Crosswalks, mapping tables, and normalisation rules: when we don't even share the same vocabulary for authority control

by Deborah Fitchett (Library, Teaching and Learning, Lincoln University)

Tēnā koutou katoa. Hello everyone. I basically want to rant about the various kinds of messes I've had to clean up when dealing with our research output metadata.

Our research systems

At Lincoln we get data from a variety of sources. Some of it's been carefully curated, some of it really hasn't. It feeds into our Research Information Management System, Elements. From there it's used in bibliographies for researcher profiles on our website, while records with full-text files feed into our Institutional Repository, DSpace, and we create thesis records manually. All the repositories we maintain are harvested into our Library Discovery Layer, Primo, and into external aggregators.

Crosswalks
(aka transformations, normalization rules, etc)

Elements items are imported into DSpace according to a set of crosswalks using XSLT. By default, an Elements node is simply copied to a specified DSpace node, but more complex custom rules can be created. This gives you more control, but within limits: XSLT is **not** a programming language, as StackExchange reminds me **every time** I search for what I **think** will be a simple function.

Primo calls its crosswalks "normalisation rules". It provides a graphical user interface to create them – which adds its own limitations.

Preferred data sources
(aka order of precedence, etc)

An Elements record can include metadata from multiple sources, so we need to configure which source we want DSpace to take data from. For example, we trust manual data from a busy researcher less than data from publisher feeds – and we trust publisher data less than manual edits by trained admins. One data source reliably sends good titles - but no abstracts. Another reliably includes abstracts, but its article titles have problems. So we prefer the source with good titles, and have a rule to look for the first abstract available regardless of source.

Primo does a lot of complicated things we can't control to decide which metadata users get shown, but we can configure which full-text they get sent to.

Mapping terms
(aka data dictionaries, mapping tables, etc)

Our Research Office requires a specific set of item types in Elements for reporting to government. We wanted a much finer-grained set in DSpace, but mapping only works on a one-to-one or many-to-one basis, so we had to settle for "good enough for government work".

When a data dictionary is missing a value, DSpace is populated with the original Elements code – for example, when all our university's departments were renamed without anyone telling me. But a missing value in a Primo mapping table isn't imported at all. Since the item type is mandatory, the resulting record throws an error message and, if you're unlucky, the entire scheduled job is disabled pending manual intervention.

Controlled vocabulary
(aka labels, subject headings, thesauri, etc)

Instead of having some resources listed under “Māori law”, some under “Māori legal system”, some under “tikanga Māori” and so forth, a controlled vocabulary lets you list them under half a dozen terms that are almost but not quite identical. It turns out the Australia New Zealand Standard Research Codes have been implemented several different ways in different versions of Elements **and** DSpace.

DSpace has safe bulk edit functionality, but Elements synchronises with it every hour (and by “synchronise” I mean “unilaterally overwrites”) so I was nervous about potential edit conflicts. Instead we made the changes directly in the PostgreSQL database. Fortunately nothing went wrong – except then Elements changed which words they capitalise and we had to go through the whole process again.

Just to confuse things, I feel strongly about spelling Māori words correctly so we also created some rules that add the diacritics in. This means our vocabulary doesn’t quite match any existing standard – but hey, I’m sure that’s fine.

Author names

Now, over the years our repository admins have carefully and lovingly curated author names including first and preferred names – then Elements overwrote it all with a barebones surname and initials. Our HR system has more data available about our own researchers, and I spent two years crafting rules to pull this in.

Unfortunately, my first attempt changed the crucial order authors are listed in. Several other attempts didn’t work at all. My second-to-last attempt couldn’t distinguish between a Lincoln author and an external author with the same surname and initials. This turned out not to just be a theoretical problem. Finally Elements added local author identifiers, so now it’s trivial to identify which author names can be expanded with our HR data.

The ideal author ID would be ORCID. Unfortunately even different modules of the same system don’t always talk together well on this level. We crosswalk the ORCID into a custom DSpace metadata field – but because it’s associated with the item, not the author, it’s useless for authority control.

Another challenge is making it available for searching in our library discovery layer, without cluttering up author listings in aggregators who harvest the same feed. We’re creating different OAI contexts for local and public purposes to deal with this.

As soon as I heard of oadoi.org – which is fantastic – look it up – I tested it to see if it could find preprints in our own repository – and it did! It turns out the oaDOI service gets its data from BASE, which harvests from our OAI feed, which includes the dc.identifier per DSpace’s out-of-the-box configuration, which is where we store item DOIs. So sometimes you get lucky.

Tips

Now, I was going to run through a series of helpful tips, but I don’t have time and we all know neither do you. So my advice is simply: give up. Google will find it all anyway… won’t it?

Thanks very much for listening. Ngā mihi nui ki a koutou.

- Ignore metadata at your peril!
- An upgrade is a new system.
- Map one-to-one or many-to-one.
- Test with all possible input.
- Make backups.
- Fix the source of the problem first, the result second.
- Get familiar with regular expressions.
- Give up and trust Google.