Support for Multiple Schemas


The basic idea is to generalise the code and DB tables around Dublin Core. Conceptually speaking, a new column is added to

```
DCTYPEregistry
```

, and the same mechanisms used for Dublin Core may be used for other schemas. I believe Jim+Downing has already implemented something along these lines.

Backwards compatibility is important and not actually too difficult. Most of the 'under the hood' changes can be made with no need to change any UI code or other code in

```
org.dspace.app.*
```

.

Database changes

From a backwards-compatibility point of view, keeping the table names the same might be easiest. However, changes to table names are masked from most code through the org.dspace.content.Item API. We may be able to use D views for other cases. There would need to be refactoring in a couple of other areas, but from an architectural consistency and code manageability / understandability viewpoint, I think this would be worth it. So that's what I've assumed below.

A new table,

```
MetadataSchemaRegistry
```

is added, and

```
DCTYPEregistry
```

is renamed

```
MetadataFieldRegistry
```

and modified to relate to the schema registry. Note the UNIQUE constraint on element / qualifier is removed (can easily see >1 schema having "title").
CREATE TABLE MetadataSchemaRegistry
{
    metadata_schema_id INTEGER PRIMARY KEY,
    namespace VARCHAR(256),
    short_id VARCHAR(32) -- e.g. 'dc'
};

CREATE TABLE MetadataFieldRegistry
{
    metadata_field_id INTEGER PRIMARY KEY,
    metadata_schema_id INTEGER REFERENCES MetadataSchemaRegistry(metadata_schema_id),
    element VARCHAR(64),
    qualifier VARCHAR(64),
    scope_note TEXT
};

DCValue would be renamed to MetadataValue,
but remain the same. (Note that source_id is removed since it's an architectural relic.)

CREATE TABLE MetadataValue
{
    metadata_value_id INTEGER PRIMARY KEY,
    item_id INTEGER REFERENCES Item(item_id),
    metadata_field_id INTEGER REFERENCES DCTypeRegistry(dc_type_id),
    text_value TEXT,
    text_lang VARCHAR(24),
    place INTEGER
};

We can create a view

DCValue

for backwards compatibility:
CREATE VIEW DCValue AS
SELECT MetadataValue.*
FROM MetadataValue, MetadataFieldRegistry
WHERE MetadataValue.metadata_field_id = MetadataFieldRegistry.metadata_field_id
AND MetadataFieldRegistry.metadata_schema_id = 1;

We could define '1' as a special value for

```
metadata_schema_id
```

for Dublin Core. (Can we make

```
metadata_value_id
```

appear as

```
dc_value_id
```

? Not that it probably matters.)

### Code Changes

By definition anything in the DSpace application/interface layer

```
org.dspace.app
```

won't be affected as it is using the

```
org.dspace.content.Item.getDC
```

method. Of course additional functionality will be needed in the UI (administration UI etc.) to realise the schema support but everything should work as before when the relevant changes are made elsewhere. Care will be needed to do everything in a way that doesn't impact performance. (Don't want to add to **ScalabilityIssues1.4**)!

```
org.dspace.administer
```

New class

```
MetadataSchema
```

. Very much along the lines of existing

```
DCType
```

and other DSpace Java objects. _Maybe belongs in

```
org.dspace.content
```

?_
DCType becomes MetadataField.

getMetadataSchema and setMetadataSchema methods added.

loadDC needs updating (see below). Maybe create a backwards-compatible class DCType?

org.dspace.content

* org.dspace.content.Item will need a few changes. getDC()/setDC() etc. need to work exactly as before. Not difficult. It will also need some extra get/set methods for the MetadataField (and maybe MetadataSchema?).

* New class MetadataValue: identical to DCValue, except with a
MetadataSchema

value. Can make

DCValue

a subclass of

MetadataValue

for backwards-compatibility.

org.dspace.search

Will work with DC with no changes as it uses APIs and not direct D access. Will need to be modified to use new metadata schema values.

dspace.cfg

search parameters can be changed to index new schemas, e.g.:

```
search.index.1 = author:dc.contributor.*
search.index.2 = author:dc.creator.*
search.index.3 = title:dc.title.*
search.index.4 = medium:vracore.material.medium
```

org.dspace.browse

Code and table views will need alteration.

Other changes

- Format of

  `[dspace]config/registries/dublin-core-types.xml`

  will need to be altered. (Perhaps even can use XSLT for backwards-compatibility)

- Custom submission form stuff could be extended to take advantage of this new stuff

- Batch importer/exporter – should be easy to retain backwards compatibility (with "

  `dublin_core.xml`

  * clearly indicating Dublin Core, can use different filename for other metadata)

- History system??

- UI changes to item display page