Apache Lucene 6

What's coming next?

Uwe Schindler
Apache Software Foundation | SD DataSolutions GmbH | PANGAEA

@thetaph1 · uschindler@apache.org

My Background

- **Committer** and **PMC member** of **Apache Lucene and Solr** - main focus is on development of Lucene Core.
- Implemented fast numerical search and maintaining the new attribute-based text analysis API. Well known as *Generics and Sophisticated Backwards Compatibility Policeman*.
- **Elasticsearch** lover.
- Working as consultant and software architect at **SD DataSolutions GmbH** in Bremen, Germany.
- Maintaining **PANGAEA** (Publishing Network for Geoscientific & Environmental Data) where I implemented the portal's geo-spatial retrieval functions with Apache Lucene Core and Elasticsearch.

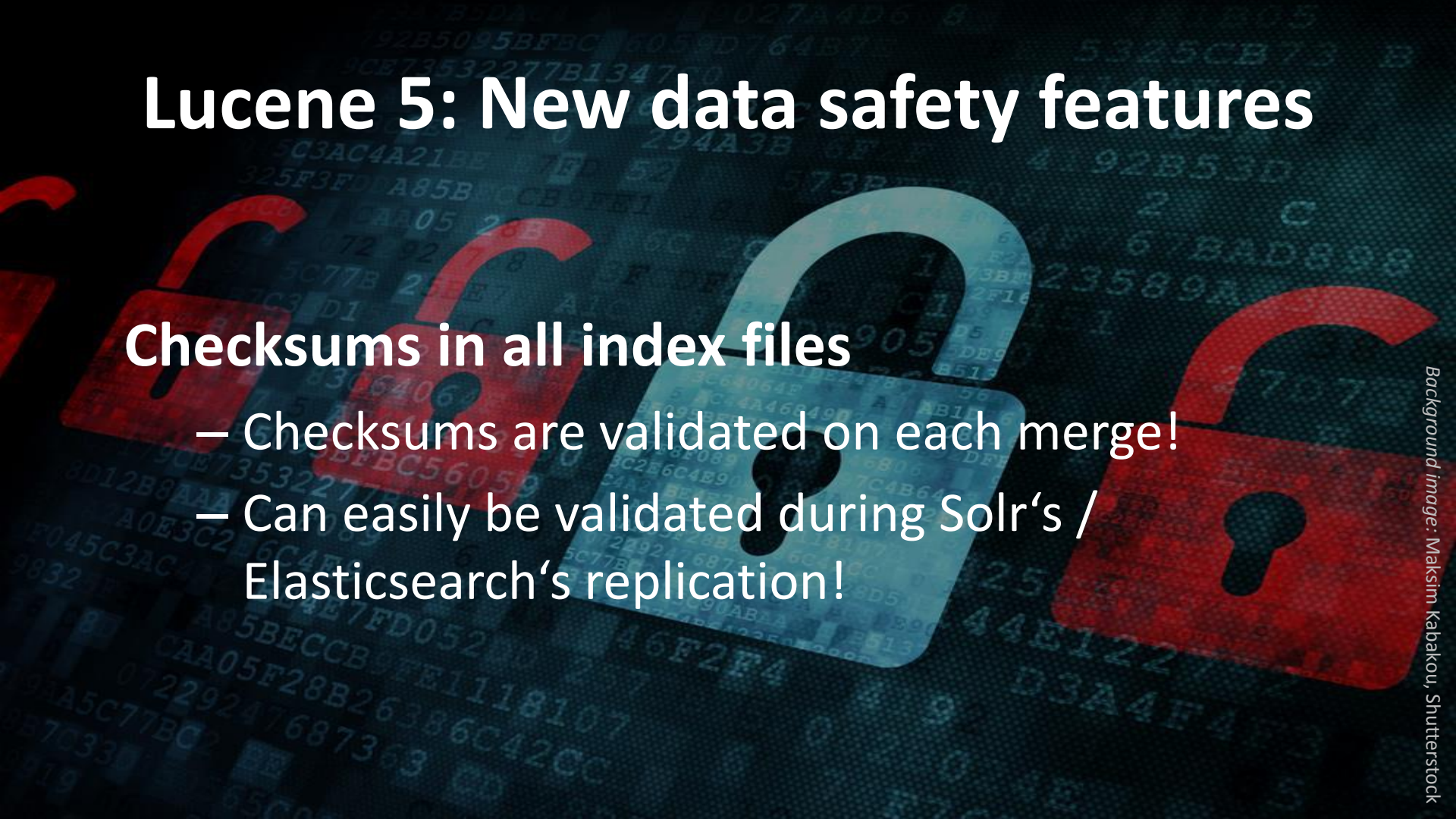
History

ON THE WAY TO Lucene 6…

Lucene 5: New data safety features

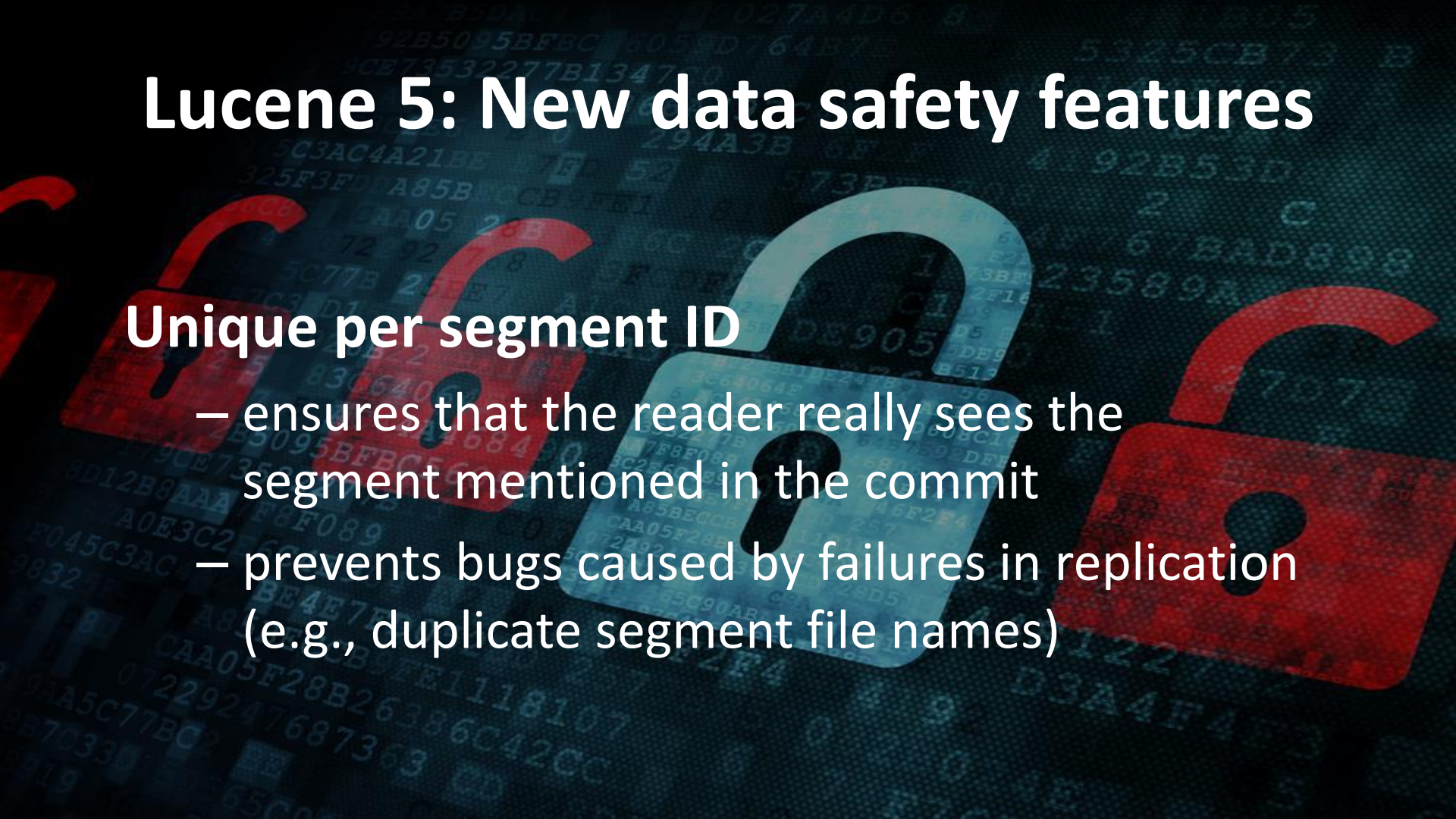
Checksums in all index files

- Checksums are validated on each merge!
- Can easily be validated during Solr‘s / Elasticsearch‘s replication!



Background image: Maksim Kabakou, Shutterstock

Lucene 5: New data safety features



Unique per segment ID

- ensures that the reader really sees the segment mentioned in the commit
- prevents bugs caused by failures in replication (e.g., duplicate segment file names)

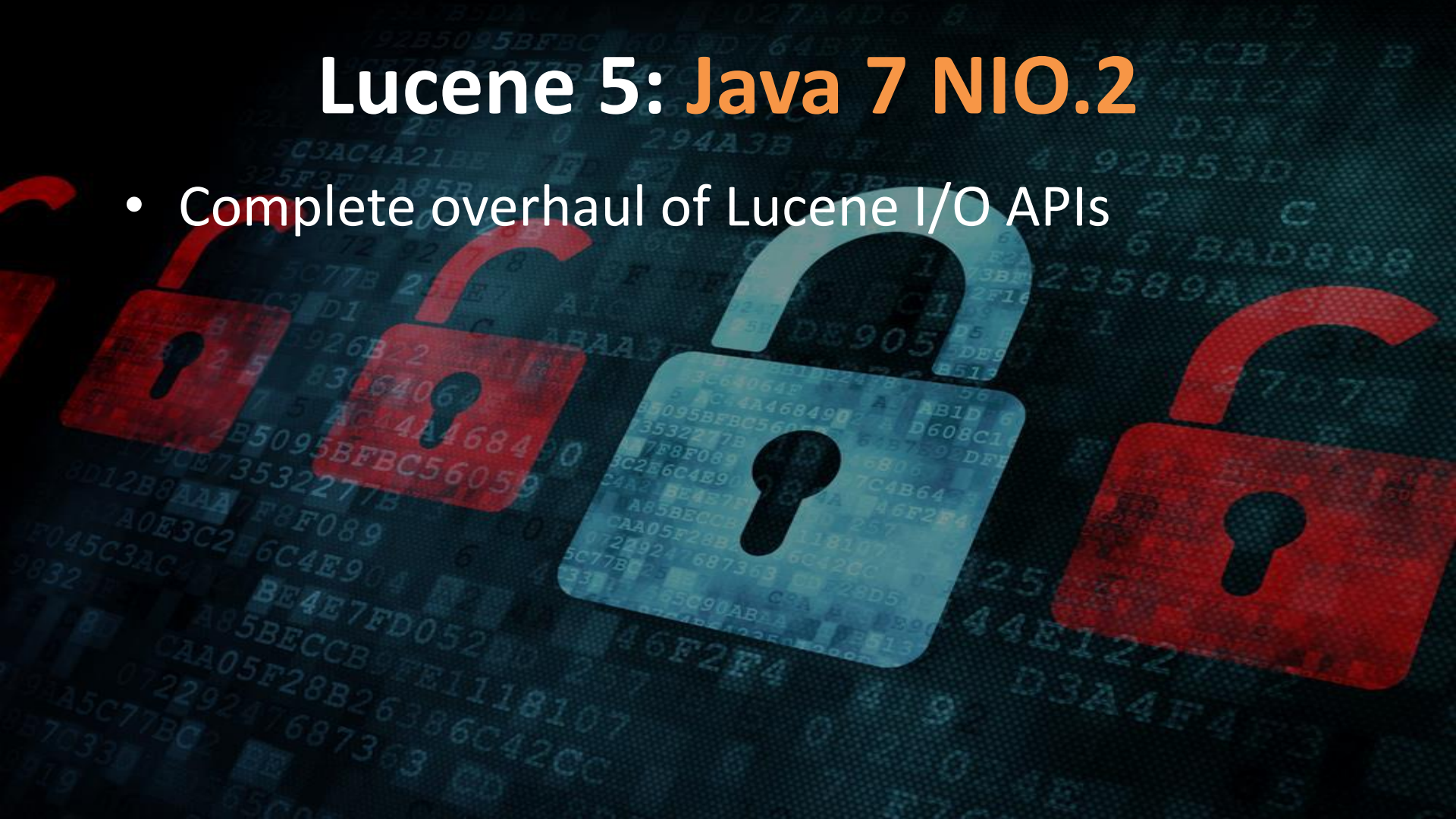
Lucene 5: New index safety features

Cutover to NIO.2

(Java 7, JSR 203)

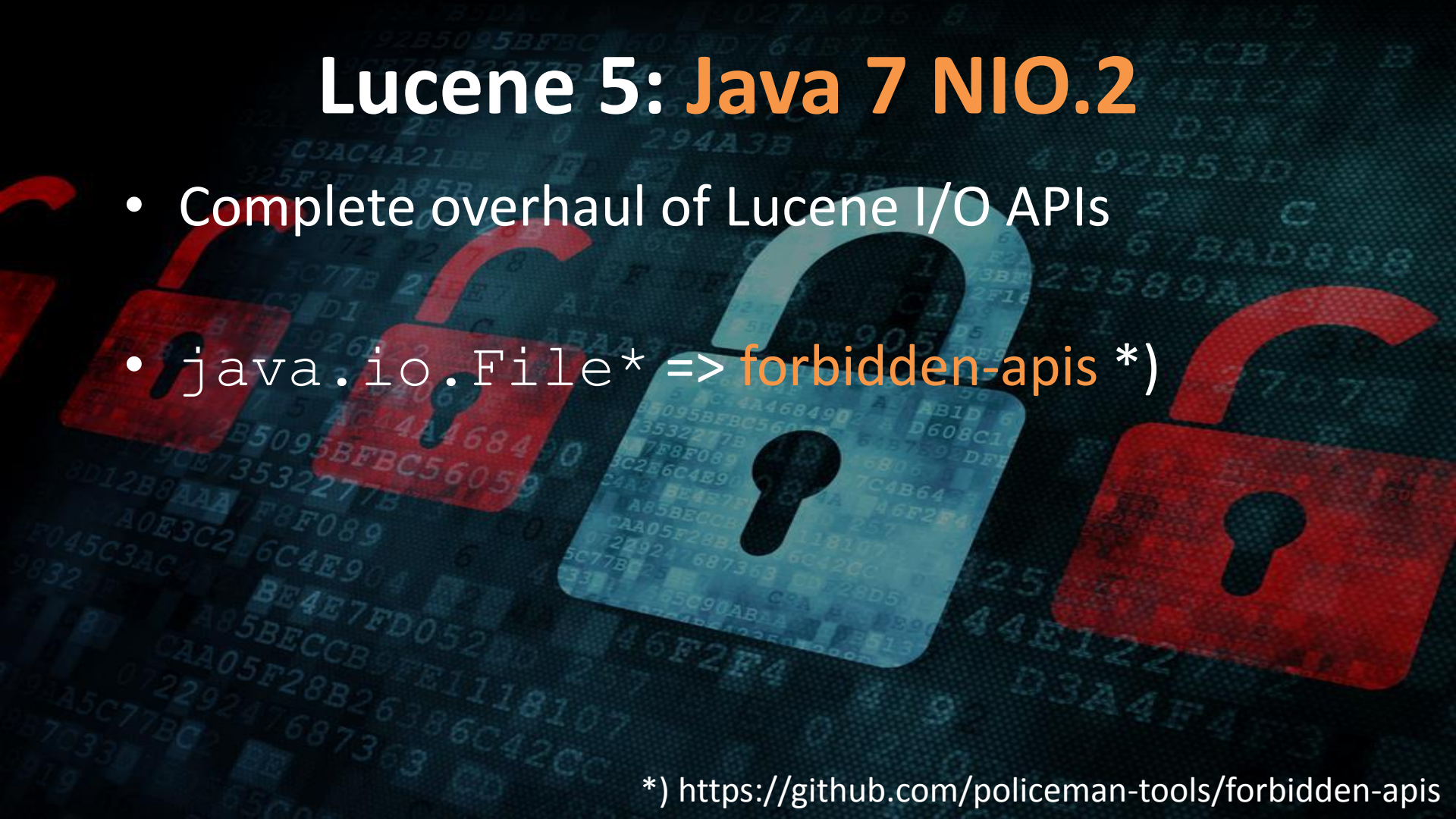
Lucene 5: Java 7 NIO.2

- Complete overhaul of Lucene I/O APIs



Lucene 5: Java 7 NIO.2

- Complete overhaul of Lucene I/O APIs
- `java.io.File*` => forbidden-apis *)



*) https://github.com/policeman-tools/forbidden-apis

Lucene 5: Java 7 NIO.2

- Complete overhaul of Lucene I/O APIs
- `java.io.File*` => forbidden-apis *)
- Atomic rename to publish commit
  - no more `segments.gen`
  - `fsync()` on directory metadata

*) https://github.com/policeman-tools/forbidden-apis

Lucene 5: Overhaul of Codec API

- Pull APIs throughout Codec components
  - E.g., PostingsFormat
- Norms are now handled by separate codec component

Lucene 5: Index merging

Lucene 5: Index merging

- Linux: Detection if index is on SSD
  - Better default merging settings
  - Other operating systems assume spinning disks (no change)

Lucene 5: Index merging

- Linux: Detection if index is on SSD
  - Better default merging settings
  - Other operating systems assume spinning disks (no change)
- Merge Scheduler: Auto Throttling
  - Automatically controls I/O rates based on indexing/merging rate
  - Stalling under high load is more unlikely!

Lucene 5: Reduced Heap Usage

- Query Filters uses new bit set types
- `CachingWrapperFilter` replacement:
  - New, highly configureable filter cache
  - Tracks filter‘s frequency of use
  - Simplifies code in Apache Solr and Elasticsearch
- Merging uses much less heap

Lucene 5: Reduced Heap Usage

- Query Filters uses new bit set types
- `CachingWrapperFilter` replacement:
  - New, highly configureable filter cache
  - Tracks filter‘s frequency of use
  - Simplifies code in Apache Solr and Elasticsearch
- Merging uses much less heap

- Most classes now implement `Accountable`
  - Allows to query heap usage
  - Nice "tree view" on heap usage of index components

Lucene 5: Reduced Heap Usage

- Query Filters uses new bit set types
- `CachingWrapperFilter` replacement:
  - New, highly configureable filter cache
  - Tracks filter's frequency of use
  - Simplifies code in Apache Solr and Elasticsearch

```
_cz(5.0.0):C8330469: 28MB
 postings [...]: 5.2MB
 ...
 field 'latitude' [...]: 678.5KB
  term index [FST(nodes=6679, ...)]: 678.3KB
```

Lucene 5: CustomAnalyzer

- Freely configurable Analyzer
- Based on SPI framework for Tokenizers, TokenFilters and CharFilters
- Similar to Apache Solr‘s schema.xml:
  - Generic names of components (like Elasticsearch)
  - Same config options like Apache Solr
- Builder API

Lucene 5: CustomAnalyzer

- Freely configurable Analyzer

```
Analyzer ana =
CustomAnalyzer.builder(Paths.get("/path/to/config"))
  .withTokenizer(StandardTokenizerFactory.class)
  .addTokenFilter(StandardFilterFactory.class)
  .addTokenFilter(LowerCaseFilterFactory.class)
  .addTokenFilter(StopFilterFactory.class,
    "ignoreCase", "false",
    "words", "stopwords.txt",
    "format", "wordset")
  .build();
```

Die, FieldCache,… die, die, die!

- FieldCache is gone from Lucene Core

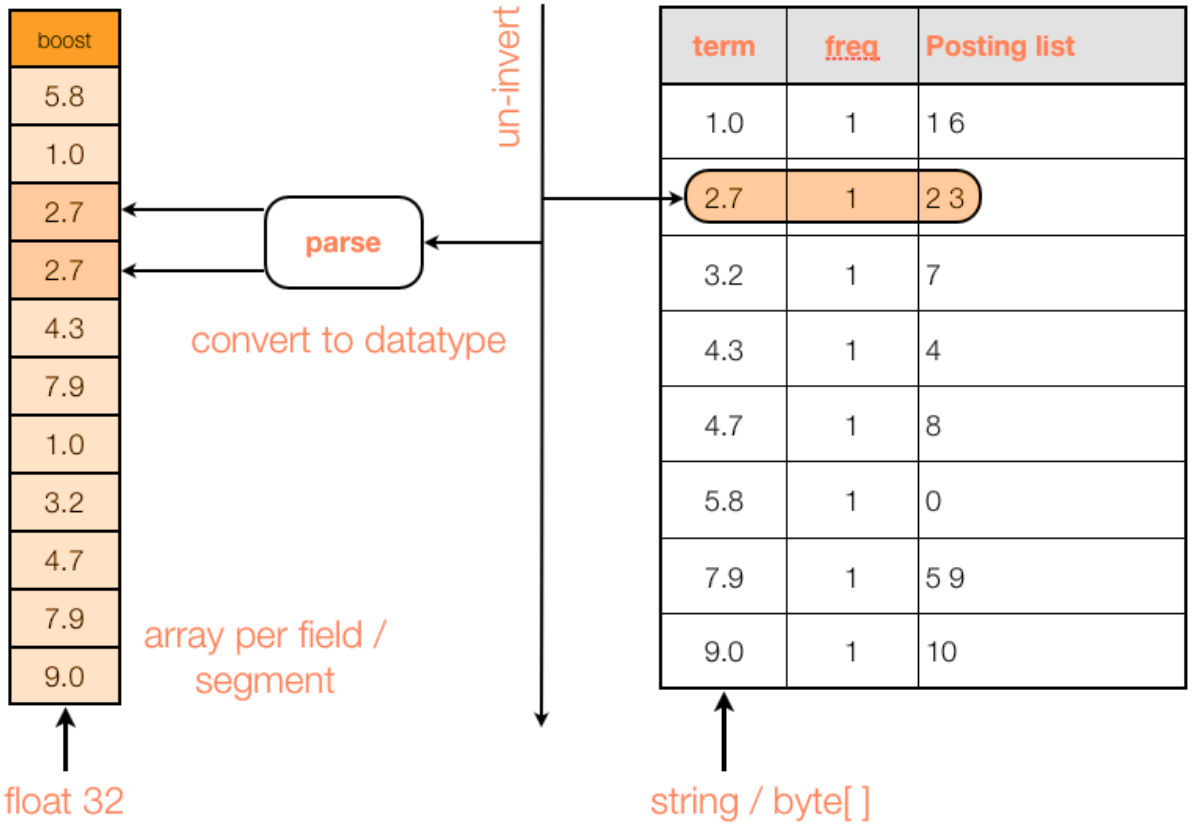


Image credits: Simon Willnauer / Trifork

Die, FieldCache,… die, die, die!

- FieldCache is gone from Lucene Core
- Use DocValues fields and APIs!

Die, FieldCache,… die, die, die!

- FieldCache is gone from Lucene Core
- Use DocValues fields and APIs!

- Not completely gone:
  - UninvertingReader in `misc/` module emulates DocValues by uninverting index
  - UninvertingReader allows to merge to a new index, automatically adding DocValues!

Lucene 5.1: Filter => Query

Lucene 5.1: Filter => Query

- Removal of Filters
  - new `Occur.FILTER` in `BooleanQuery`
  - Removed some duplicate classes already: `BooleanFilter`, `Term(s)Filter`, `NumericRangeFilter`…

Lucene 5.1: Filter => Query

- Removal of Filters
  - new `Occur.FILTER` in `BooleanQuery`
  - Removed some duplicate classes already: `BooleanFilter`, `Term(s)Filter`, `NumericRangeFilter`…

```
Query newq = new BooleanQuery.Builder()
    .add(new QueryBuilder(analyzer).createBooleanQuery("content", "some query"), BooleanClause.Occur.MUST)
    .add(new TermQuery(new Term(field, "Some Facet")), BooleanClause.Occur.FILTER)
    .build();
```

Lucene 5.1: Filter => Query

- Removal of Filters
  - new `Occur.FILTER` in `BooleanQuery`
  - Removed some duplicate classes already: `BooleanFilter`, `Term(s)Filter`, `NumericRangeFilter`…
- Backwards compatibility:
  - `Filter` extends `Query`
  - query API calls `getDocIdSet`
  - returns 0 as score (boost ignored)

Lucene 5.1: Two Phase Iterators

- Split iterators into *cheap* and *expensive* part

Lucene 5.1: Two Phase Iterators

- Split iterators into *cheap* and *expensive* part
- Used by `PhraseQuery`:
  - *Cheap* part is the „matching“ of terms (conjunction)
  - *Expensive* part is loading & checking positions

Lucene 5.1: Two Phase Iterators

- Split iterators into *cheap* and *expensive* part
- Used by `PhraseQuery`:
  - *Cheap* part is the „matching“ of terms (conjunction)
  - *Expensive* part is loading & checking positions
- Allows to share common code

Lucene 5.2: Span Queries

- **Complete rewrite**

Lucene 5.2: Span Queries

- **Complete rewrite**

- Uses Lucene 5.1 "two phase iterators"
- Shares code with `BooleanQuery` (conjunction / disjunction)

6

6

New features!

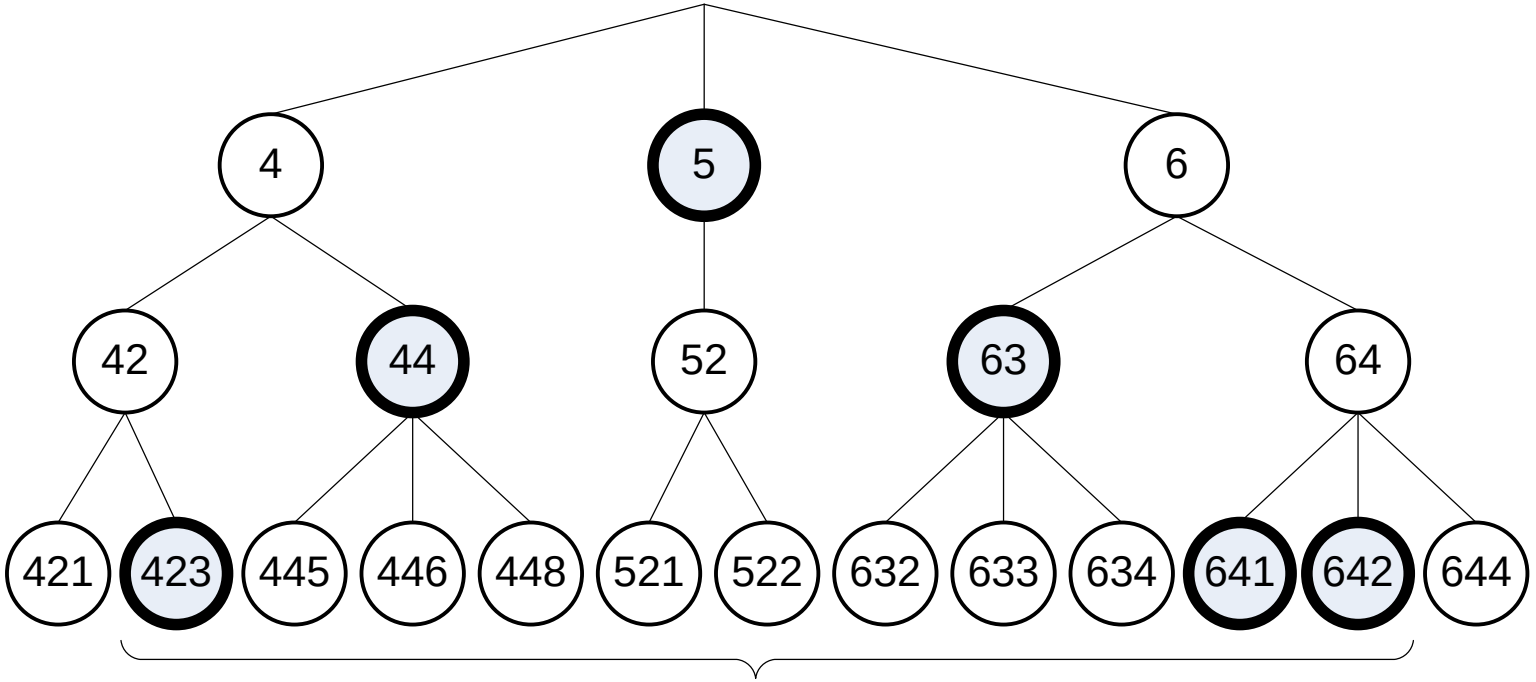
Lucene Point Values

(also known as dimensional values)

- Successor of `NumericField` (Solr: `TrieField`)
- Multidimensional (e.g. geographic coordinates): 8 dims
- Up to 128 bits / 16 bytes per value (IPv6 range queries are now possible)

See: https://www.elastic.co/blog/lucene-points-6.0

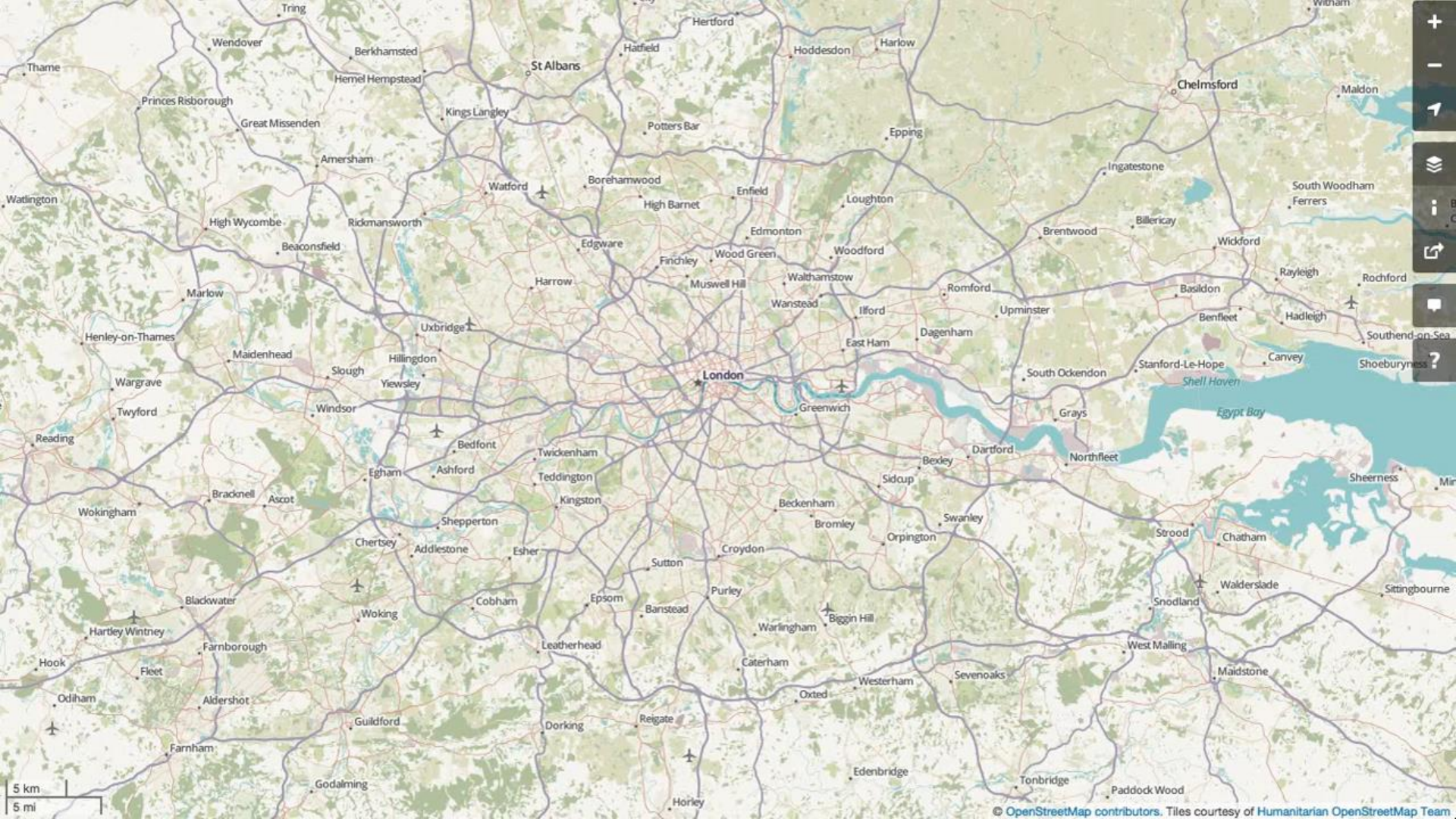
Legacy Numeric Range Queries



Block k-d-Trees

Very similar approach like **NumericField**!

- Just more dynamic
- Adapts dynamically depending on number of unique values!


Tring
Wendover
Berkhamsted
Thame
Hemel Hempstead
St Albans
Hertford
Hatfield
Hoddesdon
Harlow
Chelmsford
Maldon
Princes Risborough
Great Missenden
Kings Langley
Potters Bar
Epping
Amersham
Watford
Borehamwood
Enfield
Ingatestone
South Woodham Ferrers
Watlington
High Wycombe
Rickmansworth
High Barnet
Loughton
Billericay
Brentwood
Beaconsfield
Edgware
Edmonton
Wood Green
Woodford
Wickford
Finchley
Harrow
Muswell Hill
Walthamstow
Rayleigh
Rochford
Marlow
Romford
Basildon
Wanstead
Ilford
Upminster
Benfleet
Hadleigh
Uxbridge
Dagenham
Southend-on-Sea
Henley-on-Thames
East Ham
Maidenhead
Hillingdon
Canvey
Slough
London
South Ockendon
Stanford-Le-Hope
Shoeburyness
Wargrave
Yiewsley
Shell Haven
Twyford
Windsor
Greenwich
Egypt Bay
Grays
Reading
Bedfont
Twickenham
Dartford
Egham
Ashford
Bexley
Northfleet
Teddington
Sidcup
Sheerness
Bracknell
Ascot
Kingston
Beckenham
Wokingham
Bromley
Swanley
Shepperton
Strood
Chertsey
Addlestone
Esher
Croydon
Orpington
Chatham
Sutton
Walderslade
Purley
Sittingbourne
Blackwater
Cobham
Epsom
Snodland
Woking
Banstead
Biggin Hill
Hartley Wintney
Warlingham
Farnborough
Leatherhead
West Malling
Hook
Caterham
Maidstone
Fleet
Sevenoaks
Odiham
Aldershot
Westerham
Oxted
Guildford
Dorking
Reigate
Farnham
Edenbridge
Tonbridge
Godalming
Paddock Wood
Horley
5 km
5 mi
© OpenStreetMap contributors. Tiles courtesy of Humanitarian OpenStreetMap Team

Comparison (1D)

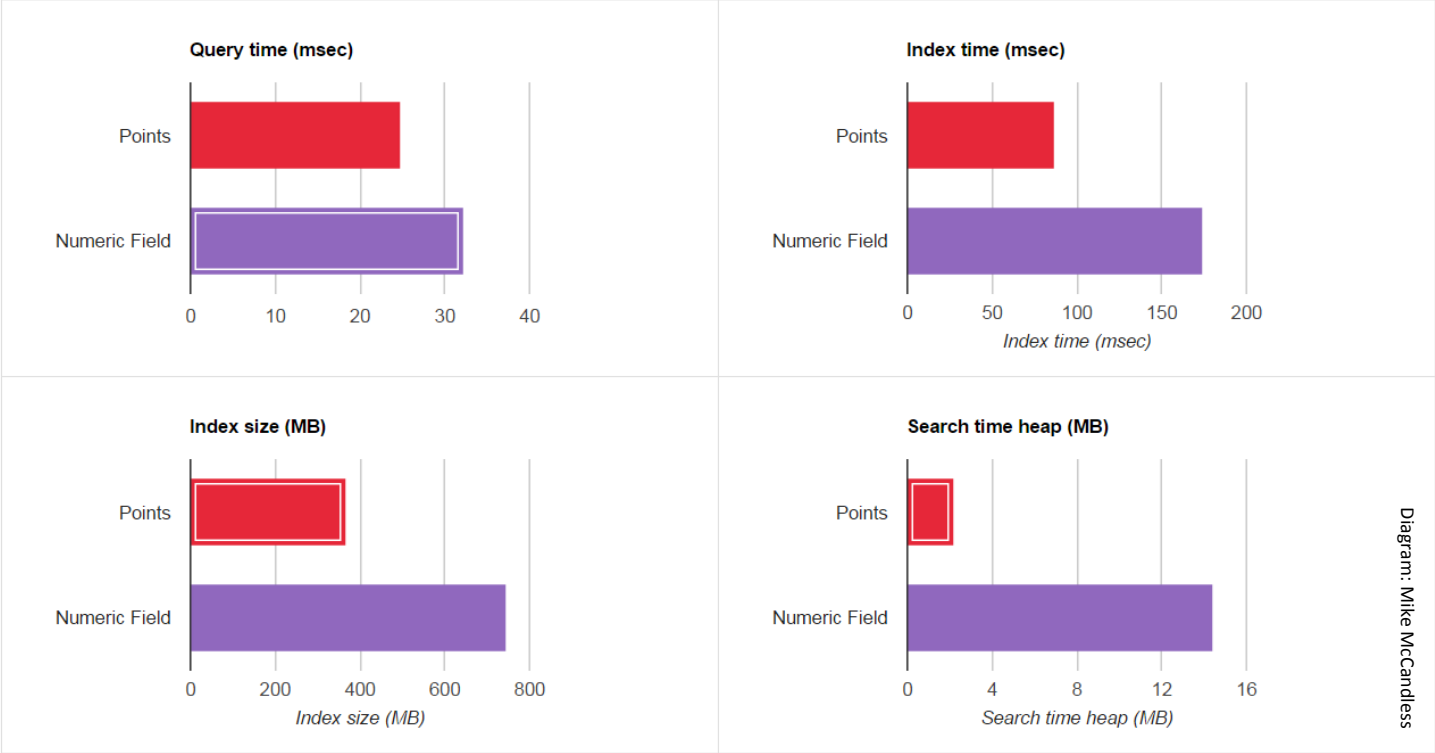


Diagram: Mike McCandless

Comparison (2D)

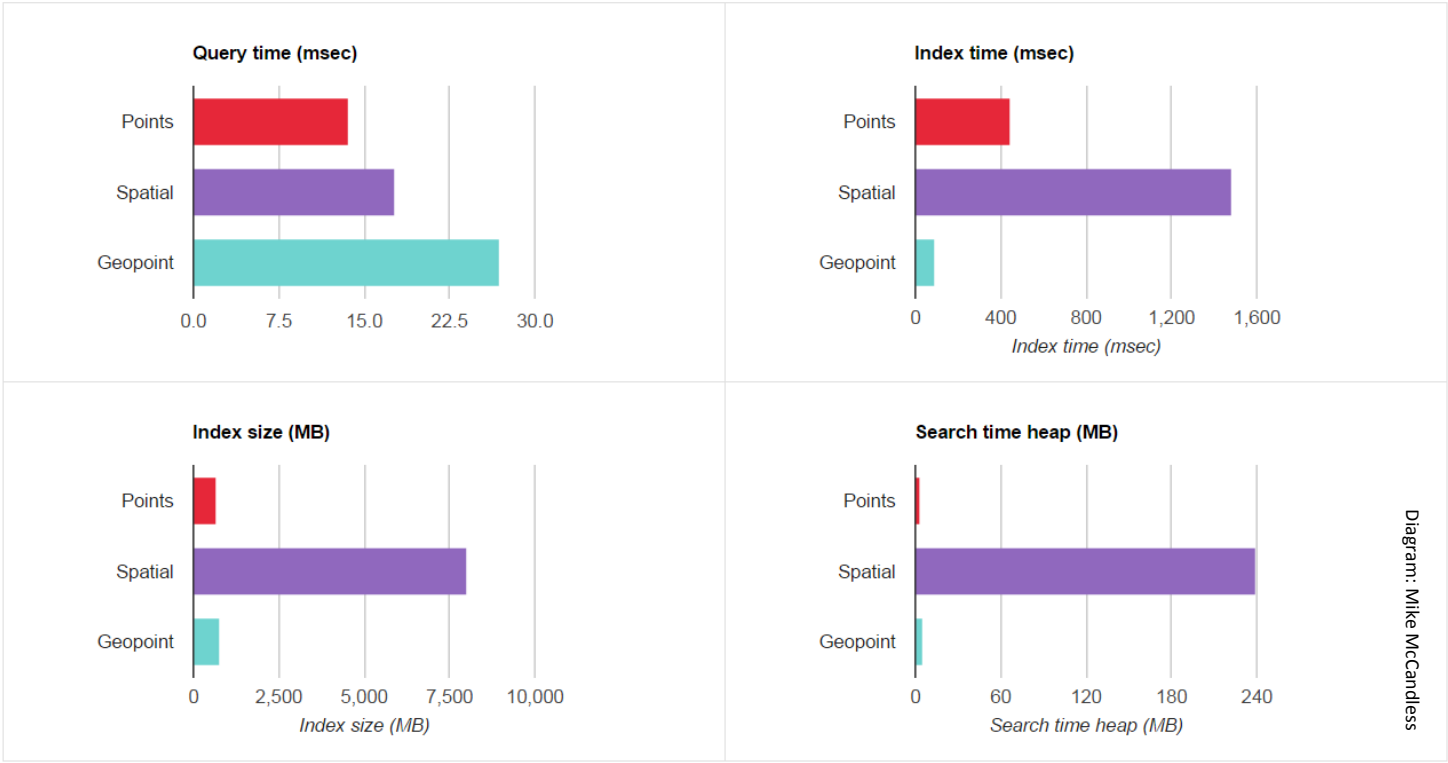


Diagram: Mike McCandless

Lucene 6: Java Version

Lucene 6: Java Version

- **Java 8** is minimum requirement!

Lucene 6: Java Version

- **Java 8** is minimum requirement!
- `lucene-core.jar` only uses *compact1* profile

Lucene 6: Java Version

- **Java 8** is minimum requirement!
- `lucene-core.jar` only uses *compact1* profile
- All other (Lucene) parts use *compact2* profile

Lucene 6: Java 9 ?

Lucene 6: Java 9 ?

- Compatibility with **Java 9** *module system* restrictions

Lucene 6: Java 9 ?

- Compatibility with **Java 9** *module system* restrictions
- **Unicode 8:** 💩 👮 🐸
  - with **ICU** or **Java 9**

Lucene 6: Java 9 ?

- Compatibility with **Java 9** *module system* restrictions
- **Unicode 8:** 💩 👮 🐸
  - with **ICU** or **Java 9**
- **Nightly tests** with *early access builds*
  - currently **Java 9 build 121**

Lucene 6: Okapi BM25

Lucene 6: Okapi BM25

It’s now the **default**!

BTW: What‘s Okapi BM25 ???

BTW: What‘s Okapi BM25 ???

- BM25 is a bag-of-words retrieval function

BTW: What‘s Okapi BM25 ???

- BM25 is a bag-of-words retrieval function
- **probabilistic model** instead vector space model

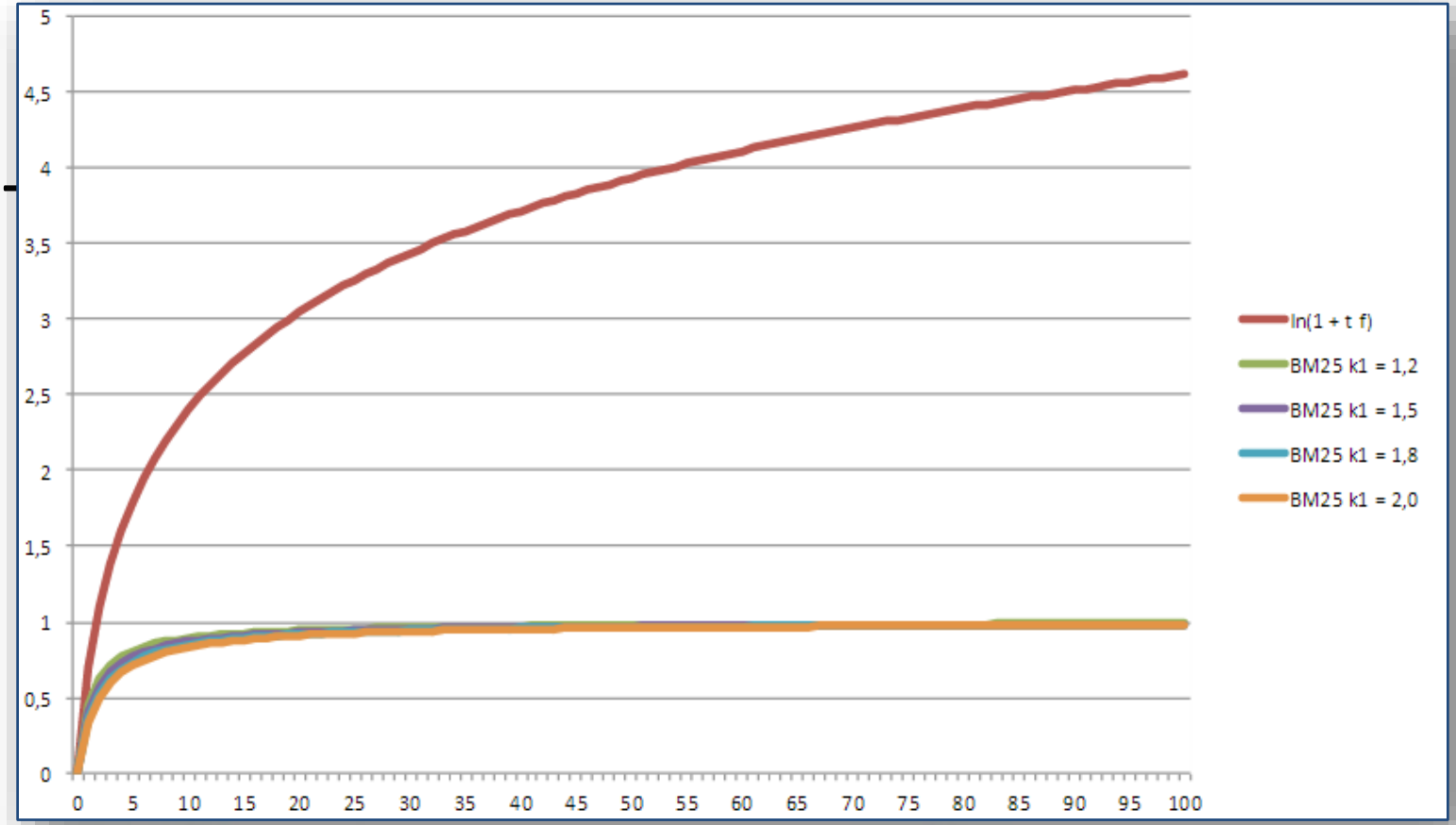
BTW: What‘s Okapi BM25 ???

- BM25 is a bag-of-words retrieval function
- **probabilistic model** instead vector space model
- function of **TF** and **IDF**

BTW: What‘s Okapi BM25 ???

BTW: What‘s Okapi BM25 ???

- TF is not unbounded: saturation!
  - Documents with high term frequency don‘t increase score too much


5
4,5
4
3,5
3
2,5
2
1,5
1
0,5
0
0 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100
ln(1 + t f)
BM25 k1 = 1,2
BM25 k1 = 1,5
BM25 k1 = 1,8
BM25 k1 = 2,0

BTW: What‘s Okapi BM25 ???

- TF is not unbounded: saturation!
  - Documents with high term frequency don‘t increase score too much
- Average document length
  - Used with tuning factor to change significance of repeated terms

BTW: What‘s Okapi BM25 ???

- TF is not unbounded: saturation!
  - Documents with high term frequency don‘t increase score too much
- Average document length
  - Used with tuning factor to change significance of repeated terms
- IDF similar to standard TF-IDF

Watch talk by Britta Weber

Tomorrow, 14:30 to 15:10, Frannz Club

Lucene 6: Query API

Lucene 6: Query API

- `Filter` completely gone!

Lucene 6: Query API

- `Filter` completely gone!

```
Query oldq = new FilteredQuery(
    new QueryBuilder(analyzer).createBooleanQuery("content", "some query"),
    new TermFilter(new Term(field, "Some Facet")));
```

Lucene 6: Query API

- `Filter` completely gone!

```
Query oldq = new FilteredQuery(
    new QueryBuilder(analyzer).createBooleanQuery("content", "some query"),
    new TermFilter(new Term(field, "Some Facet")));
```

```
Query newq = new BooleanQuery.Builder()
    .add(new QueryBuilder(analyzer).createBooleanQuery("content", "some query"), BooleanClause.Occur.MUST)
    .add(new TermQuery(new Term(field, "Some Facet")), BooleanClause.Occur.FILTER)
    .build();
```

Lucene 6: Query API

- `Filter` completely gone!
- `Query` *unmodifiable*

Lucene 6: Query API

- `Filter` completely gone!
- `Query` *unmodifiable*

```
Query newq = new BooleanQuery.Builder()
    .add(new QueryBuilder(analyzer).createBooleanQuery("content", "some query"), BooleanClause.Occur.MUST)
    .add(new TermQuery(new Term(field, "Some Facet")), BooleanClause.Occur.FILTER)
    .build();
```

Lucene 6: Query API

- `Filter` completely gone!
- `Query` *unmodifiable*
- `Query` boost removed

Lucene 6: Query API

- `Filter` completely gone!
- `Query` *unmodifiable*
- `Query` boost removed
  - new **`BoostQuery`**

Lucene 6: "Anti-Feature"

Removal of Lucene 4 index support!

Lucene 6: "Anti-Feature"

Removal of Lucene 4 index support!

- **Get rid of old index segments:** `IndexUpgrader` in latest Lucene 5 release helps!
- **Elasticsearch 5** has automatic index upgrader already implemented / **Solr** users have to manually do this

What's new

Apache Solr 6

New release bundled
with Lucene 6 release

SQL Parser and analytics framework
(streaming API)

SQL Parser

- Requires **Solr Cloud** setup
- New Streaming API behind the scenes
  - Introduced in Solr 5.1
- Presto Parser (Facebook)
- Parallelized across multiple Solr nodes

SQL Parser

Supported operators

- SELECT (fields, functions, aggregations)
- FROM (Solr core / index)
- WHERE (Solr query parser)
- GROUP BY (Solr facets/aggregations)
- ORDER BY (Solr sorting)

SQL Parser

Supported operators

- SELECT (fields, functions, aggregations)
- F
- V
- G
- C

```
SELECT fieldA, fieldB, count(*), sum(fieldC), avg(fieldY)
FROM tableA
WHERE fieldC = 'term1 term2'
GROUP BY fieldA, fieldB
HAVING ((sum(fieldC) > 1000) AND (avg(fieldY) <= 10))
ORDER BY sum(fieldC) asc
LIMIT 100
```

SQL Parser

```
Connection con = null;
try {
    con = DriverManager.getConnection("jdbc:solr://" + zkHost + "?collection=collection1&aggregationMode=map_reduce&numWorkers=2");
    stmt = con.createStatement();
    rs = stmt.executeQuery("SELECT a_s, sum(a_f) as sum FROM collection1 GROUP BY a_s ORDER BY sum desc");

    while(rs.next()) {
        String a_s = rs.getString("a_s");
        double s = rs.getDouble("sum");
    }
} finally {
    rs.close();
    stmt.close();
    con.close();
}
```

```
GROUP BY fieldA, fieldB
HAVING ((sum(fieldC) > 1000) AND (avg(fieldY) <= 10))
ORDER BY sum(fieldC) asc
LIMIT 100
```

Watch talk by Shalin Mangar!

next after, Maschinenhaus

Cross Data Center Replication (CDCR)

- Accommodate two or more data centers
- Accommodate limited band-with cross-datacenter connections
- Minimize coupling between peer clusters to increase reliability

Solr 6: More features

Solr 6: More features

- **GraphQuery** for graph traversal

Solr 6: More features

- **GraphQuery** for graph traversal
- Filters on realtime get

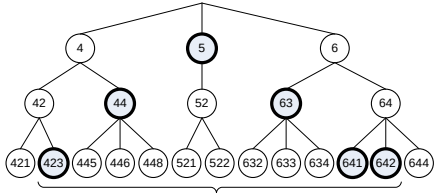
Solr 6: More features

- **GraphQuery** for graph traversal
- Filters on realtime get
- **DocValues** fields return as stored

Solr 6: More features

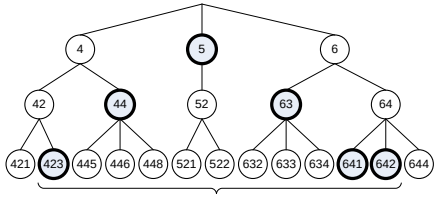
- **GraphQuery** for graph traversal
- Filters on realtime get
- **DocValues** fields return as stored
- Date support now fully **ISO-8601** conformant
  - Backed by Java 8's **java.time** API
  - *Warning:* reindex required for very early dates

Dimensional / Point Values



Delayed until version 6.2

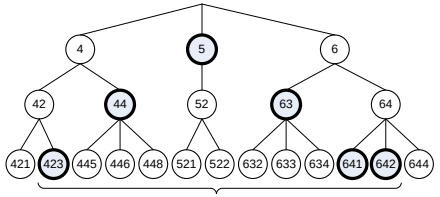
Dimensional / Point Values



Delayed until version 6.2

- *Community still discusses:*
  - Reindex requirement if old numeric fields need to be upgraded to point values
  - Requirement for DocValues *(no uninverting possible!)*

Dimensional / Point Values



Delayed until version 6.2

- *Community still discusses:*
  - Reindex requirement if old numeric fields need to be upgraded to point values
  - Requirement for DocValues *(no uninverting possible!)*
- For now it still uses deprecated `LegacyNumericField` as backend for Solr's `TrieField`

THANK YOU!

Questions?

SD DataSolutions GmbH
Wätjenstr. 49
28213 Bremen, Germany
+49 421 40889785-0
http://www.sd-datasolutions.de