Rubin Observatory

Schema Management in DM

William O'Mullane and Colin Slater

2020-09-11

1 Introduction

The Science Data Model (SDM) specification is a machine-readable specification for the physical data model for the publicly released science data products of the Rubin Observatory Project. Meanwhile the DPDD represents an "idealized" data model that cannot directly be used as a recipe for the construction of a physical database schema and is not a full, precise definition of a data model. It should be treated as a requirements document for the data model and schema rather than itself defining them. However we will not cover the evolution of the DPDD in this note.

2 SDM requirements

The SDM must contain sufficient information for a physical SQL schema definition to be derived from it, given a choice of SQL flavor (e.g., MariaDB, PostgreSQL).

The SDM must contain information that itemizes how it satisfies the DPDD requirements for the content of the data model. For example, each SDM element that realizes a data item from the DPDD might contain a field that references the appropriate DPDD Identifier.

Each element of the SDM must be described by a unique identifier ("SDM Identifier") that can be used programmatically in applications that consume the SDM YAML definition. We expect that the "leaf nodes" in the name space of these identifiers will correspond directly to column names in generated database tables; it seems unnecessary to have yet another layer of indirection at this level. Higher levels in the name space may not correspond exactly to database and/or table names, however; this has yet to be determined precisely.

Software support must be provided for verifying that the SDM provides coverage for all the data items defined in the DPDD. This should ultimately be subject to verification as part of a CI process. In order to facilitate the introduction of the SDM language and software into DM,

a transitional period should be supported during which a partially complete SDM can be used without triggering constant CI failures.

The design of the SDM and its specification language should address the need to map the physical data model that derives from it to the catalog.schema.table name space of the ADQL 2.0 and TAP standards. (Bear in mind that the way the term ”catalog” is used in this context does not correspond to the intuitive sense of ”astronomical catalog”.)

In addition to being able to be used to construct a database schema, the SDM specification must also include the information required to provide IVOA-oriented table and column metadata in query responses. The system must support:

- the assignment of UCDs from the UCD1+ standard to all column-like fields in the SDM;
- the assignment of IVOA ”utypes”, where they are useful and taken from either a external standard vocabulary or an LSST-provided vocabulary, to column-like fields as well as, potentially, to tables; and
- the definition of ”field groups” in the VOTable sense for related data items, such as a quantity and its uncertainty(ies).
- It should also support: mapping of the Science Data Model to externally provided or LSST-defined VO-DML for part or all of its content (at time of writing this appears useful but may be reassessed).

The SDM specification must include a data type definition for each column-like field. This data type is intended to be used to derive several downstream data types involved in the physical instantiation and service of the data:

- SQL database types consistent with the variety of actual database software used in LSST, which will include at least MariaDB and PostgreSQL depending on the progress of PPDB development, and should also include the HyperSQL in-memory database used in the Portal Aspect of the LSP (in Firefly);
- SQL92-based database types for use in the ADQL context;
- VOTable data types;

- text-based data formats for use when data model elements are represented as text, e.g., in the VOTable TABLEDATA format or in CSV;
- Python data types; and
- data types usable in the Parquet and Apache Arrow cross-platform ecosystem, as well as in Apache Avro.

It may be necessary for the SDM specification to allow for an override of the ”natural” mappings between these target types, so this should be kept in mind in the design, but no specific instance of a need for this has yet been documented.

The SDM specification must contain sufficient information to be able to derive the foreign-key relationships between tables. It should also allow:

- the definition of specific columns as required to be indexed (though downstream database implementations may be permitted to add indexes to additional columns for implementation-time performance optimization); and
- the definition of the columns and key relationships required to support the Qserv architecture, e.g., the designation of which ra/decl values in a table are the ”primary” ones to be used for the spatial partitioning of the table.

The SDM specification should be usable to support the mapping of sectors of the Science Data Model to external data models such as ObsCore and CAOM2. In many cases it may be possible for the SDM itself to include data elements that directly correspond to required elements of these data models; in other cases some conversion may be required. Both scenarios should be supportable.

The actual mapping to external data models may be part of the SDM specification itself or it may be deferred to additional specification file(s).

The SDM specification ”technology” may be used not only to define the LSST data model (for a Data Release, or for the instantiation of the Prompt Processing system and database), but also in narrower contexts to define smaller data models for use in science validation and to support the use of the Science Platform tools during commissioning, for instance.

Concrete use cases that must be supported by the SDM specification and associated software:

- generation of executable physical SQL schema, comparable to what is now in the ”cat” repository on Github;
- definition of the alert data model and its associated Apache Avro .avsc schema;
- population of the TAP_SCHEMA tables required by the TAP 1.1 standard with the information needed to describe the LSST data model, including foreign-key relationships;
- generation of the VOTable headers for query results from the LSST TAP service;
- population of the tables or, where appropriate, definition of the views mandated by the external standards LSST supports, e.g., the ”ivoa.ObsCore” table/view required by the ObsTAP portion of the ObsCore standard, or the standard tables of the CAOM2 data model; and
- creation of Parquet files representing the Science Data Model and usable for ingest into the databases.

The SDM-to-DPDD mapping information may be useful as part of the documentation of the Science Data Model that LSST exposes to users, to facilitate their understanding of how the SDM corresponds to the DPDD. To this end, for instance, it may be appropriate to include in the TAP_SCHEMA tables an additional column (something permitted by the standard) that provides, where appropriate, the DPDD Identifier for a data item.

3 SDM Implementation

The SDM information must be available to a variety of different tools, each with slightly different needs. Because of this, it was not sufficient to adopt a format like SQL `CREATE TABLE` statements that were only suited to one particular use, and difficult to parse for all other uses. Instead, the schemas in the `sdm_schemas` repository are in a yaml format defined by the `Felis` tool. The ease of parsing yaml makes it possible for many different tools to all share the same source of schema information, minimizing intermediate stages.

The current uses of the Felis-defined yaml files are:

1. Qserv ingest — The Felis files are used as inputs to the ingest process.
2. TAP_SCHEMA creation — The Felis tool itself is designed to generate SQL statements that populate the schema database used by the TAP standard.
3. LDM-153 generation — The tables in the document generated from the Yaml files.
4. Pipeline data product verification — Continuous integration tests verify that the pipeline outputs comply with the physical schema in `hsc.yaml`.

Most of these uses depend on the YAML schema files without relying on the Felis tool itself. This is generally a consequence of the YAML format being easy to parse by other tools.

3.1 Types of Schemas

DM maintains multiple SDM specification files, which serve different different organizational purposes.

"Specification" schemas — LDM-153. This is a schema that is designed to fulfill the needs of the DPDD's abstract schema, but might not yet be physically realized. This schema is necessary both for sizing purposes, and as a "goal" that the pipelines teams can work to as they build and evolve the pipeline output files. The result of the construction project should be for the pipelines to produce data that fully realize this schema.

Because changes to the "specification" schema potentially have impacts on multiple areas of DM (e.g. storage costs or science impacts), it is change-controlled at the DM-CCB level.

"Concrete" schemas — e.g. HSC reprocessing schemas. These schemas *are* physically realized; they are meant to describe data products that currently exist. These schemas must accurately reflect those data products, regardless of what is specified in the DPDD or LDM-153. They are not subject to change control since there is no project management impact that can be caused by any changes. Inaccuracies may cause different dependent services to break, but this is generally comparable in consequence to any other code breakage.

4 Schema Management

Here we discuss control of the schema. How do we manage changes in a controlled manner which do not break our systems? Can we do that and still remain flexible enough for a large organisation to develop with relative ease?

The DPDD is a project-level change-controlled document it shall remain subject to the RFC and DM CCB procedures for modifying it.

The "specification" schema in LDM-153 should be updated whenever necessary to reflect any changes in DM's planned data products at the end of construction. Evolution of the LDM-153 schema is expected during construction as the pipelines evolve and the measurement outputs are better understood.

LDM-153 is change-controlled by the DM-CCB, and its contents are generated from the baselineSchema.yaml file in the `sdm_schemas` repository. Procedurally, `baselineSchema.yaml` should be treated like any other LaTeX input for a change controlled document: changes to it may be merged to master via a normal ticket, but are not "official" until an approved RFC releases a new version of the LDM document. This allows some flexibility in development while maintaining control of the official version.

The `hsc.yaml` schema in `sdm_schemas` is a "concrete" schema that is not subject to change control, but the `ci_hsc` integration tests verify that the outputs from that pipeline execution comply with the schema specified in `hsc.yaml`. This ensures that an up-to-date schema is always available, so that steps like loading an HSC reprocessing run into qserv do not require fixing up all the changes to the schema since the previous ingest.

All other schemas in the `sdm_schemas` repository are "concrete" schemas reflecting specific sets of data products; these may be edited as necessary by a normal ticket workflow.

4.1 Science Data Model and Science Pipelines

Data products that are generated by the science pipelines must conform to a physical Felis-defined schema in order to be loaded into databases. The pipelines generate a variety of intermediate catalog products, often in afwTable FITS files, which must be transformed into

the user-facing tables by the pipelines team. This transformation is more than naming and units: e.g. fluxes may need to have calibrations applied, or uncertainties in pixel units may be transformed to angular units by using WCS information.

This transformation process is not change controlled; the pipelines team has full control over how the user-facing tables are generated. It is only the definition of the user-facing tables that is change-controlled.

The pipelines code implements the various column transformations via a series of "functors", each dedicated to a particular type of transformation, and the definition of which functors are applied to which columns is defined by a YAML configuration file. This YAML file has an entirely distinct format from the Felis-defined schemas, and its purpose is distinct. The pipelines code generates data products that comply with a particular Felis schema, but there is no automatic linkage between the two. Simple verification can be performed on the pipelines' YAML file to ensure that it generates columns with the correct names, but any changes to the output data products require both the Felis YAML and the pipelines' YAML to have corresponding updates applied.

A References

[LSE-163], Jurić, M., et al., 2017, *LSST Data Products Definition Document*, LSE-163, URL `https://ls.st/LSE-163`

B Acronyms

Acronym	Description
ADQL	Astronomical Data Query Language
CAOM2	Common Astronomical Observation Model 2
CCB	Change Control Board
CI	Continuous Integration

CSV	Comma Separated Values
DM	Data Management
DML	Data Manipulation Model
DMTN	DM Technical Note
DPDD	Data Product Definition Document
FITS	Flexible Image Transport System
HSC	Hyper Suprime-Cam
IVOA	International Virtual-Observatory Alliance
LDM	LSST Data Management (Document Handle)
LSP	LSST Science Platform (now Rubin Science Platform)
LSST	Legacy Survey of Space and Time (formerly Large Synoptic Survey Telescope)
LaTeX	(Leslie) Lamport TeX (document markup language and document preparation system)
PPDB	Prompt Products DataBase
RFC	Request For Comment
SDM	Science Data Model
SQL	Structured Query Language
TAP	Table Access Protocol
UCD1	Unified Content Descriptor 1
VO	Virtual Observatory
WCS	World Coordinate System
YAML	Yet Another Markup Language