

# Working With A Real World Dataset In Neo4j

## Modeling and Import

William Lyon  
@lyonwj



# William Lyon

Developer Relations Engineer @neo4j

[will@neo4j.com](mailto:will@neo4j.com)  
[@lyonwj](https://twitter.com/lyonwj)  
[lyonwj.com](http://lyonwj.com)

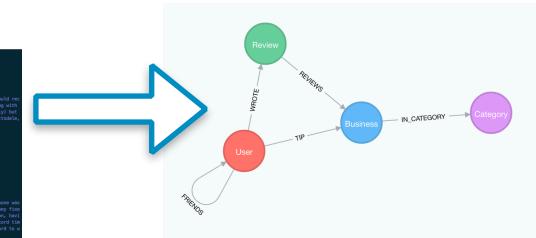


# Agenda



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

```
1   |
2   |     "type": "Review"
3   |     "id": "2C3634B0B7BD58A7D2C2D9",
4   |     "name": "Movie"
5   |     "review_count": 764
6   |     "rating": 3.6
7   |     "url": "http://www.movie.com/reviews"
8   |   }
9   | }
10  |
11  |     "type": "User"
12  |     "id": "1C9AB85255F9E4E59A0E4100B9990000",
13  |     "name": "Reviews_averages",
14  |     "review_count": 764
15  |     "rating": 3.6
16  |     "url": "http://www.movie.com/reviews"
17  |
18  |     "type": "Business"
19  |     "id": "2A78948D808520000000000000000000",
20  |     "name": "Shangri-La Hotel, Kuala Lumpur",
21  |     "review_count": 337
22  |     "rating": 4.3
23  |     "url": "http://www.movie.com/reviews"
24  |   }
25  |
26  |     "type": "Category"
27  |     "id": "1C9AB85255F9E4E59A0E4100B9990000",
28  |     "name": "Business"
29  |     "review_count": 337
30  |     "rating": 4.3
31  |     "url": "http://www.movie.com/reviews"
32  |   }
33  |
34  |     "type": "User"
35  |     "id": "1C9AB85255F9E4E59A0E4100B9990000",
36  |     "name": "Reviews_averages",
37  |     "review_count": 764
38  |     "rating": 3.6
39  |     "url": "http://www.movie.com/reviews"
40  |
41  |     "type": "Review"
42  |     "id": "2C3634B0B7BD58A7D2C2D9",
43  |     "name": "Movie"
44  |     "review_count": 764
45  |     "rating": 3.6
46  |     "url": "http://www.movie.com/reviews"
47  |
48  |     "type": "User"
49  |     "id": "1C9AB85255F9E4E59A0E4100B9990000",
50  |     "name": "Reviews_averages",
51  |     "review_count": 764
52  |     "rating": 3.6
53  |     "url": "http://www.movie.com/reviews"
54  |
55  |     "type": "Business"
56  |     "id": "2A78948D808520000000000000000000",
57  |     "name": "Shangri-La Hotel, Kuala Lumpur",
58  |     "review_count": 337
59  |     "rating": 4.3
60  |     "url": "http://www.movie.com/reviews"
61  |
62  |     "type": "Category"
63  |     "id": "1C9AB85255F9E4E59A0E4100B9990000",
64  |     "name": "Business"
65  |     "review_count": 337
66  |     "rating": 4.3
67  |     "url": "http://www.movie.com/reviews"
68  |
69  |     "type": "User"
70  |     "id": "1C9AB85255F9E4E59A0E4100B9990000",
71  |     "name": "Reviews_averages",
72  |     "review_count": 764
73  |     "rating": 3.6
74  |     "url": "http://www.movie.com/reviews"
75  |
76  |     "type": "Review"
77  |     "id": "2C3634B0B7BD58A7D2C2D9",
78  |     "name": "Movie"
79  |     "review_count": 764
80  |     "rating": 3.6
81  |     "url": "http://www.movie.com/reviews"
82  |
83  |     "type": "User"
84  |     "id": "1C9AB85255F9E4E59A0E4100B9990000",
85  |     "name": "Reviews_averages",
86  |     "review_count": 764
87  |     "rating": 3.6
88  |     "url": "http://www.movie.com/reviews"
89  |
90  |     "type": "Business"
91  |     "id": "2A78948D808520000000000000000000",
92  |     "name": "Shangri-La Hotel, Kuala Lumpur",
93  |     "review_count": 337
94  |     "rating": 4.3
95  |     "url": "http://www.movie.com/reviews"
96  |
97  |     "type": "Category"
98  |     "id": "1C9AB85255F9E4E59A0E4100B9990000",
99  |     "name": "Business"
100 |     "review_count": 337
101 |     "rating": 4.3
102 |     "url": "http://www.movie.com/reviews"
103 | }
```



# The Codez



[johnymontana / neo4j-datasets](https://github.com/johnymontana/neo4j-datasets/tree/master/yelp)

Code Issues 0 Pull requests 0 Projects 0 Wiki Graphs Settings

Branch: master neo4j-datasets / yelp / Create new file Upload files Find file History

johnymontana add convert to csv notebook Latest commit 11620f4 7 hours ago

..

img add yelp 9 days ago

src add convert to csv notebook 7 hours ago

README.md add yelp 9 days ago

README.md

Working with the Yelp Data Challenge in Neo4j

```
graph TD; User((User)) -- Friends --> User; User -- Tip --> Business((Business)); User -- Review --> Review((Review)); Business -- IN_CATEGORY --> Category((Category))
```

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

# The Data



# **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

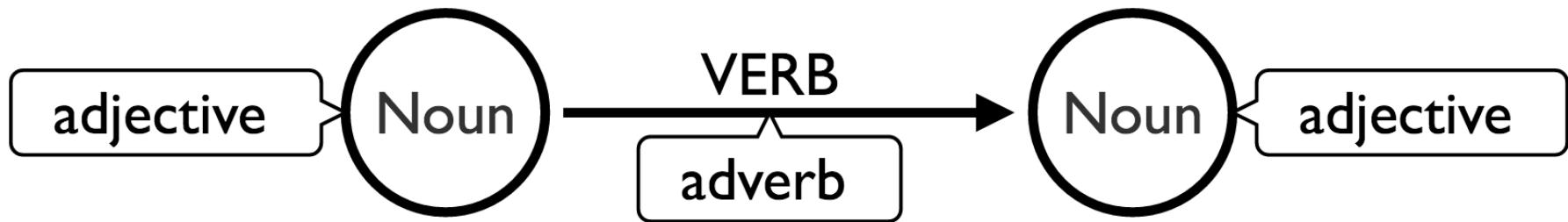
[https://www.yelp.com/dataset\\_challenge](https://www.yelp.com/dataset_challenge)

# Graph Data Model

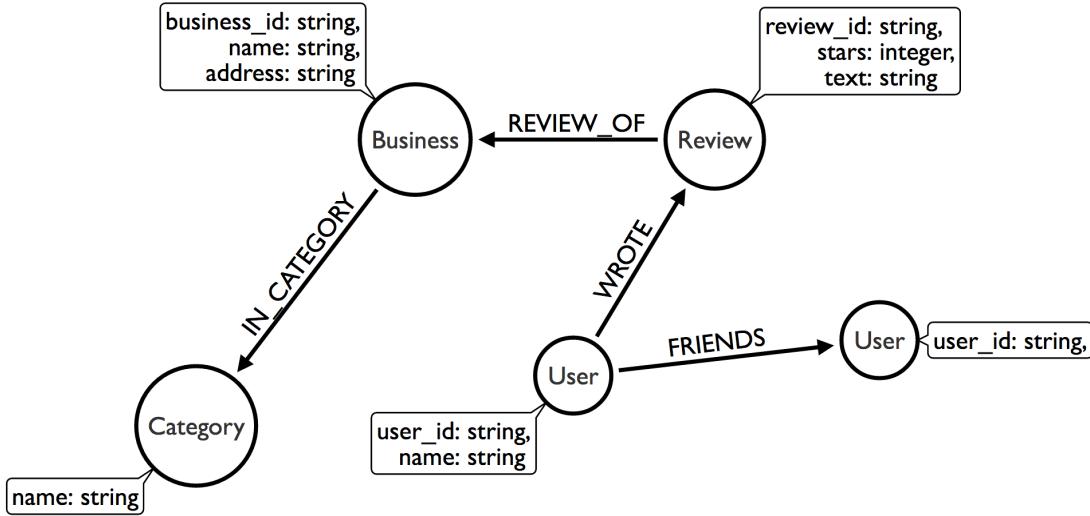
# Labeled Property Graph Model



# Labeled Property Graph Model



# Our Data Model



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

# 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*

```
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)
```

CYPHER

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

# Convert streaming JSON to JSON using jq



./jq

jq is a lightweight and flexible command-line JSON processor.

[Download jq 1.5 ▾](#)

[Try online at jqplay.org!](#)

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

# 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



```
1 [ {  
2   "user_id": "EZmocAborM6z66rTze2x20",  
3   "name": "Rob",  
4   "review_count": 761,  
5   "yelping_since": "2009-09-12",  
6   "friends": [  
7     "lJg9ekPzF91kMuvjKYX6u",  
8     "ctWAuz504Xu0ke2rop4l0",  
9     "BBCapjOneBXRSJS5Ky0v0",  
10    "J9sKtA4fVWk4hySpoPA",  
11    "Ec-epoSawV16e90tMbjw",  
12    "r2UUCzCgx1QWps1mPgqG2A",  
13    "3ybkL7N63Usn4wepInZuW",  
14    "d-lzusSagnKduiyL1TFSpw",  
15    "Ydh2zA5w1D-Ubapp8t0G4",  
16    "DeZinc-RsNmksL10LUksw",  
17    "NTuvVb-ZvQ_rFn6wKrn7A",  
18    "PCdUs3LB1h0oretyQ6_RA",  
19    "RYIhfaNekKLduqwnqKV13g",  
20    "i-m0ueJWKpKsceL7JXjT2w",  
21    "ST_2Rwfzvf-nhZYD8RsVid",  
22    "DT06vhKzyxB8MrVbTgowl",  
23    "k0Uzr44poPwN6n5420eHMo",  
24    "x-lu62BV7jYhFvg3f0kKhv",  
25    "8gy1M4qCnOL8qiwD22GL0",  
26    "8esb0FK-w5xRmLnZvnaAd0",  
27    "YeEieu7wGTM9PjBsClykt0",  
28    "EJ8pe71KG0Ux025b19ACkW",  
29    "22K0meq7zf3MwB8wlpstU4",  
30    "U08mDYLNa03g12zmHnyuA",  
31    "a-Ug_MFrYz3uIca-NaHw0",  
32    "lFXAyG1Yel02ksdax3aD0",  
33    "kjSHVCJ1f6z7gp72w08j0",  
34  ]
```

user.json

```
1 [ {  
2   {  
3     "business_id": "00180t2Pjp97XWVvIElicQ",  
4     "name": "Innovative Vapors",  
5     "neighborhood": "",  
6     "address": "227 E Baseline Rd, Ste J2",  
7     "city": "Tempe",  
8     "state": "AZ",  
9     "postal_code": "85283",  
10    "latitude": 33.3782141,  
11    "longitude": -111.936102,  
12    "stars": 4.5,  
13    "review_count": 17,  
14    "is_open": 0,  
15    "attributes": [  
16      "BikeParking": True,  
17      "BusinessAcceptsBitcoin": False,  
18      "BusinessAcceptsCreditCards": True,  
19      "BusinessParking": {'garage': False, 'street': False, 'validated': False, 'lot': True, 'value': False},  
20      "DogsAllowed": False,  
21      "RestaurantsPriceRange2": 2,  
22      "WheelchairAccessible": True  
23    ],  
24    "categories": [  
25      "Tobacco Shops",  
26      "Nightlife",  
27      "Vape Shops",  
28      "Shopping"  
29    ],  
30    "hours": [  
31      "Monday 11:0-21:0",  
32      "Tuesday 11:0-21:0",  
33      "Wednesday 11:0-21:0",  
34      "Thursday 11:0-21:0",  
35    ]  
36  }  
37 }
```

business.json

```
1 [ {  
2   {  
3     "review_id": "NxLSIC5yg0dn1Xc18IBg",  
4     "user_id": "Kpk0KGRI4fRa25Lhhbf1A",  
5     "business_id": "2aFjy99NLklCx3T_tG59A",  
6     "stars": 5,  
7     "date": "2011-10-19",  
8     "text": "If you enjoy service by someone who is as competent as he is personable, I would recommend Corey Kaplan highly. The time he has spent here has been very productive and working with him educational and enjoyable. I hope not to need him again (though this is highly unlikely) but knowing he is there if I do is very nice. By the way, I'm not from El Centro, CA, but Scottsdale, AZ.",  
9     "useful": 0,  
10    "funny": 0,  
11    "cool": 0,  
12    "type": "review"  
13  },  
14  {  
15    "review_id": "pXbbIG0XvLuTi_SPs1hQE0",  
16    "user_id": "b071Q1o1n9MK-gxRsrgr",  
17    "business_id": "2aFjy99NLklCx3T_tG59A",  
18    "stars": 5,  
19    "date": "2010-12-29",  
20    "text": "After being on the phone with Verizon Wireless trying to figure out why my phone was n't working for 4.5 hours, I was put in touch with Sharpie Tech. Well, after 10 seconds they fix d the problem. As the owner of a company that does our best numbers over the holiday season, having my phone out of order for 4.5 hours was horrible. Sharpie Tech fixed the problem in record tim e, even Verizon was shocked. I highly recommend working with this company and I look forward to w orking with them more. \n\n-Rachel Charlupski\nFounder and CEO, The Babysitting Company",  
21    "useful": 1,  
22    "funny": 0,  
23    "cool": 0,  
24    "type": "review"  
25  },  
26  {  
27    "review_id": "wplw2Lu4My1b1jEpAGs",  
28  }  
29 }
```

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 delimited 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](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-gXRsrA","pXbbIg0XvLuTi_SPs1hQE","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","2aFiy99vNLk1Cx3T_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","2aFiy99vNLk1Cx3T_tGS9A"
"5","aW3ix1KNZAv0M8q-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_sX6uve-IR-P0jQ"
"4","Y0o-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_sX6uve-IR-P0jQ"
"5","bg13j8yJcR0-00NkUYsXGQ","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_sX6uve-IR-P0jQ"
"4","CwKF9de-nskLYEqDDCfubg","oFmVZh-La7SuvpHrH_A14Q","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

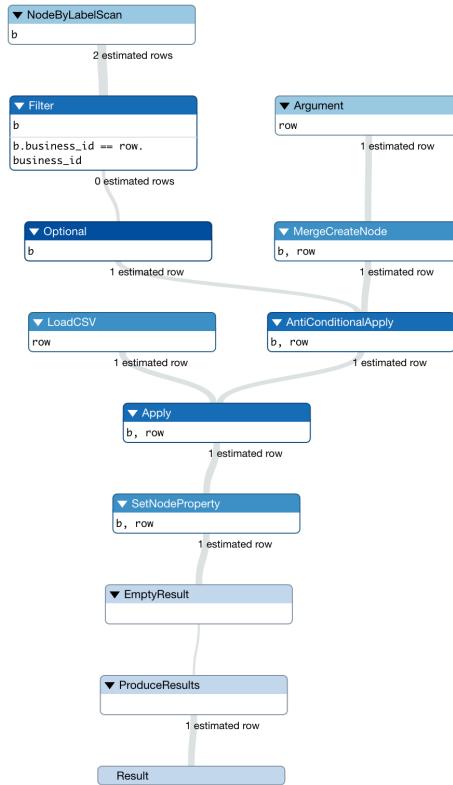


```
LOAD CSV WITH HEADERS FROM "file:///business.csv" AS row
MERGE (b:Business {business_id: row.business_id})
ON CREATE SET b.name = row.name ...
```

```
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)
```

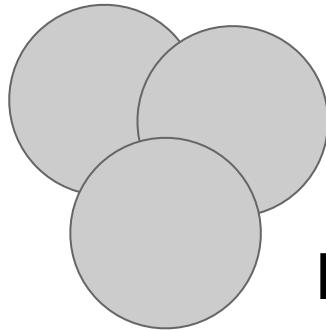
# EXPLAIN



# How does Neo4j use indexes?

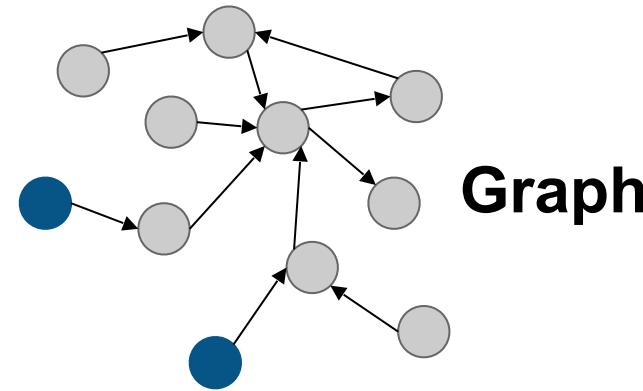


Indexes are **only** used to find the starting point for queries.



**Relational**

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



Use indexes to find the starting points for a query.

# Create Index / Constraint



## Constraint + index

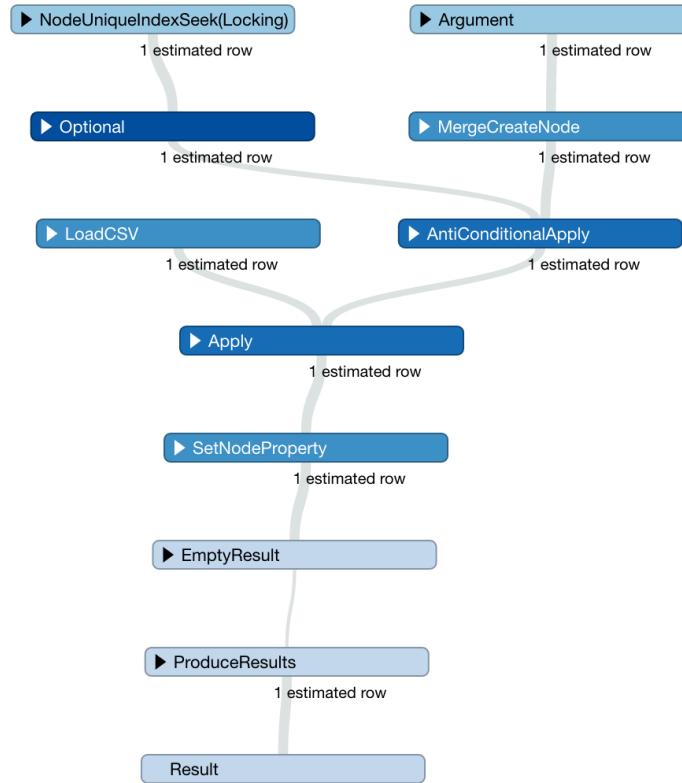
```
CREATE CONSTRAINT ON (u:User) ASSERT u.user_id IS UNIQUE;  
CREATE CONSTRAINT ON (b:Business) ASSERT b.business_id IS UNIQUE;
```

## Index

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

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

# EXPLAIN



# 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)
```

# **cypher-shell** Run multi-line Cypher scripts



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

```
~/Dropbox/Copy/neotechnology/datasets/yelp/neo4j-enterprise-3.1.1 ➜ master ➜ vim ../
src/simple_import.cypher
```

A dark terminal window with a white cursor at the bottom center. The path ~/Dropbox/Copy/neotechnology/datasets/yelp/neo4j-enterprise-3.1.1 is visible at the top, along with the current branch master and the command vim ../. The file src/simple\_import.cypher is open in the background.

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

# 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



# 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

```
1 [||||| | 90.3%] 6 [||||| | 93.4%] 11 [||||| | 85.3%] 16 [||||| | 88.9%]
2 [||||| | 97.4%] 7 [||||| | 91.5%] 12 [||||| | 91.6%] 17 [||||| | 90.8%]
3 [||||| | 92.9%] 8 [||||| | 95.4%] 13 [||||| | 92.2%] 18 [||||| | 87.0%]
4 [||||| | 91.7%] 9 [||||| | 98.1%] 14 [||||| | 95.3%] 19 [||||| | 91.4%]
5 [||||| | 87.3%] 10 [||||| | 85.9%] 15 [||||| | 96.7%] 20 [||||| | 95.5%]
Mem [||||| | 15.0G/62.9G] Tasks: 42, 207 thr; 8 running
Swp [ 0K/0K] Load average: 1.75 0.38 0.12
Uptime: 4 days, 09:56:03
```



# neo4j-import file format



Nodes

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

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

Relationships

	:START_ID(User)	:END_ID(Review)	:TYPE
123	123	127	WROTE
124	124	128	WROTE
125	125	129	WROTE
126	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 wrotesfile:
            wroteswriter = csv.writer(wrotesfile, escapechar='\\', quotechar='', quoting=csv.QUOTE_ALL)
            wroteswriter.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', ' ')])
                # (:User)-[:WROTE]->(:Review)
                wroteswriter.writerow([item['user_id'], item['review_id'], 'WROTE'])

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

```

# 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/neo4j/data/databases/yelp.db --nodes user.csv --nodes business.csv --nodes review.csv --relationships wrote.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**

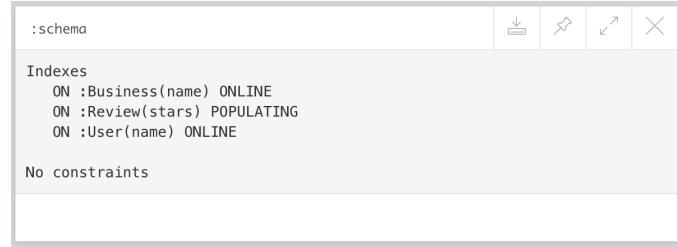


**Desktop App**

# Create Indexes



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



A screenshot of the Neo4j browser interface showing the results of a schema query. The top bar has icons for saving, running, and closing. The main area shows the following output:

```
:schema  
Indexes  
ON :Business(name) ONLINE  
ON :Review(stars) POPULATING  
ON :User(name) ONLINE  
No constraints
```

# 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

```
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](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

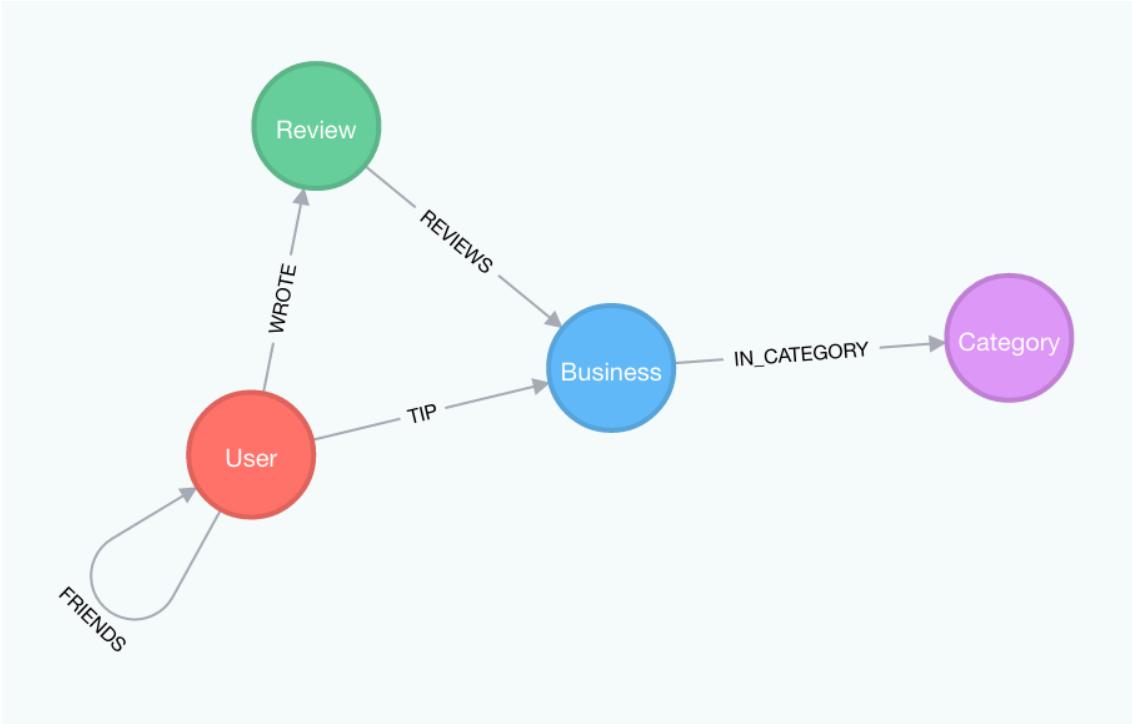
with open(BUSINESS_FILE, 'r') as file:
    with driver.session() as session:
        count = 0
        items = []
        for line in file:
            items.append(json.loads(line))
            count += 1
            if count > 1000:
                # (params, number of retries)
                q.put((items,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

q.join()
single_threaded_q.join()

```

# Querying The Graph

# The Graph



# 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?

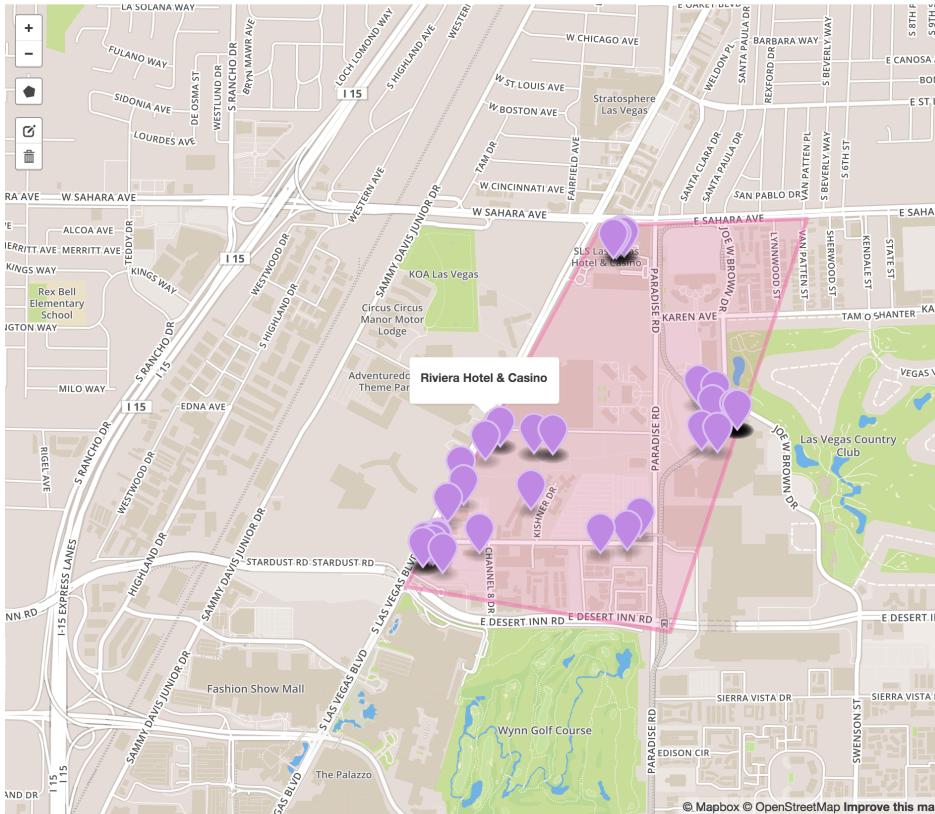


# Spatial Cypher Demo

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



neo4j PRODUCTS SOLUTIONS PARTNERS CUSTOMERS LEARN DEVELOPERS  Search

## Neo4j Sandbox v2

Greetings William Lyon

Welcome to the Neo4j Sandbox. If you have any questions or problems, feel free to reach out to us at [devrel@neo4j.com](mailto:devrel@neo4j.com).

### Your Current Sandboxes

Recommendations **Get Started** Details Data Model Code Advanced -

**Get Started with your Neo4j Sandbox**

- 1 Visit the Neo4j Browser. Login with the credentials found under the "Details" tab above. A tutorial will guide you through the datamodel and example data, while teaching you how properties work in real-world use cases.
- 2 Start building your application backed by Neo4j. Write your own code, in PHP, Java, JavaScript, Python, or one of any number of other languages, using [templates provided](#).
- 3 [Download Neo4j](#) to your own computer, or start a long-living Neo4j instance in the cloud on AWS or other hosting platforms.

**Launch a New Sandbox**

Each sandbox includes data, interactive guides with example queries, and sample code.

<b>Fundamentals Training</b> Neo4j Fundamentals classroom training with instructor-led guides  <a href="#">Launch Sandbox</a>	<b>NICAR 2017 Workshop</b> NICAR 2017 graph database workshop. Analyze campaign finance and US Congress data as a graph in Neo4j.  <a href="#">Launch Sandbox</a>
<b>Trumpworld</b> Explore connections in and around the Trump Administration using this dataset from BuzzFeed.  <a href="#">Launch Sandbox</a>	<b>Twitter</b> If signed into Neo4j Sandbox using Twitter, this Sandbox will allow you to Graph your Twitter network.  <a href="#">Launch Sandbox</a>
<b>Legi-Graph</b> US Congress modeled as a Graph - bills, votes, members, and more.  <a href="#">Launch Sandbox</a>	<b>Blank Sandbox</b> Blank Sandbox. Load your own data with LOAD CSV or create data from scratch.  <a href="#">Launch Sandbox</a>

[neo4jsandbox.com](http://neo4jsandbox.com)