VIVO and the Solr search engine

What is Solr?

Solr is an open-source, enterprise level search platform, available from Apache. It is based on the popular Lucene search engine. VIVO uses a standard instance of Solr, without modification. You can learn more about Solr at the Apache Solr home page.

VIVO maintains its data in a semantic triple-store. A triple-store is very well suited for expressing a complex, flexible web of data relationship. It is not very well suited for text-based searches. Solr provides fast searching with features like

- weighted results by field,
- searching by the stems of words, rather than exact matches,
- faceted search results,
- and much more.

Solr provides these features much more efficiently than a triple-store would.

Solr maintains its own index of data, which reflects the contents of the triple-store. As the data in VIVO changes, the contents of the Solr index must change also. In most cases this happens automatically, but not always. Sometimes the search index must be rebuilt to bring it into synchronization with the triple-store. See the section below called "When is the index updated?" for more information.

Solr is implemented as a self-contained web application, separate from VIVO. At most VIVO sites, Solr and VIVO run on the same machine, but this is not the only possible configuration. It is possible to host Solr on a different computer from VIVO.

In a typical VIVO installation, Solr is hidden behind VIVO, and the users do not access it directly. In general, they don't know that Solr exists as an application. Before releasing VIVO to the public, a site admin should take steps to secure their Solr installation to ensure only administrators and the VIVO application can modify the search index.

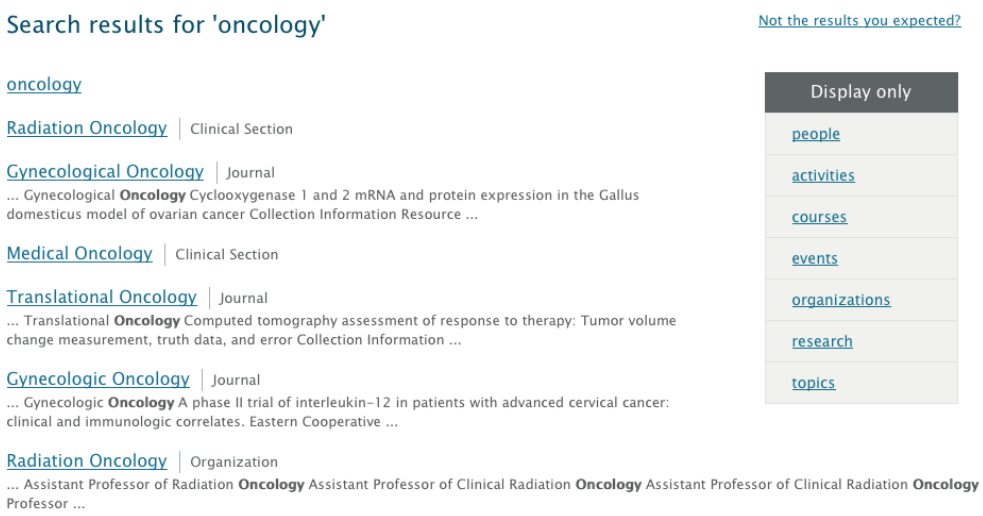
How does VIVO use Solr?

VIVO uses the Solr search engine in two ways:

- as a service to the end user,
- as a tool within the structure of the application.

Solr for the end user

Like many web sites, VIVO includes a search box on every page. The person using VIVO can type a search term, and see the results. This search is conducted by Solr, and the results are formatted and displayed by VIVO.

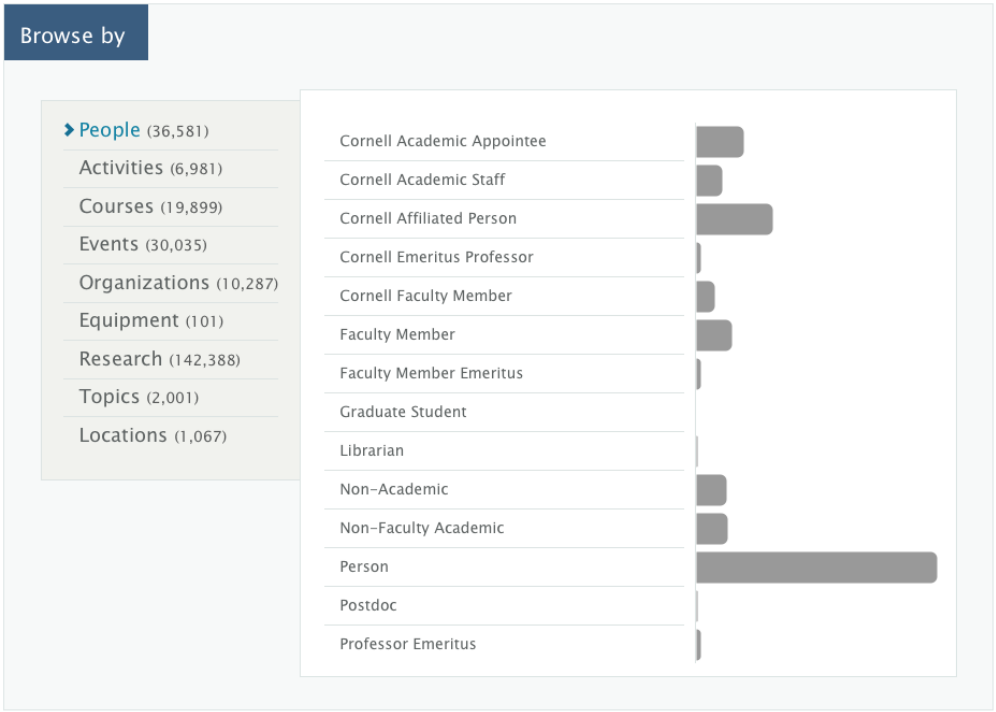


Solr allows for a "faceted" search, and VIVO displays the facets on the right side of the results page. These allow the user to filter the search results, showing only entries for people, or for organizations, etc.

Solr within VIVO

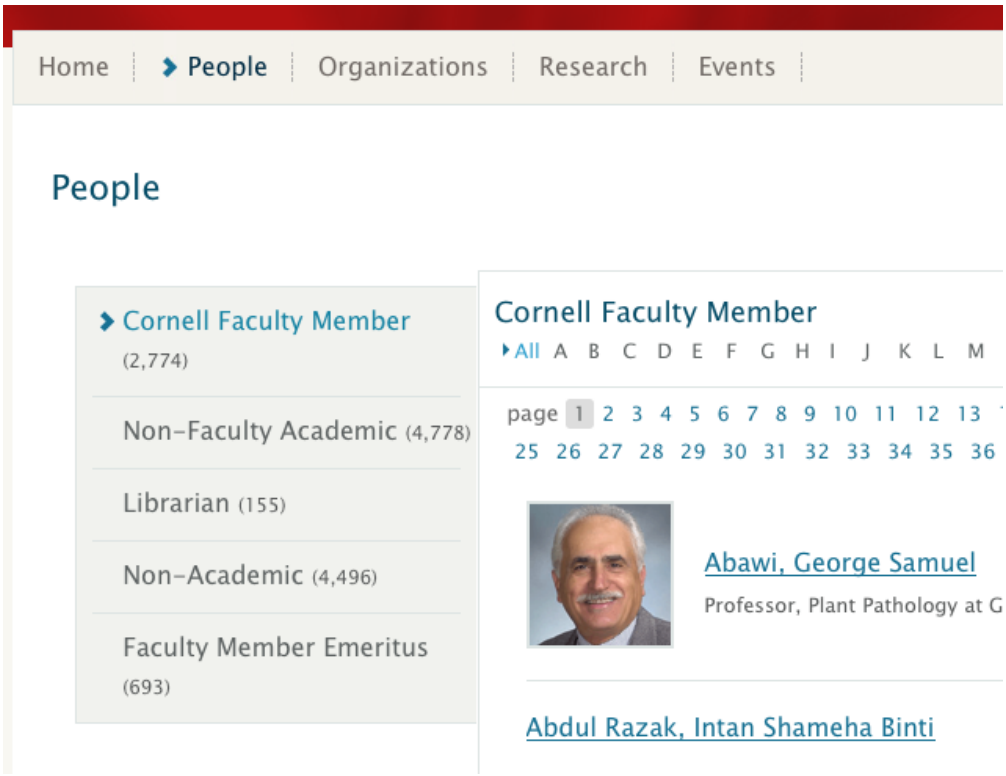
VIVO is based around an RDF triple-store, which holds all of its data. However, there are some tasks that a search engine can do much more quickly than a triple-store. Some of the fields in the Solr search index were put there specifically to help with these tasks.

For example, the browse area on the home page shows how many individuals VIVO holds for each class group.



VIVO could produce this data by issuing a SPARQL query against its data model. However, this would take several seconds for a large site, and we do not want the user to wait that long to see the home page. To avoid this delay, the class group of each individual is stored in the Solr record for that individual. Solr can count these fields very quickly, so VIVO issues a Solr query against the index, and displays the results on the home page.

Record counts on VIVO's index pages are obtained using the same type of Solr query.



How is Solr created and configured?

Solr is an external component to VIVO and must be installed separately. See "Configure and Start Solr" for detailed installation instructions.

The behavior of Solr depends extensively on its configuration files. These are stored in a directory that is called the Solr Home directory. When VIVO runs, the Solr search index is built inside the Solr Home directory.

The search index

What is in the index?

The Solr search index contains one record for each Individual in VIVO, unless that individual is explicitly excluded from the index. Exclusions are usually made for individuals that represent "context nodes" in the VIVO data model.

For example, if a professor teaches a course, the search index will contain:

- a record for the professor
- a record for the course

The VIVO data model also contains an individual that represents this teaching activity. That individual will be excluded from the index, since users would almost certainly prefer to find the teacher or the course in their search results, rather than the concept that connects the two.

What is in each record?

Each record in the search index contains several fields (see the chart below). The most commonly used field is `alltext`, In the record for a faculty member, `alltext` will contain her name, the name of her department, the names of her classes, the names of her papers and grants, etc. So, if you search for "Carpenter", you might see results for people named Carpenter, people in the Carpentry department, people who have written papers about carpentry, or have worked on grants about carpentry. You would also see results for the department itself, for the papers, and for the grants.

Solr index fields, VIVO 1.6

DocId	nameRaw	PREFERRED_TITLE
URI	nameText	siteURL
ALLTEXT	nameLowercase	siteName
ALLTEXTUNSTEMMED	nameLowercaseSingleValued	THUMBNAIL
classgroup	nameUnstemmed	THUMBNAIL_URL
type	nameStemmed	indexedTime
mostSpecificTypeURIs	acNameUntokenized	timestamp
BETA	acNameStemmed	etag
PROHIBITED_FROM_TEXT_RESULTS	NAME_PHONETIC	

When is the index updated?

During normal operation

When an individual is added, edited, or deleted through VIVO's user interface, Solr is given the new information and the index is updated.

VIVO administrators may also make changes to the data using the Advanced Data Tools, which are accessible from the Site Administration page. These tools also pass the data changes to Solr, so the index is kept current with the data.

Finally, data can be modified using the The SPARQL Update API. Again, Solr receives the changes and the index remains current.

On demand

Some tools, such as the VIVO Harvester, bypass VIVO and write directly to the data store. Solr is not notified when these tools are used, and the data becomes out of sync with the search index.

Other circumstances can cause issues with the search index. Perhaps a problem required you to restore your database to a backup, but you did not restore your search index at the same time. Perhaps you are developing a modification for VIVO, and you have emptied your database in order to test it. Perhaps VIVO crashed while data was being ingested.

In any of these circumstances, the solution is to login to VIVO as an administrator, navigate to the `Site Administration` page and click on `Rebuild search index`.

The existing search index remains in place while the new index is being built. When the rebuild is complete, the new index replaces the old one, and the old index is deleted.

Customizing the index

Creating custom fields

There are two parts to adding a custom field to VIVO's search index, defining the VIVO SPARQL query and defining the search engine's fields that will be populated.

Custom queries can be added to any file in the vivo-home/rdf/display/everytime directory (or any other file directory read by VIVO during startup). Using searchIndexerConfigurationVivo.n3 as a template, create a query that returns the data you wish to add to the search index. For example, if you wanted to create a search field for tracking open access publications you flag the content with a custom data property:

```
@prefix : <http://vitro.mannlib.cornell.edu/ns/vitro/ApplicationSetup#> .
@prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#> .

:vivodocumentModifier_openAccess
    a   <java:edu.cornell.mannlib.vitro.webapp.searchindex.documentBuilding.SelectQueryDocumentModifier> ,
        <java:edu.cornell.mannlib.vitro.webapp.searchindex.documentBuilding.DocumentModifier> ;
    rdfs:label "open access" ;
    :hasTargetField "open_access_s" ;
    :hasSelectQuery """
        PREFIX wos: <http://webofscience.com/ontology/wos#>
        SELECT ?status
        WHERE {
            ?uri wos:openAccess ?status .
        }
        LIMIT 1
        """ .
```

Second (at least in the case of Solr), you must add the new field to the search index schema. For Solr 6 and older, this can be defined directly in Solr's schema.xml at any time (schema.xml included with Vitro prior to v1.11 for reference). For later versions of Solr, schema.xml will be read during core creation, but will not read changes after the fact. Instead, you may add new fields using Solr's Schema API. For the above example, you could post:

```
curl -X POST -H 'Content-type:application/json' --data-binary '{
  "add-field": {
      "name":"open_access_s",
      "type":"text",
      "multiValued":false,
      "stored":true}
  }' http://localhost:8983/api/cores/vivocore/schema
```

Excluding specific classes from the search index

Exclusions can be added to any file in the vivo-home/rdf/display/everytime directory (or any other file directory read by VIVO during startup). You may want to overwrite searchIndexerConfigurationVivo.n3 to keep the search configuration in one place. You can exclude individual class types:

```
@prefix : <http://vitro.mannlib.cornell.edu/ns/vitro/ApplicationSetup#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

:vivoSearchExcluder_typeExcluder
    a   <java:edu.cornell.mannlib.vitro.webapp.searchindex.exclusions.ExcludeBasedOnType> ,
        <java:edu.cornell.mannlib.vitro.webapp.searchindex.exclusions.SearchIndexExcluder> ;
    :excludes
        "http://purl.org/NET/c4dm/event.owl#Event" .
```

... or all class types within a certain namespace:

```
@prefix : <http://vitro.mannlib.cornell.edu/ns/vitro/ApplicationSetup#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

:vivoSearchExcluder_namespaceTypeExcluder
    a   <java:edu.cornell.mannlib.vitro.webapp.searchindex.exclusions.ExcludeBasedOnTypeNamespace> ,
        <java:edu.cornell.mannlib.vitro.webapp.searchindex.exclusions.SearchIndexExcluder> ;
    :excludes
        "http://purl.org/NET/c4dm/event.owl#" .
```

... or the namespace of the object URI:

```
@prefix : <http://vitro.mannlib.cornell.edu/ns/vitro/ApplicationSetup#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

:vivoSearchExcluder_namespaceExcluder
    a   <java:edu.cornell.mannlib.vitro.webapp.searchindex.exclusions.ExcludeBasedOnNamespace> ,
        <java:edu.cornell.mannlib.vitro.webapp.searchindex.exclusions.SearchIndexExcluder> ;
    :excludes
        "http://localhost/private/" .
```

How does VIVO contact Solr?

VIVO must be configured to communicate with Solr. This is accomplished by updating the "runtime.properties" to point to the URL of Solr. More installation details can be found in the "Configure and Start Solr" documentation.