ExposedFaceting

Exposed faceting

Experimental faceting contrib, available as SOLR-2412.

- Optional hierarchical faceting
  - 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
  - Low memory overhead for hierarchical faceting structure: Typically 1 byte or less per unique path
  - Recursive counting of facet values at all levels of the output
- Optional custom sorting of facet values
- Total tags/facet, total matching tags/facet counting
- Startup speed and memory footprint is about the same as equivalent Solr faceting, except with many fields where memory footprint is markedly lower
- Facet search response time is the same as or lower than the Solr facet equivalent; more fields means comparatively better performance for exposed faceting
- Grouped facet structure, making it possible to facet on thousands of fields (see below)
- Limited in scope (not a good thing here)
  - Full-index structure (no segment based faceting)
  - String values only: No special handling of numbers, dates etc.
  - Single-core only: No support for distributed search

The Exposed core is a Lucene module and the Solr contrib is only a wrapper; full functionality is provided both with and without Solr.

Facet structure

The faceting structure is an index-wide list of references to term ordinals. A list of pointers from document IDs to reference sublists is maintained. Ideally, the memory requirement for this setup is `#documents*log2(#references) + #references*log2(#unique_values) bit`. In reality 3-4 times that is needed due to temporary building overhead and counting structures.

Multiple fields are handled by offsetting the ordinals of their terms by the sum of the preceding term ordinals. The downside is that a new facet structure must be calculated for any new combination of wanted fields. The upside is that there is practically no extra memory or performance cost of using multiple fields instead of a single one. This has been verified experimentally by faceting on 9000 fields in 10 million documents.

Custom sorting of facet values adds a list of size `#unique_values*log2(#unique_values) bit` to the structure. This can have considerable impact on startup time (rule of thumb: It is 5 times slower) and has practically no impact on search time. When running as a Lucene module, it is possible to avoid memory overhead and increase speed by sorting by ICUCollatorKeys (very experimental).

Hierarchical faceting adds a list of size `#unique_values*(2*log2(maximum_depth)) bit` to the structure. This has light impact on startup time (rule of thumb: Less than 2 times slower) and a hierarchical faceted search is only a little slower than a non-hierarchical one. See Fast, light, n-level hierarchical faceting for details.

When exposed faceting might be considered

- If hierarchical faceting in Solr is needed
- If both the number of documents and the number of facet fields are high (50M documents and 30 facet fields for example) and memory-requirements & CPU-load must be low

In many cases, multi-path hierarchical faceting can be simulated in Solr at the cost of memory and/or speed. In some cases it is possible to mitigate the memory & CPU-toll of many-documents-many-fields faceting in Solr by collapsing fields (less memory & CPU), using segment based faceting (less memory) or DocValues (less memory, possibly less CPU).

How to make it work

SOLR-2412 provides instructions on how to build the contrib.

TODO Add samples for requests and responses.