TORUS implemented

Jakub Skoczen, Index Data

jakub@indexdata.dk

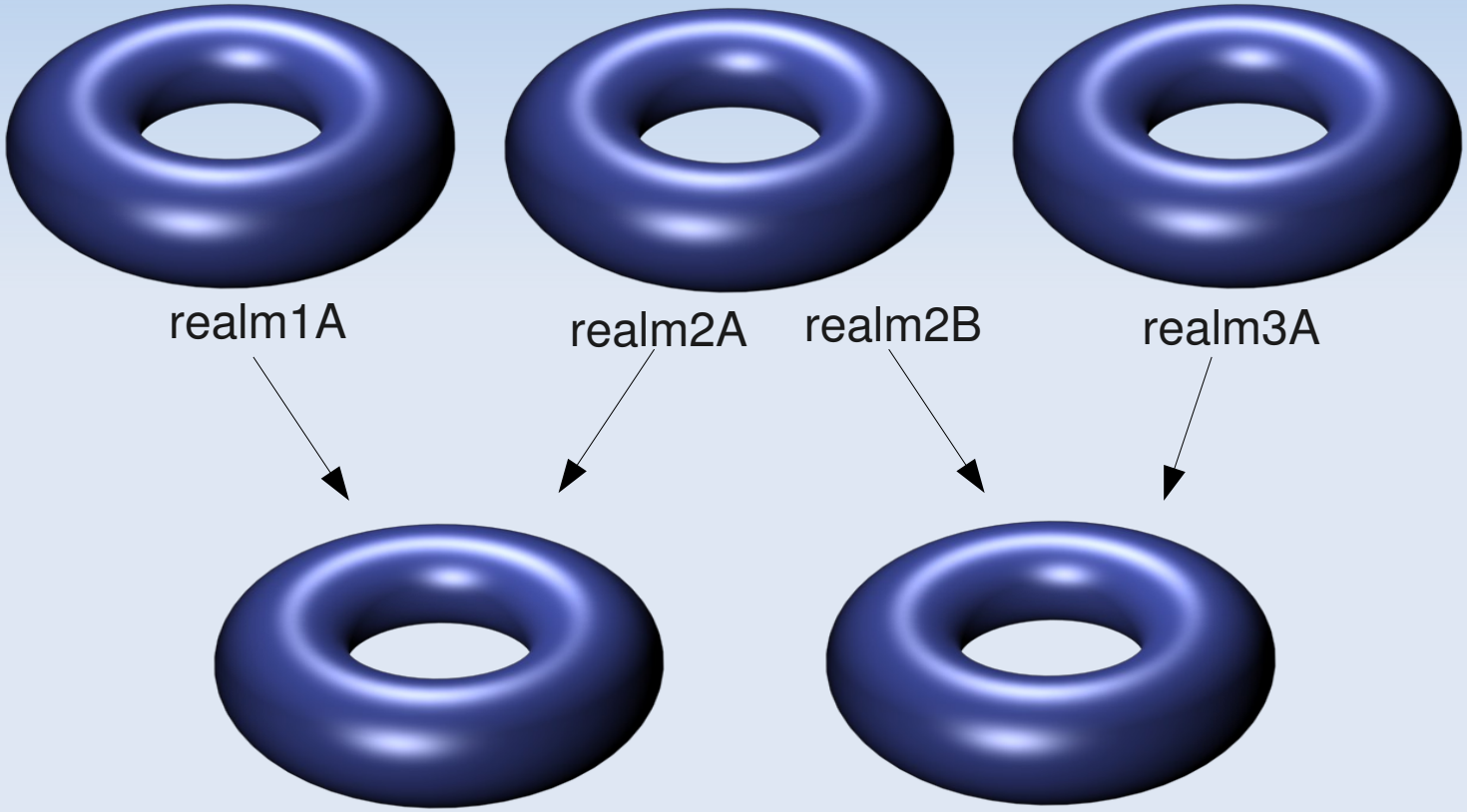
TORUS implemented

- May be viewed as a ”platform/language independent, xml-based, single inheritance model”
- Represented as a file system structure:

```
torusId/
   records/
      |- world/
      |- myrealm/
```

records ~ ”overriding” symlinks

TORUSes are stackable



TORUSes are stackable

- Makes it possible to represent hierarchical systems (oragnization>members>users) and alter/add data on each level of the hierarchy
- URI addressability - solution to overly complicated relational schemas - relations are expressed as direct links to records (one-to-one) or realms (one-to-many, many-to-many)

RESTful API

- Entities: record, layer, realm, property
- Listing records in a realm:
  GET /records/realm/?layers=original,override&query={cql}
- Creating (overriding - worldId supplied) records:
  POST record.xml /records/realm/
- Updating, Removing records from a realm:
  PUT, DELETE ../records/realm/{rec_id}

Record profiles (interfaces)

- Torus records/layers are designed to be a generic data containers, there's nothing in the API that is bound to a specific record profile

- Describes a set of required/optional properties (fields) - ”searchable” (IR target) profile, identity profile (user authentication)

Snippets of XML

Single record, final layer (after applying overrrides) response:

GET /records/auth-17/id27/?layers=final

```
<record type="searchable"
    uri="http://torus.org/records/auth-17/id27">
    <layer name="final">
        <id>auth-17/id-27</id> # internal id in the TORUS
        <zurl>z3950://loc.gov:210/voyager</zurl>
        <comment>Please ignore this comment</comment>
        <worldId>B.2</worldId> # identifies the 'parent' in the
                               # world list (internal id)
    </layer>
</record>
```