SELA

Graph database using Neo4j

Avi Avni

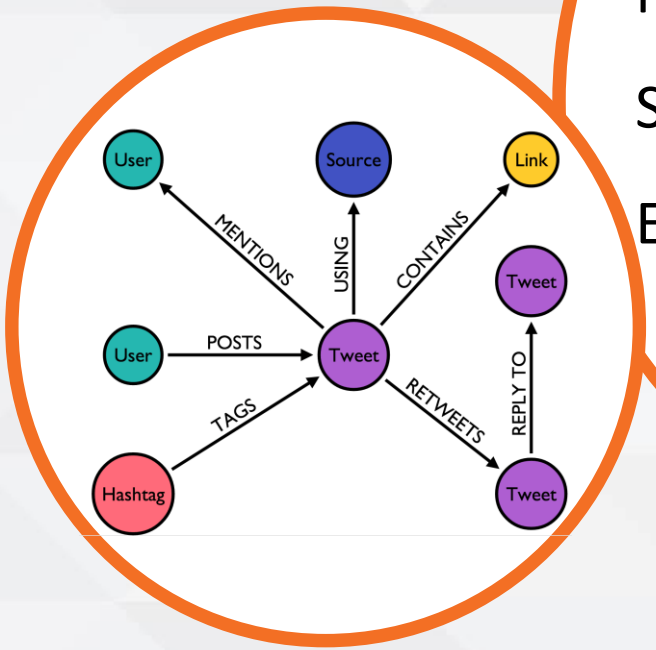
SELA

Agenda

- What is a Graph Database?
- Graph Database motivations
- Graph vs Relational databases
- Why Neo4j?
- Real world use cases
- Cypher vs Gremlin
- Data Modeling techniques
- Neo4j Deployment
- Neo4j Configuration
- Profiling Queries
- Graph Algorithms
- Neo4j and Spark

SELA

What is Graph Database?



NoSql database

Store data in a graph **structure**

Efficiently **traverses** the graph

SELA

What is Graph Database?

- Nodes
  - Entities
- Relationships
  - Connect entities and structure domain
- Properties
  - Attributes and metadata
- Labels
  - Group nodes by role

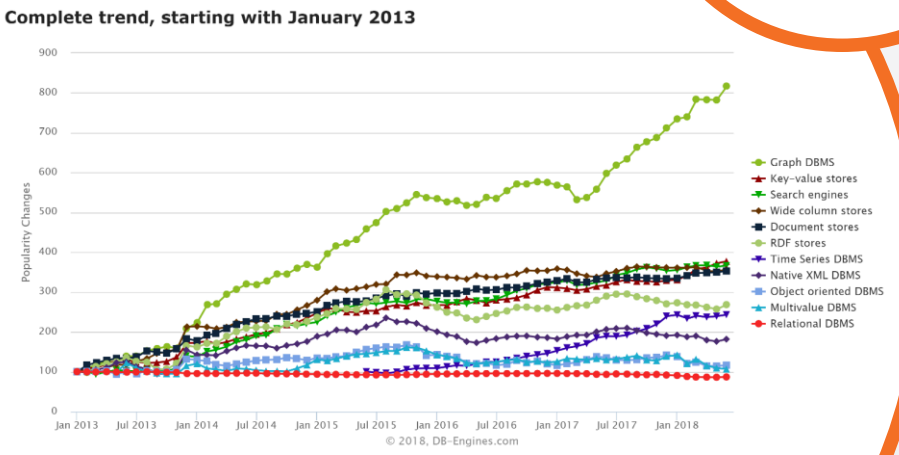
SELA

Graph Database motivation

- Solve problems that can't be solved with RDBMS
  - Think how to model Facebook data?
- Whiteboard friendly
- Productivity and Agility
  - Schema-less
  - Complex logic simple query
- Visualization of the data
- Performance

SELA

Graph Database motivation



https://db-engines.com/en/ranking_categories

Graph vs Relational databases

Graph

- Graph traversal
- Schemaless
- Related Data
  - How X related to Y?
- Clustering, Centrality, Influence
  - Who is the most influential persons?

RDBMS

- Table joins
- Strict schema
- Search data by properties
  - Get all rows with X?
- Aggregate
  - How many X in my system?

SELA

Why Neo4j?

Great **Community**

Native graph storage

Cypher query language

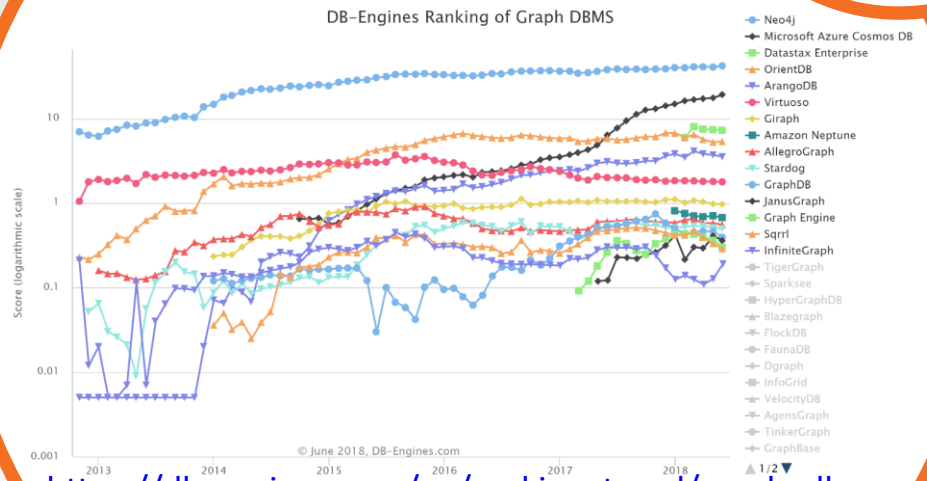
Scalability using clustering

Great query **profiling** tool

Easy to **use**

SELA

Why Neo4j?



https://db-engines.com/en/ranking_trend/graph+dbms

SELA

Real world use cases

- Real-Time recommendation
- Social Network
  - Like Facebook and Instagram
- Resource management
  - Finding the correct permission to a file
- Fraud detection
- Route Finding

SELA

Cypher query language

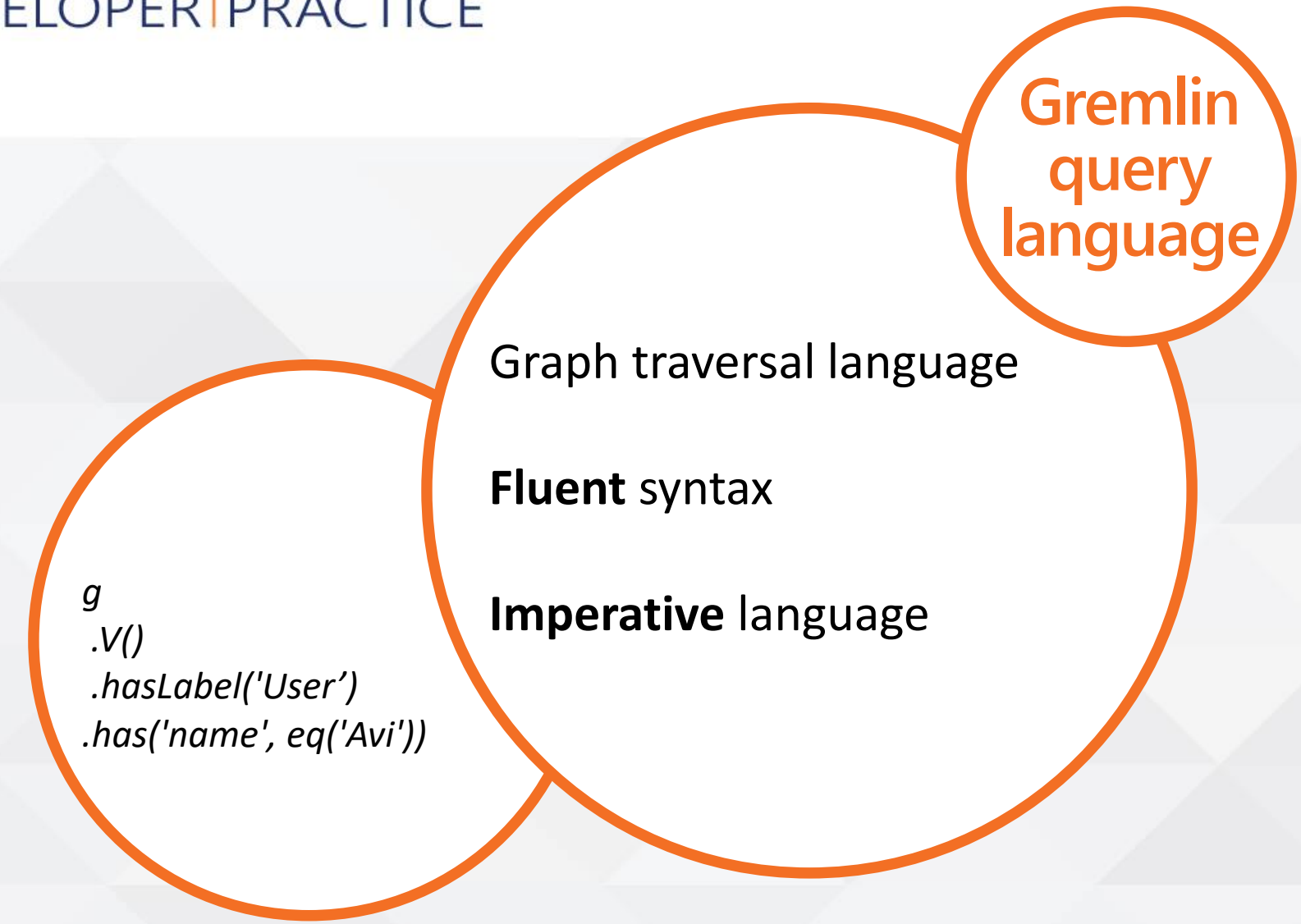
Declarative language

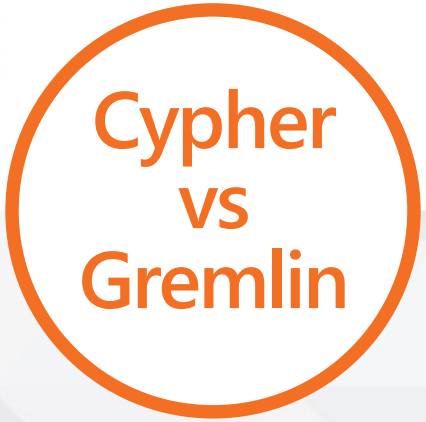
Inspired by SQL

Describes **patterns** in graphs

ASCII-art query language

```
MATCH (u:User)
WHERE u.name = "Avi"
RETURN u
```






```
CREATE (a:User { name: "Avi" })
CREATE (o:User { name: "Ofir" })
CREATE (s:User { name: "Sasha" })
CREATE (m:User { name: "Moshe" })
```

```
g.addV('User').as('a').property(single, 'name', 'Avi')
 .addV('User').as('o').property(single, 'name', 'Ofir')
 .addV('User').as('s').property(single, 'name', 'Sasha')
 .addV('User').as('m').property(single, 'name', 'Moshe')
```

Cypher vs Gremlin

```
CREATE (a)-[:Follow]->(o)
CREATE (a)-[:Follow]->(s)
CREATE (a)-[:Follow]->(m)
CREATE (o)-[:Follow]->(a)
```

```
.addE('Follow').from('a').to('o')
.addE('Follow').from('a').to('s')
.addE('Follow').from('a').to('m')
.addE('Follow').from('o').to('a')
```

Cypher vs Gremlin

```
MATCH (u:User)
WHERE u.name = "Avi"
RETURN u
```

```
g.V().hasLabel('User').has('name', eq('Avi'))
```

Cypher vs Gremlin

```
MATCH (u:User)-[:Follow]->(fu:User)
WHERE u.name = "Avi"
RETURN u, fu
```

```
g.V().as('u').hasLabel('User').has('name', eq('Avi'))
  .outE('Follow').inV().as('fu').hasLabel('User')
  .select('u', 'fu')
```

Cypher vs Gremlin

```
MATCH (u:User)-[:Follow]->(fu:User)-[:Follow]->(ffu:User)
WHERE u.name = "Avi"
RETURN u, fu, ffu
```

```
g.V().as('u').hasLabel('User').has('name', eq('Avi'))
  .outE('Follow').inV().as('fu').hasLabel('User')
  .outE('Follow').inV().as('ffu').hasLabel('User')
  .select('u', 'fu', 'ffu')
```

Cypher vs Gremlin

```
MATCH (u:User)-[:Follow]->(:User)<-[f:Follow]-(ru:User)
WHERE u.name = "Avi" AND NOT EXISTS((u)-[:Follow]->(ru))
RETURN ru, count(f) as mutual_followers
ORDER BY mutual_followers DESC
LIMIT 10
```

Cypher vs Gremlin


```
g.V().as('u').hasLabel('User').has('name', eq('A                    ').as('
UNNAMED15').inV().hasLabel('User').inE('Follow                    V().as('ru').hasLabel('User').where(__.and(__.sel
ect('  UNNAMED15').where(neq('f')), __.not(__.select('u').outE('Follow').inV().as('
GENERATED5').where(__.select('  GENERATED5').where(eq('ru'))).choose(__.is(neq('  cypher.null')),
__.constant(true), __.constant(false)).is(neq('  cypher.null')).is(eq(true))))).select('ru',
'f').group().by(__.select('ru')).by(__.fold().project('  FRESHID120', '
FRESHID124').by(__.unfold().select('ru')).by(__.unfold().select('f').is(neq('
cypher.null')).count()))).unfold().select(values).as('  GENERATED6').select('  FRESHID120').as('
FRESHID120').select('  GENERATED6').select('  FRESHID124').as('  FRESHID124').select('  FRESHID120', '
FRESHID124').project('  FRESHID120', '  FRESHID124').by(__.select('  FRESHID120')).by(__.select('
FRESHID124')).order().by(__.select('  FRESHID124'), decr).limit(10).as('  GENERATED7').select('
FRESHID120').as('  FRESHID120').select('  GENERATED7').select('  FRESHID124').as('  FRESHID124').select('
FRESHID120', '  FRESHID124').project('ru', 'mutual_followers').by(__.select('  FRESHID120').choose(neq('
cypher.null'), __.valueMap(true), __.constant('  cypher.null'))).by(__.select('  FRESHID124'))
```

SELA

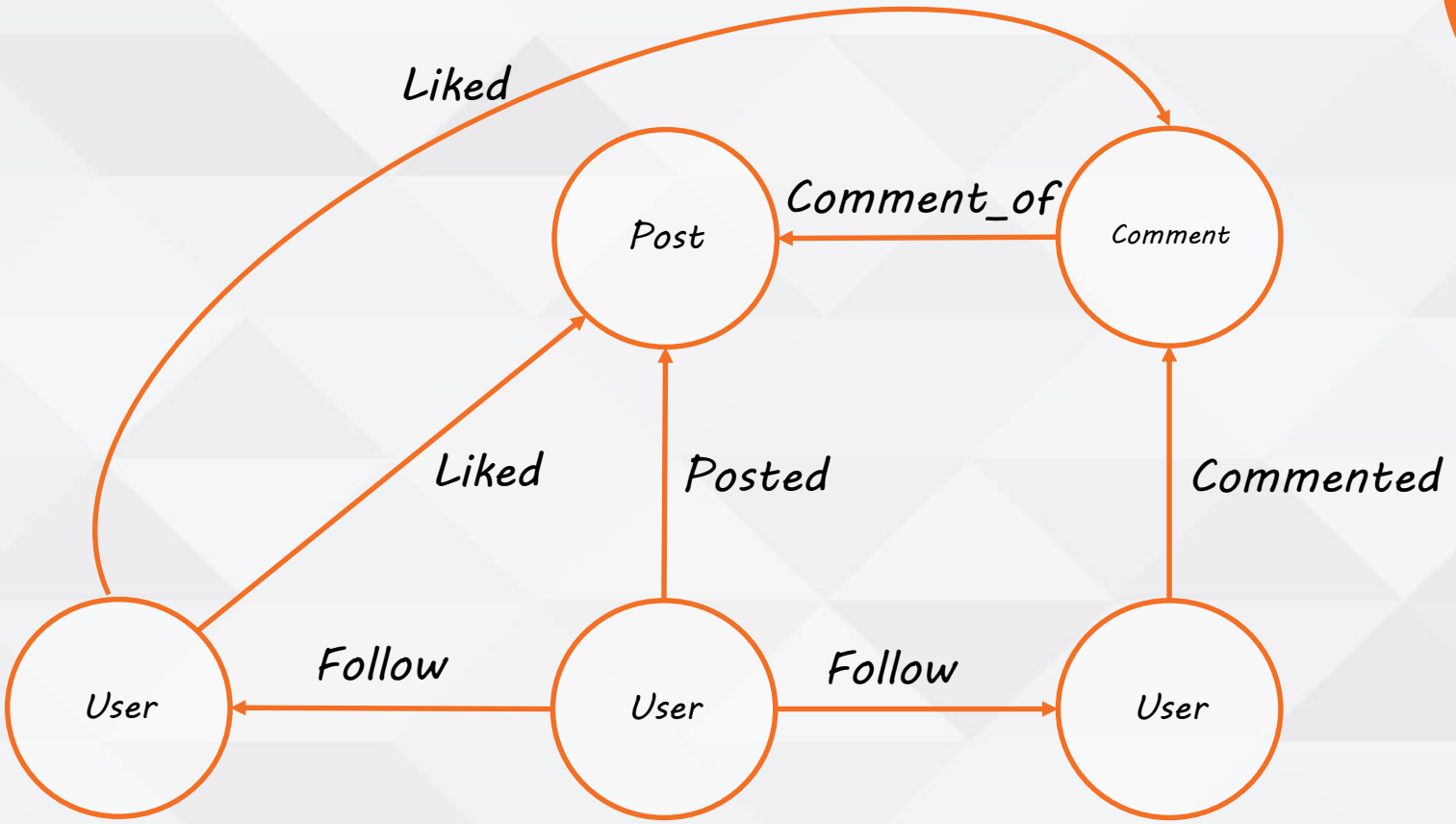
Exercise

- Use Movies dataset
- Search for movie you watched lately
- If it not exists add it to the dataset
- Search the most 10 prolific actors
- Recommend movies to the user “Joshua Elliott” based on movies he really liked and the actors acted in that movies

Use Cheat sheet
https://neo4j.com/docs/cypher-refcard/current/
Use Neo4J Sandbox
https://neo4j.com/sandbox-v2/

SELA

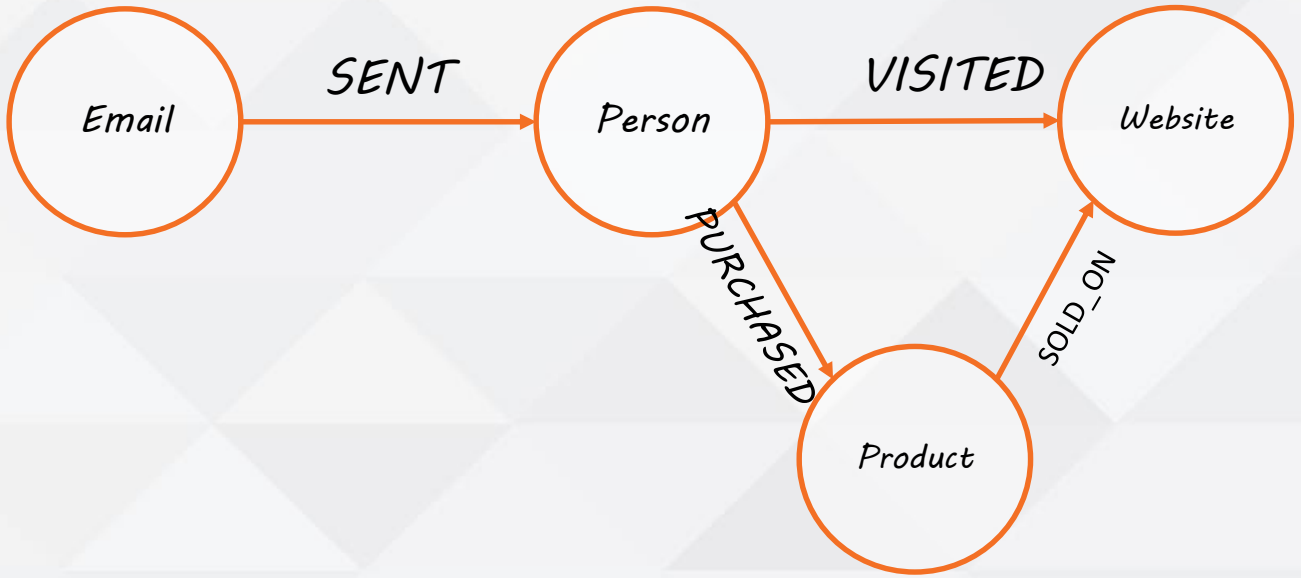
Data Modeling techniques



Data Modeling techniques

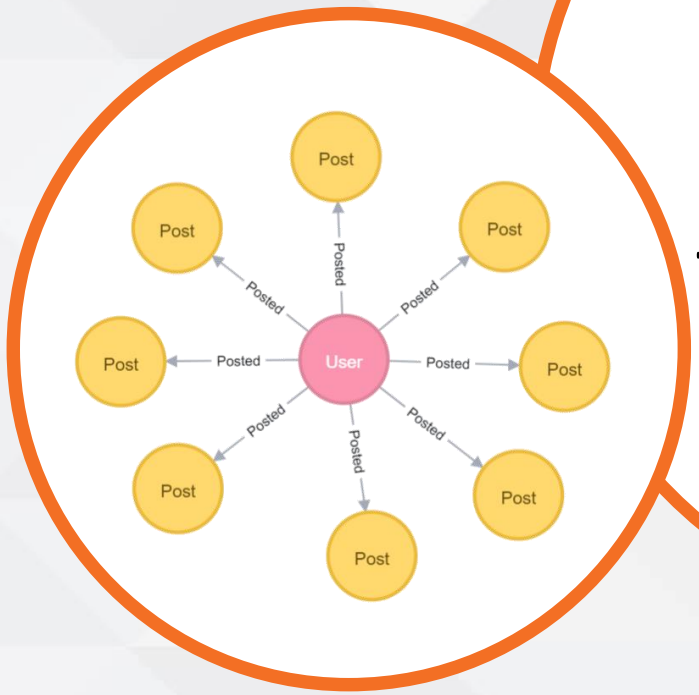
Language oriented example

We send email to people
so they will visit our website
and buy our product.



SELA

Data Modeling techniques



Star Model

- One to many relation
- User posted a post
- User followed other user

Trivial to understand

Data Modeling techniques

Social network feed requirement:

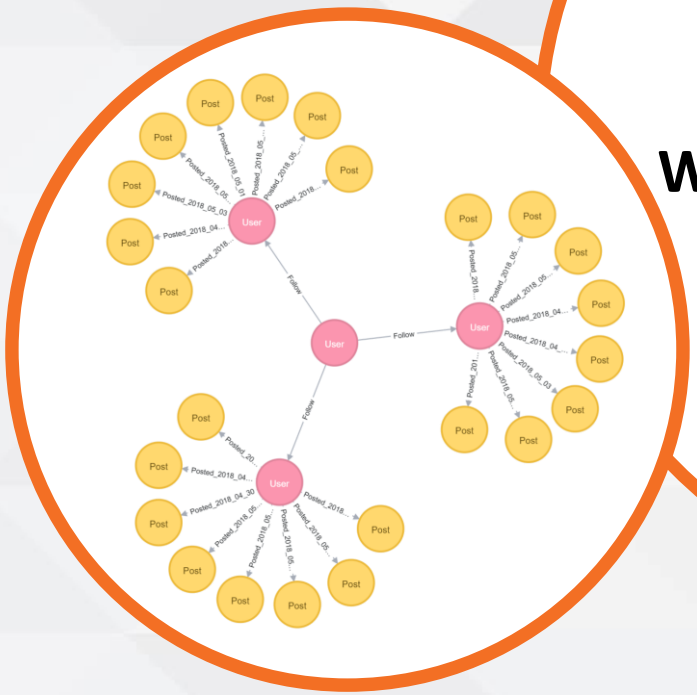
Return the X recent posts belong to my friends

What about Performance?

1000(friends)*
1000(posts)=
1,000,000(posts)

SELA

Data Modeling techniques



Dated Model

Embed the date
in the relationship type

What about Performance?

1000(friends)*
10(posts in last X days)=
10,000(posts)

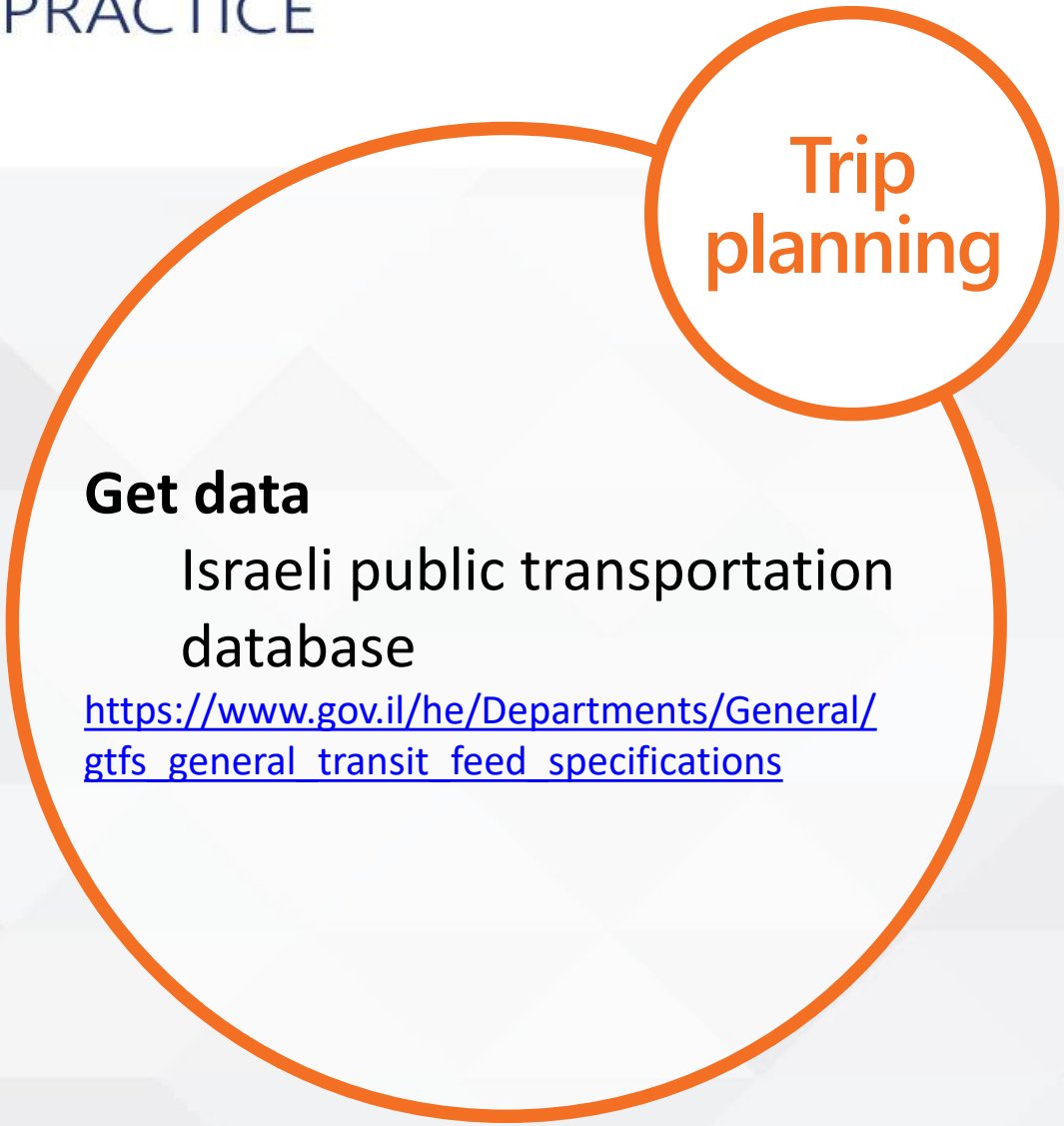
SELA

Trip planning

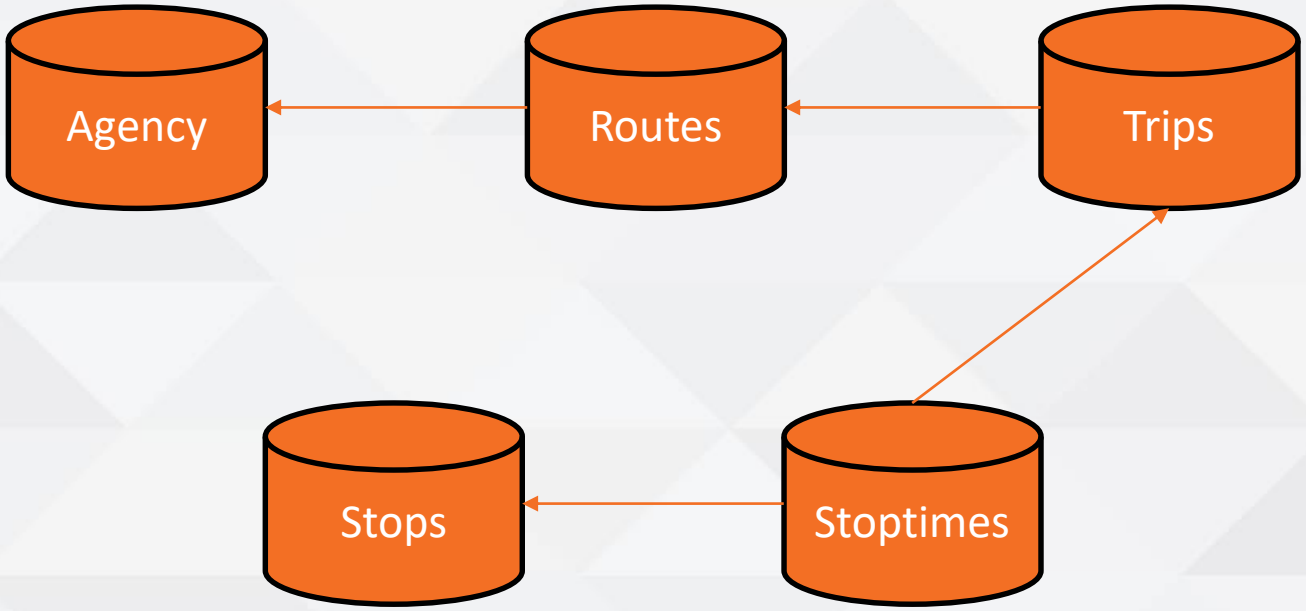
Build Moovit application

Requirement:

How to get from X to Y with public transportation?

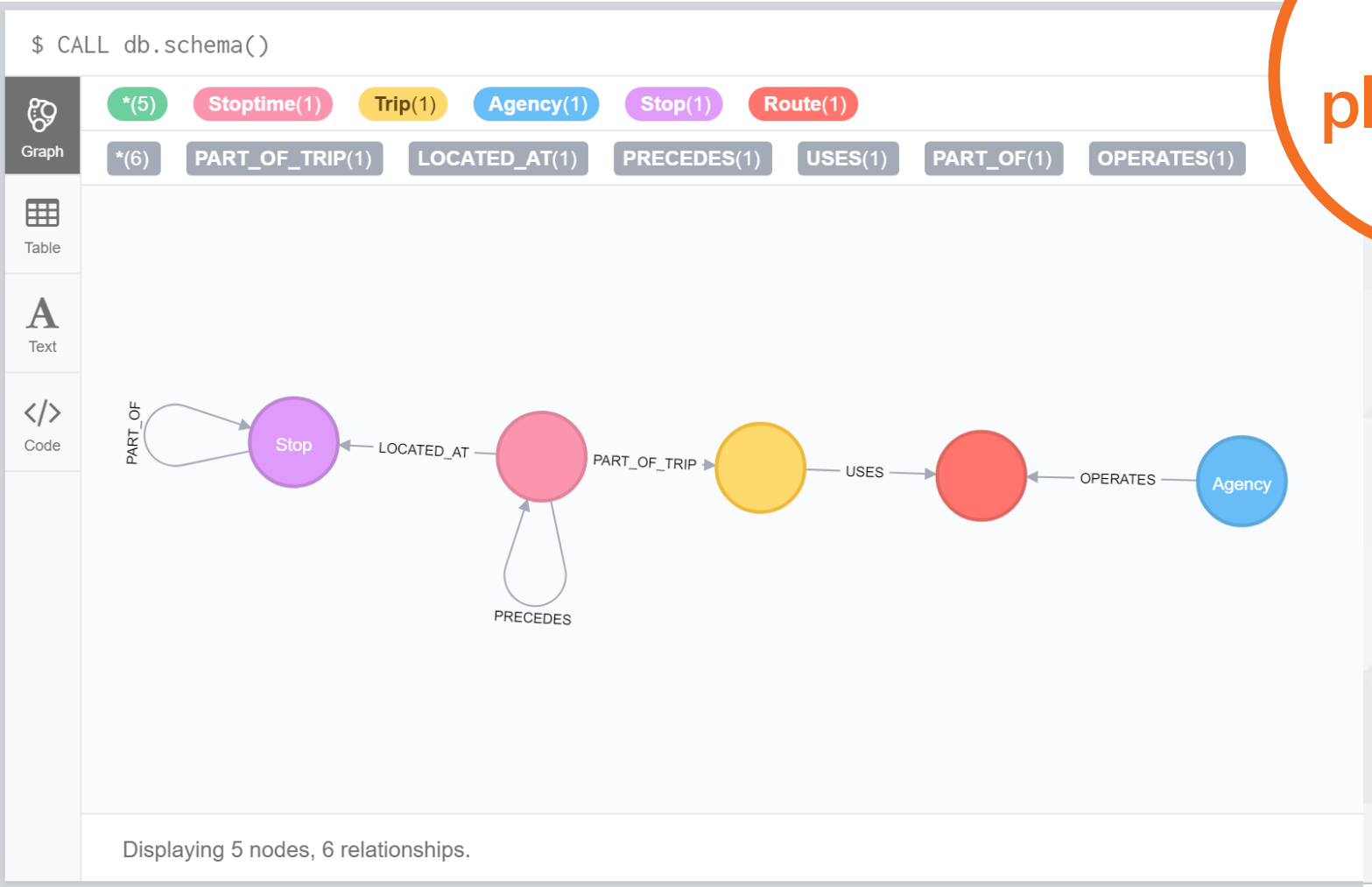


Trip planning

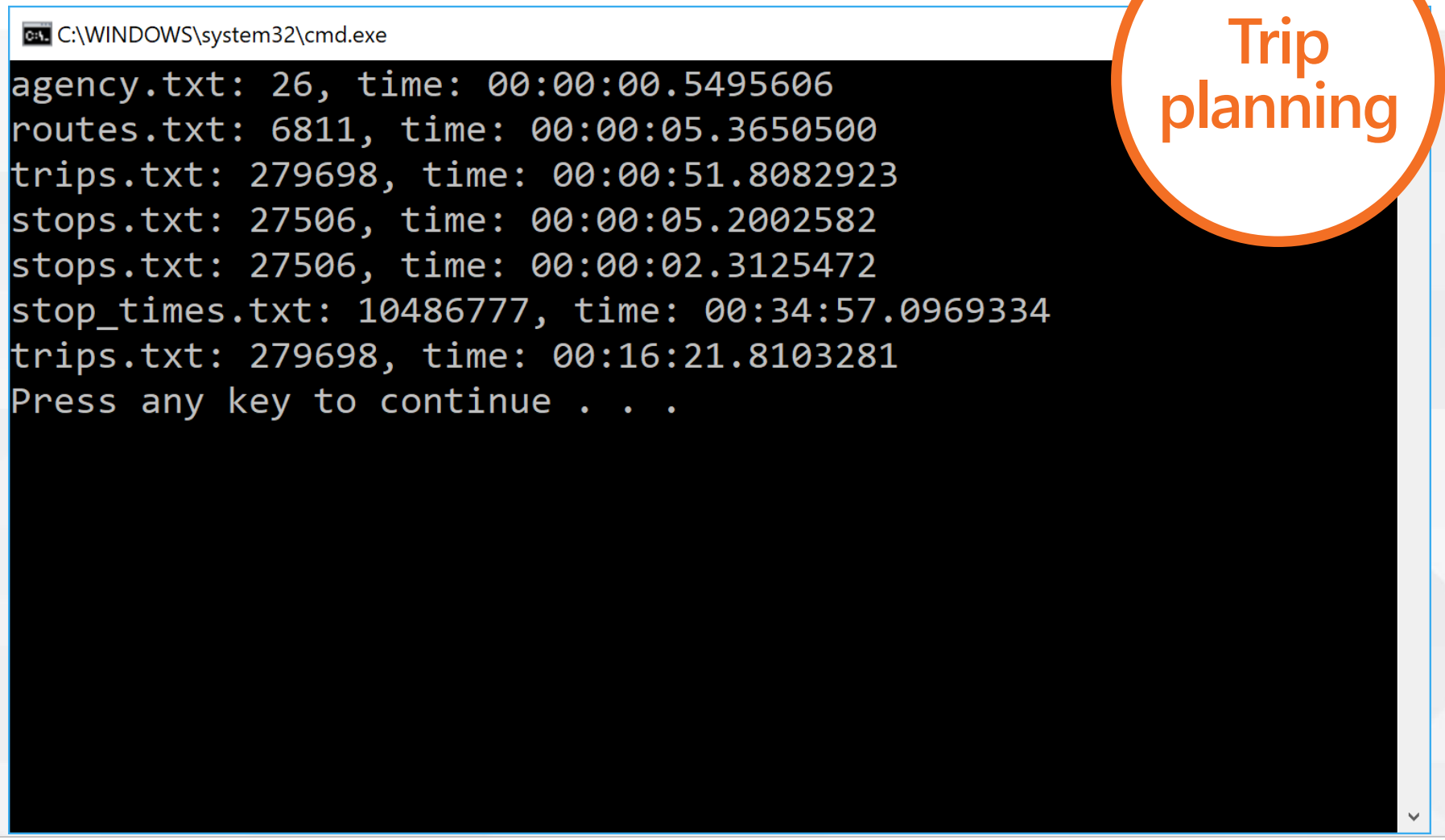


SELA

Trip planning



SELA



Trip planning

SELA

Exercise

- Model a graph database that will helps your organization to build and maintain a better product
- The solution will help to answer questions like
  - If a developer change some code what will be affected?
  - Which part of the application needs architectural/design decisions? (usually parts with most of the bugs)

SELA

Exercise

- The solution will help to
  - Developers
  - Architectures
  - Program managers
  - Product managers
  - Testers
- Consider uses data sources like
  - Source Code
  - Source Control
  - Tests results
  - Company Structure
  - Issue Tracker
  - Release Management

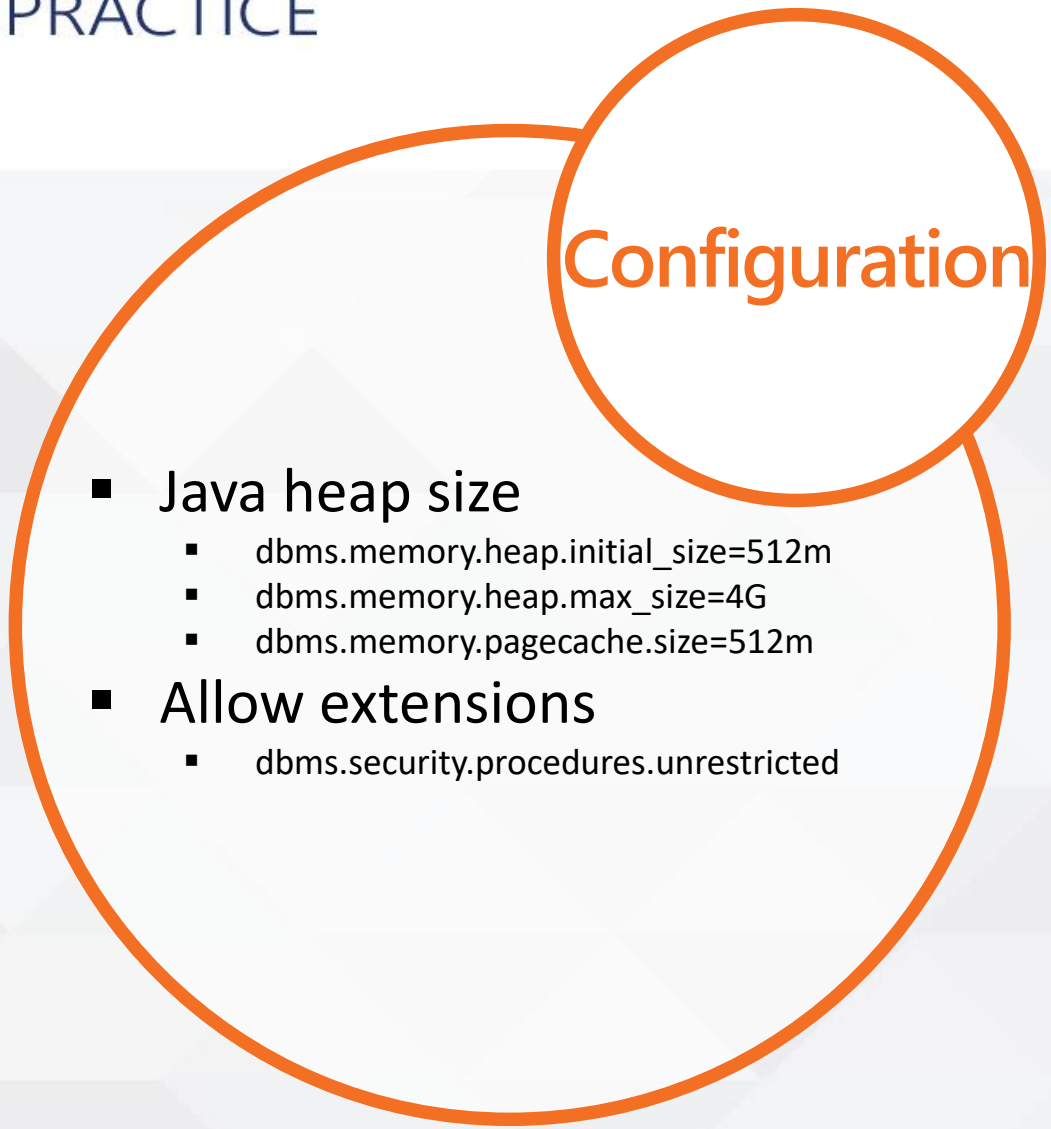
SELA

Exercise

- Draw the schema of the graph
- Write example query
- What else can be considered and how it affects the model?

Deployment

- Neo4j load the graph to memory ensure you have enough RAM
- All data is replicated on all nodes ensure you have enough disk size
- Backup the data with tool
- Monitor the database



SELA

Profiling Queries

EXPLAIN

Shows the execution plan without actually executing it or returning any results.

PROFILE

Executes the statement and returns the results along with profiling information

SELA

Profiling Queries

GOAL

Get the number of db hits down.

Db hit

An abstract unit of storage engine work

Create index for start node pattern

```
CREATE INDEX ON :User(name)

MATCH (u:User)
WHERE u.name = "AviAvni"
RETURN u
```

Profiling Queries

Reduce cardinality

```
MATCH (u:User)-[:Stared]->(r1)<-[:Stared]-
      (coUser)-[:Stared]->(r2)
WHERE u.name = "AviAvni"
RETURN DISTINCT r2.name
```

Profiling Queries

Reduce cardinality

```
MATCH (u:User)-[:Stared]->(r1)<-[:Stared]-
      (coUser)
WHERE u.name = "AviAvni"
WITH DISTINCT coUser
MATCH (coUser)-[:Stared]->(r2)
RETURN DISTINCT r2.name
```

Profiling Queries

Count relationships

```
MATCH (u:User)-[:Stared]->()
RETURN u, COUNT(*) as count
ORDER BY count DESC

MATCH (u:User)
RETURN u, SIZE((u)-[:Stared]->()) as count
ORDER BY count DESC
```

Profiling Queries

Avoid cartesian products

```
MATCH (u:User), (r:Repo)
RETURN count(u), count(r)

MATCH (u:User)
WITH count(u) as u_count
MATCH (r:Repo)
RETURN a_count, count(r)
```

SELA

Graph Algorithms

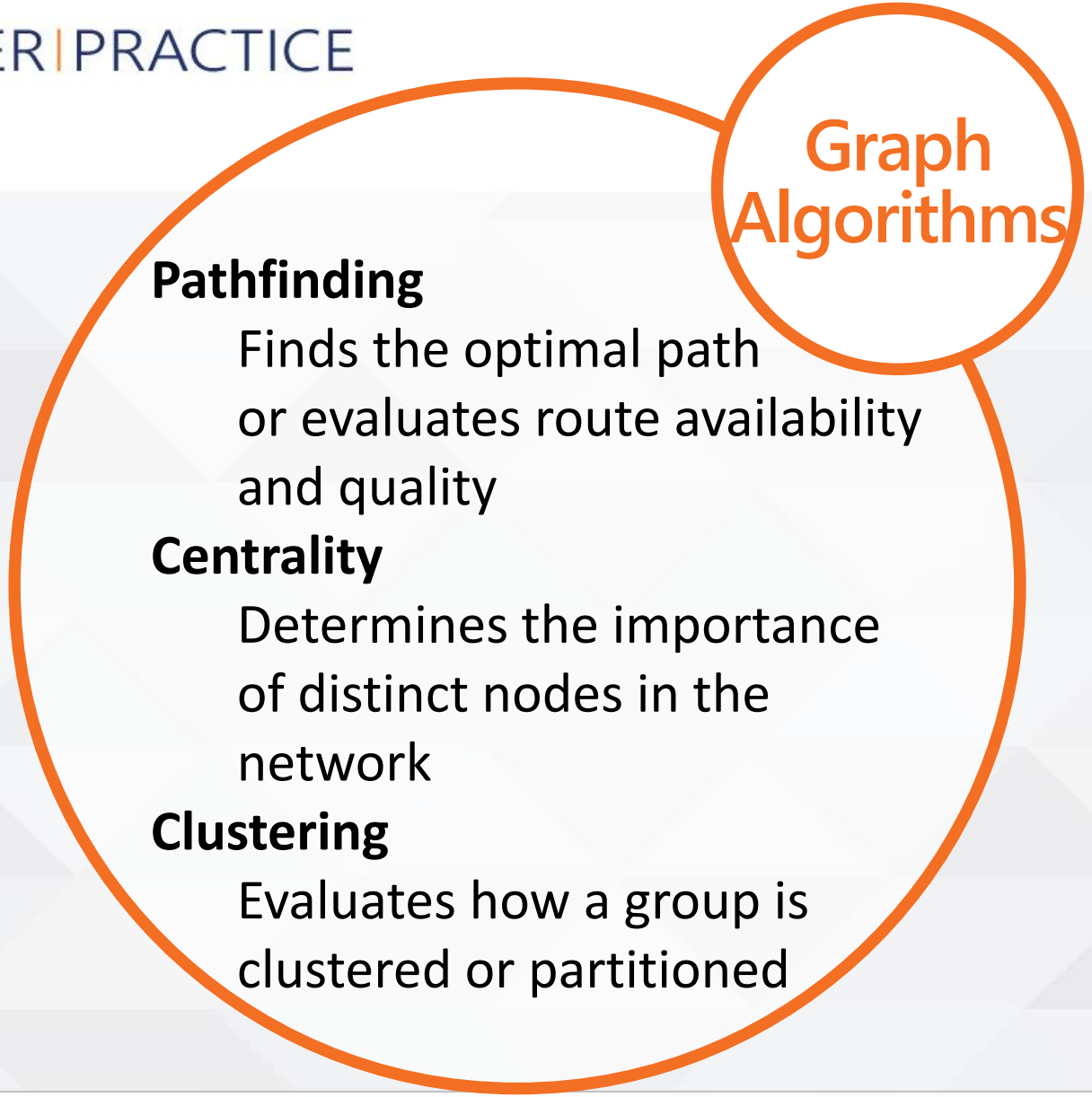
- Graph algorithms plugin
- High performance
- User friendly using stored procedure
- Stream the result
- Write the result asynchronously

SELA

Graph Algorithms

https://neo4j.com/whitepapers/graph-algorithms-neo4j-ebook/?ref=social-camp-sales-intro

https://neo4j.com/docs/graph-algorithms/current/



SELA

Neo4j and Spark

- Neo4j Spark connector
  - https://github.com/neo4j-contrib/neo4j-spark-connector
  - Manipulate neo4j graphs in spark
  - Use partition and batch carefully

Neo4j and Spark

- Spark GraphX
  - https://spark.apache.org/docs/1.2.1/graphx-programming-guide.html
  - Spark distributed graph algorithms
    - PageRank
    - Connected Components
    - Triangle Counting
  - Pregel API

SELA

Summary

- Look for **connections** in you data
- Cross language between business and development
- Choose the right model
- Use productive query language
- Plan for deployment
- Profile query before usage
- Choose the right algorithm and run it in the right server

- **Demo**: https://github.com/AviAvni/GraphDatabaseDemo
- **Neo4j Sandbox:** https://neo4j.com/sandbox-v2/