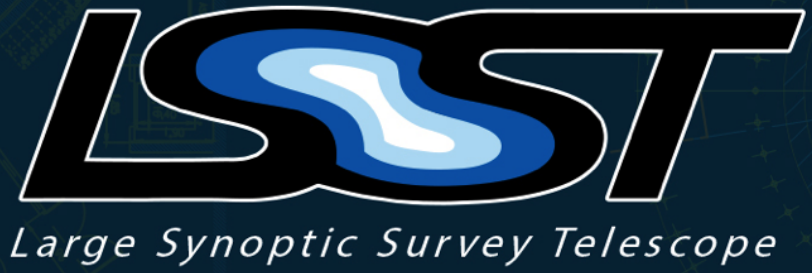
Felis - A way of describing catalogs

Brian Van Klaveren (LSST/SLAC)

IVOA Interop - College Park, MD - 11/9/2018



Why?

First and foremost, LSST needs a way to describe our Scientific Data Model.

Historically, we had a way of describing catalogs using pseudo-XML annotations in SQL comments. That's in our `"cat"` package.

The DDL was MySQL specific. Parsing properly was a bit rough.

That was insufficient, so some requirements were extracted based on needs of the project, including the development and operational environments.

The Scientific Data Model:

- MUST contain sufficient information for a physical SQL schema definition to be derived from it, given a choice of SQL flavor (e.g., MariaDB, Oracle, PostgreSQL).
- is written in YAML
- 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
- Elements MUST be described by a unique identifier ("SDM Identifier") that can be used programmatically in applications that consume the SDM YAML definition
- MAY contain additional elements beyond those required by the DPDD

The LSST Operational Environment

- We have *at least* three database technologies in use: MySQL(+ *Qserv*), Oracle, and SQLite.
  - Testing has also been done in BigQuery and Postgres.
- We are headed towards *at least* three tabular data file formats internally: FITS, Parquet, and HDF5. Maybe Avro.
- We are using three languages - C++, Python, and Java.
  - And *at least* three tabular libraries: afw.table, pandas, and astropy

JSON-LD, CSVW, and linking things

CSVW is an alternate way of annotating tables, actually CSV data, and JSON-LD is the way CSVW describes data.

CSVW is oriented towards data publishing, but a lot of Felis was derived from a combination of CSVW, SQLAlchemy, and Liquibase, in order to provide a catalog description that can be used to create database tables.

Felis will make heavy use of the `@id` property of an object. We use this to relate objects to other objects within the Felis document.

Alternatives

VOTable could help with some of the problems, but we'd still need a convention for describing indexes, constraints, override types, type mappings, and more. That would probably look like a lot of INFO, PARAM, and FIELDref elements.

Core Objects

`Schema` - which contains tables

- `name`, `description`, `tables`

`Table` - a database table

- `name`, `description`, `columns`, `primaryKey`, `constraints`, `indexes`

`Column` - A database column

- `name`, `description`, `datatype`... usually with `ivoa:ucd`, `fits:tunit` also

`Index` - A database index

- `name`, `columns`...

`Constraint` - A database constraint

- `name`, `columns`, `@type`...

`Reference` and **`Grouping`** - A grouping is a collection of references, which itself can reference another grouping. References can be simple, consisting only of the identifier of another object (`@id`), or they can be an annotated reference.

Default Data Type Mappings

Trying to get default database types based on C++/Python/Java types is kind of tough. Felis defines default mappings across languages and DBs/formats.

Type	MySQL	SQLite	Oracle	Postgres	Parquet	VOTable
boolean	BIT(1)	BOOLEAN	NUMBER(1)	BOOLEAN	BOOLEAN	boolean
byte	TINYINT	TINYINT	NUMBER(3)	SMALLINT	INT_8	unsignedByte
short	SMALLINT	SMALLINT	NUMBER(5)	SMALLINT	INT_16	short
int	INT	INTEGER	INTEGER	INT	INT_32	int
long	BIGINT	BIGINT	NUMBER(38, 0)	BIGINT	INT_64	long
float	FLOAT	FLOAT	FLOAT	FLOAT	FLOAT	float
double	DOUBLE	DOUBLE	FLOAT(24)	DOUBLE PRECISION	DOUBLE	double
char	CHAR	CHAR	CHAR	CHAR	UTF8/STRING	char[]
string	VARCHAR	VARCHAR	VARCHAR2	VARCHAR	UTF8/STRING	char[]
unicode	NVARCHAR	NVARCHAR	NVARCHAR2	VARCHAR	UTF8/STRING	unicodeChar[]
text	LONGTEXT	TEXT	CLOB	TEXT	UTF8/STRING	unicodeChar[]
binary	LONGBLOB	BLOB	BLOB	BYTEA	BYTE_ARRAY	unsignedByte[]

Vocabularies

- Would be nice to formally define SQL or dialect vocabularies
- Need some formalization on a few IVOA-specific types
  - UCDs - `ucd:ucd` -> `"ucd":"http://ivoa.net/rdf/"`
  - utype - `votable:utype` -> `"votable":"http://ivoa.net/rdf/votable/`
  - xtype - `votable:xtype` -> ...
  - unit - `votable:unit` ->
- Need to declare a minimal set of FITS-specific headers
  - `fits:tunit` instead of `votable:unit`?
  - Of course, every item has a comment: `fits:tunit/comment`

... But for now we cheat.

Cheating the Context

This is effectively the default context. This namespaces the felis vocabulary for the document and defines some pseudo-vocabularies from different things we care about.

```
{
  "@context": {
    "@vocab": "http://lsst.org/felis/",
    "mysql": "http://mysql.com/",
    "postgres": "http://postgresql.org/",
    "oracle": "http://oracle.com/database/",
    "sqlite": "http://sqlite.org/",
    "fits": "http://fits.gsfc.nasa.gov/FITS/4.0/",
    "ivoa": "http://ivoa.net/",
    "votable": "http://ivoa.net/documents/VOTable/"
  }
}
```

Example

```
---
name: sdqa
description: The SDQA Schema
tables:
- name: sdqa_Metric
  "@id": "#sdqa_Metric"
  description: Unique set of metric names and associated metadata (e.g.,...
  columns:
  - name: sdqa_metricId
    "@id": "#sdqa_Metric.sdqa_metricId"
    datatype: short
    description: Primary key.
  - name: metricName
    "@id": "#sdqa_Metric.metricName"
    datatype: string
    description: One-word, camel-case, descriptive name of a possible metr...
    length: 30
  ...
  primaryKey: "#sdqa_Metric.sdqa_metricId"
  constraints:
  - name: UQ_sdqaMetric_metricName
    "@id": "#UQ_sdqaMetric_metricName"
    "@type": Unique
    columns:
    - "#sdqa_Metric.metricName"
  mysql:engine: MyISAM
```

Sample Output

MySQL

```
CREATE TABLE sdqa.`sdqa_Metric` (
    `sdqa_metricId` SMALLINT NOT NULL COMMENT 'Primary key.' AUTO_INCREMENT,
    `metricName` VARCHAR(30) COMMENT 'One-word, camel-case, descriptive name of a po
    `physicalUnits` VARCHAR(30) COMMENT 'Physical units of metric.',
    `dataType` CHAR(1) COMMENT 'Flag indicating whether data type of the metric valu
    definition VARCHAR(255),
    PRIMARY KEY (`sdqa_metricId`),
    CONSTRAINT `UQ_sdqaMetric_metricName` UNIQUE (`metricName`)
)COMMENT='Unique set of metric names and associated metadata (e.g., ''nDeadPix'';, '
```

Postgres

```
CREATE TABLE sdqa."sdqa_Metric" (
    "sdqa_metricId" SMALLSERIAL NOT NULL,
    "metricName" VARCHAR(30),
    "physicalUnits" VARCHAR(30),
    "dataType" CHAR(1),
    definition VARCHAR(255),
    PRIMARY KEY ("sdqa_metricId"),
    CONSTRAINT "UQ_sdqaMetric_metricName" UNIQUE ("metricName")
)

COMMENT ON TABLE sdqa."sdqa_Metric" IS 'Unique set of metric names and associated me
```

Future Directions

Groupings and References

Some work needs to be done to get the groupings and references finished.

Populating TAP_SCHEMA

It's a core requirement from the LSST Scientific Data Model.

Metadata in Parquet

Use the native `List<KeyValue>` objects in Parquet on Table and Column objects to store metadata. When presented with more complicated objects (Groupings), serialize them to JSON under their proper name.

~~VOTable~~ JSON

A VOTable Vocabulary can allow us to mix in VOTable semantics in JSON without defining an explicit mapping. JSON-LD, with it's well-defined processing algorithms and API, can enable transformation of JSON into user-friendly formats for user consumption and processing. Plus we have linking.

CSVW

CSVW is another options for annotating CSV data. It is used by Google for describing datasets. Felis is very compatible with CSVW.

More info

https://felis.lsst.io and https://github.com/lsst-dm/felis

(In retrospect, the obvious name was lynx…)