Metadata

We need to store metadata for pretty much everything, including every Image, Source, DIASource, Object etc. It consists of two parts:

- provenance (execution trace, or in other words information "how we came up with the data")
- information about data not captured by provenance, including summary statistics. It has to be flexible (key:value pairs are considered) and it has to scale to billions of entries.

Information related to Provenance is documented in Provenance

Related discussions: DataAccWG telecons:

- May 09, 2007
- May 25, 2007

Possible Implementations

There are a number of potential designs for implementing the metadata requirements. One (or more!) of these might be selected.

Table per value type

This strawman from Jacek is an example of the table-per-value-type implementation. Each type of metadata value (including information from multiple keys) is stored in its own table containing as columns an id and a value.

Strawman

- + easily extensible
- + could use one copy of each value
- + may compress well
- + fairly simple to search
- - difficult to SELECT *
- - multiple tables to persist object

Table per key

Each metadata key would have its own table containing as columns an id and a value of appropriate type.

- + easily extensible
- + compresses well
- + fairly simple to search
- - difficult to SELECT *
- - multiple tables to persist object

Table per key group

This implementation modifies the table-per-key one by grouping keys that are frequently accessed together, making some queries more efficient.

- + easily extensible
- + may compress well
- 0 fewer tables to persist object
- 0 harder to search
- - difficult to SELECT *

Table per data type

In this implementation, each underlying data type that has metadata attached to it is represented by a single table. Columns in this table correspond to the keys allowed for the underlying type's metadata.

- + simple to search
- + simple to SELECT *
- + single table to persist object
- - extensible by adding columns (can be expensive)
- - compresses poorly

Single table, field per value type

One huge table with an id column, a key column, and many nullable columns, one for each value type, is used in this implementation.

- + fairly simple to search
- + single table to persist object
- + easily extensible
- 0 harder to SELECT *
- - compresses poorly

Single table, string value

A variant of the previous implementation in which all values are stored exclusively as strings in a single value column.

- + fairly simple to search
- + single table to persist object
- + easily extensible
- + simple to SELECT *
- 0 may compress poorly

Single table, single string

This implementation stores all key/value pairs in a single string column.

- + very easy to persist objects
- + easily extensible
- + trivial to SELECT *
- 0 may compress poorly
- - nearly impossible to search (requires LIKE or regexps and full-table scan)

XML

This is listed for completeness only. We do not believe it to be adequately scalable.