Operations.

2 Introduction

2.1 Purpose

Alongside the main LSST Data Releases [4], the UK Data Access Centre (DAC) will host and serve ancillary datasets to facilitate the science goals of the UK community. Here we describe the workflow for obtaining, preparing, ingesting, and publishing these data.

2.2 The Datasets

An up-to-date list of ancillary datasets of interest to the UK community is maintained on the LSST:UK Wiki page (last updated 22nd November 2021):

https://lsst-uk.atlassian.net/wiki/spaces/LUSC/pages/766869516/Identification+of+ancillary+surveys

The datasets primarily consist of catalogues, with images also being requested for some surveys. Catalogues will be hosted in a database system. Images will be stored as files. It is likely that the images will be retrieved and accessed using their metadata stored in Rubin Data Butler [1] repositories or via existing or new SIAP services [2].

In addition to the catalogues listed on the Wiki page, the UK DAC will serve data produced by UK work packages such as WP3.5 (Near infrared data fusion) and WP3.11 (Crossmatching). The mechanism for ingesting and publishing these data will follow a similar workflow.

This document concentrates on catalogue data. The mechanisms for ingesting and sharing image data, via the Rubin Science Platform (RSP) [3], is – at the time of writing – still being finalised. This aspect of data ingestion will be revisited later.

2.3 The database system - Qserv

LSST catalogue data will be hosted in the Qserv database system [5]. Based around MySQL instances, Qserv, at scale, is distributed across tens of nodes, with each node hosting sections of the data. Qserv employs shared scanning techniques to enable fast query times on exceptionally large datasets. Most or all the catalogues from the ancillary surveys will be implemented under Qserv. As use cases become clearer, it might be that some access requirements for ancillary data can be satisfied by calls to existing services hosting the data. For example, Virtual Observatory TAP access is available for most archives and TAP access is supported from the RSP. However, if ancillary datasets are to be queried wholesale alongside LSST data then hosting them in the same Qserv instance is required.

The ancillary catalogues listed on the UK DAC Wiki typically contain of the order of 10^9 objects. Output from WP3.5 will slightly exceed LSST in terms of number of sources and potentially store significantly more attributes.

At the time of writing, the UK-DAC has a Kubernetes-deployed, two-node Qserv instance with 2 Terabytes of disk space per node running on the IRIS Somerville cloud platform [6]. This is sufficient for initial tests but, ultimately, the ancillary datasets will need to be hosted on the same Qserv instance as is used to publish LSST releases.

2.4 Glossary of Acronyms

BCP – Bulk CoPy

CSV – Comma-Separated Values

DC2 – DESC Data Challenge 2

DR – Data Release

MSSQL – Microsoft SQL server

MySQL – My Structured Query Language

REST - REpresentational State Transfer

RSP – Rubin Science Platform

SIAP – SImpe Image Access Protocol

TAP – Table Access Protocol

TSV – Tab-Separated Values

WFAU – Wide-field Astronomy Unit

3 Ingestion Workflow, Generic Version Of

3.1 Obtaining the data

The first step of the ingestion process is to obtain a suitable source copy of the catalogue to be ingested. Though, prior to this, an understanding of a given dataset is needed. Question such as:

- What tables are required?
- What is the schema (columns, datatypes, etc.)?

Care also needs to be taken in identifying where null or default values might occur in the data and ensuring they are handled correctly.

Most archives, hosting astronomy catalogues, offer bulk download methods. In addition, the science archives hosted by the Wide Field Astronomy Unit (WFAU), University of Edinburgh, already have copies of many of the catalogues. At the time of writing, we have not identified bulk download access for a few catalogues (specifically, DES and Spitzer-SERVS), but experience has shown that every archive approached to date has enabled access in at least a scriptable way.

For example, the main catalogues in PanSTARRS DR2 were downloaded using their “casjobs” interface. Tables were downloaded 50—100K rows at a time, resulting in thousands of files for ingest.

3.2 Data preparation

Ideally data to be ingested are to be retrieved in comma-separated-value (CSV)/ tab-separated-value (TSV) format, as this is format required by the second stage. This format is a common download option in existing archives. If the catalogue is not available in this format the data can be converted using, for example, AstroPy or STILTS. This is possibly at the expense of additional disk storage overheads if it cannot be converted on the fly. For PanSTARRS, two queues were used, with some of the output files downloaded directly as CSV, while other sets were retrieved as AstroPy table objects into memory and saved to CSV.

For datasets already hosted in WFAU’s science archives the CSVs can readily be generated by doing a bulk copy (BCP) outgest from Microsoft SQL Server.

3.3 Partitioning

Large catalogues, to be ingested into Qserv, first need to be partition into chunks. With these chunks being further partitioned into sub-chunks. The chunks are then distributed evenly across the nodes during the ingest process. Partitioning is carried out using RA and Dec to divide the sky up into approximately equal-sized regions. An overlap region is also specified to account for small differences in measured coordinates between catalogues or moving sources.

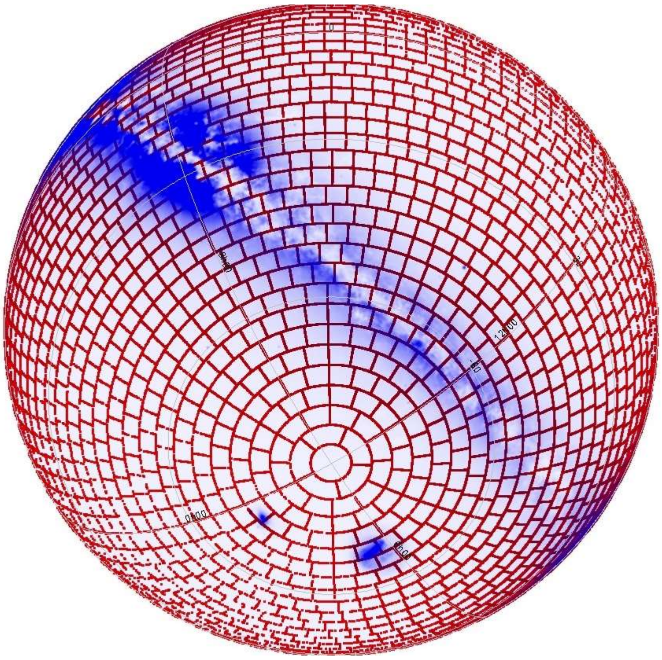


Figure 1 Spherical plot to visualize the chunking.

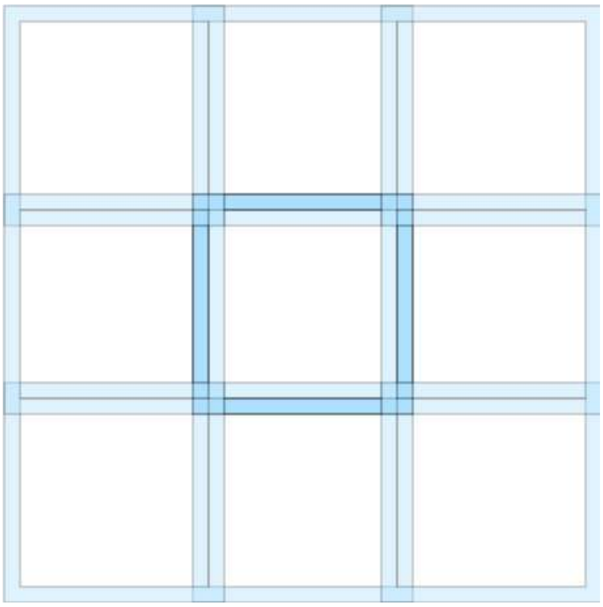


Figure 2 A chunk split into sub-chunks with overlap regions

A tool to perform the partitioning, “sph-partition” (https://github.com/lsst/partition), is provided by the Vera Rubin Observatory.

The size of the boxes is configurable but to be able to combine queries across datasets, the partitioning scheme needs to be the same across the various catalogues. The partitioning parameters and schema of the catalogue being partitioned are supplied via a JSON object (typically stored as a file) together with input file locations and output directory. For example:

```
sph-partition  --config-file=config.json  --in.path=catalogue.csv  --
mr.num-workers=6 --mr.pool-size=32768 --mr.block-size=16 --verbose --
out.dir=chunks/
```

The partitioning into chunks and sub-chunks is specified in terms of number of stripes and sub-stripes (num-stripes, num-sub-stripes).

There is a balance to be struck in choosing chunk size: given the data density, it should be small enough to hold several chunks in memory, but as the number of chunks increases so does the book-keeping requirement at query time. In practice it is likely that we will match the partitioning of the main LSST Qserv releases. At the time of writing, Rubin staff anticipate a few hundred thousand chunks. Data Preview 0.1 (DP0.1) used `num-stripes=340` and `num-sub-stripes=3` which gives a chunk/sub-chunk size of approximately 32 and 10.5 arcminutes. Overlap was set at 1 arcminute.

Smaller catalogues (for example metadata tables) requiring ingest can be replicated in full to all Qserv nodes rather than being partitioned.

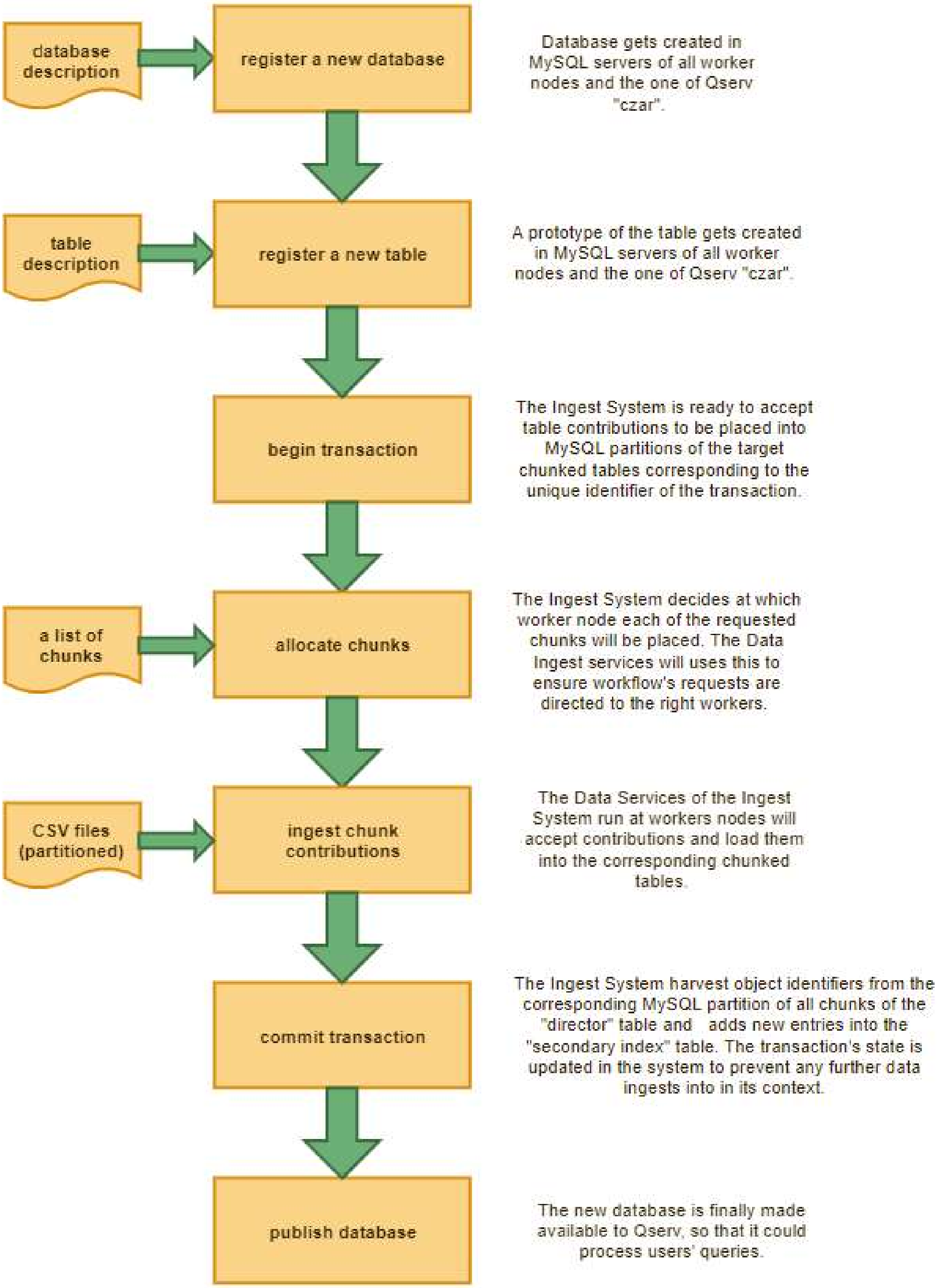
Partitioning the data more than doubles the storage requirement. So, for the largest datasets, partitioning and ingest might need to be done in stages.

3.4 Qserv workflow

Once the data are partitioned, the basic workflow for loading them into Qserv is described in Figure 3, which is reproduced from the Qserv ingest user guide.

https://confluence.lsstcorp.org/pages/viewpage.action?pageId=135954545

At the core of the ingest system is a REST server integrated into the main replication controller.



Database gets created in MySQL servers of all worker nodes and the one of Qserv "czar".

A prototype of the table gets created in MySQL servers of all worker nodes and the one of Qserv "czar".

The Ingest System is ready to accept table contributions to be placed into MySQL partitions of the target chunked tables corresponding to the unique identifier of the transaction.

The Ingest System decides at which worker node each of the requested chunks will be placed. The Data Ingest services will uses this to ensure workflow's requests are directed to the right workers.

The Data Services of the Ingest System run at workers nodes will accept contributions and load them into the corresponding chunked tables.

The Ingest System harvest object identifiers from the corresponding MySQL partition of all chunks of the "director" table and adds new entries into the "secondary index" table. The transaction's state is updated in the system to prevent any further data ingests into in its context.

The new database is finally made available to Qserv, so that it could process users' queries.

Figure 3: Workflow for ingesting catalogue data into Qserv.

Each step in the above workflow can be run by issuing HTTP JSON requests to the REST server.

3.4.1 Register a new database

The database name, partitioning scheme and configuration of the secondary index are specified in a JSON request and posted to the replication REST servers. For example:

```
curl http://localhost:8080/ingest/database -X POST -H "Content-Type:
application/json" -d '{"database":"tmpWP311", "auto_build_secondary_index":1,
"local_load_secondary_index":1, "num_stripes":340, "num_sub_stripes":3,
"overlap":0.01667, "auth_key":""}'
```

3.4.2 Register a new table

Schemas (column names and column types) need to be encoded in JSON format and posted to the ingest table REST service. In addition to the column declarations, the JSON object also contains information about whether the table is partitioned, if it is a director table, and the column names for various key attributes (longitude, latitude, chunk ID and sub-chunk ID). A director table is often the primary table in a database. Other, child, tables are usually linked to it via an index key. In the LSST schema, the director table is Object (deepest stack measurements), and a child table is Source (epoch measurements).

3.4.3 Start a transaction

Before data are loaded into a table a transaction needs to be assigned. A request, specifying the database to use, is made to the replication server. The response contains a unique transaction ID (integer number). Data can then be ingested using that transaction ID. If the ingestion fails for any reason, then the transaction can be aborted, and all data being ingested under that transaction are removed.

3.4.4 Allocate chunks

The partitioned chunks need to be assigned to the correct node for ingestion. Chunk ID numbers are sent to the REST server. The returned JSON contains information about the destination worker node and the port to use.

3.4.5 Ingest chunk contributions

The binary tool, “qserv-replica-file”, is then used to ingest data from a given transaction into the worker node’s Qserv instance. Again, a JSON request is used to drive the execution and many chunks can be sent in a single JSON.

3.4.6 Commit transaction

Assuming no errors are reported, the transaction is committed via another JSON request. If errors are returned then the whole transaction should be aborted and the cause investigated.

3.4.7 Publish the database

After all the data are loaded and there are no open transactions listed for a database it can be published. Once published, the database appears on the main Qserv node and is available for querying.

That completes the ingest workflow. If any additional table indices are required, these are added after publication.

3.4.8 Indices

Database query times can be reduced by creating indices for specific column(s). Indices allow lookups and reduce the need for full table scanning. However, this comes at the expense of increased storage requirements so should only be created for a small subset of columns that are expected to give performance improvements. In astronomy application this is often for example, coordinates (RA/Dec) and the main unique identifier (objectID). As these can be added post publication, additional indices can be investigated for any common slow running queries.

3.4.9 Verifying the ingestion and published database

After publication and initial index creation some test queries should be carried out to confirm the data have ingested correctly. Row counts should be performed on all ingested tables as well as simple lookup queries using the main table identifier (for example, objectID).

3.5 The qserv-ingest tool

The individual Qserv workflow steps described above in Section 3.4 have been wrapped up in the tool, "qserv-ingest".

https://github.com/lsst-dm/qserv-ingest

Written for the ingest of DESC Data Challenge 2 (DC2) [9] into Qserv, it uses the Argo workflow engine [7] to orchestrate parallel jobs and ingest using the Qserv replication service under Kubernetes.

The partitioned data are hosted on a webserver and accessed via URL. A manifest yaml file (ingest.yaml) is used to describe the location of the partitioned data to be loaded and the URLs of the various ingest services deployed on the Kubernetes Qserv install.

A metadata.json file needs to be created in the base directory of the URL specified in the manifest.yaml. This in turn references various JSON files detailing the database, tables, and indices. All partitioned chunk IDs and their relative path locations also need to be specified in metadata.json. As large area catalogues will be partitioned into O(10^5) chunks, this could become unwieldy to represent in a single JSON file so a future version of qserv-ingest will be able to discover and list files on the webserver

A script, "argo-submit.sh", is used to start the process, and can also be used to perform individual steps in the workflow if errors are encountered.

3.6 User access to published databases

Once a Qserv database is published in can be made accessible via the UK RSP. This involves creating a Felis (https://github.com/lsst-dm/felis) metadata file, which describes tables, columns datatypes, etc.

To date we have only done this for a very simple database/table, as a proof of concept. Three columns (sourceId, RA, and Dec) were outgested from the UKIDSS udsSource table. The data were ingested into Qserv, the required Felis YAML file was created and the RSP was rebuilt to include this additional database. This database is queryable though the UK RSP, at the time of writing, only supports synchronous queries for example those actioned via a URL call. For example:

https://rsp.lsst.ac.uk/api/tap/syncREQUEST=doQuery&QUERY=SELECT+TOP+1+*+from+UDS.udsSource&LANG=ADQL

4 Test ingest case – WISE catwise_2020

Early on, the Data Preview DP0.1 catalogues had been ingested to a two-node Qserv instance, but this had been a naively followed recipe on an already prepared and external dataset. The Qserv ingest system API had also undergone several revisions since then (at this time of writing, version 8, compared to version 4 for initial experiments).

The WISE catalogue, catwise_2020, was used to investigate the ingestion of ancillary datasets. It was chosen because it is one of the requested surveys, it has all-sky coverage with some 10^9 sources and the data were already held in the WFAU Science Archives.

Two subsets of catwise_2020 were used: a 1 in 1000 sample of the full catalogue distributed across the whole sky and a fully sampled region in RA/Dec (15 <= RA <= 85 and 50 <= dec <= 60).

The two datasets were exported to CSV files from Microsoft SQL Server using the BCP command. The resulting sampled all-sky CSV file contained around two million sources and the sky area CSV file, around twenty-five million sources.

Each file was then separately partitioned using sph-partition and the same scheme as DP0.1, that is `num-stripes=340` and `num-sub-stripes=3`, overlap 0.016667. The partitioning of the sampled data produced 290,532 chunk files in around 20 minutes. The fully sampled data were partitioned in a similar time to approximately 3,000 files. It was noted that over ten million sources were present in the overlap files which represents a significant overhead (40% of the full catalogue). As ancillary datasets are to be queried alongside LSST data releases then the optimum overlap size will be determined and set by the Vera Rubin Observatory. Not all queries will be affected by overlap overheads, for example full table scans will not need to search through the overlap data as these are held in separate tables.

4.1 Ingesting the sampled data

To gain an understanding of the individual steps, the sampled data were ingested following each of the steps described above in section 3.4. The JSON required to describe the table was generated from the existing Microsoft SQL schema. As the step allocating the chunks required many chunkIDs to be requested, a simple Python script was written to submit the JSONs and parse the returned JSON object. The chunk files were then allocated and ingested in batches of 10,000 (qserv-replica-file). Despite the small total number of sources, the ingestion took nearly 90 minutes due to the large number of small requests/files.

Once ingested, the published the catalogue was verified via performing simple count and main ID lookup queries. Again, despite the smallish number of rows the count took longer than might be expected due to the large number of individual partitioned tables on each of the two nodes.

4.2 Ingesting the sub-area data

These data were loaded into Qserv using the qserv-ingest tool. JSON files (metadata.json, and so on) were created describing the database setup and contents. The files and the partitioned data were then hosted on a web server.

A manifest file was created on the Qserv jump node that points qserv-ingest to the metadata.json file and the URLs within the ingest system: query_url, queue_url and replication_url.

The OVELAY variable was then set in env.sh referencing the manifest directory to use and then argo-submit.sh was executed.

The twenty-five-million row table loaded in just over 15 minutes using ten parallel ingest jobs.

SQL queries were again carried out to verify the ingest. These were done manually but this can also be made a step in the qserv-ingest argo workflow. This is done using a dbbench.ini file in the manifest folder.

4.3 Qserv Prerequisites

The setup and administration of a Qserv instance is described in [8]. However, at a high level, a Qserv instance (or, at least, the topical version of Qserv) has the following prerequisites:

- A working Qserv Operator setup (https://qserv-operator.lsst.io/)
- Argo (https://github.com/k8s-school/argoproj-helper)
- Helm (https://helm.sh/)
- Skateful (https://github.com/k8s-school/skateful)
- Qserv-ingest (https://github.com/lsst-dm/qserv-ingest)

Specifically, for DP0.1, one also needs access to a source copy of the catalogues – for example, on the ‘ccnetlsst’ servers hosted by CC-IN2P3.

5 Summary of Supported Software and Processes

Software used for this activity will be maintained in a GitHub repository:

https://github.com/lsst-uk/DataIngesttion-WP2.2/

5.1 Download tools/scripts

For data already held in our other archives, it is straightforward to outgest data via BCP that can directly be fed to sph-partition.

For other datasets, each external archive will have their own interface for obtaining bulk downloads, and a combination of bespoke Python scripts (possibly employing VO interfaces, such as pyVO) and wget/curl is envisaged.

5.2 Data conversion

It is anticipated that all datasets can be obtained in delimited ASCII format that can then be directly used by sph-partition. If this turns out not to be the case or datasets are too large to partition in one go, then they can be converted and split using AstroPy or STILTS.

The sph-partition tool will be employed to partition catalogues where necessary.

5.3 Qserv ingest scripts

We will use Argo workflow and qserv-ingest tool to load data into Qserv. As this tool is being developed to handle the LSST data releases it will be capable of ingesting the ancillary datasets. Our experience of qserv-ingest is generally good and it is still being actively developed and improved. Currently qserv-ingest can only handle CSV delimited input files, as large datasets require partitioning this is not an issue as sph-partition has CSV as an output option (the default being TSV)

A set of JSON files need to be generated for each dataset that then allow qserv-ingest to create the database structure and ingest the partitioned data. We are still trying to understand the best way of routinely generating these description files. A Felis format file is also required to publish the data via the RSP so one approach might be to start with that and then parse that base YAML file. Some of the datasets will have hundreds or thousands of columns so an automatic way of generating the required file is needed. We have metadata files for the datasets held in our archives and other external archive have similar information. A lot of the information is discoverable via the VO. Simple python scripts should be able to parse these sources into the correct formats.

Ingests will be verified using test queries described in qserv-ingest's dbbench.ini file.

6 Priority Ancillary Surveys for Phase B

In terms of prioritising datasets to ingest, WISE and Gaia have been chosen as they offer all-sky coverage, are a reasonable size (1–3 billion sources) to test the system under load and they are already being used by WP3.11.

In parallel we will investigate and begin downloading some of the datasets currently not already available in our archive system, starting with Spitzer and DES.

We will also work on the schema for WP3.5 data and ingest their runs as they become available. This will start with the ingestion of their recent Generation 3 Butler run of the SXDS field.

We will also be ingesting further LSST data preview datasets (DP0.2, DP1, DP2) as they become available.

Once we have performed ingests at scale, efforts will concentrate on downloading external datasets and building the metadata files to drive their ingestion. We will also start investigating how to host and serve image data, starting with those being produced by WP3.5.

6.1 Excluded datasets

At this stage we have no plans to ingest the HSC catalogue data. Although initially identified as being required, the work-packages using HSC have their own working copies and there was no response when the UK community were explicitly asked if the data needed to be ingested into Qserv.

6.2 Timeline

The priority datasets described above will be ingested to our new 10-node Qserv instance. Subsequent external ancillary ingests in full might then need to wait until the LSST partitioning scheme is decided upon and our production Qserv instance becomes available. The reason for this is that if the partitioning scheme changes then the data will need to be partitioned again. If a new Qserv instance is built the data would likely need to be re-ingested as it is not trivial to move the underlaying database files around.

Efforts will therefore move to obtaining UK-DAC copies of all other datasets, generating the required metadata and working on the WP3.5 and WP3.11 ingests to support their work.

It is anticipated that there will be several data releases for WP3.5 and WP3.11. However, as LSST will quickly exceed the depths of VISTA and the cross-matched surveys it is expected that later LSST releases will not always trigger new ancillary releases for WP3.5. During Phase C of operations, the following releases are envisaged:

- LSST Crossmatch Catalogues (based on DP1), Mar’24
- LSST-VISTA fused data release (based on DP1), Sep’24
- LSST-VISTA fused data release (based on DR1), Sep’25
- LSST Crossmatch Catalogues (based on DR1), Sep’25
- LSST-VISTA fused data release (based on DR2), Mar’26
- LSST Crossmatch Catalogues (based on DR2), Mar’26
- LSST-VISTA fused data release (based on DR3), Mar’27
- LSST Crossmatch Catalogues (based on DR3), Mar’27

7 References

[1] Timothy Jenness, *The Vera C. Rubin Observatory Data Butler and pipeline execution system*, SPIE 2022 Conference Presentation, Paper 12189-40.

[2] Patrick Dowler, Doug Tody, François Bonnarel, *IVOA Simple Image Access Version 2.0*, IVOA Recommendation 2015-12-23 (https://www.ivoa.net/documents/SIA/20151223/, accessed 3rd June 2022)

[3] G. Dubois-Felsmann, K. T. Lim, X. Wu, et al., *LSST Science Platform Design*, Rubin Observatory report LDM-542 (2017), https://ls.st/LDM-542

[4] M. Jurić, et al., *Data Products Definition Document*, Rubin Observatory report LSE-163 (2021), https://docushare.lsstcorp.org/docushare/dsweb/Get/LSE-163/.

[5] D. L. Wang et al, Qserv: A distributed shared-nothing database for the LSST catalog, 2011, https://www.researchgate.net/publication/254003056_Qserv_A_distributed_shared-nothing_database_for_the_LSST_catalog, accessed 17th June 2022

[6] Somerville service https://lsst-uk.atlassian.net/wiki/spaces/LUSC/pages/2756968452/Somerville+Service, accessed 17th June 2022

[7] Argo workflows - https://argoproj.github.io/argo-workflows/, accessed 17th June 2022

[8] Qserv installation on Somerville, https://lsst-uk.atlassian.net/wiki/spaces/LUSC/pages/2964455429/New+Qserv+install+on+Somerville+MkII, accessed 17th June 2022

[9] Abolfathi, B, et al., *The LSST DESC DC2 simulated sky survey*, Astrophysical Journal, Supplement Series, 2021, 253(1), 34.