1. A relational model for VOResource

Markus Demleitner
msdemlei@ari.uni-heidelberg.de

- Why?
- The tables
- Some queries
- The haswords ADQL function

2. Why?

The current interface for searchable registries relies on ADQL 1.0, which has never made it to REC, and the implementations are of inconsitent quality.

We now have TAP and a REC-status ADQL. ObsCore shows how to define tables in TAP servers.

Let's define RegTAP!

Thoughts have been collected on the RestfulRegistryInterfaceReq wiki page.

3. The Tables: How?

Paul Harrison translated the VOResource schema and much of the vicinitiy (VODataService, SimpleDALRegExt...) to a relational schema using

- JAXB to go from XML to Java objects and then
- JPA (plus a lot of tweaking) to turn those into relational tables.

I used this as a starting point to "hand-optimize" tables. Additionally, I created utypes basically as XPaths into VOResource trees.

Here they are (boredom alert – 14 table definitions ahead. Bear with me, we need to discuss this).

4. resource 1/3

ivoid	vor:Resource.identifier
resource_type	vor:Resource.type
created	vor:Resource.created
shortname	vor:Resource.shortName
status	vor:Resource.status
title	vor:Resource.title
updated	vor:Resource.updated
contentlevel	vor:Resource.content.contentLevel
res_description	vor:Resource.content.description
referenceurl	vor:Resource.content.referenceURL
contact_address	vor:Resource.curation.contact.address

Observations: All tables have a column ivoid; most joins will use it. It's also this table's primary key.

Most columns here translate easily from VOResource to the relational model. The utypes lead you to the schema equivalent.

5. resource 2/3

contact_email	vor:Resource.curation.contact.email
contact_telephone	vor:Resource.curation.contact.telephone
contact_ivoid	vor:Resource.curation.contact.name.ivo-id
contact_name	vor:Resource.curation.contact.name
subject	vor:Resource.content.subject
content_type	vor:Resource.content.type
source_format	vor:Resource.content.source.format
source_value	vor:Resource.content.source
version	vor:Resource.curation.version

Here's a simplification: Only one contact is allowed (our ingester discards all but the last one). Rationale: Multiple contacts are bewildering anyway, and keeping that isn't worth the cost of an extra table.

subject is supposed to be a comma separated list. Having an extra table here might be nice, though – it would, e.g., provide an easy way to figure out a complete vocabulary ("select distinct subject from..."), so I'd be easily convinced.

There's the compound source element flattened here. It is understood that all corresponding fields get NULLed when the parent element is missing.

6. resource 3/3

publisher_ivoid	vor:Resource.curation.publisher.ivoId
publisher_name	vor:Resource.curation.publisher
regionofregard	vor:Resource.coverage.regionOfRegard
stc_profile	vor:Resource.coverage.STCResourceProfile
covered_region	Resource.coverage.spaceCoverage
waveband	vor:Resource.coverage.waveband
footprint_ivoid	vor:Resource.coverage.footprint.ivo-id
footprint_url	vor:Resource.coverage.footprint
rights	vor:Resource.rights
harvested_from	N/A

The main evil things here are covered_region and stc_profile. In my current implementation, the first is completely missing, where the second simply is an STC-S representation of whatever STC-X came with the RR. I don't really see how to represent what subset of STC here. Temporal coverage? Spectral coverage? How about spatial unions and intersections?

harvested_from is housekeeping and won't be part of the standard.

7. capability

ivoid	N/A
cap_index	N/A
cap_type	vor:Resource.capability
cap_description	vor:Resource.capability.description
standard_id	vor:Resource.Capability.standardID

cap_index is a small integer assigned by the ingester. It's part of capability's primary key (ivoid, cap_index)

This table is referenced from interface, and it references resource.

8. capability_detail

ivoid	N/A
cap_index	N/A
detail_utype	N/A
detail_value	N/A

This table references capabilities and should contain utype-value pairs for stuff from registry extensions. Example:

```
gavo=# select top 3 detail_utype, detail_value from rr.capability_detail;
             detail_utype             I detail_value
--------------------------------------+--------------
 Resource.capability.imageServiceType I Cutout
 Resource.capability.maxFileSize      I 10000000
 Resource.capability.maxRecords       I 500
```

Enumerations are still possible, but no parent/child relations. That would be a problem if we wanted to represent all of TAPRegExt in the database.

9. interface

ivoid	N/A
cap_index	N/A
intf_index	N/A
intf_type	vor:Resource.Capability.interface.type
intf_role	vor:Resource.capability.interface.role
std_version	vor:Resource.capability.interface.version
query_type	vor:Resource.capability.interface.queryType
result_type	vor:Resource.capability.interface.resultType
wsdl_url	vor:Resource.capability.interface.wsdlURL

This subsumes the attributes of the various VOResource interfaces.

10. accessurl

ivoid	N/A
cap_index	N/A
intf_index	N/A
form	vor:Resource.capability.interface.accessURL.use
url	vor:Resource.capability.interface.accessURL

We could fold this table into interface if we allowed only one access URL per interface. This would make many queries much nicer. Is there anyone out there who'd be hurt by that?

11. intf_param

ivoid	N/A
cap_index	N/A
intf_index	N/A
description	vor:Resource.tableset.schema.table.column.description
name	vor:Resource.tableset.schema.table.column.name
ucd	vor:Resource.tableset.schema.table.column.ucd
unit	vor:Resource.tableset.schema.table.column.unit
utype	vor:Resource.tableset.schema.table.column.utype
flag	vor:Resource.tableset.schema.table.column.flag
std	vor:Resource.tableset.schema.table.column.std
datatype	vor:Resource.tableset.schema.table.column.dataType
form	vor:Resource.capability.interface.param

This references interface.

12. tableschema

ivoid	N/A
schema_index	N/A
schema_description	vor:Resource.tableset.schema.description
schema_name	vor:Resource.tableset.schema.name
schema_title	vor:Resource.tableset.schema.title
schema_utype	vor:Resource.tableset.schema.utype

This is pretty much like capability, except it is referenced by the rsctable table and thus collects tables.

13. rsctable

ivoid	N/A
schema_index	N/A
table_description	vor:Resource.tableset.schema.table.description
table_name	vor:Resource.tableset.schema.table.name
table_index	N/A
table_title	vor:Resource.tableset.schema.table.title
table_role	vor:Resource.tableset.schema.table.type
schema_utype	vor:Resource.tableset.schema.utype

The primary key here is (ivoid, table_index), i.e., we table indices must be unique within a resource rather than a schema. This helps when referencing this from tablecolumn.

14. tablecolumn

ivoid	N/A
table_index	N/A
description	vor:R.t.schema.table.column.description
name	vor:R.t.schema.table.column.name
ucd	vor:R.t.schema.table.column.ucd
unit	vor:R.t.schema.table.column.unit
utype	vor:R.t.schema.table.column.utype
flag	vor:R.t.schema.table.column.flag
std	vor:R.t.schema.table.column.std
datatype	vor:R.t.schema.table.column.dataType
arraysize	vor:R.t.schema.table.column.dataType.arraysize
delim	vor:R.capability.interface.param.dataType.delim
extended_schema	vor:R.capability.interface.param.dataType.extendedSchema
extended_type	vor:R.capability.interface.param.dataType.extendedType

This table has no primary key since it is not referenced from anywhere. The datatype is a bit of a troublemaker here. VODataService allows these to be either TAPDataType or VOTableDataType, and this is currently not reflected here.

On the other hand – do we actually want delim here? extended_whatever?

15. relationship

ivoid	N/A
relationship_type	vor:Resource.content.relationship.relationshipType
related_id	vor:Resource.content.relationship.relatedResource.ivoId
related_name	vor:Resource.content.relationship.relatedResource

16. creator

ivoid	N/A
logo	vor:Resource.curation.creator.logo
creator_ivoid	vor:Resource.curation.creator.name.ivoId
creator_name	vor:Resource.curation.creator.name

We need this since a resource can have multiple creators. In some bright future, this may be used to save people the hassle of having to parse author lists, which would be Great Progress.

Of course, everyone has author lists in the creator field right now, and it's unclear whether many authors will have ivoids. I could thus be swayed to strike this table.

17. resource, capabilityvalidation

ivoid	N/A
validatedby	vor:Resource.validationLevel.validatedBy
level	vor:Resource.validationLevel

ivoid	N/A
cap_index	N/A
validatedby	vor:Resource.capability.validationLevel.validatedBy
level	vor:Resource.capability.validationLevel

Merge those? `cap_index is NULL` would work out nicely as a marker for resource-level validation, but of course you couldn't have a foreign key then...

18. rdate

ivoid	N/A
date_value	vor:Resource.curation.date
value_role	vor:Resource.curation.date.role

You've done it! These are all tables I'm proposing for a searchable registry.

That's 14 in all, with maybe 2 or 3 we could still save.

19. Digression: More tables

In implementation, I needed some additional tables:

- registries – registries I'm harvesting with dates of last full and incremental harvests
- imported – a table keeping track of which harvested files have already been ingested
- authorities – what registries claim authority over which authorities? I'm ignoring all records from registries that aren't authority for the record's ivoid, except when there's no managing registry at all; this table lets me do this.
- oaisets – for the OAI-PMH interface, I keep track of OAI sets declared (that's not worth it)
- oairecs – preformatted OAI-PMH records by ivoid for the OAI-PMH interface

20. Use cases I

On the wiki page, some use cases were presented. Those that can be answered via VOResource can be answered using this schema (see http://docs.g-vo.org/rr-usecases.rstx).

Some examples:

"Find all SIA services that provides infrared images"

```
SELECT ivoid, url
  FROM rr.capability
    NATURAL JOIN rr.resource
    NATURAL JOIN rr.accessurl
  WHERE standard_id='ivo://ivoa.net/std/SIA'
    AND waveband LIKE '%Infrared%'
```

21. Use cases II

"Find all searchable catalogs that provide a column containing redshift"

```
  SELECT ivoid, url
  FROM rr.capability
    NATURAL JOIN rr.tablecolumn
    NATURAL JOIN rr.accessurl
  WHERE standard_id='ivo://ivoa.net/std/ConeSearch'
    AND ucd='src.redshift'
```

"Find all SIA services that provides infrared images"

```
  SELECT ivoid, url
  FROM rr.capability
    NATURAL JOIN rr.resource
    NATURAL JOIN rr.accessurl
  WHERE standard_id='ivo://ivoa.net/std/SIA'
    AND 1=ivo_hasword('infrared', waveband)
```

22. The hasword function

We should mandate a function (say)
`ivo_hasword(needle, haystack) -> integer`

returning 1 if needle is contained in haystack "as a word" and in any capitalization. This should be more or less like Postgres' information retrieval functions. We want this for matching descriptions, authors, etc.

23. Open questions

- Should the table names be plural forms?
- Can we save the accessurl table? The creator table? The rdate table? Either one of the validation tables?
- Erm – I left out securityMethod. Just add it as a text field in interface
- Where do we go from here?