HierarchicalFaceting

Approaches to Hierarchical Facets in Solr

This document contains various suggestions and solutions for dealing with "Hierarchical Facets" - a concept which can mean differnet things to differnet people depending on their data.

'facet.prefix' Based Drill Down

⚠️ Solr1.2

Transcribed (with slight edits) from Mastering the Power of Faceted Search by Chris Hostetter of Lucid Imagination (starting ~28min in)

For simple categories, *facet.field* works fine as-is. However, categorization schemes are frequently organized in a hierarchically-structured scheme, and the user experience for interacting with that taxonomy involves drill down, starting at the top (more general) and whittling the way down (more specific).

This is a basic approach that works well for most use cases and takes advantage of basic Solr faceting parameters by encoding the facet terms at index time.

Flattened Data “breadcrumbs”

```
Doc#1: NonFic > Law
Doc#2: NonFic > Sci
Doc#3: NonFic > Hist, NonFic > Sci > Phys
```

In this example, we have documents associated with multiple categories, like Doc#3. We also have documents that are mapped to internal nodes, like Doc#2.

You must perform some index time processing on this flattened data in order to create the tokens needed for a facet.prefix approach. When we index the data we create specially formatted terms that encode the depth information for each node that appears as part of the path, and include the hierarchy separated by a common separator (“depth/first level term/second level term/etc”). We also add additional terms for every ancestors in the original data.

Indexed Terms

```
Doc#1: 0/NonFic, 1/NonFic/Law
Doc#2: 0/NonFic, 1/NonFic/Sci
Doc#3: 0/NonFic, 1/NonFic/Hist,
            0/NonFic, 1/NonFic/Sci, 2/NonFic/Sci/Phys
```

Terms Containing Another Term in the Beginning

In case you are indexing terms that may have another term in the beginning, adding a separator at the end of each term helps distinquish these terms:

```
Doc#1: 0/Books/, 1/Books/Book/
Doc#2: 0/Books/, 1/Books/BookPart/
```

Then in the query always include a trailing slash, e.g. "facet.prefix = 1/Books/Book/" to avoid matching "1/Books/BookParts".

Initial Query

With this type of index data, we can then go on and query this to get a drill-down. Initially, we can say we want to facet on the category field with the *facet.prefix* “1/NonFic”: things that are children of NonFic at a depth of 1.

```
facet.field = category
facet.prefix = 1/NonFic
facet.mincount = 1
```

```
<result numFound=”3” ...
<lst name=”facet_fields”>
  <lst name=”category”>
     <int name=”1/NonFic/Sci”>2</int>
     <int name=”1/NonFic/Hist”>1</int>
     <int name=”1/NonFic/Law”>1</int>
```

Drill Down

If we drill down into NonFic/Sci, we just add the *fq* (filter query) as normal and tweak the *facet.prefix* from the children 1/NonFic to the children of 2/NonFic /Sci

```
fq = {!raw f=category}1/NonFic/Sci
facet.field = category
facet.prefix = 2/NonFic/Sci
facet.mincount = 1
```

```
<result numFound=”2” ...
<lst name=”facet_fields”>
  <lst name=”category”>
     <int name=”2/NonFic/Sci/Phys”>1</int>
```

We’ve used the depth prefix that lets us look one level deep, but by tweaking the encoding, alternative user experiences can be created.

PathHierarchyTokenizerFactory

⚠️ Solr 3.1

The solr.PathHierarchyTokenizerFactory is designed to output file path hierarchies as synonyms, but can also be used in other simple hierarchies.

Flattened Data

```
Doc #1: /usr/local/apache
Doc #2: /etc/apache2
Doc #3: /etc/apache2/conf.d
```

Output Tokens

```
Doc #1: /usr, /usr/local, /usr/local/apache
Doc #2: /etc, /etc/apache2
Doc #3: /etc, /etc/apache2, /etc/apache2/conf.d
```

Initial Query

```
facet.field = category
facet.mincount = 1
```

```
<result numFound=”3” ...
<lst name=”facet_fields”>
  <lst name=”category”>
     <int name=”/etc”>2</int>
     <int name=”/etc/apache2”>2</int>
     <int name=”/etc/apache2/conf.d”>1</int>
     <int name=”/usr”>1</int>
     <int name=”/usr/local”>1</int>
     <int name=”/usr/local/apache”>1</int>
```

Unlike the *facet.prefix* approach, it isn’t as easy to constrain the depth of the taxonomies, but for small numbers of terms this may be a good approach.

Pivot Facets

⚠️ SOLR-792] [https://issues.apache.org/jira/browse/SOLR-792

Pivot facets are query time constructs that allow arbitrary facet results, but they should be used wisely to avoid performance bottlenecks.

You can think of it as "Decision Tree Faceting" which tells you in advance what the "next" set of facet results would be for a field if you apply a constraint from the current facet results, e.g. "for facet A, the constraints/counts are X/N, Y/M,” and if you were to constrain A by X, then the constraint counts for B would be S/P, T/Q, etc. Another way to think of it is each field is treated as a vector containing the constraint counts for that field, and taking a "cross product" to produce an N-dimensional matrix showing the counts for each permutation.

This feature can be easily applied to hierarchical facets in some cases, particularly those where a particular document only appears at one point in the taxonomy.

Flattened Data “breadcrumbs”

```
Doc#1: NonFic > Law
Doc#2: NonFic > Sci
Doc#3: NonFic > Sci > Phys
```

At index time, we split the data into a separate field for each level of the hierarchy.

Indexed Terms

```
Doc#1: category_level0: NonFic; category_level1: Law
Doc#2: category_level0: NonFic; category_level1: Sci
Doc#3: category_level0: NonFic; category_level1: Sci, category_level2:Phys
```

When querying Solr, we specify the facet.pivot parameter, which is a comma-separated list of fields to “pivot” on:

```
facet.pivot = category_level0,category_level1,category_level2
```

```
<result numFound=”3” ...
<lst name=”facet_pivot”>
  <arr name=”category_level0,category_level1,category_level2”>
    <lst>
      <str name=”field”>category_level0</str>
      <str name=”value”>NonFic</str>
      <int name=”count”>3</int>
       <arr name=”pivot”>
         <lst>
             <str name=”field”>category_level1</str>
             <str name=”value”>Law</str>
             <int name=”count”>1</int>
         </lst>
         <lst>
             <str name=”field”>category_level1</str>
             <str name=”value”>Sci</str>
             <int name=”count”>2</int>
             <arr name=”pivot”>
               <lst>
                  <str name=”field”>category_level2</str>
                  <str name=”value”>Phys</str>
                  <int name=”count”>1</int>
               </lst>
              </arr>
          </lst>
       </arr>
```

Strict hierarchical facets

SOLR-64

Strict Facet Hierarchies:

- a response format that can more efficiently encapsulate hierarchies
- maybe format flexible enough to encompass other hierarchies (non-tree, defined via solrconfig.xml, etc)
- strict hierarchy field faceting, with a single field value per document containing all the ordered constraints (a path)
- ability to select depth to return to the client
- perhaps an ability to select a variable depth based on the number of items selected at that node
- expand node if count > 100.... or maybe expand node if count > 10% of hits

Multipath hierarchical faceting

SOLR-2412

Hierarchical faceting with slow startup, low memory overhead and fast response. Distinguishing features as compared to SOLR-64 and SOLR-792 are

- Multiple paths per document
- Query-time analysis of the facet-field; no special requirements for indexing besides retaining separator characters in the terms used for faceting
- Optional custom sorting of tag values
- Recursive counting of references to tags at all levels of the output

This is a shell around LUCENE-2369, making it work with the Solr API. The underlying principle is to reference terms by their ordinals and create an index wide documents to tags map, augmented with a compressed representation of hierarchical levels.

Faceting Module

https://issues.apache.org/jira/browse/LUCENE-3079

TBD (To Be Documented)