Indexing & search Engines

Presentation of Apache Solr

ADALTAS

What is it

- Search indexing engine base on apache Lucene
- scalable and Fault tolerant
- replication
- load balanced

Apache Lucene

Apache Lucene TM is a high-performance, full-featured text search engine library written entirely in Java.

Provides full-text search, especially cross-platform.

Core of Apache Solr.

Information Retrieval

- Helps you to find material (ie documents) of an unstructured nature (like text) from large collections.

How good a SE performs

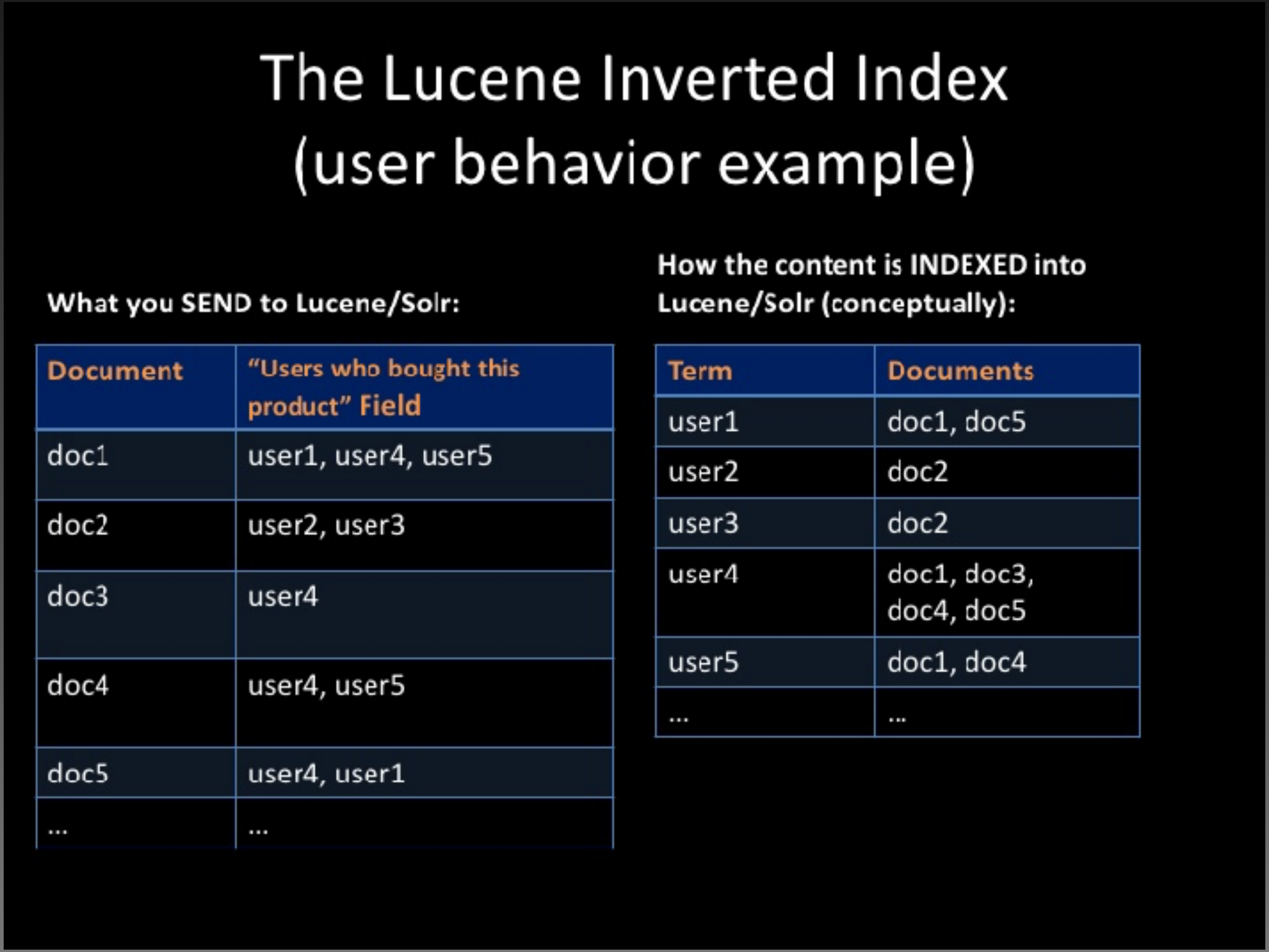
- Precision: Fraction of retrieved docs which are relevant to user’s search

- Recall: fraction of relevant instances that are retrieved

Indexing engine

- A document is composed of an id and a list of terms
- For each term a reference to the document containing it is created

Indexing engine



The Lucene Inverted Index
(user behavior example)

What you SEND to Lucene/Solr:

Document	"Users who bought this product" Field
doc1	user1, user4, user5
doc2	user2, user3
doc3	user4
doc4	user4, user5
doc5	user4, user1
…	…

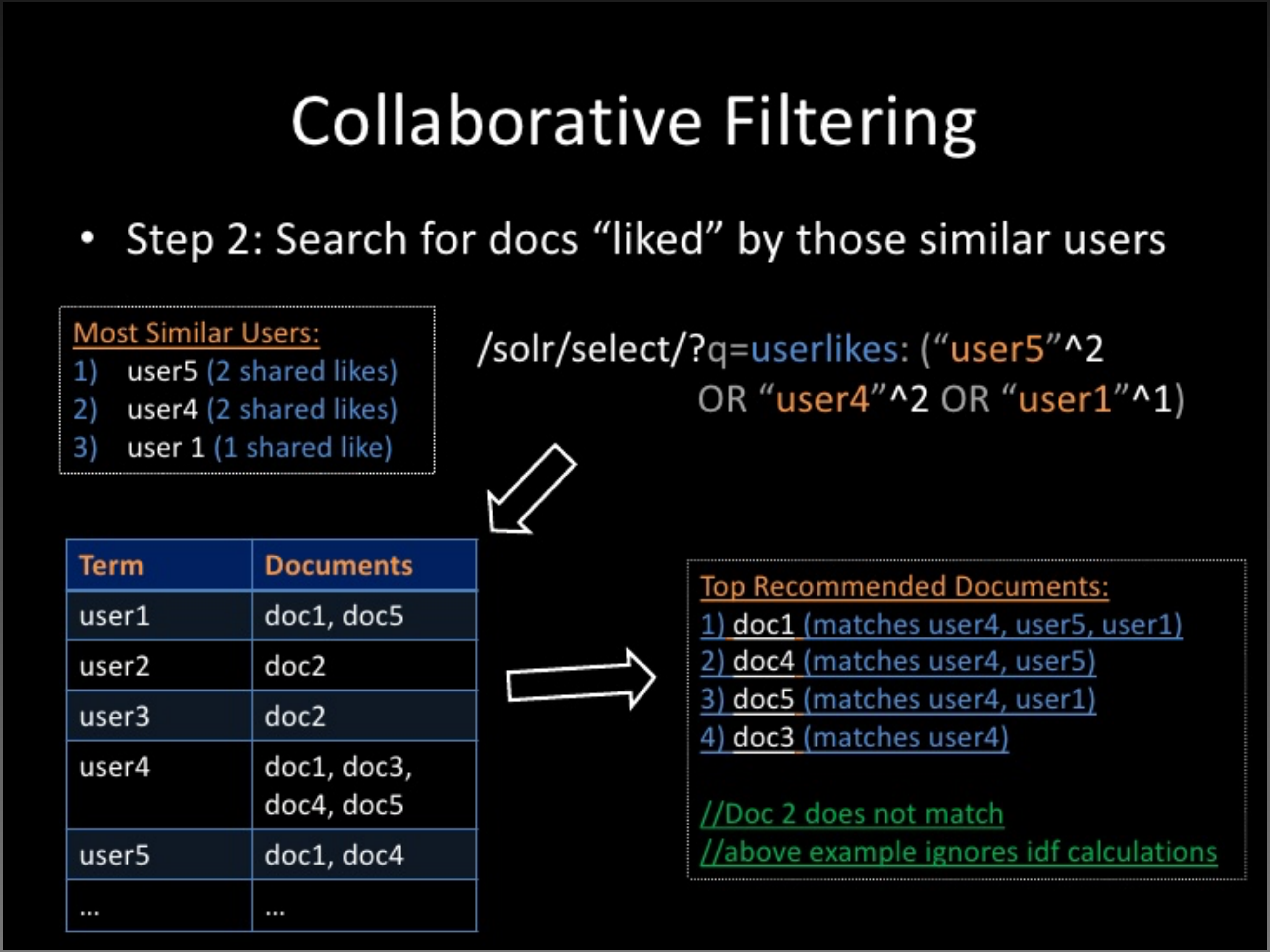
How the content is INDEXED into Lucene/Solr (conceptually):

Term	Documents
user1	doc1, doc5
user2	doc2
user3	doc2
user4	doc1, doc3, doc4, doc5
user5	doc1, doc4
…	…

What you can do

- Search text in document
- Autocomplete with suggestion
- Highlights of result
- More like this feature

What you can do



Collaborative Filtering

- Step 2: Search for docs "liked" by those similar users

Most Similar Users:
1) user5 (2 shared likes)
2) user4 (2 shared likes)
3) user 1 (1 shared like)

```
/solr/select/?q=userlikes: ("user5"^2
                  OR "user4"^2 OR "user1"^1)
```

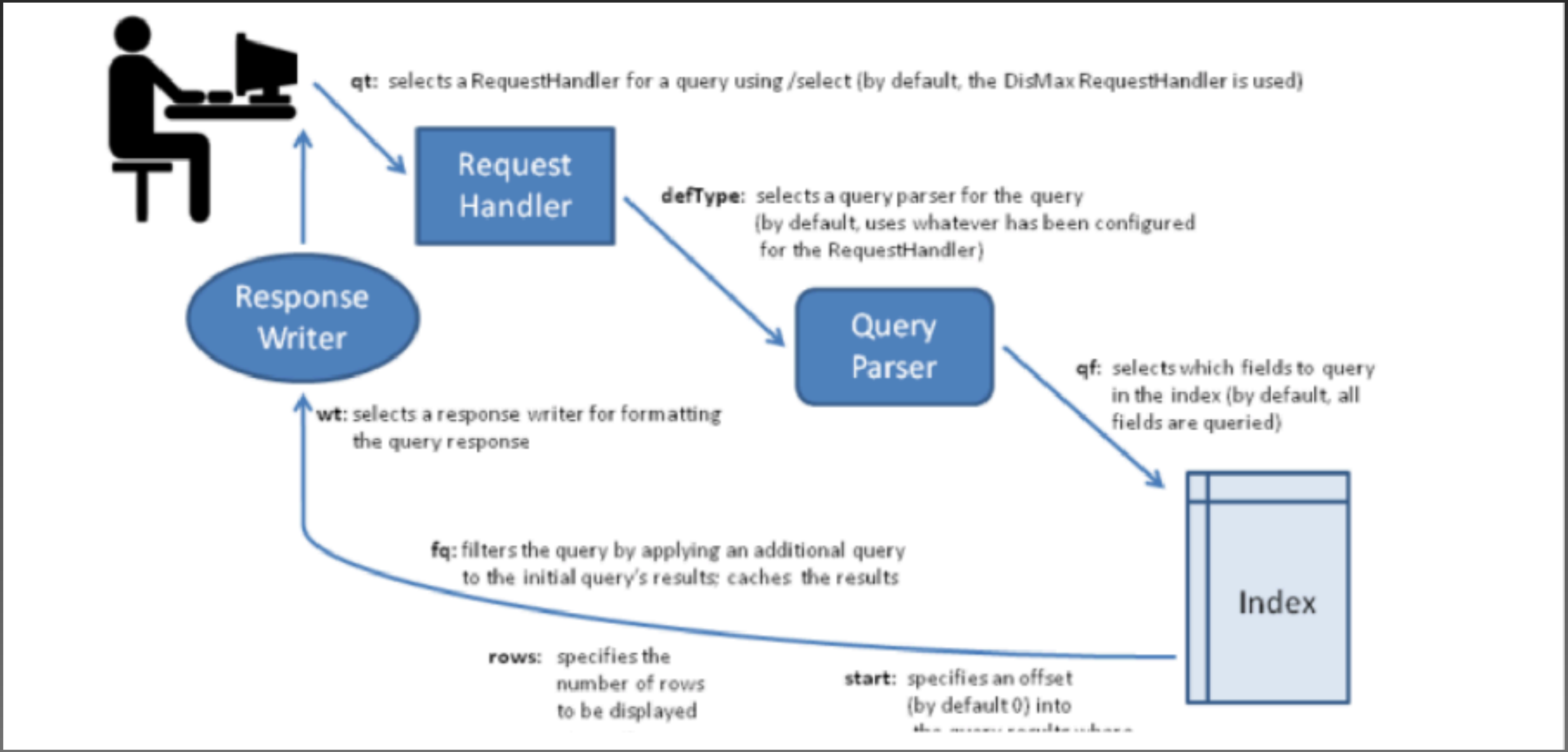
Term	Documents
user1	doc1, doc5
user2	doc2
user3	doc2
user4	doc1, doc3, doc4, doc5
user5	doc1, doc4
…	…

Top Recommended Documents:
1) doc1 (matches user4, user5, user1)
2) doc4 (matches user4, user5)
3) doc5 (matches user4, user1)
4) doc3 (matches user4)

//Doc 2 does not match
//above example ignores idf calculations

Building a solr-powered recommandation engine by Trey Grainger

How Search works


qt: selects a RequestHandler for a query using /select (by default, the DisMax RequestHandler is used)
Request Handler
defType: selects a query parser for the query (by default, uses whatever has been configured for the RequestHandler)
Query Parser
qf: selects which fields to query in the index (by default, all fields are queried)
Index
Response Writer
wt: selects a response writer for formatting the query response
fq: filters the query by applying an additional query to the initial query's results; caches the results
rows: specifies the number of rows to be displayed
start: specifies an offset (by default 0) into

Definitions

RequestHandler:

Solr plug-in that defines the logic to be used when Solr processes a request.

```
<requestHandler name="/query" class="solr.SearchHandler">
   <lst name="defaults">
     <str name="echoParams">explicit</str>
     <str name="wt">json</str>
     <str name="indent">true</str>
     <str name="df">text</str>
   </lst>
</requestHandler>
```

Definitions

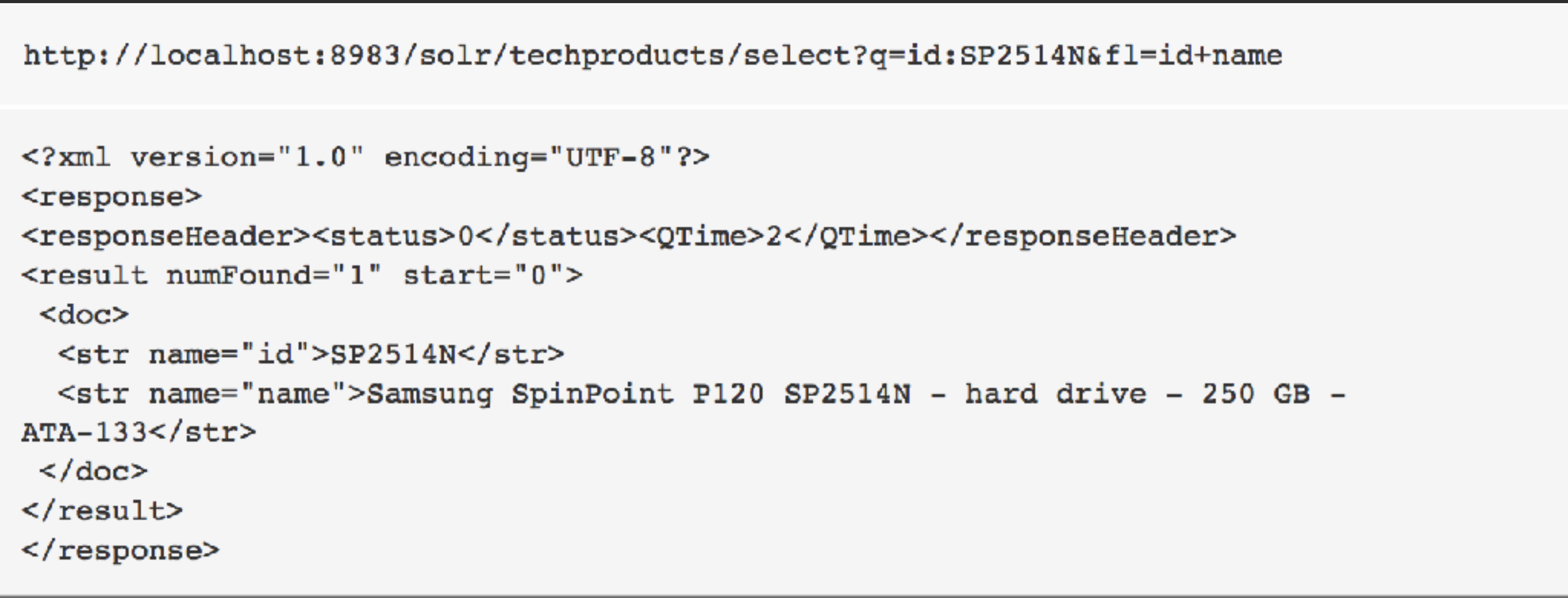
QueryParser:

It interprets the terms and parameters of a query.

- The Standard Query Parser
- The DisMax Query Parser
- The Extended DisMax Query Parser
- Other Parsers

Definitions

- The Standard Query Parser



```
http://localhost:8983/solr/techproducts/select?q=id:SP2514N&fl=id+name
```

```
<?xml version="1.0" encoding="UTF-8"?>
<response>
<responseHeader><status>0</status><QTime>2</QTime></responseHeader>
<result numFound="1" start="0">
 <doc>
  <str name="id">SP2514N</str>
  <str name="name">Samsung SpinPoint P120 SP2514N - hard drive - 250 GB -
ATA-133</str>
 </doc>
</result>
</response>
```

Definitions

- The DisMax query parser

Designed to process simple phrases (without complex syntax) entered by users and to search for individual terms across several fields using different weighting (boosts) based on the significance of each field

Definitions

- The DisMax query parser

Query Examples:

- http://localhost:8983/solr/techproducts/select?q=video&fl=name+score
- http://localhost:8983/solr/techproducts/select?defType=dismax&q=video
- http://localhost:8983/solr/techproducts/select?defType=dismax&q=video&fl=*,score
- http://localhost:8983/solr/techproducts/select?defType=dismax&q=video&qf=features^20.0+text^0.3
- http://localhost:8983/solr/techproducts/select?defType=dismax&q=video&bq=cat:electronics^5.0&inStock:true)

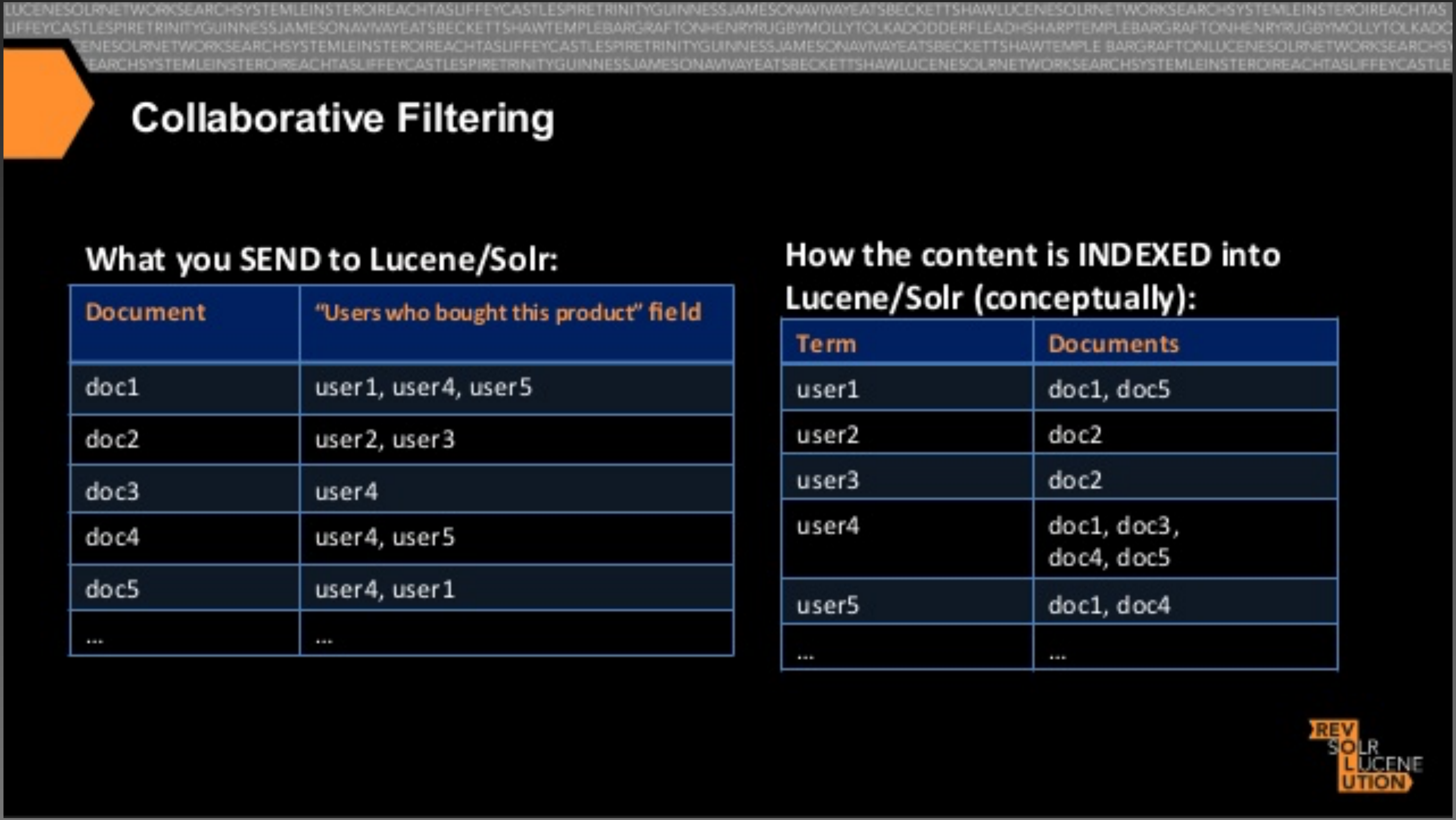
Definitions

Query Filter:

Runs a query against the entire index and caches the results. Performs queries at search time against data already in the index

Response Writer

Example: collaborative filtering



Collaborative Filtering

What you SEND to Lucene/Solr:

Document	"Users who bought this product" field
doc1	user1, user4, user5
doc2	user2, user3
doc3	user4
doc4	user4, user5
doc5	user4, user1
…	…

How the content is INDEXED into Lucene/Solr (conceptually):

Term	Documents
user1	doc1, doc5
user2	doc2
user3	doc2
user4	doc1, doc3, doc4, doc5
user5	doc1, doc4
…	…