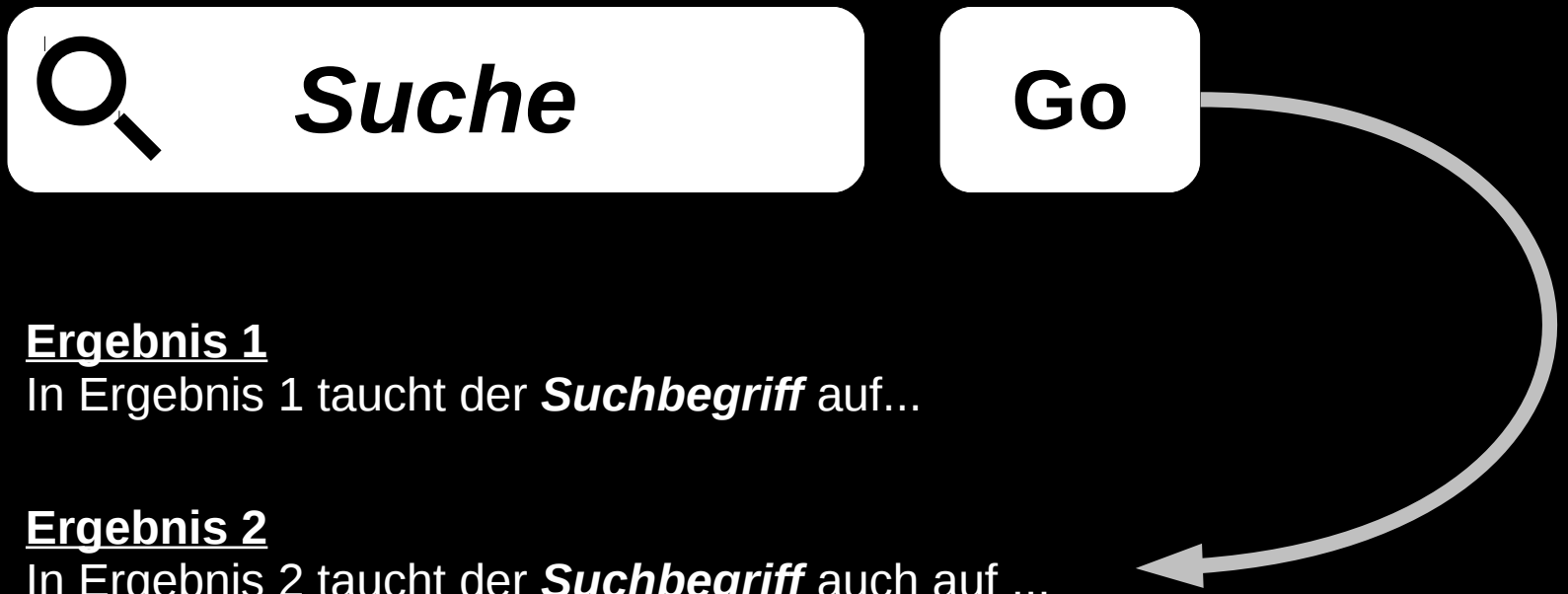
Suchen und Finden mit Lucene und Solr

Florian Hopf
04.07.2012

http://techcrunch.com/2010/08/04/schmidt-data/


Google
was kann Goole alles?
was kann google alles machen
was kann google alles
was kann man in google alles machen
was kann man mit google earth alles machen
Weitere Informationen
Suche
Alles
Bilder
Maps
Videos
News
Shopping
Diskussionen
Mehr
Karlsruhe
Standort ändern
Web
Seiten auf Deutsch
Seiten aus Deutschland
Übersetzte Seiten
Mehr Optionen
Ergebnisse für was kann google alles machen
Stattdessen suchen nach: was kann Goole alles?
Was man mit Google alles machen kann ;) - ComputerBase Forum
www.computerbase.de › ... › Webseiten und soziale Netzwerke
10 Beiträge - 9 Autoren - 22. Jan. 2011
Was man mit Google alles machen kann ;) Webseiten und soziale Netzwerke.
Was man mit google alles machen kann » ToolBlog
www.toolblog.de/2006/03/was-man-mit-google-alles-machen-kann/
31. März 2006 – "Hat noch jemand den Überblick über die Produktpalette von Google? Ich nicht", schreibt Perun in seinem Weblog. Da geht es ihm wie mir.
Was kann man bei Google Earth alles machen und wie funktioniert ...
www.gutefrage.net/.../was-kann-man-bei-google-earth-alles-machen-...
4 Antworten - 5. Mai 2009
Ich habe mir jetzt Google Earth heruntergeladen. Was kann man alles ... ach mensch mach es dir doch nicht so schwer ... du kannst mit googe ...
was kann man im internet alles machen, ohne ... - 13 Antworten - 14. Jan. 2012
was kann man alles bei googlemail oder so ... - 2 Antworten - 14. Dez. 2011
Was kann man alles auf Google Earth machen ... - 8 Antworten - 5. Sept. 2011
Was kann man mit einem Google Konto alles ... - 2 Antworten - 21. Febr. 2011
Weitere Ergebnisse von gutefrage.net »

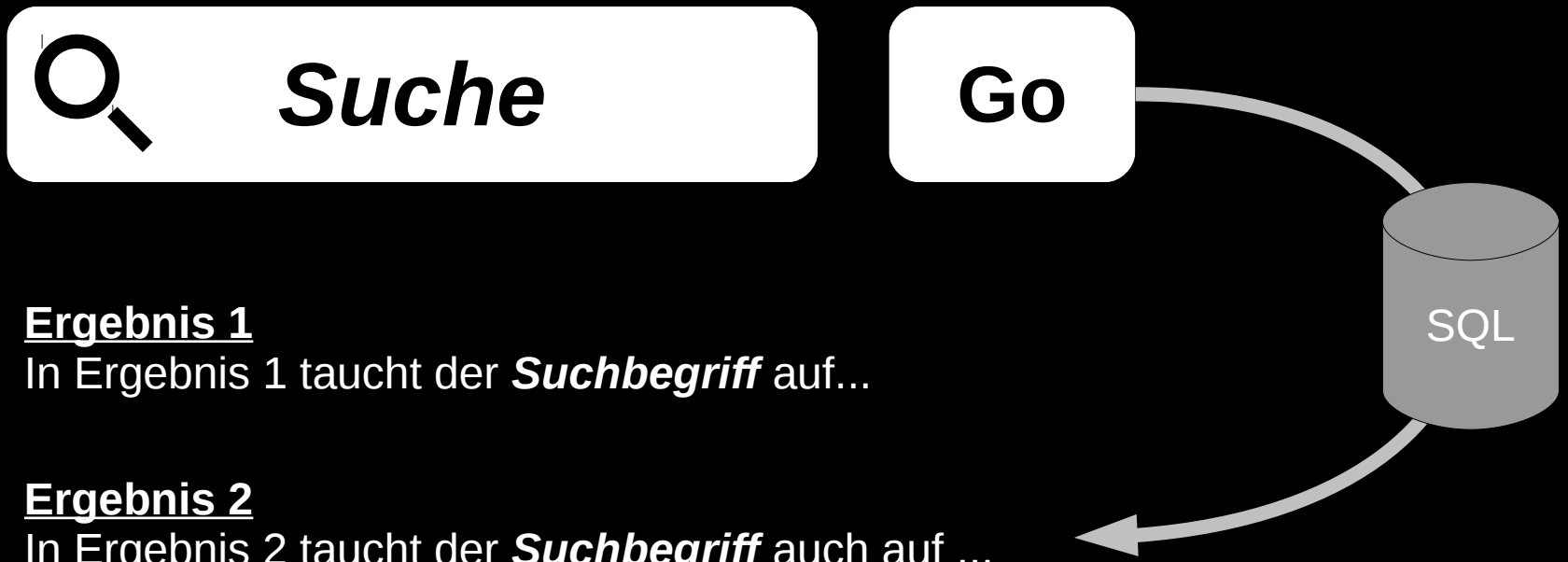

Suche
Go



Ergebnis 1
In Ergebnis 1 taucht der ***Suchbegriff*** auf...

Ergebnis 2
In Ergebnis 2 taucht der ***Suchbegriff*** auch auf ...

Ergebnis 3
… hier steht der ***Suchbegriff*** aus Ergebnis 3...



Ergebnis 1
In Ergebnis 1 taucht der ***Suchbegriff*** auf...

Ergebnis 2
In Ergebnis 2 taucht der ***Suchbegriff*** auch auf ...

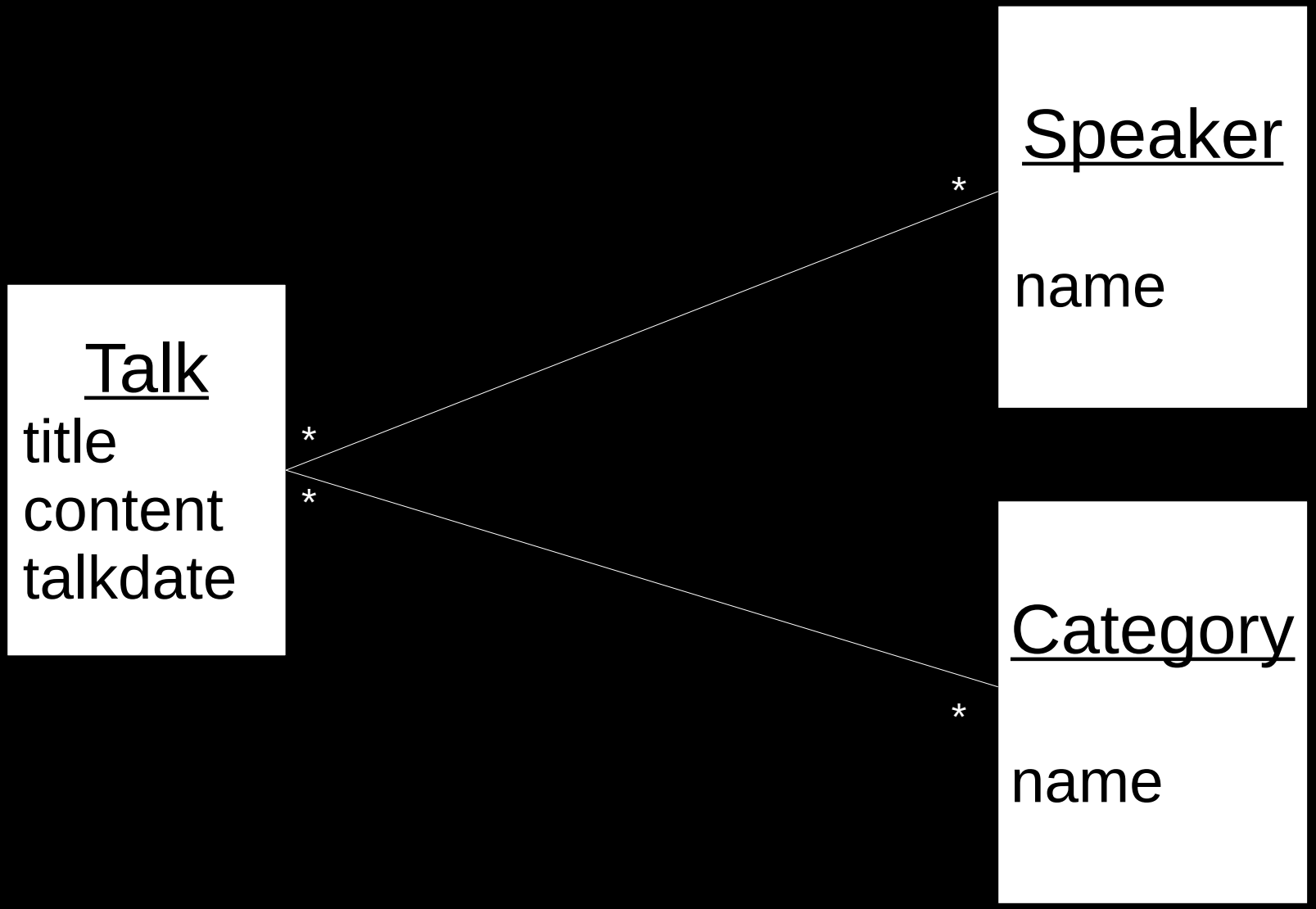
Ergebnis 3
… hier steht der ***Suchbegriff*** aus Ergebnis 3...

Apache Karaf

Datum	: 25.04.12
Ort	: Uni KA, HS101
Speaker	: Christian Schneider, Open Source Softwarearchitekt bei Talend
	: Achim Nierbeck, Solution Architect bei der ISB AG
Homepage	: http://www.liquid-reality.de/
	: http://notizblog.nierbeck.de/
Promo-Flyer	: -
Slides	: Google Docs
Bilder	: -
Aufzeichnung	: -
Zus. Links	: http://karaf.apache.org/, http://www.eclipse.org/equinox/, http://felix.apache.org/

Apache Karaf ist ein OSGi Container, der auf Eclipse Equinox und Apache Felix aufsetzt und diese in Hinsicht Deployment, Management und Usability erweitert. Karaf lässt sich über eine Shell Konsole, JMX und eine Webkonsole verwalten. Die Karaf Shell ist von der Bedienung an die Unix Bash Shell angelehnt und unterstützt Historie, Tab Completion und eine eingebaute Hilfefunktion. Das Deployment von Bundles wird durch Karaf stark vereinfacht, da diese in Features gruppiert und direkt aus Maven Repositories deployed werden können.

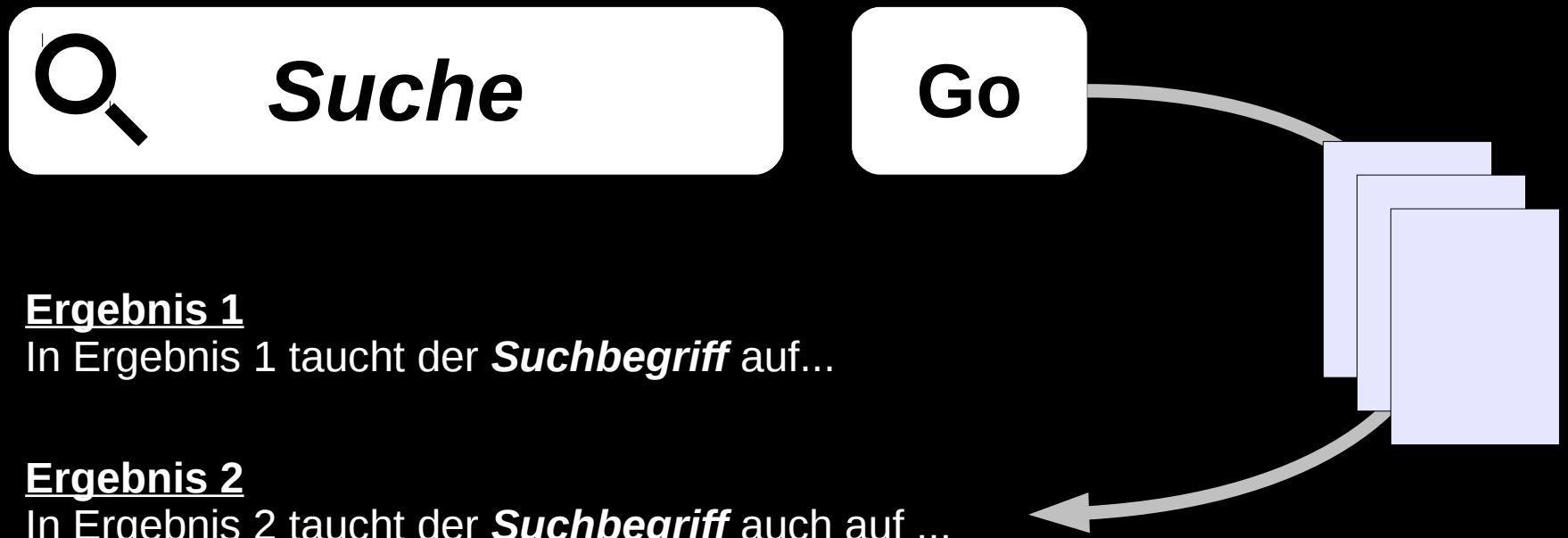
Der Vortrag gibt zunächst einen Überblick über Karaf. Anhand eines Beispielprojektes wird live gezeigt, wie mit Hilfe von Karaf eine Anwendung entwickelt, deployed und debugged werden kann. Es wird auch gezeigt, wie Karaf hilft, typische Probleme beim Deployment in OSGi aufzuspüren und zu lösen.


Talk
title
content
talkdate
Speaker
name
Category
name
*
*
*
*

```
mysql> select * from talk where title like "%apache%";
+----+-----------------------------------------+---------+------------+
| id | title                                   | content | talkdate   |
+----+-----------------------------------------+---------+------------+
|  1 | Apache Karaf                            | ...     | 2012-04-25 |
|  2 | Integration ganz einfach mit Apache Camel | ...     | 2012-04-04 |
+----+-----------------------------------------+---------+------------+
2 rows in set (0.00 sec)
```

```
mysql> select * from talk t join talk_category tc join category c where t.id =
tc.talk and tc.category = c.id and (c.name like '%OSGi%' or t.title like
'%OSGi%' or t.content like '%OSGi%');
+----+-----------------------------------------+---------+------------
+------+----------+----+------+
| id | title           | content | talkdate   | talk | category | id | name |
+----+-----------------------------------------+---------+------------
+------+----------+----+------+
|  1 | Apache Karaf    | ...     | 2012-04-25 |    1 |        1 |  1 | OSGi |
|  2 | Integration ... | ...     | 2012-04-04 |    2 |        1 |  1 | OSGi |
+----+-----------------------------------------+---------+------------
+------+----------+----+------+
2 rows in set (0.00 sec)
```

- Performance?
- Ranking?
- False Positives
- Ähnlichkeitssuche (Meyer <=> Meier)
- Flexibilität?
- Wartbarkeit?



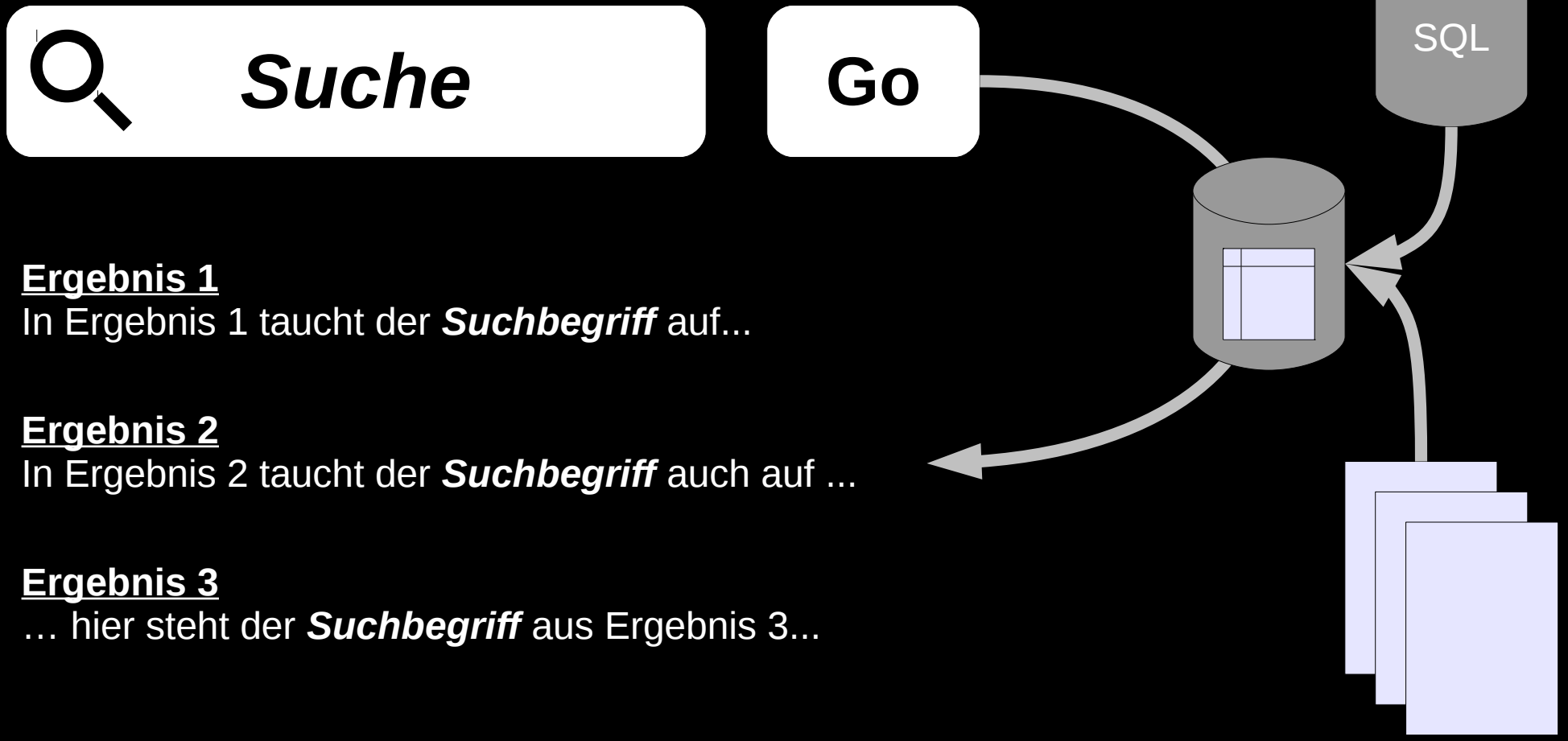
Ergebnis 1
In Ergebnis 1 taucht der ***Suchbegriff*** auf...

Ergebnis 2
In Ergebnis 2 taucht der ***Suchbegriff*** auch auf ...

Ergebnis 3
… hier steht der ***Suchbegriff*** aus Ergebnis 3...

```
File directory = new File(dir);
File[] textFiles = directory.listFiles(new TextFiles());

for (File probableMatch : textFiles) {
    String text = readText(probableMatch);
    if (text.matches(".*" + Pattern.quote(term) + ".*")) {
        filesWithMatches.add(probableMatch.getAbsolutePath());
    }
}
```

- Skalierbarkeit?
- Unterschiedliche Formate?
- Ranking?
- Kombination mit Datenbank?



Ergebnis 1
In Ergebnis 1 taucht der ***Suchbegriff*** auf...

Ergebnis 2
In Ergebnis 2 taucht der ***Suchbegriff*** auch auf ...

Ergebnis 3
… hier steht der ***Suchbegriff*** aus Ergebnis 3...

Dokument 1

Die Stadt liegt in den Bergen.

Dokument 2

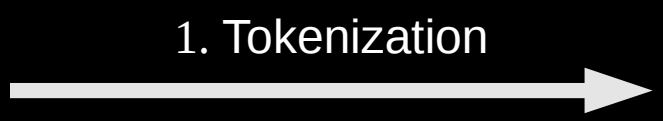
Vom Berg kann man die Stadt sehen.

Dokument 1

Die Stadt liegt in den Bergen.

Dokument 2

Vom Berg kann man die Stadt sehen.



Die	1
Stadt	1,2
liegt	1
in	1
den	1
Bergen	1
Vom	2
Berg	2
kann	2
man	2
die	2
sehen	2

Dokument 1

Die Stadt liegt in den Bergen.

Dokument 2

Vom Berg kann man die Stadt sehen.

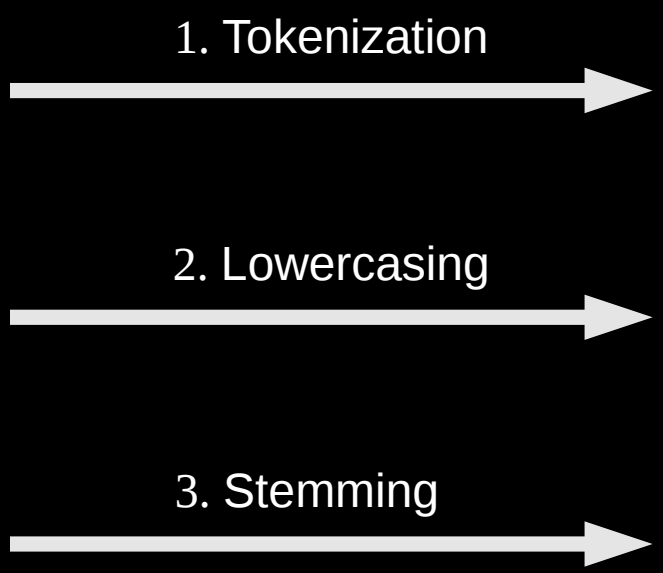
die	1,2
stadt	1,2
liegt	1
in	1
den	1
bergen	1
vom	2
berg	2
kann	2
man	2
sehen	2

Dokument 1

Die Stadt liegt in den Bergen.

Dokument 2

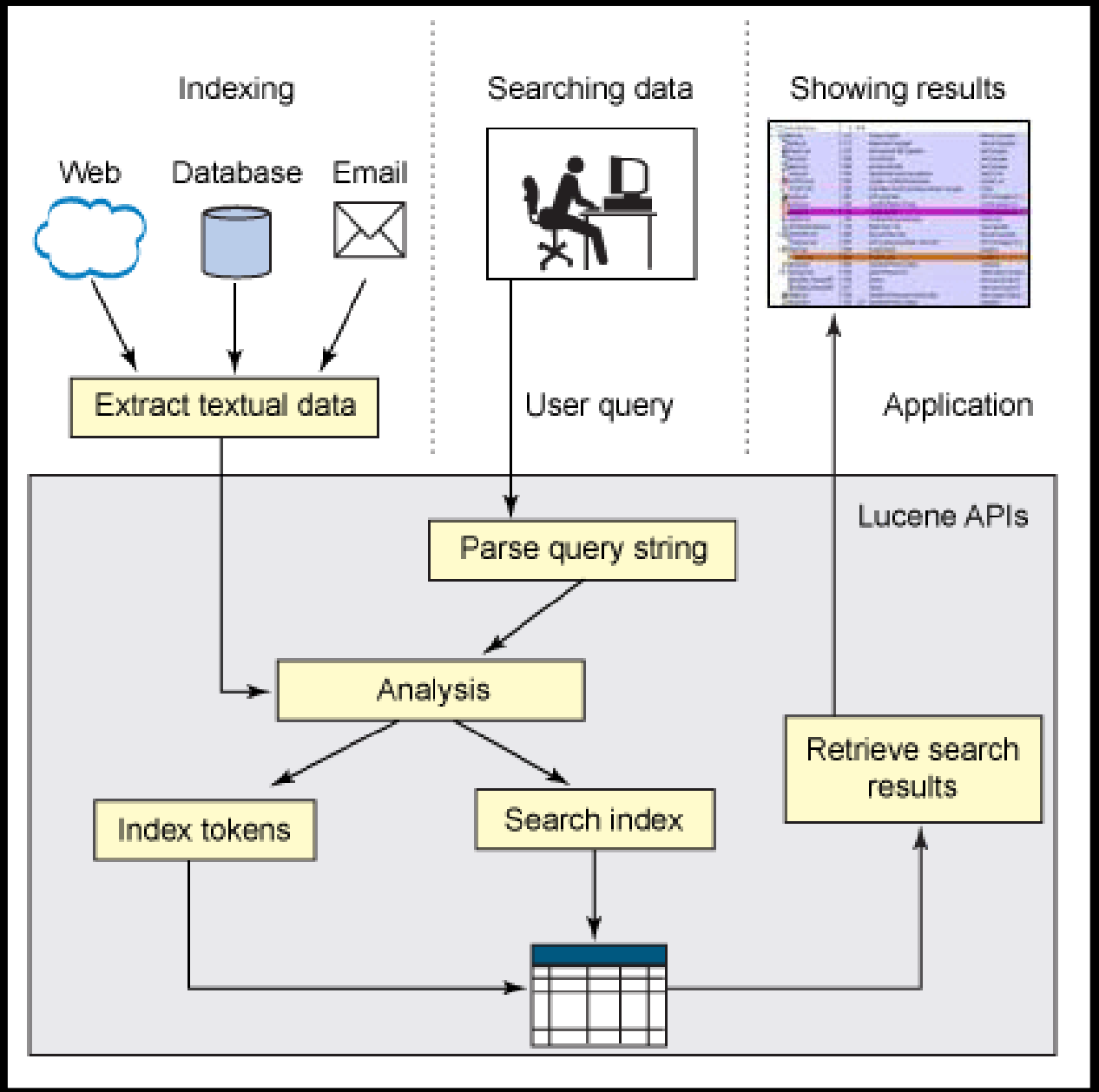
Vom Berg kann man die Stadt sehen.



die	1,2
stadt	1,2
liegt	1
in	1
den	1
berg	1,2
vom	2
kann	2
man	2
seh	2


Lucene

- Java-Bibliothek
- Invertierter Index
- Analyzer
- Query-Syntax
- Relevanz-Algorithmus
- KEIN Crawler
- KEIN Document-Extractor



Quelle: http://www.ibm.com/developerworks/java/library/os-apache-lucenesearch/

- Indexieren:
  - Erstellen eines Documents
  - Festlegen des Analyzers
  - Indexieren über IndexWriter
- Suchen:
  - Verwendung des selben Analyzers
  - Parsen der Query mit QueryParser
  - Auslesen über IndexSearcher/IndexReader
  - Ausgabe über Document

Document

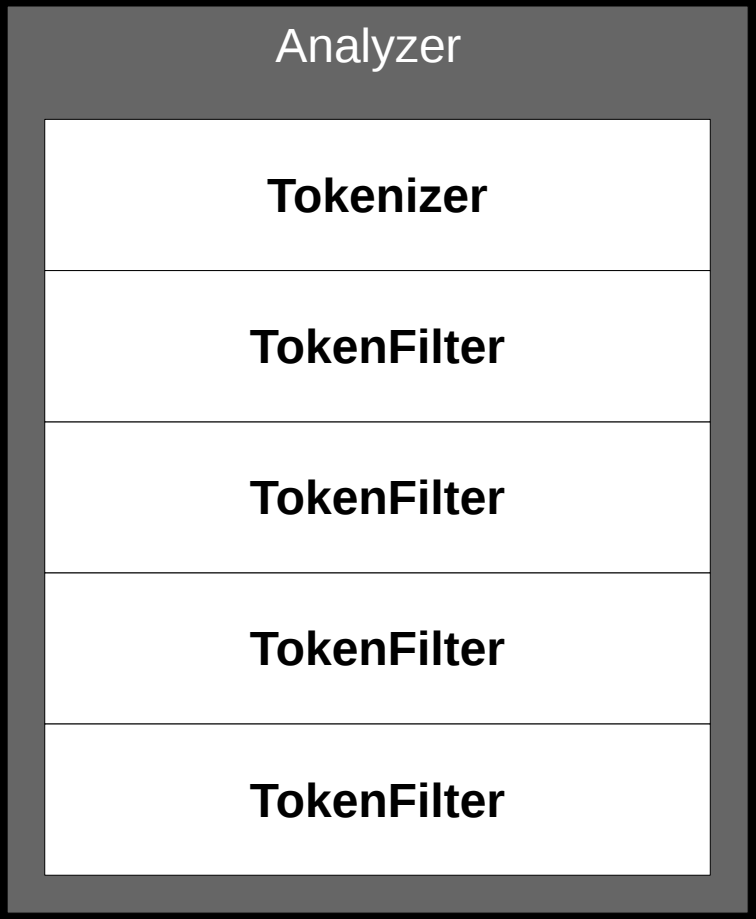
title	Integration ganz einfach mit Apache Camel
date	20120404
speaker	Christian Schneider

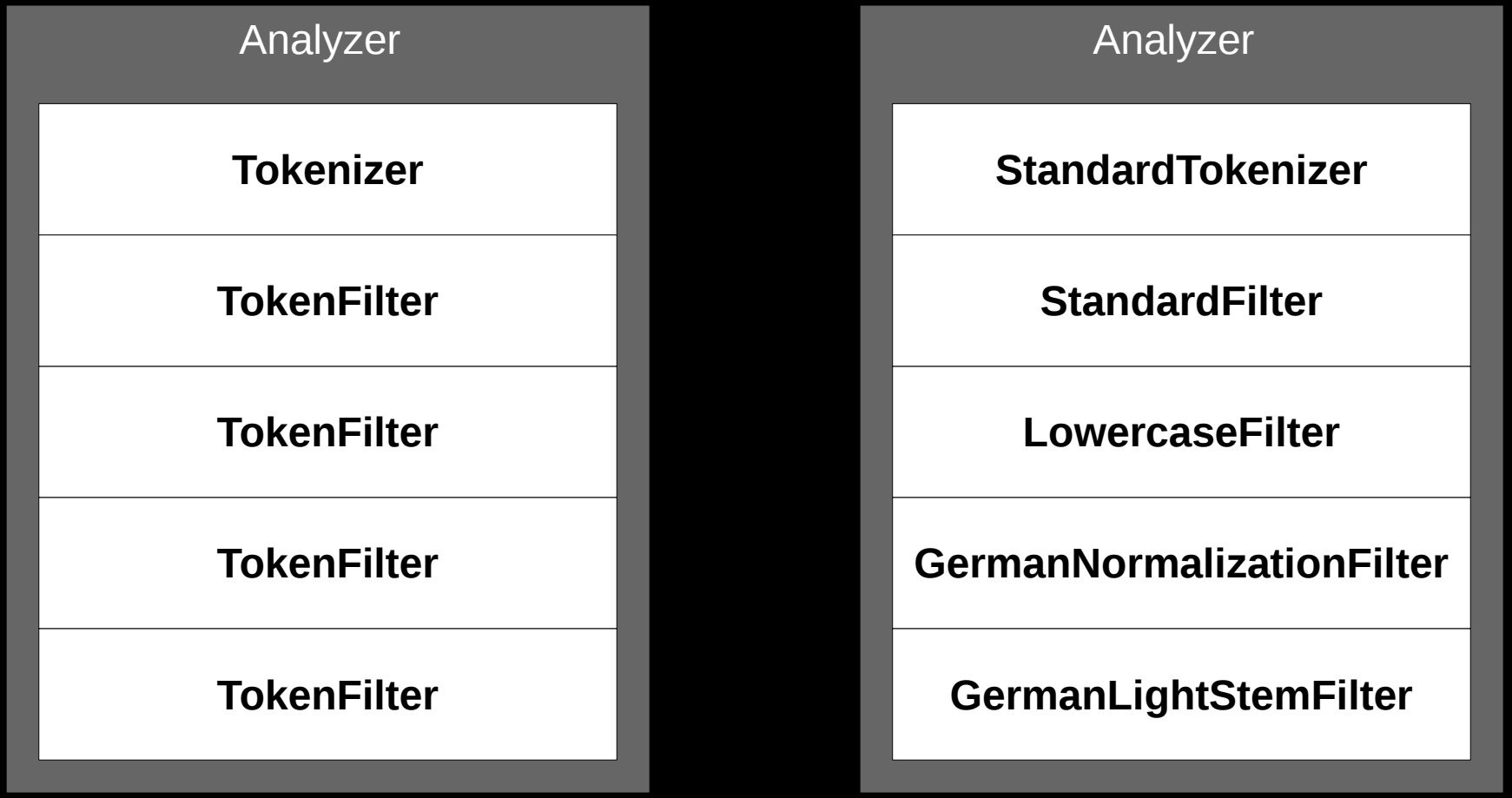
Document

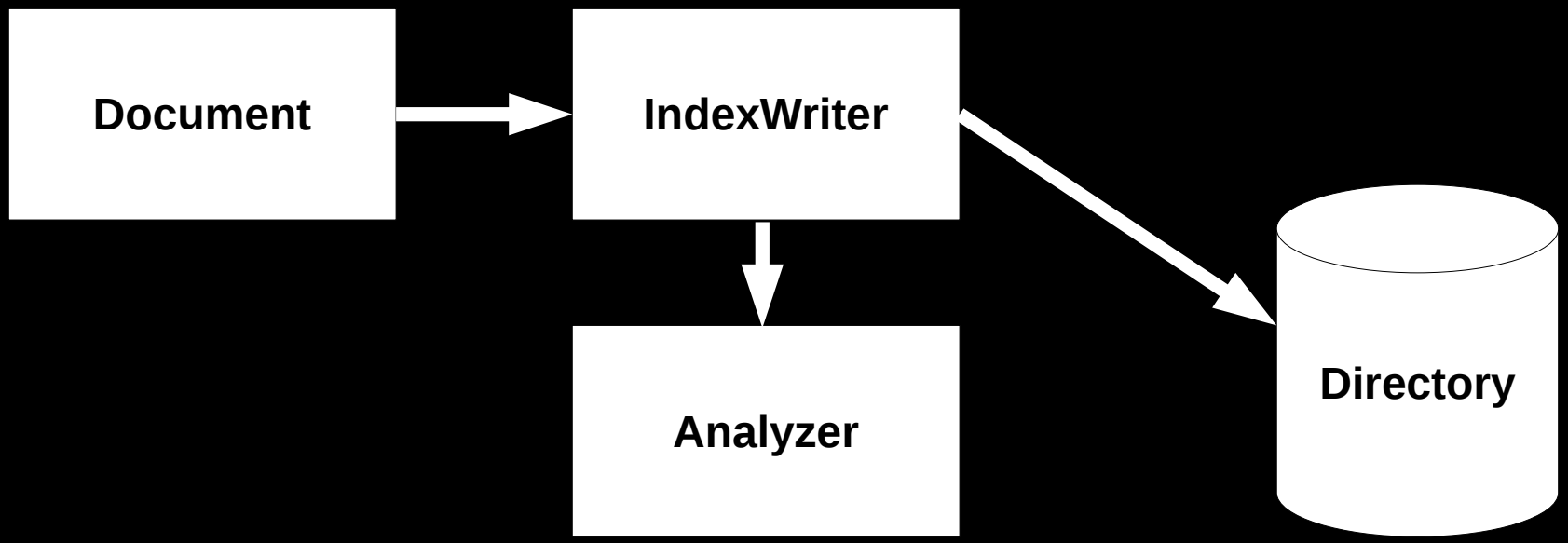
title	Apache Karaf
date	20120425
speaker	Christian Schneider
speaker	Achim Nierbeck

- Index
  - ANALYZED
  - NOT_ANALYZED
  - NO
- Store
  - YES/NO
- Feldtyp
  - String, Numeric, Boolean

```
Document camel = new Document();
camel.add(new Field("title", "Integration ganz einfach mit Apache Camel",
                        Field.Store.YES, Field.Index.ANALYZED));
camel.add(new Field("date", "20120404", Field.Store.NO,
                        Field.Index.ANALYZED));
camel.add(new Field("speaker", "Christian Schneider", Field.Store.YES,
                        Field.Index.ANALYZED));

Document karaf = new Document();
karaf.add(new Field("title", "Apache Karaf", Field.Store.YES,
                        Field.Index.ANALYZED));
karaf.add(new Field("date", "20120424", Field.Store.NO,
                        Field.Index.ANALYZED));
karaf.add(new Field("speaker", "Christian Schneider", Field.Store.YES,
                        Field.Index.ANALYZED));
karaf.add(new Field("speaker", "Achim Nierbeck", Field.Store.YES,
                        Field.Index.ANALYZED));
```

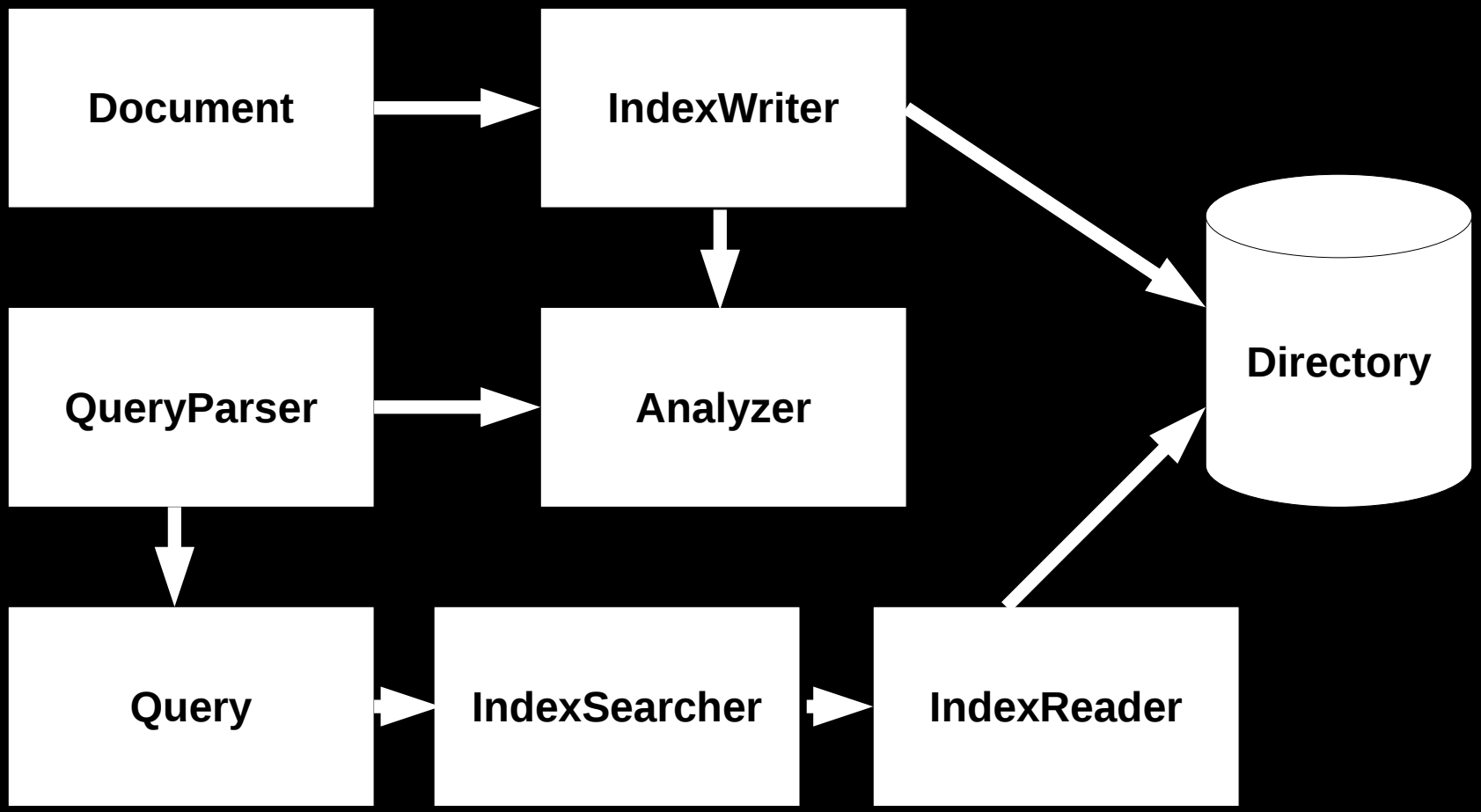

Analyzer
Tokenizer
TokenFilter
TokenFilter
TokenFilter
TokenFilter


Analyzer
Tokenizer
TokenFilter
TokenFilter
TokenFilter
TokenFilter
Analyzer
StandardTokenizer
StandardFilter
LowercaseFilter
GermanNormalizationFilter
GermanLightStemFilter


Document
IndexWriter
Analyzer
Directory

```
Directory dir = FSDirectory.open(new File("/tmp/testindex"));
IndexWriterConfig config = new IndexWriterConfig(Version.LUCENE_36,
                                    new GermanAnalyzer(Version.LUCENE_36));
config.setOpenMode(IndexWriterConfig.OpenMode.CREATE);
IndexWriter writer = new IndexWriter(dir, config);

writer.addDocument(camel);
writer.addDocument(karaf);

writer.commit();
writer.close();
```

DEMO


Document
IndexWriter
Directory
QueryParser
Analyzer
Query
IndexSearcher
IndexReader

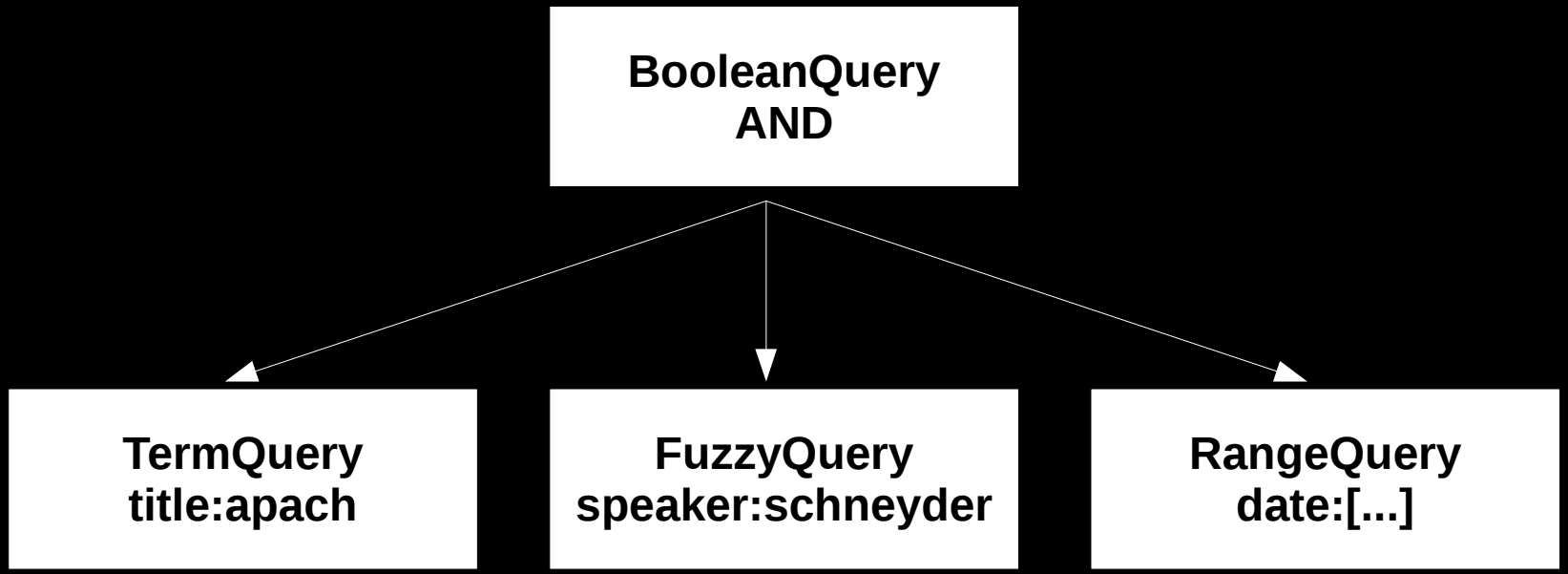
```
IndexReader reader = IndexReader.open(dir);
IndexSearcher searcher = new IndexSearcher(reader);
QueryParser parser = new QueryParser(Version.LUCENE_36, "title",
                                     new GermanAnalyzer(Version.LUCENE_36));
Query query = parser.parse("apache");

TopDocs result = searcher.search(query, 10);
assertEquals(2, result.totalHits);

for(ScoreDoc scoreDoc: result.scoreDocs) {
    Document doc = searcher.doc(scoreDoc.doc);
    String title = doc.get("title");
    assertTrue(title.equals("Apache Karaf")
               || title.equals("Integration ganz einfach mit Apache Camel"));
}
```

- TermQuery
  - *Apache*
  - *title:Apache*
- Boolean Query
  - *Apache AND Karaf*
- PhraseQuery
  - *"Apache Karaf"*

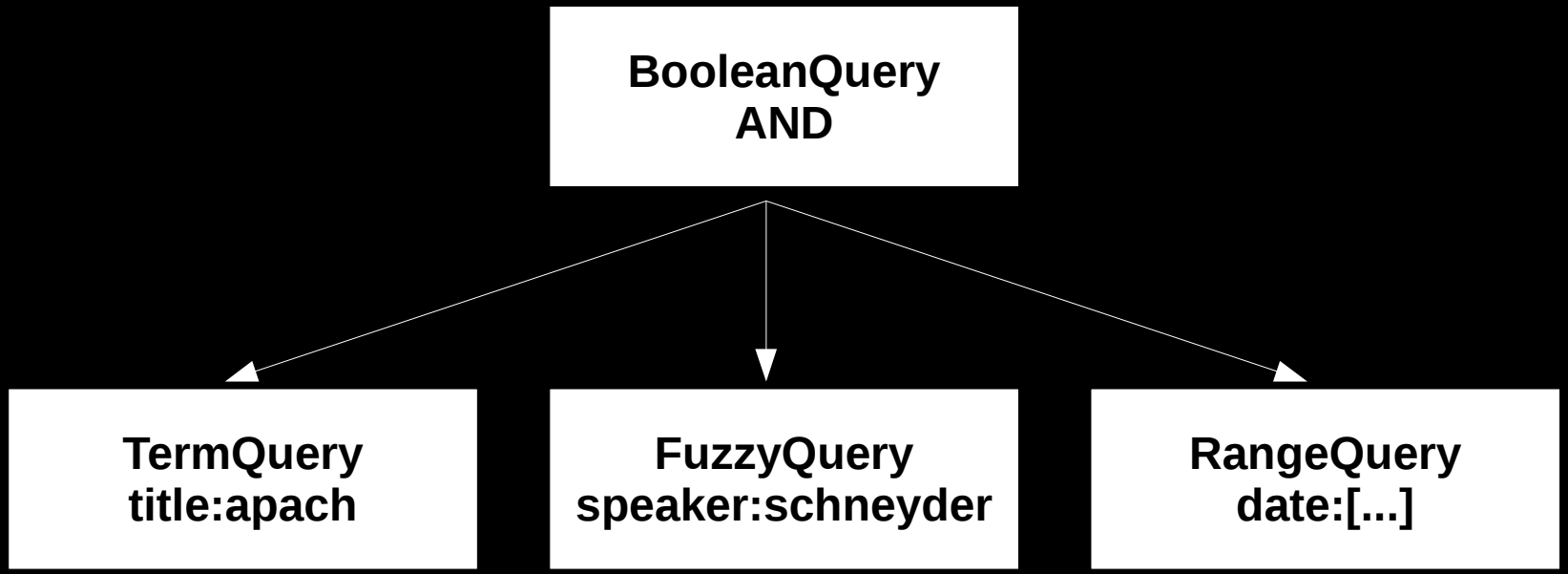
- WildcardQuery
  - *Integ**
  - *Te?t*
- RangeQuery
  - *date:[20120705 TO 20121231]*
- FuzzyQuery
  - *Schneyder~*

title:Apache AND speaker:schneyder~ AND date:[20120401 TO 20120430]

title:Apache AND speaker:schneyder~ AND date:[20120401 TO 20120430]

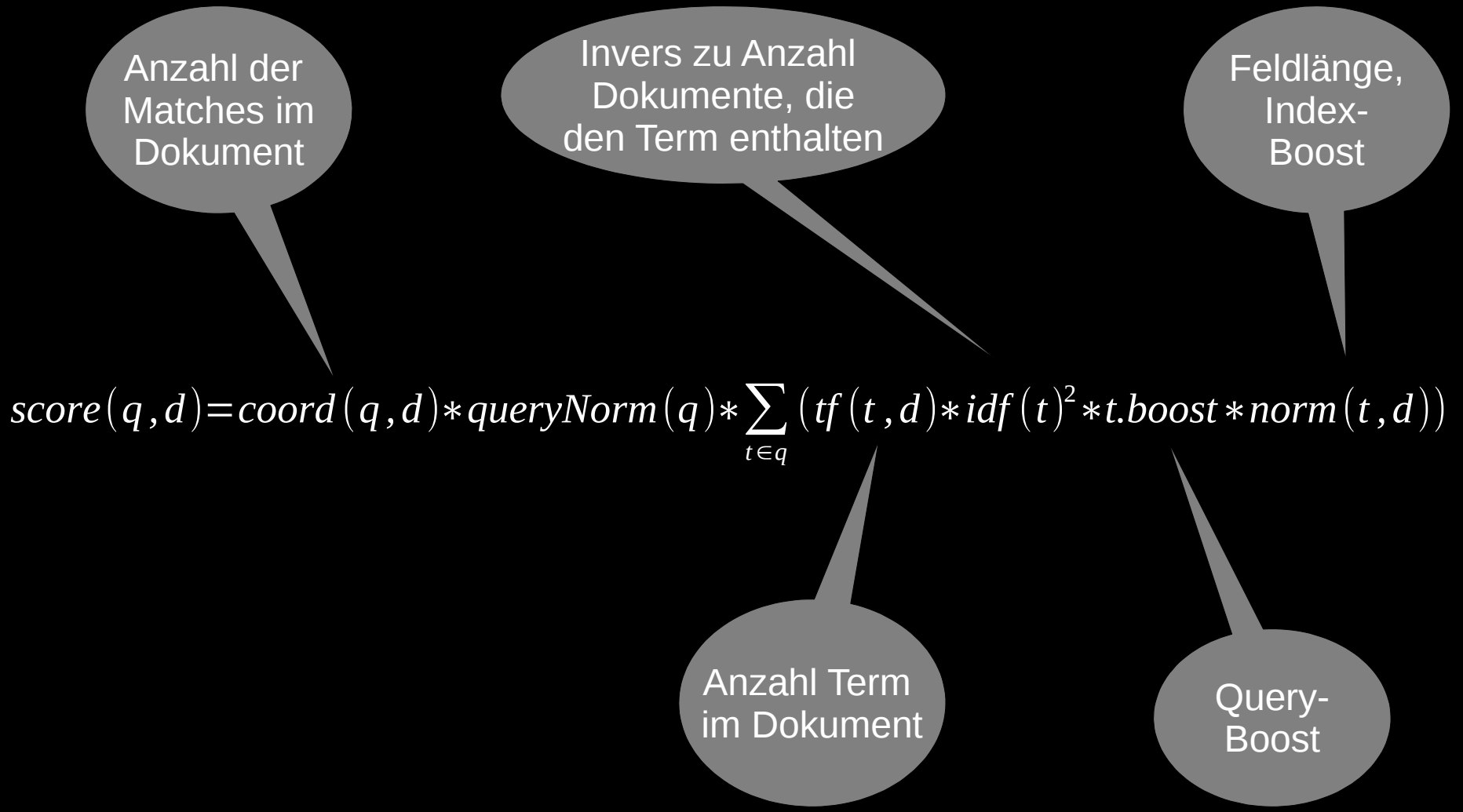


title:Apache AND speaker:schneyder~ AND date:[20120401 TO 20120430]

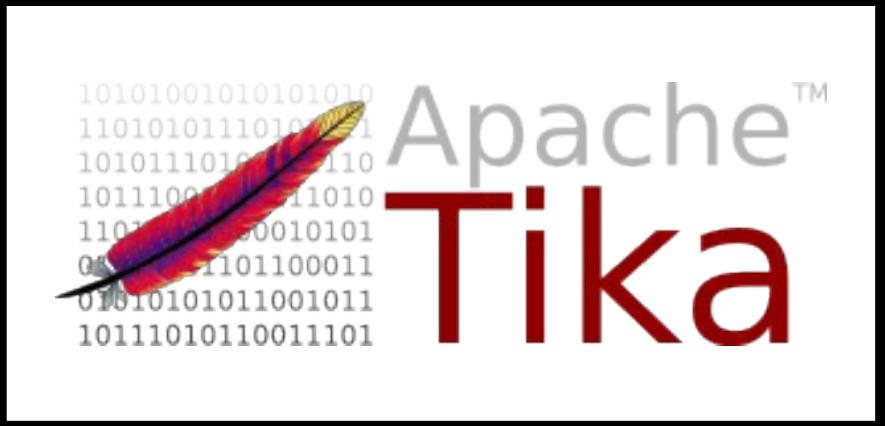


- FilterQueries
  - Ausschlusskriterium, kann gecacht werden
- Sortierung
- Boosting
  - Indexing-Time
  - Query-Time

$$score(q,d) = coord(q,d) * queryNorm(q) * \sum_{t \in q} \left(tf(t,d) * idf(t)^2 * t.boost * norm(t,d) \right)$$


Anzahl der Matches im Dokument
Invers zu Anzahl Dokumente, die den Term enthalten
Feldlänge, Index-Boost
$$score(q,d) = coord(q,d) * queryNorm(q) * \sum_{t \in q} (tf(t,d) * idf(t)^2 * t.boost * norm(t,d))$$
Anzahl Term im Dokument
Query-Boost

DEMO


Apache™
Tika

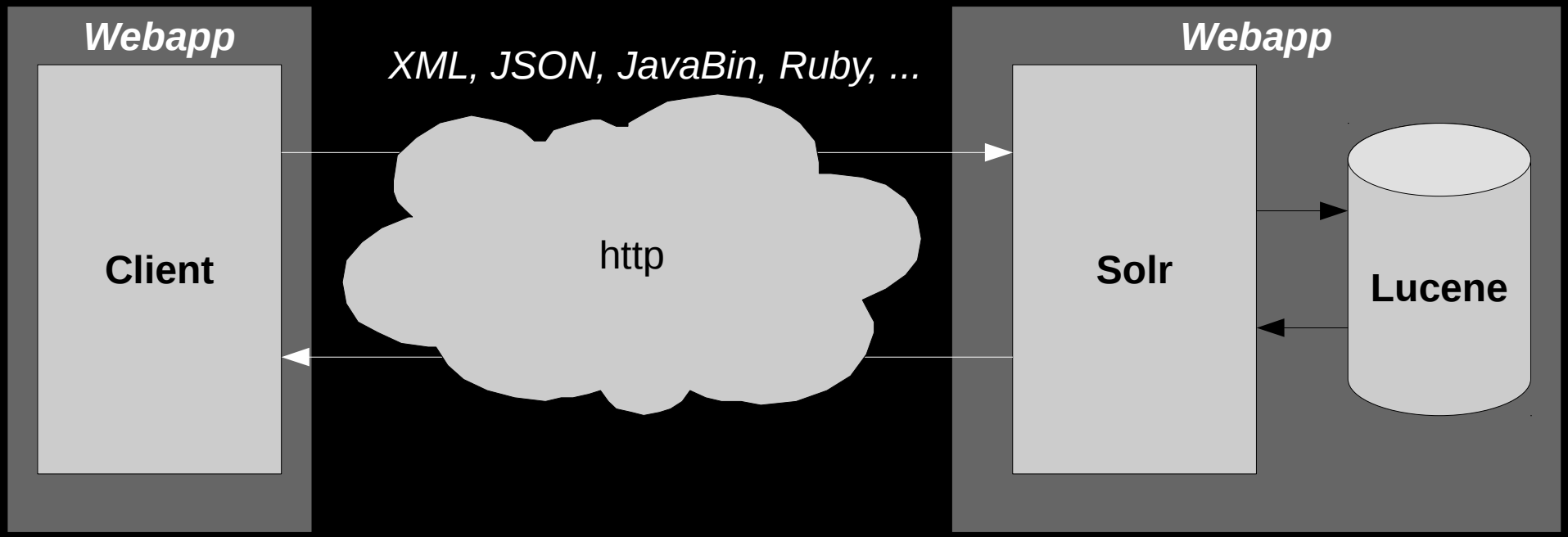
- Parser API
- Zahlreiche Formate
- Integriert OpenSource-Libs
- Betrieb embedded oder über Server

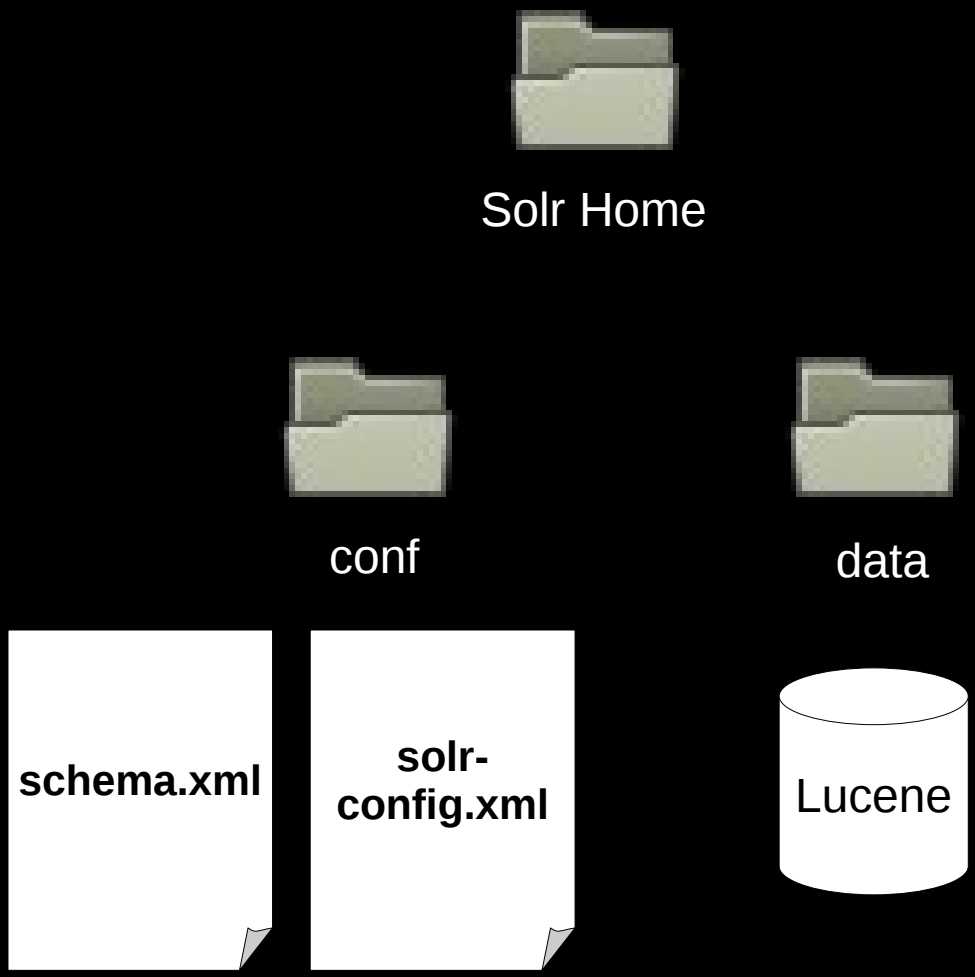
```
FileInputStream in = new FileInputStream(file);
AutoDetectParser parser = new AutoDetectParser();
Metadata metadata = new Metadata();
metadata.add(Metadata.RESOURCE_NAME_KEY, file.getName());
BodyContentHandler contentHandler = new BodyContentHandler();

parser.parse(in, contentHandler, metadata);

String title = metadata.get(Metadata.TITLE);
String author = metadata.get(Metadata.AUTHOR);
String content = contentHandler.toString();
```

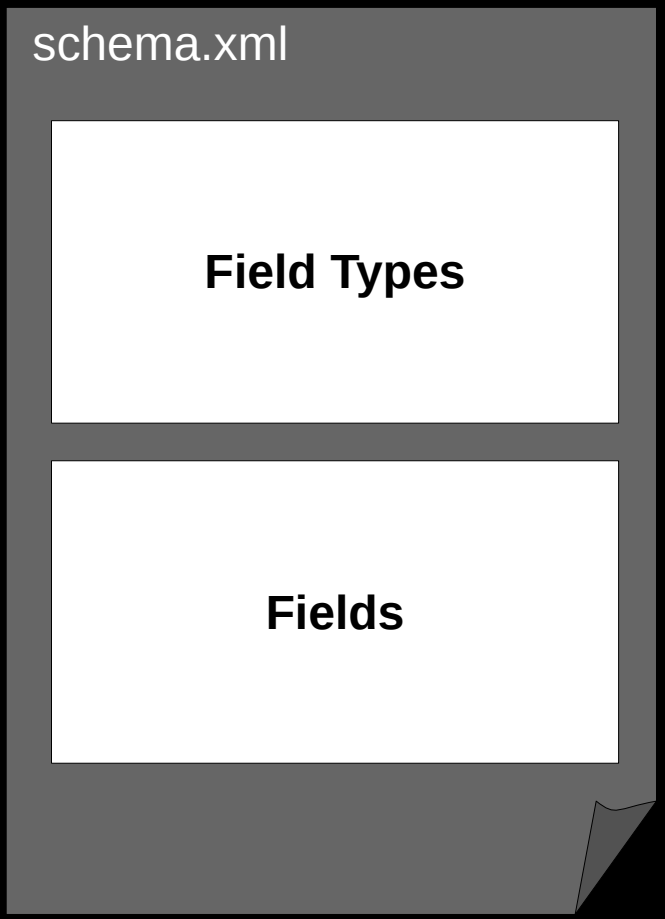
DEMO


Apache
Solr

- Enterprise Search Server
- Basiert auf Lucene
- HTTP API
- Index-Schema
- Integriert häufig verwendete Lucene-Module
- Facettierung
- Dismax Query Parser
- Admin-Interface

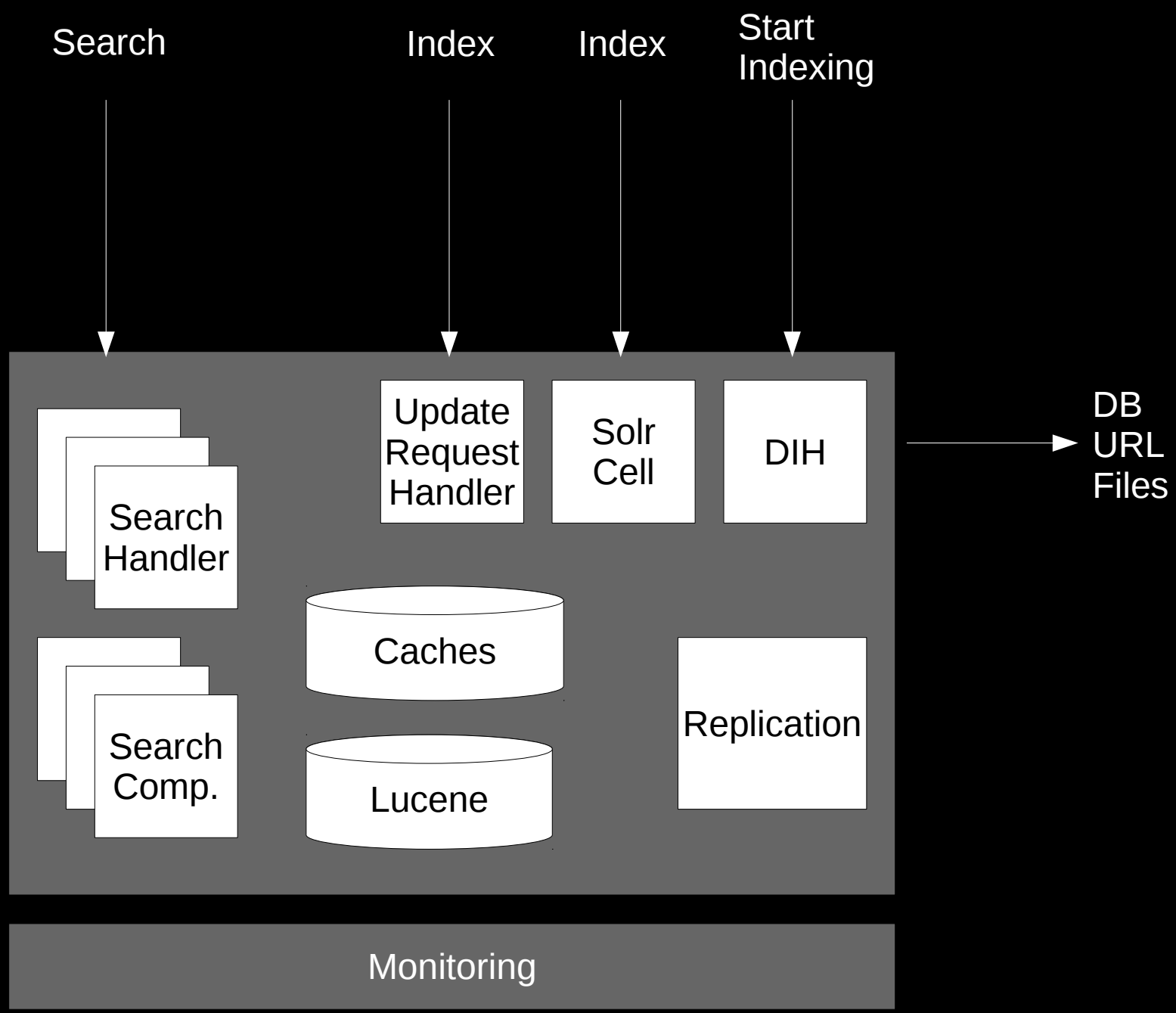

Webapp
Client
XML, JSON, JavaBin, Ruby, ...
http
Webapp
Solr
Lucene

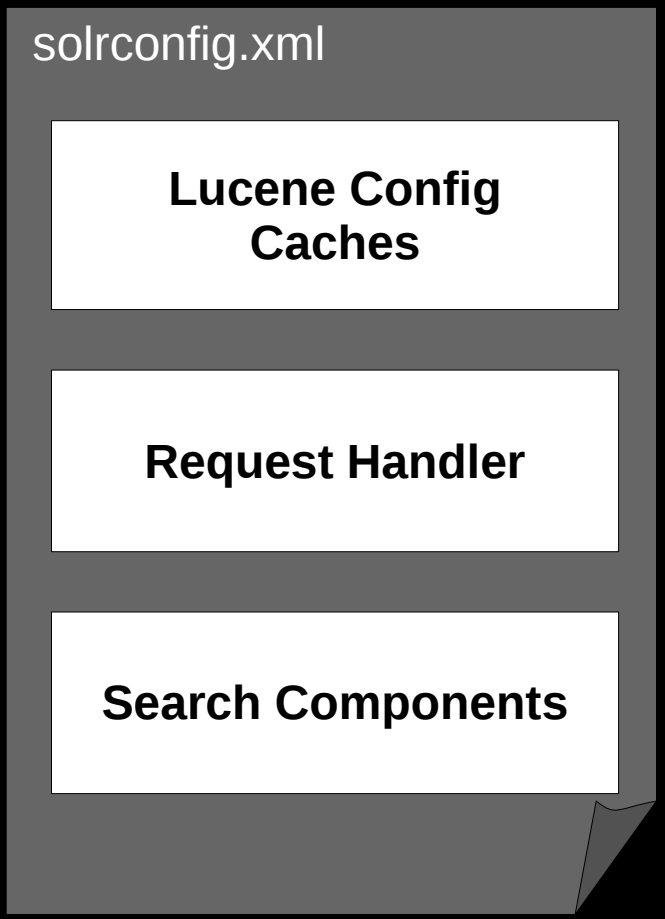

Solr Home
conf
data
schema.xml
solr-
config.xml
Lucene


schema.xml
Field Types
Fields

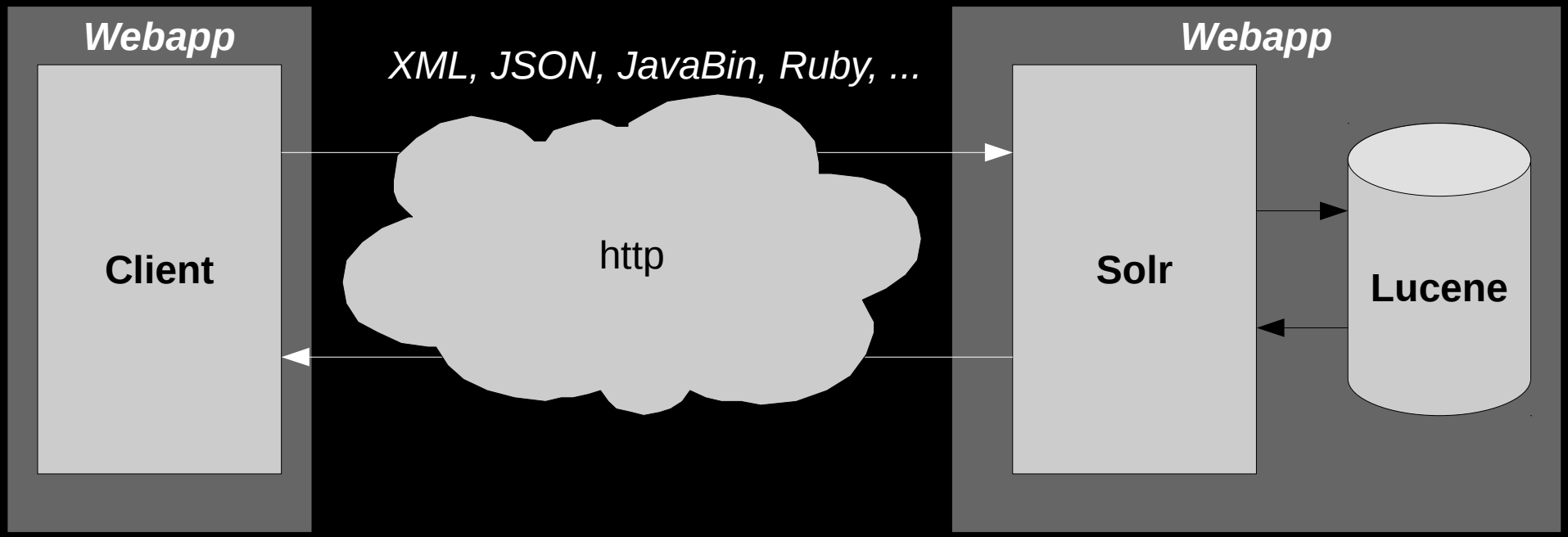
```
<!-- The StrField type is not analyzed, but indexed/stored verbatim. -->
<fieldType name="string" class="solr.StrField" sortMissingLast="true" />

<!-- German -->
<fieldType name="text_de" class="solr.TextField" positionIncrementGap="100">
  <analyzer type="index">
    <tokenizer class="solr.StandardTokenizerFactory"/>
    <filter class="solr.LowerCaseFilterFactory"/>
    <filter class="solr.GermanNormalizationFilterFactory"/>
    <filter class="solr.GermanLightStemFilterFactory"/>
  </analyzer>
  <analyzer type="query">
    <tokenizer class="solr.StandardTokenizerFactory"/>
    <filter class="solr.SynonymFilterFactory" synonyms="synonyms.txt"
            ignoreCase="true" expand="true"/>
    <filter class="solr.LowerCaseFilterFactory"/>
    <filter class="solr.GermanNormalizationFilterFactory"/>
    <filter class="solr.GermanLightStemFilterFactory"/>
   </analyzer>
 </fieldType>
```

```
<fields>
  <field name="path" type="string" indexed="true" stored="true" required="true" />
  <field name="title" type="text_de" indexed="true" stored="true"/>
  <field name="category" type="string" indexed="true" stored="true"
       multiValued="true" omitNorms="true"/>
  <field name="date" type="date" indexed="true" stored="true"/>
  <field name="speaker" type="string" indexed="true" stored="true"
       multiValued="true"/>
  <field name="speaker_search" type="text_ws" indexed="true" stored="false"
       multiValued="true"/>
  <field name="content" type="text_de" indexed="true" stored="true"/>
</fields>

<uniqueKey>path</uniqueKey>
<copyField source="speaker" dest="speaker_search"/>
```


Search
Index
Index
Start Indexing
Update Request Handler
Solr Cell
DIH
DB URL Files
Search Handler
Caches
Search Comp.
Lucene
Replication
Monitoring


solrconfig.xml
Lucene Config
Caches
Request Handler
Search Components

```
<requestHandler name="/jugka" class="solr.SearchHandler">
    <lst name="defaults">
        <str name="echoParams">explicit</str>
        <int name="rows">10</int>
        <str name="q.op">AND</str>
        <str name="q.alt">*:*</str>
        <str name="defType">edismax</str>
        <str name="qf">content title category^0.8 speaker_search^0.8</str>
    </lst>
</requestHandler>
```

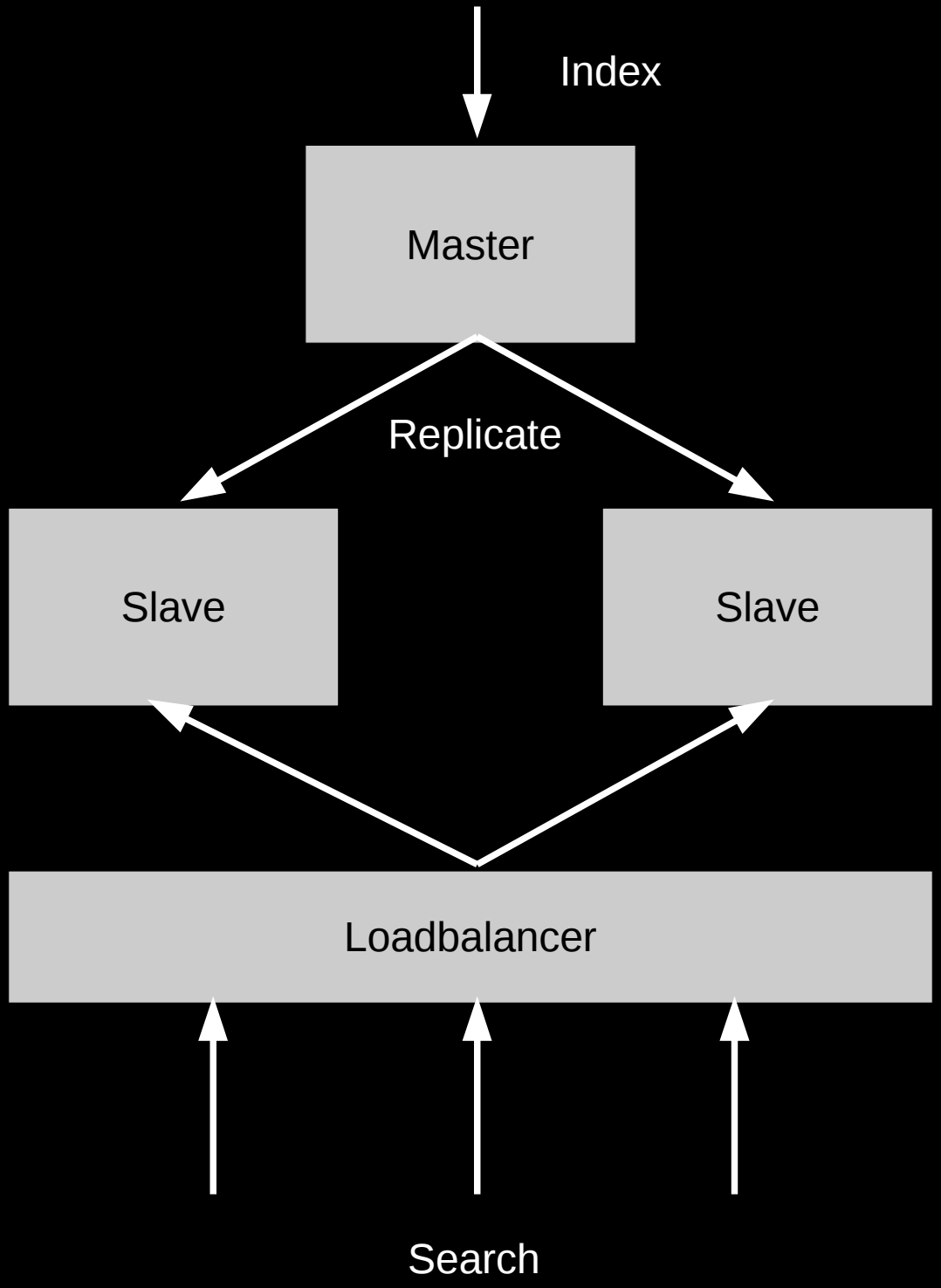

Webapp
Client
XML, JSON, JavaBin, Ruby, ...
http
Webapp
Solr
Lucene

```
SolrServer server = new CommonsHttpSolrServer("http://localhost:8082/solr/");

SolrInputDocument document = new SolrInputDocument();
document.addField("path", "/tmp/foo");
document.addField("title", "Apache Karaf");
document.addField("category", "OSGi");
document.addField(„category“, „Integration“);

server.add(document);
server.commit();
```

```
SolrQuery solrQuery = new SolrQuery("apache");
solrQuery.setQueryType("/jugka");
QueryResponse response = server.query(solrQuery);
assertEquals(1, response.getResults().size());
assertEquals("Apache Karaf", response.getResults().get(0).get("title"));
```

DEMO

```
<str name="facet">on</str>
<str name="facet.field">category</str>
<str name="facet.field">speaker</str>
<str name="facet.mincount">1</str>


<lst name="facet_fields">
  <lst name="category">
    <int name="Integration">2</int>
    <int name="Architektur">1</int>
    <int name="OSGi">1</int>
  </lst>
  <lst name="speaker">
    <int name="Christian Schneider">2</int>
    <int name="Achim Nierbeck">1</int>
  </lst>
</lst>
```


Index
Master
Replicate
Slave
Slave
Loadbalancer
Search

- Geospatial Search
- More like this
- Spellchecker
- Suggester
- Result Grouping
- Function Queries
- Sharding

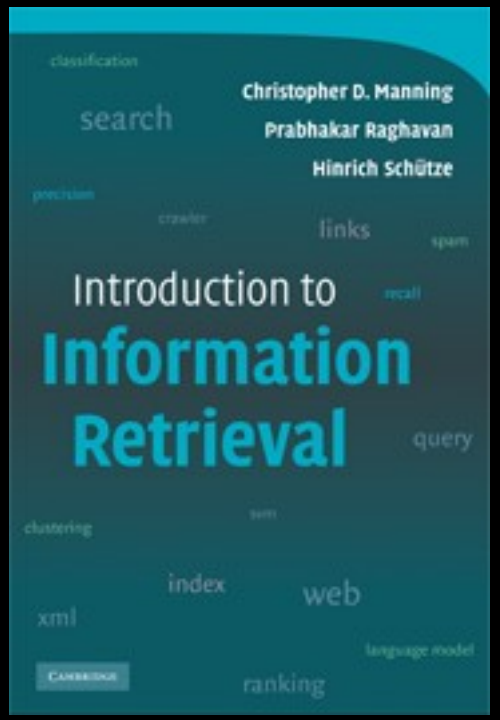

elasticsearch.

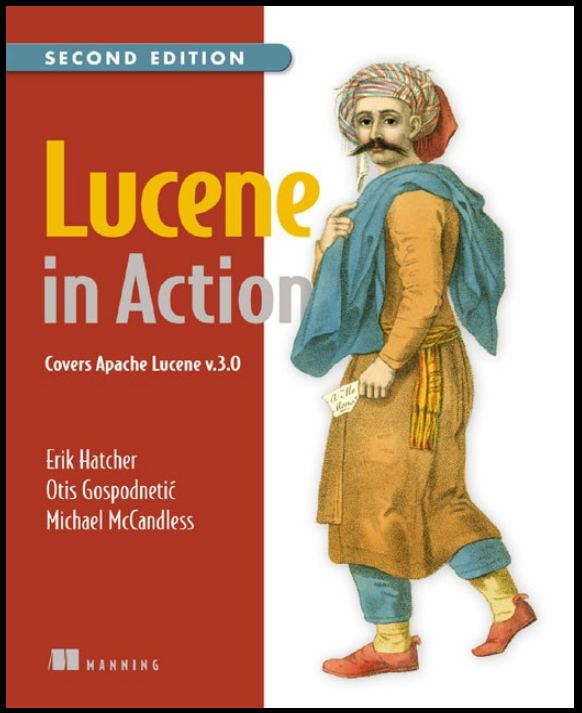
- Suchserver basierend auf Apache Lucene
- RESTful API
- Dokumentenorientiert (JSON)
- Schemafrei
- Distributed Search
- Near Realtime Search
- No Commits (Transaction Log)

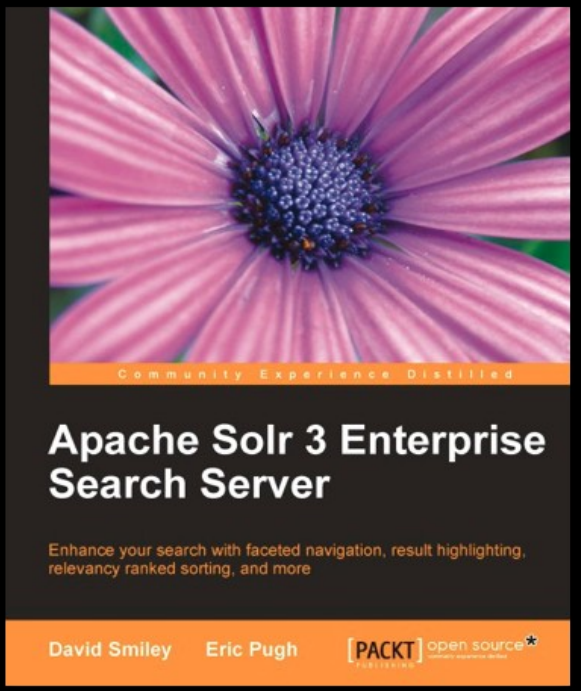
```
curl -XPOST 'http://localhost:9200/jugka/talk/' -d '{
    "speaker" : "Florian Hopf",
    "date" : "2012-07-04T19:30:00",
    "title" : "Suchen und Finden mit Lucene und Solr"}'

{"ok":true,"_index":"jugka","_type":"talk",
"_id":"CeltdivQRGSvLY_dBZv1jw","_version":1}
```

```
curl -XGET 'http://localhost:9200/jugka/talk/_search?q=solr'
{"took":29,"timed_out":false,"_shards":
{"total":5,"successful":5,"failed":0},"hits":
{"total":1,"max_score":0.054244425,"hits":
[{"_index":"jugka","_type":"talk","_id":"CeltdivQRGSvLY_dBZv1jw"
,"_score":0.054244425, "_source" : {
    "speaker" : "Florian Hopf",
    "date" : "2012-07-04T19:30:00",
    "title" : "Suchen und Finden mit Lucene und Solr"}
```

- http://lucene.apache.org
- http://tika.apache.org
- http://lucene.apache.org/solr/
- http://elasticsearch.org
- http://github.com/fhopf/lucene-solr-talk







http://nlp.stanford.edu/IR-book/

Vielen Dank!

http://www.florian-hopf.de
@fhopf