# Querying: Help Your Users Find What They Need

Xavier Morera
PASSIONATE ABOUT TEACHING

@xmorera www.xaviermorera.com

# Querying Is the Whole Point of Search

# Querying

Request for information

Of something a user needs

Made to the search engine

That should retrieve good results

Fast

- Or the definition of fast

# Querying: Database vs. Search Engine

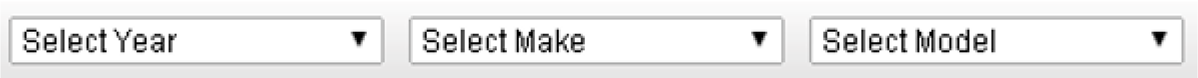
## Database

Returns rows that match for query

Ranked by a column

Cannot tune like search engine

Think in terms of “columns”



Even with indices, not as efficient

Usually does not scale without significant cost

## Search Engine

Returns relevant documents first

For those documents that match

Ways of improving relevancy

Extremely flexible search

Amazingly fast

Scale on commodity hardware

# Querying Solr from Your Code

**Scope and focus for this course**

**Create the request and manage results**

- Different types of queries
- Directly from your .NET code

**But some things are done from Solr side**

- Configure how Solr responds to requests

# Understanding Solr Configuration

**Solrconfig.xml**

- Most parameters to configure Solr

**Listeners, managing HTTP communications, Admin UI, legacy replication, …**

**Process requests to Solr**

- Request Handlers

Demo

Solr Query Configuration Demo

M6 D0

# Demo Summary

**Solrconfig.xml to configure Solr**

**Specify how to handle requests**

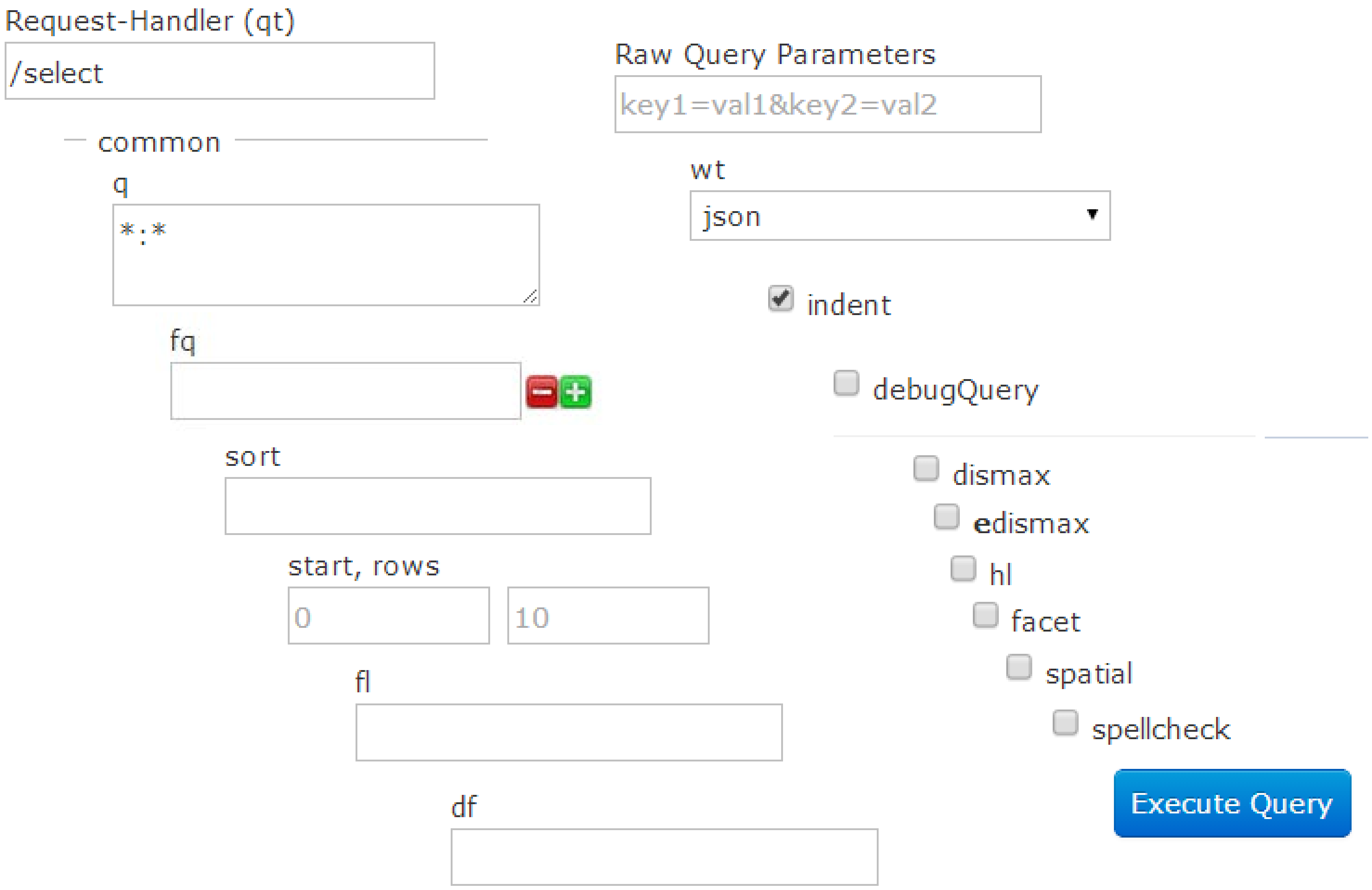
- Request handlers (/select for query)

**Include parameters**

- Defaults, Appends & Invariants

**Do some things from your app, some from Solr**

# Searching in Solr



Request-Handler (qt)
/select

common

q
*:*

fq

sort

start, rows
0 10

fl

df

Raw Query Parameters
key1=val1&key2=val2

wt
json

☑ indent

□ debugQuery

□ dismax
□ edismax
□ hl
□ facet
□ spatial
□ spellcheck

# Querying

Query

Phrase Queries

Proximity Queries

Filter Queries

Range Queries

Sorting & Paging

```
SolrQuery q = new SolrQuery(“General Charles");

var courses = solr.Query(q);
```

# Query, Phrase Query & Proximity Query

**Query for General Charles** ***(hypothetical library)***

**q=General Charles for Solr, on specified fields**

- Returns documents with the words General and Charles
- Results can contain only one of the words, i.e. General OR Charles

# About Queries

- Boolean operators: We just queried General AND Charles
  - Returned documents must contain both terms
  - AND is binary operator and has precedence
    - Can be replaced by +General +Charles
    - Unary operator
  - If we use OR it will match any term
  - Exclude from search with NOT (-)
  - Combine terms (grouping)

    PFDepth OR Babel AND Started

    (PFDepth OR Babel) AND Started

```
SolrQuery q = new SolrQuery(“General AND Charles");

var courses = solr.Query(q);
```

# Query, Phrase Query & Proximity Query

- What if you wanted more precise results?
  - Returns documents with both General and Charles
  - Narrows down results
  - Other options: default operator q.op and mm

```
SolrQuery q = new SolrQuery(@““”General Charles“”");

var courses = solr.Query(q);
```

# Query, Phrase Query & Proximity Query

**Exact matches**

**Returns documents with the phrase General Charles**

- But won’t match ...General .NET Charles...

**What other options do I have?**

```
new SolrQuery(@“““”General Charles“”~3");
```

- **Use proximity**
  - **Returns documents with both General and Charles**
    - Near each other by 3 words

```
new SolrQuery(@“General Charl*");
```

- **Wildcards**
  - Can match documents with **Charlie**

```
new SolrQuery(“General") +
     new SolrQuery (“Charles ");
```

- **Overloaded operators like &&, ||**
  - **Other possibilities include SolrQueryInList and many more**

```
New SolrQuery(SolrQuery.All)
```

- **Query for all values**

```
var q = new SolrQueryByField("author", “Xavier Morera”);

var courses = solr.Query(q);
```

## Query, Phrase Query & Proximity Query

**Query() has several overloads , i.e. by specific fields**

**Query for text in specific fields, handling special character escaping**

**q=author:("Xavier Morera")**

```
var qOpt = new QueryOptions {
    FilterQueries = new ISolrQuery[] {
        new SolrQueryByField("author", queryFromUser)
    }};
var courses = solr.Query(q, qOpt);
```

# Filter Queries

**Difference between query and filter query**

- With q you affect ranking
- With fq filter results and uses caching

**fq=author:("Xavier Morera")**

```
var qOpt = new QueryOptions {
    FilterQueries = new ISolrQuery[] {
         new SolrQueryByRange<DateTime>("releasedate",
            DateTime.Now.AddYears(-1),DateTime.Now)
    }};
var courses = solr.Query(q, qOpt);
```

## Range Queries

**Specify ranges for searching on specific fields**

fq=releasedate:[2015-05-19T14:29:01.665Z TO 2016-05-19T14:29:01.665Z]

**Works with date, int, text, ...**

**Can also query for any value in range using** SolrHasValueQuery(“author”)

```
var qOpt = new QueryOptions {
    Rows = 10,
    StartOrCursor = new StartOrCursor.Start(10),
    OrderBy = new[] {new SortOrder(“releasedate", Order.DESC)}
    }};
var courses = solr.Query(q, qOpt);
```

# Paging & Sorting

**Paging using Rows and Start (StartOrCursor)**

**Order by fields**

```
SolrQuery q = new SolrQuery(@“General Charles",
   new QueryOptions {
        Fields = new[] {“coursetitle", “courseid"}
   });
var courses = solr.Query(q);
```

# Tip: Get Only the Fields You Need

Important for performance, especially with long fields

Or documents with many fields that are not required in results

Remember: search engines are made for speed!

Demo

## Creating the Foundation: Search Library

M6 D1

# Demo Summary

**Created the foundation of a search library**

- Modularized functionality
- Models

**Console application for testing**

**Indexing application**

# Takeaway

Querying is the whole point

Fulfill a user need: query

Database search vs. search engine

Solrconfig.xml

Query, phrase query, proximity, filter queries, range queries, sorting and paging

Fields