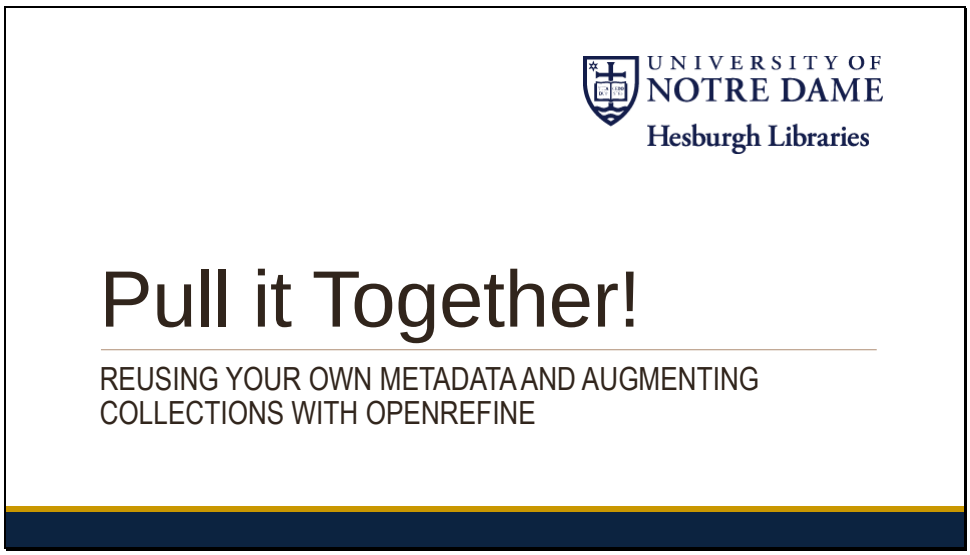
Pull it Together! Reusing Your Own Metadata and Augmenting Collections with OpenRefine

Ruth Kitchin Tillman

Publication Date

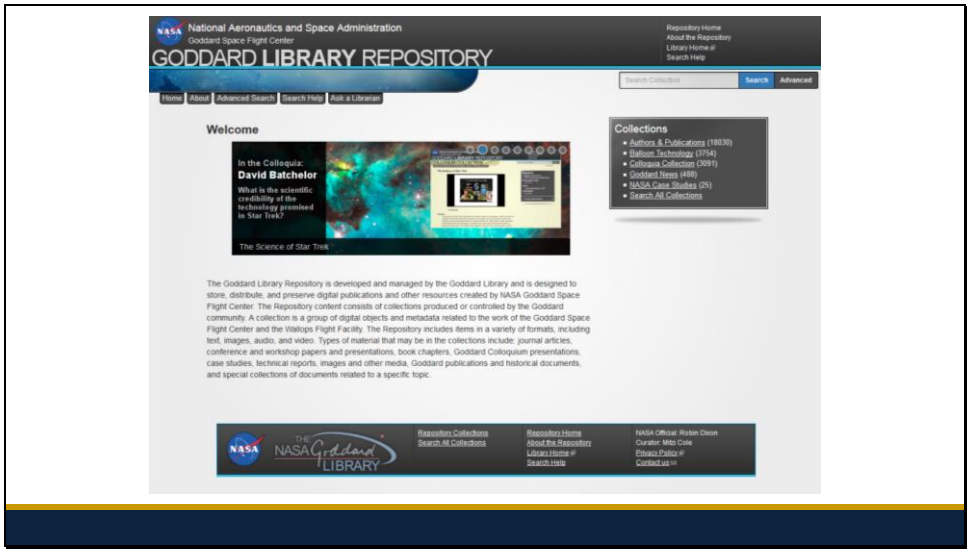
08-12-2023

Slide 1



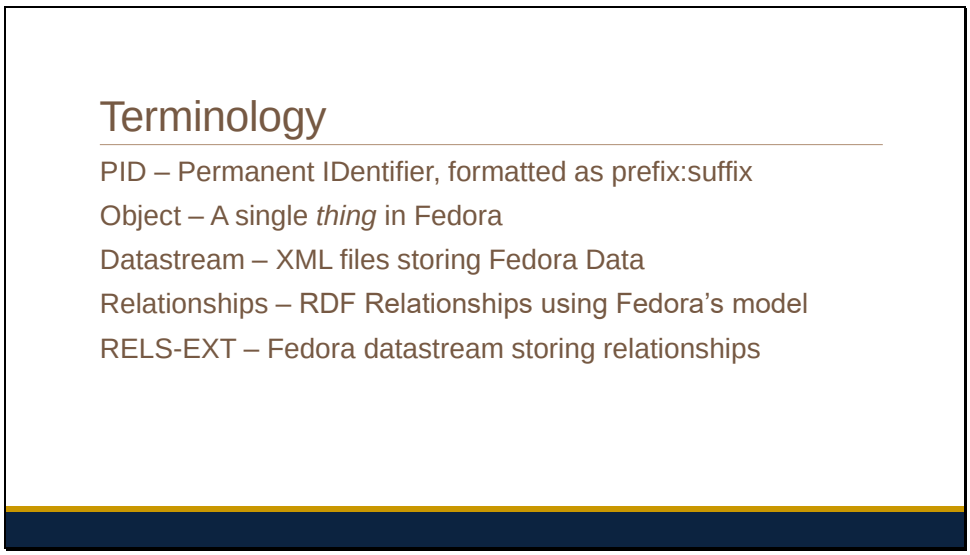
Good morning. Name is Ruth Kitchin Tillman. I'm the Digital Collections Librarian at the University of Notre Dame. However, today I'm going to be talking about work I did in a position I left earlier this year at the NASA Goddard Library. At Goddard, as at Notre Dame, I work a great deal with our institutional repository, built on the Fedora Commons Repository software. This project is one I did last Fall to export, augment, and batch update data from the repository.

Slide 2



First, a quick introduction to the system on which I was working. The Goddard Library Repository was launched to the public in February of 2011. It provides a central location for collection work produced by the Goddard Space Flight Center. It has 5 collections: Authors & Publications, Colloquia, Case Studies, Balloon Technology, and a newly-added collection for scanned issues of the Goddard News. The repository runs on a custom Fedora 3.3/Drupal 7 interface.

Slide 3



I'm not going to assume you already know Fedora terms, so I'll briefly define a few. These apply to Fedora 3 but not to Fedora 4. When talking about Fedora 4, I'll note differences. First the ***PID*** or Permanent IDentifier. Each PID consists of a PID prefix and a PID suffix. The former may be the same for an entire repository or, in this case, different for every collection.

Objects are what we would consider *things* in Fedora. An object may be a metadata record about an item with the item, such as a PDF, attached. An object may just be the metadata, possibly with a link. And object may be a link with very bare metadata.

Fedora Objects are made up of ***datastreams***, three common ones are – audit files noting changes made, brief Dublin Core records, and records recording relationships relationships. Core Fedora datastreams are XML files, but datastreams can be any attachment as well.

Objects in Fedora may have ***relationships*** with each other. For example, if I have a record or for an article and a local authority record for a creator, I can create a relationship between the two. I can say "Article 29" dc:creator "Author 13." I could also say "Article 29" isMemberOfCollection "collection:5". I would define article and author or collection using their PIDs as you will see later on.

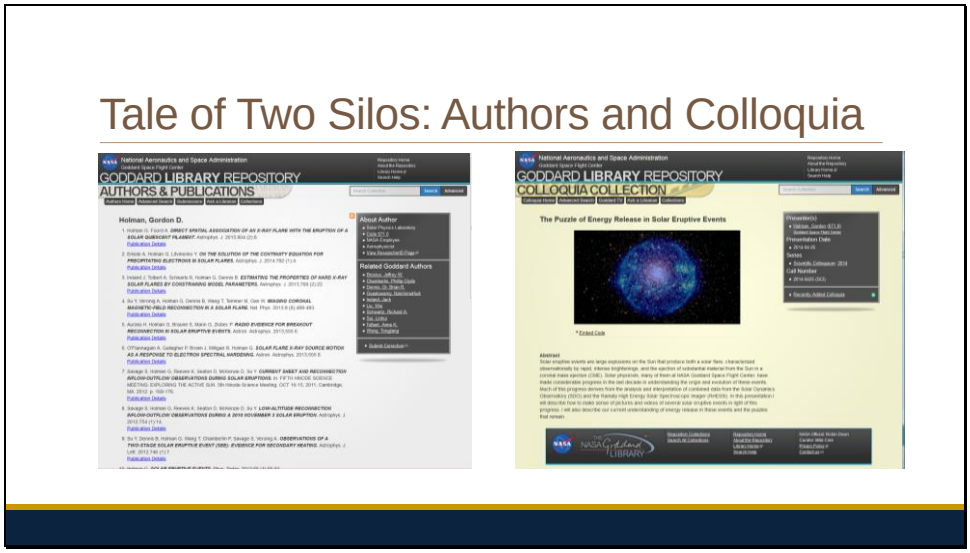
RELS-EXT is the core Fedora datastream where it stores those kind of relationships. Each object has a RELS-EXT datastream. Relationships can get very complex but the ones I'm talking about today are pretty simple.

Slide 4

The focus of this talk will be on reusing our own data to augment other data. However, steps 2-4 of the process could be done using data from an external source as well. Let's start with my particular challenge. The five collections in the repository all have different purposes and come from different sources. Each is its own silo of information and relationships. However, two of the collections could augment each other, if only we could bring together the data.

Those two collections are the Authors & Publications collection and the Colloquia collection.

Slide 5



On the left is the repository's author page for Gordon D. Holman, including his organizational code/position/title, his Goddard co-authors, and a list of his publications. On the right is a colloquia Holman delivered in 2014. In its sidebar, we get a link to a textual search of colloquia for any other presentations under his and a similar search for code 671. These two do not meet. One cannot move from an author's colloquia to their publications nor from publications to colloquia. Searching for other colloquia also relies on a textual search, which may be a problem depending on the type of data we're given. So what are the collections and why were they built this way?

In brief, Authors and Publications records the publication output of all Goddard-affiliated authors. We harvested publication records from 4 indexing services, crosswalked them into the PUB_MD article XML format, attached Open Access PDFs when possible, and created local authority files for each author using extended MADS XML. The collection currently dates back to 2003 and will not go back beyond 2000.

The Colloquia collection, on the other hand, provides information about and access to colloquia delivered *at* Goddard by a variety of speakers and scientists. It goes back much farther than Authors & Publications, including information about presentations delivered as far back as the 1970s, with newer videos online and physical copies of older ones, or audiocassettes of their recordings, available in the library. The metadata is recorded in the Goddard Extended Metadata Schema, GEMS, based on whatever information was recorded by the colloquia's organizer. When the speaker was from Goddard, we worked to get their organizational code, but hadn't done any further authority control.

As you see, the collections differ greatly in purposes and sources. Add to this that less than 20% of colloquia are delivered by Goddard authors and you'll understand why the two weren't created to interrelate.

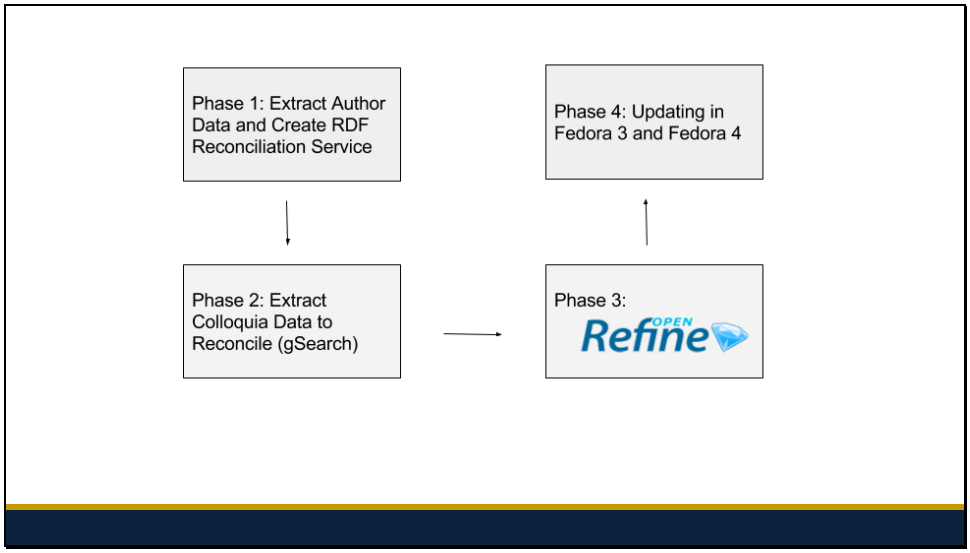
Slide 6

Still, we asked, what would it take to create relationships between the two collections? As we re-envisioned the site in terms of Fedora 4, which will involve a large transition anyway, as we thought more about RDF and relationships between objects, could we bring these two together?

Slide 7

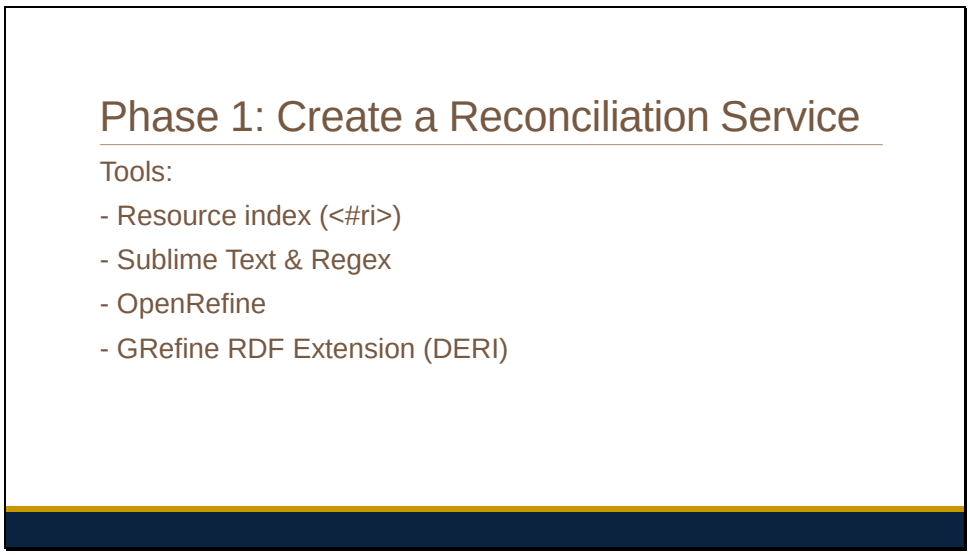
When relating one object to another, it uses Fedora's Permanent Identifier, or "PID." I'll be referring to the AuthorPID or unique identifier for an author's record as well as the ColloquiaPID, the unique identifier for a colloquia's record. As I said earlier, Fedora's architecture traditionally uses a file called RELS-EXT to store relational metadata, or how one object relates to collections, other objects, and models in Fedora. Based our models in Fedora 3 and our tentative remodeling for Fedora 4, we'd need to be update the record with either a RELS-EXT statement that Colloquia rel:isPartOf AuthorPID for Fedora 3 or a simple dc:creator relationship to the speaker's object record in Fedora 4.

Slide 8



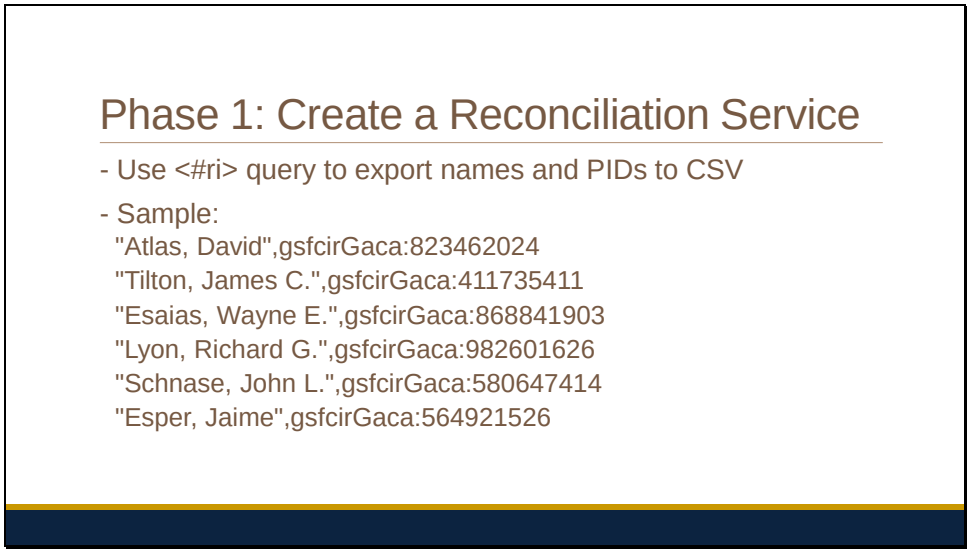
The process took four stages: Using the author data to create a reconciliation service, extracting the colloquia data to be reconciled, performing the reconciliation in OpenRefine, and finally updating the colloquia in both Fedora 3 and Fedora 4 test instances.

Slide 9



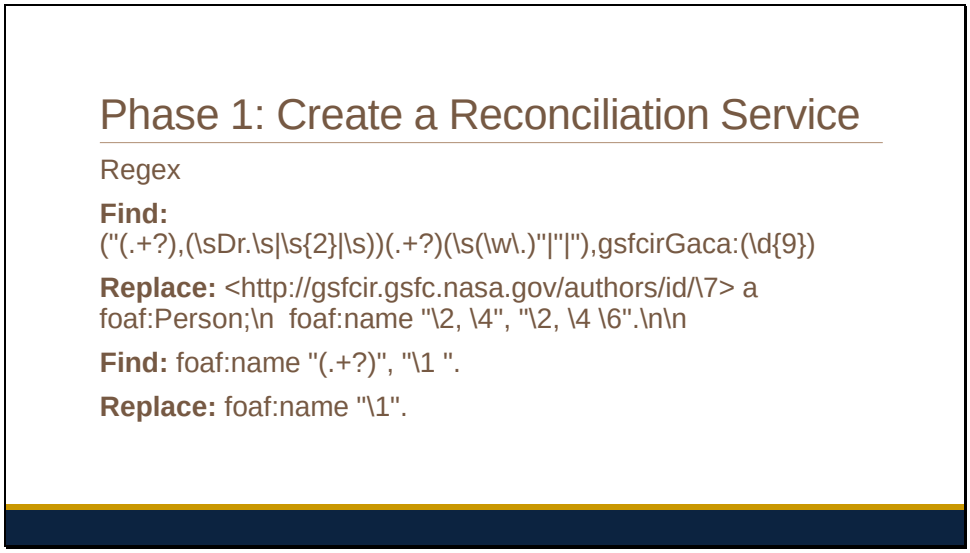
First, I had to come up with a way to match names from the Author collection with names in the Colloquia collection. To do this, I created a very basic reconciliation service with one or two versions of the author's name (depending on whether they used middle initials in their publications) and the URL of the author's page. This took a few steps and used the tools above, but was much less painful than it initially sounds.

Slide 10



We've all experienced difficulty in figuring out the best way to get our data back out of our systems, whether proprietary like an ILS or open-source and locally hosted, like Fedora. Fedora 3 indexes can be rather difficult to work with if you don't use an external indexer like Solr (and if you do). The Fedora Resource Index, for example, did not index our MADS authority files. Its gSearch index returned far more than I needed to work, although I could have parsed through that as I later did with colloquia. Fortunately, we stored the most authoritative versions of author names as Last, First Middle in a core Dublin Core file attached to the author object which contained the MADS file and which I could query from the resource index interface. I queried the Authors collection and pulled out the authoritative name and the author's PID into a CSV.

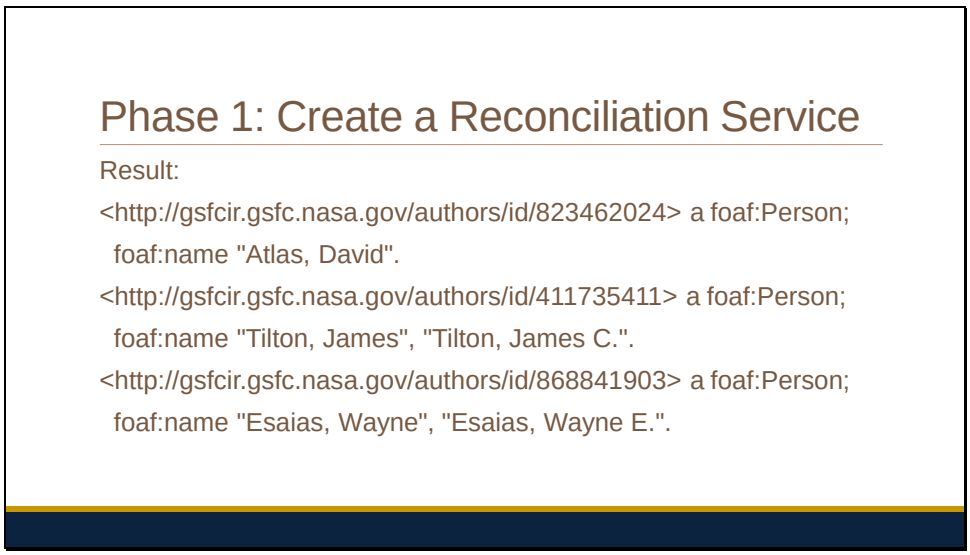
Slide 11



I could have used OpenRefine's functions to create an RDF file with author information. However, I needed to edit the author information with Regular Expressions to add a plain Lastname, Firstname in addition to Last, First Middle and decided to make the file as a part of that process. First query looks a bit like gibberish, but it sorts out the possible ways names might appear and captures them individually to be returned in the replacement. However, some people didn't have middle initials at all, which would make their record have two identical names, one with a space on the end—not a problem for the project, but also not good data—so I cleaned it up with the second Find and Replace. You may notice "Dr." in the "find" query. All I can say is that I did not create most of that name metadata.

I did all the regular expressions work in this project in Sublime Text 2, which has excellent Regex support.

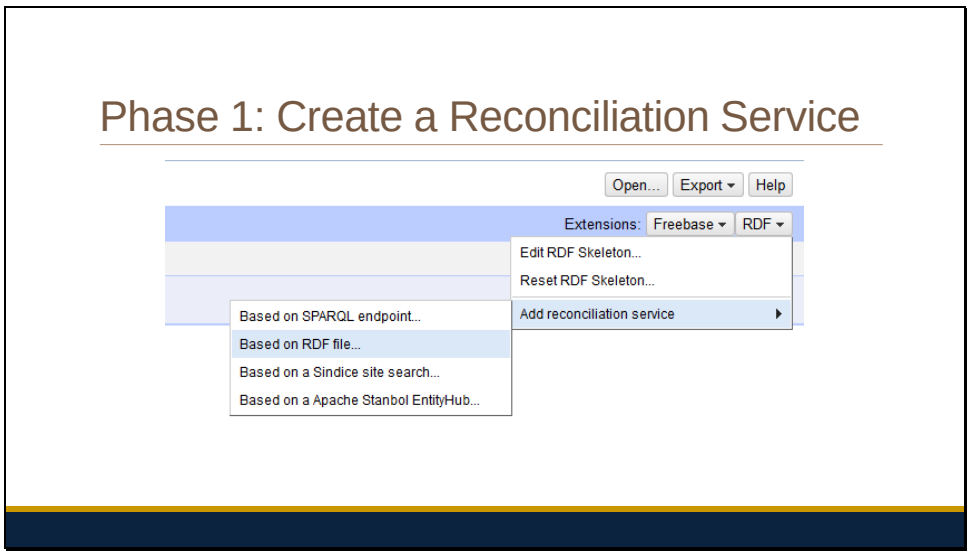
Slide 12



Here's what that apparent-gibberish on the previous screen returns. This is RDF which I'll then use to populate the reconciliation service. A URL to the author's page, an RDF type of foaf:Person (which makes OpenRefine happier than no rdf:type), and one or two versions of their name. I chose FOAF because, as you'll see later, it's one of the default vocabularies supported by the tool I would be using. I added the necessary PREFIX information above the names and saved it as a Turtle/RDF file. Turtle is the kind of RDF I find personally most usable because I can see that X has property Y very easily.

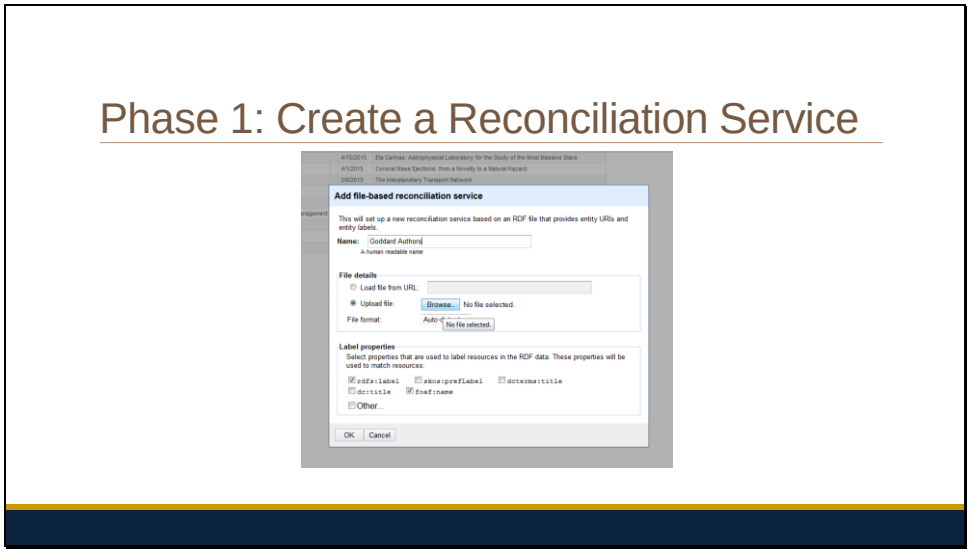
Even though I was going to attempt to create a relationship between two kinds of Objects, Author Objects and Colloquia Objects, I used URLs instead of Fedora's default Object identifier, PIDs, here for reasons I'll explain during the reconciliation process.

Slide 13



I had already added the DERI-RDF plugin to my OpenRefine instance. From the RDF drop-down, I selected “Add reconciliation service” and “based on an RDF file.”

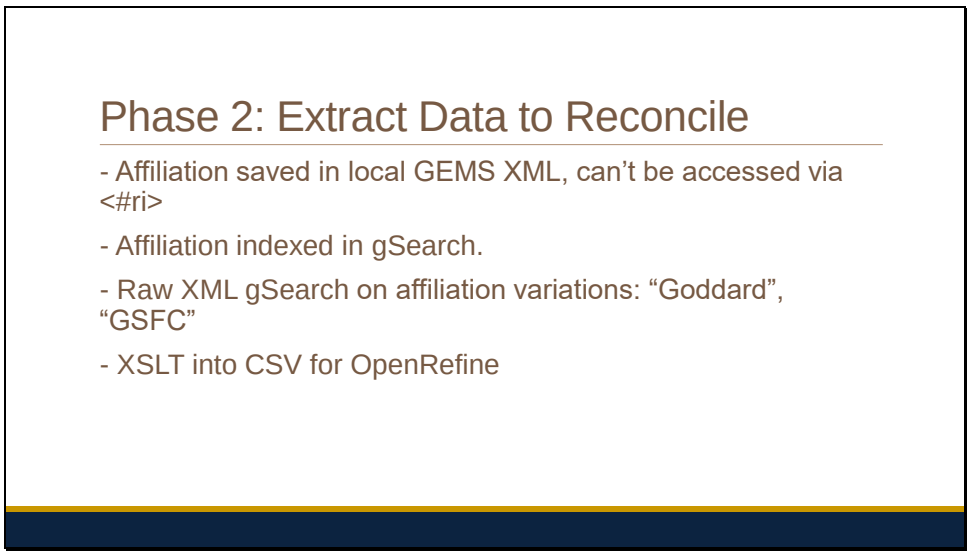
Slide 14



Adding a file-based reconciliation service with the DERI plugin is quite easy. I simply had to give it a name, upload the file, and select label properties that the service should look for. As you see, foaf:name is one of the default label properties.

Now OpenRefine is set up to use the Authors collection names and URIs for reconciliation, but what about the colloquia data?

Slide 15

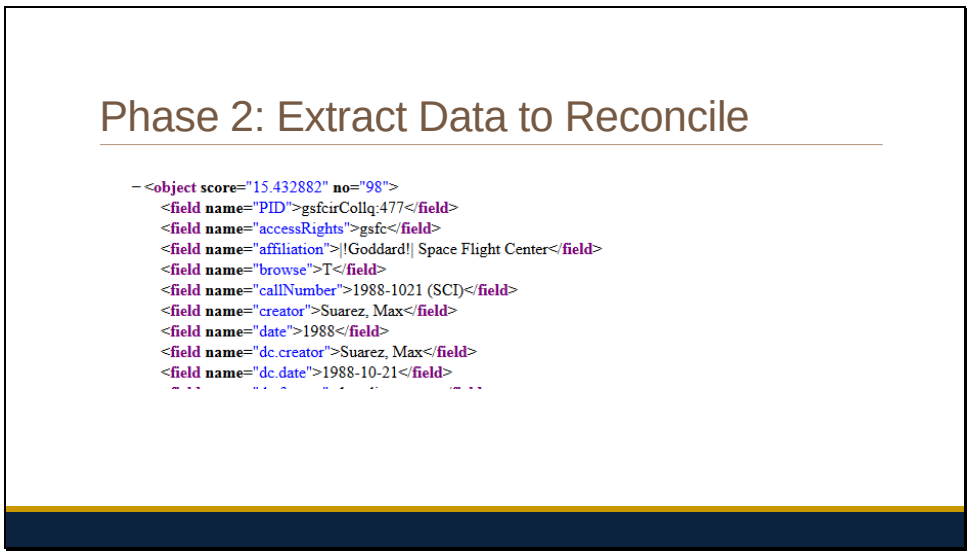


Extracting colloquia data proved to be trickier than using the resource index. Since we have over 5,000 colloquia and fewer than 20% have Goddard authors, I wanted to limit my dataset to colloquia where at least one Author came from Goddard.

Fortunately, our affiliation data was indexed in the larger gSearch, Fedora's internal site search. So I performed a gSearch query to return raw XML results for two affiliation variations—any affiliation containing Goddard or GSFC.

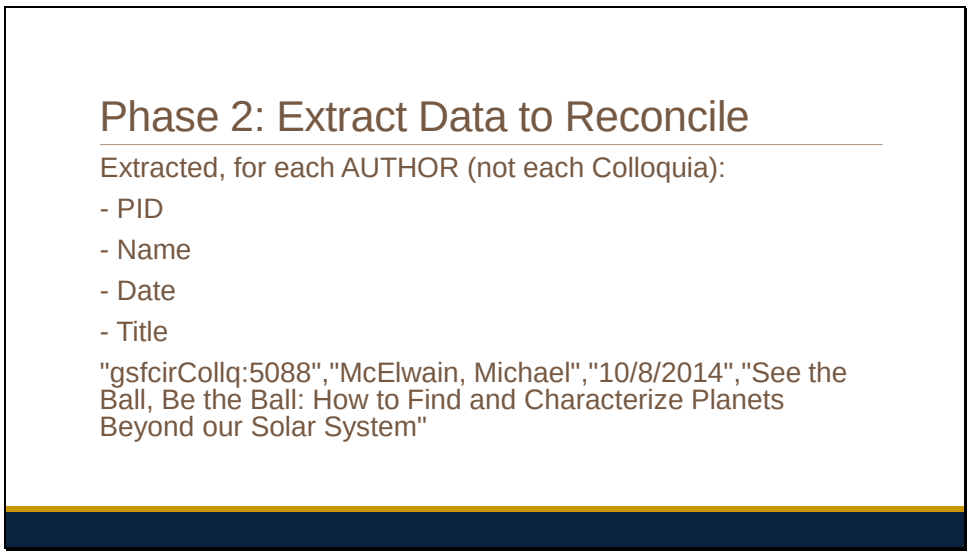
I used XSLT to crosswalk only the important data into a CSV, which could then be imported into OpenRefine.

Slide 16



This is a sample of the gSearch result data. As you see, the affiliation search highlights Goddard even when it's only a part of the field. That way I could be sure I was getting results for any variations that existed such as NASA Goddard, NASA Goddard Space Flight Center, and plain old Goddard.

Slide 17

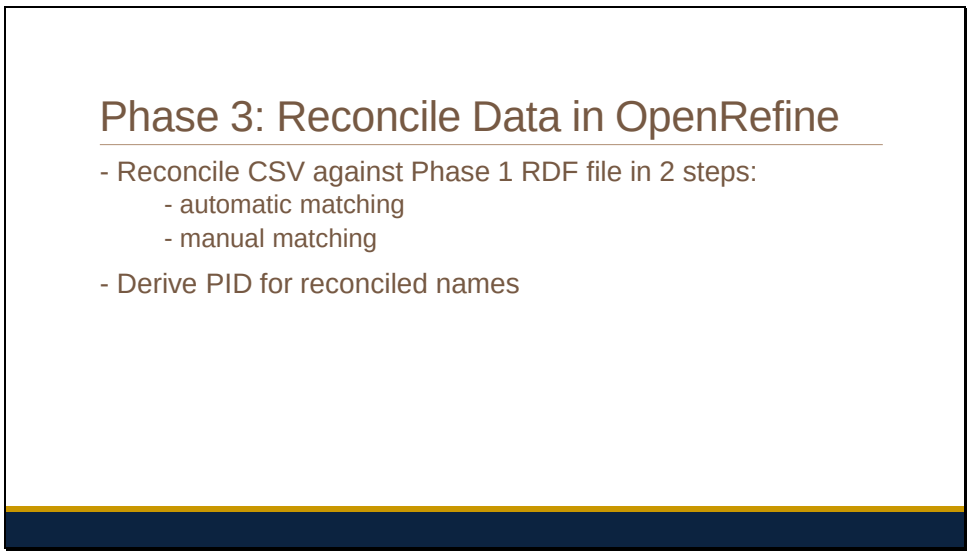


While the results I returned were for full colloquia records, each with one to many speakers, I was able to write an XSL Transformation that focused on extracting data for each creator mentioned in the record.

For each *speaker* in a colloquia, my resulting CSV had a line with the object PID, the speaker's name (note the same Lastname, Firstname format), and the date and title of the presentation in case they proved helpful in determining whether or not something was a match.

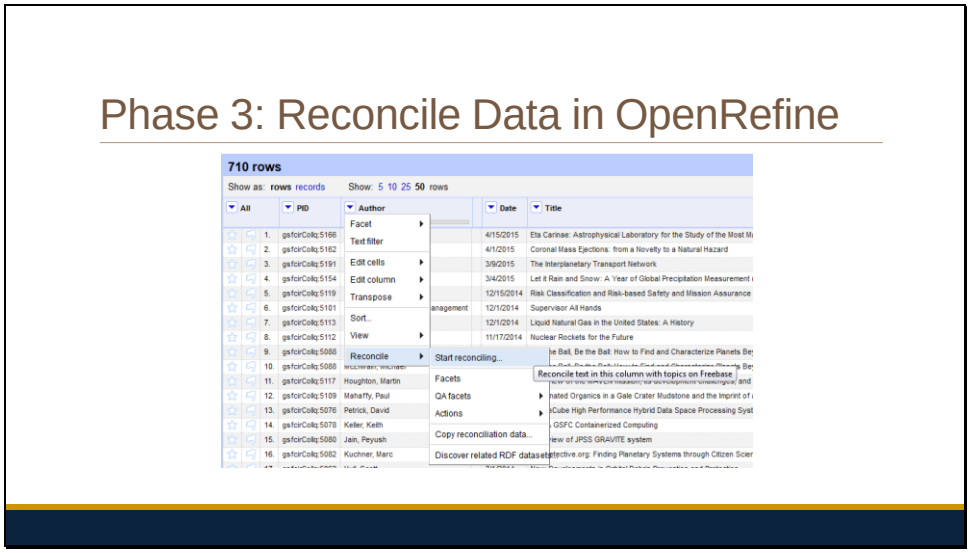
I then imported the CSV into OpenRefine.

Slide 18



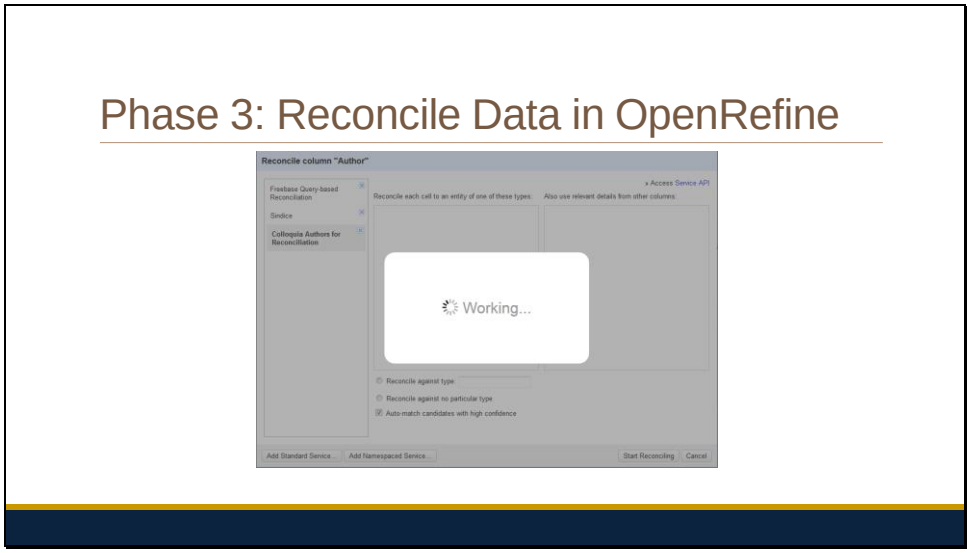
The RDF reconciliation service is remarkably straightforward. I set it running on the data, took a walk around the building, and came back to 180 matches. I then went through the rest of the results and selected from possible matches where the names didn't quite line up. Once that was done, I used Google Refine Expression Language to extract the author URI into a new column and derived the PID.

Slide 19



To reconcile on a column of data, simply select Reconcile from the drop-down menu and click Start reconciling.

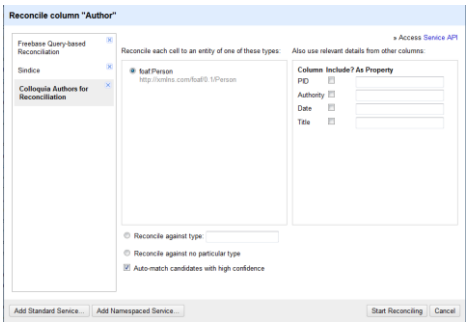
Slide 20



The reconciliation popup shows services active, including the Goddard one, and you just need to click to get it started querying the service. Even a pre-loaded set of triples like mine has to be re-parsed by the service in the context of the data you're using before it offers options for reconciling.

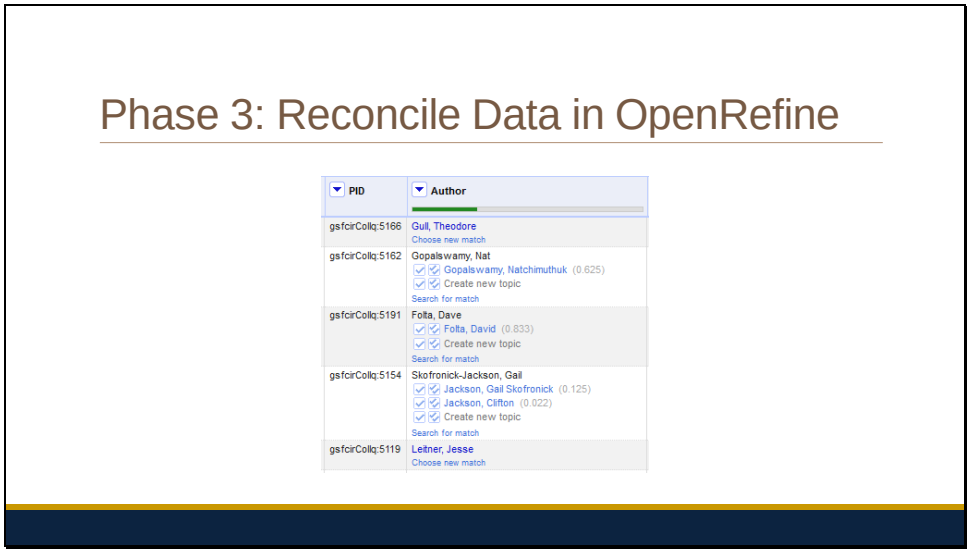
Slide 21

Phase 3: Reconcile Data in OpenRefine



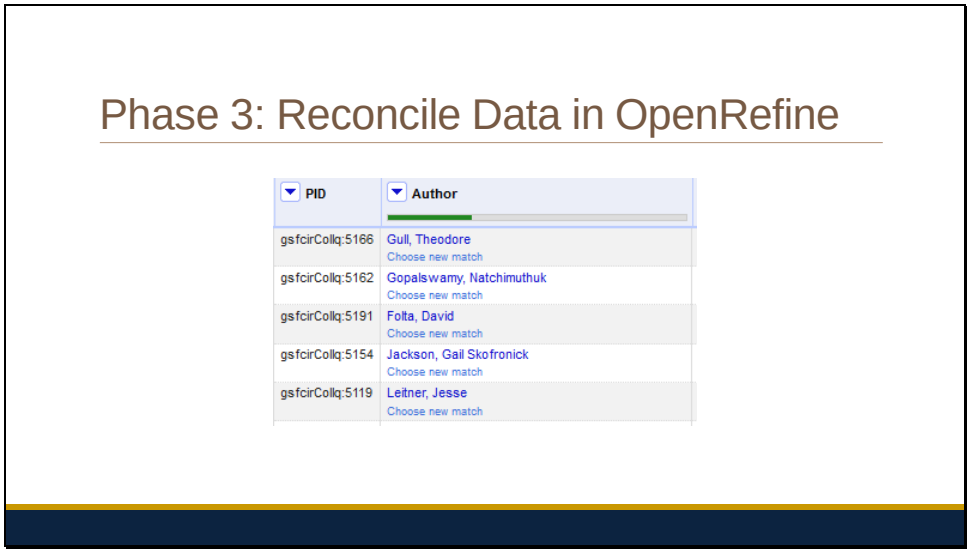
Since the only rdf:type of data was foaf:Person, it selects that as the default type against which you can reconcile. You could also specify another type or choose to reconcile against no particular type if you had multiple types. Sadly, the side column which lets you include additional data is not currently working for services with the DERI RDF plugin. In some cases, including another project of mine, it have been very useful.

Slide 22



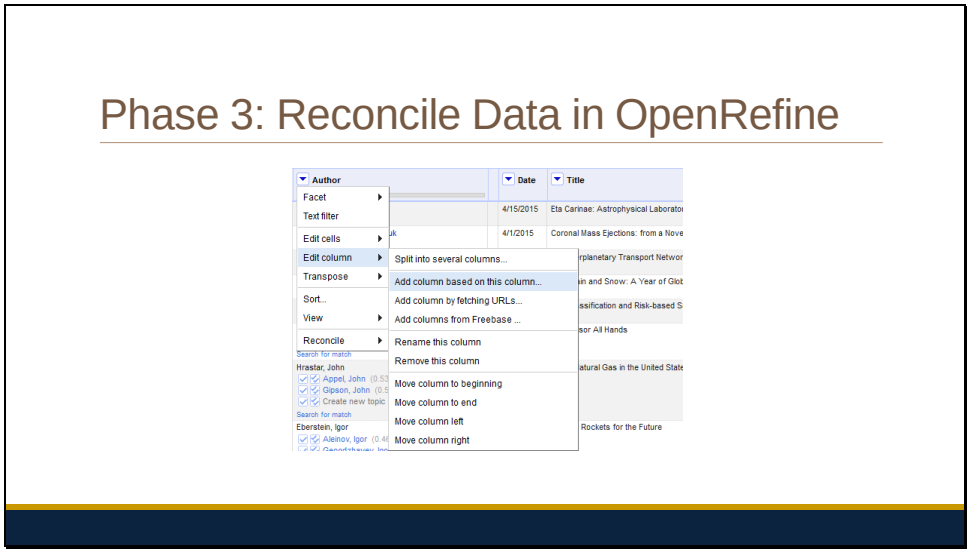
Here's an example of matches after the basic reconciliation has been run. The green line shows the percentage of matches found. Each of the others is followed by one or two links along with their matching confidence factor. This is where it's helpful to have URLs instead of system PIDs as a person's identifier. While Nat Gopalswamy and Gail Skofronick-Jackson are clear matches, it's helpful to click David Folta's link and see that yes, he works on flight dynamics analysis and thus is likely to be talking about the interplanetary transport network (title cut off in this image).

Slide 23



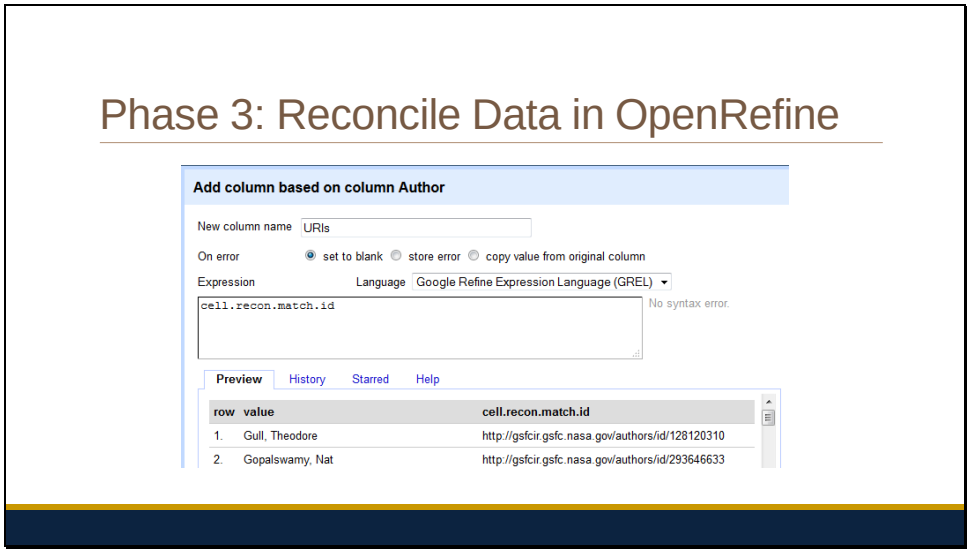
I quickly reconciled the data. Once I finished, I had the data to update over 250 colloquia. But next, I needed to get it out of OpenRefine.

Slide 24



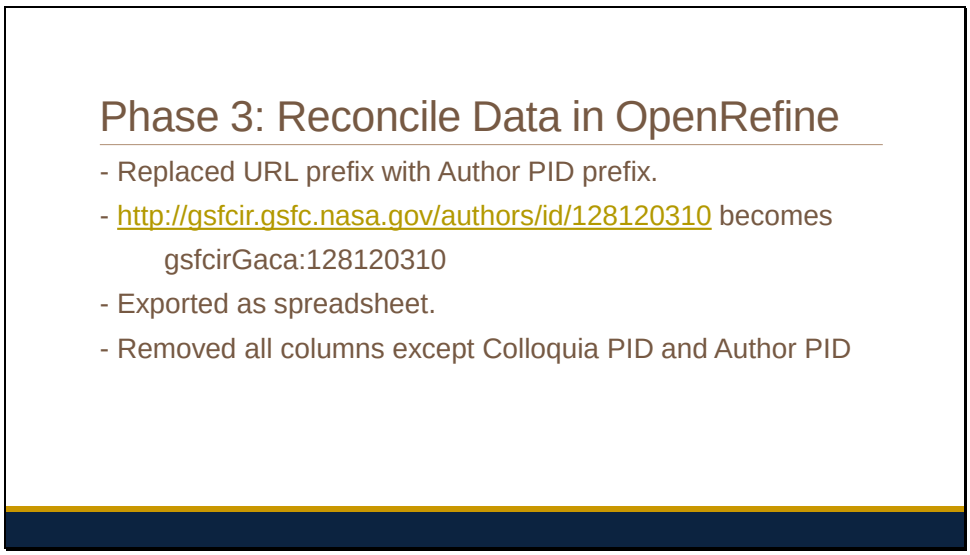
To retrieve reconciliation data, you must pull it into another column. From the column dropdown, choose to add a column based on this column.

Slide 25



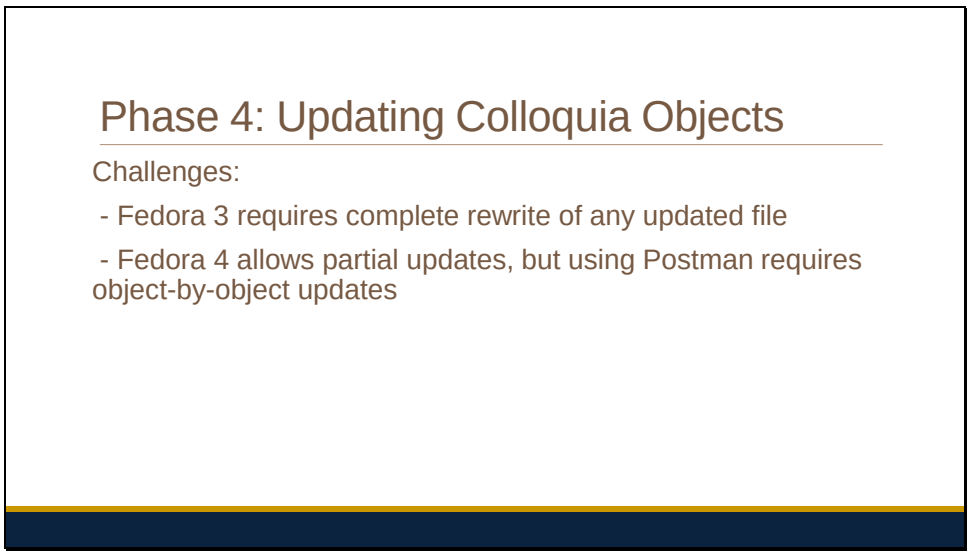
Then, using GREL, you can extract the cell reconciliation match id into your new column. The result is in the preview, a name in the original column and a second column with the URI.

Slide 26



Since I'll need the Author PID, I choose to do a quick find & replace on the column using the author PID prefix. I then exported it as a spreadsheet and removed the author name, colloquia title, and date, leaving only the Colloquia PID and the Author PID.

Slide 27

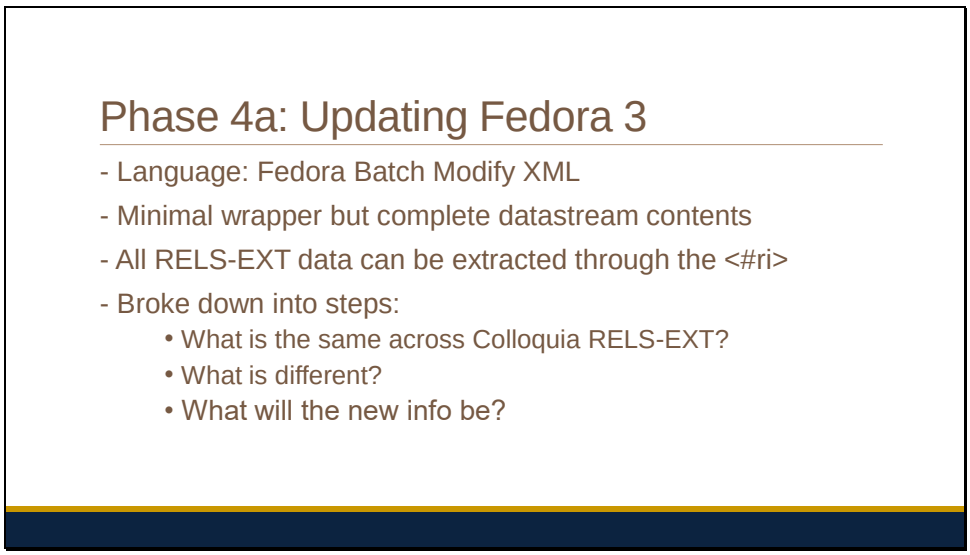


Getting data out proved easy enough. Reconciling data wasn't even that hard. Getting the data back in? This is where it gets tricky. Doing it manually might be a last resort for 250 objects, but for larger projects it would be prohibitive.

One of the first things I did after starting the project was look into how one could do batch updates in Fedora 3. I'd seen an option in the client, but had never experimented with the language. Fedora 3 uses Fedora Batch Modify files, which allow you to create, delete, or update any datastream within any object. The catch? When updating a datastream, you must replace the file with a new one. So if I wanted to add a new rel:isPartOf, I'd have to completely recreate the RELS-EXT datastream.

Since we were unsure about doing the project until after migration to Fedora 4 and I wanted to examine the differences between the two, I also looked into what it would take to update the objects in Fedora 4. Fortunately, the language it uses, SPARQL-update, is great for updating objects, but at first glance it looked like it would take even longer. The Postman interface I'd been using to send update queries would be *more* time consuming than a Fedora Batch Modify file because I'd have to do it one-by-one.

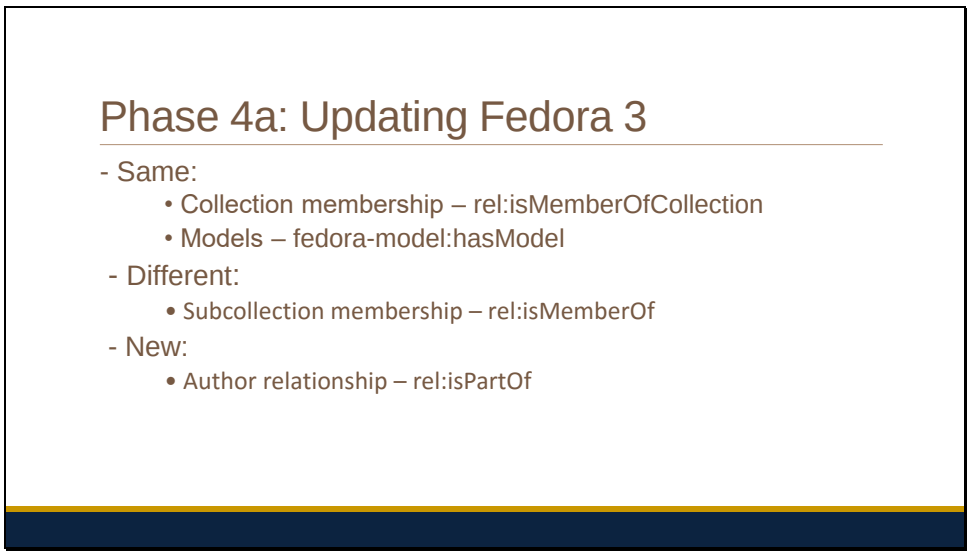
Slide 28



Let's look first at Fedora 3. What is Fedora Batch Modify language? It's a very minimal XML wrapper which includes PID, datastream ID or the official name of the datastream such as (RELS-EXT or DC for the Dublin Core datastream), and replacement datastream title enclosing a complete XML file.

So in order to do the update, I was going to have to recreate the RELS-EXT for every single colloquia I wanted to update. Since RELS-EXT data can all be extracted through the resource index, I broke it down into steps to determine what I'd need to recreate a file. What is the same across all Colloquia RELS-EXT files? What can just be part of my template? Next, what is different? What am I going to have to get out of the current system? And finally, what does the new info I'm adding look like?

Slide 29

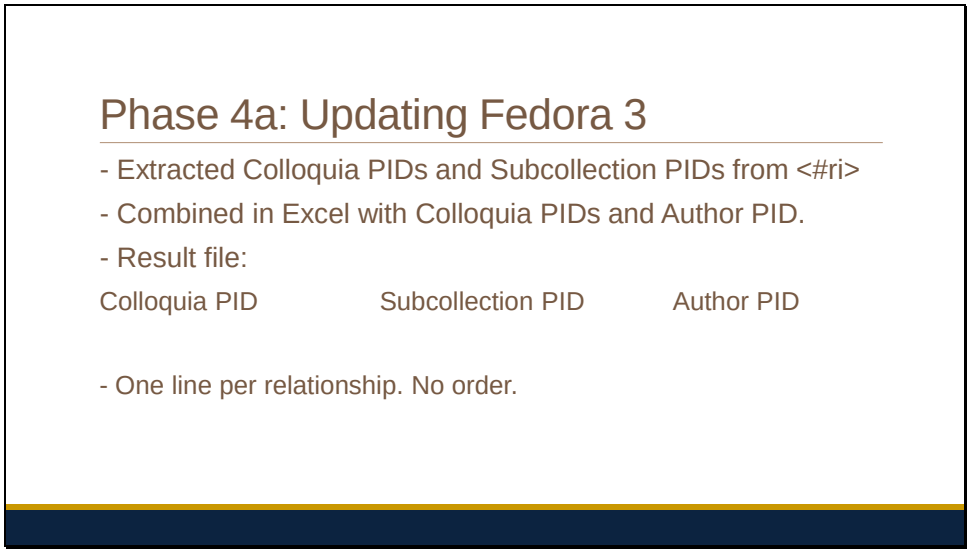


Fortunately, our RELS-EXT files defining relationships for colloquia were pretty straightforward. All colloquia belong to the colloquia collection. All use the same two Fedora models, which are also expressed via relationships.

What's different between colloquia is that we subdivide our colloquia by the hosting organization—engineering, science, systems, etc. Of course, some co-host, which means a colloquia may have several subcollection memberships. We use rel:isMemberOf here instead of isMemberOfCollection to avoid system confusion.

And finally, the new data is the author PIDs using the rel:isPartOf.

Slide 30

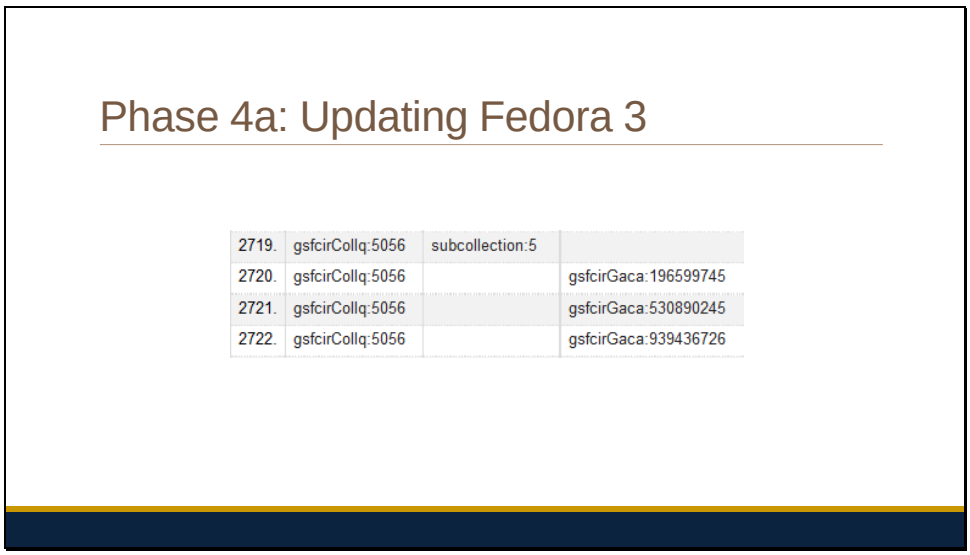


So, I extracted all Colloquia PIDs and their respective Subcollection PIDs from the resource index. I added a new column in the Excel file which already had Colloquia and Author PIDs (bear with me, I know it sounds a little odd) and put them in following the other pairs.

In the resulting file, each line had a single relationship, a Colloquia PID and either a Subcollection PID or an Author PID. A colloquia with 2 subcollections and 3 authors would have 5 rows. Most just had one row with a subcollection, since I'd exported subcollection info for every single colloquia as it was much faster than extracting based on several hundred PIDs. Obviously, it needed more work done to get it usable.

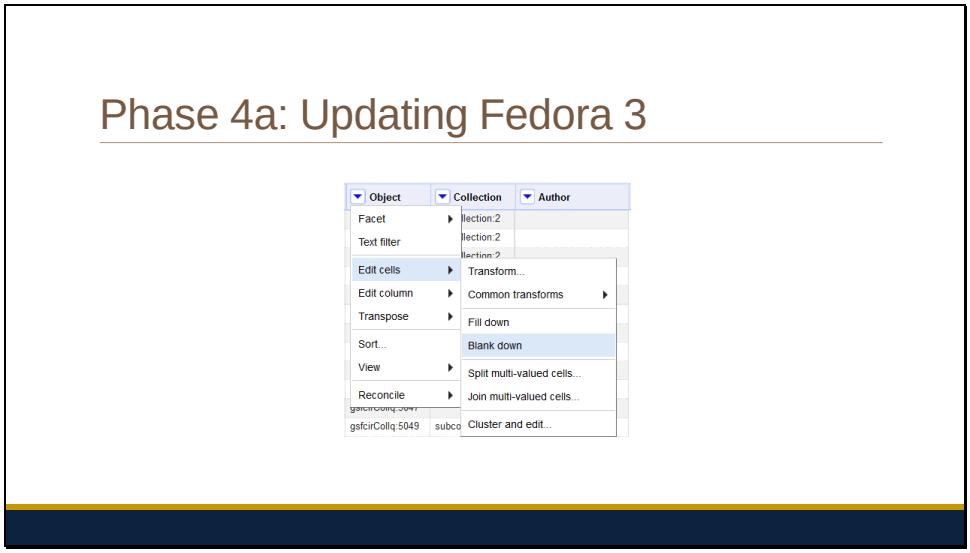
First, I ordered the rows on the Colloquia PID, so that the information for each object would be adjacent. This was critical for the next steps, after I imported it into OpenRefine.

Slide 31



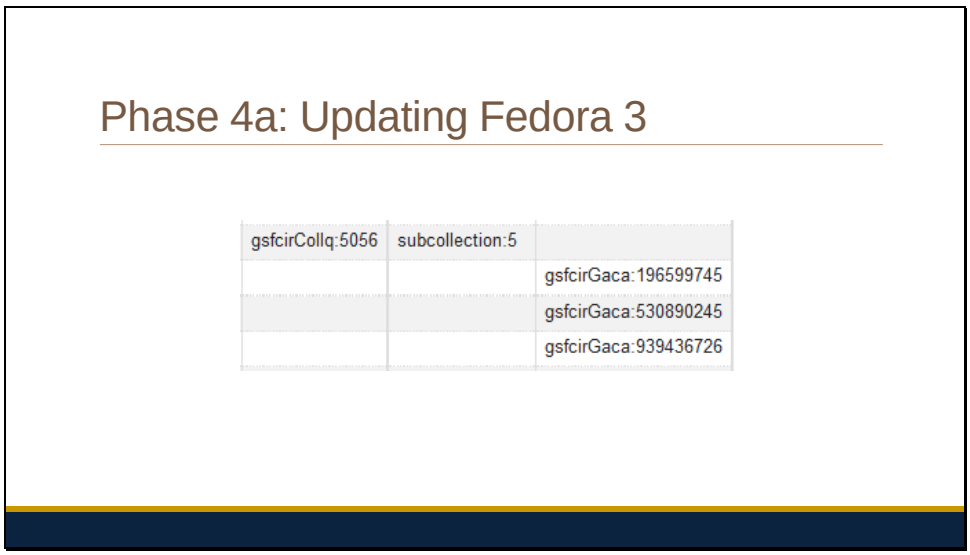
Here's an example of what the data looked like in OpenRefine for a colloquia with one subcollection and three authors. First column is ColloquiaPIDs, second is subcollections, third is authors. Looking back at this, I find myself shocked that I ever got it into the right XML format and uploadd.

Slide 32



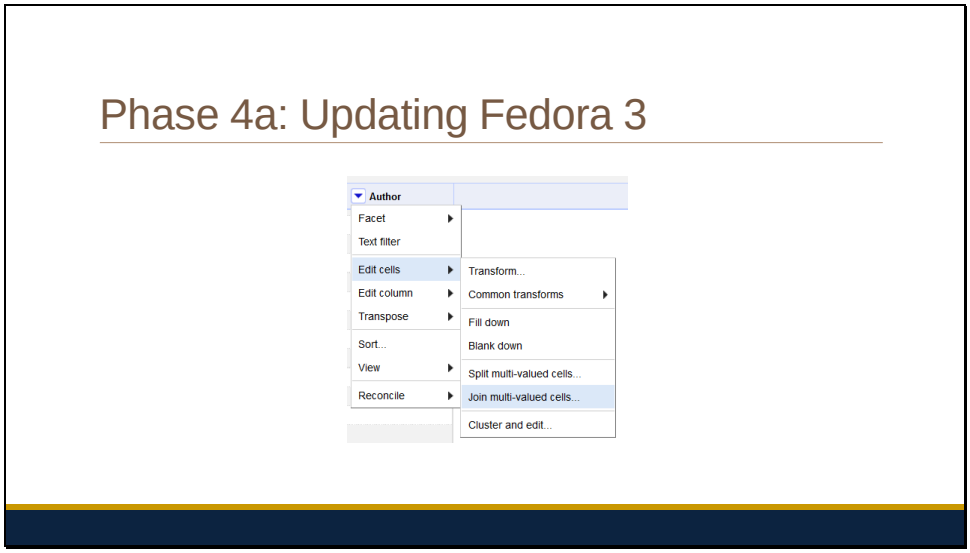
I used the “Blank Down” feature in OpenRefine, which removes duplicate data in *subsequent* (and only adjacent) rows…

Slide 33



To end up with this. Now OpenRefine allows you to toggle at the top from viewing this as 4 rows to viewing it as one “record.”

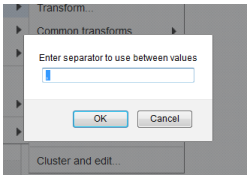
Slide 34



Since OpenRefine now views it as one record with several rows, I can then manipulate the data further. The join multi-valued cells function combines multiple rows in a record which all fall into the same column.

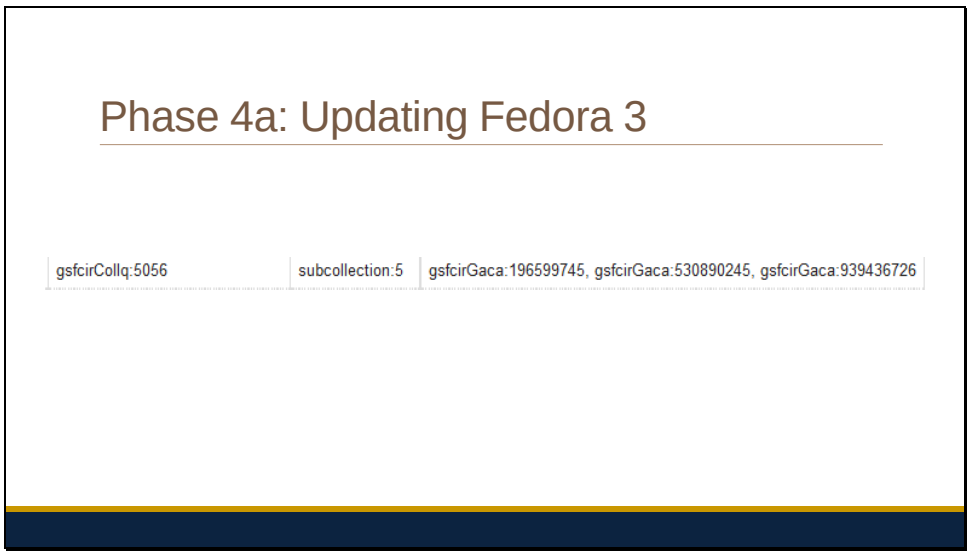
Slide 35

Phase 4a: Updating Fedora 3



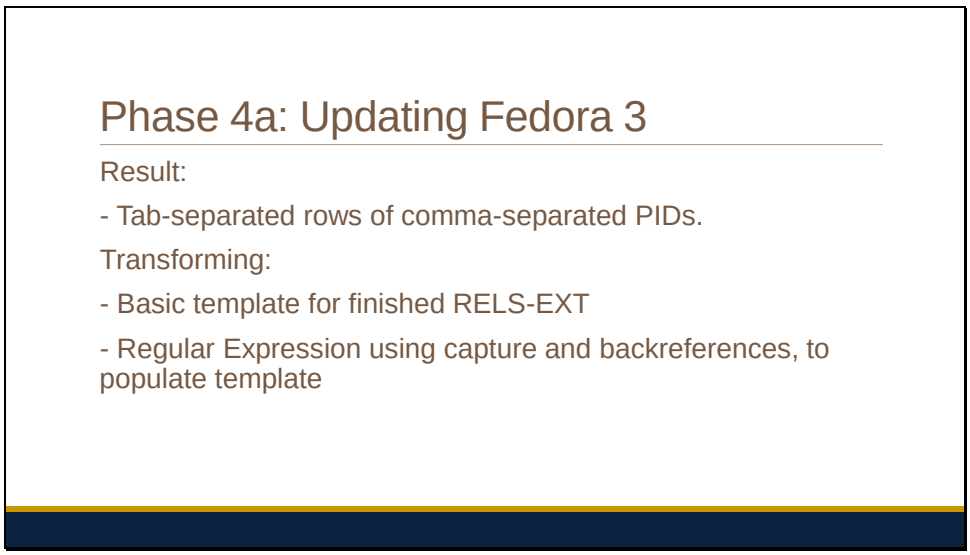
I just choose a separator, in this case the default comma OpenRefine suggests…

Slide 36



And here we go. I now have all my data on one line.

Slide 37

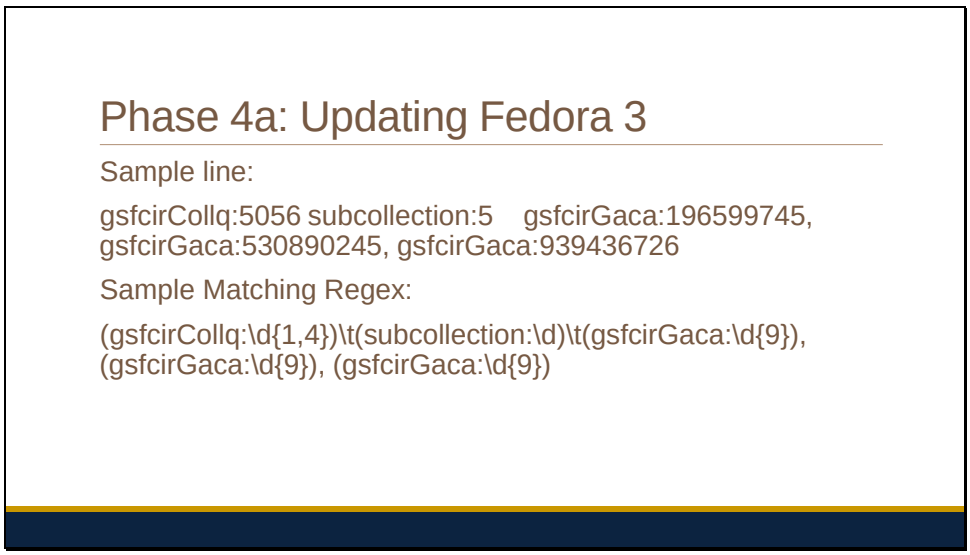


This resulting data is tab-separated with sibling PID-types comma-separated, which will matter for later manipulation with Regular expressions. I exported it back into Excel, where I sorted again and removed all rows which didn't have anything in the authors column.

Now I just needed to get it into RELS-EXT. My thoughts first went to scripting or even XSLT, but just as I found when creating the authors RDF file, I quickly realized that the simplest method would be to use regular expressions. I created a template for the RELS-EXT file with variations to allow for up to three subcollections and up to four authors (both the maximum number my visual scans had observed).

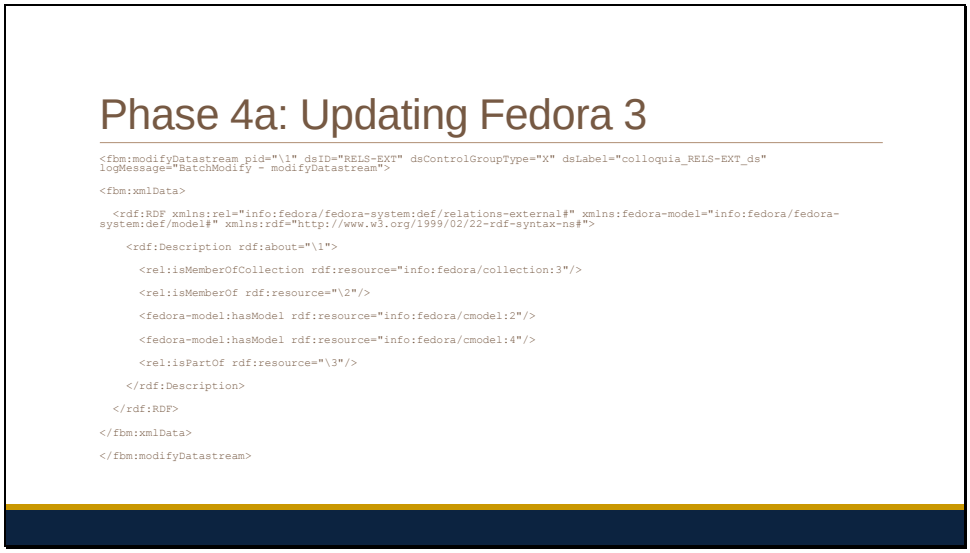
I then wrote the surprisingly-simple regular expression and variations thereof necessary to capture the PIDs and populate the template using backreferences.

Slide 38



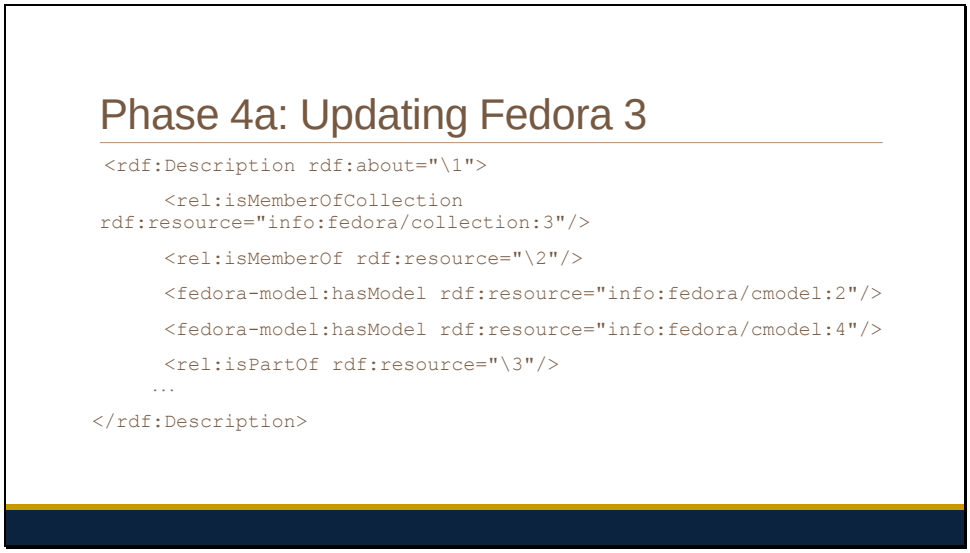
Here's the example from previous images, one Colloquia PID, one Subcollection PID, and three Author PIDs. This variation on the regular expression looks for this combo of PIDs and wraps each in parentheses to be returned later.

Slide 39



This is the overall file, but it's a bit hard to read on the screen, so let me pare down to the RDF Description section.

Slide 40

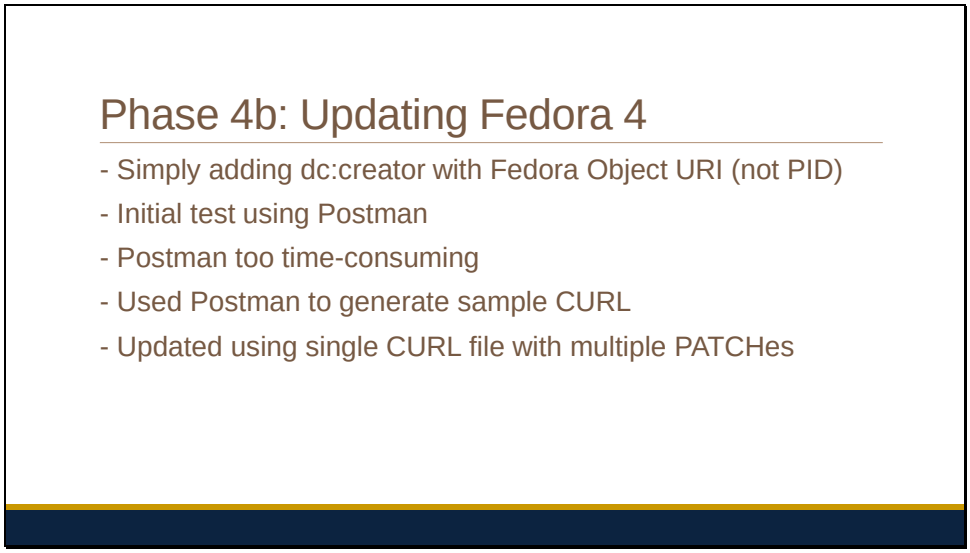


As you see, in rdf:about, I have a backreference, backslash 1. This refers to the Colloquia Object PID capture. I use that backreference in the larger file as well. Backreference 2 occurs within a rel:isMemberOf statement for the Subcollection PID and backreference 3 occurs within a rel:isPartOf statement for the first Author PID. Backreferences 4 and 5 would be duplicates of that isPartOf statement.

After I'd run through the variations, I added Fedora Batch Modify XML headers and footers and checked the document's XML validity to confirm that I hadn't missed a line of data. But my calculations had been correct and the document was good to go. I ran it against one of our sandboxes and then ran queries to confirm that it functioned as expected. The site didn't yet support display, but I could use the same queries that returned an author's publications to now return their colloquia as well.

On to Fedora 4!

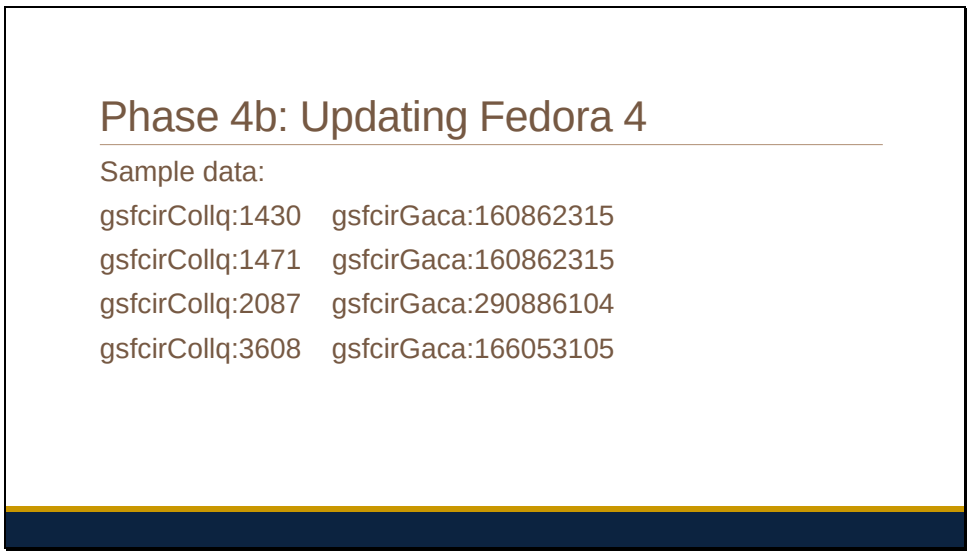
Slide 41



As I'd mentioned previously, what we needed to do in Fedora 4 would be far simpler. I simply needed to send a SPARQL-update with additional dc:creator information and the author's Fedora object URI.

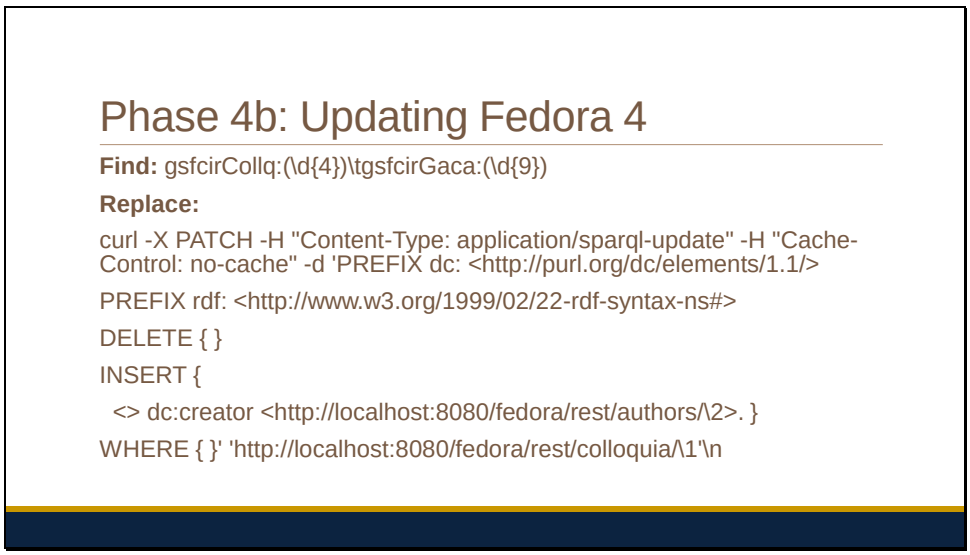
I did several updates using Postman, a service that allowed me to send update queries with complete success, but quickly realized it would be far too time-consuming. So I investigated alternatives. Postman helpfully let me export one of my queries in a syntax called CURL, which is a method of creating or updating certain kinds of resources, and I was soon up and running with the same Regular Expression method to batch-generate what are called PATCH updates in one giant CURL file. A PATCH update doesn't need to overwrite the whole thing, it essentially says "oh hey, add this piece of information to this other bigger piece of information." It's what a lot of database updating looks like, in fact, but not nearly as possible in XML.

Slide 42



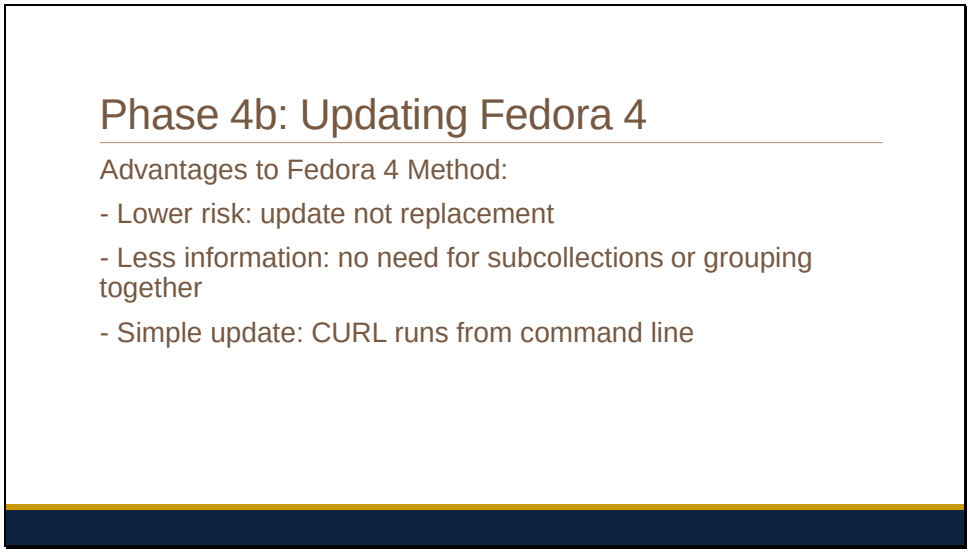
No subcollections here and no need to combine all updates for an object into a single statement. I could PATCH a colloquia as many times as I wanted, which made the process very simple. I took my original OpenRefine export…

Slide 43



…and ran this find and replace to generate the PATCH. You may notice that instead of using PIDs, I'm using the Fedora object URLs. We're still figuring out how we want to transition PIDs, but my current model has a similar structure to our front-end URL model. Thus, I captured the numeric portion of the PID and concatenated it with the collection's base internal URL.

Slide 44

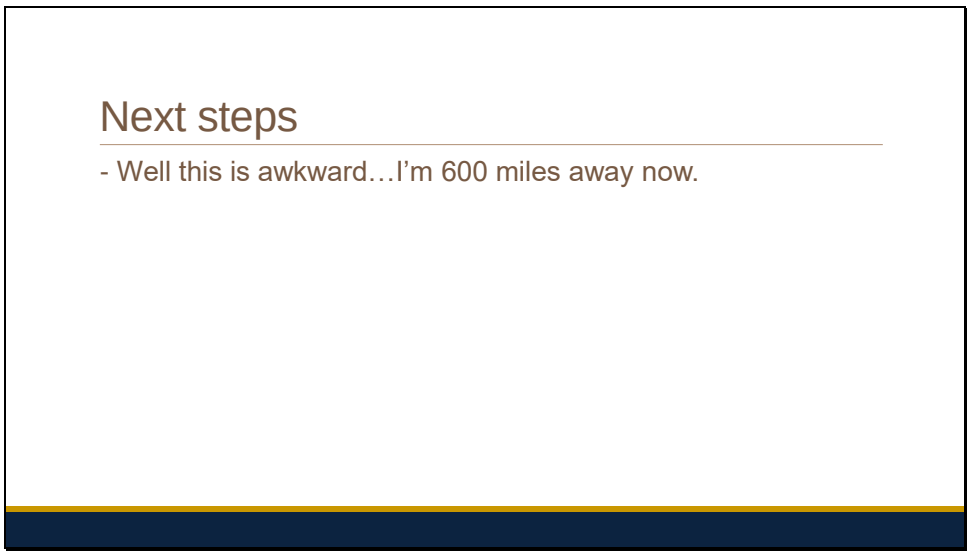


Unsurprisingly, I found the Fedora 4 method much simpler to work with, assuming we will be able to derive the URLs of our final repository structure from the current PIDs. It was much lower-risk and for a collection with a more complicated RELS-EXT, we would definitely have to wait until Fedora 4 to do this kind of project.

Because there was less information to update, there were far fewer steps.

And finally, CURLs are simply easier to run.

Slide 45



All in all, I was surprised by how little this project took—3 days total with less than half of each spent on the project. I did a great deal of intellectual groundwork beforehand by working with OpenRefine or regular expressions and thanks to Christina Harlow's excellent Code4Lib DMV workshop last summer which made me decide the project was feasible to begin with.

Our timeline on implementation was uncertain when I left, just as was our Fedora 4 migration timeline. While I could have added the data right away, our Drupal modules wouldn't search for and thus wouldn't display it. One major decision we'd needed to make is whether we were only going to record colloquia author information for authors who also have publications in our repository or whether we might add new records for Goddard employees who've never published but delivered colloquia.

Since I explored both Fedora 3 and Fedora 4 options, however, the data was ready for whenever we decide to move forward on implementation.

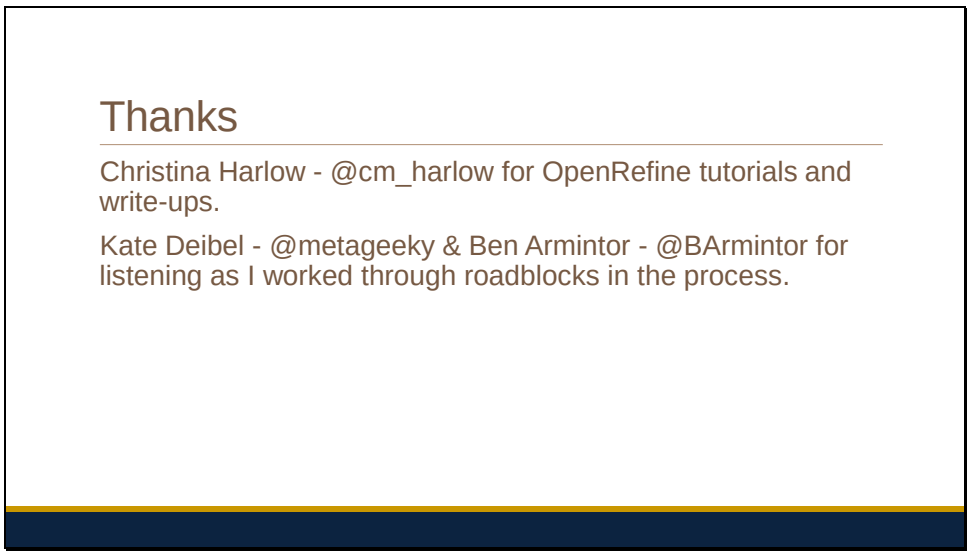
Slide 46

Read More About It

Tillman, Ruth K. "Extracting, Augmenting, and Updating Metadata in Fedora 3 and 4 Using a Local OpenRefine Reconciliation Service." Code4Lib Journal, no. 31 (January 28, 2016). Accessed May 9, 2016. http://journal.code4lib.org/articles/11179

Github: https://github.com/ruthtillman/local-reconciliation-project or http://bit.ly/1WkkDWq

Slide 47



I'd like to close with especial thanks to Christina Harlow, whom many of you may know, for her OpenRefine workshop and her tutorials. Without her work, this would've taken much longer and I wouldn't have had nearly such a good footing to start. I also greatly appreciate Kate Deibel and Ben Armintor, who helped me talk through a few issues I encountered.

Slide 48

Tools Used:

OpenRefine: https://github.com/OpenRefine or http://openrefine.org/

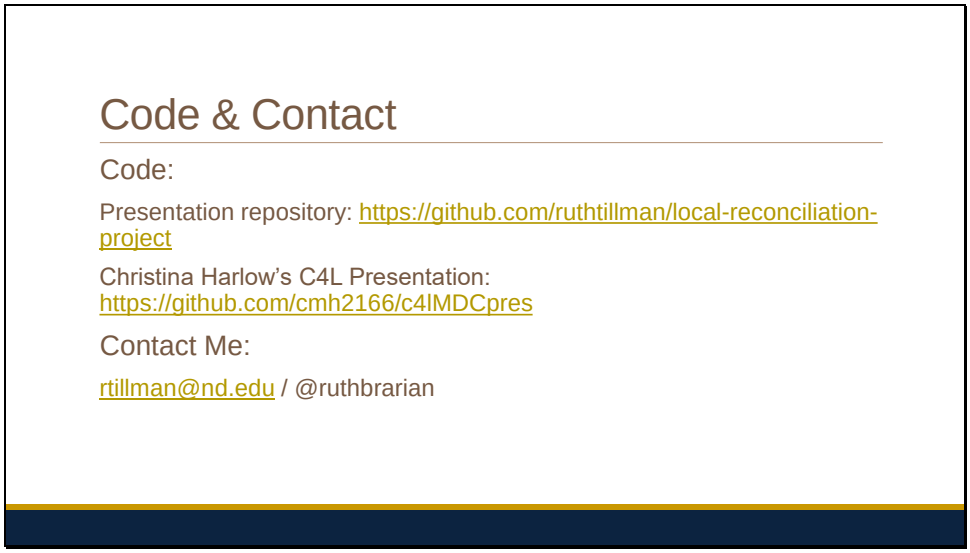
DERI RDF Refine plugin for OpenRefine: http://refine.deri.ie/

Sublime Text (trial version available): http://www.sublimetext.com/

cURL: http://curl.haxx.se/

Postman (Chrome only): https://www.getpostman.com/

Slide 49



I've put these slides and a selection of code snippets used in the project in a repository which you can visit here. I'm also happy to share that link afterward. You can contact me at rtillman@nd.edu

Slide 50

Attributions

Image 1: “Silos” by Anthony Arrigo, Flickr. https://www.flickr.com/photos/anthonyfarrigo/

Image 2/3: “Skybridge” by Billie Ward. Flickr. https://www.flickr.com/photos/wwward0/21180929605/