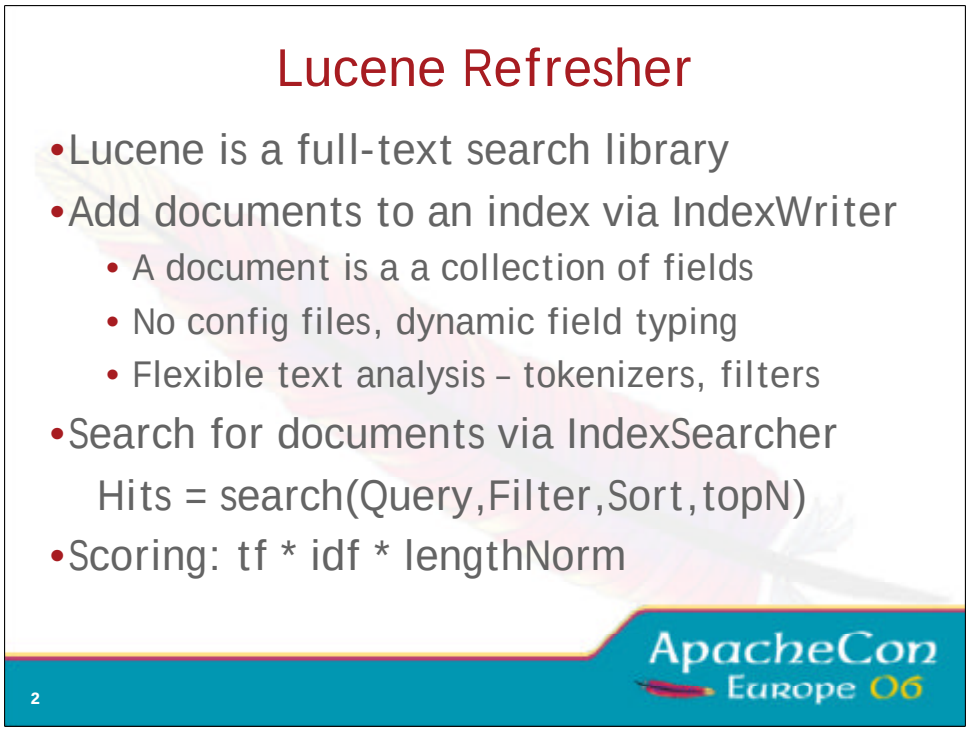
Apache Solr

Yonik Seeley
yonik@apache.org

29 June 2006
Dublin, Ireland

History

- Search for a replacement search platform
  - commercial: high license fees
  - open-source: no full solutions
- CNET grants code to Apache, Solr enters Incubator 17 Jan 2006
- Solr is a Lucene sub-project
- Users: CNET Reviews, CNET Channel, shopper.com, news.com, nines.org, krugle.com, oodle.com, booklooker.de



Scoring:

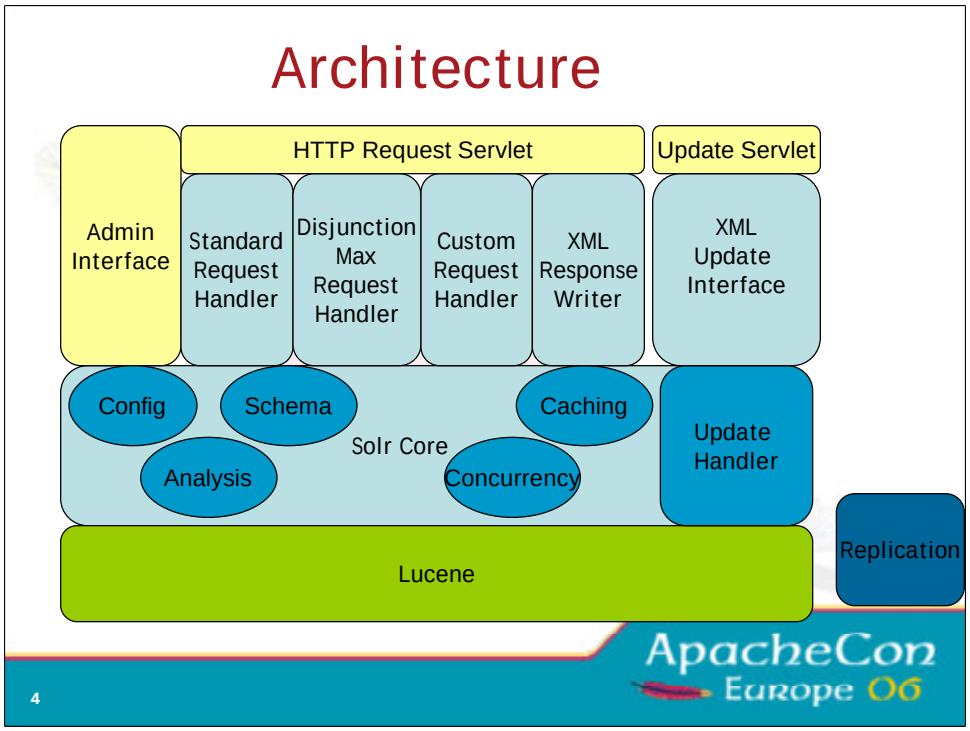
•tf = term frequency in the document – a document that contains more instances of the search term will score higher

•idf = inverse document frequency – rarer terms in the corpus are considered more important and score higher

•lengthNorm = matches in shorter fields score higher (balances out the tf factor)

What Is Solr

- A full text search server based on Lucene
- XML/HTTP Interfaces
- Loose Schema to define types and fields
- Web Administration Interface
- Extensive Caching
- Index Replication
- Extensible Open Architecture
- Written in Java5, deployable as a WAR



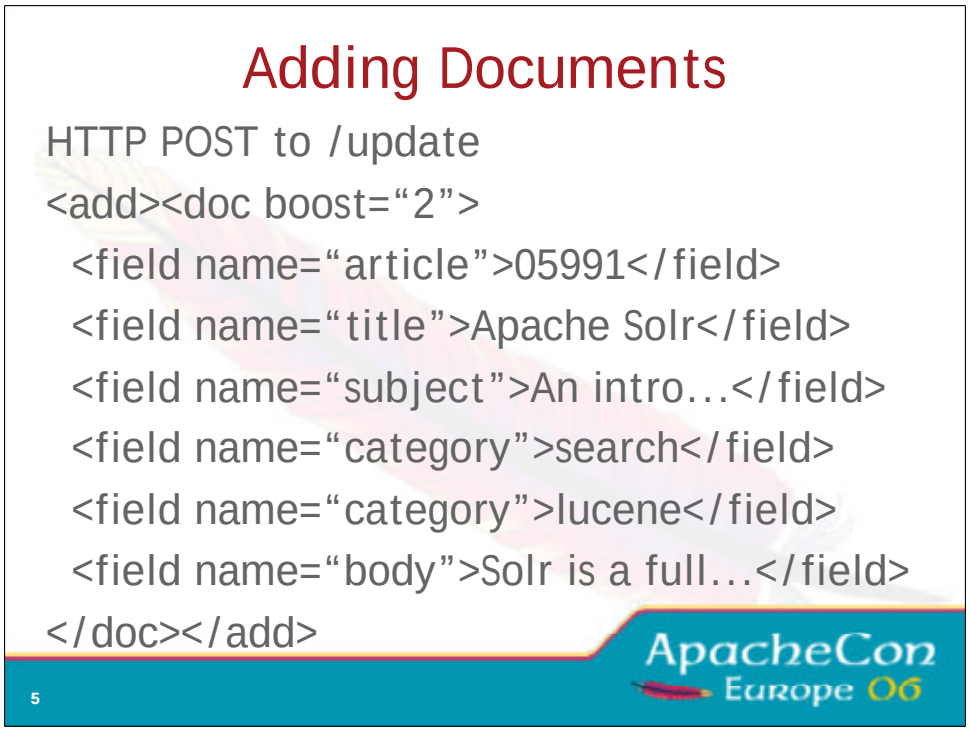
•Yellow components rely on a servlet container, all other components can run outside of a container or embedding or easier testing.

•Configuration is done via XML files

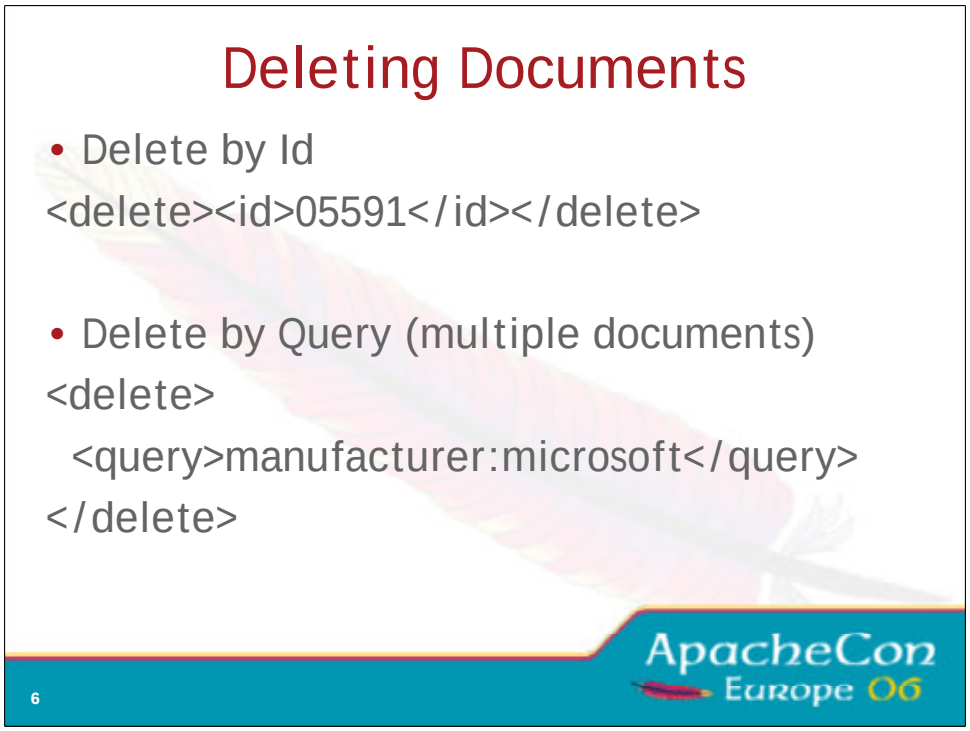
•Request handlers are pluggable, allowing one to do custom server-side processing in a more efficient manner

•Pluggable response writers for different output formats (different XML, JSON, etc)

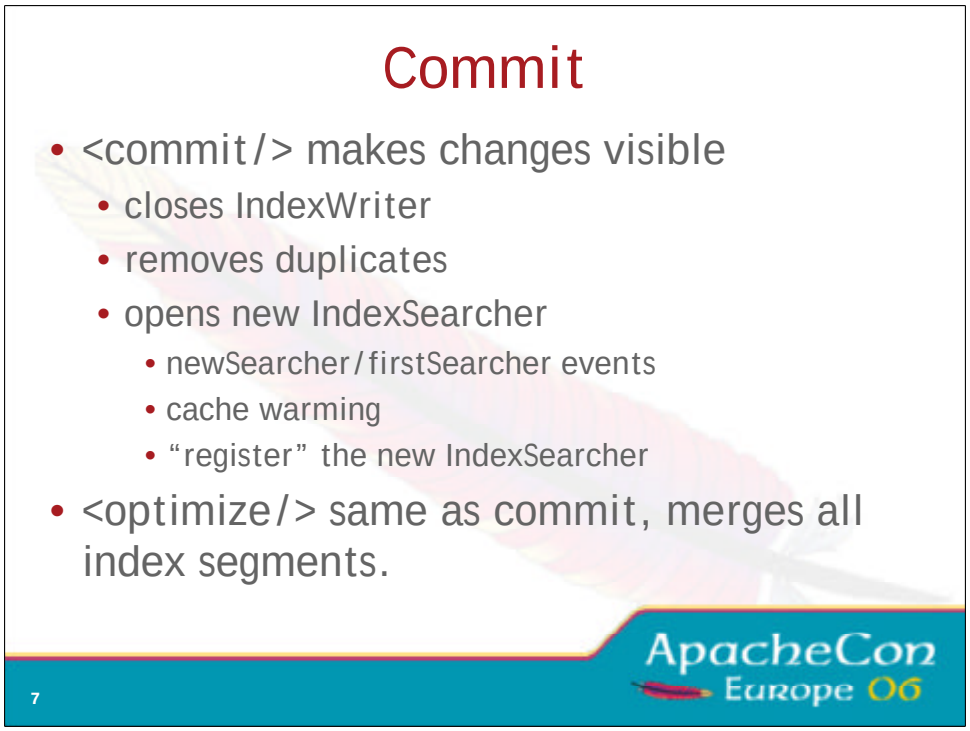
•Majority of replication implementation is outside the JVM – implemented as shell scripts, rsync, etc



- Adds are done via an HTTP POST of XML documents
- The "article" field is the unique id field, defined in the schema.
- An add will overwrite other documents with the same id.
- There may be boosts per document or per field – a boost makes a document score higher when querying against it
- Multi-valued fields like category are permissible… just add the field twice
- Adds are only concerned with content, not analysis... notice no indexed, stored, modifiers on the fields – that's the job of the schema.

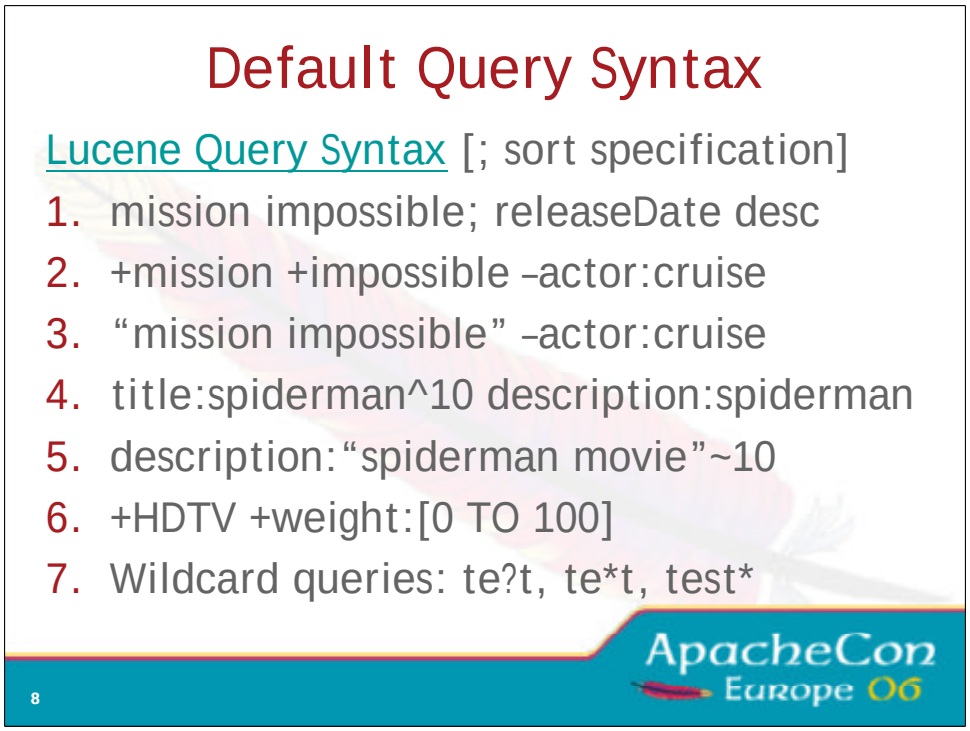


Deletes are also done simply by posting an XML document to the /update URL.



commit and optimize are rather heavyweight in Solr/Lucene, so updates should be batched.

After an index searcher becomes "registered", it can accept live search requests. A single searcher normally handles all live requests concurrently.

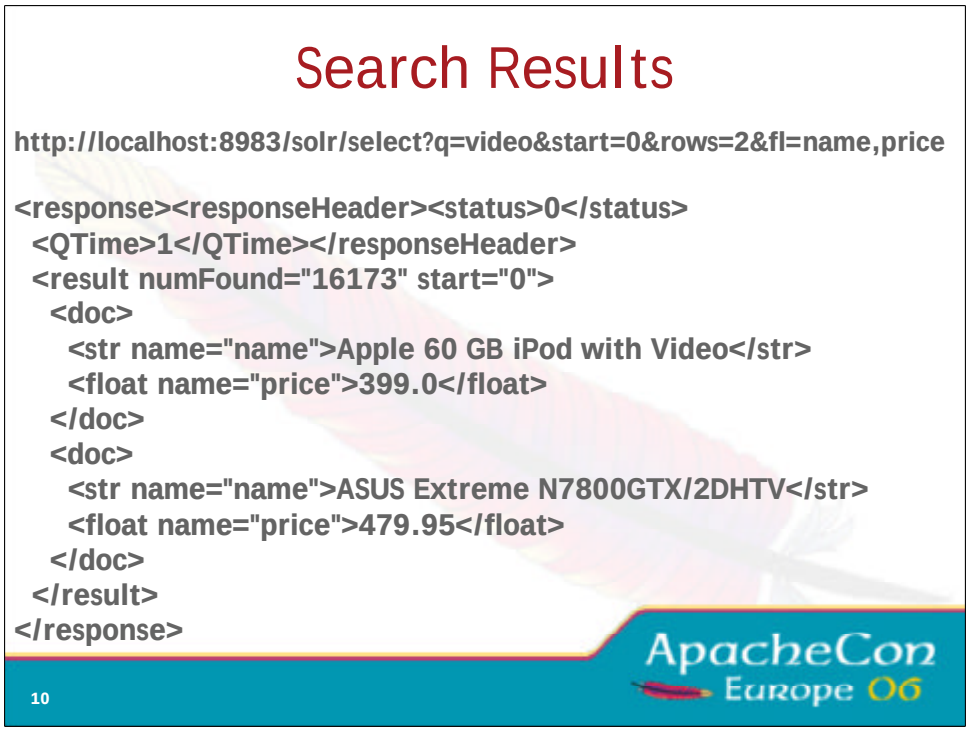


1. OR query: finds all documents with the word “mission” OR the word “impossible”, and sorts descending by the releaseDate field.
2. AND query: finds all documents with BOTH “mission” AND “impossible”
3. phrase query: finds all documents with “mission” directly followed by “impossible”
4. boosting: queries across two different fields, but boosts matches in the title field
5. phrase query slop: finds all documents where “spiderman” occures within 10 words of “movie”
6. range query: finds HDTV with weight between 0 and 100 inclusive.

Default Parameters

Query Arguments for HTTP GET/POST to /select

param	default	description
q		The query
start	0	Offset into the list of matches
rows	10	Number of documents to return
fl	*	Stored fields to return
qt	standard	Query type; maps to query handler
df	(schema)	Default field to search



- status, always 0 in a successful response
- QTime, the server-side query time in milliseconds
- numFound, the total number of documents matching the query
- start, the offset into the ordered list of results
- field types in <doc> include str, boolean, int, long, float, double, date, lst, arr
  - lst is a named list <lst><int name="foo">33</int><int name="bar">42</int></lst>
  - arr is an array <arr><int>33</int><int>42</int></arr>
- multivalued fields are returned in an <arr> element.

Caching

IndexSearcher's view of an index is fixed
- Aggressive caching possible
- Consistency for multi-query requests

filterCache – unordered set of document ids matching a query

resultCache – ordered subset of document ids matching a query

documentCache – the stored fields of documents

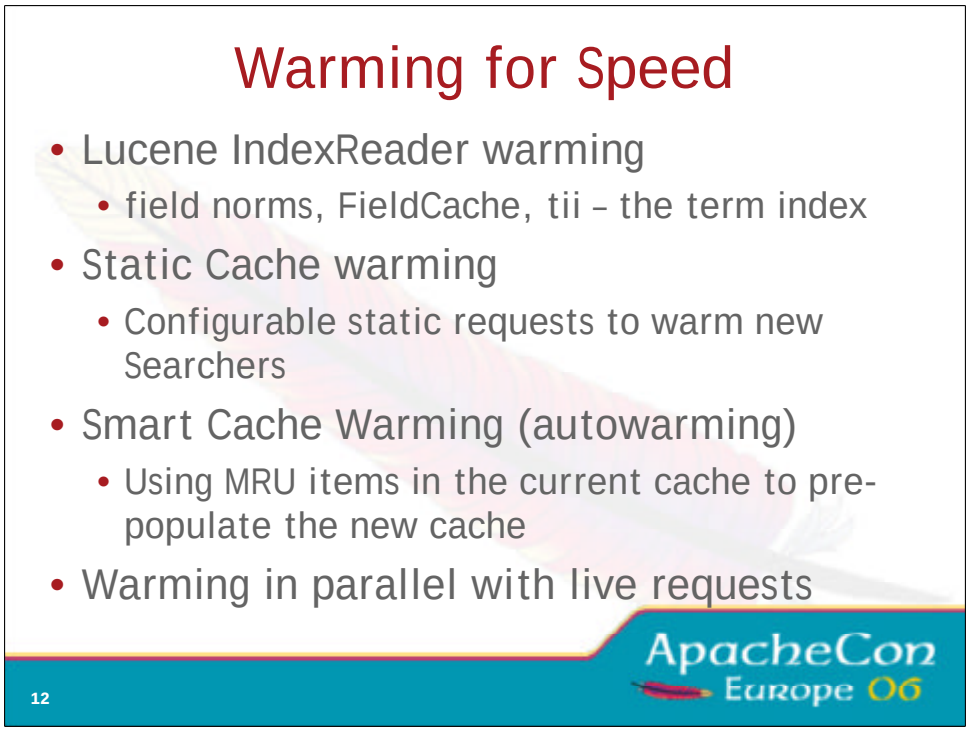
userCaches – application specific, custom query handlers

filterCache: key=Query, val=DocSet

resultCache: key=(Query,Sort,Filter), val=DocList

documentCache: key=docid, val=Document

userCache: key=Object, val=Object



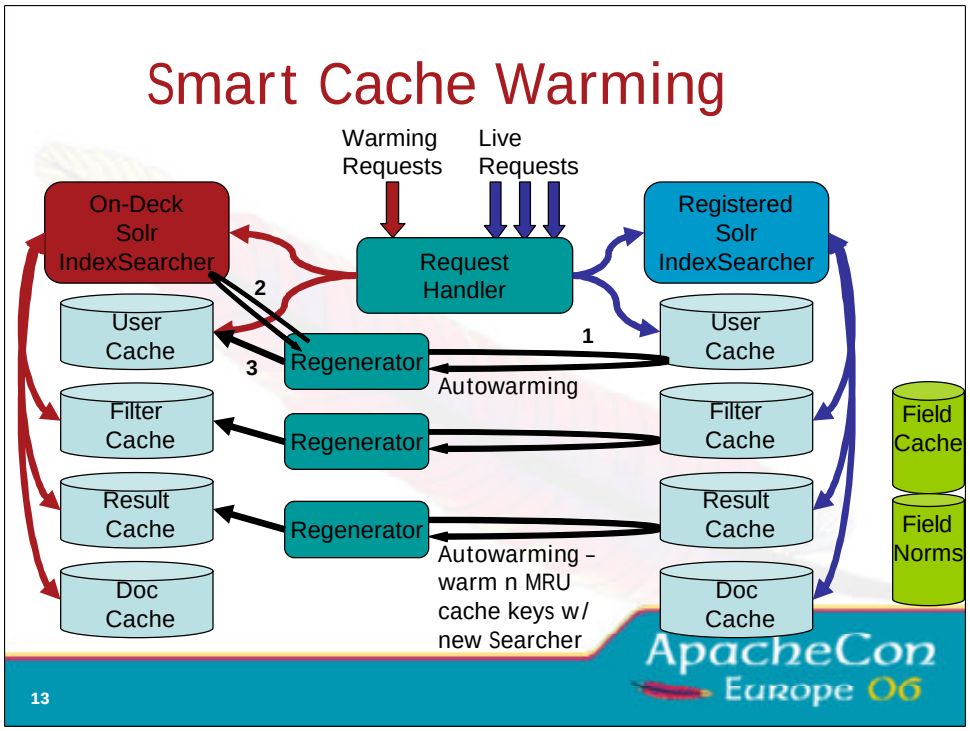
Lucene Internal Caches

•per-field norms are read the first time a field is searched on

•FieldCache entry is populated the first time each indexed field is used to sort

•tii – the term index is loaded on the first random access search

Warming of a new searcher happens in parallel with the current searcher handling live requests



•Static warming requests configured from solrconfig.xml, triggered by events (newSearcher or firstSearcher)

•Autowarming: The top keys from the old (current) cache are re-queried using the new IndexSearcher to pre-populate the new cache(s).

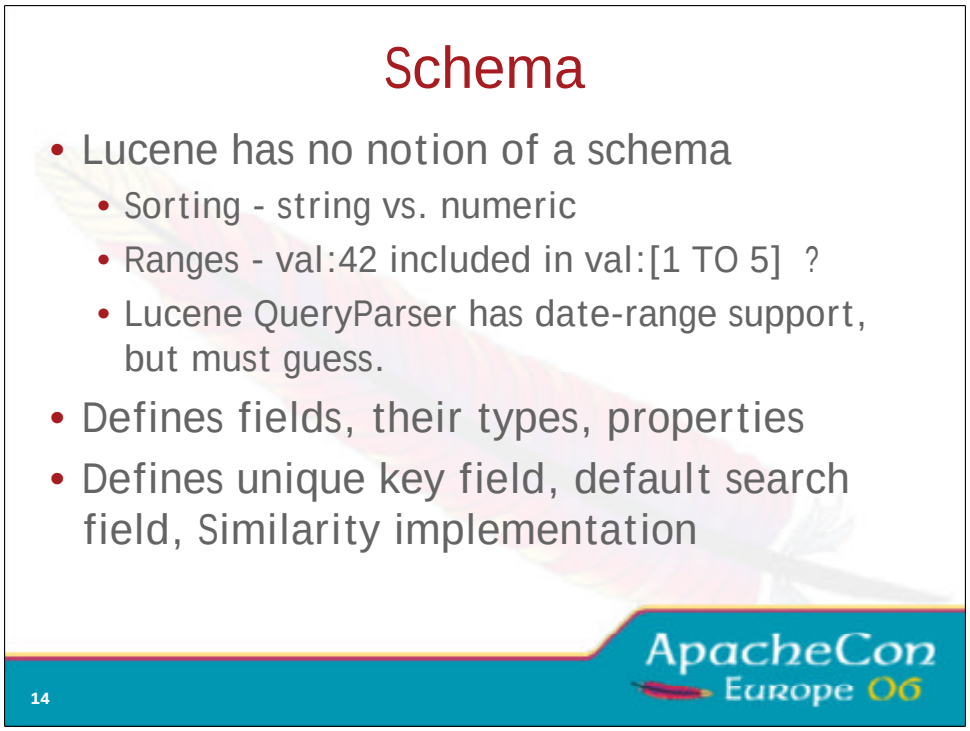
•Cache specific regenerators are used that take keys from old caches and use the new Searcher to pre-populate the new caches.

•The docCache does not have autowarming done since document ids change from one searcher to the next.

•Lucene also has some internal caches (FieldCache and field norms) than benefit from warming.

•After all warming is completed, the new IndexSearcher is registered, and starts serving live requests

•The old index searcher hangs around until all of it's requests have completed, then it is closed.



•Fields may still be sparse... just because a field is defined in the schema does not mean it needs to have a value for every document!

Field Definitions

- Field Attributes: name, type, indexed, stored, multiValued, omitNorms

```
<field name="id“        type="string"       indexed="true" stored="true"/>
<field name="sku“       type="textTight”  indexed="true" stored="true"/>
<field name="name“     type="text“          indexed="true" stored="true"/>
<field name=“reviews“ type="text“       indexed="true“ stored=“false"/>
<field name="category“ type="text_ws“ indexed="true" stored="true“
   multiValued="true"/>
```

- Dynamic Fields, in the spirit of Lucene!

```
<dynamicField name="*_i"  type="sint“    indexed="true"  stored="true"/>
<dynamicField name="*_s"  type="string“ indexed="true"  stored="true"/>
<dynamicField name="*_t"  type="text“    indexed="true"  stored="true"/>
```

15

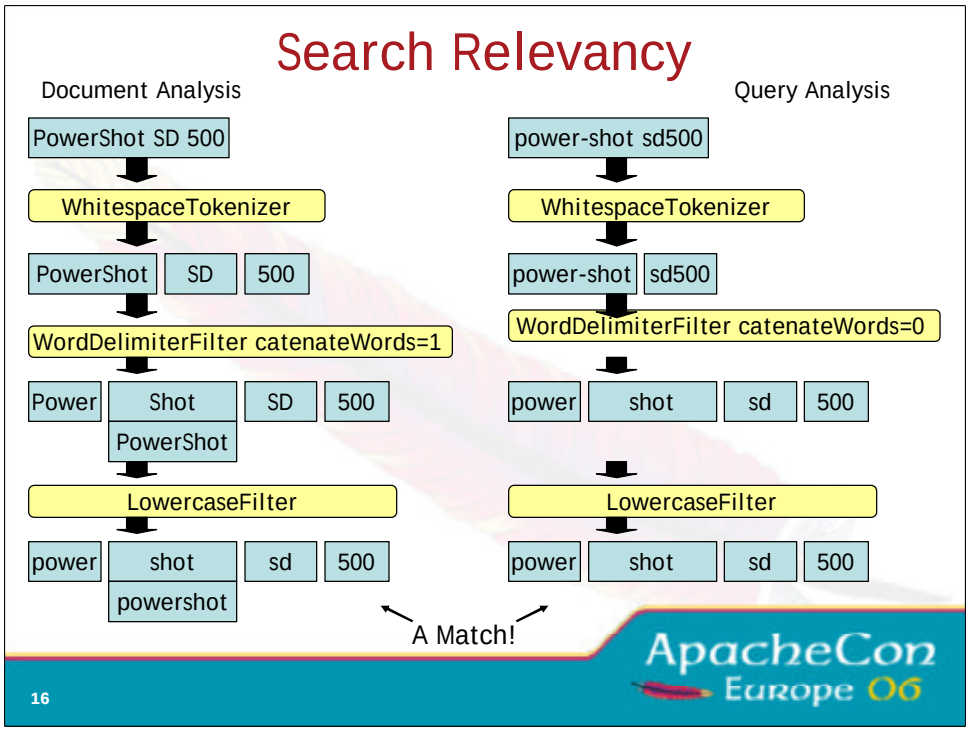
ApacheCon Europe 06

type: the field type, defined as a fieldtype declaration in the schema

indexed: a field must be indexed if you want to search on it or sort by it

stored: a field must be stored if you want it returned in search results

omitNorms: saves memory, good for sparse fields and when lengthNorm and index-time boosting is not needed

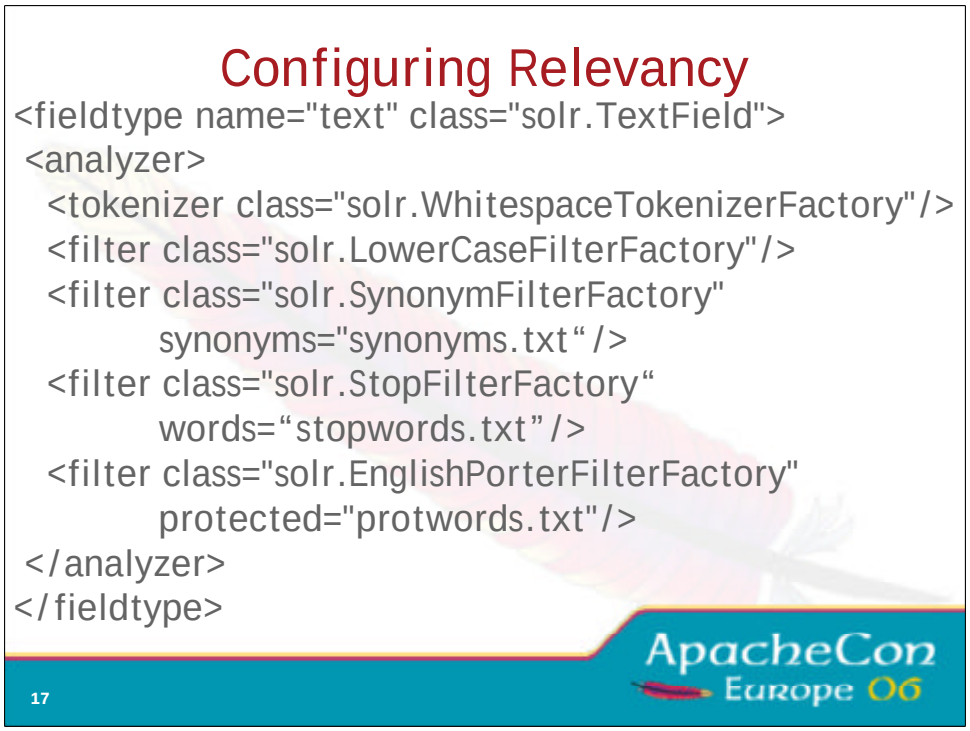


LHS: index analyzer, RHS: query analyzer

“PowerShot SD 500” is the value of a particular field, while “power-shot sd500“ is what someone typed into a search box.

When extra tokens can be injected (WordDelimiterFilter or SynonymFilter), you often want different analyzers for indexing and querying.

More than one token may be indexed at the same position (“shot” and “powershot” for example)



•fieldtype name is arbitrary... the behavior is governed by the implementing class

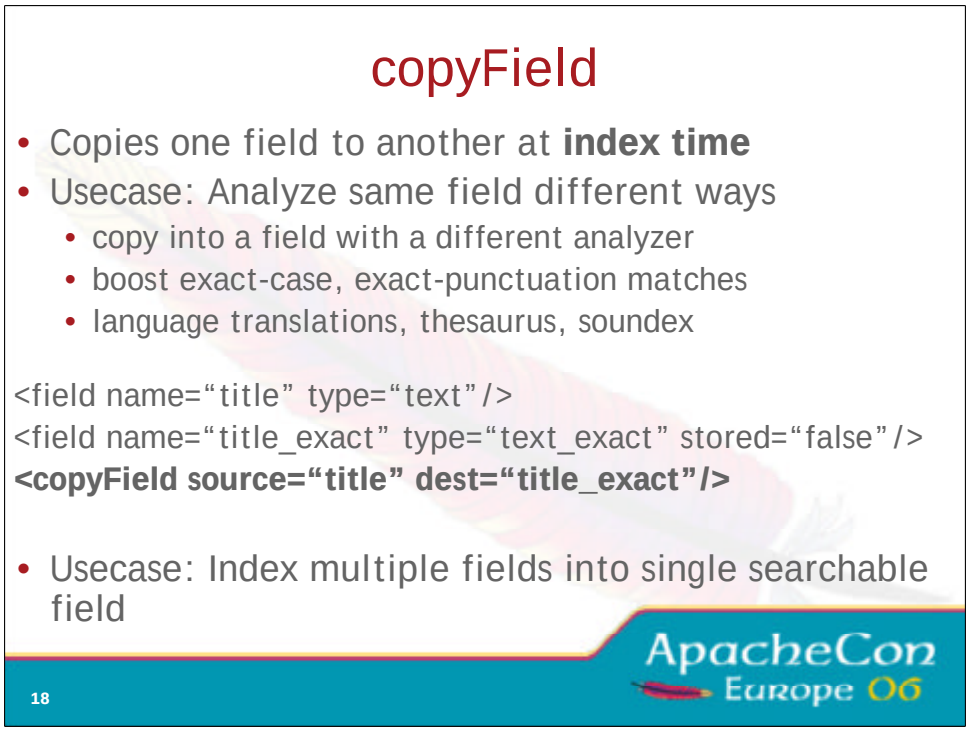
•one can set defaults for fields on the fieldtype such as indexed, stored, multiValued, omitNorms

•different analyzers may be configured for a TextField... an “index” analyzer, and a “query” analyzer.

•SynonymFilter may inject synonyms or “normalize” by replacing words.

•StopFilter – common words that don’t increase relevancy “and, or, a”, there is a default english stopword list, or one may configure via external file.

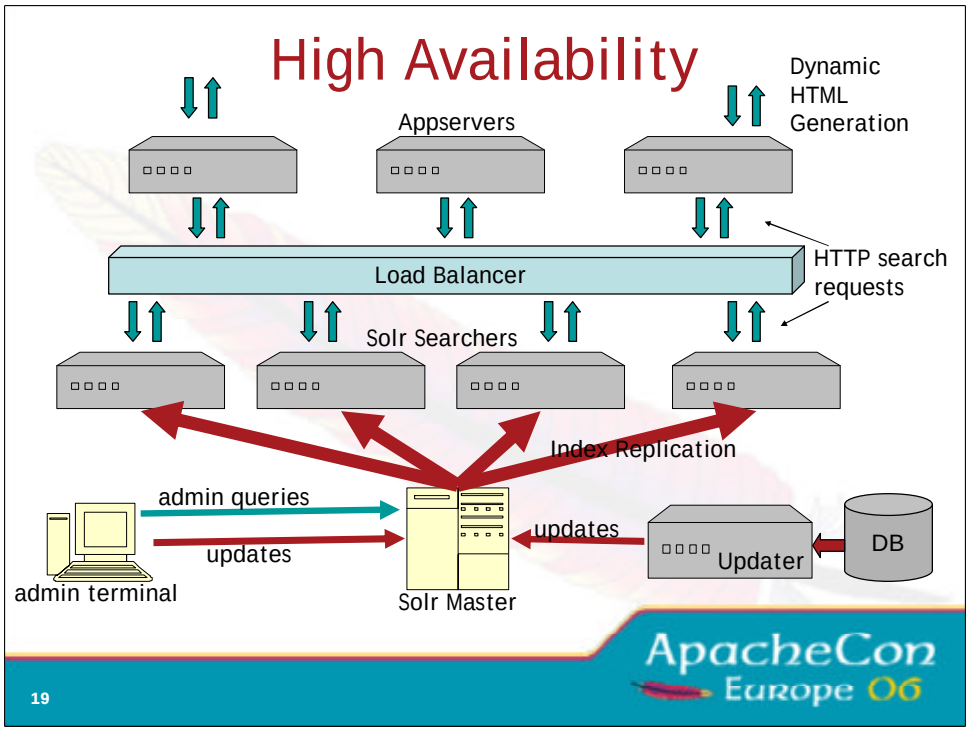
•EnglishPorterFilter – Porter stemmer - reduces related words, different tenses, to a common root.

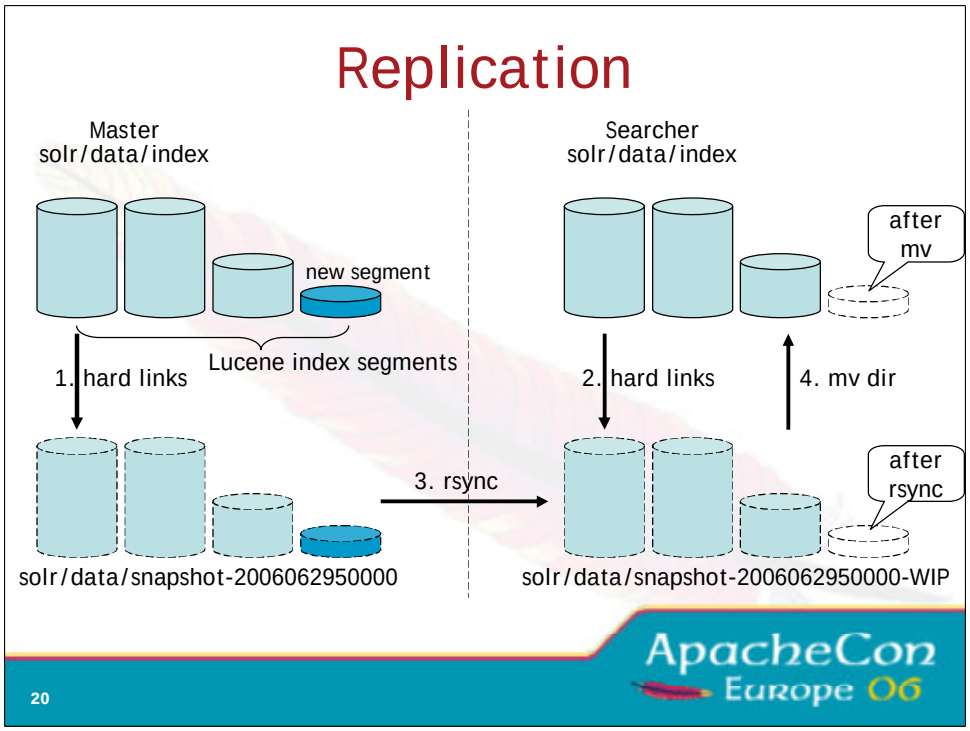


copyField decouples data providers (updaters) from details of text analysis and indexing… you can add new text analysis features in the future without having to change clients updating the index.

copyField only takes efffect when adding a new document to the index. It causes one field in the input to be copied to another.

Since the fields being copied to contain redundant data, they should normally be unstored.


High Availability
Dynamic
HTML
Generation
Appservers
Load Balancer
HTTP search
requests
Solr Searchers
Index Replication
admin queries
updates
admin terminal
Solr Master
updates
Updater
DB
ApacheCon
Europe 06
19



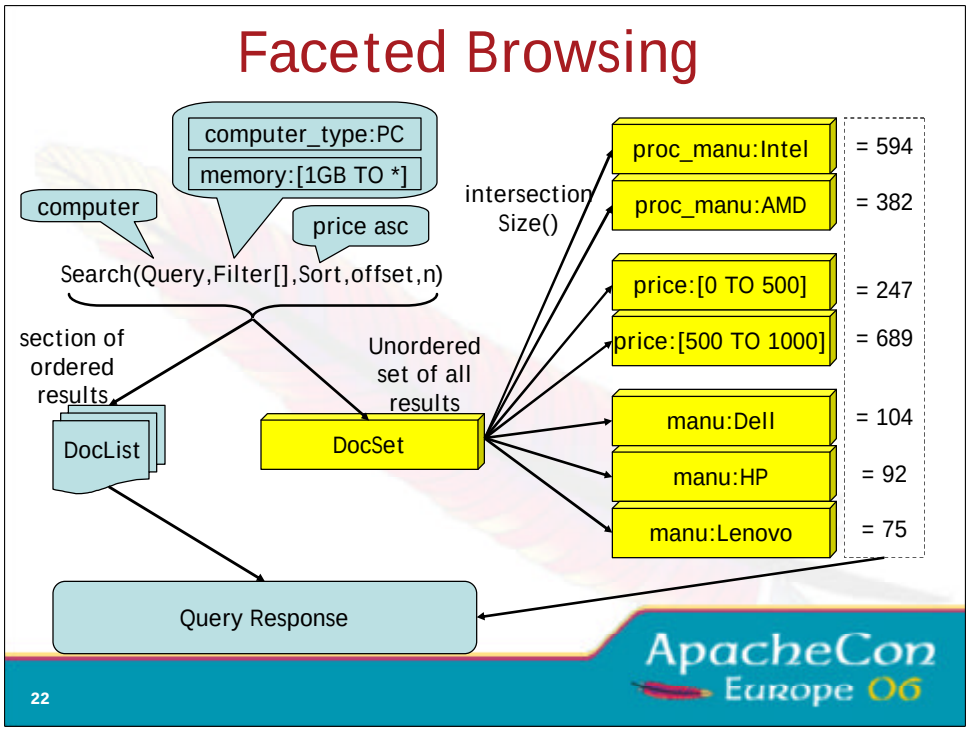
for performance reasons rsync is instructed to use timestamps & file sizes to determine which files have changed.

Step 1, taking an index snapshot, normally happens for every commit or optimize on the master. This is configurable.

A cron task on the searcher periodically checks the master for a new snapshot. If it exists, this kicks off steps 2-4.

Other replication strategies are possible!

Faceted browsing: listing category or facet counts, and allowing the user to narrow search results by those facets.



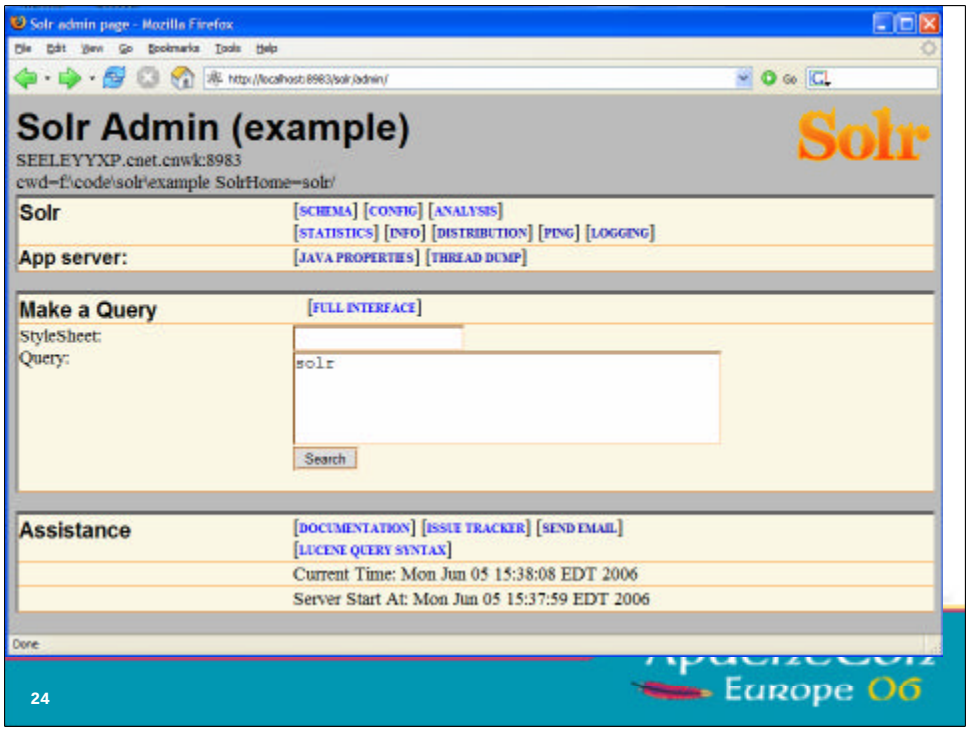
Faceted Browsing currently requires a custom query handler.

Solr Optimizations for Faceted Browsing:

- •OpenBitSet – 3 to 4 times faster than java.util.BitSet for calculating intersection sizes
- •HashDocSet – much faster and more memory efficient for smaller sets

Web Admin Interface

- Show Config, Schema, Distribution info
- Query Interface
- Statistics
  - Caches: lookups, hits, hitratio, inserts, evictions, size
  - RequestHandlers: requests, errors
  - UpdateHandler: adds, deletes, commits, optimizes
  - IndexReader, open-time, index-version, numDocs, maxDocs,
- Analysis Debugger
  - Shows tokens after each Analyzer stage
  - Shows token matches for query vs index


Solr admin page - Mozilla Firefox
Solr Admin (example)
SEELEYYXP.cnet.cnwk:8983
cwd=f:\code\solr\example SolrHome=solr/
Solr
[SCHEMA] [CONFIG] [ANALYSIS]
[STATISTICS] [INFO] [DISTRIBUTION] [PING] [LOGGING]
App server:
[JAVA PROPERTIES] [THREAD DUMP]
Make a Query
[FULL INTERFACE]
StyleSheet:
Query:
solr
Search
Assistance
[DOCUMENTATION] [ISSUE TRACKER] [SEND EMAIL]
[LUCENE QUERY SYNTAX]
Current Time: Mon Jun 05 15:38:08 EDT 2006
Server Start At: Mon Jun 05 15:37:59 EDT 2006
Done
24
Europe 06

Selling Points

- Fast
- Powerful & Configurable
- High Relevancy
- Mature Product
- Same features as software costing $$$
- Leverage Community
  - Lucene committers, IR experts
  - Free consulting: shared problems & solutions

Where are we going?

- OOTB Simple Faceted Browsing
- Automatic Database Indexing
- Federated Search
  - HA with failover
- Alternate output formats (JSON, Ruby)
- Highlighter integration
- Spellchecker
- Alternate APIs (Google Data, OpenSearch)

ApacheCon Europe 06

Resources

- WWW
  - http://incubator.apache.org/solr
  - http://incubator.apache.org/solr/tutorial.html
  - http://wiki.apache.org/solr/
- Mailing Lists
  - solr-user-subscribe@lucene.apache.org
  - solr-dev-subscribe@lucene.apache.org