Australian Government

Geoscience Australia

Generalised Data Framework

Solution Architecture

Author:	Alex Ip
Document Version:	0.3
Status:	Draft
Date Saved:	27/07/2015
TRIM Number:	D2015-48046

Amendment History

Version #	Date Created	Amendment Details	Author
0.0	03/03/2014	Initial draft created from D2014-211083	Alex Ip
0.1	1/05/2015	Minor edits & updates	Alex Ip
0.2	19/05/2015	Draft ingestion flowcharts added	Alex Ip
0.3	27/07/2015	Status updated to reflect GDF trial	Alex Ip

CONTENTS

FIGURES

TABLES

1 Executive summary

The development of the Australian Geoscience Data Cube (AGDC) in early 2013 enabled the temporal analysis of the entire available 27-year Landsat archive of Australia for the very first time. The careful partitioning and indexing of the data combined with the use of the high-volume persistent parallel file systems and computational resources at the National Computational Infrastructure (NCI) permitted the efficient exploitation of the HPC resources there to complete high-resolution, multi-decadal, continental-scale analyses in a matter of hours.

While the original implementation of the AGDC was useful to support current use cases against the Landsat archive and other EO data collections with similar or smaller volumes (e.g. MODIS), it was necessary to plan and develop the next version of the system to cater for future needs. In particular, the system needs to cater for data collections with extremely large numbers of repeated observations through time, or even those with higher dimensionality such as sub-surface geophysics, atmospheric or ocean data.

The original AGDC was developed to meet operational needs using Earth Observation (EO) data, so its data model was effectively tailored to the EO domain. In order to better support cross-domain data fusion analyses involving non-EO regular gridded data, it has been necessary to develop a more generic, preferably extensible, implementation based on a new multidimensional data model.

The original AGDC was based around 2D tile files indexed using a relational database. These tiles were arranged as required to simulate multi-dimensional data structures. While adequate for immediate needs, it was clear that this approach would not scale out to adequately manage gridded data collections with an extremely large number of repeated observations through time. One major difficulty in designing a flexible and extensible solution is that not all future use cases can be known, so a more dynamic approach was required.

A critical requirement for any new development for the AGDC would be that it must continue to support current use cases without any interruption. For example, from a user's perspective, the Application Programming Interfaces (APIs) which were developed against the existing 2D system need to function identically. Similarly, the "Pixel Drill" interface which uses temporally optimised files would also need to be supported in the same framework.

This document details some of the drivers for change and outlines the design of a comprehensive redevelopment which manages the generalised multidimensional storage of regular gridded data in an extensible framework which is intended to eventually support other data structures including regular and irregular point clouds. This development is known as the Generalised Data Framework (GDF), and its development has taken place within the Geoinformatics and Data Services Section of Geoscience Australia.

2 Introduction

2.1 Purpose

The purpose of this design document is to describe the architecture of a General Data Framework to manage regular gridded data based on current and projected needs. This next phase of development will take place while the original version of the AGDC is meeting the immediate needs of stakeholders and will provide a flexible, sustainable and interoperable engine for the AGDC into the future.

The intended audience of this document is both scientist/developers and ICT staff including but not limited to architects.

2.2 Background

The original Australian Geoscience Data Cube (AGDC) was initially developed in early 2013 to meet operational requirements for the temporal analysis of Petabyte-scale Landsat Earth Observation (EO) data produced during the "Unlocking the Landsat Archive" project in 2012-2013. The Landsat EO data has been carefully cross-calibrated and prepared as High Performance Data (HPD) to facilitate High Performance Computing (HPC) on the National Computational Infrastructure (NCI) supercomputer, Raijin.

The original AGDC essentially functioned as a database-indexed filestore in which the data is stored as regular storage units (i.e. files) and the associated metadata (including spatio-temporal attributes) is stored in a separate, relatively small database. This approach allows the data to be queried and presented for analysis in many ways while exploiting the high-capacity parallel filesystems of the NCI.

The general approach employed by the initial version of the AGDC is to use GDAL (Geospatial Data Abstraction Library) library functions to partition the spatially-irregular 2D source datasets into spatially-regular, time-stamped 2D tiles. These tiles are indexed using a Relational Database Management System (RDBMS) and this allows the tiles to be queried and arranged in structures including dense 3D x-y-t temporal stacks for temporal analysis.

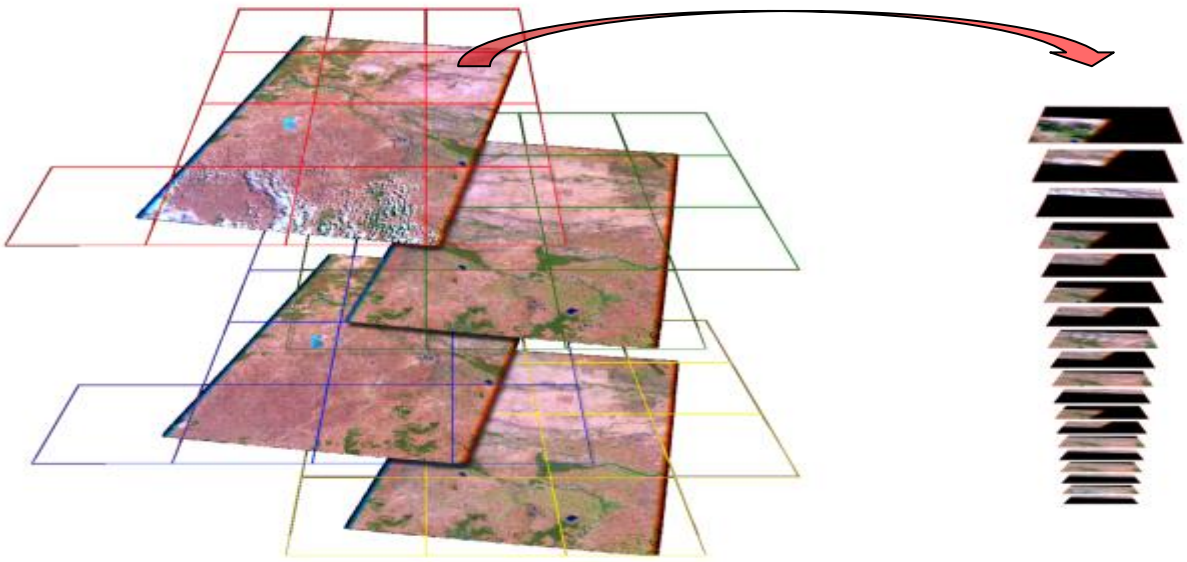


Figure 1 - The 2D "dice'n'stack" approach employed by current AGDC

While extremely flexible and perfectly adequate for the current Landsat data collection (currently up to 1200 timeslices per cell for the whole 27-year archive), this "dice'n'stack" approach is subject to inherent limitations in dealing with temporally-deep (i.e. high observational frequency) data collections such as Himawari 8/9 which deliver many thousands of observations which must be traversed through time.

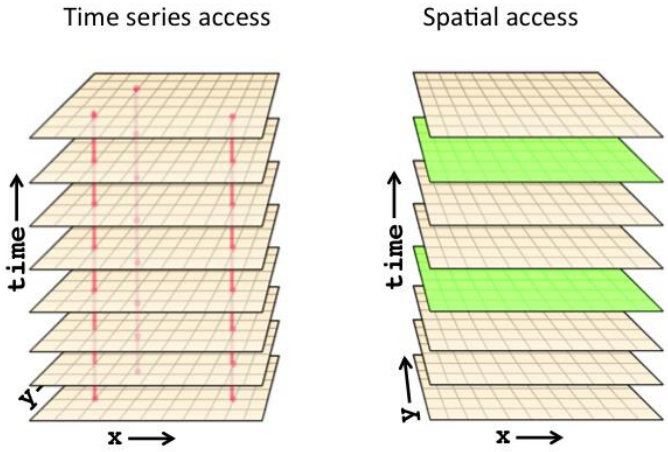


Figure 2 - Time Series access vs. spatial access (Unidata, Chunking Data: Why it Matters, 2013)

Aside from the whole-of-archive processing undertaken "natively" through the Python API, there is an important requirement to deliver seamless spatio-temporal subsets of data via web services such as WMS & WCS. Ideally, established third-party systems (e.g. Hyrax, OPeNDAP, etc.) should be employed in preference to the design, development, testing and maintenance of bespoke subsystems wherever possible.
Some of the issues with the original AGDC approach are as follows:

1. The original AGDC system is highly EO specific, with entities such as "satellite", "sensor" and "band" represented by specific tables and fields in the database schema. The AGDC has been marketed as a system capable of delivering cross-domain data-fusion analyses with heterogeneous gridded data collections, but such collections can only be dealt with as "fake" EO data using the original schema.
2. Large numbers of relatively small tile files are required for large collections. For example, there are approximately 9x more tile files required than Landsat source scene datasets. At the time of writing, there are over 8M tiles for the Landsat archive, and this number continues to increase over time. Whilst the PostgreSQL database is capable of scaling up to handle many millions of records, there will inevitably be limitations to the capacity of the filesystem used to store extremely large numbers of small tile files without resorting to measures such as partitioning.
3. While flexible, 2D tiles are simply not efficient for time-series analyses of temporally-deep collections. Each timeslice is currently represented as a single file; so assembling many individual timeslices into temporal stacks will require a large overhead in file handling. Many temporal analyses require an entire pixel-based time series to fit in memory, so an x-y-t temporal stack must be read in as a series of "straws" of limited spatial extent (potentially down to a single pixel). As temporal depth increases, spatial extent must be reduced in order to maintain the same volume of data read to fit into memory. This means that once these memory constraints come into play, the number of file open operations will actually be proportional to the square of observation count rather than the observation count itself.
4. 2D tiles need to be aggregated to assemble seamless spatio-temporal ranges for delivery via web services (WCS, etc) using bespoke mechanisms at considerable I/O cost. This approach may cause some third-party delivery mechanisms to fail when attempting to deal with the entire 450TB+ archive as a single, seamless spatio-temporal aggregation.

5. The GDAL[1] data model limited to 2D multi-variate (i.e. multi-band) files, and cannot handle multi-variate, multi-dimensional structures. GDAL-readable datasets do not provide any temporal context for timeslices other than through custom per-band metadata. Currently we create one temporal stack file per temporally-varying quantity during analysis, resulting in an even larger number of intermediate 2D files being required for complex analyses.
6. The GDAL-readable (GeoTIFF) storage units employed in the original AGDC are effectively limited to three (i.e. 2+1) dimensions (e.g. x-y-λ or x-y-t) and cannot be used to store data of higher dimensionality (e.g. x-y-z-λ-t). In the original system, higher dimensionality can only be simulated through accessing exponentially larger numbers of 2D storage units in sequence specified by the indexing database.

Due to the issues described above, it has been necessary to plan and design the Generalised Data Framework (GDF) as the successor to the original AGDC while still meeting operational needs with the original version. Note that, ideally, the interfaces for the APIs which were designed and implemented against the original system would not need to change in the short term, but the implementation behind them could be redeveloped to exploit the new features and efficiencies provided by the GDF.

2.3 Scope

The scope of this document is as follows:

Items in scope:
- Multidimensional Data model and database schema development
- File storage driven from database (using netCDF4/HDF5)
- Ingestion of regular gridded datasets (including full metadata)
- Design for potential integration of regular and irregular point clouds and other data structures
- Future integration of the AGDC version 1 API

Items outside scope
- Product generation (i.e. upstream pipeline prior to ingestion)
- Data configurations other than georeferenced, regular gridded (raster) data
- Specific AGDC analytical applications using API, or API design
- Delivery of data via OGC-compliant web services (to be implemented with NCI)
- Consumers of web services
- Implementation of regular and irregular point clouds and other data structures

[1] Geospatial Data Abstraction Library - http://www.gdal.org/

2.4 Assumptions & Constraints

The following assumptions are made in developing this architecture document:

- Sufficient high-speed persistent storage is available attached directly to processing nodes by high-bandwidth links (as at the NCI).
- Scientific applications will be developed against the AGDC primarily using Python, but other languages and tools should be supported later.
- A working, populated prototype of the new system should be in operation by June 30th, 2015..
- All code developed will be open-sourced and not dependent on any particular infrastructure.
- No single access mode will be favoured over others (e.g. spatial vs temporal) for the main data store. Ideally, data should not be duplicated in different formats to support different access modes.

2.5 Glossary of terms

Terms/Acronym	Description

2.6 Bibliography

OGC. (2012, 06 12). *Earth Observation Metadata profile of Observations & Measurements.* (F. H. Jerome Gasperi, Ed.) Retrieved 11 18, 2014, from Observations & Measurements | OGC: https://portal.opengeospatial.org/files/?artifact_id=47040

OGC. (n.d.). *OGC network Common Data Form (netCDF) standards suite | OGC*. Retrieved 11 17, 2014, from Open Geospatial Consortium (OGC): http://www.opengeospatial.org/standards/netcdf

OGC, O. G. (n.d.). *Observations and Measurements | OGC*. Retrieved 11 17, 2014, from Open Geospatial Consortium (OGC): http://www.opengeospatial.org/standards/om

Oldfield, S. (2014, 10 14). *Govdex AGDC Users Forum - Data Cube APIs.* Retrieved 10 22, 2014, from Govdex: https://govdex.gov.au/confluence/pages/viewpage.action?pageId=283130420

Robertson, C. (2014, 11 11). *Multidimensional File Aggregation - Australian Geoscience Data Cube Collaborations - Dashboard.* Retrieved 11 18, 2014, from Govdex (Australian Commonwealth Government): https://govdex.gov.au/confluence/display/AGDC/Multidimensional+File+Aggregation

Unidata. (2013, 01 29). *Chunking Data: Why it Matters.* Retrieved 10 22, 2014, from Unidata: http://www.unidata.ucar.edu/blogs/developer/entry/chunking_data_why_it_matters

Unidata. (2013, 09 25). *Unidata | NetCDF.* Retrieved 10 22, 2014, from Unidata: http://www.unidata.ucar.edu/software/netcdf/

3 Original State

3.1 Logical Model

At its highest level, the original AGDC can be regarded as a database-indexed store of regular, spatially-partitioned, time-stamped 2D tile files.

Below is a diagram showing the highest level view of a generic AGDC application using the original implementation with 2D (GeoTIFF) files used as the storage units indexed by the RDBMS. The core of the AGDC is represented by the rectangular box containing the data cube index and data cube data subsystems.

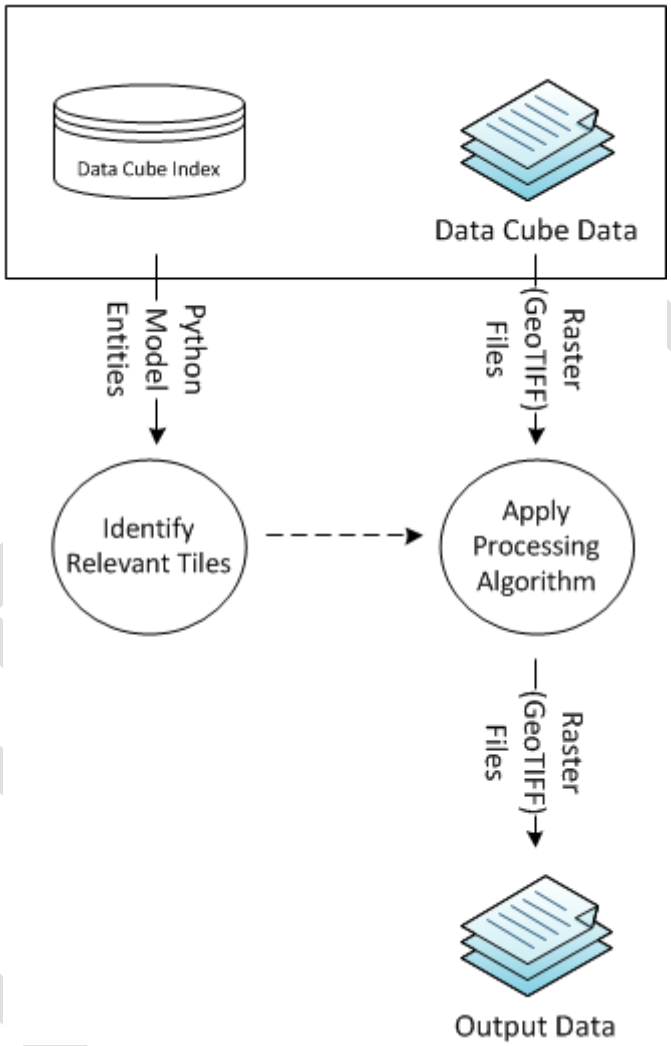


Figure 3 - Anatomy of an AGDC application (Oldfield, 2014)

Note that any development of the AGDC would still conform to the high level model shown above, (Figure 3) in that it would still consist of separate storage and indexing subsystems.

The logical model for the original AGDC is highly EO-specific, with entities such as "satellite", "sensor" and "band" as shown in the following diagram:

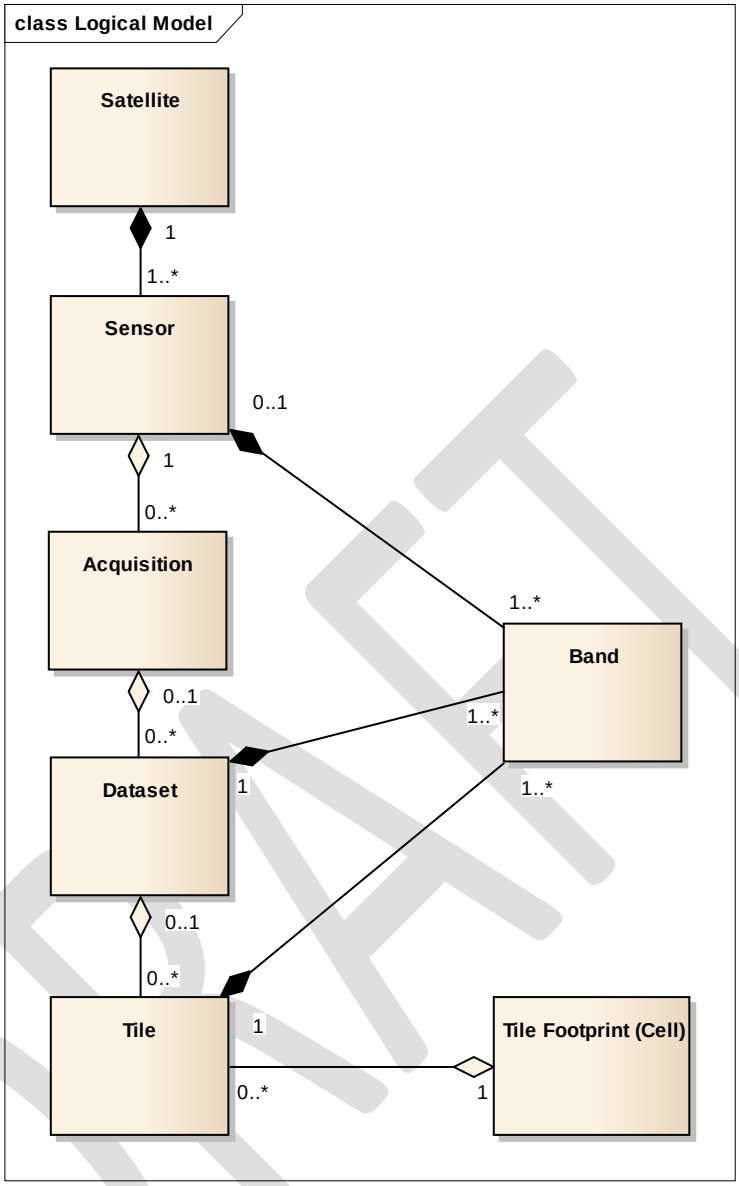


Figure 4 - Original AGDC EO-specific logical model

The objects in the original AGDC logical model implemented in the database schema can only be used manage non-EO data if arbitrary mappings are made between domain-specific entities. For example, in bathymetry data, "vessel" could be mapped to "satellite", "instrument" to "sensor", "sounding" to "band", etc. Such arbitrary mappings across domains could rapidly become unwieldy and confusing. Explicit and consistent mappings between schema entities and entities in the Observations and Measurements (O&M) (OGC) schema should also be identified and managed.

The initial design of the AGDC schema was intended to manage only basic metadata including the spatio-temporal parameters of the observations and sufficient metadata to reference the source dataset for other, more detailed, metadata. Operational requirements, combined with the archiving of some source datasets, have resulted in the inclusion of more domain-specific metadata attributes in the database. These additional metadata attributes are managed as individual fields added to the relevant tables, and require specialised query extensions to search, filter and sort data using these fields.

3.2 File Storage Layout

The existing UTM scene-based Landsat (5, 7 & 8) data has been (un)projected to EPSG:4326 and divided up into 1 degree x 1degree tiles of 4000 x 4000 pixels. Each tile file is structured to reflect the contents of the source dataset (or the relevant subset of bands). Under this tiling schema, separate tile files have been created to hold the 6-7 reflective bands from the NBAR dataset, the 1 or 2 thermal band(s) from the L1T, the single band of PQA data and the four bands from FC as shown in Appendix A (section 8). At this point, we are using LZW-compressed GeoTIFF as the storage format.

There is no predefined structure required for the AG-DC file store, and each tile record in the database has a self-contained tile_path field which holds the fully-qualified location of its associated tile file. The original file store for Landsat data has been arbitrarily structured as follows:

<tile_root_directory> (Directory)

<satellite>_<sensor> (Directory)

<tile_x_index>_<tile_y_index> (Directory)

<year> (Directory)

<tile_dataset>

<tile_dataset>

...

mosaic_cache (Directory)

<n-s_overlap_mosaic>

<n-s_overlap_mosaic>

...

Figure 5 – Original Landsat Tile Storage Layout

Note that the mosaic_cache directory contains mosaics for N-S scene overlaps for Landsat data. These are either VRT files referencing two <tile_dataset> files in the parent directory, or, in the case of the PQA datasets which have no defined no-data value, geoTIFF files containing a mosaic of the two PQA <tile_dataset> files in the parent directory,

3.3 Database Schema

Please refer to Appendix B (section 9) for detailed documentation on the original AGDC database schema.

4 Requirements

The following business requirements are drivers for the development of the next generation of the AGDC. We need to:

- Support scalable, efficient processing and delivery of large-scale, temporally-deep data collections
- Ideally provide a generic, extensible system for managing all forms of multidimensional, regular gridded data across diverse domains including subsurface, marine and climate.
- Conform to existing best practices in the management of large-scale gridded scientific data
- Avoid duplication of data in different structures for different purposes
- Leverage existing third-party tools and delivery mechanisms wherever possible
- Support as many different modes of access and delivery mechanisms as possible, including OGC web service standards
- Leverage domain expertise and HPC infrastructure through our partnership with the NCI
- Support large-scale processing environments other than NCI (e.g. AWS, etc)
- Maintain flexibility and future-proofing through modular architecture

5 Architecture Recommendation

5.1 Fully generic, parameterised multi-dimensional storage system using a standard file format such as NetCDF

The most comprehensive approach to the proposed development would be to preserve the existing highest-level architecture of the AGDC based on a RDBMS-indexed file store, but extend it to manage fully-parameterised, multi-dimensional storage units instead of 2D tiles. These multidimensional storage units could be aggregated as required to produce effectively seamless access to data across the relevant dimensions, especially for delivery via web services.

Although NetCDF provides an archetype for the multidimensional file format due to the completeness of its data model and implementation, the data model should be applicable to any gridded data format of any dimensionality, even including the original 2.5D geoTIFF file format if the configuration is constrained accordingly. However, it would not be desirable to expend significant resources implementing a geoTIFF driver to support the existing filestore due to the inherent limitations of this 2D format.

As of mid-April 2015, the generalised multidimensional database schema for the GDF has been successfully scale-tested by remapping the metadata stores for the entire Landsat and MODIS holdings from the original AGDC. A few test queries have been implemented against all of this metadata (concurrently across multiple databases) from behind a low-level, declarative demonstrator API and the database appears to be more performant than the original AGDC. It should be noted that, at this stage, this work has so far involved only the metadata, and not the actual raster data.

As at June 30, 2015, a working prototype implementation of the GDF was completed to the point where a 1TB test area was converted from AGDCv1 tiles into managed multidimensional NetCDF-CF files and the data accessed seamlessly as arrays via a low-level data access API. Note that a generic ingestor still needs to be implemented in order to process data directly rather than via the AGDCv1 tiles.

5.1.1 Key Features

5.1.1.1 Multidimensional / Multivariate Storage

To ensure continuing viability of the AGDC, we need to prepare for large volume, temporally-dense data collections (e.g. Himawari 8/9) by applying regular temporal partitioning to the temporally-aggregated data in a manner analogous to the regular spatial partitioning originally

applied to the 2D data. For example, a multidimensional storage unit could hold an arbitrary 1° longitude x 1° latitude x 1 year of Landsat data, which would require only 27 file open operations instead of around 1200 2D file open operations to access the full temporal depth of the Landsat archive.

Multiple quantities in a single observation such as spectral bands for EO data can be stored either in individual structured (x-y-t) variables within a storage file or, if they are of a homogeneous datatype, as indexed locations in an additional array dimension (λ). The new system should permit either approach.

5.1.1.2 Generalised Dimensionality with Parameterised management

Using GDAL-compatible storage files, EO data was originally stored as either x-y-t or x-y-λ. The new system generalises dimensionality to manage different dimensionality such as x-y-z-λ-t. The ordering of the dimensions has a direct bearing on the structure of the array, with the fastest-varying axis providing the best performance through being able to access contiguous data without "striding".

Consistent with the principles applied to the original AGDC, all degrees of freedom of the multidimensional storage format should be fully parameterised to provide complete flexibility through configuration of database records. There will necessarily be parameters governing the entire storage unit (e.g. datatype) as well as parameters governing each dimension of the storage unit. NetCDF also provides a buffering system called "chunking" where the buffer size can be configured for each dimension.

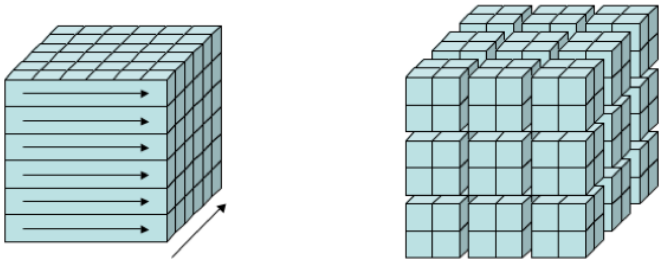


Figure 6 - Contiguous vs. Chunked array data (Unidata, Chunking Data: Why it Matters, 2013)

For example, NetCDF could provide the following degrees of freedom for each dimension of the array in the storage unit:

- Resolution (data element size)
- Extent (overall size)
- Chunk Size (i.e. buffering parameters)
- Reference System / Origin / Unit of Measurement
- Order within array

An exploration of some of these degrees of freedom was conducted by Calum Robertson of the Cooperative Research Centre for Spatial Information (CRCSI) for GA (Robertson, 2014), and the results can be viewed on the AGDC Govdex site. This study was of limited use, however, as it failed to examine the benefits of chunking and was limited to testing the reading and writing of entire netCDF files rather than small subsets using chunking.

Using the arbitrary x-y-t example of 1° longitude x 1° latitude x 1 year of Landsat data, the following quantities would need to be managed (shown with possible values):

- x-axis resolution = 4000 pixels/degree
- x-axis extent = 1 degree longitude
- x-axis chunk size = 128 elements (for example)
- x-axis CRS / origin / unit = EPSG:4326 / 0° longitude / degree

- y-axis resolution = 4000 pixels/degree
- y-axis extent = 1 degree latitude
- y-axis chunk size = 128 elements (for example)
- y-axis CRS / origin / unit = EPSG:4326 / 0° latitude / degree
- t-axis resolution (i.e. time increment) = 1 second
- t-axis extent = 1 year
- t-axis chunk size = 64 elements (for example)
- t-axis CRS / origin / unit = seconds since epoch / 01/01/1970 0:00 / seconds

Such a completely parameterised approach would permit the construction of an automated optimisation engine to determine the "sweet-spot" in parameter settings in order to deliver the best possible performance for specified use-cases in a given environment.

5.1.1.3 Standard Multidimensional Storage Format (e.g. NetCDF)

In order to manage multidimensional array data, it is necessary to select or develop a file format in which to store the actual data. The cost of designing, developing, testing, implementing and maintaining a bespoke format would be extreme compared to the cost of adopting an established format and leveraging third party development against that format, so the latter course of action would clearly be preferred provided that the format selected could deliver acceptable functionality and performance.

The network Common Data Form (NetCDF) Climate and Forecasting (CF) format is an Open Geospatial Consortium (OGC) standard which is the most promising candidate storage format due to the richness of its multidimensional data model and the availability of third-party tools. Led by the weather and climate science community, the global scientific community is converging on NetCDF as the dominant format for the management and exchange of array-oriented scientific data. Delivery mechanisms such as THREDDS, Hyrax and OPeNDAP have already been implemented against NetCDF, and GA has already completed some limited proof-of-concept development against this format in the context of the AGDC. It should be noted that, in addition to many large scientific agencies overseas including JAXA, ESA & NASA, the Australian Bureau of Meteorology (BoM), NCI and CSIRO are committed to storing and delivering their gridded data in NetCDF format.

From the Unidata website (Unidata, 2013):

Unidata's Network Common Data Form (netCDF) is a set of software libraries and machine-independent data formats that support the creation, access, and sharing of array-oriented scientific data. It is also a community standard for sharing scientific data. Data in netCDF format is:

- **Self-Describing.** A netCDF file includes information about the data it contains.
- **Portable.** A netCDF file can be accessed by computers with different ways of storing integers, characters, and floating-point numbers.
- **Scalable.** Small subsets of large datasets in various formats may be accessed efficiently through netCDF interfaces, even from remote servers.
- **Appendable.** Data may be appended to a properly structured netCDF file without copying the dataset or redefining its structure.
- **Sharable.** One writer and multiple readers may simultaneously access the same netCDF file.
- **Archivable.** Access to all earlier forms of netCDF data will be supported by current and future versions of the software. (Unidata, 2013)

Formats such as native HDF5 are also popular for managing gridded data, but care must be exercised because the bare HDF5 format is far less constrained than its netCDF4-wrapped HDF5 implementation, so every application is effectively customised and there are consequently no generic utilities available "out-of-the-box". This means that although HDF5 provides low-level libraries to assist in the manipulation of files, its unconstrained use could effectively create a bespoke format which would require greater development effort to deliver higher-level functionality such as web services.

5.1.1.4 Data Model and Schema

The data model and resultant schema design to manage the storage units should ideally provide the following features:

- Domain-neutrality (i.e. not Earth Observation specific)
- Storage format agnosticism (not specifically dependent on NetCDF or any other file format)
- Generalised dimensionality (not limited to x-y-t: could also manage x-y-z-t or x-y-z-λ-t)
- Multiple measurements (e.g. spectral bands in EO datasets) stored either as individual variables or as an additional array dimension depending on the constraints of the storage format and data

Note that the original system is based on a relational database, but the option exists to investigate other database models for cataloguing the same files.

5.1.2 Pros and Cons

The adoption of a multidimensional storage format and a fully-parameterised management system would have the following pros and cons:

Pros:

- Increased storage & retrieval efficiency (fewer files, more contiguous reads)
- Improved performance for temporal data access (or across another dimension)
- Unified approach to all regular gridded data
- Same system could manage both raw data and derived intermediate results
- Potentially able to aggregate storage units for seamless access to data outside AGDC
- Ability to leverage third-party utilities and libraries (e.g. THREDDS, Hyrax, OPeNDAP, etc)
- Complete alignment with clear directions in wider, global scientific data management practices (e.g. climate science)
- Ability to handle multiple and/or higher dimensionalities – not limited to x-y-t.

Cons:

- Greater design and development effort required
- Increased complexity to implement completely generalised system. Need to provide simpler domain-specific views.
- Reduced flexibility – unable to insert/pre-pend new data without restructuring storage files
- Need to re-write ingestion and API implementations to work with temporally-aggregated data
- Unable to use some "mainstream" visualisation tools with multidimensional storage unit files
- Need to batch ingest – not advisable to add individual new datasets in a piecemeal fashion. (N.B: Could update existing data or append new data, but not insert new data into an existing range)

Special notes on the netCDF format:

- Any major change in the netCDF specification should not necessitate a complete rewrite of the archive, as the netDCF specification maintains backwards compatibility (see "archivability" in section 5.1.1.3). A collection-level reformatting would only be undertaken if new features of the revised format were considered desirable (e.g. better performance).
- A change of metadata content or format would not necessitate a complete re-write of the archive, as the authoritative metadata should be managed in the database and updates could be applied retrospectively to existing files if required.
- Although the netCDF library is not available by default on many operating systems and may need to be compiled, it is an increasingly common component for scientific computing. This is also true of HDF5.

5.1.3 Implications for End Users and Systems

As an interim measure, the AGDCv1 API for science applications could be retrofitted to the new GDF data access API. End users writing applications against the revised AGDCv1 API not be affected in any way, as the API itself should not need to change, only the implementation behind the interface. It should be noted, however, that the retrieval of 2D spatial arrays from the GDF in order to directly simulate the current operations from 2D geoTIFF files is not likely to be the most efficient way in which to access the data from a multidimensional storage system.

One key difference between the original 2D AGDC and the proposed multidimensional development is that the data collection to be ingested would ideally be ingested in batches, as opposed to the current situation where individual datasets can be ingested on an ad-hoc basis. The design incorporates versioning which will permit rollback of ingestions (subject to storage space limitations). Batch-wise ingestion will actually lead to better practice around the management of dynamic collections in that the state of the collection is known explicitly at any point in time, compared to the current situation with the current Landsat data holdings in the AGDC where data is ingested in a relatively uncontrolled manner.

5.1.4 Costs and Benefits

5.1.5 System context

5.1.6 Static View

5.1.7 Data Views

For interoperability with external systems, it is important for the data model to map explicitly to the Observations & Measurements (O&M) standard (ISO-19156).

The Earth Observation Profile is a particular specialisation of the general O&M schema for a single domain, and some of the main features are shown below in figure 6.

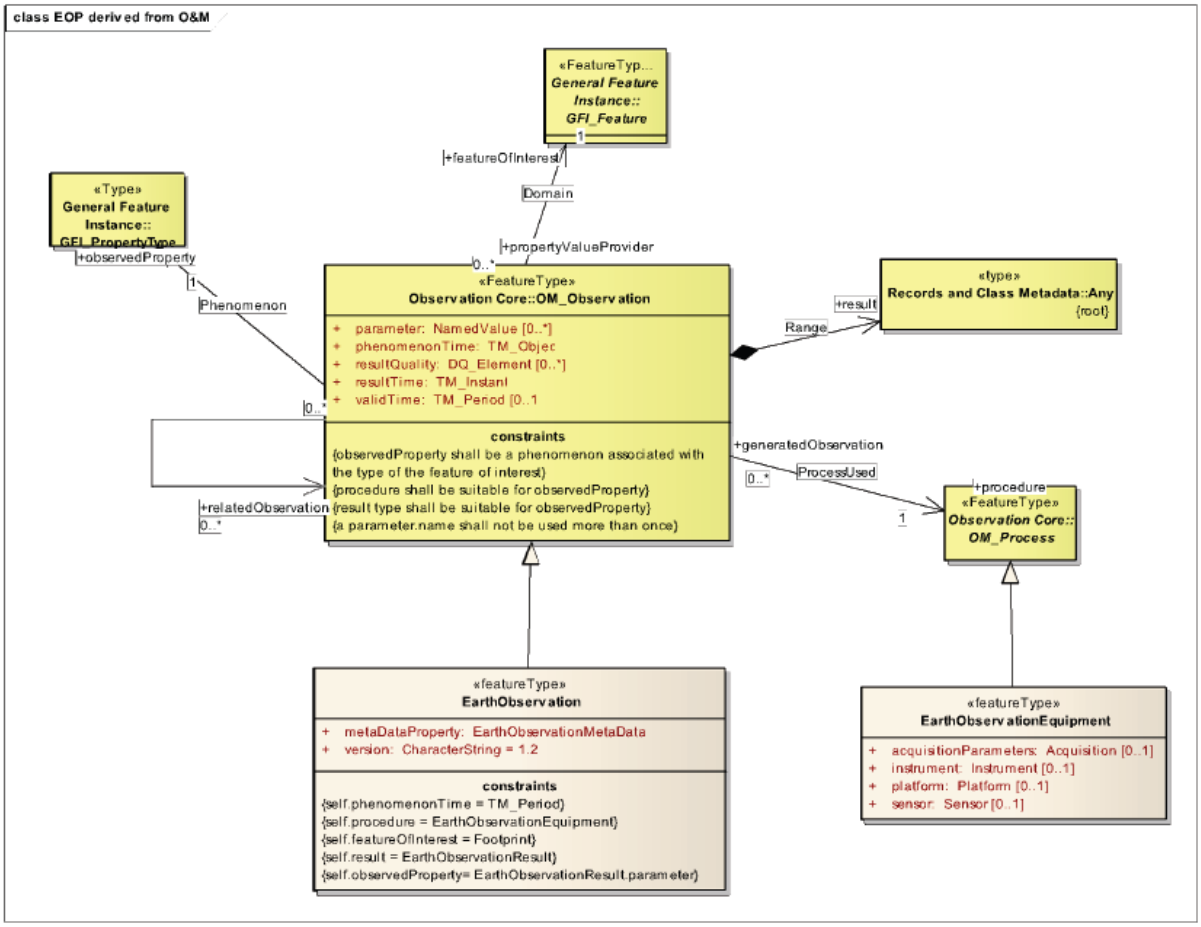


Figure 7 - Observations & Measurements (O&M) Earth Observation Profile (EOP) (OGC, 2012)

One approach to developing a conceptual schema for the AGDC would be to implement the core O&M schema with extensions for each domain. The disadvantage of such an approach would be an inability to manage metadata in a generic way across domains, since every set of domain-specific tables would require corresponding domain-specific queries. Semantically-rich cross-domain querying of such a database would effectively require custom extensions for every permutation of domain pairs. An alternative approach is to work at a level of generalisation which transcends specific domains in order to provide a uniform interface, but this is not achievable in financial year 2014-2015 due to resource constraints.

As the conceptual schema is refined, a physical (relational database) demonstration schema is being developed in parallel. The UML diagrams reflecting the current state of this physical schema are shown in Appendix D (Section 11).

Figure 7 below shows some of the major entities involved in the management of data within the AGDC. Note that the original AGDC data model can be represented within the proposed model.

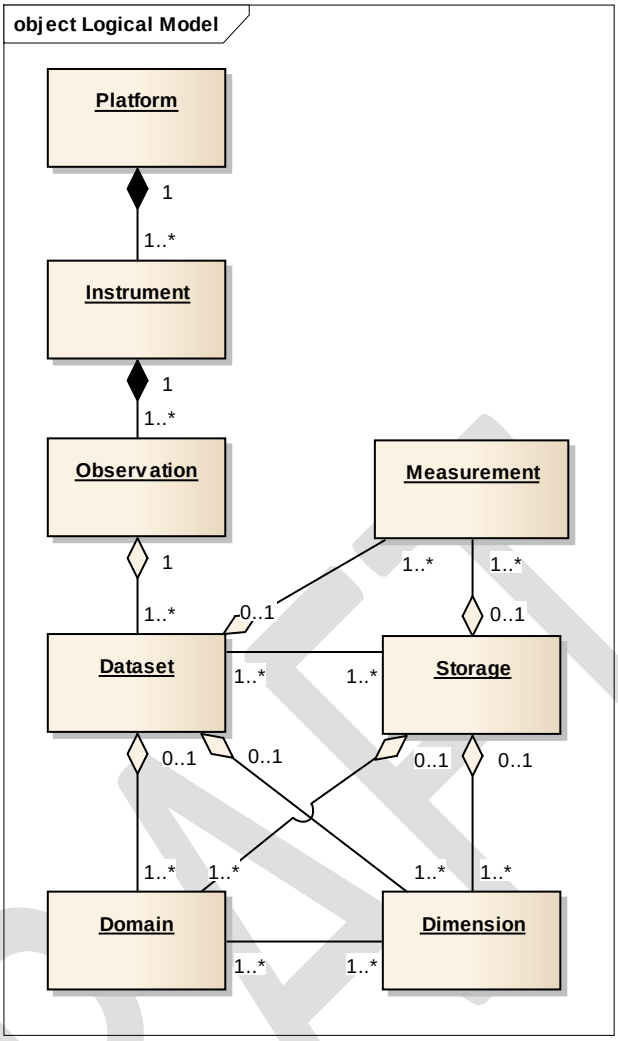


Figure 8 – Platform, Instrument, Observation, Dataset & Measurement

The entities in the proposed data model shown in figure 7 can be mapped to standard O&M entities as shown in figure 8 below. Note that the ***storage*** and ***storage_type*** entities manage the internal GDF storage units and have a many-to-may relationship with the ***dataset*** entity representing source datasets. Note that the meta-type entities (e.g. ***dataset_type***, ***measurement_metatype***, ***storage_type***, etc.) have been omitted for clarity, but these would define which dimensions and domains need to be defined for each instance of dataset and storage.

Table 1 – Conceptual mappings between GDF and O&M Entities

GDF	O&M EOP	Example Entities
Platform	Process - Platform (SensorML)	Satellite
Instrument	Process - Instrument (SensorML)	Sensor
Observation	Observation	Satellite Image Acquisition
Dataset	Sampling Feature	Landsat ARG25 Scene, MODIS MOD09 Swath
Measurement	Result	Spectral Band (reflectance value)

Table 2 – Conceptual Mappings between Entities in GDF and AGDCv1 schema

GDF Entity	AGDCv1 Entity	Example Instance
platform	satellite	Landsat 8
instrument	sensor	OLI-TIRS
observation	acquisition	LS8, WRS=(90,84), Date = 2014-04-01
dataset	dataset	ARG25 for LS8, WRS=(140,-35), Date = 2014-04-01
dataset_type	processing_level	ARG25
measurement_type	band	Red Reflectance Band
storage	Tile	Long=148 - 149, Lat=-36 - -35, Date=2014-01-01 - 2014-12-01
storage_type	tile_type	<geometry and indexing of storage units>
storage_type_measurement_type	band_source	<configuration of data within storage units>

5.1.7.1 Generalised dimensionality with associated reference systems

Each dimension (e.g. Longitude, Latitude, Height/Depth, Time or Spectral) will be associated with one or more domains (e.g. Spatial-XY, Spatial-Z, Temporal, Spectral). For each specific storage_type, each of its domains will have an associated reference system (including details such as measurement unit) through the ***storage_type_dimension*** table.

Note that the new model permits the same dimension to be referenced by multiple domains: e.g. the z (height/depth) dimension could possibly be referenced in a spatial-z domain as well as a spatial-xyz domain. The cardinality of the proposed data model implements a many-to-many relationship between dimensions and domains in order to cater for this more general model.

The proposed model implements three possible indexing types for each array axis:

1. Regular (e.g. latitude or longitude, which will have fixed increments defined by the storage dimensional parameters)
2. Irregular (e.g. time of observation, which has ordered but arbitrary indexing values with irregular increments)
3. Fixed (e.g. spectral band number, which will have the same arbitrary indexing values across all storages of the same type)

In the case of the netCDF file format, internal indexing mechanisms such as those used in the Climate & Forecasting (CF) convention can be used to make the storage units self-describing. The CF convention uses a separate internal indexing array variable to fully describe each array axis.

5.1.7.2 Multi-variate measurements

Observations producing multiple measurements (such as multispectral and hyperspectral EO data) can be managed by storing the measurements either as separate, multidimensional variables in a storage unit or, if they are of the same datatype and scale, as values across another array dimension. The proposed data model can accommodate both approaches.

Figure 11 shows the association between ***measurement*** and ***storage_type***. The measurement entity defines the variables to be used within the storage units, each of which will have the same dimensionality as managed by the ***storage_type***.

5.1.7.3 Management of multidimensional array storage units

A ***dataset*** entity represents a source dataset ingested into the AGDC, and, unlike the original system, this would have a many-many relationship with the ***storage*** storage units due to the temporal aggregation of timeslices sourced from different datasets into the same ***storage*** instance. This approach is an extension of the original regular spatial partitioning of the data to regular spatio-temporal partitioning. Higher dimensionality than 3D could also be regularly partitioned as required.

One consequence of the temporal aggregation of data is that insertion of new data will require re-indexing of existing data storage units. Although it is possible to ingest individual datasets, its likely computational cost would mean that this ad-hoc approach would not be the preferred method for ingesting data. Instead, it would be better to ingest (or update) an entire collection (or a temporal subset defined by the boundaries defined by the chosen ***storage_type***) in a batch-wise operation from a larger number of datasets.

5.1.8 Dynamic View

5.1.8.1 Gridded Data Ingestion

The draft high-level procedure for ingesting gridded data into the Generalised Data Framework (GDF) is shown in the following flowcharts.

Note that the term "HyperTile" has been used instead of "tile", since the general pattern is that data from datasets with particular explicit dimensionality (e.g. xyλ for EO, or xyz for geophysics) will be assembled into collections with higher dimensionality (e.g. xyλt for EO, or xyzv for geophysics where v is model iteration). This means that the subsets of the source datasets will

not necessarily be 2D tiles. HyperTiles are simply regular subsets of their source datasets and, as such, will have the same dimensionality as their source datasets.

During an ingestion operation, temporary Storage files (i.e. netCDF/HDF files of the same dimensionality as the storage units) will be created matching the spatio-temporal extents of the final Storage files to be created, and the new HyperTiles would be stacked in them as they are created (indexed in the database). Any unchanged data from any superseded Storage already in the cube will be merged with the new Hypertiles from the temporary Storage files in the final stage of data copying.

5.1.8.1.1 Notes on draft flowcharts below (remove this section after editing diagrams)

Note that the temporary storages should be deleted after the data has been copied from them. There should be another activity block in there to explicitly represent that operation, since each set of temporary hypertiles in held in the temporary storages (and their corresponding records) serve absolutely no purpose once their corresponding new storages have been created and populated.

The sequence of the actual status changes needs to be adjusted: i.e. the old data should not be flagged as superseded until the new data is fully assembled and ready to flag as active. This will be represented by another (duplicated) decision in the flowchart, and it will be adjusted soon. Ideally, any changes brought about by the ingestion should not be visible until all data transformation and copying is complete.

The obsolete storages (and corresponding dataset records) are retained mainly to allow rollback, but they could be archived on tape to provide complete historical coverage if required. The draft flowchart does not show when the superseded storages should be deleted, but they can be removed as soon as a given ingestion has been validated.

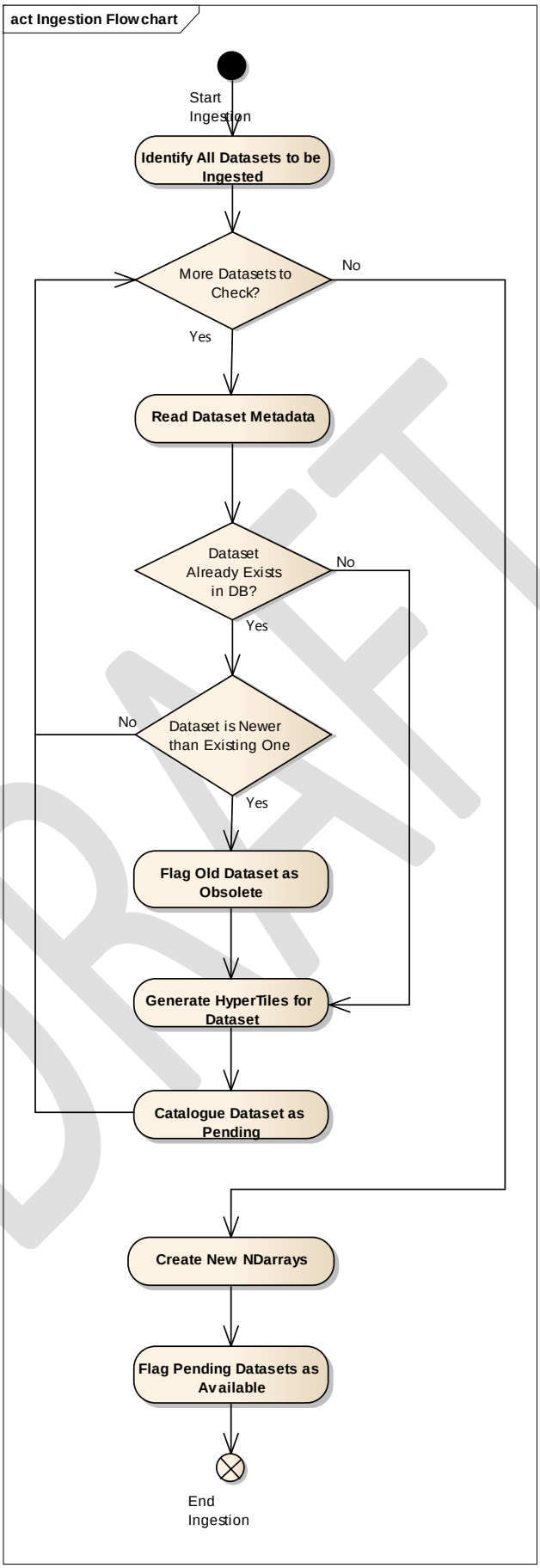


Figure 9 – Top-Level workflow for gridded data ingestion

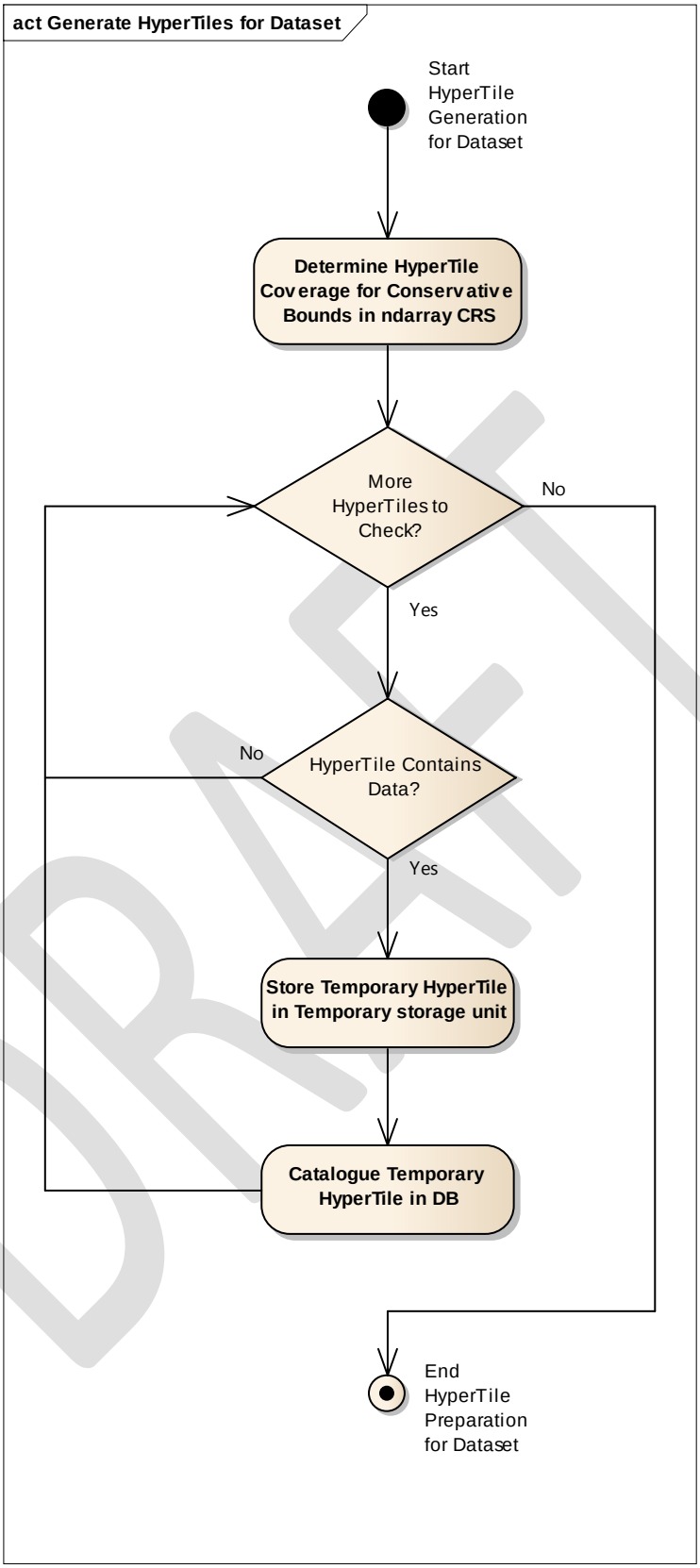


Figure 10 – Hypertile generation for gridded data ingestion

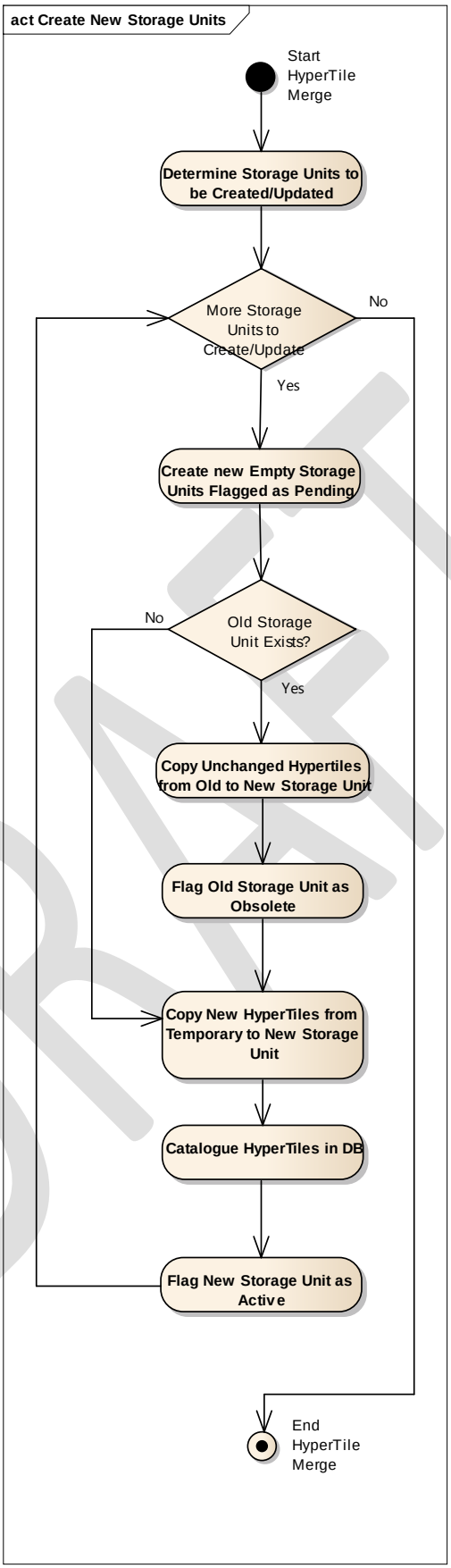


Figure 11 – Storage creation for gridded data ingestion

5.1.9 Security

The GDF is designed to work with a federated system of databases, where only the custodian of each collection would have write access to both the database and files. This model is the simplest, most flexible and most reliable, and could easily be extended to provide finer granularity of access control.

5.1.10 Software Technologies

The initial implementation of the GDF is in Python, but interfaces for other languages could be developed later.

5.1.11 Infrastructure Deployment

The operational system will be implemented on Raijin at the NCI, with databases on virtual PostgreSQL servers in the NCI partner cloud. Should the database become a performance bottleneck, then new database servers could be spun up as required with replication and load balancing.

Note that there is no dependency on specific aspects of the NCI infrastructure or software environment – for example, a complete, self-contained (but very small scale) implementation of the AGDC has been set up on a portable VM for training purposes.

5.1.11.1 Specification of all computers and the deployment all software components onto those computers (virtual and physical)

5.1.11.2 Storage (physical and logical) – Database and files system storage

5.1.11.3 Network components at an appropriate level of detail to enable network staff to configure the network to support application operations

5.1.11.4 Third party components (including version numbers) – the diagram should show where they are hosted and how they relate to other

5.1.11.5 Protocols used for communication between GA components and externally hosted components and middleware (SQL proxies)

5.1.12 Development Environment

5.1.13 Operational View

5.1.14 Support

5.1.15 Description of Availability, Performance, Scalability and Extensibility mechanisms and how the system will meet the business requirements in these areas

6 Recommendations

7 Implementation Approach

It is proposed that the implementation be broken up into the following tasks, each shown with its status as at 11/05/2015:

- Determine configuration parameters required for multi-variate nD storage format (based on netCDF-CF data model) – completed
- Design parameterised low-level functions for creating, populating and reading storage files – underway
- Develop a basic optimisation engine to determine suitable parameter sets for current data collections – pending
- Complete design of new schema from proposed data model and implement database – completed (subject to minor adjustment/enhancement)
- Develop new generic ingestion mechanism to ingest datasets directly into new storage format and database schema – high-level design completed, implementation pending
- Remap all existing metadata in the original AGDC database into the new database to test scalability (optional) – completed
- Create 3D storage structures for testing from existing data – completed
- Design and develop declarative API for data access – completed initial version

8 Appendix A: Original AGDCv1 Tile Contents for Landsat Data

The contents of tiles are defined using the band_source table (see section 9.4).

Below are the contents of the original 2D geoTIFF tiles for the current Landsat data holdings

Level 1 Topographic (ORTHO)
1. LS5-B60 – Thermal Infrared
or
1. LS7-B61 – Thermal Infrared Low Gain
2. LS7-B62 – Thermal Infrared High Gain
or
1. LS8-B10 – Thermal Infrared 1
2. LS8-B11 – Thermal Infrared 2
(Byte datatype)

ARG-25 (NBAR)
1. LS5/7-B10 – Visible Blue
2. LS5/7-B20 – Visible Green
3. LS5/7-B30 – Visible Red
4. LS5/7-B40 – Near Infrared
5. LS5/7-B50 – Middle Infrared 1
6. LS5/7-B70 – Middle Infrared 2
or
1. LS8-B1 – Coastal Aerosol
2. LS8-B2 – Visible Blue
3. LS8-B3 – Visible Green
4. LS8-B4 – Visible Red
5. LS8-B5 – Near Infrared
6. LS8-B6 – Middle Infrared 1
7. LS8-B7 – Middle Infrared 2
(Int16 Datatype)

Pixel Quality (PQA)*
1. PQ – Bit-array of PQ tests
(UInt16 Datatype)

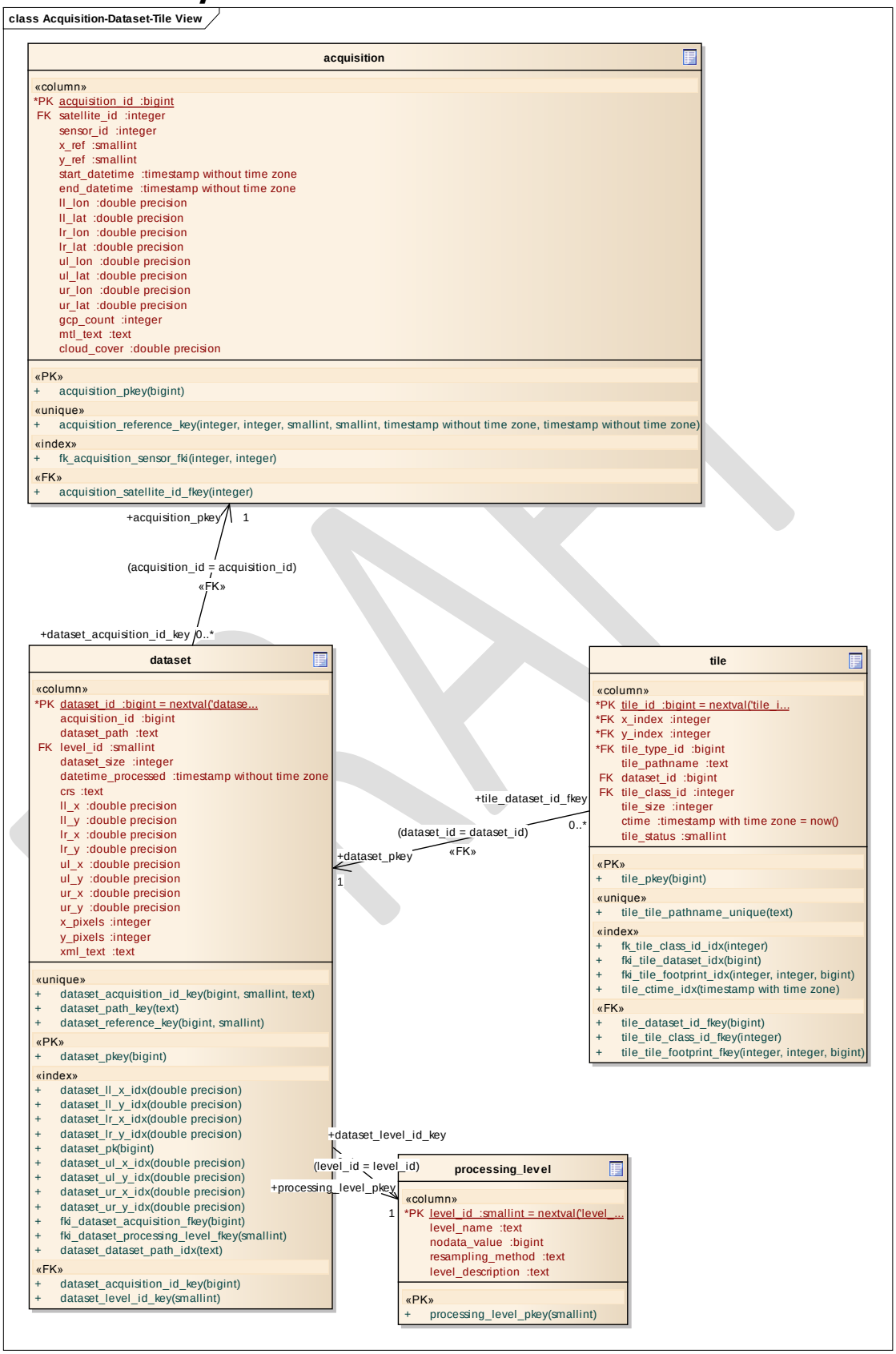
Fractional Cover (FC)**
1. Photosynthetic Veg. (PV)
2. Non-Photosynthetic Veg. (NPV)
3. Bare Soil (BS)
4. Un-mixing Error (UE)
(Float32 Datatype)

Digital Surface Model (DSM)***
1. Elevation
2. Slope
3. Aspect
(Float32 Datatype)

* PQA Geoscience Australia
** QDERM
*** Single, static source dataset, i.e. not time varying. Resampled from 1" DSM. Licensed for Government Use Only

9 Appendix B: AGDCv1 Schema

9.1 AGDCv1 Acquisition-Dataset-Tile Entities

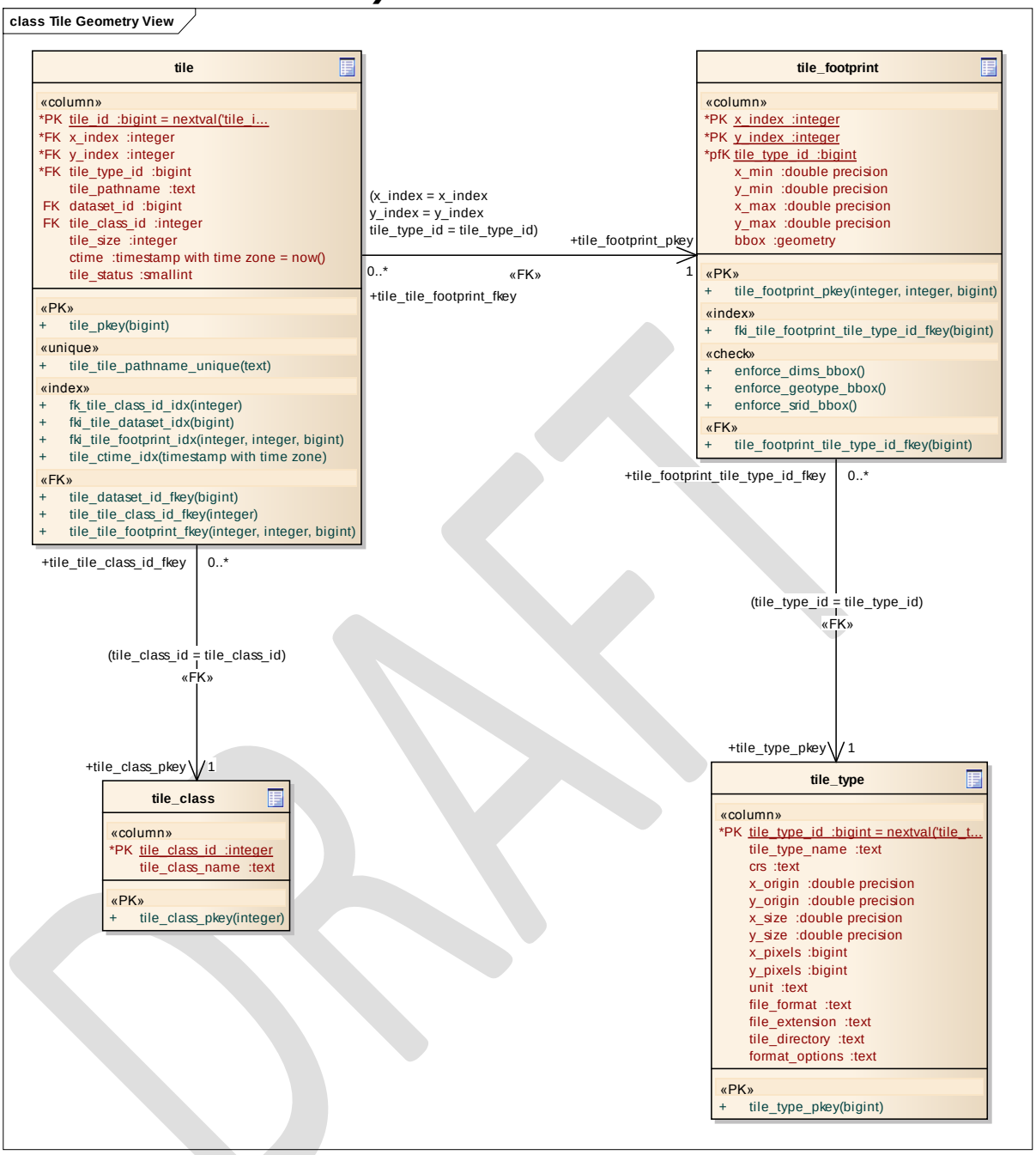


The original AGDC schema manages the following entities:

- **Acquisition** – table managing source acquisitions independent of processing level. For example, this identifies a given Landsat scene or MODIS swath from which different products are generated.
- **Dataset** – table managing source datasets with specific processing level. This record maintains metadata from the relevant source dataset
- **Tile** – table managing 2D, multi-band storage unit (see section 8.2 for further details)
- **Processing_level** – lookup table defining processing level: e.g. L1T, NBAR, PQ or FC for Landsat, MOD09 for MODIS, etc.

Note that there is also an experimental table **acquisition_footprint** which is partially populated but not currently used.

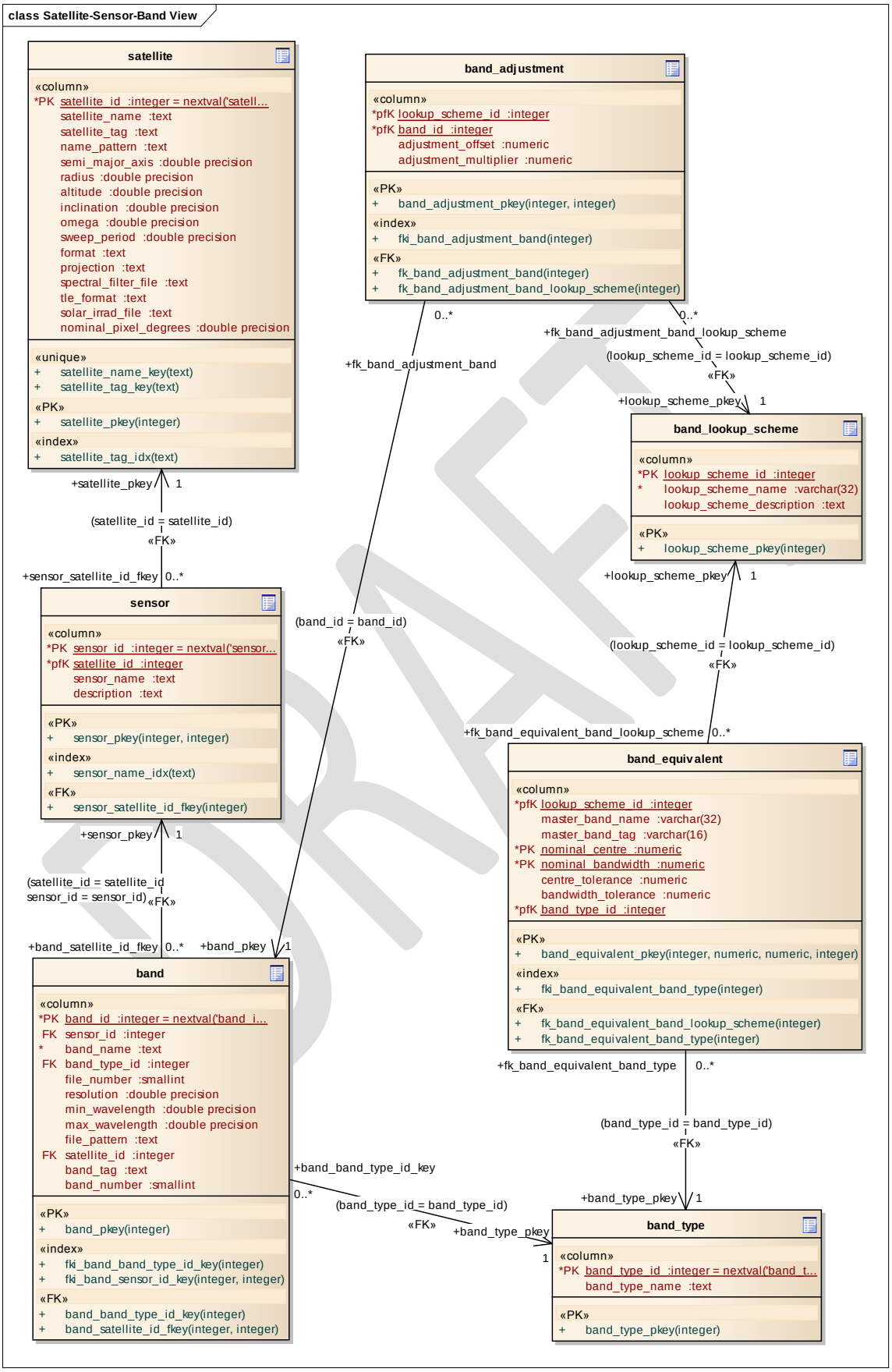
9.2 AGDCv1 Tile Geometry Entities



The original AGDC schema manages the following entities:

- **Tile** – table managing 2D, multi-band storage units.
- **Tile_footprint** – table managing geometric information for each "cell" with a particular tile_type_id, x_index and y_index. This table includes a PostGIS geometry field for spatial queries.
- **Tile_class** – Lookup table managing classes of tiles: e.g. non-overlapped tile, tile with overlap, two-tile mosaic, etc. The tile_class is used to distinguish between source tiles from single datasets and the mosaics created when two (or possibly more) datasets overlap.
- **Tile_type** – table defining geometric parameters for a given tiling scheme: e.g. Landsat uses tile_type 1 to define the 1 x 1degree, 4000 x 4000pixel EPSG:4326 tiling scheme. MODIS uses a different tile_type record to define the 10 x 10degree, 2000 x 2000pixel EPSG:4326 tiling scheme. Note that the AGDC can theoretically work with any projection system supported by GDAL/PROJ4.

9.3 AGDCv1 Satellite-Sensor-Band Entities



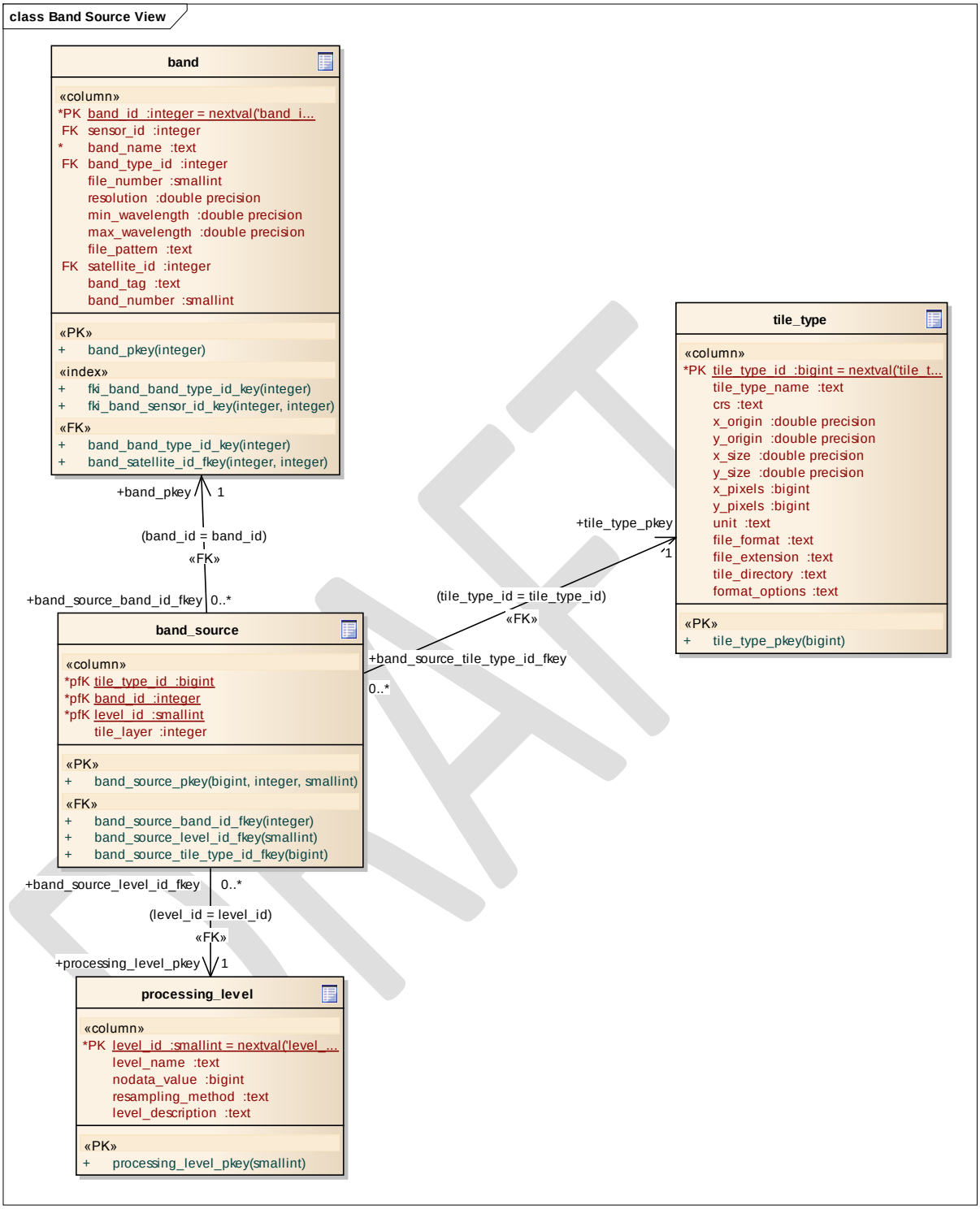
The original AGDC schema manages the following entities:

- **Satellite** – table managing satellite metadata. Specific records include those for Landsat 5, 7 & 8, and MODIS Aqua and Terra.
- **Sensor** – table managing sensor metadata. Specific records include TM for Landsat 5, ETM for Landsat 7 and OLI-TIRS for Landsat 8. Sensors are aggregated by satellite.
- **Band** – table managing spectral bands for each sensor. Bands can be aggregated by sensor but note that sensor-independent derived values such as elevation and related values or Fractional Cover fractions are not linked to a satellite or sensor.
- **Band_type** – lookup table defining type of band: e.g. Reflectance, Thermal, etc. Note that non-spectral value types such as elevation, slope and aspect also have records in this table

In addition to the above tables, the following tables define equivalences between bands for the purposes of matching spectrally similar bands in a sensor agnostic manner. **An expanded view of the current contents of these tables can be seen in appendix C (Section 10).**

- **Band_equivalent** – table defining sets of centre wavelength, bandwidth and tolerances for each of these under different matching schemes for the purposes of grouping spectrally similar bands. Nearly-equivalent bands are grouped and identified by a master_band_tag mnemonic
- **Band_lookup_scheme** – lookup table defining band lookup scheme names and descriptions. Current entries include "LANDSAT-LS5/7" which adjusts Landsat 8 bands to simulate their nearly-equivalent Landsat 5/7 band spectral characteristics, "LANDSAT-LS8" which adjusts Landsat 5 & 7 bands to simulate their nearly-equivalent Landsat 8 spectral characteristics, and "LANDSAT-UNADJUSTED" which matches equivalent bands but performs no adjustments.
- **Band_adjustment** – table containing slope & intercept linear adjustment factors to adjust nearly-equivalent reflectance values.

9.4 AGDCv1 Band Source Entities



The band_source table defines the contents of tiles for each combination of tile type and processing level. Note that bands are already aggregated by sensor, and sensors are aggregated by satellite.

The band_source table is queried by the ingestor to determine what tiles to create and which bands to include in each tile.

10 Appendix C: AGDCv1 Band Lookup Schemes

lookup scheme name	satellite	processing level	mnemonic	tile layer	intercept adjustment	slope adjustment
LANDSAT-LS5/7	LS5	NBAR	B	1	0	1
LANDSAT-LS5/7	LS5	NBAR	G	2	0	1
LANDSAT-LS5/7	LS5	NBAR	R	3	0	1
LANDSAT-LS5/7	LS5	NBAR	NIR	4	0	1
LANDSAT-LS5/7	LS5	NBAR	SWIR1	5	0	1
LANDSAT-LS5/7	LS5	NBAR	SWIR2	6	0	1
LANDSAT-LS5/7	LS5	ORTHO	TIR	1	0	1
LANDSAT-LS5/7	LS7	NBAR	B	1	0	1
LANDSAT-LS5/7	LS7	NBAR	G	2	0	1
LANDSAT-LS5/7	LS7	NBAR	R	3	0	1
LANDSAT-LS5/7	LS7	NBAR	NIR	4	0	1
LANDSAT-LS5/7	LS7	NBAR	SWIR1	5	0	1
LANDSAT-LS5/7	LS7	NBAR	SWIR2	6	0	1
LANDSAT-LS5/7	LS7	ORTHO	TIR-LG	1	0	1
LANDSAT-LS5/7	LS7	ORTHO	TIR-HG	2	0	1
LANDSAT-LS5/7	LS8	NBAR	NB	1	0	1
LANDSAT-LS5/7	LS8	NBAR	B	2	0	1.02
LANDSAT-LS5/7	LS8	NBAR	G	3	0	0.98
LANDSAT-LS5/7	LS8	NBAR	R	4	0	0.98
LANDSAT-LS5/7	LS8	NBAR	NIR	5	0	1.01
LANDSAT-LS5/7	LS8	NBAR	SWIR1	6	0	1
LANDSAT-LS5/7	LS8	NBAR	SWIR2	7	0	0.99
LANDSAT-LS5/7	LS8	ORTHO	TIR1	1	0	1
LANDSAT-LS5/7	LS8	ORTHO	TIR2	2	0	1
LANDSAT-LS8	LS5	NBAR	B	1	0	0.98

LANDSAT-LS8	LS5	NBAR	G	2	0	1.02
LANDSAT-LS8	LS5	NBAR	R	3	0	1.02
LANDSAT-LS8	LS5	NBAR	NIR	4	0	0.99
LANDSAT-LS8	LS5	NBAR	SWIR1	5	0	1
LANDSAT-LS8	LS5	NBAR	SWIR2	6	0	1.01
LANDSAT-LS8	LS5	ORTHO	TIR	1	0	1
LANDSAT-LS8	LS7	NBAR	B	1	0	0.98
LANDSAT-LS8	LS7	NBAR	G	2	0	1.02
LANDSAT-LS8	LS7	NBAR	R	3	0	1.02
LANDSAT-LS8	LS7	NBAR	NIR	4	0	0.99
LANDSAT-LS8	LS7	NBAR	SWIR1	5	0	1
LANDSAT-LS8	LS7	NBAR	SWIR2	6	0	1.01
LANDSAT-LS8	LS7	ORTHO	TIR-LG	1	0	1
LANDSAT-LS8	LS7	ORTHO	TIR-HG	2	0	1
LANDSAT-LS8	LS8	NBAR	NB	1	0	1
LANDSAT-LS8	LS8	NBAR	B	2	0	1
LANDSAT-LS8	LS8	NBAR	G	3	0	1
LANDSAT-LS8	LS8	NBAR	R	4	0	1
LANDSAT-LS8	LS8	NBAR	NIR	5	0	1
LANDSAT-LS8	LS8	NBAR	SWIR1	6	0	1
LANDSAT-LS8	LS8	NBAR	SWIR2	7	0	1
LANDSAT-LS8	LS8	ORTHO	TIR1	1	0	1
LANDSAT-LS8	LS8	ORTHO	TIR2	2	0	1
LANDSAT-UNADJUSTED	LS5	NBAR	B	1	0	1
LANDSAT-UNADJUSTED	LS5	NBAR	G	2	0	1
LANDSAT-UNADJUSTED	LS5	NBAR	R	3	0	1
LANDSAT-UNADJUSTED	LS5	NBAR	NIR	4	0	1
LANDSAT-UNADJUSTED	LS5	NBAR	SWIR1	5	0	1

LANDSAT-UNADJUSTED	LS5	NBAR	SWIR2	6	0	1
LANDSAT-UNADJUSTED	LS5	ORTHO	TIR	1	0	1
LANDSAT-UNADJUSTED	LS7	NBAR	B	1	0	1
LANDSAT-UNADJUSTED	LS7	NBAR	G	2	0	1
LANDSAT-UNADJUSTED	LS7	NBAR	R	3	0	1
LANDSAT-UNADJUSTED	LS7	NBAR	NIR	4	0	1
LANDSAT-UNADJUSTED	LS7	NBAR	SWIR1	5	0	1
LANDSAT-UNADJUSTED	LS7	NBAR	SWIR2	6	0	1
LANDSAT-UNADJUSTED	LS7	ORTHO	TIR-LG	1	0	1
LANDSAT-UNADJUSTED	LS7	ORTHO	TIR-HG	2	0	1
LANDSAT-UNADJUSTED	LS8	NBAR	NB	1	0	1
LANDSAT-UNADJUSTED	LS8	NBAR	B	2	0	1
LANDSAT-UNADJUSTED	LS8	NBAR	G	3	0	1
LANDSAT-UNADJUSTED	LS8	NBAR	R	4	0	1
LANDSAT-UNADJUSTED	LS8	NBAR	NIR	5	0	1
LANDSAT-UNADJUSTED	LS8	NBAR	SWIR1	6	0	1
LANDSAT-UNADJUSTED	LS8	NBAR	SWIR2	7	0	1
LANDSAT-UNADJUSTED	LS8	ORTHO	TIR1	1	0	1
LANDSAT-UNADJUSTED	LS8	ORTHO	TIR2	2	0	1

The above table is a summary of the current values in band equivalence schemes drawn from the following query:

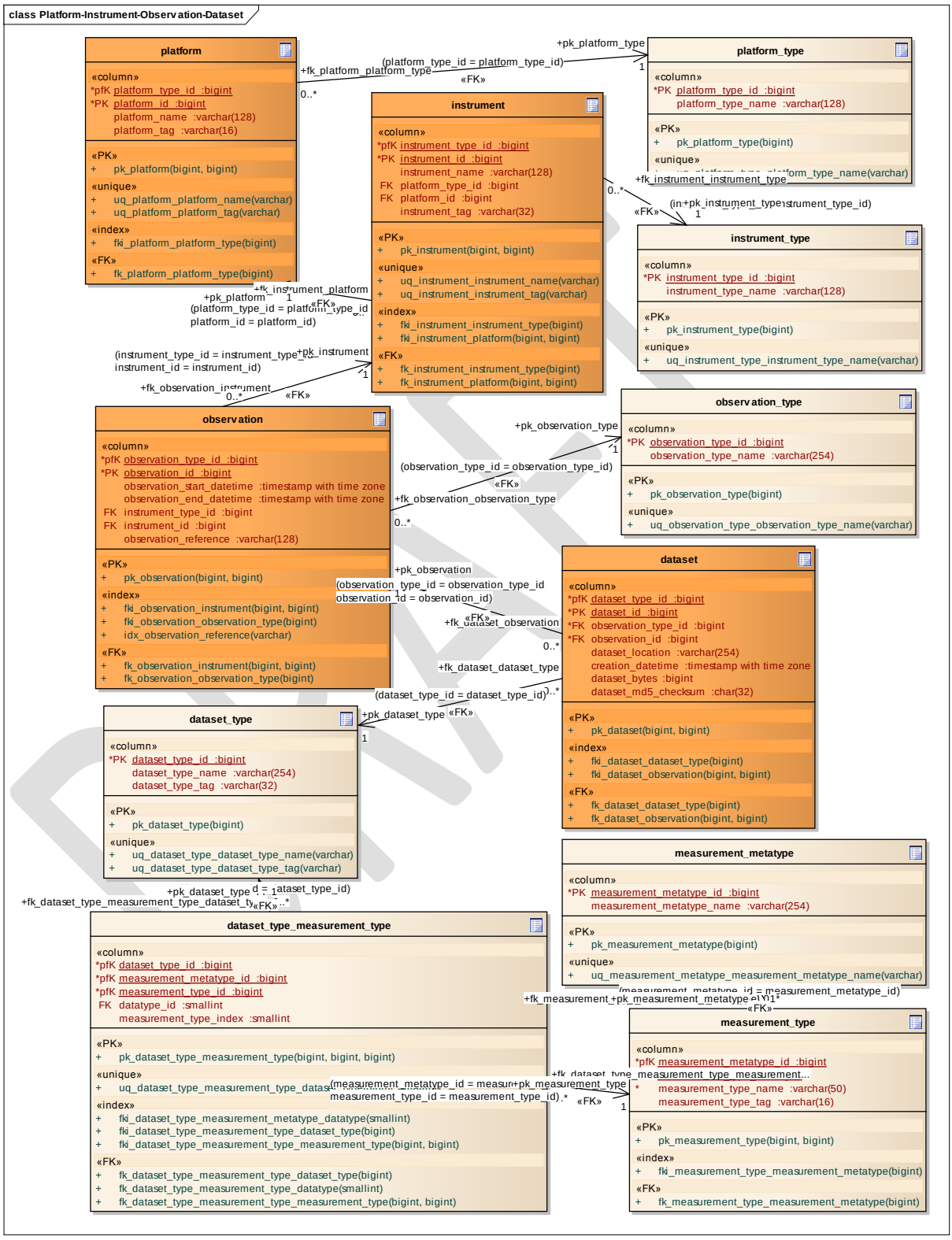
```
SELECT distinct
    band_lookup_scheme.lookup_scheme_name,
    coalesce(satellite.satellite_tag, 'DERIVED') as satellite_tag,
    processing_level.level_name,
    band_equivalent.master_band_tag,
    band_source.tile_layer,
    COALESCE(band_adjustment.adjustment_offset, 0.0)::float AS adjustment_offset,
    COALESCE(band_adjustment.adjustment_multiplier, 1.0)::float AS adjustment_multiplier
   FROM band
   JOIN band_type using(band_type_id)
   JOIN band_source using (band_id)
   JOIN processing_level using(level_id)
   JOIN band_equivalent ON abs((band.max_wavelength::numeric +
band.min_wavelength::numeric) / 2.0 - band_equivalent.nominal_centre) <=
band_equivalent.centre_tolerance AND abs(band.max_wavelength::numeric -
```

```
band.min_wavelength::numeric - band_equivalent.nominal_bandwidth) <=
band_equivalent.bandwidth_tolerance and band_equivalent.band_type_id = band.band_type_id
  JOIN band_lookup_scheme USING (lookup_scheme_id)
  LEFT JOIN band_adjustment USING (lookup_scheme_id, band_id)
  LEFT JOIN sensor using(satellite_id, sensor_id)
  LEFT JOIN satellite using(satellite_id)
  where sensor_name not in ('OLI','TIRS') -- Exclude duplicate LS8 sensors for clarity
  ORDER BY 1,2,3,5
```

The BandLookup Python class provides a simple means for users to reference bands by mnemonic rather than hard-coded band numbers.

11 Appendix D: GDF Schema

11.1 Alignment with Observation & Measurement



11.2 Generalised dimensionality with associated reference systems

The dimensional configuration of the ***storage*** structures is governed by the associated **storage_type**, ***storage_type_dimension*** and ***storage_type_domain*** tables. The ***storage_dimension*** table manages the instance-specific indexing value for each dimension.

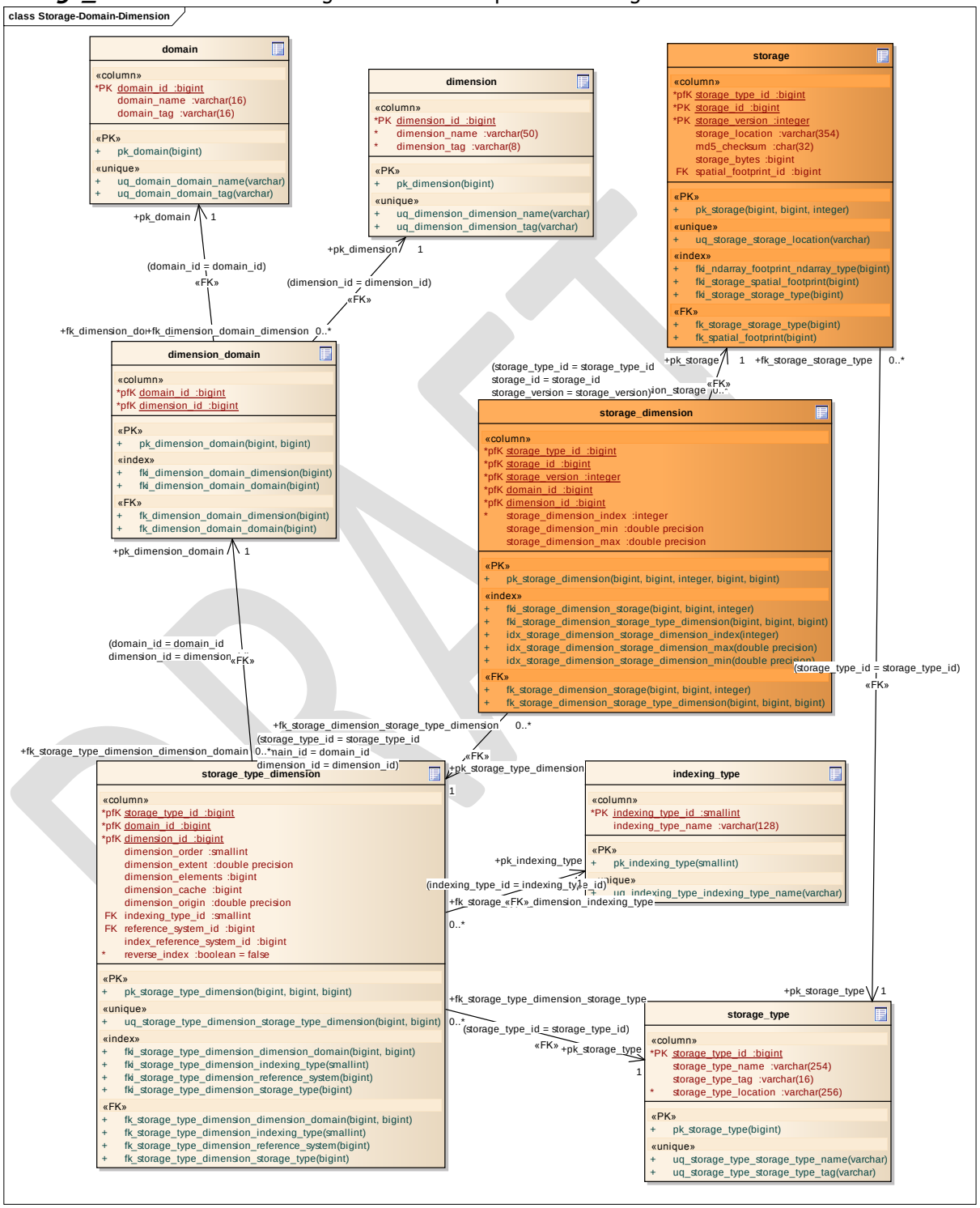


Figure 12 - Generalised Dimensionality with associated reference systems

11.3 Multi-variate measurements

Observations producing multiple measurements (such as multispectral and hyperspectral EO data) can be managed by storing the measurements either as separate, multidimensional variables in a storage unit or, if they are of the same datatype and scale, as values across another array dimension.

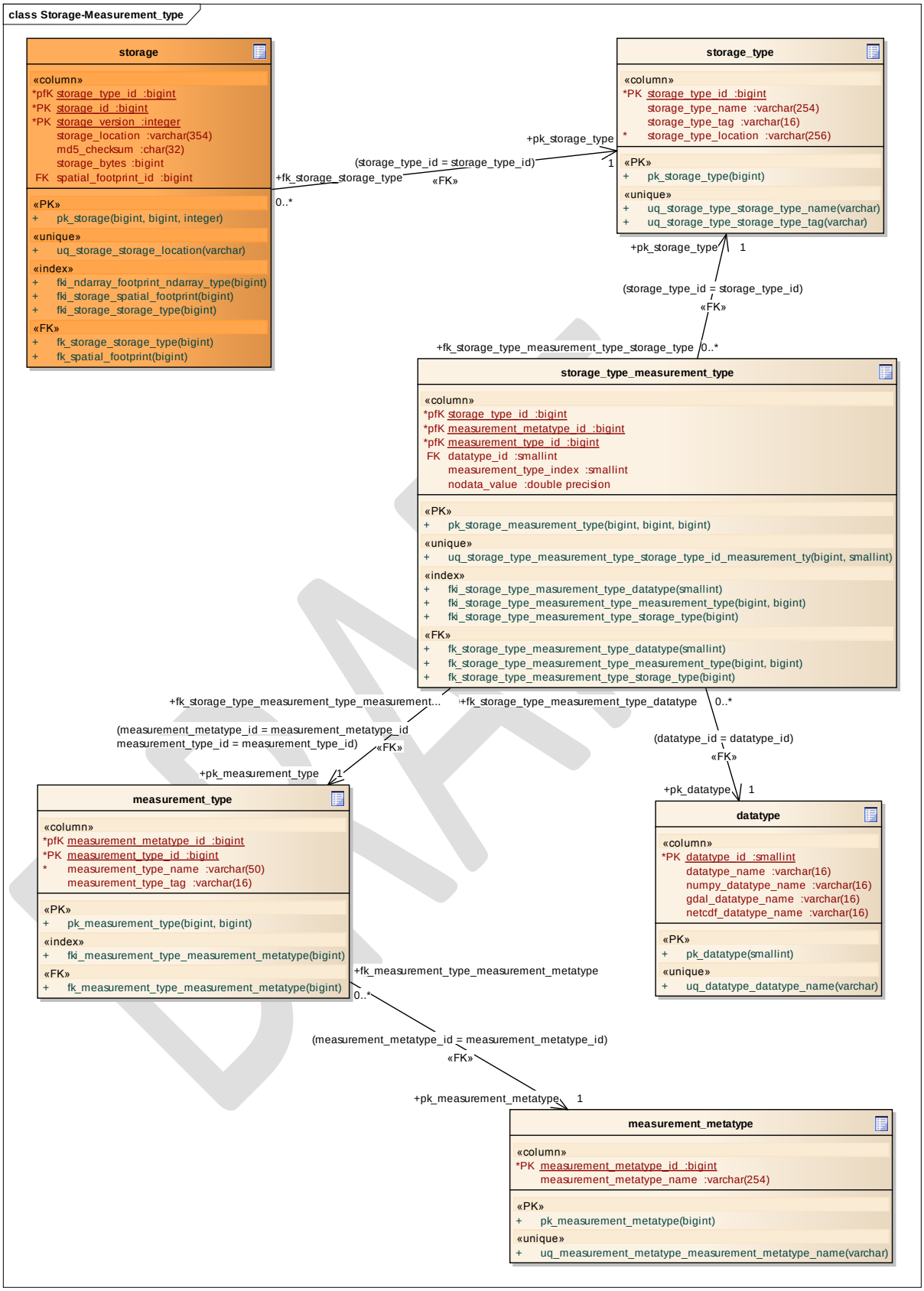


Figure 13 - Multi-variate measurements

11.4 Management of multidimensional array storage units

A ***dataset*** entity represents a source dataset ingested into the AGDC, and, unlike the original system, this would have a many-many relationship with the ***storage*** storage units due to the temporal aggregation of timeslices sourced from different datasets into the same ***storage*** instance. Note that the many:many relationship between ***storage*** and ***dataset*** is managed by the ***storage_dataset*** table which would reflect the contents of the ***storage*** storage structures. This aspect of the schema is shown below:

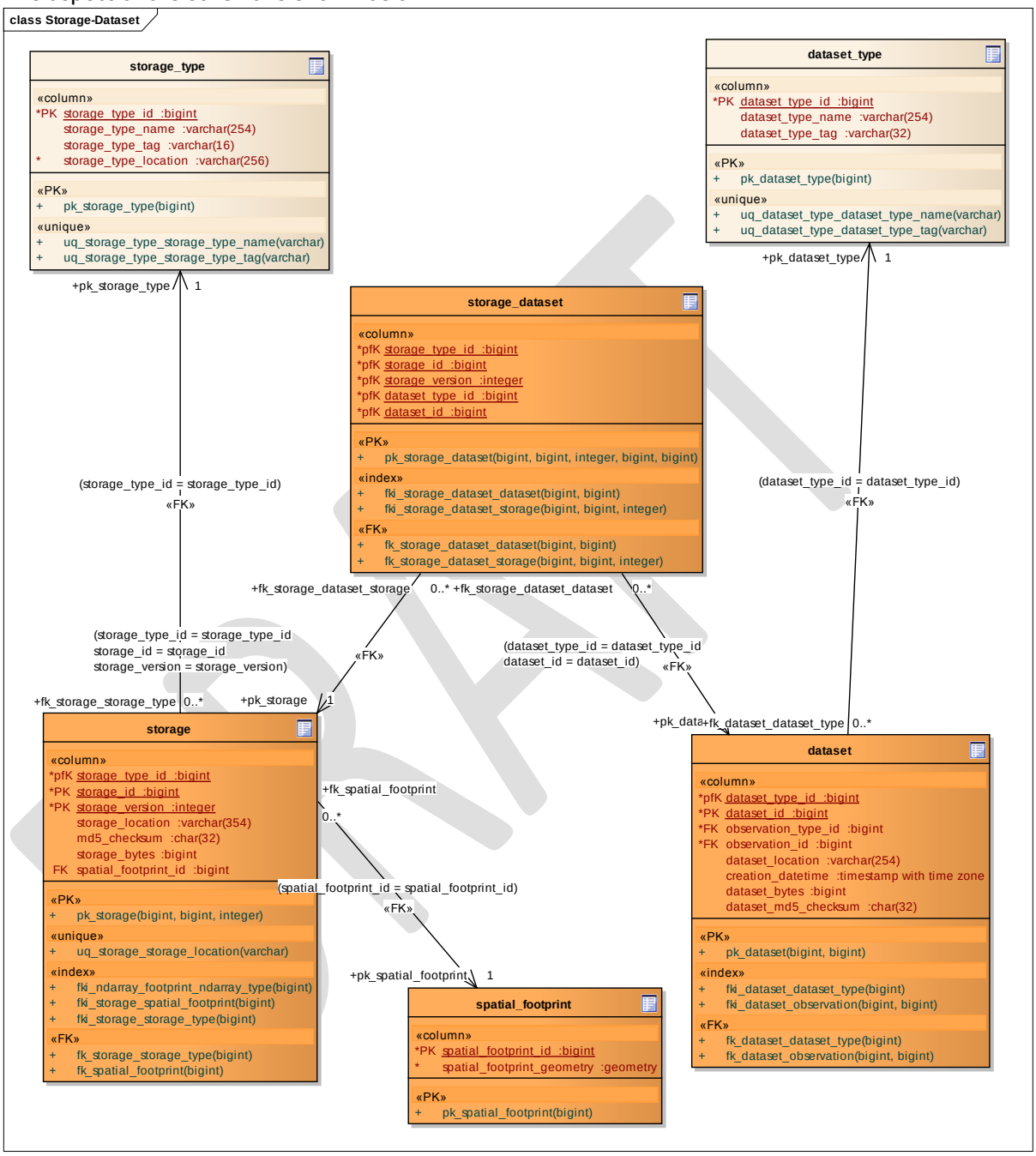


Figure 14 - Management of multidimensional array storage units

End of Document