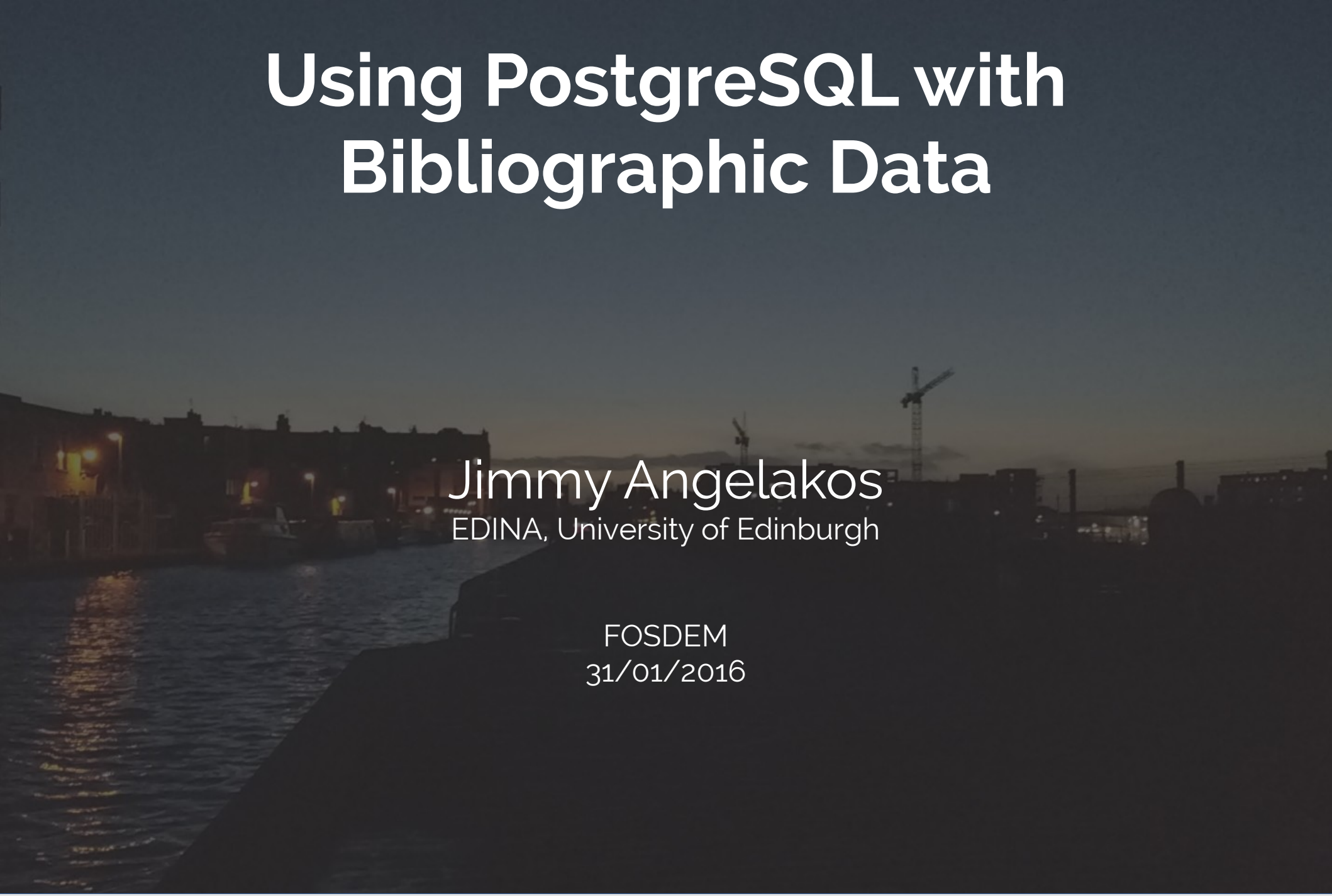
Using PostgreSQL with Bibliographic Data



Jimmy Angelakos
EDINA, University of Edinburgh

FOSDEM
31/01/2016

EDINA

Bibliography?

- The study of books
- Part of Library and Information Science
- Enumerative
  - Cataloguing books and topics
- Descriptive
  - Describing each item as an object

Bibliographic Data

- Bibliographic records
  - a.k.a. Metadata
  - Author / Title / Subject term / Keywords
  - Used to be compiled in indexes (or indices) using physical cards
  - Now found in Bibliographic Databases
- Machine-readable records started in the '60s
  - Library of Congress – MARC

MARC

- MAchine-Readable Cataloging
  - Predates the Web by decades
  - MARC21 (1999)
- Arcane format
  - Control & Data fields
  - Numeric codes
  - Indicators & Subfields
  - Whitespace significance

```
LEADER 00000cas a2200877   4500
001 422
003 WlSwUW
005 20150315000001.0
007 t|
008 960924c18879999iluqr p       0   a0eng c
010   $a   05035765
016 7 $a0370513$2DNLM
016 7 $aA24810000$2DNLM
022 0 $a0002-9556$10002-9556$21
030   $aAJPCAA
032   $a020360$bUSPS
035   $a(OCoLC)1408768
035   $a(SFX)954925377908
035   $a(SFX)   05035765
035   $a422
037   $bUniversity of Illinois Press, 54 E. Gregory Dr., Champaign, IL 61820
042   $ansdp$apcc
049   $aSC00320053010
050 00$aBF1$b.A5
060 00$aW1$bAM517
082 04$a150.5
090   $a954925377908
210   $aAM J PSYCHOL
210   $aAMERICAN JOURNAL OF PSYCHOLOGY
210   $aAMERICAN JOURNAL OF PSYCHOLOGY, THE
```

MARCXML

- An attempt to turn the chaos into order
- Nearly as bad as MARC21 itself
- Even commercial software doesn't produce valid output sometimes
- BRACE...

```
<collection xmlns="http://www.loc.gov/MARC21/slim"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.loc.gov/MARC21/slim
http://www.loc.gov/standards/marcxml/schema/MARC21slim.xsd" id="SC00320053010"
count="76" libcount="55">

  <record xmlns="http://www.loc.gov/MARC21/slim">
    <leader>     cas a2200877   4500</leader>
    <controlfield tag="001">422</controlfield>
    <controlfield tag="003">WlSwUW</controlfield>
    <controlfield tag="005">20150315000001.0</controlfield>
    <controlfield tag="007">t|</controlfield>
    <controlfield tag="008">960924c18879999iluqr p    0 a0eng c </controlfield>
    <datafield ind1=" " ind2=" " tag="010">
      <subfield code="a">   05035765</subfield>
    </datafield>
    <datafield ind1="7" ind2=" " tag="016">
      <subfield code="a">0370513</subfield>
      <subfield code="2">DNLM</subfield>
    </datafield>
    <datafield ind1="7" ind2=" " tag="016">
      <subfield code="a">A24810000</subfield>
      <subfield code="2">DNLM</subfield>
    </datafield>
    <datafield ind1="0" ind2=" " tag="022">
```

Serials?

- "Materials issued in any medium under the same title in a succession of discrete parts"
- Academic journals, periodicals, newspapers, newsletters, magazines
- Electronic & print serials
- ISSN (like ISBN is for books)

EDINA and SUNCAT

- EDINA National Data Centre (1996)
  - University of Edinburgh Data Library (1983)
- SALSER (1994) : Scottish Academic Libraries SERials
- SUNCAT (2003) : Serials UNion CATalogue
  - 103 UK research libraries, National Libraries of Scotland and Wales, Universities

Legacy software

- SUNCAT depends on:
  - Ex Libris ALEPH 500
    - Requires Oracle!
    - Not even valid MARCXML!
  - Index Data Zebra
  - Z39.50 protocol
    - Pre-web technology, text-based TELNET protocol
    - Not going anywhere anytime soon
  - SRU (REST) /SRW (SOAP) possible replacements

PostgreSQL Features

- JSONB data type
  - XML and JSON can become interchangeable
- GIN/GiST indexing
  - Rapid operations (XML, JSON access no longer a full-text search)

MARC/JSON

- Why not? It's structured data
- JSON is functionally equivalent to XML
- Lossless so we can reconstruct MARC21
- Only reference: An obscure 2010 blog post
  - Did not take into account field ordering
  - Did not take into account duplicate tags
- So...

```
{
"001": "422",
"003": "WlSwUW",
"005": "20150315000001.0",
"007": "t|", "008": "960924c18879999iluqr p       0   a0eng c",
"010": [{"ind1": " ", "ind2": " ", "sub": [{"a": "   05035765"}]}],
"016": [{"ind1": "7", "ind2": " ", "sub": [{"a": "0370513"}, {"2": "DNLM"}]},
        {"ind1": "7", "ind2": " ", "sub": [{"a": "A24810000"}, {"2": "DNLM"}]}],
"022": [{"ind1": "0", "ind2": " ", "sub": [{"a": "0002-9556"}, {"l": "0002-9556"},
"030": [{"ind1": " ", "ind2": " ", "sub": [{"a": "AJPCAA"}]}],
"032": [{"ind1": " ", "ind2": " ", "sub": [{"a": "020360"}, {"b": "USPS"}]}],
"035": [{"ind1": " ", "ind2": " ", "sub": [{"a": "(OCoLC)1408768"}]},
        {"ind1": " ", "ind2": " ", "sub": [{"a": "(SFX)954925377908"}]},
        {"ind1": " ", "ind2": " ", "sub": [{"a": "(SFX)   05035765"}]},
        {"ind1": " ", "ind2": " ", "sub": [{"a": "422"}]}],
"037": [{"ind1": " ", "ind2": " ", "sub": [{"b": "University of Illinois Press, 54
"042": [{"ind1": " ", "ind2": " ", "sub": [{"a": "nsdp"}, {"a": "pcc"}]}],
"049": [{"ind1": " ", "ind2": " ", "sub": [{"a": "SC00320053010"}]}],
"050": [{"ind1": "0", "ind2": "0", "sub": [{"a": "BF1"}, {"b": ".A5"}]}],
"060": [{"ind1": "0", "ind2": "0", "sub": [{"a": "W1"}, {"b": "AM517"}]}],
"082": [{"ind1": "0", "ind2": "4", "sub": [{"a": "150.5"}]}],
"090": [{"ind1": " ", "ind2": " ", "sub": [{"a": "954925377908"}]}],
"210": [{"ind1": " ", "ind2": " ", "sub": [{"a": "AM J PSYCHOL"}]},
        {"ind1": " ", "ind2": " ", "sub": [{"a": "AMERICAN JOURNAL OF PSYCHOLOGY"}
        {"ind1": " ", "ind2": " ", "sub": [{"a": "AMERICAN JOURNAL OF PSYCHOLOGY,
```

Python & Psycopg

- Psycopg
  - Python PostgreSQL adapter
  - Is a C extension (CPython)
  - Gives you the speed of libpq
  - Very expressive combination with Python
  - No time wasted converting between object types
  - Few lines of code needed for DB operations
  - Very readable code

Large XML data files

- SUNCAT data dump ~24GB
- DOM not practical
  - Requires loading the whole datafile in RAM
  - We don't require random access to elements

XML non-DOM parsing

- Streaming XML / Expat
  - a.k.a. Event-based a.k.a. SAX
- xmltodict
  - Like working with JSON
    - Python Dicts are very similar
  - Fast, Expat-based
  - Streaming mode (small memory footprint)
    - We can work with this huge file on a simple desktop

```
vyruss@zuul:~/devel/marc$ sudo apt-get install python-virtualenv python-dev
libpq-dev
(...)
vyruss@zuul:~/devel/marc$
```

```
vyruss@zuul:~/devel/marc$ sudo apt-get install python-virtualenv python-dev
libpq-dev
(...)
vyruss@zuul:~/devel/marc$ virtualenv ve
New python executable in ve/bin/python
Installing setuptools, pip...done.
vyruss@zuul:~/devel/marc$
```

```
vyruss@zuul:~/devel/marc$ sudo apt-get install python-virtualenv python-dev
libpq-dev
(...)
vyruss@zuul:~/devel/marc$ virtualenv ve
New python executable in ve/bin/python
Installing setuptools, pip...done.
vyruss@zuul:~/devel/marc$ . ve/bin/activate
(ve)vyruss@zuul:~/devel/marc$
```

```
vyruss@zuul:~/devel/marc$ sudo apt-get install python-virtualenv python-dev
libpq-dev
(...)
vyruss@zuul:~/devel/marc$ virtualenv ve
New python executable in ve/bin/python
Installing setuptools, pip...done.
vyruss@zuul:~/devel/marc$ . ve/bin/activate
(ve)vyruss@zuul:~/devel/marc$ pip install --upgrade pip
Downloading/unpacking pip from
https://pypi.python.org/packages/py2.py3/p/pip/pip-8.0.2-py2.py3-none-
any.whl#md5=2056f553d5b593d3a970296f229c1b79
  Downloading pip-8.0.2-py2.py3-none-any.whl (1.2MB): 1.2MB downloaded
Installing collected packages: pip
  Found existing installation: pip 1.5.4
    Uninstalling pip:
      Successfully uninstalled pip
Successfully installed pip
Cleaning up...
(ve)vyruss@zuul:~/devel/marc$
```

```
vyruss@zuul:~/devel/marc$ sudo apt-get install python-virtualenv python-dev
libpq-dev
(...)
vyruss@zuul:~/devel/marc$ virtualenv ve
New python executable in ve/bin/python
Installing setuptools, pip...done.
vyruss@zuul:~/devel/marc$ . ve/bin/activate
(ve)vyruss@zuul:~/devel/marc$ pip install --upgrade pip
Downloading/unpacking pip from
https://pypi.python.org/packages/py2.py3/p/pip/pip-8.0.2-py2.py3-none-
any.whl#md5=2056f553d5b593d3a970296f229c1b79
  Downloading pip-8.0.2-py2.py3-none-any.whl (1.2MB): 1.2MB downloaded
Installing collected packages: pip
  Found existing installation: pip 1.5.4
    Uninstalling pip:
      Successfully uninstalled pip
Successfully installed pip
Cleaning up...
(ve)vyruss@zuul:~/devel/marc$ pip install xmltodict
Collecting xmltodict
  Using cached xmltodict-0.9.2.tar.gz
Installing collected packages: xmltodict
  Running setup.py install for xmltodict ... done
Successfully installed xmltodict-0.9.2
(ve)vyruss@zuul:~/devel/marc$
```

```
(ve)vyruss@zuul:~/devel/marc$ pip install psycopg2
Collecting psycopg2
/home/vyruss/devel/marc/ve/local/lib/python2.7/site-
packages/pip/_vendor/requests/packages/urllib3/util/ssl_.py:315:
SNIMissingWarning: An HTTPS request has been made, but the SNI (Subject Name
Indication) extension to TLS is not available on this platform. This may
cause the server to present an incorrect TLS certificate, which can cause
validation failures. For more information, see
https://urllib3.readthedocs.org/en/latest/security.html#snimissingwarning.
  SNIMissingWarning
/home/vyruss/devel/marc/ve/local/lib/python2.7/site-
packages/pip/_vendor/requests/packages/urllib3/util/ssl_.py:120:
InsecurePlatformWarning: A true SSLContext object is not available. This
prevents urllib3 from configuring SSL appropriately and may cause certain SSL
connections to fail. For more information, see
https://urllib3.readthedocs.org/en/latest/security.html#insecureplatformwarni
ng.
  InsecurePlatformWarning
  Using cached psycopg2-2.6.1.tar.gz
Installing collected packages: psycopg2
  Running setup.py install for psycopg2 ... done
Successfully installed psycopg2-2.6.1
(ve)vyruss@zuul:~/devel/marc$
```

Why convert to JSON?

- We don't need an intermediate JSON file
- We parse the XML data as Python Dicts and insert it straight into the database as JSON
- Many millions of records will take time
- We only care about throughput here to get this done as fast as possible

```
#!/usr/bin/env python
# Our imports and database connection
import sys
import io
import json
import xmltodict
import psycopg2
import psycopg2.extras
import datetime
import cStringIO  # Faster than simple Python string I/O

conn = psycopg2.connect(database='marc', user='marc')
conn.set_isolation_level(  # For faster INSERTs
    psycopg2.extensions.ISOLATION_LEVEL_READ_UNCOMMITTED)
cur = conn.cursor()
count = 0
buffr = cStringIO.StringIO()
itercount = 0
```

[...]

```
# Our XML event callback handler function
def handle(_, item):
    global conn, cur, count, itercount, buffr
    out = {}
    tag = {}
    subfields = []
    for i,j in item.items():  # Iterate over the XML stream
        if i == 'leader':
            out[i]=j
        elif i == 'controlfield':
            for k in j:
                if '#text' in k:
                    out[k['@tag']] = k['#text']
                else:
                    out[k['@tag']] = None
```

[...]

```
else:               # if i == 'datafield' :
    for k in j:  # Nested loop to iterate over subfields
        if type(k['subfield']) != list:
            l = k['subfield']
            if '#text' in l:
                subfields = [{l['@code']:l['#text']}]
            else:
                subfields = [{l['@code']:None}]
        else:
            for l in k['subfield']:
                if '#text' in l:
                    subfields.append({l['@code']:l['#text']})
                else:
                    subfields.append({l['@code']:None})
        tag['ind1'] = k['@ind1']
        tag['ind2'] = k['@ind2']
        tag['subfields'] = subfields
        if k['@tag'] in out.keys():
            out[k['@tag']].append(tag)
        else:
            out[k['@tag']] = [tag]
        subfields = []
        tag = {}                                                    [...]
```

```
# So let's output this dict we've created into JSON
tmp = json.dumps(out).replace('\\','\\\\')  # Watch out!
    buffr.write(tmp)
    buffr.write('\n')
    count += 1
    if count % 10000 == 0:
        count = 0
        buffr.seek(0)
        cur.copy_from(buffr,'suncat')  # Should be fast
        conn.commit()
        buffr = cStringIO.StringIO()
        itercount += 1
        timeb = datetime.datetime.now()
        print itercount * 10000,'records inserted in',
              timeb - timea
        # This is just an example loop, you would have to
        # iterate once more for the last < 10000 items
    return True
```

[...]

```
# Our main function to open and parse the XML file
def convert(xml_file):
    with open(xml_file, "rb") as f:  # Notice "rb" mode!
        xmltodict.parse(f, item_depth=3,
                        item_callback=handle,
                        process_namespaces=False,
                        xml_attribs=True,
                        strip_whitespace=False)
        return

if __name__ == "__main__":
   cur.execute('CREATE TABLE suncat (data JSONB)')
   timea = datetime.datetime.now()
   convert(sys.argv[1])  # Grab filename from cmd line arg
   cur.copy_from(buffr,'suncat')
   conn.commit()
   conn.close()
   timeb = datetime.datetime.now()
   print 'TOTAL time was:', timeb - timea
                                                    [...]
```

Now let's query this thing

- JSONB data type lets us access structured data
- However, we have JSONB arrays here which cannot be accessed directly with our operators

```
marc=# SELECT data -> '245' FROM suncat limit 3;
                                                           ?column?
--------------------------------------------------------------------------------
 [{"ind1": "1", "ind2": "0", "subfields": [{"a": "Rapport annuel."}]}]
 [{"ind1": "1", "ind2": "0", "subfields": [{"a": "Programme /"}, {"c": "St. Magn
 [{"ind1": " ", "ind2": "0", "subfields": [{"a": "Background Notes on Countries
(3 rows)

Time: 0.586 ms
```

IMMUTABLE function for indexing

- ERROR: functions in index expression must be marked IMMUTABLE
- Does not modify DB
- Guaranteed to return the same results for the same input forever

```
-- We want this to "concatenate" our JSONB Array contents

CREATE OR REPLACE FUNCTION json2arr(_json JSONB,
                                    _1lev TEXT,
                                    _2lev TEXT)
RETURNS TEXT IMMUTABLE
AS $$
SELECT ARRAY(
    SELECT elem -> 'a' FROM (
        SELECT jsonb_array_elements(e -> _2lev) elem FROM (
            SELECT jsonb_array_elements(_json -> _1lev) e
        ) f
    ) g
)::TEXT
$$ LANGUAGE SQL
```

Will it blend?

- Let's test the indexing function now

```
marc=# SELECT json2arr(data, '210', 'subfields') ||' '||
               json2arr(data, '222', 'subfields') ||' '||
               json2arr(data, '245', 'subfields') ||' '||
               json2arr(data, '246', 'subfields')
        FROM suncat
        LIMIT 3;
                                                                ?column?
-------------------------------------------------------------------------------------
 {} {} {"\"Rapport annuel.\""} {}
 {} {} {"\"Programme /\"",NULL} {"\"St. Magnus Festival programme\""}
 {"\"BACKGROUND NOTES ON COUNTRIES OF THE WORLD\""} {} {"\"Background Notes on C
(3 rows)

Time: 2.113 ms
```

- The titles (we have multiple "title" MARC tags)

```
marc=# SELECT to_tsvector('english',
marc(#                    json2arr(data, '210', 'subfields') ||' '||
marc(#                    json2arr(data, '222', 'subfields') ||' '||
marc(#                    json2arr(data, '245', 'subfields') ||' '||
marc(#                    json2arr(data, '246', 'subfields'))
marc-# FROM suncat
marc-# LIMIT 3;
                              to_tsvector
-----------------------------------------------------------------------
 'annuel':2 'rapport':1
 'festiv':5 'magnus':4 'null':2 'programm':1,6 'st':3
 'background':1,8 'bolivia':15 'countri':4,11 'note':2,9 'world':7,14
(3 rows)

Time: 2.289 ms
```

- The titles as **tsvector**

```
marc=# \x
Expanded display is on.
marc=# SELECT show_trgm(json2arr(data, '210','subfields') ||' '||
                        json2arr(data, '222','subfields') ||' '||
                        json2arr(data, '245','subfields') ||' '||
                        json2arr(data, '246','subfields'))
        FROM suncat
        LIMIT 3;
-[ RECORD 1 ]-----------------------------------------------------------
show_trgm | {"  a","  r"," an"," ra",ann,app,"el ",nnu,nue,ort,por,ppo,rap,"rt "
-[ RECORD 2 ]-----------------------------------------------------------
show_trgm | {"  f","  m","  n","  p","  s"," fe"," ma"," nu"," pr"," st",agn,"al
-[ RECORD 3 ]-----------------------------------------------------------
show_trgm | {"  b","  c","  n","  o","  t","  w"," ba"," bo"," co"," no"," of","

Time: 1.848 ms
```

- The titles as **trigrams**

postgresql.conf

- maintenance_work_mem = 2GB
  - Heavy CREATE INDEX operations needed
  - Because YOLO
  - Restart server after change

```
marc=# \timing
Timing is on.

CREATE INDEX suncat_tsvector_idx
ON suncat
USING gin (to_tsvector('english',
                       json2arr(data, '210','subfields') ||' '||
                       json2arr(data, '222','subfields') ||' '||
                       json2arr(data, '245','subfields') ||' '||
                       json2arr(data, '246','subfields')));

CREATE INDEX
Time: 357891.027 ms
```

- ~6 minutes to create tsvector GIN index

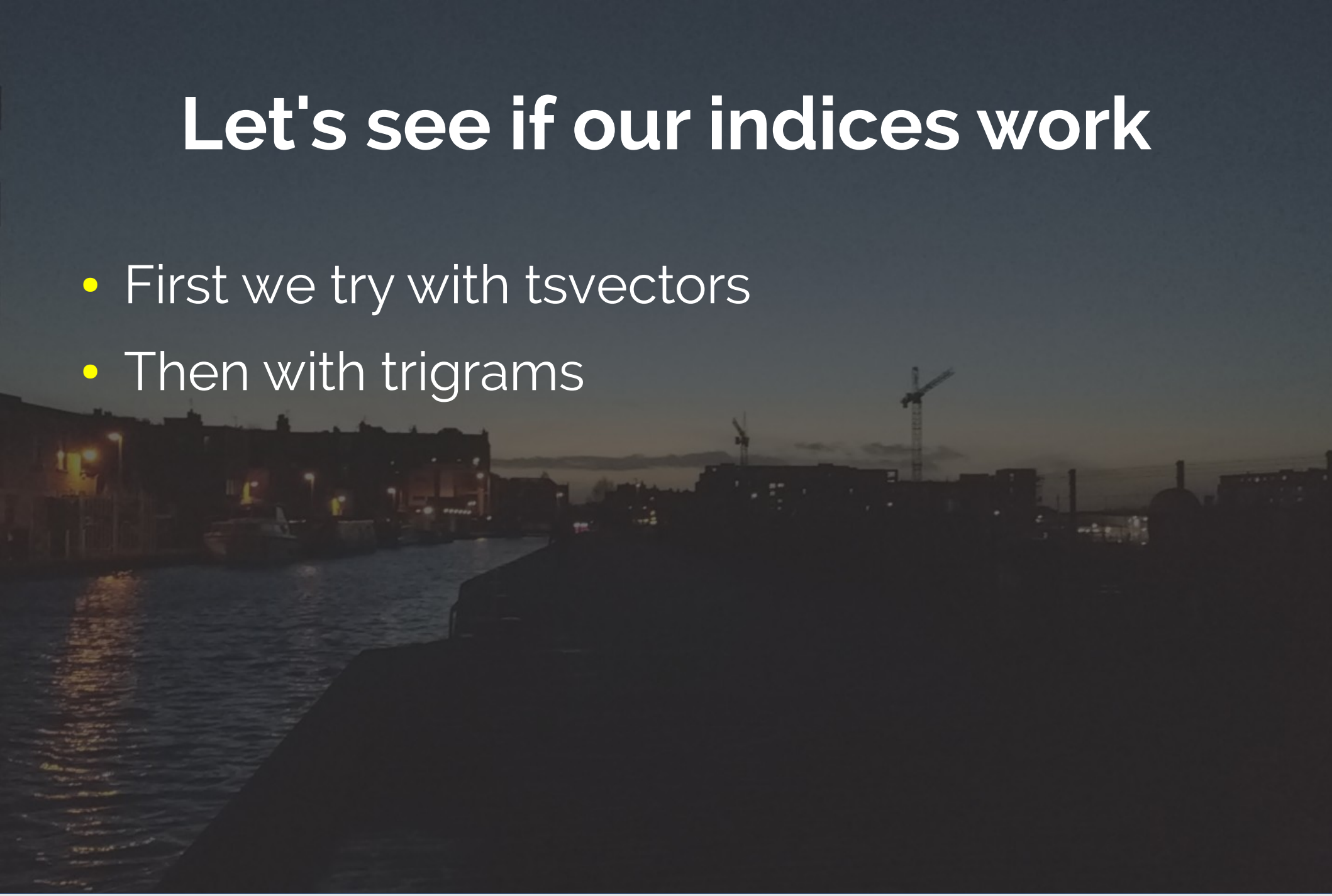
```
CREATE INDEX suncat_trigram_idx
ON suncat
USING gin (json2arr(data, '210','subfields') gin_trgm_ops,
           json2arr(data, '222','subfields') gin_trgm_ops,
           json2arr(data, '245','subfields') gin_trgm_ops,
           json2arr(data, '246','subfields') gin_trgm_ops);

CREATE INDEX
Time: 430615.077 ms
```

- ~7 minutes to create trigram GIN index

Let's see if our indices work

- First we try with tsvectors
- Then with trigrams



```
-- TSVECTOR QUERY (fast, requires correct spelling) ----------------------
SELECT DISTINCT "SUNCAT ID",
                "Rank",
                "Title"
FROM ( SELECT jsonb_array_elements(jsonb_array_elements(data->'049')->
                                   'subfields')->'a' AS "SUNCAT ID",
              jsonb_array_elements(jsonb_array_elements(data->'245')->
                                   'subfields')->'a' AS "Title",
              ts_rank(to_tsvector('english',
                                  json2arr(data, '210', 'subfields') ||' '||
                                  json2arr(data, '222', 'subfields') ||' '||
                                  json2arr(data, '245', 'subfields') ||' '||
                                  json2arr(data, '246', 'subfields')),
                      plainto_tsquery('english',
                                      'the american journal of psychology'),
                      32) AS "Rank"
       FROM suncat
       WHERE to_tsvector('english',
                         json2arr(data, '210', 'subfields') ||' '||
                         json2arr(data, '222', 'subfields') ||' '||
                         json2arr(data, '245', 'subfields') ||' '||
                         json2arr(data, '246', 'subfields'))
             @@ plainto_tsquery('english',
                                'the american journal of psychology')
     ) AS q
ORDER BY "Rank" DESC;
```

```
-[ RECORD 1 ]-----------------------------------------------------
SUNCAT ID | "SC00320053010"
Rank      | 0.491977
Title     |
-[ RECORD 2 ]-----------------------------------------------------
SUNCAT ID | "SC00320053010"
Rank      | 0.491977
Title     | "The American journal of psychology"
-[ RECORD 3 ]-----------------------------------------------------
SUNCAT ID | "SC00320053010"
Rank      | 0.491222
Title     | "The American journal of psychology"
-[ RECORD 4 ]-----------------------------------------------------
SUNCAT ID | "SC00320053010"
Rank      | 0.491222
Title     |
-[ RECORD 5 ]-----------------------------------------------------
SUNCAT ID | "SC00320053010"
Rank      | 0.48685
Title     |
```

```
Sort  (cost=343.40..345.90 rows=1000 width=68)
  Sort Key: q."Rank" DESC
  ->  HashAggregate  (cost=283.57..293.57 rows=1000 width=68)
        Group Key: q."Rank", q."SUNCAT ID", q."Title"
        ->  Subquery Scan on q  (cost=52.01..208.57 rows=10000 width=68)
              ->  Bitmap Heap Scan on suncat  (cost=52.01..108.57 rows=10000 width=1314)
                    Recheck Cond: (to_tsvector('english'::regconfig, ((((((json2arr(data, '210
                    ->  Bitmap Index Scan on suncat_tsvector_idx  (cost=0.00..52.01 rows=1 wid
                          Index Cond: (to_tsvector('english'::regconfig, ((((((json2arr(data,
(9 rows)
```

- EXPLAIN tsvector query

```
-- TRIGRAM QUERY (slower, some poor results, tolerates misspelling) --
SELECT DISTINCT "SUNCAT ID",
                "Similarity",
                "Title"
FROM ( SELECT jsonb_array_elements(jsonb_array_elements(data->'049')->
                                   'subfields')->'a' AS "SUNCAT ID",
              jsonb_array_elements(jsonb_array_elements(data->'245')->
                                   'subfields')->'a' AS "Title",
              similarity(json2arr(data, '245', 'subfields'),
                         'the american journal of psychology')
              AS "Similarity"
       FROM suncat
       WHERE json2arr(data, '245', 'subfields') %
             'the american journal of psychology'
       AND similarity(json2arr(data, '245', 'subfields'),
                      'the american journal of psychology') > 0.7
     ) AS q
ORDER BY "Similarity" DESC;
```

```
-[ RECORD 1 ]-------------------------------------------
SUNCAT ID  | "SC00313502803"
Similarity | 1
Title      | "The American journal of psychology."
-[ RECORD 2 ]-------------------------------------------
SUNCAT ID  | "SC00320053010"
Similarity | 1
Title      | "The American journal of psychology."
          [...]
-[ RECORD 4 ]-------------------------------------------
SUNCAT ID  | "SC00320053010"
Similarity | 0.885714
Title      | "AMERICAN journal of psychology."
          [...]
-[ RECORD 13 ]------------------------------------------
SUNCAT ID  | "SC00320053010"
Similarity | 0.875
Title      | "The American journal of psychology /"
          [...]
-[ RECORD 20 ]------------------------------------------
SUNCAT ID  | "SC00771206102"
Similarity | 0.789474
Title      | "The American psychological journal."
          [...]
-[ RECORD 30 ]------------------------------------------
SUNCAT ID  | "SC00590379406"
Similarity | 0.769231
Title      | "The American journal of pathology"
```

```
Unique  (cost=5392691.32..5599391.32 rows=2067000 width=68)
  ->  Sort  (cost=5392691.32..5444366.32 rows=20670000 width=68)
        Sort Key: q."Similarity" DESC, q."SUNCAT ID", q."Title"
        ->  Subquery Scan on q  (cost=647.02..337795.85 rows=20670000 width=68)
              ->  Bitmap Heap Scan on suncat  (cost=647.02..131095.85 rows=20670000 width=1314
                    Recheck Cond: (json2arr(data, '245'::text, 'subfields'::text) % 'the ameri
                    Filter: (similarity(json2arr(data, '245'::text, 'subfields'::text), 'the a
                    ->  Bitmap Index Scan on suncat_trigram_idx  (cost=0.00..646.50 rows=6200
                          Index Cond: (json2arr(data, '245'::text, 'subfields'::text) % 'the a
(9 rows)
```

- EXPLAIN trigram query

```
-- COMBINED QUERY (fast, tolerates misspelling) ------------------------------------
SELECT DISTINCT "SUNCAT ID",
                (sml + rank)/2 AS "Combined Score",
                "Title"
FROM ( SELECT jsonb_array_elements(jsonb_array_elements(data->'049')->'subfields')->'a'
              AS "SUNCAT ID",
              jsonb_array_elements(jsonb_array_elements(data->'245')->'subfields')->'a'
              AS "Title",
              ts_rank(to_tsvector('english',
                                  json2arr(data, '210', 'subfields') ||' '||
                                  json2arr(data, '222', 'subfields') ||' '||
                                  json2arr(data, '245', 'subfields') ||' '||
                                  json2arr(data, '246', 'subfields')),
                      plainto_tsquery('english',
                                      'the american journal of psychology'),
                                      32)
              AS rank,
              similarity(json2arr(data, '245', 'subfields'),
                         'the american journal of psychology')
              AS sml
       FROM suncat
       WHERE to_tsvector('english',
                         json2arr(data, '210', 'subfields') ||' '||
                         json2arr(data, '222', 'subfields') ||' '||
                         json2arr(data, '245', 'subfields') ||' '||
                         json2arr(data, '246', 'subfields'))
             @@ plainto_tsquery('english', 'the american journal of psychology')
       AND json2arr(data, '245', 'subfields') % 'the american journal of psychology'
       AND similarity(json2arr(data, '245', 'subfields'),
           'the american journal of psychology') > 0.7
     ) AS q
ORDER BY "Combined Score" DESC;
```

```
Sort  (cost=399.16..401.66 rows=1000 width=72)
  Sort Key: (((q.sml + q.rank) / '2'::double precision)) DESC
  ->  HashAggregate  (cost=334.33..349.33 rows=1000 width=72)
        Group Key: ((q.sml + q.rank) / '2'::double precision), q."SUNCAT ID", q."Title"
        ->  Subquery Scan on q  (cost=52.01..259.33 rows=10000 width=72)
              ->  Bitmap Heap Scan on suncat  (cost=52.01..109.33 rows=10000 width=1314)
                    Recheck Cond: (to_tsvector('english'::regconfig, ((((((json2arr(data, '210
                    Filter: ((json2arr(data, '245'::text, 'subfields'::text) % 'the american j
                    ->  Bitmap Index Scan on suncat_tsvector_idx  (cost=0.00..52.01 rows=1 wid
                          Index Cond: (to_tsvector('english'::regconfig, ((((((json2arr(data,
(10 rows)
```

- EXPLAIN combined query

```
-- MONSTER QUERY FUNCTION ! ------------------------------------------------------------------------------
CREATE OR REPLACE FUNCTION suncatquery (_terms TEXT)
RETURNS TABLE ("SUNCAT ID" TEXT, "Title 1" TEXT, "Title 2" TEXT, "Title 3" TEXT, "Title 4" TEXT,
               "Similarity Score" NUMERIC(4,2))
AS $$
    SELECT DISTINCT trim("SUNCAT ID", '"'),
                    trim("Title 1", '"'),
                    trim("Title 2", '"'),
                    trim("Title 3", '"'),
                    trim("Title 4", '"'),
                    ((sml + rank)/2)::NUMERIC(4,2) AS score
    FROM ( SELECT (jsonb_array_elements(jsonb_array_elements(data->'049')->'subfields')->'a')::TEXT AS "SUNCAT ID",
                  (jsonb_array_elements(jsonb_array_elements(data->'210')->'subfields')->'a')::TEXT AS "Title 1",
                  (jsonb_array_elements(jsonb_array_elements(data->'222')->'subfields')->'a')::TEXT AS "Title 2",
                  (jsonb_array_elements(jsonb_array_elements(data->'245')->'subfields')->'a')::TEXT AS "Title 3",
                  (jsonb_array_elements(jsonb_array_elements(data->'246')->'subfields')->'a')::TEXT AS "Title 4",
                  ts_rank(to_tsvector('english',
                                      json2arr(data, '210', 'subfields') ||' '||
                                      json2arr(data, '222', 'subfields') ||' '||
                                      json2arr(data, '245', 'subfields') ||' '||
                                      json2arr(data, '246', 'subfields')),
                          plainto_tsquery('english', _terms),
                          32) AS rank,
                  similarity(json2arr(data, '245', 'subfields'), _terms) AS sml
           FROM suncat
           WHERE to_tsvector('english',
                             json2arr(data, '210', 'subfields') ||' '||
                             json2arr(data, '222', 'subfields') ||' '||
                             json2arr(data, '245', 'subfields') ||' '||
                             json2arr(data, '246', 'subfields'))
                 @@ plainto_tsquery('english', _terms)
           AND json2arr(data, '245', 'subfields') % _terms
           AND similarity(json2arr(data, '245', 'subfields'), _terms) > 0.7
         ) AS q
    ORDER BY score DESC;
$$ LANGUAGE SQL;
```

```
marc=# select * from suncatquery('JOURNAL oF british Sociology');
-[ RECORD 1 ]----+----------------------------------------
SUNCAT ID        | SC00320059210
Title 1          | Br. j. sociol.
Title 2          | British journal of sociology
Title 3          | British journal of sociology.
Title 4          | BJS
Similarity Score | 0.70
-[ RECORD 2 ]----+----------------------------------------
SUNCAT ID        | SC00320059210
Title 1          |
Title 2          |
Title 3          | British journal of sociology.
Title 4          | BJS
Similarity Score | 0.70
-[ RECORD 3 ]----+----------------------------------------
SUNCAT ID        | SC00320059210
Title 1          | BR J SOCIOL
Title 2          | The British journal of sociology
Title 3          | The British journal of sociology.
Title 4          | BJS
Similarity Score | 0.67
-[ RECORD 4 ]----+----------------------------------------
SUNCAT ID        | SC00320059210
Title 1          | BRITISH JOURNAL OF SOCIOLOGY
Title 2          | The British journal of sociology
Title 3          | The British journal of sociology.
Title 4          | BJS
```


Thank you =)
Twitter: @vyruss
Stand by for EDINA developer blogs
around http://edina.ac.uk
– blog post to come