Working With A Real World Dataset In Neo4j

Modeling and Import

William Lyon
@lyonwj

William Lyon

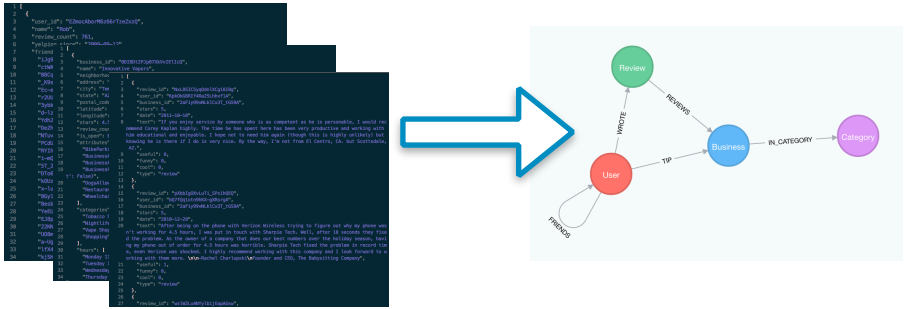
Developer Relations Engineer @neo4j

will@neo4j.com
@lyonwj
lyonwj.com

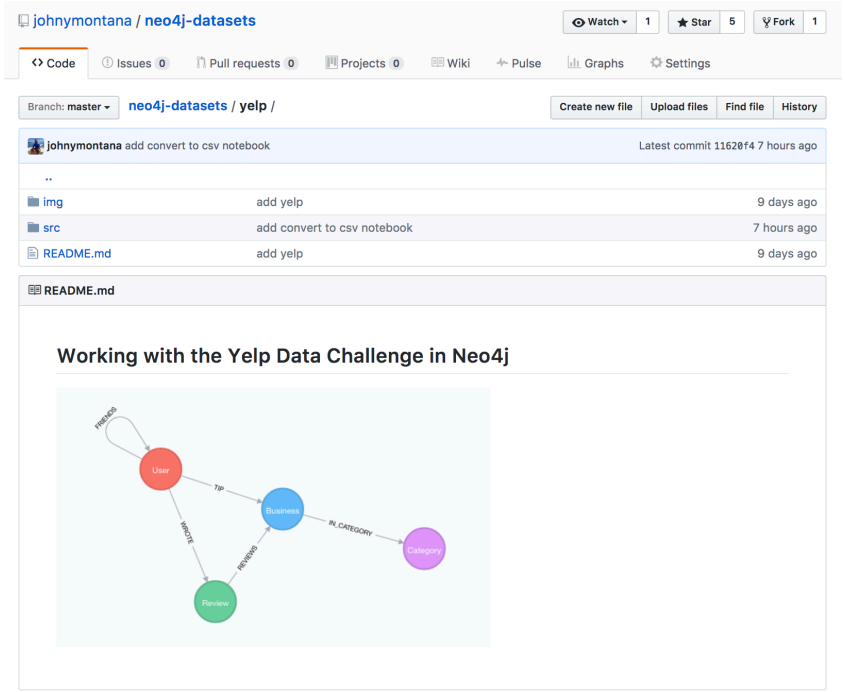
neo4j

Agenda

- The Data!
- Build a graph data model
- Import
  - LOAD CSV
  - Import Tool (`neo4j-import`)
- Neo4j drivers
  - Python



The Codez



https://github.com/johnymontana/neo4j-datasets/tree/master/yelp

The Data

neo4j

Yelp Dataset Challenge

Round 9 Of The Yelp Dataset Challenge: Our Largest Yet!
We've had 8 rounds, over $50,000 in cash prizes awarded, hundreds of academic papers written, and we are excited to see round 9.

Our dataset has been updated for this iteration of the challenge - we're sure there are plenty of interesting insights waiting there for you. This set includes information about local businesses in 11 cities across 4 countries.

This round also includes photos! These photos nicely complement reviews, business attributes, check-ins, and tips, and open the door to even more exciting research. An auxiliary file has been provided for download (see the "Get the Data" link on this page), containing 200,000 pictures from 85,901 businesses described in the main dataset. The photo archive includes a json file linking each photo to its corresponding business in the dataset, and listing its caption (if any), and type of content as determined by our image classifier (we currently only list labels for some restaurants).

This treasure trove of local business data is waiting to be mined and we can't wait to see you push the frontiers of data science research with our data.

The Challenge Dataset:

- **4.1M** reviews and **947K** tips by **1M** users for **144K** businesses
- **1.1M** business attributes, e.g., hours, parking availability, ambience.
- Aggregated check-ins over time for each of the **125K** businesses
- **200,000** pictures from the included businesses

Get the Data

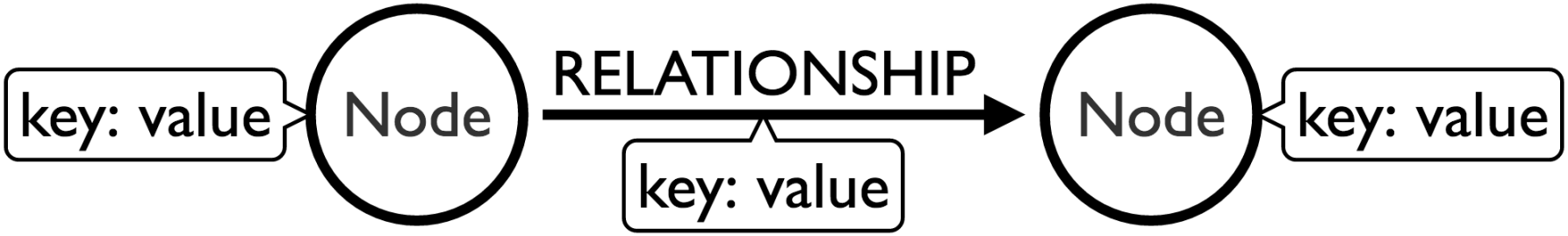
Cities:

- U.K.: Edinburgh
- Germany: Karlsruhe
- Canada: Montreal and Waterloo
- U.S.: Pittsburgh, Charlotte, Urbana-Champaign, Phoenix, Las Vegas, Madison, Cleveland

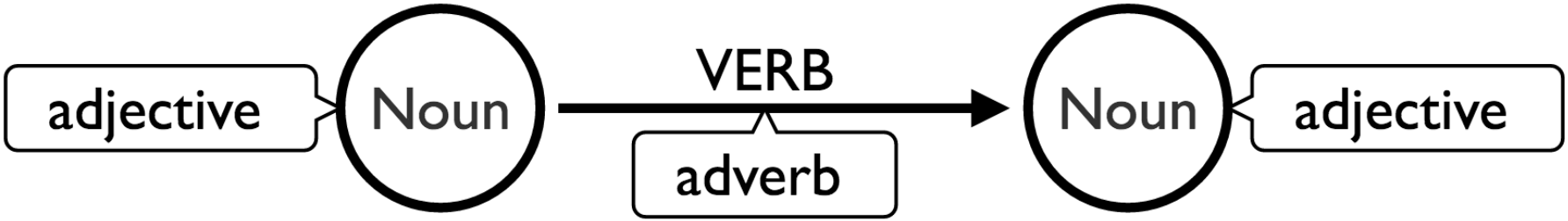
https://www.yelp.com/dataset_challenge

Graph Data Model

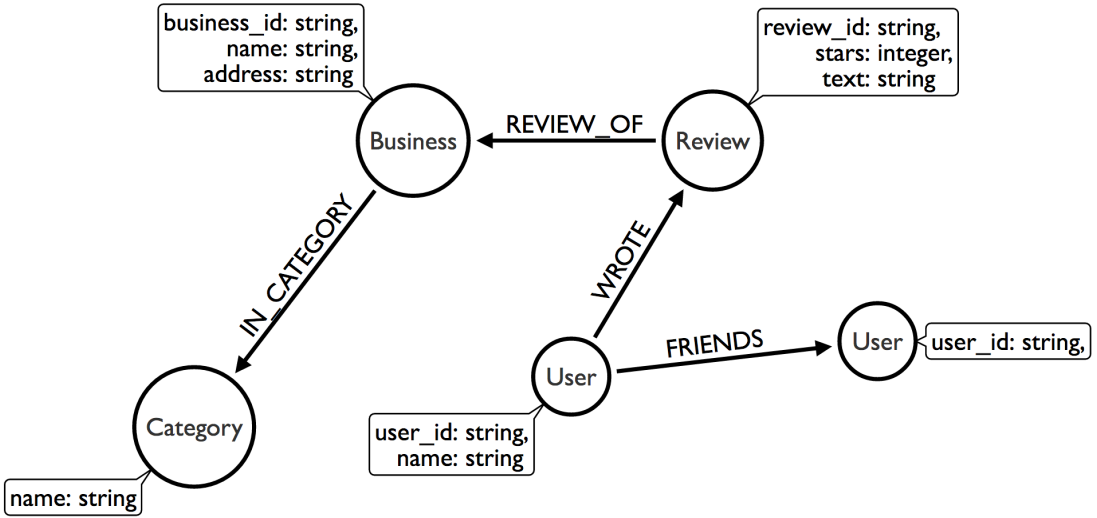
Labeled Property Graph Model



Labeled Property Graph Model



Our Data Model



- Identify “entities”
- What properties are relevant?
- Identify unique ids
- Find connections
- Repeat

http://www.apcjones.com/arrows/#

Import

apoc.load.json

Calling Procedures & Functions within Cypher

User defined **Functions** can be used in any expression or predicate, just like built-in functions.

Procedures can be called stand-alone with `CALL procedure.name();`

But you can also integrate them into your Cypher statements which makes them so much more powerful.

Load JSON example

CYPHER

```
WITH 'https://raw.githubusercontent.com/neo4j-contrib/neo4j-apoc-
procedures/master/src/test/resources/person.json' AS url

CALL apoc.load.json(url) YIELD value as person

MERGE (p:Person {name:person.name})
   ON CREATE SET p.age = person.age, p.children = size(person.children)
```

https://neo4j-contrib.github.io/neo4j-apoc-procedures/

Convert streaming JSON to JSON using jq


```
head -n 1000 yelp_academic_dataset_user.json | jq -s '.' >
user.json
```

https://stedolan.github.io/jq/

Convert streaming JSON to JSON using jq


```
# convert streaming json files to standard json
declare -a arr=("review" "checkin" "user" "tip" "business")

for i in "${arr[@]}"
do
    echo "$i"
    cat ../data/yelp_academic_dataset_$i.json | jq -s '.' > ../
data/$i.json
done
```

The Data

user.json

business.json

review.json

apoc.load.json

```
// Import user.json
CALL apoc.load.json("file:/Users/lyonwj/reviews/user.json") YIELD
value AS user
RETURN user LIMIT 5
```

apoc.load.json

```
// Import user.json
CALL apoc.load.json("file:/Users/lyonwj/reviews/user.json") YIELD
value AS user
MERGE (u:User {user_id: user.user_id})
SET u.name                = user.name,
    u.review_count        = user.review_count,
    u.average_stars       = user.average_stars,
    u.fans                = user.fans
```

```
// Import business.json
CALL apoc.load.json("file:/Users/lyonwj/reviews/business.json") YIELD value AS
business
MERGE (b:Business {business_id: business.business_id})
SET b.address = business.address,
              b.lat     = business.latitude,
              b.lon     = business.longitude,
              b.name    = business.name,
              b.city    = business.city,
              b.postal_code = business.postal_code,
              b.state = business.state,
              b.review_count = business.review_count,
              b.stars = business.stars,
              b.neighborhood = business.neighborhood
WITH b, business.categories AS categories
UNWIND categories AS cat
MERGE (c:Category {name: cat})
MERGE (b)-[:IN_CATEGORY]->(c)
```

```
/// Import review.json
CALL apoc.load.json("file:/Users/lyonwj/reviews/review.json")
YIELD value AS review
MERGE (b:Business {business_id: review.business_id})
MERGE (u:User {user_id: review.user_id})
MERGE (r:Review {review_id: review.review_id})
ON CREATE SET r.text   = review.text,
              r.type   = review.type,
              r.date   = review.date,
              r.stars  = review.stars,
              r.useful = review.useful
MERGE (u)-[:WROTE]->(r)
MERGE (r)-[:REVIEWS]->(b)
```

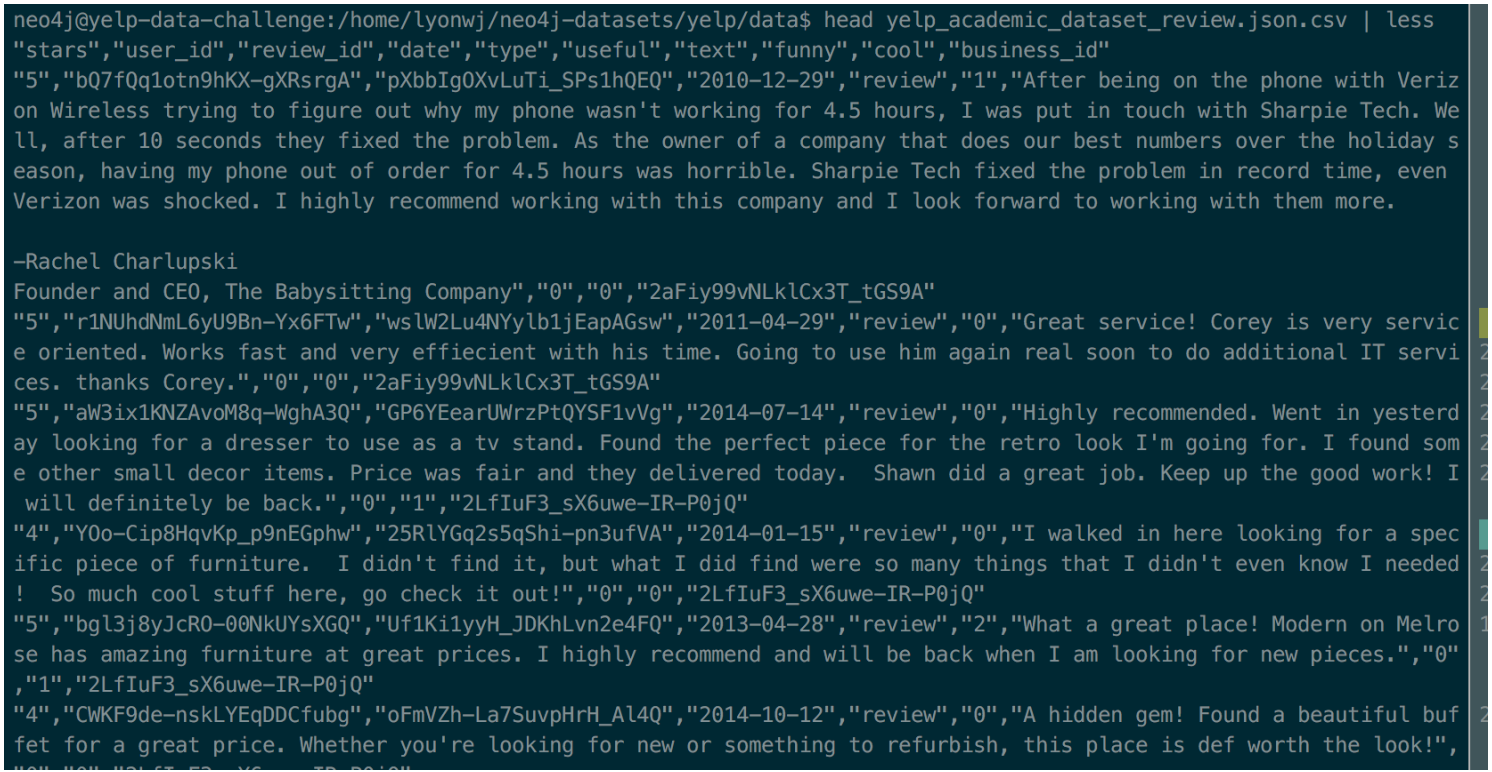
Import

LOAD CSV

```
# Convert Yelp Data Challenge streaming JSON files into CSV

import json
import csv

YELP_REVIEW_FILE = "../data/yelp_academic_dataset_review.json"
YELP_TIP_FILE = "../data/yelp_academic_dataset_tip.json"
YELP_USER_FILE = "../data/yelp_academic_dataset_user.json"
YELP_CHECKIN_FILE = "../data/yelp_academic_dataset_checkin.json"
YELP_BUSINESS_FILE = "../data/yelp_academic_dataset_business.json"

files = [YELP_REVIEW_FILE, YELP_TIP_FILE, YELP_USER_FILE, YELP_CHECKIN_FILE, YELP_BUSINESS_FILE]

with open(YELP_REVIEW_FILE, "r") as file:
    with open(YELP_REVIEW_FILE + '.csv', 'w') as csvfile:
        writer = csv.writer(csvfile, escapechar='\\', quotechar='"', quoting=csv.QUOTE_ALL)
        writer.writerow(json.loads(file.readline()).keys())
        for line in file:
            l = []
            item = json.loads(line)
            for k,i in item.items():
                # Represent a list of items as a semicolon delimitted string
                if type(i) == list:
                    l.append(';'.join(i))
                # Aggressive quoting and escape char handling
                if type(i) == str:
                    l.append(i.replace('"', '').replace('\\', ''))
                else:
                    l.append(i)
            writer.writerow(l)
```

https://github.com/johnymontana/neo4j-datasets/blob/master/yelp/src/Yelp_convert_csv.ipynb

What do the csv files look like?



```
neo4j@yelp-data-challenge:/home/lyonwj/neo4j-datasets/yelp/data$ head yelp_academic_dataset_review.json.csv | less
"stars","user_id","review_id","date","type","useful","text","funny","cool","business_id"
"5","bQ7fQq1otn9hKX-gXRsrgA","pXbbIgOXvLuTi_SPs1hQEQ","2010-12-29","review","1","After being on the phone with Veriz
on Wireless trying to figure out why my phone wasn't working for 4.5 hours, I was put in touch with Sharpie Tech. We
ll, after 10 seconds they fixed the problem. As the owner of a company that does our best numbers over the holiday s
eason, having my phone out of order for 4.5 hours was horrible. Sharpie Tech fixed the problem in record time, even
Verizon was shocked. I highly recommend working with this company and I look forward to working with them more.

-Rachel Charlupski
Founder and CEO, The Babysitting Company","0","0","2aFiy99vNLklCx3T_tGS9A"
"5","r1NUhdNmL6yU9Bn-Yx6FTw","wslW2Lu4NYylb1jEapAGsw","2011-04-29","review","0","Great service! Corey is very servic
e oriented. Works fast and very effiecient with his time. Going to use him again real soon to do additional IT servi
ces. thanks Corey.","0","0","2aFiy99vNLklCx3T_tGS9A"
"5","aW3ix1KNZAvoM8q-WghA3Q","GP6YEearUWrzPtQYSF1vVg","2014-07-14","review","0","Highly recommended. Went in yesterd
ay looking for a dresser to use as a tv stand. Found the perfect piece for the retro look I'm going for. I found som
e other small decor items. Price was fair and they delivered today.  Shawn did a great job. Keep up the good work! I
 will definitely be back.","0","1","2LfIuF3_sX6uwe-IR-P0jQ"
"4","YOo-Cip8HqvKp_p9nEGphw","25RlYGq2s5qShi-pn3ufVA","2014-01-15","review","0","I walked in here looking for a spec
ific piece of furniture.  I didn't find it, but what I did find were so many things that I didn't even know I needed
!  So much cool stuff here, go check it out!","0","0","2LfIuF3_sX6uwe-IR-P0jQ"
"5","bgl3j8yJcRO-00NkUYsXGQ","Uf1Ki1yyH_JDKhLvn2e4FQ","2013-04-28","review","2","What a great place! Modern on Melro
se has amazing furniture at great prices. I highly recommend and will be back when I am looking for new pieces.","0"
,"1","2LfIuF3_sX6uwe-IR-P0jQ"
"4","CWKF9de-nskLYEqDDCfubg","oFmVZh-La7SuvpHrH_Al4Q","2014-10-12","review","0","A hidden gem! Found a beautiful buf
fet for a great price. Whether you're looking for new or something to refurbish, this place is def worth the look!",
```

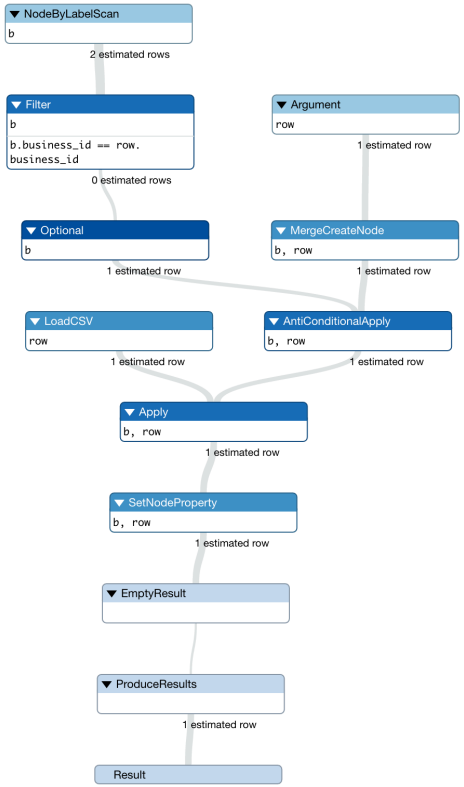
Naive Import

```
LOAD CSV WITH HEADERS FROM "file:///reviews.csv" AS row
MERGE (b:Business {business_id: row.business_id})
MERGE (u:User {user_id: row.user_id})
MERGE (r:Review {review_id: row.review_id})
  ON CREATE SET r.stars = toInteger(row.stars),
                r.text  = row.text
MERGE (r)-[:REVIEW_OF]->(b)
MERGE (u)-[rr:WROTE]-(r)
  ON CREATE SET rr.date = row.date
```

Break Up MERGEs

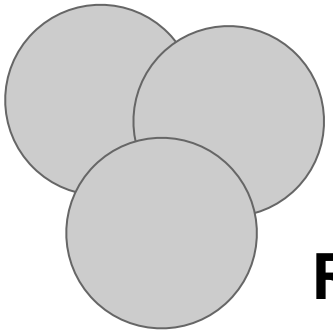

```
LOAD CSV WITH HEADERS FROM “file:///business.csv" AS row
MERGE (b:Business {business_id: row.business_id})
ON CREATE SET b.name = row.name …

LOAD CSV WITH HEADERS FROM “file:///user.csv” AS row
MERGE (b:User {user_id: row.user_id})
ON CREATE SET b.name = row.name …

LOAD CSV WITH HEADERS FROM “file:///review.csv" AS row
MATCH (u:User {user_id: row.user_id})
MATCH (b:Business {business_id: row.business_id})
CREATE (r:Review {review_id: row.review_id})
SET r.stars = row.stars, b.text = row.text
CREATE (u)-[:WROTE]->(r)
CREATE (r)-[:REVIEW_OF]->(b)
```

EXPLAIN


NodeByLabelScan
b
2 estimated rows
Filter
b
b.business_id == row.business_id
0 estimated rows
Argument
row
1 estimated row
Optional
b
1 estimated row
MergeCreateNode
b, row
1 estimated row
LoadCSV
row
1 estimated row
AntiConditionalApply
b, row
1 estimated row
Apply
b, row
1 estimated row
SetNodeProperty
b, row
1 estimated row
EmptyResult
ProduceResults
1 estimated row
Result

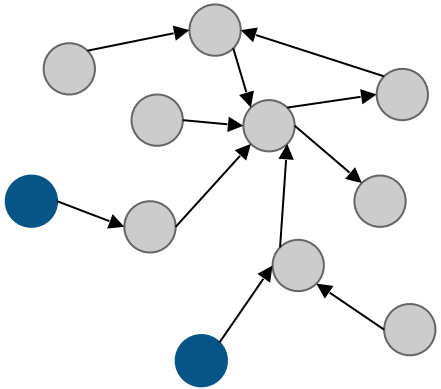
How does Neo4j use indexes?

Indexes are **only** used to find the starting point for queries.



Relational

Use index scans to look up rows in tables and join them with rows from other tables



Graph

Use indexes to find the starting points for a query.

Create Index / Constraint

Constraint + index

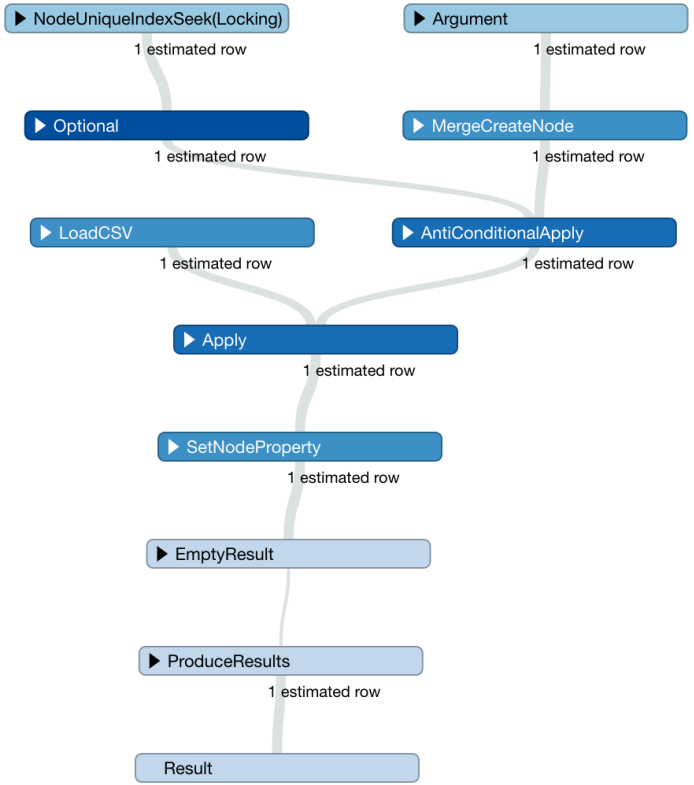
```
CREATE CONSTRAINT ON (u:User) ASSERT u.user_id IS UNIQUE;
CREATE CONSTRAINT ON (b:Business) ASSERT b.business_id IS UNIQUE;
```

Index

```
CREATE INDEX ON :Business(name);
```

http://neo4j.com/docs/developer-manual/current/cypher/schema/constraints/

EXPLAIN


NodeUniqueIndexSeek(Locking)
1 estimated row
Argument
1 estimated row
Optional
1 estimated row
MergeCreateNode
1 estimated row
LoadCSV
1 estimated row
AntiConditionalApply
1 estimated row
Apply
1 estimated row
SetNodeProperty
1 estimated row
EmptyResult
ProduceResults
1 estimated row
Result

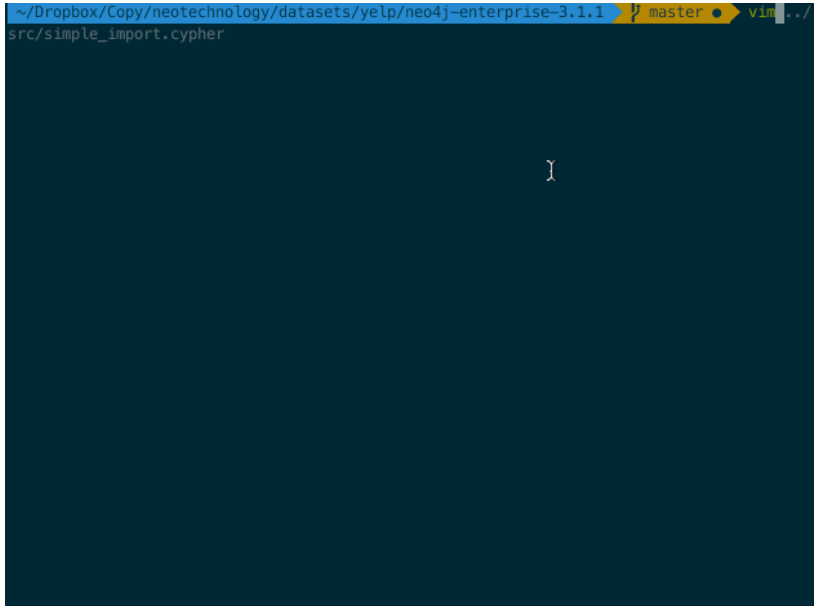
PERIODIC COMMIT

```
USING PERIODIC COMMIT 20000
LOAD CSV WITH HEADERS FROM “file:///review.csv" AS row
MATCH (u:User {user_id: row.user_id})
MATCH (b:Business {business_id: row.business_id})
CREATE (r:Review {review_id: row.review_id})
SET r.stars = row.stars, b.text = row.text
CREATE (u)-[:WROTE]->(r)
CREATE (r)-[:REVIEW_OF]->(b)
```

https://neo4j.com/docs/developer-manual/current/cypher/clauses/using-periodic-commit/

cypher-shell Run multi-line Cypher scripts

```
cat ../src/simple_import.cypher | bin/cypher-shell
```



Replaces `neo4j-shell` in Neo4j 3.x+

https://neo4j.com/docs/operations-manual/current/tools/cypher-shell/

Parallel Inserts w/ apoc.periodic.iterate


```
// Periodic iterate - LOAD CSV
WITH 'LOAD CSV WITH HEADERS FROM "file:///
yelp_academic_dataset_business.json.csv" AS row RETURN row' AS
load_csv
CALL apoc.periodic.iterate(load_csv, 'WITH {row} CREATE
(b:Business) SET b.business_id = row.business_id, b.name =
row.name', {batchSize: 50000, parallel: True, iterateList: True,
retries:3}) YIELD batches, total
RETURN *
```

Parallel Inserts w/ apoc.periodic.iterate


```
// Periodic iterate - LOAD CSV
WITH 'LOAD CSV WITH HEADERS FROM "file:///
yelp_academic_dataset_business.json.csv" AS row RETURN row' AS
load_csv
CALL apoc.periodic.iterate(load_csv, 'WITH {row} CREATE
(b:Business) SET b.business_id = row.business_id, b.name =
row.name', {batchSize: 50000, parallel: True, iterateList: True,
retries:3}) YIELD batches, total
RETURN *
```

batches	load_csv	total
15	LOAD CSV WITH HEADERS FROM "file:///yelp_academic_dataset_business.json.csv" AS row RETURN row	144071

Import

Neo4j Import Tool

neo4j

MAX DE MARZI

Graphs with Neo4j

VIDEOS SERVICES CONTACT ABOUT HOME

OCT 16 2015

LEAVE A COMMENT

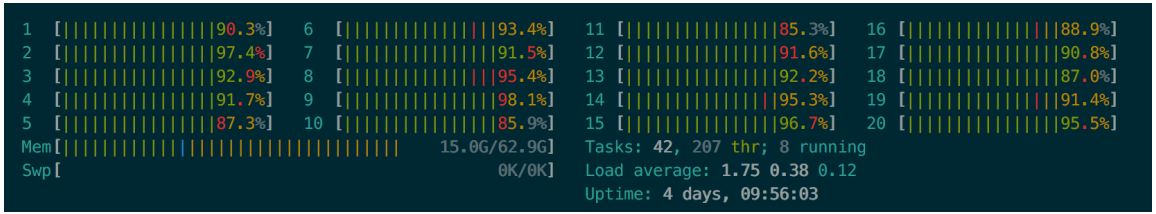
JAVA, PROBLEMS, RANDOM, TESTING

BENCHMARKS AND SUPERCHARGERS

https://maxdemarzi.com

Neo4j Import Tool (neo4j-import)

- Command line tool
- Initial import only
- Creates *foo.db*
- Specific CSV file format



http://neo4j.com/docs/operations-manual/current/tutorial/import-tool/


name: "Bob"
User
WROTE
Review
stars: 5

neo4j-import file format

Nodes

:ID(User)	:LABEL	name
123	User	Will
124	User	Bob
125	User	Heather
126	User	Erika

:ID(Review)	:LABEL	stars:int
127	Review	3
128	Review	2
129	Review	5
130	Review	1

Relationships

:START_ID(User)	:END_ID(Review)	:TYPE
123	127	WROTE
124	128	WROTE
125	129	WROTE
126	130	WROTE

In [81]:

```
def convertForNeo4jImport():
    # Create User csv file and FRIENDS relationship csv
    count = 0
    with open(YELP_USER_FILE, 'r') as file:
        # open user node file
        with open('../data/import/user.csv', 'w') as csvfile:
            writer = csv.writer(csvfile, escapechar='\\', quotechar='"', quoting=csv.QUOTE_ALL)
            writer.writerow([':ID(User)', ':LABEL', 'name'])
            with open('../data/import/friends.csv', 'w') as friendscsv:
                friendwriter = csv.writer(friendscsv, escapechar='\\', quotechar='"', quoting=csv.QUOTE_ALL)
                friendwriter.writerow([":START_ID(User)", ":END_ID(User)", ":TYPE"])
                for line in file:
                    item = json.loads(line)
                    writer.writerow([item['user_id'], "User", item['name']])
                    for friend in item['friends']:
                        friendwriter.writerow([item['user_id'], friend, 'FRIENDS'])

    # Create Business csv file
    with open(YELP_BUSINESS_FILE, 'r') as file:
        #count = 0
        with open('../data/import/business.csv', 'w') as csvfile:
            writer = csv.writer(csvfile, escapechar='\\', quotechar='"', quoting=csv.QUOTE_ALL)
            writer.writerow([':ID(Business)', ':LABEL', 'name'])
            for line in file:
                item = json.loads(line)
                try:
                    writer.writerow([item['business_id'], 'Business', item['name']])
                except Exception as e:
                    print(item)
                    throw(e)


    # Create Review csv file, WROTE relationship csv,
    with open(YELP_REVIEW_FILE, 'r') as file:
        with open('../data/import/review.csv', 'w') as csvfile:
            writer = csv.writer(csvfile, escapechar="\\", quotechar='"', quoting=csv.QUOTE_ALL)
            writer.writerow([':ID(Review)', ':LABEL', 'stars:int', 'text'])

            with open('../data/import/wrote.csv', 'w') as wrotefile:
                wrotewriter = csv.writer(wrotefile, escapechar="\\", quotechar='"', quoting=csv.QUOTE_ALL)
                wrotewriter.writerow([':START_ID(User)', ':END_ID(Review)', ':TYPE'])

                with open('../data/import/review_of.csv', 'w') as review_of_file:
                    reviewwriter = csv.writer(review_of_file,  escapechar="\\", quotechar='"', quoting=csv.QUOTE_ALL)
                    reviewwriter.writerow([':START_ID(Review)', ':END_ID(Business)', ':TYPE'])

                    for line in file:
                        item = json.loads(line)
                        # (:Review {review_id, stars, text})
                        writer.writerow([item['review_id'], 'Review', item['stars'], item['text'].replace('\n', ' ').re

                        # (User)-[:WROTE]->(:Review)
                        wrotewriter.writerow([item['user_id'], item['review_id'], 'WROTE'])

                        # (:Review)-[:REVIEW_OF]->(:Business)
                        reviewwriter.writerow([item['review_id'], item['business_id'], 'REVIEW_OF'])
```

https://github.com/johnymontana/neo4j-datasets/blob/master/yelp/src/Yelp_convert_csv.ipynb

neo4j-import

```
neo4j-import --into /var/lib/neo4j/data/databases/yelp.db
--nodes user.csv
--nodes business.csv
--nodes review.csv
--relationships friends.csv
--relationships wrote.csv
--relationships review_of.csv
```

Now included in `neo4j-admin import`

neo4j-import

```
neo4j@yelp-data-challenge:/home/lyonwj/neo4j-datasets/yelp/data/import$ neo4j-import --into /var/lib/neo
4j/data/databases/yelp.db --nodes user.csv --nodes business.csv --nodes review.csv  --relationships wrot
e.csv --relationships review_of.csv --multiline-fields=true --bad-tolerance 1000
WARNING: neo4j-import is deprecated and support for it will be removed in a future
version of Neo4j; please use neo4j-admin import instead.

Neo4j version: 3.1.2
Importing the contents of these files into /var/lib/neo4j/data/databases/yelp.db:
Nodes:
  /home/lyonwj/neo4j-datasets/yelp/data/import/user.csv

  /home/lyonwj/neo4j-datasets/yelp/data/import/business.csv

  /home/lyonwj/neo4j-datasets/yelp/data/import/review.csv
Relationships:
  /home/lyonwj/neo4j-datasets/yelp/data/import/wrote.csv

  /home/lyonwj/neo4j-datasets/yelp/data/import/review_of.csv

Available resources:
  Free machine memory: 20.38 GB
  Max heap memory : 13.98 GB
  Processors: 20
```

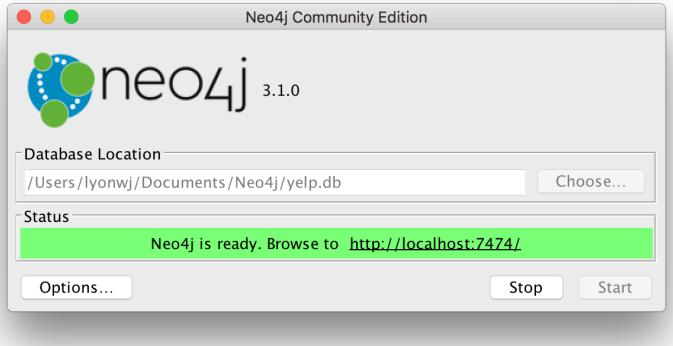
```
IMPORT DONE in 6m 36s 363ms.
Imported:
  5326654 nodes
  38029547 relationships
  9479804 properties
Peak memory usage: 151.23 MB
```

Now included in `neo4j-admin import`

neo4j-import

```
#*****************************************************************
# Neo4j configuration
#
# For more details and a complete list of settings, please see
# https://neo4j.com/docs/operations-manual/current/reference/configuration-settings/
#*****************************************************************

# The name of the database to mount
dbms.active_database=yelp.db
```

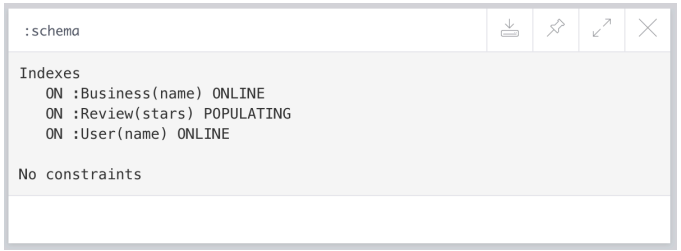
neo4j.conf



Desktop App

Create Indexes

```
CREATE INDEX ON :User(name);
CREATE INDEX ON :Business(name);
CREATE INDEX ON :Review(stars);
```



Neo4j Drivers

Neo4j Drivers

All Neo4j Drivers

Thanks to the Neo4j contributor community, there are additionally drivers for almost every popular programming language, most of which mimic existing database driver idioms and approaches. Get started with your stack now, see the dedicated page for more detail.

.NET	Java	Spring	Neo4j-OGM	JavaScript
Python	Ruby	PHP	R	Go
Erlang / Elixir	C/C++	Clojure	Perl	Haskell

https://neo4j.com/developer/language-guides/

```
from neo4j.v1 import GraphDatabase

driver = GraphDatabase.driver("bolt://localhost:7687", auth=("neo4j", "password"))

business_obj = {
    "business_id": "123",
    "name": "Bob's Liquor",
    "city": "San Mateo",
    "state": "CA",
    "categories": ["Liquor", "Lottery", "UPS Locker"]
}

with driver.session() as session:
    session.run('''
        WITH {business} AS business
        CREATE (b:Business {business_id: business.business_id})
        SET b.name = business.name,
            b.city = business.city
        WITH *
        UNWIND business.categories AS cat
        MERGE (c:Category {name: cat})
        MERGE (b)-[:IN_CATEGORY]->(c)
    ''', parameters={'business': business_obj}).consume()
```

https://github.com/johnymontana/neo4j-datasets/blob/master/yelp/src/Yelp_import_examples.ipynb

```
from neo4j.v1 import GraphDatabase
import json

driver = GraphDatabase.driver("bolt://localhost:7687", auth=("neo4j", "password"))

BUSINESS_FILE = "../data/yelp_academic_dataset_business.json"

with open(BUSINESS_FILE, 'r') as file:
    with driver.session() as session:
        for line in file:
            item = json.loads(line)

            session.run('''
                WITH {business} AS business
                CREATE (b:Business {business_id: business.business_id})
                SET b.name = business.name,
                    b.city = business.city
                WITH *
                UNWIND business.categories AS cat
                MERGE (c:Category {name: cat})
                MERGE (b)-[:IN_CATEGORY]->(c)
            ''', parameters={'business': item}).consume()
```

```
from neo4j.v1 import GraphDatabase
import json

driver = GraphDatabase.driver("bolt://localhost:7687", auth=("neo4j", "password"))

BUSINESS_FILE = "../data/yelp_academic_dataset_business.json"

with open(BUSINESS_FILE, 'r') as file:
    with driver.session() as session:
        count = 0
        tx = session.begin_transaction()
        for line in file:
            item = json.loads(line)
            tx.run('''
                WITH {business} AS business
                CREATE (b:Business {business_id: business.business_id})
                SET b.name = business.name,
                    b.city = business.city
                WITH *
                UNWIND business.categories AS cat
                MERGE (c:Category {name: cat})
                MERGE (b)-[:IN_CATEGORY]->(c)
            ''', parameters={'business': item})
            count += 1
            if count > 1000:
                tx.commit()
                tx = session.begin_transaction()
                count = 0
        tx.commit()
```

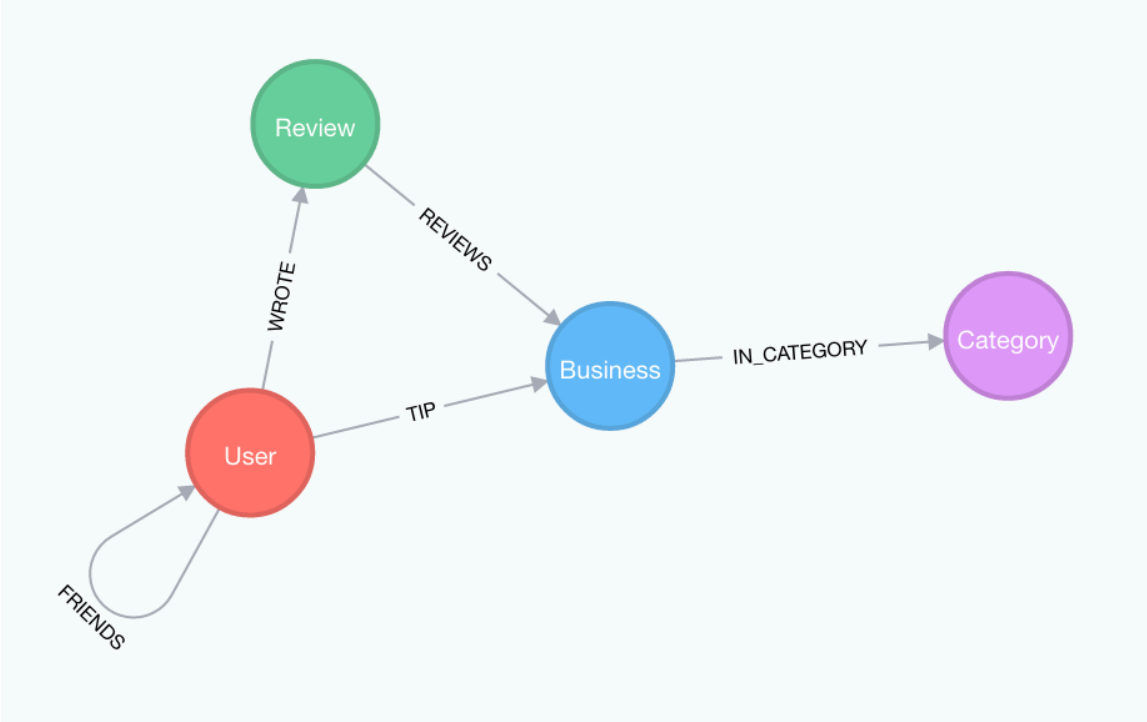
```
from neo4j.v1 import GraphDatabase
import json

driver = GraphDatabase.driver("bolt://localhost:7687", auth=("neo4j", "password"))

BUSINESS_FILE = "../data/yelp_academic_dataset_business.json"

with open(BUSINESS_FILE, 'r') as file:
    with driver.session() as session:
        count = 0
        items = []
        for line in file:
            item = json.loads(line)
            items.append(item)
            count += 1
            if count > 1000:
                session.run('''
                    WITH {businesses} AS businesses
                    UNWIND businesses AS business
                    CREATE (b:Business {business_id: business.business_id})
                    SET b.name = business.name,
                        b.city = business.city
                    WITH *
                    UNWIND business.categories AS cat
                    MERGE (c:Category {name: cat})
                    MERGE (b)-[:IN_CATEGORY]->(c)
                ''', parameters={'businesses': items}).consume()
                count = 0

        # run query again for partial remainder...
```

```
from neo4j.v1 import GraphDatabase
import json, queue, threading

driver = GraphDatabase.driver("bolt://localhost:7687", auth=("neo4j", "password"))

BUSINESS_FILE = "../data/yelp_academic_dataset_business.json"

INSERT_QUERY = '''
WITH {businesses} AS businesses
                    UNWIND businesses AS business
                    CREATE (b:Business {business_id: business.business_id})
                    SET b.name = business.name,
                        b.city = business.city
                    WITH *
                    UNWIND business.categories AS cat
                    MERGE (c:Category {name: cat})
                    MERGE (b)-[:IN_CATEGORY]->(c)
'''

# init the work queue
global q
q = queue.LifoQueue()

def run_query():
    while True:
        items = q.get()
        with driver.session() as session:
            session.run(INSERT_QUERY, parameters={'businesses': items}).consume()
            q.task_done()

for i in range(16):
    t = threading.Thread(target=run_query)
    t.daemon = True
    t.start()

with open(BUSINESS_FILE, 'r') as file:
    with driver.session() as session:
        count = 0
        items = []
        for line in file:
            items.append(json.loads(line))
            count += 1
            if count > 1000:
                q.put(items)
                items = []
                count = 0
            q.put(items)

q.join()
```

```
from neo4j.v1 import GraphDatabase, CypherError
import json, queue, threading

driver = GraphDatabase.driver("bolt://localhost:7687", auth=("neo4j", "password"))

BUSINESS_FILE = "../data/yelp_academic_dataset_business.json"

INSERT_QUERY = '''
WITH {businesses} AS businesses
                    UNWIND businesses AS business
                    CREATE (b:Business {business_id: business.business_id})
                    SET b.name = business.name,
                        b.city = business.city
                    WITH *
                    UNWIND business.categories AS cat
                    MERGE (c:Category {name: cat})
                    MERGE (b)-[:IN_CATEGORY]->(c)
'''

# init the work queue
global q
q = queue.LifoQueue()

global single_threaded_q
single_threaded_q = queue.LifoQueue()



def run_query():
    while True:
        parts = q.get()
        with driver.session() as session:
            try:
                session.run(INSERT_QUERY, parameters={'businesses': parts[0]}).consume()
                q.task_done()
            except CypherError as e:
                if parts[1] < 2:
                    q.put((parts[0], parts[1]+1))
                else:
                    single_threaded_q.put((parts[0], parts[1]+1))
                q.task_done()

def run_single_threaded():
    while True:
        parts = single_threaded_q.get()
        with driver.session() as session:
            session.run(INSERT_QUERY, parameters={'businesses': parts[0]}).consume()
            single_threaded_q.task_done()

for i in range(16):
    t = threading.Thread(target=run_query)
    t.daemon = True
    t.start()

t = threading.Thread(target=run_single_threaded)
t.daemon = True

with open(BUSINESS_FILE, 'r') as file:
    with driver.session() as session:
        count = 0
        items = []
        for line in file:
            items.append(json.loads(line))
            count += 1
            if count > 1000:
                #     (params, number of retries)
                q.put((items, 0))
                items = []
                count = 0
            q.put((items, 0))

q.join()
single_threaded_q.join()
```

Querying The Graph

The Graph


Review
REVIEWS
WROTE
Business
IN_CATEGORY
Category
User
TIP
FRIENDS

What Business Has Highest Reviews

```
MATCH (b:Business)<-[:REVIEW_OF]-(r:Review)
WITH b, avg(r.stars) AS mean
RETURN b.name, mean ORDER BY mean DESC LIMIT 25
```

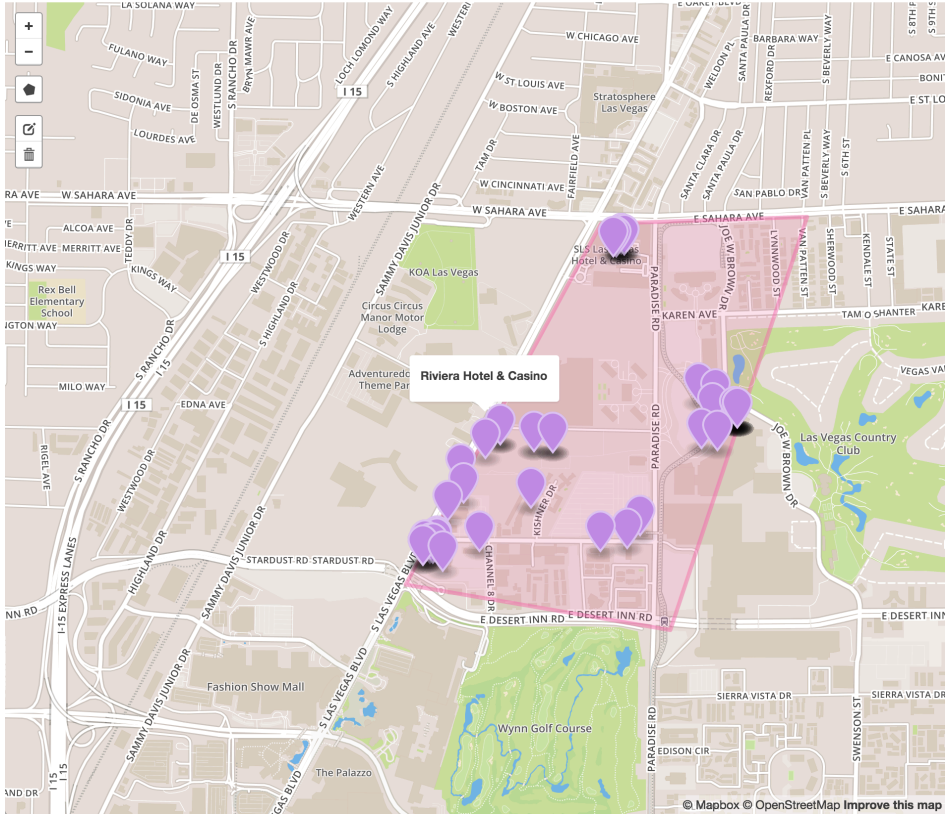
b.name	mean
Red Leg Burgers & Bratwurst	5
Publix Ballantyne	5
Auto Glass Authority	5
Fabrizio Divari Art	5
AKA Dog Obedience Training, LLC	5
Home Fur Good	5
PURE Radiance	5
Paul's Clock Repair	5
Uncle Bob's Self Storage	5
Dryer Fires LV	5
Public Storage	5
Monica's Flowers	5

What kind of questions can we ask?

Spatial Cypher Demo

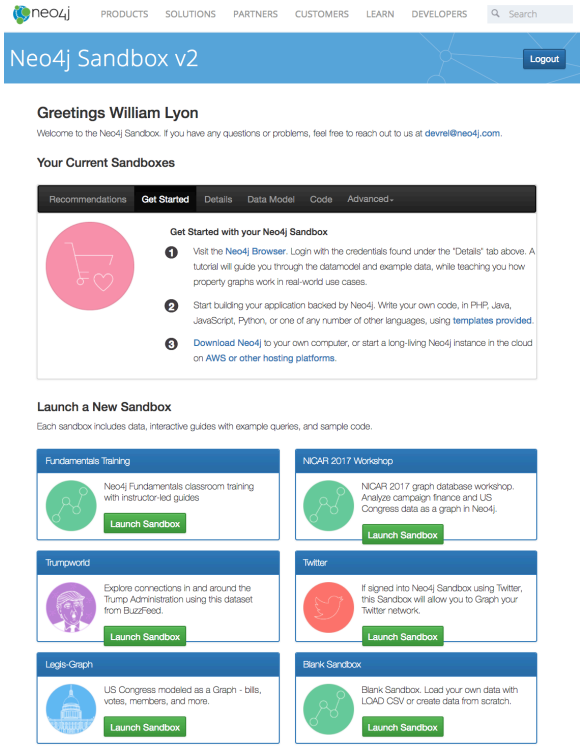
Restaurants

Select a category then use the polygon icon below to draw a polygon on the map! Double click when you have selected the last point.
Businesses matching the selected category will be displayed within your polygon.



http://www.lyonwj.com/scdemo/index.html

Neo4j Sandbox



neo4jsandbox.com