

# SAN FRANCISCO MAY 25-26 2011

---

## Boosting Documents in Solr by Recency, Popularity, and User Preferences

Timothy Potter

[thelabdude@gmail.com](mailto:thelabdude@gmail.com), May 25, 2011



**LUCENE**  
**REVOLUTION**

Presented by

**lucid**  
IMAGINATION

Apache  
**Solr** 

*Lucene*

# What I Will Cover

- Recency Boost
- Popularity Boost
- Filtering based on user preferences



# My Background

- Timothy Potter
- Large scale distributed systems engineer specializing in Web and enterprise search, machine learning, and big data analytics.
- 5 years Lucene
  - **Search solution for learning management sys**
- 2+ years Solr
  - **Mobile app for magazine content**
    - *Solr + Mahout + Hadoop*
  - **FAST to Solr Migration for a Real Estate Portal**
  - **VinWiki: Wine search and recommendation engine**

# Boost documents by age

- Just do a descending sort by age = done?
- Boost more recent documents and penalize older documents just for being old
- Useful for news, business docs and local search



# Solr: Indexing

In schema.xml:

```
<fieldType name="tdate"  
  class="solr.TrieDateField"  
  omitNorms="true"  
  precisionStep="6"  
  positionIncrementGap="0" />
```

```
<field name="pubdate"  
  type="tdate"  
  indexed="true"  
  stored="true"  
  required="true" />
```

```
Date published = DateUtils.round(item.getPublishedOnDate  
( ), Calendar.HOUR);
```

# FunctionQuery Basics

- FunctionQuery: Computes a value for each document
  - Ranking**
  - Sorting**

constant	pow	<b>recip</b>
literal	abs	max
fieldvalue	log	<b>min</b>
ord	sqrt	<b>ms</b>
rord	map	sqedist - Squared Euclidean Dist
sum	scale	hsin, ghhsin - Haversine Formula
sub	query	geohash - Convert to geohash
product	linear	strdist

# Solr: Query Time Boost

- Use the recip function with the ms function:

```
q={!boost b=$recency v=$qq}&  
  recency=recip(ms(NOW/HOUR, pubdate), 3.16e-11, 0.08, 0.05)&  
  qq=wine
```

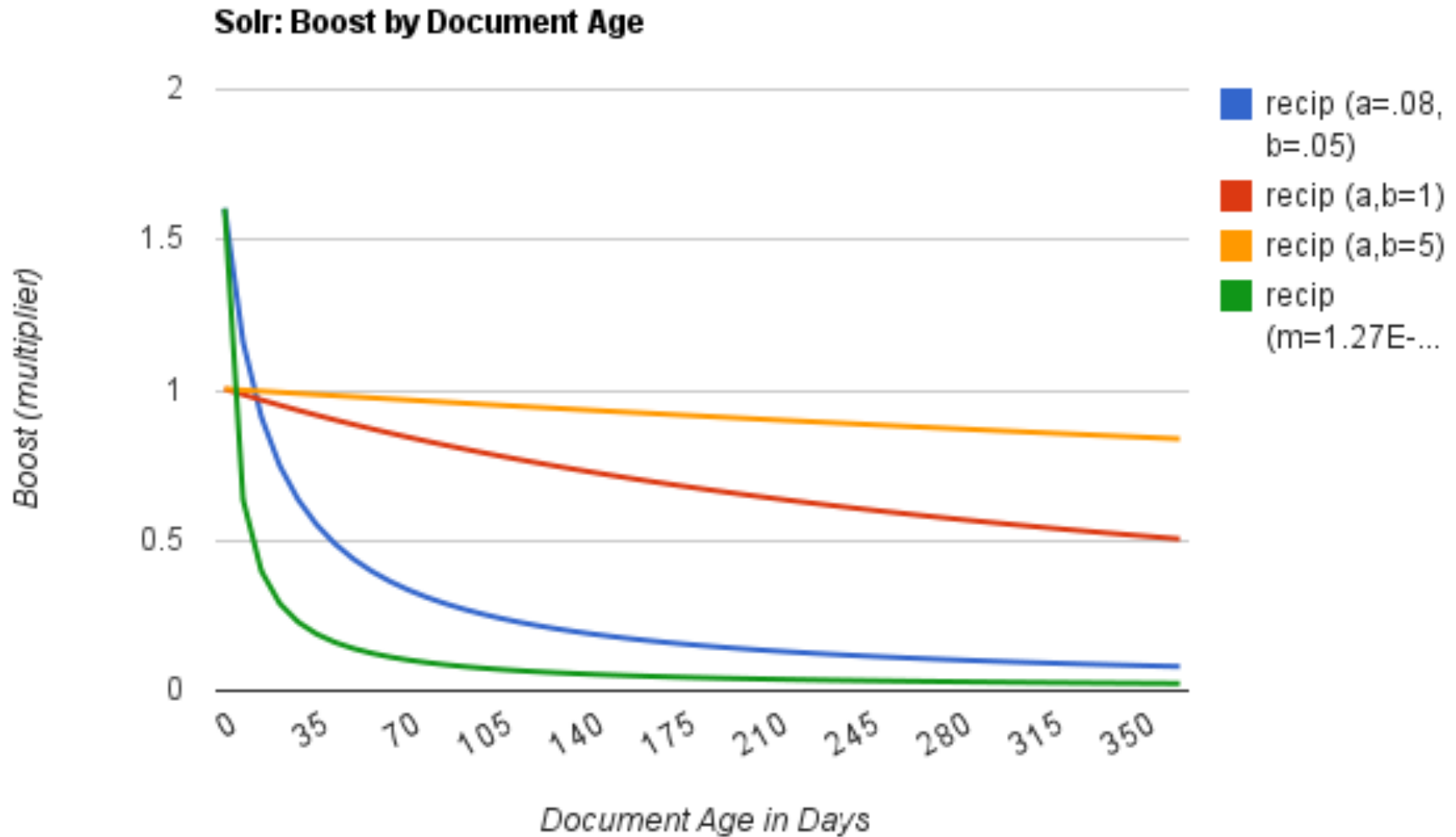
- Use edismax vs. dismax if possible:

```
q=wine&  
boost=recip(ms(NOW/HOUR, pubdate), 3.16e-11, 0.08, 0.05)
```

- Recip is a highly tunable function

- **recip(x,m,a,b) implementing  $a / (m * x + b)$**
- **$m = 3.16E-11$   $a = 0.08$   $b = 0.05$   $x = \text{Document Age}$**

# Tune Solr recip function



# Tips and Tricks

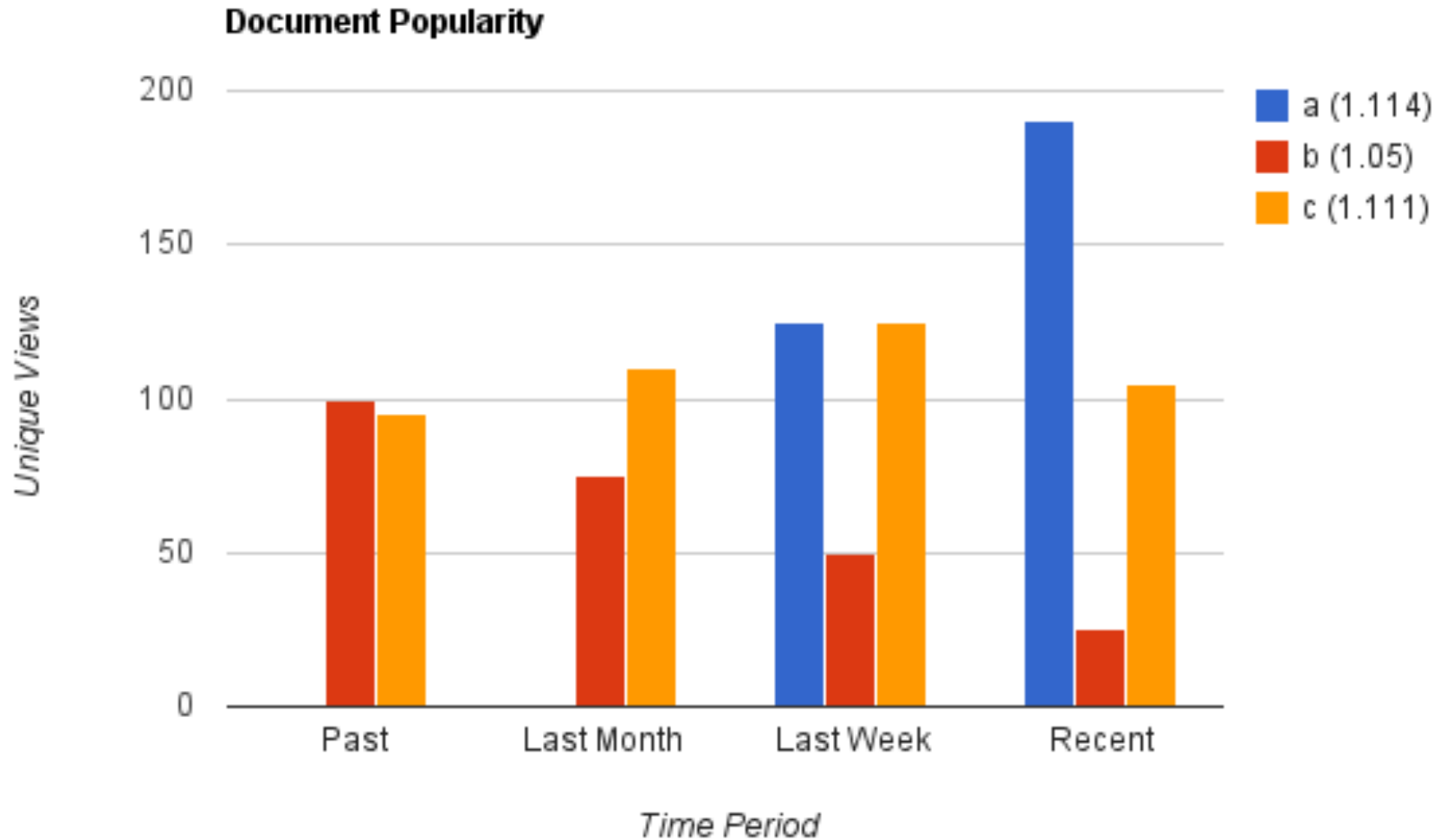
- Boost should be a multiplier on the relevancy score
- `{!boost b=}` syntax confuses the spell checker so you need to use **spellcheck.q** to be explicit  
**`q={!boost b=$recency v=$qq}&spellcheck.q=wine`**
- Bottom out the old age penalty using min:
  - **`min(recip(...), 0.20)`**
- Not a one-size fits all solution – academic research focused on when to apply it

# Boost by Popularity



- Score based on number of unique views
- Not known at indexing time
- View count should be broken into time slots

# Popularity Illustrated



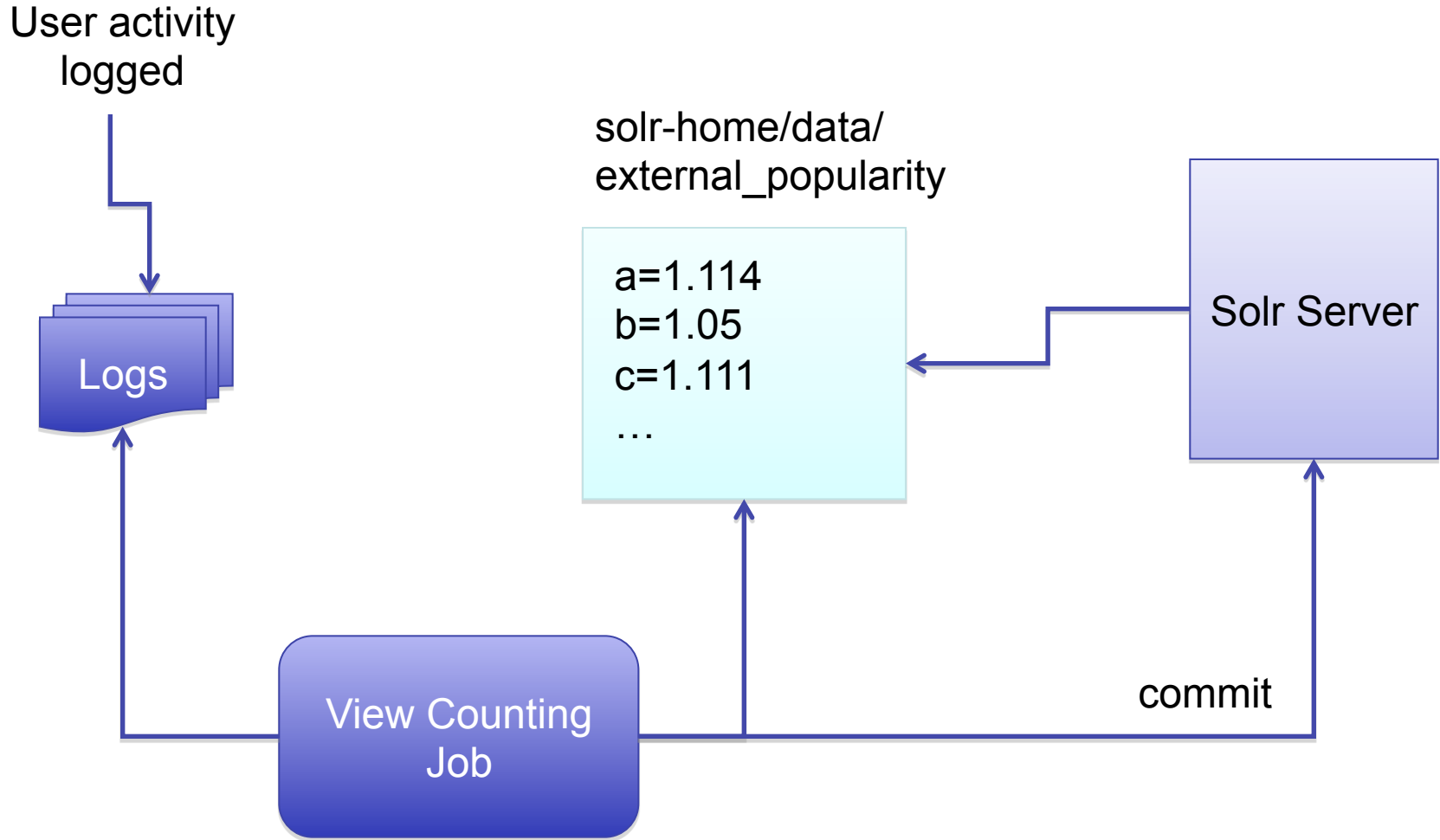
# Solr: ExternalFileField

In schema.xml:

```
<fieldType name="externalPopularityScore"  
  keyField="id"  
  defVal="1"  
  stored="false" indexed="false"  
  class="solr.ExternalFileField"  
  valueType="pfloat"/>
```

```
<field name="popularity"  
  type="externalPopularityScore" />
```

# Popularity Boost: Nuts & Bolts



# Popularity Tips & Tricks

- For big, high traffic sites, use log analysis
  - **Perfect problem for MapReduce**
  - **Take a look at Hive for analyzing large volumes of log data**
- Minimum popularity score is 1 (not zero) ... up to 2 or more
  - **$1 + (0.4 * \text{recent} + 0.3 * \text{lastWeek} + 0.2 * \text{lastMonth} \dots)$**
- Watch out for spell checker “buildOnCommit”

# Filtering By User Preferences

- Easy approach is to build basic preference fields in to the index:
  - **Content types of interest – content\_type**
  - **High-level categories of interest - category**
  - **Source of interest – source**
  
- We had too many categories and sources that a user could enable / disable to use basic filtering
  - **Custom SearchComponent with a connection to a JDBC DataSource**

# Preferences Component

- Connects to a database
- Caches DocIdSet in a Solr FastLRUCache
- Cached values marked as dirty using a simple timestamp passed in the request

Declared in solrconfig.xml:

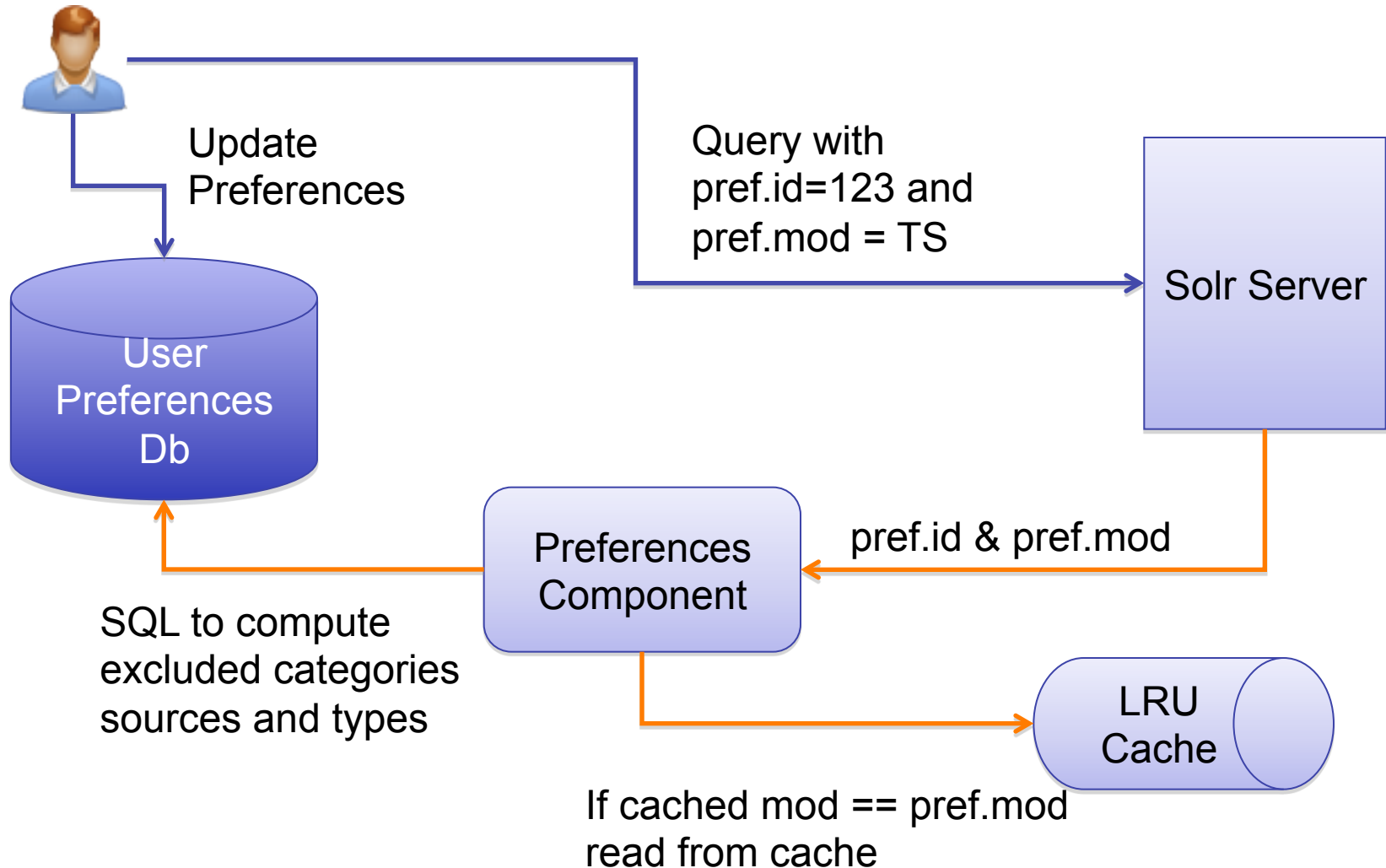
```
<searchComponent
  class="demo.solr.PreferencesComponent"
  name="pref">
  <str name="jdbcJndi">jdbc/solr</str>
</searchComponent>
```

# Preferences Filter

- Parameters passed in the query string:
  - **pref.id = primary key in db**
  - **pref.mod = preferences modified on timestamp**
    - *So the Solr side knows the database has been updated*
- Use simple SQL queries to compute a list of disabled categories, feeds, and types
  - **Lucene FieldCaches for category, source, type**
- Custom SearchComponent included in the list of components for edismax search handler

```
<arr name="last-components">  
  <str>pref</str>  
</arr>
```

# Preferences Filter in Action



# Wrap Up

- Use recip & ms functions to boost recent documents
- Use ExternalFileField to load popularity scores calculated outside the index
- Use a custom SearchComponent with a Solr FastLRUCache to filter documents using complex user preferences

# Contact

- Timothy Potter
  - [thelabdude@gmail.com](mailto:thelabdude@gmail.com)
  - <http://thelabdude.blogspot.com>
  - <http://www.linkedin.com/in/thelabdude>