DateRangeQueries

Issues concerning DateRangeQueries

Caching

A DateFilter does not cache, so each search re-enumerates the terms in the range. In fact, DateFilter by itself is practically of no use (Erik Hatcher, message). The cache is keyed by IndexReader.

If you have a set of canned date ranges, there are two approaches worth considering:

DateFilter wrapped by a CachingWrappingFilter
RangeQuery wrapped in a QueryFilter (which does cache)

Kevin A. Burton posted some results (response time in ms):

Before caching the Field	After caching the field
2238	2253
1910	10
1899	10
1901	6
1904	8
1906	6

Erik Hatcher also wrote(message): One more point... caching is done by the IndexReader used for the search, so you will need to keep that instance (i.e. the IndexSearcher) around to benefit from the caching.

Using a Filter Instead

In many cases (I would argue: all cases) it doesn't make sense to Query on a Range of Dates. More then likely what you really want to do is *Filter* on a Range of Dates. Querying scores items based on the frequency of terms – which is something most people don't care about when dealing with dates, particularly given the overhead involved. Consider using a RangeFilter (or possibly a RangeFilter wrapped in a FilteredQuery) instead.

http://nagoya.apache.org/eyebrowse/BrowseList?listName=lucene-user@jakarta.apache.org&by=thread&from=943115

Using indexed hierarchical prefixes

When the above advice does not help enough, consider indexing a date CCYYMMDD as multiple prefixes on the same index positions:

C CC CCY CCYY CCYYMM CCYYMMD CCYYMMDD2

and use the fewest prefixes possible to search for a range. For example to search for every date in the 1990's, search for 199. To search for the date range Jan 2007 up to and including Jan 2008, search for (2007 OR 200801).

A similar scheme can be used for general numerical range searching. This trades off index size for search performance.

A generalization of this is available in NumericRangeQuery in Lucene Core 2.9. This works by mapping values to be indexed to a 64 bit long value, and by indexing various length prefixes of these 64 bit values. Order preserving mappings for dates and floating points are available. See SearchNumericalFields.

When longer dates or numbers need to be indexed, for example CCYYMMDDhhmmss with hours, minutes and seconds added, consider indexing the hhmmss separately, possibly with hierarchical prefixes themselves. A search with CCYYMMDDhhmmss accuracy would then need a BooleanQuery with required clauses for the CCYYMMDD and hhmmss parts.