Solr 'n Stuff

Open source search and analytics

Follow @lucene_solr

Recent Posts

The Unofficial Solr Guide

Solr Query Syntax

Solr Query Syntax

The default Solr query syntax used to search an index uses a superset of the Lucene query syntax.

Trying a basic query

The main query for a solr search is specified via the **q** parameter. Standard Solr query syntax is the default (registered as the "lucene" query parser).

[illegible]

If this is new to you, please check out the Solr Tutorial.

Adding **debug=query** to your request will allow you to see how Solr is parsing your query.

http://localhost:8983/solr/query?debug=query&q=hello

```
{
  "responseHeader":{
    "status":0,
    "QTime":0,
    "params":{
      "q":"hello",
      "debug":"query"}},
  "response":{"numFound":0,"start":0,"docs":[]
  },
  "debug":{
    "rawquerystring":"hello",
    "querystring":"hello",
    "parsedquery":"text:hello",
    "parsedquery_toString":"text:hello",
    "QParser":"LuceneQParser"}}
```

The **`response`** section will normally contain the top ranking documents for the query. In this example, no documents matched the query.
In the debug section, one can see how the query was parsed, and the fact that **`text`** was used as the default field to search.

Basic Queries

A "term" query is a single word query in a single field that must match exactly.
In this example **`text`** is the field name, and **`hello`** is the word we are going to match in that field.

```
text:hello
```

Phrase Query

A phrase query matches multiple terms (words) in sequence.

```
text:"yonik seeley"
```

This query **will match** text containing **`Yonik Seeley`** but will **not match** **`Yonik C Seeley`** or **`Seeley Yonik`**.

Internally, a phrase query is created when the fieldType produces multiple terms for the given value. For the example above, the fieldType splits on whitespace and lowercases the result, and we get two terms… [**yonik**, **seeley**]. If our fieldType had been **`string`**, a single term of **`yonik seeley`** would have been produced since **`string`** fields do not change values or do text analysis in any way.

Proximity Query (aka Sloppy Phrase Query)

A proximity query, is like a phrase query with a tilda (~) followed by a slop that specifies the number of term position moves (edits) allowed.

```
text:"solr analytics"~1
```

This query **will match** text containing **`solr analytics`**, **`solr faceted analytics`** (edit distance 1), and **`analytics solr`** (edit distance 1). It will not match **`solr super faceted analytics`** or [illegible] the matching positions.

Boolean Query

A boolean query contains multiple clauses. A clause may be optional, mandatory, or prohibited.

```
solr search
```

The default operator is "OR", meaning that clauses are optional. When there are no mandatory clauses, at least one of the optional clauses in a query must match for the full query to match. The example query above will thus match documents containing **`solr`** or **`search`** (or both) in the default search field.

Boolean query examples:

```
+solr +search facet -highlight     /* uses + for mandatory and - for prohibited */
solr AND search OR facet NOT highlight  /* this is equivalent to the first query */
```

Semantics: **`solr`** and **`search`** must both match, **`highlight`** must not match. **`facet`** may or may not match but will contribute to the query score if it does (i.e. the presence of the **`facet`** only affects scores of matching documents, not which documents match.)

```
+text:solr text:facet text:analytics
text:(+solr facet analytics)  /* This is equivalent to the first query *?
```

The query above just demonstrates an optional syntax for specifying multiple clauses with the same field.

```
(+title:solr +title:(analytics faceting)) OR (+body:solr +body:(analytics faceting))
```

The query above uses parenthesis for precedence. Both **`solr`** and either **`analytics`** or **`faceting`** must match in the **`title`** field or in the **body** field.

Boosted Query

Any query clause can be boosted with the **^** operator. The boost is multiplied into the normal score for the clause and will affect its importance relative to other clauses.

Boosted Query Examples:

```
text:solr^10 text:rocks
text:(solr^10 rocks)
(inStock:true AND text:solr)^123.45 text:hi
```

Range Query

A range query selects documents with values between a specified lower and upper bound. Range queries work on numeric fields, date fields, and even string and text fields.

Range Query Examples:

```
  age:[18 TO 30]       // matches age 18-30 inclusive (endpoints 18 and 30 are include
d)
  age:[18 TO 30}       // matches age>=18 and age<30
  age:[65 TO *]        // "open ended" range matches age>65
  age:[* TO *]         // matches all documents with age set
```

Range Query on Text Fields:

```
  name:[apple TO banana]  // matches *any* term/word in the name field between apple an
d banana
```

Constant Score Query

A ConstantScoreQuery is like a boosted query, but it produces the same score for every document that matches the query. The score produced is equal to the query boost. The **^=** operator is used to turn any query clause into a ConstantScoreQuery.

Constant Score Query Examples:

```
+color:blue^=1 text:shoes
(inStock:true text:solr)^=100 native code faceting
```

Filter Query

(Since Solr 5.4)
A filter query retrieves a set of documents matching a query from the filter cache. Since scores are not cached, all documents that match the filter produce the same score (0 by default). Cached filters will be extremely fast when they are used again in another query.

Filter Query Example:

```
description:HDTV OR filter(+promotion:tv +promotion_date:[NOW/DAY-7DAYS TO NOW/DAY+1DA
Y])
```

The power of the **`filter()`** syntax is that it may be used anywhere within a lucene/solr query syntax. Normal **`fq`** support is limited to top-level conjunctions. However when normal top-level **`fq`** filter caching can be used, that form is preferred.

Query Comments

(Since Solr 5.3)
One can embed comments in a query using C-style comments surrounded with **`/* */`**. Comments can be nested.

Query Comments Example:

```
description:HDTV /* TODO: +promotion:tv +promotion_date:[NOW/DAY-7DAYS TO NOW/DAY+1DAY]
*/
```

Solr 'n Stuff

Open source search and analytics

Tags

block join block join example excludeTags facet analytics Faceted Search facet functions faceting performance facet statistics field collapsing frange function queries function query geo search JSON facets lucene lucene 6 lucidworks multi-select faceting nested aggregations nested facets off-heap off-heap FieldCache pivot facets post filter result grouping scalability solr solr 4.0 solr 5 solr 6 solr facet solr facet example solr facets solr features solr garbage collection solr performance solr query solr query example solr search solr sorting performance solr spatial solr tutorial sorting performance spatial search subfacets