Metadata Store

Metadata Store is a central repository that stores metadata information about all data sets kept in the Data Store, including provenance, and information about results from recently executed queries. Typically, for Level 1 and Level 2 metadata will be produced by pipelines, and automatically captured. Already uploaded metadata may need to be updated (not all information is known at the time of ingest). For Level 3, users will be able to upload metadata (e.g., through a web form). Note that in many cases information will be extracted directly from the data ad-hoc, or kept locally with the data. General philosophy: keep the metadata stored inside central repository (searching over a distributed file system will be too slow).

Metadata should contain enough information to easily service SIA v2 queries. It is also worth exploring if/how much we want to be compatible with Common Archive Observation Model (CAOM-2.0).

Metadata Store will contain:

- "global" database with the following tables:
    - "User". Contains information about all users, including limits for different resources
    - "Repo". Contain basic information about every repository.
    - "FileRepo". Contains file-repository-specific information.
    - "DbRepo". Contains database-repository-specific information.
- a dedicated database for each file-repository with the following tables:
    - "FileMeta". Contains information about one file
    - If it contains FITS files, also: "FitsMeta", "FitsKeyValues", "FitsPositions"
    - If it contains config files, also tables-to-be-defined.
- a dedicated database for each database-repository with the following tables:
    - MySQL's information_schema. Most likely, it will be "federated" from the right place
    - "DbMeta". Contains information about one database
    - "DDT". A Data Definition Table will be maintained to augment standard MySQL-managed metadata. DDT will contain:
        - UI-specific information such as display precision, or whether to display a given column by default
        - VO-compliance related metadata, such us UIDs for each column.

For most of the above tables, there will be one <name>Annotations table, for keeping annotations added by users.

A very rough sketch of schema for the above is captured in the description of

⚠ DM-1634 - Jira project doesn't exist or you don't have permission to view it.

Provenance. Metadata Store is responsible for keeping track of all provenance (execution trace), or, in other words, information "how we came up with the data". Provenance will be used primarily:

- to build supported virtual data products,
- to QA the data, for example to detect which parts of data were affected by a faulty algorithm or a bad node,
- as a recipe how to regenerate intermediate results.
- Things to capture for each data set
    - Intention
    - Tasks run
    - Options and configs used: default values for algorithms run, overwrites for tasks run, customizations
    - User who ran
    - Timestamp of beginning of execution

Random notes

- Locations: Repository root path (product id) – url for image repository, database name / host for database. Physical location should be wrapped and not exposed to users, to allow for data migration
- Appropriate per-user access control measures will be needed.

Repository-specific Metadata

Repository-specific metadata will be managed per-repository. For example, image metadata corresponding to Data Release X will be stored in the Exposure tables in the database for Data Release X (and will be immutable), while image metadata for images imported by user Joe will reside in Joe's workspace (and might be either immutable, or not, depending on user's preferences).

Image Metadata

Image Metadata will be extracted directly from FITS headers, or provided by pipelines as it generates images. Supported FITS header format - anything that cfitsio reads. While the structure of production images will be well defined and thus easy to capture and optimally index, image files coming from other sources (e.g., imported by users) might contain custom keywords, and their semantics might be harder to comprehend. In such cases, all metadata will be captured (in a form of flat key-value pairs), and limited indexing will be provided. External data sets imported as precursor data will go through necessary transformations (in the files or on-the-fly) to convert them into a supported version.

Potential complication: we might need to develop mapping from original FITS keywords to more canonical and useful standardized names. (Question: will that be necessary for LSST L1/L2 production?)

Note that FITS headers are generally static, and all updates / versions will be tracked through the Image Metadata stored in the database.

Database Metadata

Typically, this information will be fetched directly from Qserv, or MySQL Information Schema. In addition, for each table, a Data Definition Table (DDT) will be maintained to augment standard MySQL-managed metadata. DDT will contain:

- UI-specific information such as display precision, or whether to display a given column by default
- VO-compliance related metadata, such us UIDs for each column.

Essential Tools

- Scan metadata: verify if all files are present in data sets
- Scan all data sets, find and record directories/files/databases/tables not recorded in metadata
- Scan all data sets, verify checksums
- Prune data sets: delete directories/files/databases/tables that fall under certain threshold