SolrFacetingOverview

What is Faceted Search

From the user perspective, faceted search (also called faceted navigation, guided navigation, or parametric search; Microsoft calls them "Refiners") breaks up search results into multiple categories, typically showing counts for each, and allows the user to "drill down" or further restrict their search results based on those facets.

Solr provides a faceting component which is part of the standard request handler and can also be used by various other request handlers to include "Facet counts" based on some simple criteria.

This page briefly provides some general background information:

Facet Indexing

Faceting is done on indexed rather than stored values. This is because the primary use for faceting is drill-down into a subset of hits resulting from a query, and so the chosen facet value is used to construct a filter query which literally matches that value in the index. For the stock Solr request handlers this is done by adding an `fq=<facet-field>:<quoted facet-value>` parameter and resubmitting the query.

Because faceting fields are often specified to serve two purposes, human-readable text and drill-down query value, they are frequently indexed differently from fields used for searching and sorting:

- They are often not tokenized into separate words
- They are often not mapped into lower case
- Human-readable punctuation is often not removed (other than double-quotes)
- There is often no need to store them, since stored values would look much like indexed values and the faceting mechanism is used for value retrieval.

As an example, if I had an "author" field with a list of authors, such as:

- *Schildt, Herbert; Wolpert, Lewis; Davies, P.*

I might want to index the same data differently in three different fields (perhaps using the Solr copyField directive):

- For searching: Tokenized, case-folded, punctuation-stripped:
  - schildt / herbert / wolpert / lewis / davies / p
- For sorting: Untokenized, case-folded, punctuation-stripped:
  - schildt herbert wolpert lewis davies p
- For faceting: Primary author only, using a `solr.StringField`:
  - Schildt, Herbert

Then when the user drills down on the "Schildt, Herbert" string I would reissue the query with an added fq=author:"Schild, Herbert" parameter.

Facet Operation

Currently SimpleFacets has 3 modes of operation, selected by a combination of SimpleFacetParameters and schema.xml Field definitions:

FacetQueries

Any number of facet.query parameters can be passed to the request handler. The filter for each distinct facet.query is retrieved from the filterCache (or generated if not cached yet) and intersected with the filter for the base query to obtain the count.

FacetFields

Any number of facet.field parameters can be passed to the request handler. For each facet.field, one of two approaches will be used based on the facet.method or the field type:

- **Enum Based Field Queries**: If `facet.method=enum` or the field is defined in the schema as boolean, then Solr will iterate over all of the indexed terms for the field, and for each term it will get a filter from the filterCache and calculate the intersection with the filter for the base query. This is excellent for fields where there is a small set of distinct values. The average number of values per document does not matter. For example, faceting on a field with U.S. States e.g. `Alabama, Alaska, ... Wyoming` would lead to fifty cached filters which would be used over and over again. The filterCache should be large enough to hold all of the cached filters.
- **Field Cache**: If `facet.method=fc` then a field-cache approach will be used. This is currently implemented using either the the Lucene FieldCache or (starting in Solr 1.4) an UnInvertedField if the field either is multi-valued or is tokenized (according to FieldType.isTokened()). Each document is looked up in the cache to see what terms/values it contains, and a tally is incremented for each value. This is excellent for situations where the number of indexed values for the field is high, but the number of values per document is low. For multi-valued fields, a hybrid approach is used that uses term filters from the filterCache for terms that match many documents.