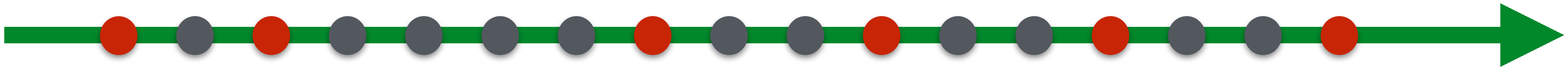
Apache Lucene

Adrien Grand

What is Lucene?

- An information retrieval library
  - Can be used to build search apps
  - Not a runtine, use Solr or Elasticsearch
- Written in Java
- Developed at the Apache Software Foundation
  - Contributors include IBM, Twitter, Elastic, Lucidworks, …

History



1999: creation on Sourceforge

2001: moved to the ASF

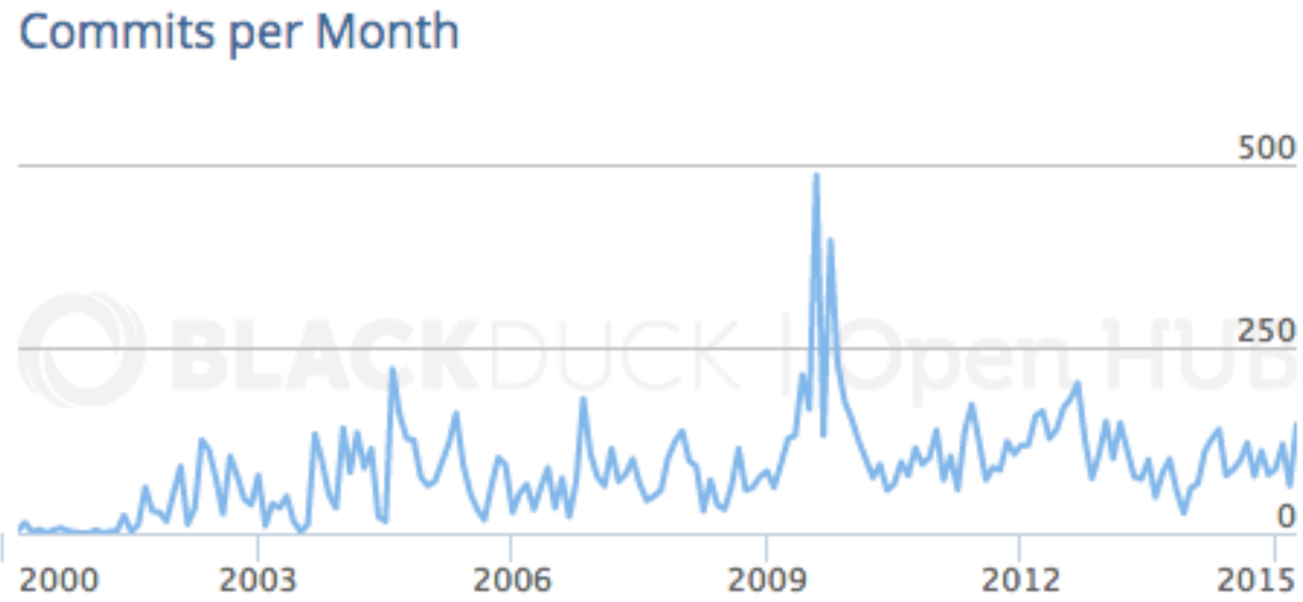
May 2006 2.0 release

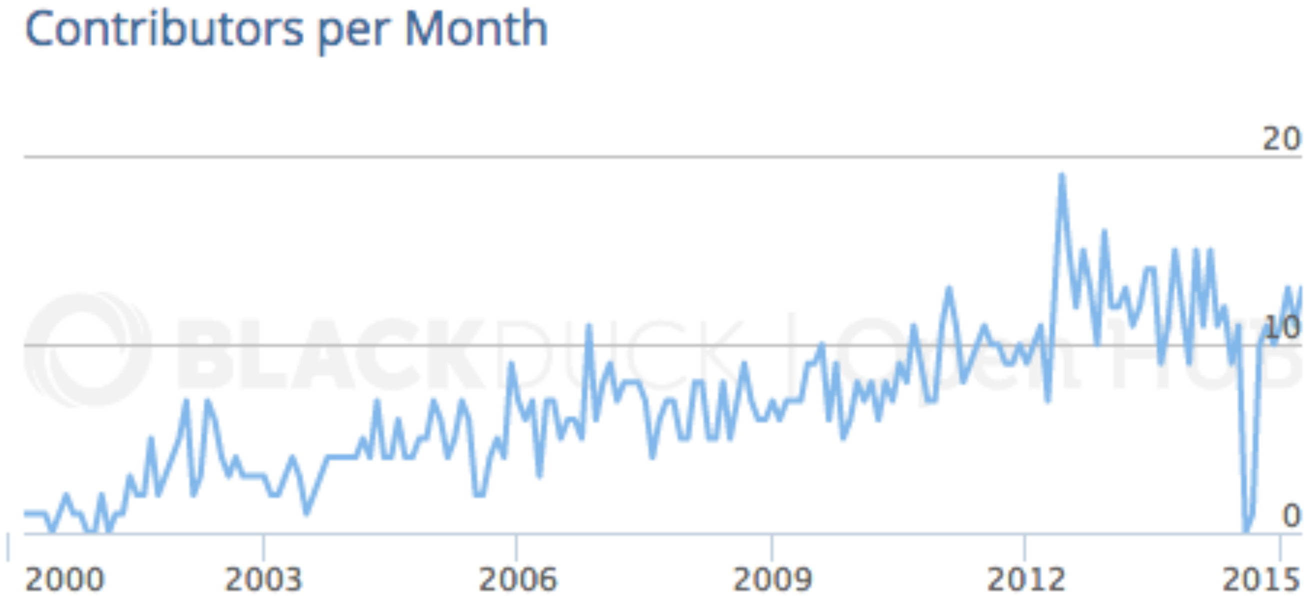
November 2009 3.0 release

October 2012 4.0 release

February 2015 5.0 release

Activity





source
https://www.openhub.net/p/lucene

Features

- Full-text search
- Structured search
- Highlighting
- Faceting
- Suggestions

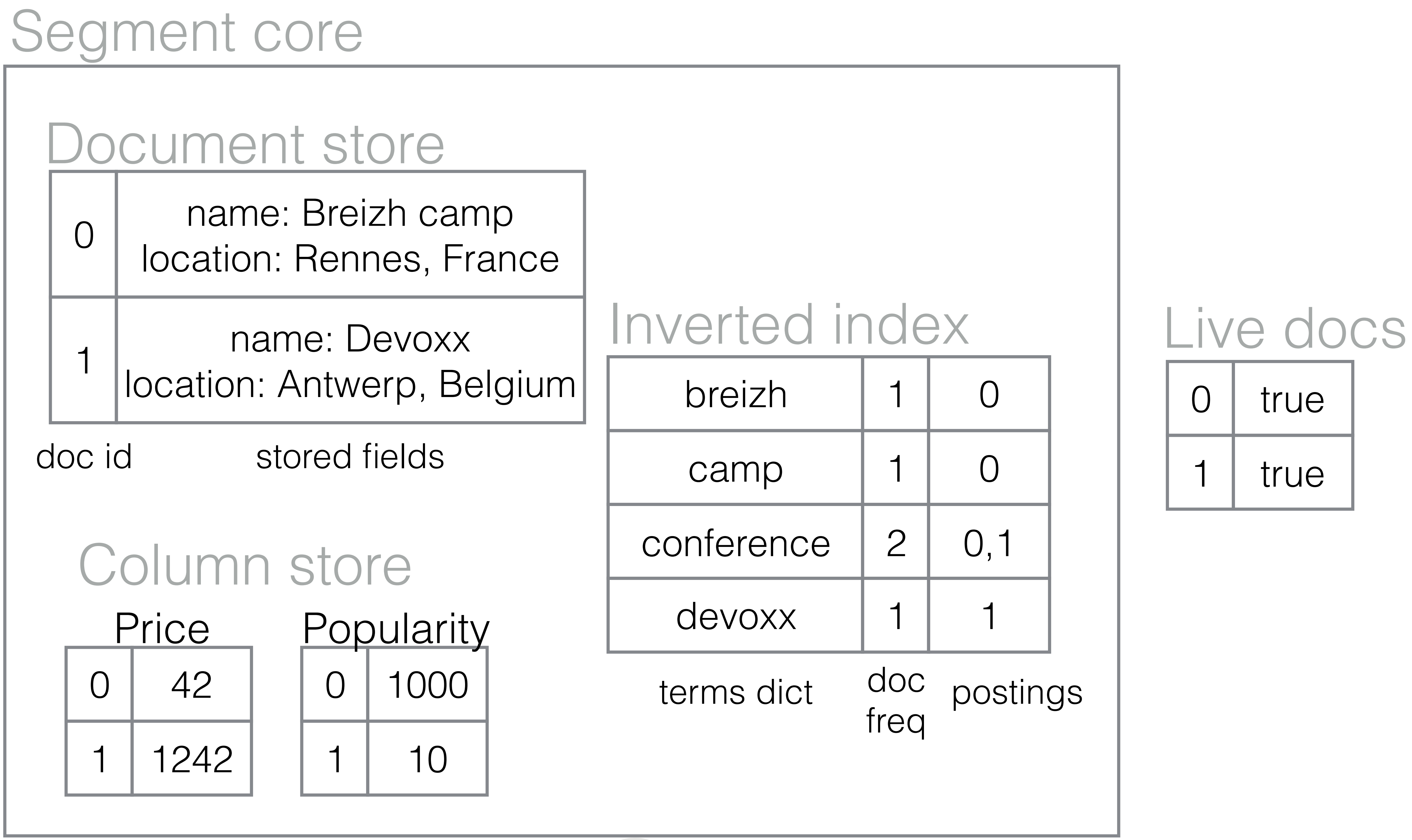
Design

- Embeds
  - an inverted index, for efficient query execution
  - a document store, to get original data back
  - a column store, for sorting and analytics

More history

- Lucene 3.4 Added a faceting module
- Lucene 4.0: Added a column store to the index
- Lucene 4.1: More efficient structured search
- Lucene 4.1: More efficient PK lookups
- Lucene 4.1: Built-in compression of the doc store
- Lucene 4.5: Column store moved from memory to disk
- Lucene 4.8: Checksums on all index files
- Lucene 5.1: Better query execution plans with 2-phases iterators

Design



Segment core

Document store

doc id	stored fields
0	name: Breizh camp location: Rennes, France
1	name: Devoxx location: Antwerp, Belgium

Column store

Price

0	42
1	1242

Popularity

0	1000
1	10

Inverted index

terms dict	doc freq	postings
breizh	1	0
camp	1	0
conference	2	0,1
devoxx	1	1

Live docs

0	true
1	true

Design

- Index divided into immutable segments
- To add more documents, add more segments
- In-place updates are not supported
  - To update documents, delete then add

Merging

- Background merges
  - Keep the number of segments low for fast search
  - Reclaim space from deleted documents

Merging

- Writing/Merging segments is expensive
  - IndexWriter buffers pending docs in memory
- Refresh/Reopen:
  - Flush in-memory buffer into a segment
  - Make segment searchable
- Commit
  - Flush in-memory buffer to a segment
  - “fsync” data to disk

Index safety

Only data which has been committed is safe.

If you need better safety, write the data somewhere else too: other database, transaction log, …

Advices

- Don’t give all machine memory to Java
    - Performance factor #1 is the filesystem cache
- Reopen asynchronously, typically every X seconds
- Batch writes before committing

Pros/cons

- Fast search
- Cross-field index intersections
    - On the contrary to many databases!
- Powerful combinations of features
    - Run facets on docs that match a particular query

- Not realtime
    - Yet “near” realtime
- No fine-grained updates
- Ingestion speed
    - Yet fast enough for most use-cases
- Disk usage: data is duplicated for each access pattern

Backward compatibility

- Version N can read indices of version N-1
- Public API: minor versions are backward compatible
  - IndexWriter, IndexSearcher, Query, Document, …
  - Unless we discover API is trappy
- Internal/Experimental APIs **will** break
  - Collector, Scorer, Comparator, …

SimpleText

```
IndexWriterConfig iwConfig = new IndexWriterConfig(new WhitespaceAnalyzer());
iwConfig.setCodec(new SimpleTextCodec());
try (Directory dir = FSDirectory.open(new File("/tmp/my_index").toPath());
    IndexWriter writer = new IndexWriter(dir, iwConfig)) {

  Document document = new Document();
  document.add(new TextField("name", "Breizh C@mp", Store.YES));
  document.add(new TextField("desc", "la conférence des développeurs du Grand Ouest", Store.NO));
  document.add(new StoredField("location", "Rennes, France"));
  document.add(new NumericDocValuesField("founded_year", 2011));
  writer.addDocument(document);

  document = new Document();
  document.add(new TextField("name", "Devoxx France", Store.YES));
  document.add(new TextField("desc", "la conférence des développeurs passionnés", Store.NO));
  document.add(new StoredField("location", "Paris, France"));
  document.add(new NumericDocValuesField("founded_year", 2012));
  writer.addDocument(document);

  writer.commit();

  document = new Document();
  document.add(new TextField("name", "Riviera DEV", Store.YES));
  document.add(new TextField("desc", "la conférence des développeurs du Sud Est", Store.NO));
  document.add(new StoredField("location", "Sophia-Antipolis, France"));
  document.add(new NumericDocValuesField("founded_year", 2009));
  writer.addDocument(document);

  writer.commit();
}
```

```
% ls /tmp/my_index
_0.scf
_0.si
_1.scf
_1.si
segments_2
```

```
% cat _0.si
  version 6.0.0
  number of documents 2
  uses compound file true
  diagnostics 8
    key os
    value Linux
    key java.vendor
    value Oracle Corporation
    key java.version
    value 1.8.0_25
    key lucene.version
    value 6.0.0
    key os.arch
    value amd64
    key source
    value flush
    key os.version
    value 3.13.0-53-generic
    key timestamp
    value 1434102490791
  attributes 0
  files 2
    file _0.si
    file _0.scf
  id ??hFq? E?q??h??
checksum 00000000001526513595
```

```
% cat _0.scf
cfs entry for: _0.dat
field founded_year
  type NUMERIC
  minvalue 2011
  pattern 0
0
T
1
T
END
checksum 00000000003242224815
[…]
```

```
cfs entry for: _0.fld
doc 0
  field 0
    name name
    type string
    value Breizh C@mp
  field 2
    name location
    type string
    value Rennes, France
doc 1
  field 0
    name name
    type string
    value Devoxx France
  field 2
    name location
    type string
    value Paris, France
END
checksum 00000000002801255432
```

```
cfs entry for: _0.pst
field desc
  term Grand
    doc 0
      freq 1
      pos 5
  term Ouest
    doc 0
      freq 1
      pos 6
  term conférence
    doc 0
      freq 1
      pos 1
    doc 1
      freq 1
      pos 1
[…]
END
checksum 00000000002149012390
```

Thank you!

@jpountz