Penn Libraries

Innovations with Solr at Penn: Novel methods of incorporating multiple record sets and cross-referencing headings

Michael Gibney

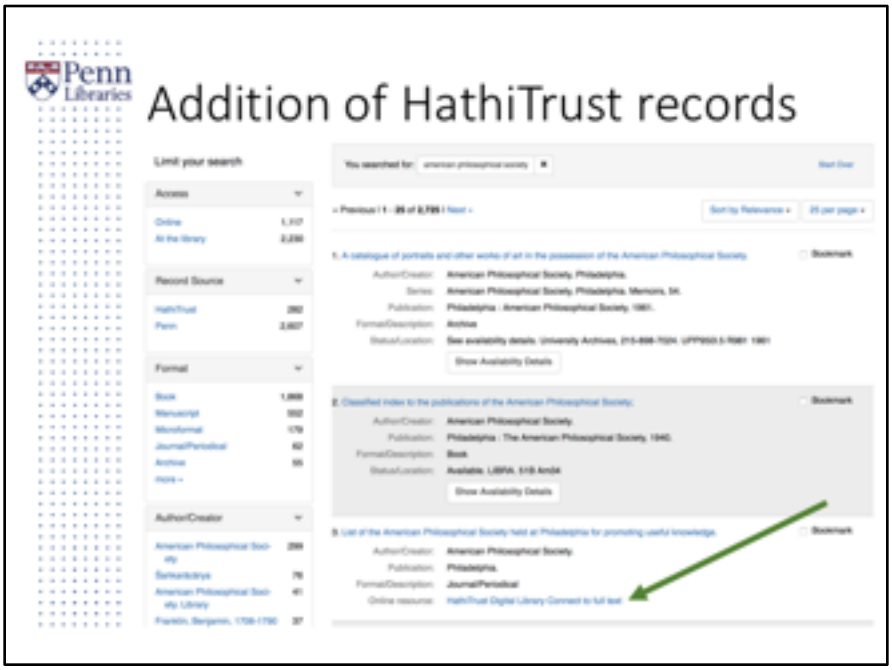
Emily Morton-Owens

Jon Shaw

Franklin is the discovery interface for the Penn Libraries.

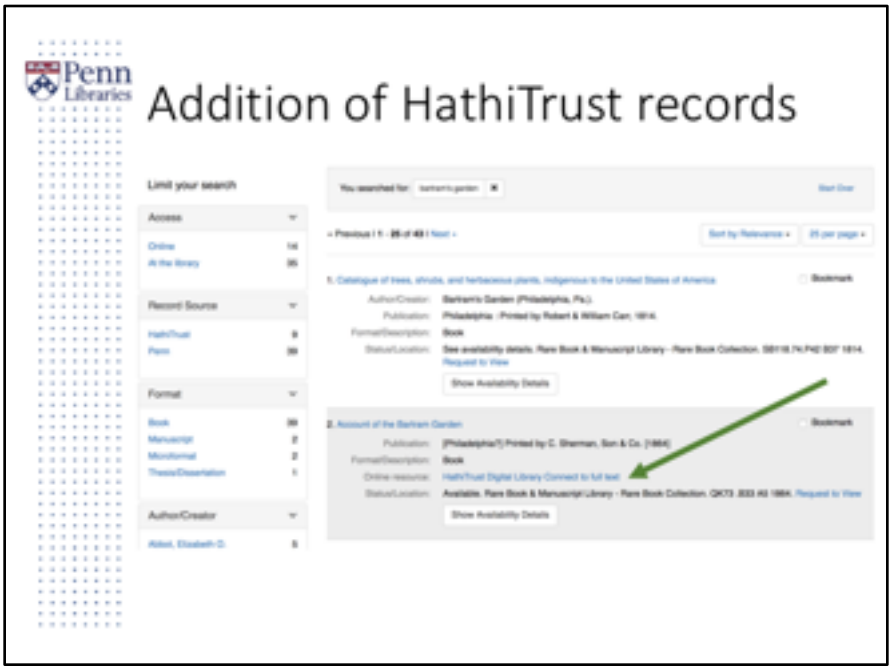
Starting in 2008, we replaced the OPAC provided with Voyager with a Solr-based catalog. Itwere presented using custom software built with Apache Cocoon. Franklin was improved in 2015 with the addition of Summon. We use the Summon API to run the search in the catalog and Summon simultaneously.

[x] In this presentation, we are discussing the Catalog part of Franklin.

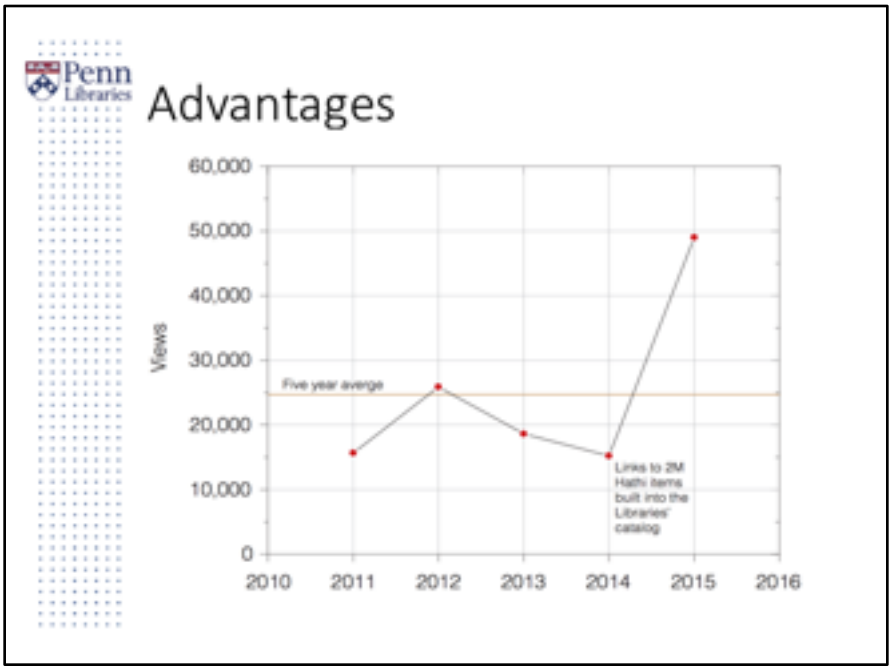


In 2014 Penn added 2.1 million HathiTrust records to the index. The goal here was to capitalize on the availability of this data to make the HathiTrust content more visible to users, and (we assumed) increase use of the HT. As you'll see, this has also driven an institutional interest in expanding the breadth of our catalog in other ways.

In this example search, HT records are interspersed in our catalog results using the same relevance ranking as our own local records.

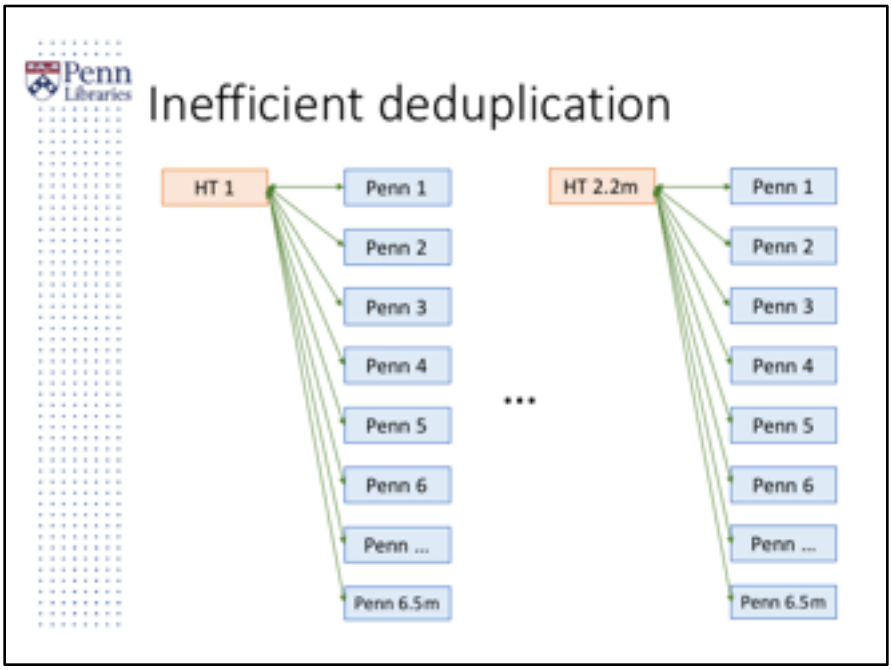


In this example, you can see that a HT record has been matched up with a local record, potentially saving the user a trip to the reading room.



The main advantage of this is that it did work—HT pageviews from Penn initially went up 264%. This showed us that surfacing content in this particular way did have the ability to drive discovery. Think of it as an affordance that doesn't rely on the user knowing what the HT is or where the information comes from or how to get it!

[HathiTrust page views by Penn readers exploded by 264% from 4,352 views in 3rd quarter 2014 to 15,854 views in 4th quarter 2014. https://pennlibnews.wordpress.com/2015/08/04/hathitrust-update-adds-384000-e-books-to-franklin-catalog/]



There were two problems with the initial incorporation of the HathiTrust records. The first was the approach taken for deduplicating the HT records against Penn's records, which was admittedly a bit quick and dirty.

When the index was built, each HT record would be compared to each Penn record in turn, and merged together if they matched. For this project, we match on OCLC numbers. We specify that one record set is the primary one, and its metadata is retained, but the holdings from the other are merged. In a sense we're really changing the secondary records to squash them into the primary one.

In 2017 numbers, this requires each of 2.6 million HT records to be compared to each of 6.5 million Penn records. A quick comparison, but millions of them. So maybe it was dirty… but not quick or clean.

You can probably think ahead to how badly this would scale if additional record sets were added beyond two, and how many comparisons it would take and how long it would take. This was a non-hypothetical concern as we already identified a third set we were interested in adding, CRL, to try to increase awareness among users that they have access to that collection.

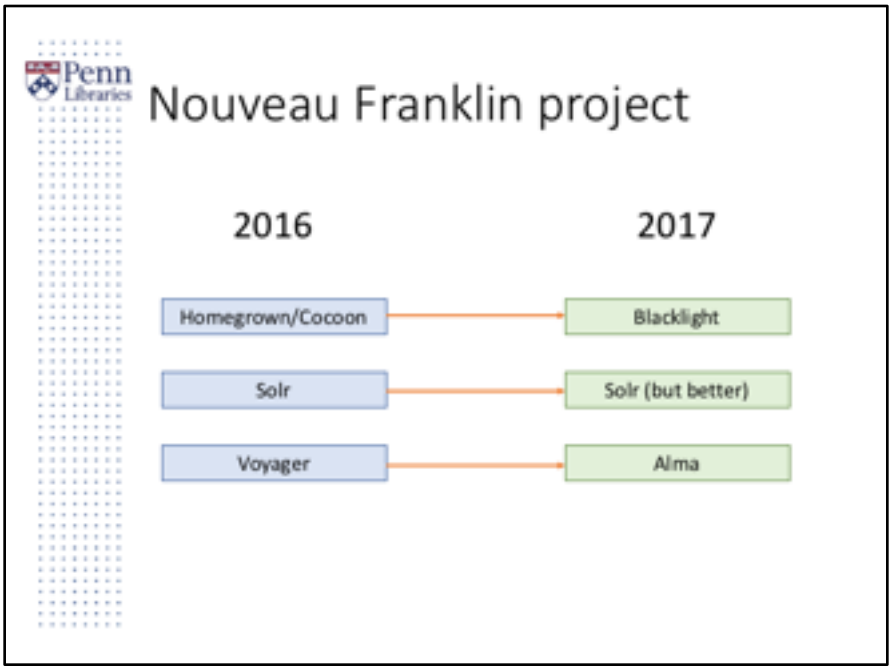
The second problem was that we had one set of users who really did not like the new data—our STEM users. Broadly speaking, the HT collections are (for copyright reasons) older materials, and just as broadly speaking, our STEM users like the newest materials. There was a perception that the HT records were just noise. I cheated and plucked these examples from deeper in the current search results, but they're examples of things that STEM users probably don't want when searching for "science."

Also, the original implementation relied on the outdated and confusing concept of "Penn Library Web," meaning library-licensed online resources. It was very unclear to users who chose this facet, whether they were faceting to exclude records that were *only* available from the HT or all records that were included in the HT load, regardless of whether they were also in Penn's collection.

Various proposed workarounds, like suggesting that STEM users limit by date, all had their downsides, for example, important STEM journals like Nature began their publishing runs many years ago.

The original method was not just inefficient, but conceptually flawed in a way that

was manifested in the user interface.



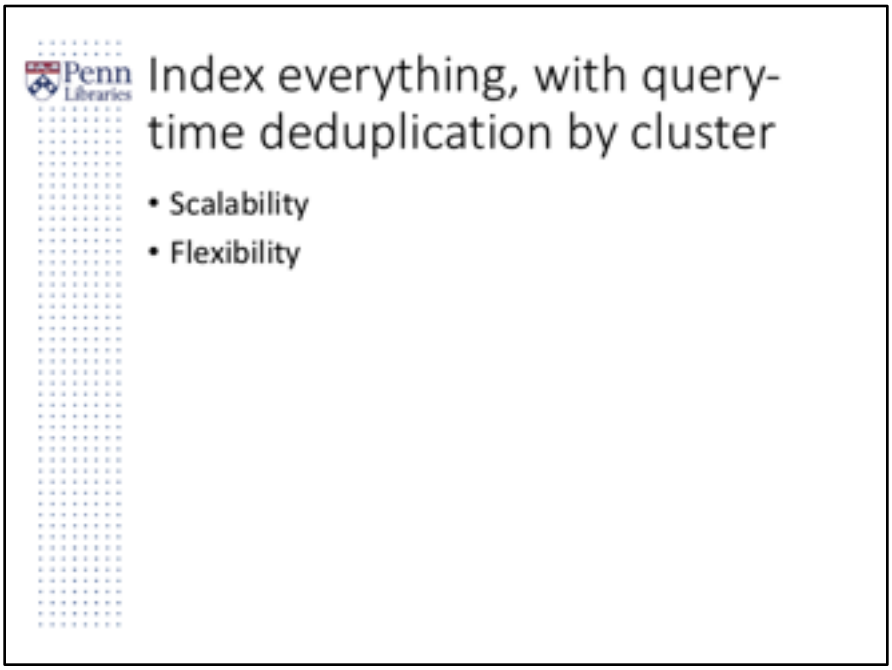
This past year, we had the opportunity to address these issues afresh, due to our ambitious double-barreled software migration. Penn elected to switch from Voyager to Alma. We also wanted to switch to Blacklight, which wasn't a huge conceptual leap because we already used Solr… but Blacklight would allow us to be using a common codebase rather than homegrown software. The Alma migration was already going to require enough changes to Franklin that we decided to rip off this Band-aid all at once. And we wanted to use this opportunity to address some of the most salient pain points revealed by our previous implementation.

[x] Next, Michael is going to describe the Solr configuration that we switched to, which addressed the flaws in our HT integration, as well as an additional improvement to Franklin involving the subject headings.

Penn Libraries

Query-time Deduplication

Leveraging Solr to integrate records from multiple sources



Scalability

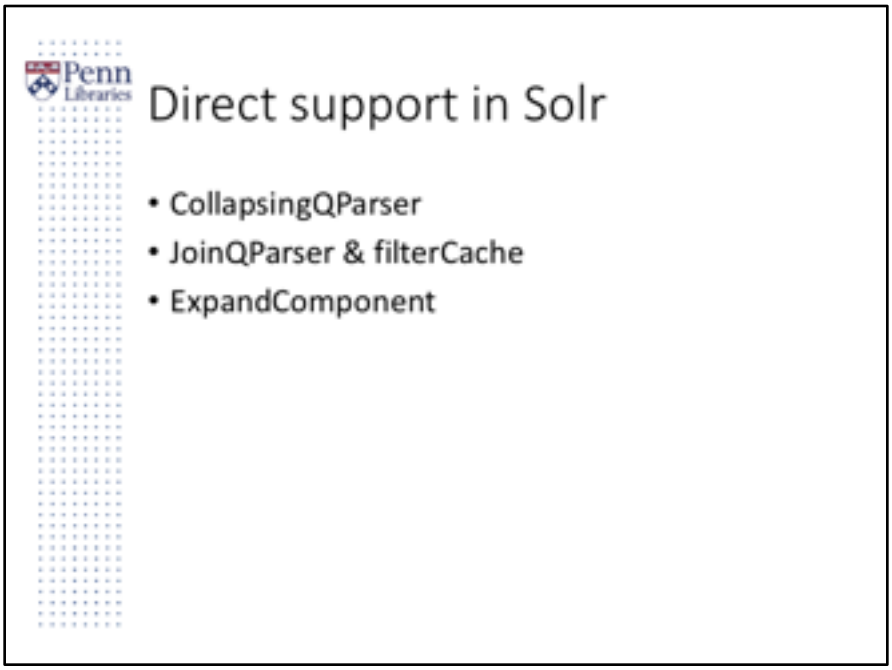
Use Solr for what it is designed to do

Avoid complexity and pitfalls of pre-processing

Flexibility

With index-time deduplication, decisions about supported query strategies are pre-determined, and static.

Query-time deduplication supports multiple approaches, dynamically



CollapsingQParser

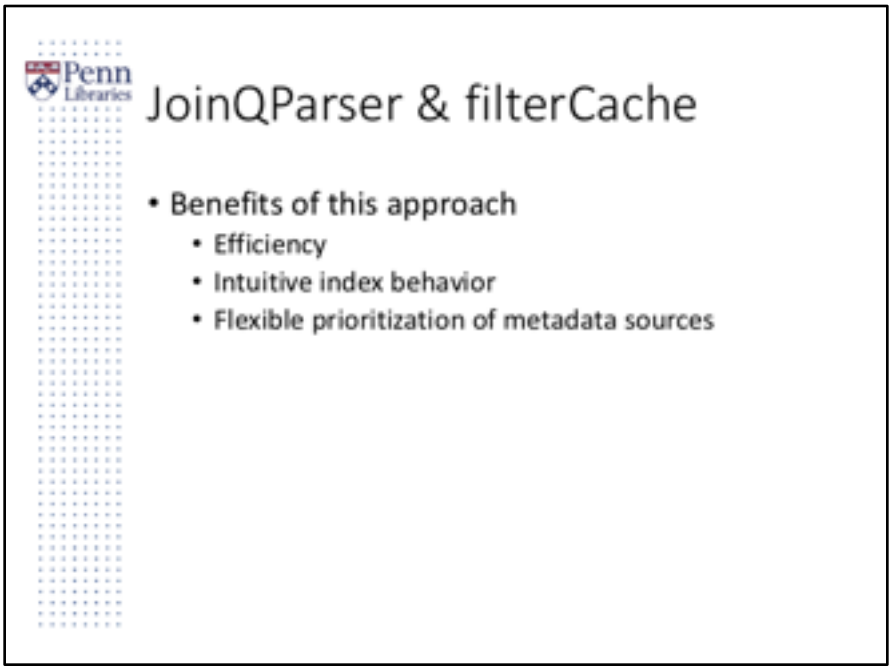
search over *all* records, deduplicate for search results display

JoinQParser & filterCache

custom define (and cache) a top-level set of records over which to search

ExpandComponent

for each top-level search result, also provide information about other records in the same cluster



Efficiency

User queries execute over a smaller portion of the index

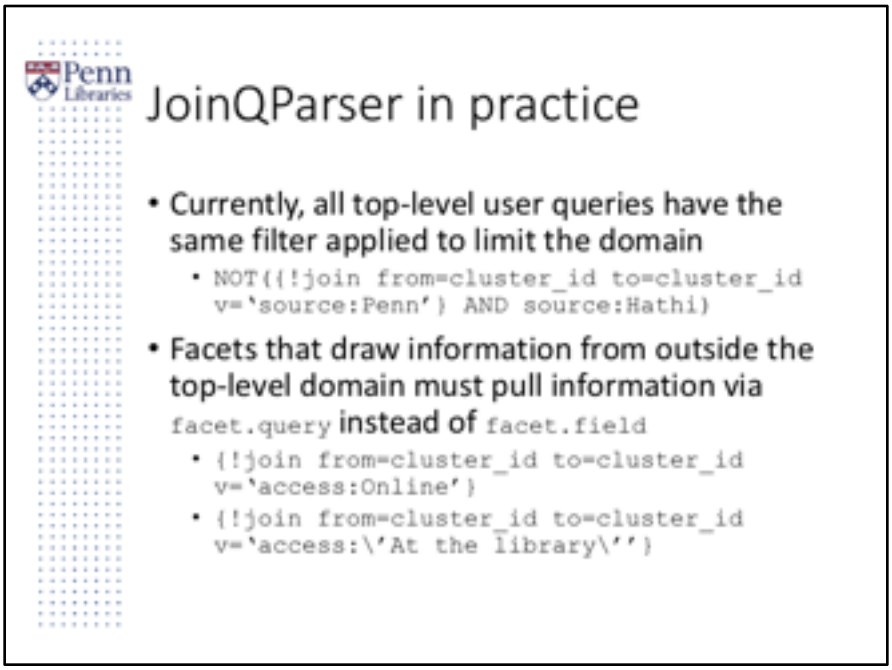
Intuitive behavior

Avoid unwanted clustering of records from the same curated record source

(e.g., local records that are associated with the same cluster id)

Facets behave predictably, always evaluated over the same top-level record set

Flexible prioritization of metadata sources



In practice, the OCLC ID is used as the cluster_id

Extension of top-level search domain

```
• NOT({!join from=cluster_id to=cluster_id
  v='source:Penn'} AND source:(LC OR Hathi OR CRL))
  AND NOT({!join from=cluster_id to=cluster_id
  v='source:LC'} AND source:(Hathi OR CRL))
  AND NOT({!join from=cluster_id to=cluster_id
  v='source:Hathi'} AND source:CRL)
```

- Importance of filterCache
- Flexibility of query strategies for different uses/environments

Example extending to more record sources …

Importance of filterCache

The join queries are relatively expensive, but once executed (e.g., in a warming query),

for user queries the associated `BitDocSet` from the filterCache is used,

which is extremely efficient

Flexibility, again!

In an environment desiring more flexibility (e.g., a multi-tenant environment), multiple

top-level domain filter query alternatives could be used

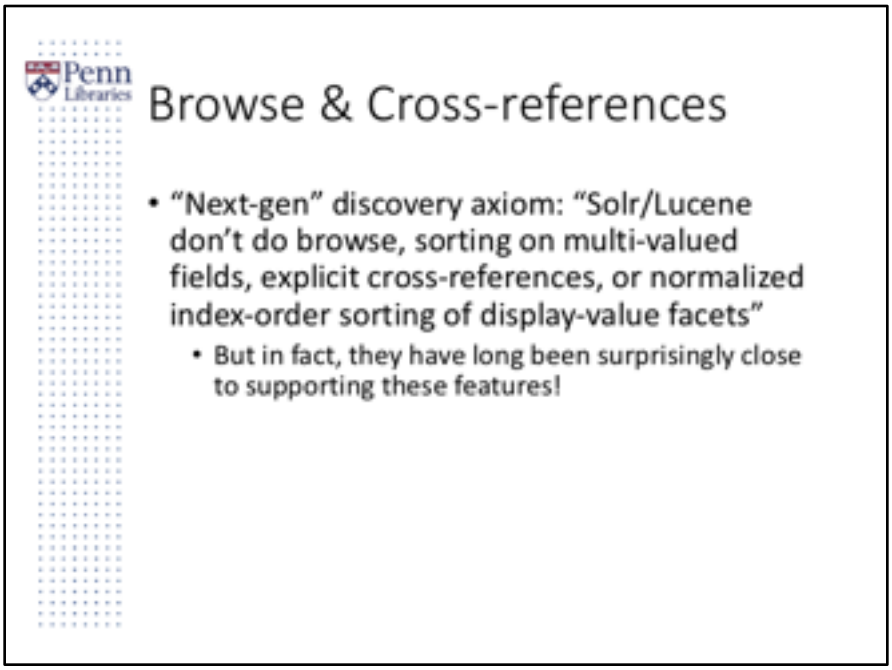
Create different "pseudo-sources" for brief records and full records, with different

metadata prioritization, accordingly

Penn Libraries

Browse and Cross-References

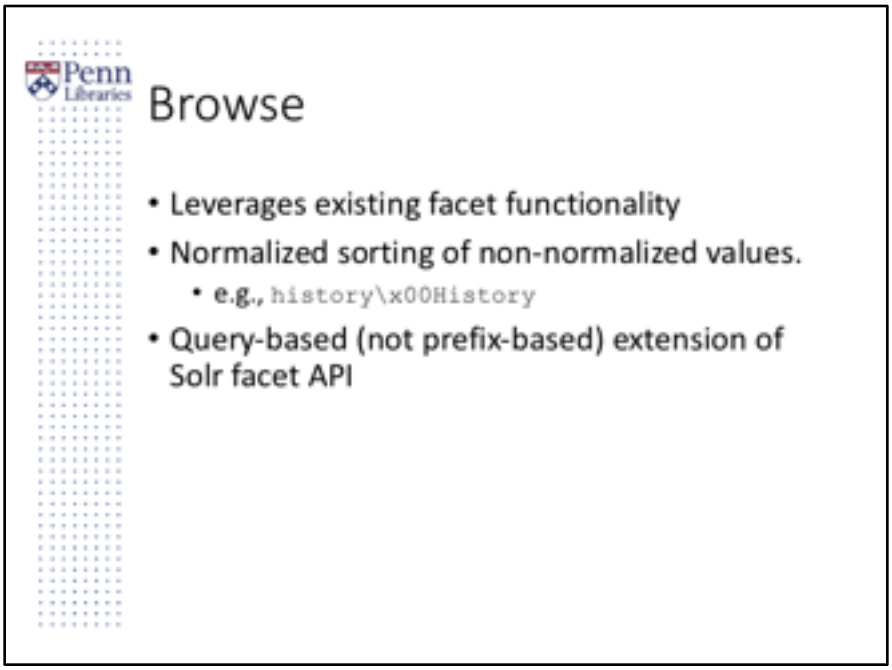
A Solr-native approach



Can we have our cake, and eat it, too?
Can we implement these features in a way that is fully integrated with Solr, and use them in concert with the features that Solr brings to the table?

Extending Solr for browse and cross-reference support

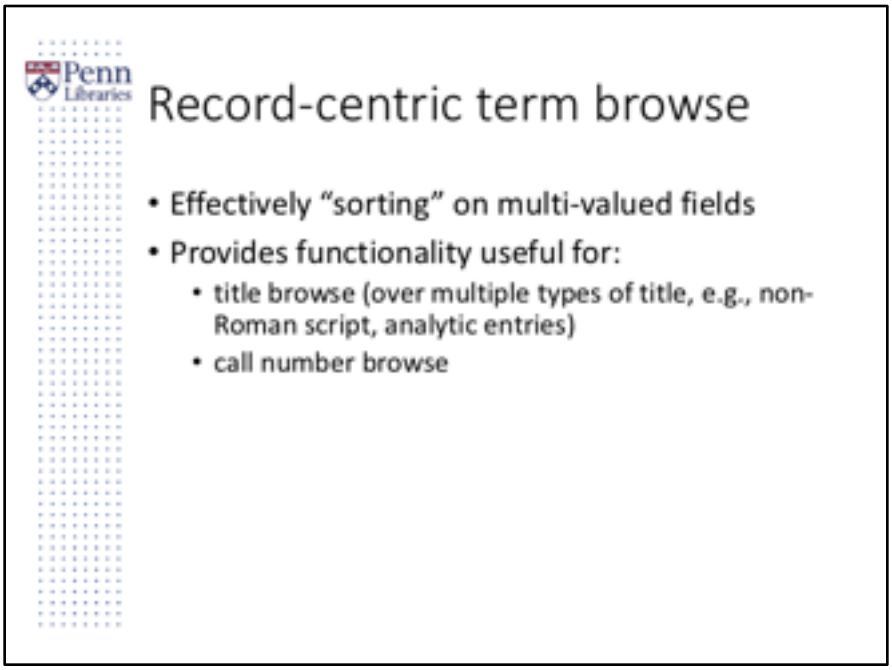
- The "solrplugins" project of the Penn libraries extends Solr to support these features
  - https://github.com/upenn-libraries/solrplugins
- The companion project "blacklight_solrplugins" may be used to integrate the Solr functionality in Blacklight discovery interfaces
  - https://github.com/upenn-libraries/blacklight_solrplugins



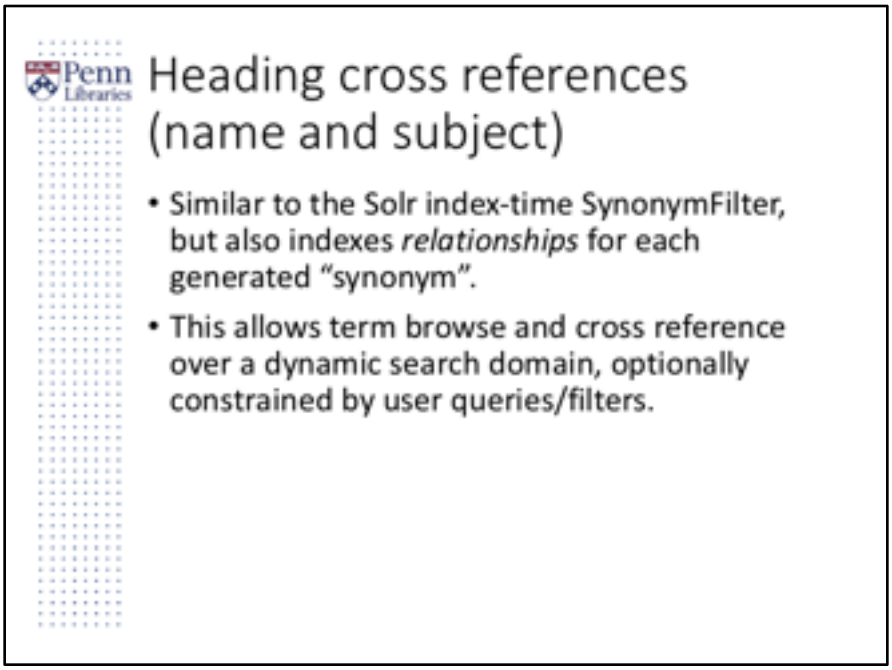
Browse functionality is provided by leveraging faceting functionality
Normalized (e.g., case-insensitive) index-order facet sorting is implemented via a scheme that indexes each display term prepended (null-delimited) with its normalized form.

e.g., `history\x00History`

Solr API is extended to allow user-query-specified arbitrary index-order result windows, and "paging" through index terms



This piggybacks off the Solr facet API enhancement for retrieving arbitrary index-order result windows, and for each term in the index, expands associated documents.



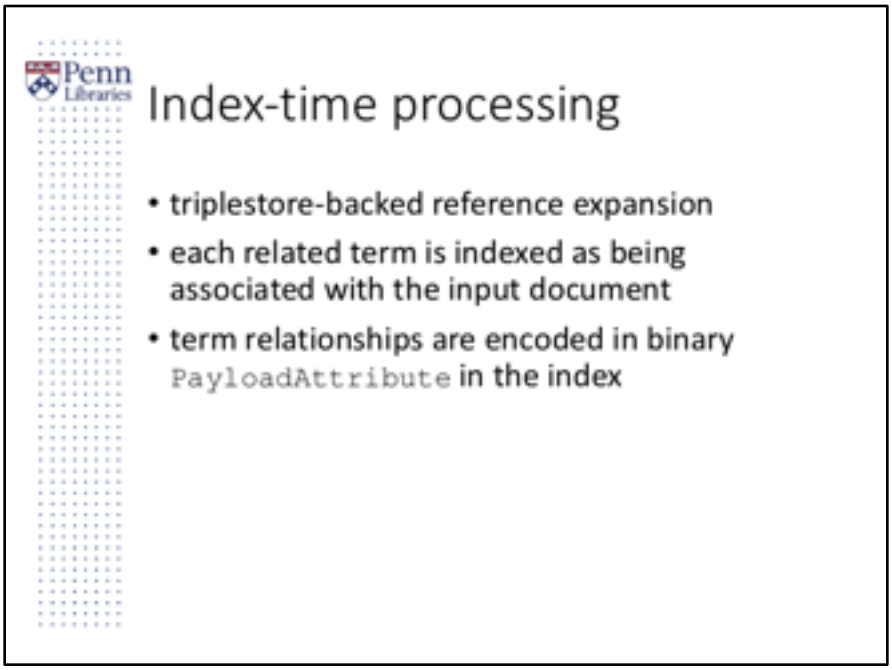
Examples:

facet for format:"recorded music", author:"Bach, Johann Sebastian", then browse

over associated author/performer names

facet for subject:"United States—History—Civil War, 1861-1865", language:"French", and

browse associated authors or other subjects



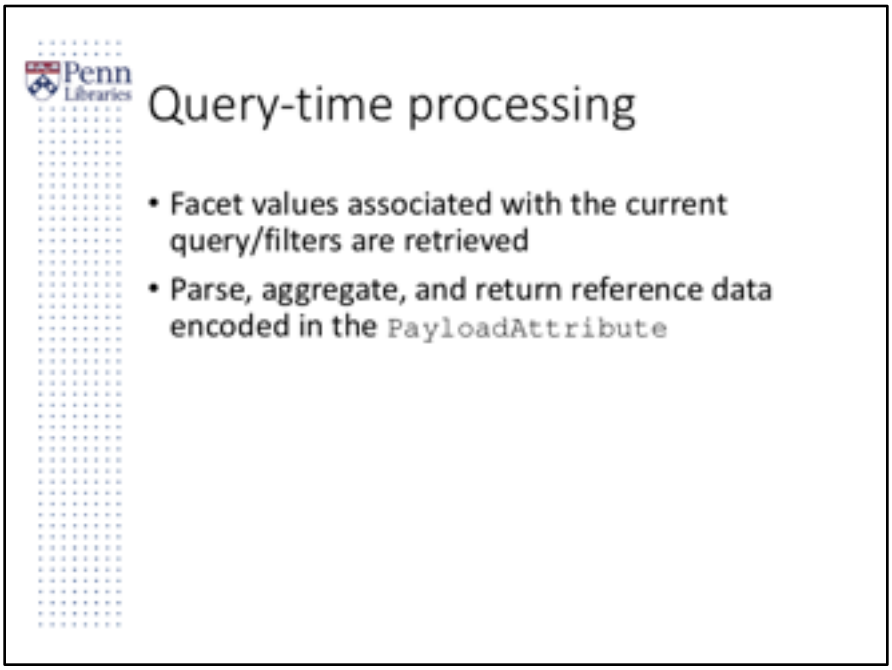
related terms are generated by a service backed by a triplestore populated with LC heading ntriple dump

But any scheme could be used to generate the references

Lucene per-doc-per-term `PayloadAttribute` is used to store any information about relationships associated with that term, e.g., as applicable:

reference to the term as it appears in document

reference to alternate/broader/narrower/preferred forms



Facet values associated with the current query/filters are accessed for a small index-order window

For each term/doc, the PayloadAttribute information is parsed and aggregated to build a data structure describing relationships for the given term.

Penn Libraries

What sets to integrate

& Enhancing the Discovery Experience

Penn Libraries

Impactful integration

Collections that are…

Digital

Unique

Desired

Penn Libraries

Augmented Solr Index

Remediated Controlled vocabularies

Record clustering based on descriptive traits

Mapping subjects

Penn Libraries

Thank you

https://github.com/upenn-libraries/solr-source-deduplication

mgibney@upenn.edu
egmowens@upenn.edu
jshaw4@upenn.edu