Creating metadata out of thin air and managing large batch imports

The University of Cincinnati Libraries processed a large (over 500,000 items) collection of birth and death records from the city of Cincinnati from 1865-1912, and successfully created dublin_core.xml submission packages from spreadsheets with minimal information, using batch methods to create a 524,360 record DSpace community, in the OhioLINK Digital Resource Commons (DRC). This repository may be the largest in the world, in terms of number of records if not size of digital objects, judging by the numbers from other repositories currently reported by the Registry of Open Access Repositories.[1]

This paper will discuss the methods and scripting used to create and manage the submission package, and consider what worked well, and what presented challenges.

We are currently processing an archive of over 37,000 letters and notes of the scientist and polio researcher Dr. Albert B. Sabin. Similar methods are being used, with the additional layer that we need to review about one fourth of the items for possible redaction of personal medical information, or classified (by the U.S. military) information. This paper will also discuss how an IP limited test environment was used to load, redact, export and reload the same materials, in an efficient manner.

The Hamilton County Births and Deaths collection can be found at http://drc.libraries.uc.edu/handle/2374.UC/2032 and is part of the OhioLINK Digital Resources Commons network of repositories, a series of separate virtual instances of the DSpace software for over thirty institutions from the state of Ohio (U.S.). This collection is one of the most heavily used resources in the UC Archives and Rare Books Library, and although genealogy researchers primarily use this resource today, we believe it has, especially in digital form, potential for epidemiology research as well.

The process for creating this collection involved applying for a grant, through the Library Services and Technology Program in the United States, and once the grant was awarded, we completed a formal Request for Proposal process to select a scanning vendor. Over 500,000 index cards in file cabinets were

[1] http://roar.eprints.org/cgi/roar_search/advanced?location_country=&software=dspace&type=&order=-recordcount%2F-date

scanned in a nine-month period by our chosen vendor. (The index cards had been transcribed in the early 20th century from now crumbling ledger books. The index cards are considered to be the official birth and death records.)

The originals were not amenable to Optical Character Recognition technology, as they were either handwritten, faded, with indistinct type, or a combination of the three. We knew, therefore, that hand keying of a tremendous amount of data would be required. When preparing the grant proposal, in order to estimate our costs, we had to make strategic decisions about which elements of the record to key into defined fields, leaving the remaining data to be keyed into a free form field. Figure 1 shows an example of a death record and Figure 2 the database entry hand-keyed by a vendor for this record. Figure 3 shows an example of a birth record, and Figure 4 shows the database entry for this record.

Figure 1 - death record:

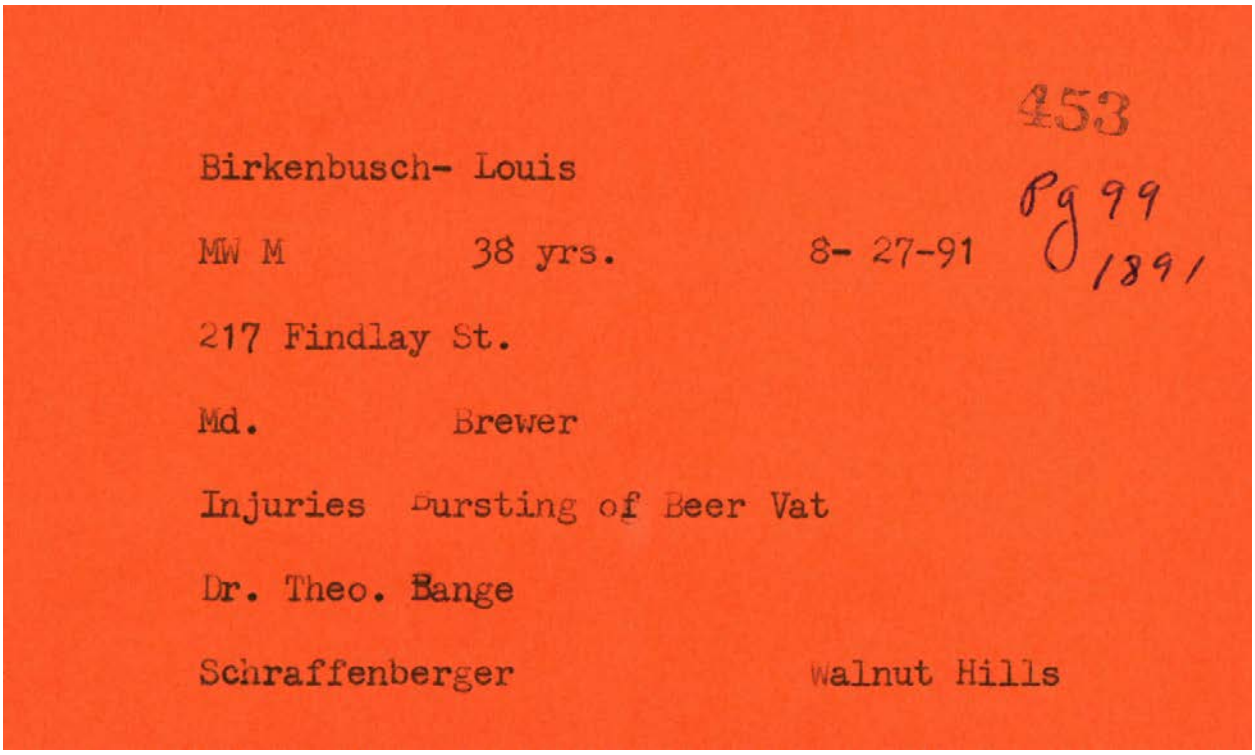


Figure 2 - Original Data Entry

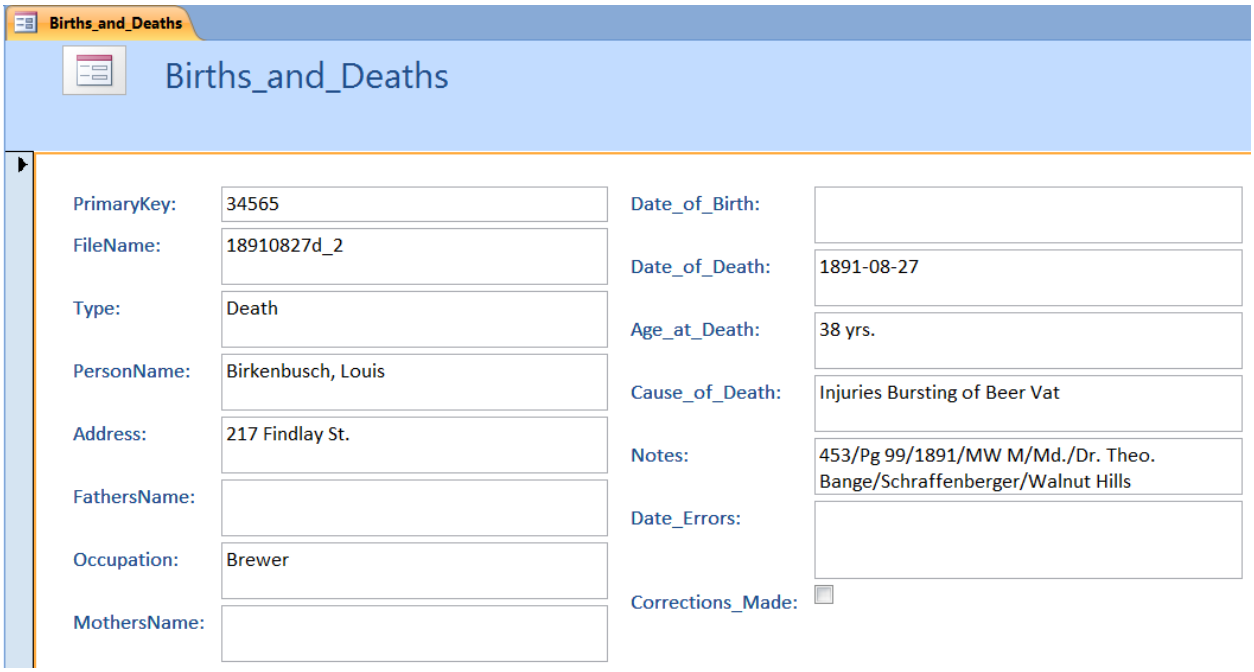


Figure 3 -- birth record:

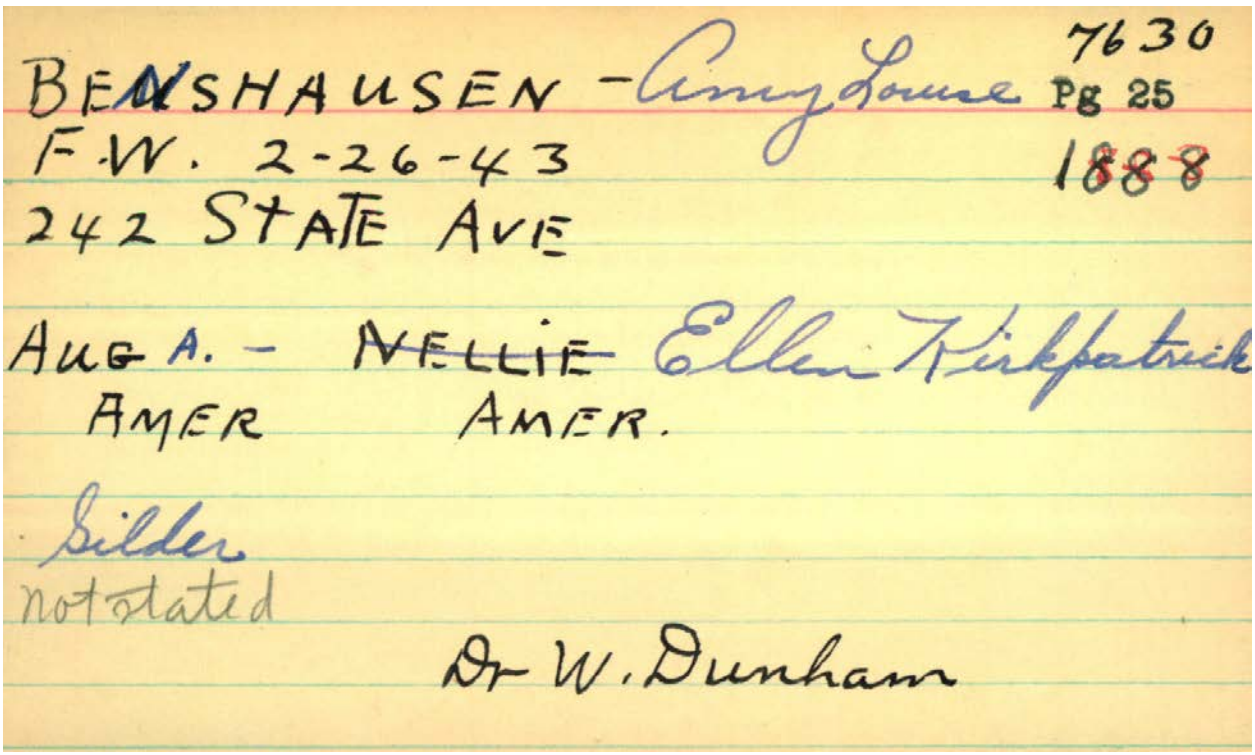
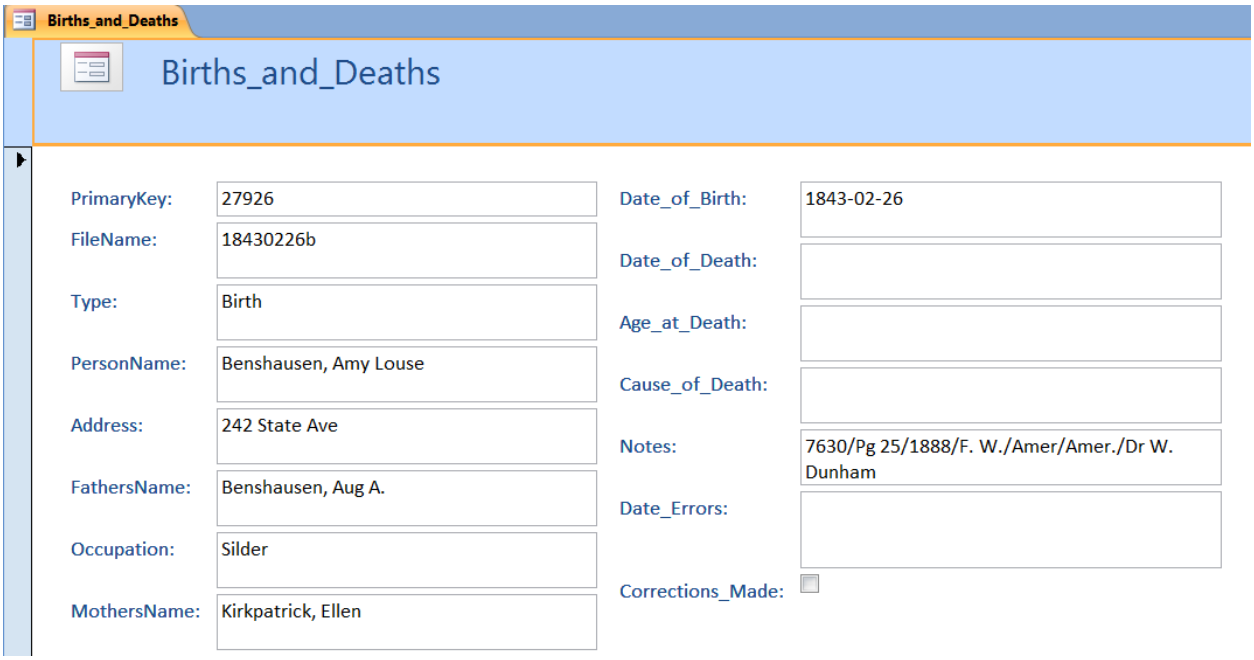


Figure 4 -- Original Data Entry



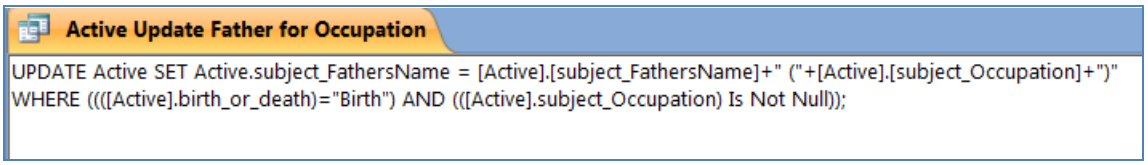
The requirement to decide what elements to capture as metadata, before digital objects exist, is not an infrequent component of digital collections work. Although the intellectual effort is similar to cataloging of published materials, as well as the need to make strategic decisions about the level of metadata and description based on cost, digital collections specialists often need to establish metadata standards before digital objects are created.

Once the grant was awarded, we embarked on a formal Request for Proposal, essentially a bid document, as required by our University and the State of Ohio. Thus by the time we began to work with the chosen vendor, more than a year had passed since first examining the materials. Extensive communications and testing took place with the vendor to make clear our scanning standards and data entry standards. One challenge was to avoid duplicate file names – filenames were created based on birth or death date, but the vendor would be encountering the records in alphabetical, surname order, not date order. The card files themselves were organized by surname, but we intended the records in the repository to have an order by date, and it was import to avoid duplicate file names. The vendors preferred platform for data entry was Microsoft Access – an SQL query was established that would discover duplicates in filenames, allowing the vendor to re-name files as they discovered duplicate filenames.

The vendor manually entered, using double-blind processes, the names of the persons born or deceased, relevant dates, addresses, parents' names (and occupations when stated), and cause of death, and placed all other uncategorized information in a descriptive field. This data was entered into a Microsoft Access database that was initially segmented by processing batch.

A combination of Structured Query Language and VBA scripting was used to transform the vendor's spreadsheet into dublin_core.xml files, with contents manifests, in order to achieve the structure of the Simple Archive Format expected by DSpace. The digital objects received from the vendor were in subdirectories that corresponded to the surname ranges of each file drawer. The digital objects with their submission packages were also re-organized by me into an alternate file structure that was sorted by date ranges instead of surname. These processes required attention to detail, knowledge of scripting languages, and ability to deal with batch commands with large numbers of records that were being physically re-located. Figure 5 shows an example of some of the SQL statements used to transform the metadata within Microsoft Access.

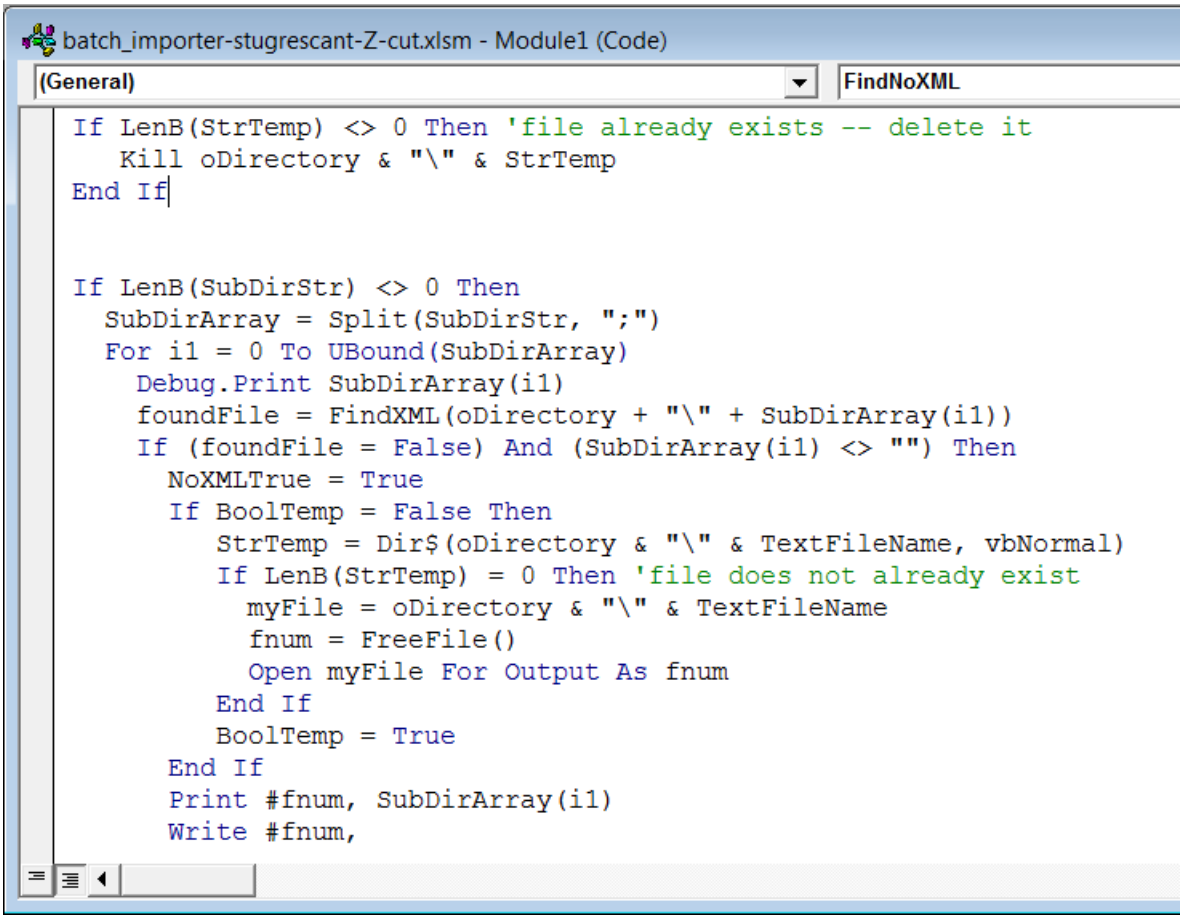
Figure 5 - SQL query used to add occupation to Father's Name.



Once the fields were structured according to our standards in Microsoft Access, with standard fields that applied to all records added, the collection was segmented into surname ranges, corresponding to the surname order in which the scanned digital objects were stored on our Storage Access Network. Each sub-set was exported to Excel, where we had a macro developed by OhioLINK that would convert the csv data into dublin_core.xml files, with contents manifests for each record. However, significant swaths of records were found where the filename in the database didn't match a filename on disk, and vice-versa. Often these records could be matched up, but this was tedious labor, and it was difficult to scan a directory of 50,000 or more digital objects (while processed, existing in a windows file structure) and find the specific sub-directories that did not contain a dublin_core.xml file. I used the VBA language to extend the excel macro to write to a report, rather than abnormally ending, whenever records existed in the data that could not be found. More challenging, however, was writing a program in the VBA scripting language to search directories for records that did not have a dublin_core.xml file after running

the macro – in other words physical digital objects, where there was no corresponding entry in our database. Writing a script to look for the absence rather than presence of a file proved more difficult, but this was accomplished. Figure 6 shows a segment of this code.

Figure 6 - example of code reading directories into an array and then looking to see if each directory contains a dublin_core.xml file



The entire excel macro can be found at the Wiki for the OhioLINK DRC project[2]. I also extended the macro to identify files as license files, archival masters or thumbnail images, when creating the contents manifest, if you can identify for the macro a consistent pattern in the file naming convention that you use.

Figure 7 shows the resultant dublin_core.xml file for the death record shown above:

[2] https://sites.google.com/a/ohiolink.edu/drmc/bulk-submission/bulk-submission----alternate-e

Figure 7 -- dublin_core.xml record after sql and vba transformations:

dublin_core.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<dublin_core>
  <dcvalue element="subject" qualifier="none">Birkenbusch, Louis</dcvalue>
  <dcvalue element="title" qualifier="none">Birkenbusch, Louis (Death, 1891-08-27)</dcvalue>
  <dcvalue element="description" qualifier="none">Address: 217 Findlay St.</dcvalue>
  <dcvalue element="subject" qualifier="none">Occupation -- Brewer</dcvalue>
  <dcvalue element="date" qualifier="created">1891-08-27</dcvalue>
  <dcvalue element="coverage" qualifier="temporal">1891</dcvalue>
  <dcvalue element="coverage" qualifier="spatial">Cincinnati (Ohio)</dcvalue>
  <dcvalue element="date" qualifier="issued">1891-08-27</dcvalue>
  <dcvalue element="description" qualifier="none">Age at death: 38 yrs.</dcvalue>
  <dcvalue element="subject" qualifier="none">Cause of death -- Injuries Bursting of Beer Vat</dcvalue>
  <dcvalue element="description" qualifier="none">453/Pg 99/1891/MW M/Md./Dr. Theo. Bange/Schraffenberger/Walnut Hills</dcvalue>
  <dcvalue element="description" qualifier="notes">Original record filed in drawer labeled &amp;#039;BIRD-BLACKNER&amp;#039;.</dcvalue>
  <dcvalue element="type" qualifier="none">Image</dcvalue>
  <dcvalue element="publisher" qualifier="OLinstitution">University of Cincinnati</dcvalue>
  <dcvalue element="publisher" qualifier="OLrepository">University of Cincinnati. Archives and Rare Books Library</dcvalue>
  <dcvalue element="publisher" qualifier="digital">University of Cincinnati. University of Cincinnati Libraries</dcvalue>
  <dcvalue element="language" qualifier="iso">en_US</dcvalue>
  <dcvalue element="contributor" qualifier="author">Cincinnati (Ohio). Health Dept.</dcvalue>
  <dcvalue element="format" qualifier="medium">paper</dcvalue>
  <dcvalue element="format" qualifier="mimetype">image/jpeg</dcvalue>
  <dcvalue element="rights" qualifier="uri">http://drc.libraries.uc.edu/fairuse.html</dcvalue>
  <dcvalue element="relation" qualifier="ispartof">Cincinnati Birth and Death Records, 1865-1912</dcvalue>
  <dcvalue element="date" qualifier="digitized">2010-02-17</dcvalue>
  <dcvalue element="identifier" qualifier="other">34565 (File Order Number)</dcvalue>
 </dublin_core>
```

The dublin_core.xml packages were re-organized into sets that corresponded to ten collections, which represented ten different time periods (for example, 1865-1875; 1876-1879, etc.), with the hope that this sub-division into 10 collections would facilitate browsing and discovery in the DSpace environment.

During the first quarter of 2011, each submission package was built initially with 5,000 records, and then gradually increased up to 83429 records in one batch load. I used a Secure File Transfer Protocol (sFTP) process to load the records from computers at the University of Cincinnati Libraries to our test DSpace instance at OhioLINK, and I ran the import command to load these batches of records into our DSpace collections within that test instance. (The test instance is a clone of our production instance, but is IP limited to participants in the OhioLINK system.) After I verified that each set of records had loaded without errors, I notified OhioLINK staff that the same submission package could be imported into the parallel collection on our production DSpace instance.

Our intention was to keep the test system well ahead of the production system, so that if we experienced significant response time or access issues as the size of the repository grew, we would discover this in the test system before the production system became similarly hampered. A Google docs spreadsheet was used to keep track of which records had been processed, the directory structure where the submission packages were stored, and to inform the staff at OhioLINK which records were ready to be processed. (See Figure 8 for an example of this spreadsheet.)

Figure 8 -- Google Docs spreadsheet used to track load progress

Birth&DeathLoad

File Edit View Insert Format Data Tools Help Last edit was 346 days ago

Batch:

	A	B	C	D	E	F	G	H	I
1	Batch:	Source directory on test:	#records ftp'd to drcuc-test:	#records loaded in drcuc-test:	drcuc-test load status:	drc production load status:	#records in ftp'd to production:	#records loaded in production:	notes
2	1	/local/local/dspace/upload /births_and_deaths/batch-1_Images_1 /A-BACHMANN/	12463	12463	Loaded 2010-12-30	Loaded 2011-01-04	12463	12463	Adams-BACHMANN required reloading as source records were lost on original drcuc-test. Reload was complete 2011-07-25
3	2	/local/local/dspace/upload /births_and_deaths/batch-1_Images_1 /BACHMANN-BELS/	14125	14125	Loaded 2011-01-19	Complete 2011-01-27	14125	14125	
4	3	/local/local/dspace/upload /births_and_deaths/batch-1_Images_1 /BEL-BROE/	25974	25974	Loaded 2011-01-21	Complete 2011-02-04	25974	25974	
5	4	/local/local/dspace/upload /births_and_deaths/batch-1_Images_1 /BROE-CARTER/	17652	17652	Loaded 2011-01-26	Records ftp'd to production source directory; load in process 2011-02-07	17652	17652	
6	5	/local/local/dspace/upload /births_and_deaths/CARTER-DEW	29011	29011	Load complete 2011-02-08	Complete	29011	29011	per email from In, some original source records replaced and should be re-copied before production load
7	6	/local/local/dspace/upload /births_and_deaths/DEW-FILL	32911	32911	load complete 2011-02-12	Complete 2011-02-28	32911	32911	per email from In, some original source records replaced and should be re-copied before production load

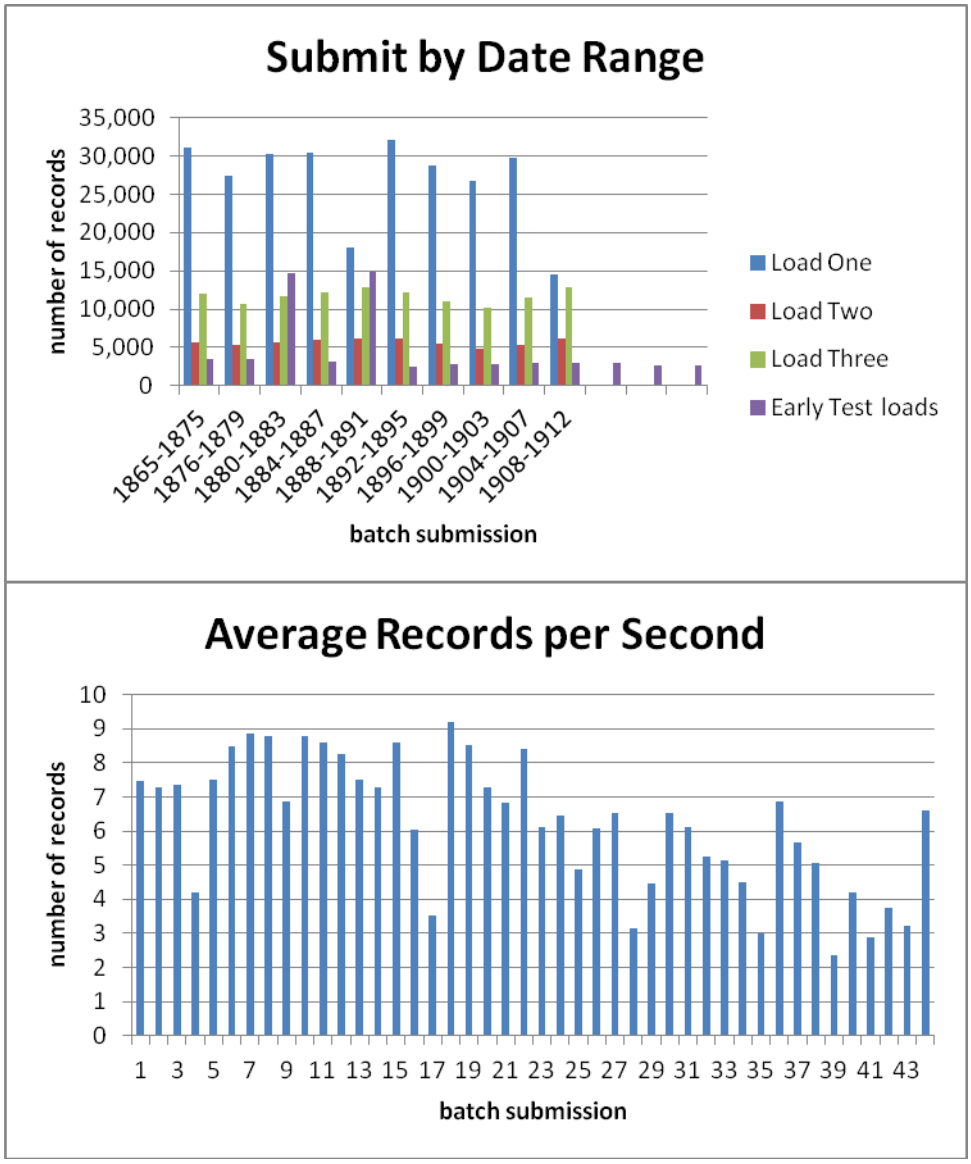
During this time period we were loading records on a DSpace 1.6.2 platform, with a PostGreSQL database back-end, for both our test and production systems, adding these records to an existing set of collections. However, record loads starting taking significantly longer per record – an undesirable effect of an import program for 1.6.2 that 'prunes' the indexes after each record is loaded[3]. Although prepared for the ever-increasing load times, after we crossed the 200,000 records threshold, the test system GUI began to perform very badly. And before we had determined a solution, the production system, now at 132,136 records for this collection alone, also started behaving very badly, with all queries against all collections performing significantly more slowly, to the point they reached unacceptable levels, shortly followed by both test and production systems failing to restart and to come online at all. This was the very result we had hoped to avoid by loading records first on a test platform before duplicating the same quantity of records on our production platform.

[3]DSpace scalability testing by Tom De Mulder and others uncovered this problem with DSpace 1.5 and 1.6 in 2010. "For any batch of size n, where n > 1, this is (n - 1) times more than is necessary," from http://dspace.2283337.n4.nabble.com/DSJ-Created-DS-470-Batch-import-times-increase-drastically-as-repository-size-increases-patch-to-mitm-td3291379.html. See also http://tdm27.wordpress.com/2010/01/19/dspace-1-6-scalability-testing/.

After several attempts to tweak PostGreSQL parameters and DSpace parameters, and to add memory to our servers, OhioLINK developers built a new DSpace 1.7.1 instance on new hardware on an Oracle, instead of PostGreSQL, back-end, and we were able to successfully load all 500,000+ records of this collection onto that platform. OhioLINK developers were also able to rebuild a new 1.6.2 test instance, still on PostGreSQL, with the 132,136 records from the previous production system, and a 3rd UC instance that was also 1.6.2 and contained all other collections. OhioLINK developers concluded that both previous DSpace test and production systems (and test had been originally cloned from production) had suffered from the same file corruption that had intensified and compounded as our instances grew. They concluded that PostGreSQL was not the issue per se. However, we had better diagnostic tools with Oracle, because by now we were working with another Ohio government entity (OARnet) for IT support and they were an Oracle shop. This made it clear that Oracle was our preferred platform. This file corruption was mysterious and unsettling, as it seemed it could happen again – thus the importance of having diagnostic tools to more easily discover it.

The very good news was that for our other collections already built, an export and re-import process did not replicate that file corruption. And, for the over half a million record collection, the import program in DSpace 1.7.1 had been optimized to avoid the sequential indexing issue, and we were able to reload all records from the same submission packages that I originally uploaded to OhioLINK, in three days, getting us to an instance in DSpace 1.7.1 that was over 500,000 records at the end of April 2011. Record loads in DSpace 1.7.1 averaged a blinding 6 per second. (See figure below of data provided by John Davison, Assistant Director of Library Systems for DRC Development at OARnet.)

Figure 9 Submission Data in DSpace 1.7.1 - April 28, 2011



In sum, it took three months of struggling with the import process, and the system performance issues we were facing in DSpace 1.6.2 on PostGreSQL, to conclude that we absolutely had to migrate to the next version of DSpace and an Oracle back end, on new disk hardware -- once we did so, we finished the bulk of the load remarkably quickly.

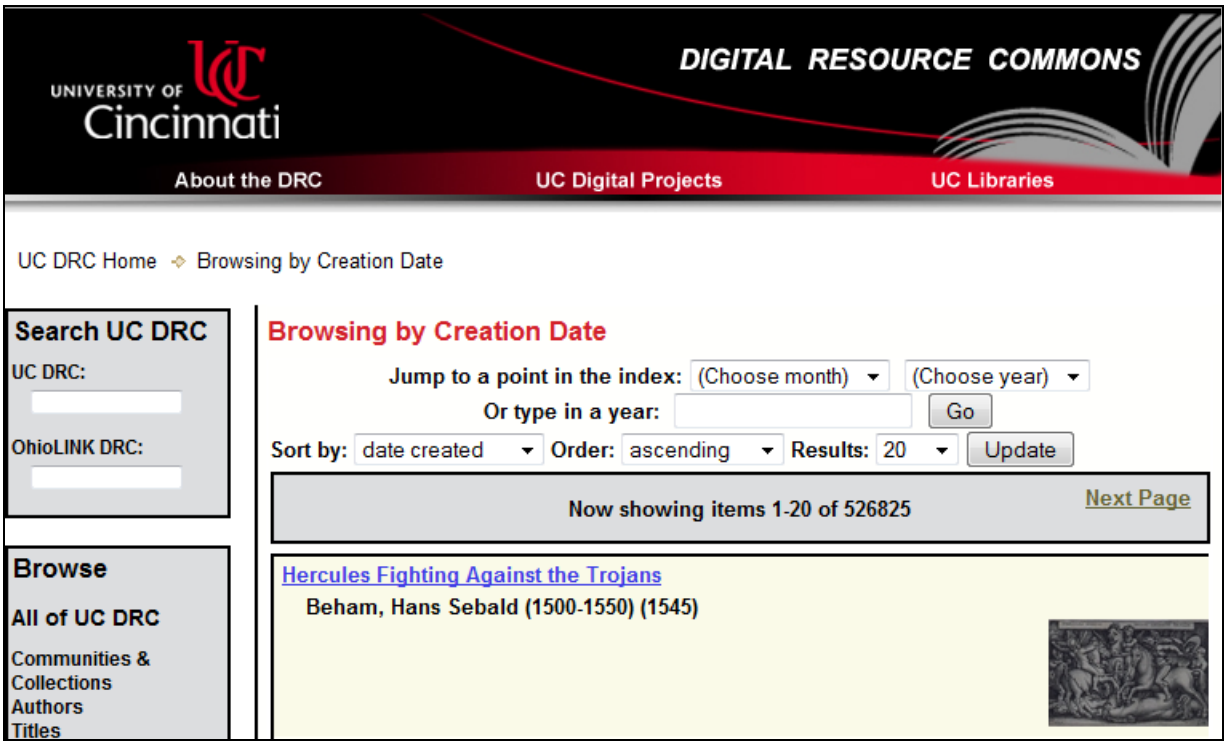
It wasn't until early August of 2011 that we had loaded all remaining records, and made the determination to keep these records in a DSpace 1.7.1 instance separate from our other collections for an interim period of time, and it was at that time, eight months from beginning the data load, on August 31st, 2012, that the complete collection was announced to the public, announcing the new domain name

for the separate DSpace instance for this collection. (The public had seen the records we had loaded earlier into production; a web page was created that announced our progress to the public over these eight months, but now users had to be directed from the partial collection on the system with the original domain name, which was the system that still supported our other collections.)

Thus by August 2011, we had two production instances – one was over 500,000 records and running 1.7.1 on top of Oracle; one was much smaller and running 1.6.2 on PostGreSQL, and in addition our test instance was over 137,000 records, and also running 1.6.2 on PostGreSLQ. All three systems were performing well – the occasional need to restart Tomcat would present itself, but system availability and response time stayed good in general. However, our half a million record collection existed in a separate DSpace instance from the 1.6.2 instance that still supported the other collections previously built. Furthermore, in the 1.7.1 instance, we did not have 'handles' or permanent record URI's, because we intended to merge these two DSpace platforms, creating one handle sequence. Furthermore our users were somewhat confused by the existence of two UC DRCs – some continued to look for Birth and Death records on the original system.

I'm happy to add that we began a merger project in May of 2012, and as of June 25, 2012, we have been able to merge all collections onto one platform – now DSpace 1.8.2 on Oracle, with one handle sequence. We have also upgraded our test platform to 1.8.2 on Oracle. Our other collections were successfully exported and re-imported to the new platform, maintaining all handles. The Birth and Death records were re-loaded from the original submission packages, the second or third such load for all of these records, in a process that this time took only a matter of days. The production University of Cincinnati DRC (http://drc.libraries.uc.edu) as of July 5, 2012, contains 526,825 records. (See Figure 10.) The test University of Cincinnati DRC as of this same date contained 536,976 records as our Sabin project (see below) was now well underway.

Figure 10 - Record total as of 7/5/2012 in drc.libraries.uc.edu



Response time has been very good in the 1.8.2 environment. The community/collection hierarchy of DSpace makes it possible for users who do not want to see results from this large collection to ignore it for their browses and searches. But as we add other historical records, such as for wills and morgue records, researchers will also be able to choose to search across all of these similar collections.

Our Albert B. Sabin collection has more recently presented some of the same challenges. We again outsourced the scanning processes and asked the vendor to create a spreadsheet with minimal metadata. (We provided the initial spreadsheet with entries from our Finding Aid for Folder Names.) For example, for a letter we requested that the vendor key the name of the author, the recipient, and the date of the letter. We used SQL and VBA to transform this minimal metadata into a complete dublin_core.xml record. (See figure 11 for an example of an SQL update query, in this case catching examples of single initials in names that the vendor failed to properly terminate with a period!) (See figures 12 and 13 for examples of the metadata as received from the vendor, and an example as transformed into a dublin_core.xml file.) I am again using the DSpace import command to import these records, still in the DSpace Simple Archive Format, to our test DSpace instance.

Figure 11 - Example SQL update query

02 - Fix Initials Query-SubjectRecipient

```
UPDATE SourceTable SET SourceTable.SubjectRecipient = [SourceTable].[SubjectRecipient]+"."
WHERE (((StrComp(UCase(Right([SourceTable].SubjectRecipient,1)),Right([SourceTable].SubjectRecipient,1),0)=0)=True) And ((Right([SourceTable].SubjectRecipient,1))<>".") And
((IsNumeric(Right([SourceTable].SubjectRecipient,1)))=False));
```

Figure 12 - Data provided by Scanning Vendor

B2 'Correspondence, General

	B	C	D	E	F	G	H	I	J	K
1	Box Title	Folder Number	Folder Title	Directory Name	File Name	File Type	Scan Date	Author	Recipient	Date
2	Correspondence, (	2	B Virus -- 1955-58	bvirus_1955-58	bvirus_1955-58_0(	letter	2010/11/23	Spaar, F W	Sabin, Albert B. (A	1955/09/05

Figure 13 - Resultant Dublin_Core.xml record

dublin_core.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<dublin_core>
  <dcvalue element="type" qualifier="none">letter</dcvalue>
  <dcvalue element="date" qualifier="digitized">2010-11-23</dcvalue>
  <dcvalue element="contributor" qualifier="author">Spaar, F. W.</dcvalue>
  <dcvalue element="subject" qualifier="none">Sabin, Albert B. (Albert Bruce), 1906-1993 -- Correspondence</dcvalue>
  <dcvalue element="date" qualifier="issued">1955-09-05</dcvalue>
  <dcvalue element="title" qualifier="none">B Virus -- 1955-58 -- Correspondence, General -- letter, 1955-09-05</dcvalue>
  <dcvalue element="type" qualifier="none">text</dcvalue>
  <dcvalue element="subject" qualifier="none">Spaar, F. W. -- Correspondence</dcvalue>
  <dcvalue element="date" qualifier="created">1955-09-05</dcvalue>
  <dcvalue element="coverage" qualifier="temporal">1955</dcvalue>
  <dcvalue element="description" qualifier="none">Letter from Spaar, F. W. to Sabin, Albert B. dated 1955-09-05.</dcvalue>
  <dcvalue element="description" qualifier="none">&lt;a href=&quot;http://digitalprojects.libraries.uc.edu/sabin/fairuse/&quot;&gt;Sabin Collection Fair Use Policy</dcvalue>
  <dcvalue element="language" qualifier="iso">en_US</dcvalue>
  <dcvalue element="relation" qualifier="ispartofseries">Sabin Archives. Correspondence. General. Box 01.  File 02 (B Virus -- 1955-58)</dcvalue>
  <dcvalue element="publisher" qualifier="digital">University of Cincinnati. University of Cincinnati Libraries</dcvalue>
  <dcvalue element="publisher" qualifier="OLrepository">University of Cincinnati. Hauck Center for the Albert B. Sabin Archives</dcvalue>
  <dcvalue element="publisher" qualifier="OLinstitution">University of Cincinnati</dcvalue>
  <dcvalue element="rights" qualifier="uri">http://digitalprojects.libraries.uc.edu/sabin/fairuse/</dcvalue>
  <dcvalue element="relation" qualifier="ispartof">The Albert B. Sabin Archives</dcvalue>
  <dcvalue element="format" qualifier="mimetype">application/pdf</dcvalue>
  <dcvalue element="format" qualifier="medium">paper</dcvalue>
</dublin_core>
```

In the DSpace 1.6.2 environment load times were increasing exponentially as the size of our database grew. But in the 1.8.2 test environment, where we now have 536,976 records, I just batch imported (to our test platform) 1712 records with 64 GB of data and the load times averaged .6 seconds per record – not quite the same blinding speed reached earlier, but still very acceptable.

Approximately one-fourth of the Sabin Archive requires human review for content that could violate the privacy of medical information, and the collection also contains documents that once were classified by the military and now are not, but require careful examination to be certain of that last conclusion. An archivist is funded by the grant (in this case from the National Endowment for the Humanities). The archivist became familiar with the collection and was able to identify by file and folder organization which documents required this review. I segregated those records and loaded them into our test platform, again using batch import, so that the archivist could complete this manual review.

The archivist reviews the letters for personal health information, uses either Acrobat X or Photoshop to line through the sensitive information, rebuilds the PDF (taking care that the underlying OCR'd layer is

changed by the redaction), and in some cases redacts metadata as well. The DSpace record is exported, and the revised PDF and xml file are uploaded to our digital projects Storage Access network, from where I rebuild the submission package, and re-import these records.

The dublin_core.xml files that the archivist has exported from the edited records contain fields such as accession date that we wish to remove from the submission package before reloading these records. I found the dspace_migrate shell script that is part of the DSpace install, but it would have been onerous to be required to download these records from our SAN to my workstation and then up to the OhioLINK DSpace test server (which would not accept Linux to Linux FTP or SCP). Happily I found that I could run this script outside of the DSpace environment on our local (LAMP) server, so that the records didn't have to be moved until they were ready to be sent again to the OhioLINK server. [4]

After re-building the submission package, the records are again loaded to test, and the Sabin archivist rechecks the collection, at which point we notify OhioLINK staff that the second submission package can now be imported into our production system.

In the 1.6.2 Test environment our archivist did not have the ability to upload archival masters to the individual record. When a letter has been redacted, we wanted the un-redacted letter to be stored with the record, but only available to authorized individuals. This was one reason why the redacted collections needed to have the submission packages rebuilt, rather than a direct export of the collection in our test environment to the production environment. However, now that we are on DSpace 1.8, our archivist can upload those archival masters in the GUI environment, and I anticipate that for the second half of our collection processing, we will do this direct export and import, perhaps using the AIP export and import processes that are new to 1.8.

Although multi-stepped, these procedures are working and allow us to build this archive with a level of review that is satisfying our university's legal counsel.

The Sabin archivist also contacted the U.S. National Archives and Records Administration's Information Security Oversight Office (ISOO) for advice on the best method of handling those letters and papers that during Dr. Sabin's military career had been stamped as either "Restricted" or "Confidential" by the U.S.

[4] I found that regardless of server environment where the migrate shell script was executed, it could not delete our handle fields because they contained language codes. Using regular expressions, I was able to extend this part of the dspace_migrate shell script from 1.6.2 and solve this problem.

military. We discovered that documents marked "restricted" were the lowest level of classified national security information, a classification that had not been used since the 1950's, and were not longer considered classified. For documents marked "confidential," we have a little more work to do, determining what the subject of the document is. Since much of the correspondence relates to Dr. Sabin's work while serving in the U.S. Army Medical Corps, including epidemiology rates in the military during World War II and disease and vaccine research, it was not considered protected as classified national security information. However, when we come across other materials marked "confidential" with different subject matters, we contacted the ISOO to make sure the documents were now unclassified before publishing them in the repository. To indicate that a document is no longer considered classified, the Sabin Archivist strikes through the classification stamp (using Adobe Photoshop) and adds a description note in the item's metadata designating it as unclassified. The subsequent exporting and re-processing of the records then proceeds as per our redacted records.

Each record in this collection is linked to a policy statement about Fair Use, and where applicable, a policy statement about redaction.

DSpace has proven to be capable of serving large numbers of records with associated indices. Processing large record sets is not easy, but is achievable with a replicable set of procedures and methods, and the willingness to use scripting languages to manipulate large sets of data. In addition to the ability to manipulate the metadata and create the correct submission packages, a significant amount of disk storage is required to manipulate and stage large datasets. (Our current local Storage Area Network was just almost doubled in size to 36 Terabytes, space we will use.) At times I think my job title should be changed to 'Chief Data Wrangler', but for now Digital Projects Coordinator will still suffice.

Linda Newman
Digital Projects Coordinator
University of Cincinnati Libraries
Cincinnati, Ohio (United States) 45221-0033
Telephone: 523-556-1555
Email: Linda.Newman@uc.edu
http://digitalprojects.libraries.uc.edu
http://drc.libraries.uc.edu