

# Solr for newbies

<https://hectorcorrea.com/solr-for-newbies>



Hector Correa  
hector\_correa@princeton.edu  
Wintersession 2023  
Princeton, NJ

# Workshop Outline

- Introduction
  - Concepts, quick tour, installation
- Schema
  - Types, Fields, Tokenizers, Filters
- Searching
  - Search parameters, Facets, Highlighting
- Miscellaneous
  - Configuration, synonyms, spell checking

# I. Introduction

<https://github.com/hectorcorrea/solr-for-newbies/blob/main/tutorial.md#part-i-introduction>

# What is Solr?

"Solr is the popular, blazing-fast, open source enterprise **search platform** built on Apache Lucene." - Solr's Home Page

"Solr is a scalable, ready-to-deploy enterprise **search engine** that's optimized to search large volumes of **text-centric data** and return **results sorted by relevance**." - Solr in Action [p. 4]

# Why Solr at the Princeton University Library?

- We have lots of text centric searches
- Relevance in results is important to us
- Solr is free and open-source
- Facets

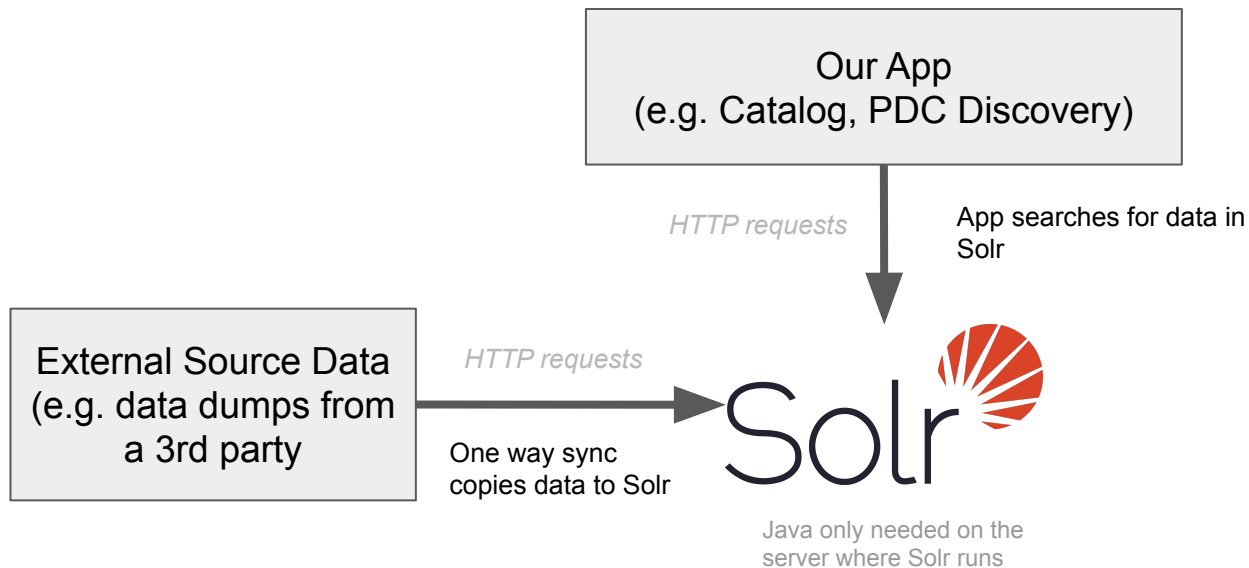
## Where do we use it?

- [Catalog](#)
- [PDC Discovery](#)
- [Figgy](#), [Finding Aids](#), [Digital PUL](#), and many more

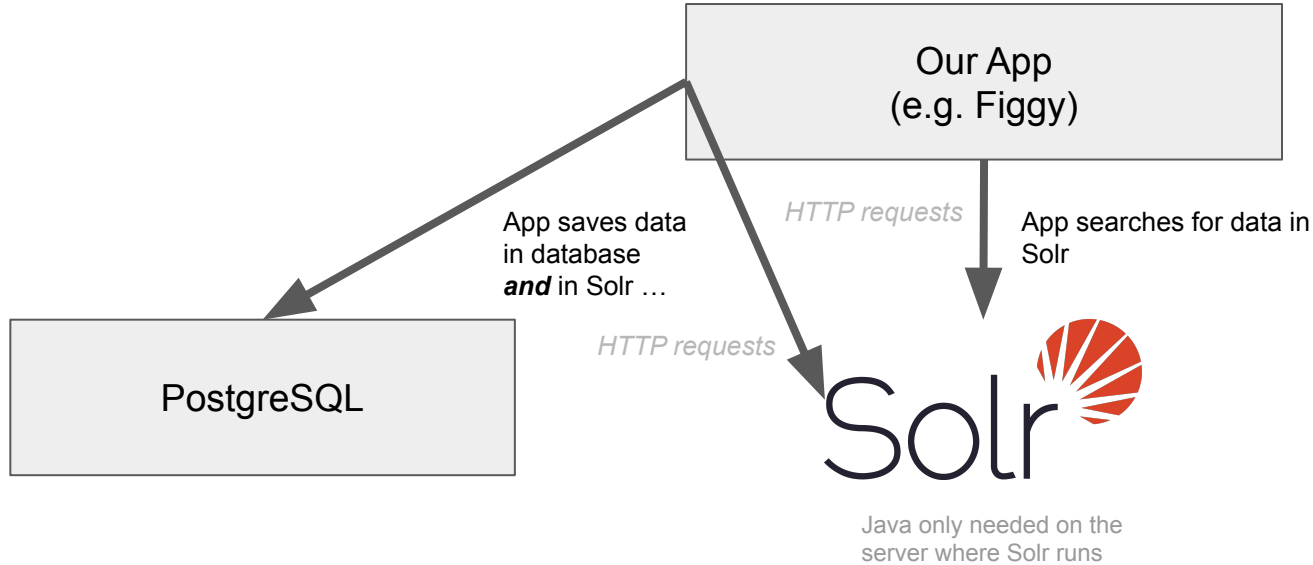
# Related tools and concepts

- [Apache Solr](#)
  - The Java app that we use, uses Lucene under the hood
  - Accessible via HTTP requests, no need to use Java to index or query
- [Apache Lucene](#) - brains behind Solr, used internally by Java applications
- [Apache Software Foundation](#) (ASF)
- [ElasticSearch](#)
  - Another product like Solr that also uses Lucene, not from ASF, weird licensing
  - Increasingly popular
- Indexing - when we ingest data into Solr
- Querying - when we search for data in Solr
- Tokenizing - a process of splitting a large text into smaller pieces (tokens)

# Solr in our apps



## Solr in our apps (cont.)





# Is Solr a database?

- Technically yes
  - Solr is a document-oriented database (a NoSQL database)
  - Solr is not a relational database like PostgreSQL, MySQL, Oracle, MS Access
- In practice no
  - Although we use Solr to store data...
  - ...we do this to **power the search feature** of our applications
  - ...and the source data lives somewhere else, in a “real” database

# Demo time



- Dashboard
- Logging
- Security
- Core Admin
- Java Properties
- Thread Dump
- bibdata ▾
- Overview
- Analysis
- Documents
- Paramsets
- Files
- Ping
- Plugins / Stats
- Query
- Replication
- Schema
- Segments info

Request-Handler (qt)  
/select

— common

q

\*,\*

q.op

OR

fq

sort

start, rows

0 10

fl

df

paramset(s)

Select paramset(s)...

wt

-----

☒ indent on

☐ debugQuery

defType

-----

☐ hl

☐ facet

☐ spatial

☐ spellcheck

Raw Query Parameters

JSON Query [?](#)

Execute Query

http://localhost:8983/solr/bibdata/select?indent=true&q.op=OR&q=\*,\*&useParams=

```
{
  "responseHeader": {
    "status": 0,
    "QTime": 2,
    "params": {
      "q": "*,*",
      "indent": "true",
      "q.op": "OR",
      "useParams": "",
      "_": "1673980970233"
    }
  },
  "response": {
    "numFound": 30424,
    "start": 0,
    "numFoundExact": true,
    "docs": [
      {
        "id": "00007345",
        "authors_other_txt_en": ["Giannakis, Georgios B."],
        "title_txt_en": "Signal processing advances in wireless and mobile communications /",
        "responsibility_txt_en": "edited by G.B. Giannakis ... [et al.]",
        "subjects_txt_en": ["Signal processing", "Wireless communication systems"],
        "_version_": 1755296534163881984
      },
      {
        "id": "00007476",
        "author_txt_en": "Sklar, Bernard",
        "author_date_s": "1927-",
        "title_txt_en": "Digital communications : fundamentals and applications /",
        "responsibility_txt_en": "Bernard Sklar.",
        "subjects_txt_en": ["Digital communications"],
        "_version_": 1755296534169124865
      },
      {
        "id": "00008006",
        "author_txt_en": "Dell'Oro, Suzanne Paul.",
        "title_txt_en": "Tunneling earthworms /",
        "responsibility_txt_en": "by Suzanne Paul Dell'Oro.",
        "subjects_txt_en": ["Earthworms", "Earthworms"],
        "subjects_form_txt_en": ["Juvenile literature"],
        "_version_": 1755296534169124866
      },
      {
        "id": "00008014",
        "author_txt_en": "Baskerville, David.",
        "title_txt_en": "Music business handbook and career guide /",
        "responsibility_txt_en": "David Baskerville.",
        "urls_ss": ["http://www.loc.gov/catdir/enhancements/fy0656/00008014-d.html", "http://www.loc.gov/catdir/enhancements/fy0656/00008014-t.html"],
        "subjects_txt_en": ["Music", "Music trade"],
        "subjects_form_txt_en": ["Handbooks, manuals, etc", "Handbooks, manuals, etc"],
        "_version_": 1755296534169124867
      }
    ]
  }
}
```

## Your turn: Installing Solr

<https://github.com/hectorcorrea/solr-for-newbies/blob/main/tutorial.md#installing-solr-for-the-first-time>

- Install Docker
- Create a Solr core named `bibdata`
- Load data to our `bibdata` Solr core
- Run basic queries

<https://github.com/hectorcorrea/solr-for-newbies/blob/main/tutorial.md#searching-for-documents>

- If don't want to install Solr

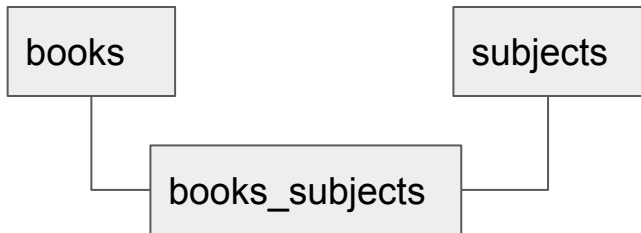
## II. Schema

<https://github.com/hectorcorrea/solr-for-newbies/blob/main/tutorial.md#part-ii-schema>

# Solr's Document Model

| id | book_title                     | subjects        |
|----|--------------------------------|-----------------|
| 1  | Princeton guide for dog owners | guides, animals |
| 2  | Princeton tour guide           | guides          |
| 3  | Cats and dogs                  | animals         |

## Relational Model



## Solr Document Model

```
solr_doc {  
  id: "1",  
  book_title: "Princeton guide for dog owners",  
  subjects: ["guides", "animals"]  
}
```

# Inverted Indexes - How Solr indexes our data

| id | book_title                     | subjects        |
|----|--------------------------------|-----------------|
| 1  | Princeton guide for dog owners | guides, animals |
| 2  | Princeton tour guide           | guides          |
| 3  | Cats and dogs                  | animals         |

## Traditional Index

| id | book_title                     |
|----|--------------------------------|
| 3  | cats and dogs                  |
| 1  | princeton guide for dog owners |
| 2  | princeton tour guide           |

## Inverted Index

| key       | ids  |
|-----------|------|
| princeton | 1, 2 |
| owners    | 1    |
| dogs      | 1, 3 |
| guide     | 1, 2 |
| tour      | 2    |
| cats      | 3    |

# Storing information in Solr

- It's **very** different from what we do in other databases
  - Remember that we use Solr to power our searches
  - i.e. we store data for *search* and display purposes *not* for preservation
- It's common to store the same field more than once
  - e.g. display value vs searchable value vs value for facets

# A field in Solr...

- Can be `String`, `Text`, `Number`, `Date`, ...
- Can be single-value or multi-value
- `String` values are stored as-is
  - No transformations at all, including extra spaces, `"hello "` != `"hello"`
- `Text` supports many transformations
  - `Text General` indexes `"Mechanism"` as `"mechanism"`
  - `Text English` indexes `"Mechanism"` as `"mechan"`
- A field can be stored and/or indexed
  - Stored if we need to display it to the user
  - Indexed if we need to search for values by this field
  - It is possible to index a field but not store it. In this case you can search values on it, get docs that match, but never retrieve the actual value *\*head-explodes\**



# A *text field* in Solr...

- Can have index and query ***analyzers***
- These analyzers alter the data as it's indexed or queried
- Two type of analyzers: tokenizer and filters
- Tokenizer
  - Breaks down text in tokens (e.g. "hello world" becomes "hello" and "world")
- Filters
  - Lower case ("HELLO" becomes "hello")
  - Strip punctuation ("hello!" becomes "hello")
  - Handle diacritics ("México" becomes "Mexico")
  - Extract stems of words (e.g. mechanism => mechan)
  - Drop stop words (e.g. the, and, or)
  - Handle Chinese-Japanese-Korean bigrams ([examples](#))
  - [Tons more](#)

# Example of text processed with tokenizer + filters

| Transformation   | Result                                       |  |
|------------------|--|--|
| Original text    | "Princeton guide for dog owners"             |  |
| Tokenizer        | "Princeton", "guide", "for", "dog", "owners" |  |
| Lowercase filter | "princeton", "guide", "for", "dog", "owners" |  |
| Stop word filter | "princeton", "guide", "dog", "owners"        |  |
| Stem filter      | "princeton", "guid", "dog", "owner"          |  |

# Fields in our `bibdata` Solr core

- Where did the fields in our `bibdata` Solr core come from?
- Fields, dynamic fields, and copy fields
- **Fields** define a type (string, text, date) and other properties like multi-value, stored, indexed, and so on.
- **Dynamic fields** are patterns to create fields on the fly.
- **Copy fields** are directives to copy the value of one field to another.

## Fields in our bibdata Solr core (cont)

```
{  
  "id":"000000018",  
  "author_txt_en":"Tarbell, H. S.",  
  "authors_other_txts_en":["Tarbell, Martha,"],  
  "title_txt_en":"The complete geography.",  
  "publisher_txt_en":"New York,",  
  "subjects_txts_en":["Geography"]  
}
```

## Fields in our `bibdata` Solr core (cont)

| Field in source data               | Schema  | Action   | Resulting field type      |
|------------------------------------|---|--|---------------------------|
| <code>id</code>                    | Matches <code>id</code> field defined as string     | Saves value  | <code>string</code>       |
| <code>author_txt_en</code>         | Matches <code>*_txt_en</code> dynamic field defined | Creates field <code>author_txt_en</code> and saves value                             | <code>text_en</code>      |
| <code>authors_other_txts_en</code> | Does <i>not</i> match any dynamic field defined     | Creates <code>authors_other_txts_en</code> field (guesses the type), and saves value | <code>text_general</code> |
|                                    |   |  |                           |

## Your turn: Customizing our schema

<https://github.com/hectorcorrea/solr-for-newbies/blob/main/tutorial.md#customizing-our-schema>

- Recreating our Solr core
- Handling `_txts_en` fields
- Customizing the title field
- Customizing the author field
- Customizing the subject field
- Populating the `_text_` field (copyField)
- Testing our changes

<https://github.com/hectorcorrea/solr-for-newbies/blob/main/tutorial.md#putting-it-all-together>

- Use the Analysis Screen to visualize the differences

# III. Searching

<https://github.com/hectorcorrea/solr-for-newbies/blob/main/tutorial.md#part-iii-searching>

- Query Parsers
  - Standard, DisMax, and eDisMax
- Searching
  - Basic parameters (`deftype`, `q`, `sort`, `rows`, `start`, `fl`, `fq`)
  - `qf`
  - `debug`
  - Ranking
  - Ranges
- Faceting
- Hit highlighting



# IV. Miscellaneous

<https://github.com/hectorcorrea/solr-for-newbies/blob/main/tutorial.md#part-iv-miscellaneous-optional>

- Solr's directories and configuration files
- Synonyms
- Request Handlers
- Search Components
- Spell checking

# Other recommended resources

- Book: [Solr in Action](#)
  - Old but still relevant
  - Great for general overview and in-depth analysis of some features
- Solr's official [reference guide](#)
  - Good if you know what you are after
- We have several [Solr books in the catalog](#) (some of them online)

Thank you for attending!

Stay in touch

[hector\\_correa@princeton.edu](mailto:hector_correa@princeton.edu)

<https://hectorcorrea.com/solr-for-newbies>