Executing Queries in Azure Search

Chad Campbell
INDEPENDENT SOFTWARE ENGINEER
@chadcampbell www.ecofic.com

Overview

Running Queries

Build Simple Queries

Create Full Queries

Get Query Statistics

Prerequisites

Familiar with Search Indexes

Familiar with Search Documents

Beer Data

beers.md

breweries.md

Running Queries

Search Index

```
…/beers?api-version=2015-02-28

…/beers/docs?api-version=2015-02-28
```

Query Methods

GET

POST

GET approach

pass parameters by query string

must URL-encode each parameter value

length limited to 8kb

POST approach

use up to 16 MB of characters

easier to read parameters

pass parameters in request body

URL-encoding is not needed

Searching with Simple Queries

Simple Query Operators

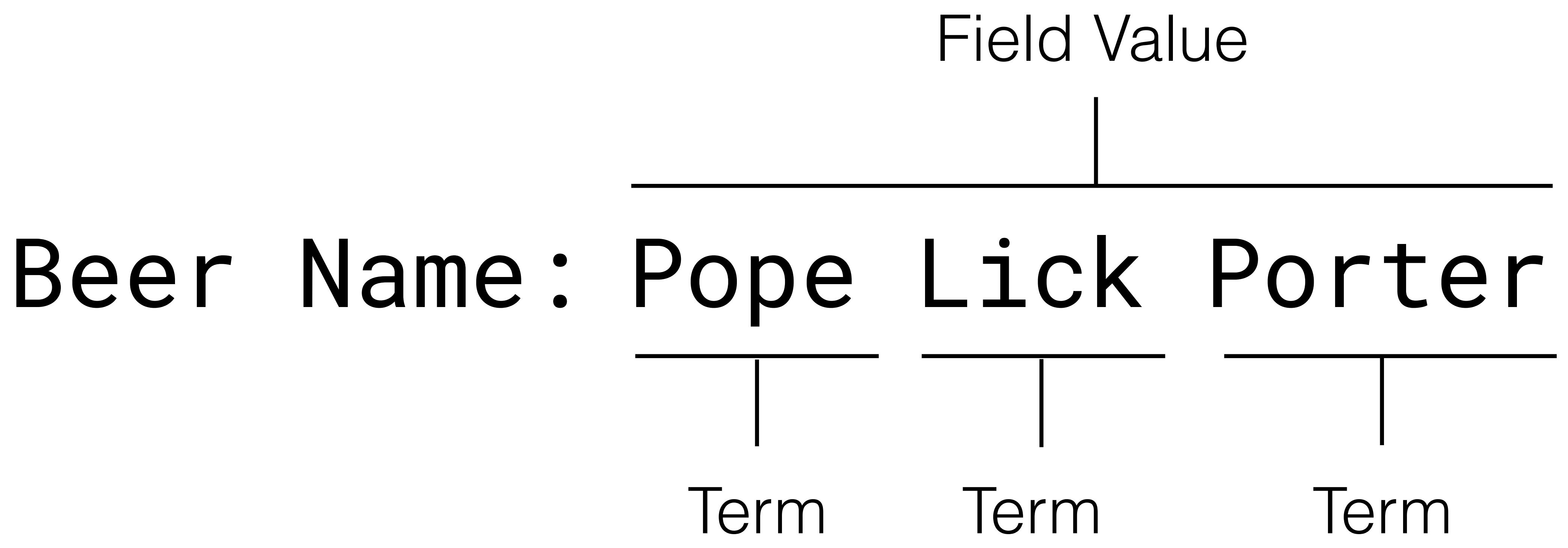
*	star
+	and
|	or
“…”	phrase
(…)	parenthesis
-	not

Suffix (*) Operator

Used after a phrase

Represents other characters

Terms and Fields



Suffix (*) Operator

`search` **parameter defaults to "*"**

OR (|) Operator

matches either term beside “|”

OR Syntax

Pipe Syntax	Default Syntax (no pipe)
ex: porter | stout	ex: porter stout
Easier to read	Typically what users will enter

AND (+) Operator

find documents with multiple terms

Operator Questions

1. Can I use multiple operators in a single query?

2. What takes precedence?

Operator Precedence

Parentheses

Exponents

Multiply

Divide

Add

Subtract

Simple Query Order of Operations

right to left

Query Comparison

12 results	**6 results**

Precedence (...) Operator

controls order of operations

useful for complex searches

helps with readability

NOT(-) Operator

excludes terms from searches

Example Queries

-stout

-(porter|stout)

-pale ale

-pale ale interpretations

Give me everything that does *not* have the word “pale” *or* has the word “ale”

Give me everything that does *not* have the word “pale” but *may* have the word “ale”

searchMode

Dictates whether terms with the NOT operator get ANDed or ORed with other terms in a query.

searchMode Parameter

Can be set to “all”

Simple queries are great for keyword searches

Searching with Full Queries

Overview

Handle Misspellings

Boost Terms

Regular Expressions

queryType Parameter

specifies which query language to use

defaults to “simple”

more powerful queries require “full”

Syntactical Differences

star operator

AND operator

NOT operator

wildcards

? - placeholder for a single character

* - placeholder for zero or more characters

Star (*) Character

cannot be used as first character of a query

can be used in the middle of terms

searchMode Parameter

default approach acts like an “OR”

“all” behaves like an “AND”

Field-Level Queries

Search at a more granular level

Restricts searches to specific fields

Mobile Usage

People search more from mobile devices than from their computer

Mobile searches are more likely to be inaccurate

Fuzzy Searches

Adds forgiveness to a query

Proximity Searches

Help you find words that are relatively close to each other

Term Boosting

For terms more important than others

Regular Expressions

Great way to find terms that match patterns

String Matching

Ch*	*col*	*late
$\underline{\text{Ch}}\text{ocolate}$	$\text{Cho}\underline{\text{col}}\text{ate}$	$\text{Choco}\underline{\text{late}}$
StartsWith	**Contains**	**EndsWith**

```
search: /…/
```

Regular Expression

Syntax

```
{
  "search":"name:/.*col.*/",
  "queryType":"full",
  "select":"id, name"
}
```

Infix Query Example

Regular Expression

```
{
  "search":"name:/.*er/",
  "queryType":"full",
  "select":"id, name"
}
```

Suffix Query Example

Regular Expression

Full Queries

Regular Expressions

Term Boosting

Proximity Searches

Fuzzy Searches

Field-Level Queries

Getting Query Statistics

Getting Query Statistics

Query duration

Result count

count Parameter

Specifies whether to include total number of results

count is in the @odata.count property

value is an approximation

result counts

`value.length` **- number of results on the current page**

`@odata.count` **- total number of results across the index**

count Parameter

Defaults to false

Small, negative performance impact when true

Getting Query Duration

Not built-in to Azure Search

Takes time to send a query to Azure Search

Takes time to get results from Azure Search

```
var queryStartTime = GetCurrentMilliseconds();
var results = ExecuteSearch();
var queryDuration = GetCurrentMilliseconds() -
  queryStartTime;
```

Get Query Duration

Pseudocode

```
function onSearchComplete() {
  var duration = new Date().getTime() -
    viewModel.searchStartedAt;
  $('#queryDuration').text(duration + ' ms’);
  ...
}
```

Get Query Duration

Growler Code

Summary

Query Statistics

Full Queries

Simple Queries

Running Queries