The metadatavalue table

The `metadatavalue` table is an important table in DSpace because it holds the metadata values of DSpace objects (ie. article title, authors, identifiers). The table uses the so-called EAV (Entity-attribute-value) model as further explained in this article.

All the metadata is:

- flat, two levels deep (the form is schema.element.qualifier, e.g. "dc.contributor.author"; qualifier can be NULL, e.g. "dc.title")
- not type-checked (everything is a string) with unlimited length (the type is "text" in Postgres, "CLOB" in Oracle).
- the same object may have multiple values of the same metadata field
- the order of metadata values is not defined, but DSpace does preserve the order wherever possible (because author names are modeled this way and their order is significant)

In the past, up to and including DSpace 4, the `metadatavalue` table used to hold metadata **only for items**. In DSpace 5 and later, it holds the metadata for all DSpace objects (definition: bitstream, bundle, item, collection, community, site, group, eperson).

The EAV model

The table doesn't model the metadata fields (e.g. dc.title, dc.relation.uri, ...) as column names (called attributes in the relation model), but uses the EAV (Entity-attribute-value) model instead, modeling the field name itself as a value in the `metadata_field_id` column.

The EAV model is typically used for modeling sparse matrices (tables where there is a large ratio of NULL cells to cells containing data). But the reason why it is used in DSpace is that it makes **changing the metadata schema** (adding/removing fields) easier to do in the application (DSpace) rather than the database (where DDL must be used, which may take a long time and typically makes the table unavailable until the operation is completed). Adding /removing metadata fields is thus effectively a routine user operation (available to what DSpace calls the "Administrators" group) rather than a task for the sysadmin of the DSpace server.

Metadata in DSpace: schema and fields registry, values

The typical operation is to look up a certain metadata value for a DSpace object with a given ID (DSpace 6+ uses a UUID in the `dspace_object_id` column, DSpace 5 uses an integer in the `resource_id` column). First, we find the metadata field in the metadata registry. Say, we're looking for the metadata value (the list of authors) of an **item** (resource_type_id = 2) with a given **ID** (dspace_object_id='f2cf9909-0357-4037-8305-4a426bf9a826') where the metadata field is "**dc.contributor.author**". First, we look up the ID of the "dc" schema in the `metadataschemaregistry` table (example code for DSpace 6+):

```
SELECT metadata_schema_id
FROM metadataschemaregistry
WHERE short_id = 'dc';
```

This gives us `metadata_schema_id` of "1". Next, we look up the metadata field "**dc.contributor.author**" in the `metadatafieldregistry` table. We use the metadata_schema_id of the "dc" schema we just found, we query for element = 'contributor' and qualifier = 'author':

```
SELECT metadata_field_id
FROM metadatafieldregistry
WHERE metadata_schema_id = 1
AND element = 'contributor'
AND qualifier = 'author';
```

This gives us the `metadata_field_id` value of "9". Finally, we can use this value to ask for the metadata field containing authors in the `metadatavalue` table:

```
SELECT text_value
FROM metadatavalue
WHERE metadata_field_id = '9'
AND dspace_object_id='f2cf9909-0357-4037-8305-4a426bf9a826';
```

The same, written as a single query:

```
SELECT text_value
FROM metadatavalue
WHERE metadata_field_id = (
  SELECT metadata_field_id
  FROM metadatafieldregistry, metadataschemaregistry
  WHERE metadatafieldregistry.metadata_schema_id = metadataschemaregistry.metadata_schema_id
  AND short_id = 'dc'
  AND element = 'contributor'
  AND qualifier = 'author'
)
AND dspace_object_id='f2cf9909-0357-4037-8305-4a426bf9a826';
```

The rows returned by the query will contain all the authors of the item in the `text_value` column.

If you use these queries often and in an ad-hoc manner, you may find it helpful to use these snippets of SQL as functions. You can find the definitions here: Helper SQL functions for DSpace 6

```
SELECT text_value
FROM metadatavalue
WHERE metadata_field_id = ds6_metadata_field2id('contributor', 'author')
AND dspace_object_id='f2cf9909-0357-4037-8305-4a426bf9a826';
```

Example

The following query selects the titles and handles of collections whose titles end with "Research".

Example 1

```
SELECT metadatavalue.text_value, handle.handle
FROM collection
INNER JOIN metadatavalue ON collection.uuid = metadatavalue.dspace_object_id
INNER JOIN handle ON collection.uuid = handle.resource_id
WHERE metadatavalue.metadata_field_id = 64
AND metadatavalue.text_value LIKE '%Research';
```

See also

- Metadata for all DSpace objects
- Storage Layer#RDBMS/DatabaseStructure
- Metadata and Bitstream Format Registries
- Helper SQL functions for DSpace 6