Analytics & Graphs:
Neo4j Connector for Apache Spark

Michael Hunger

Director Developer Relations

Java Champion

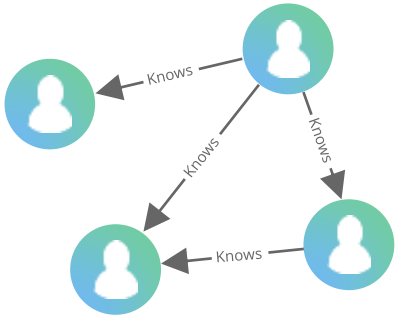
Twitter: @mesirii

medium.com/@mesirii

github.com/jexp

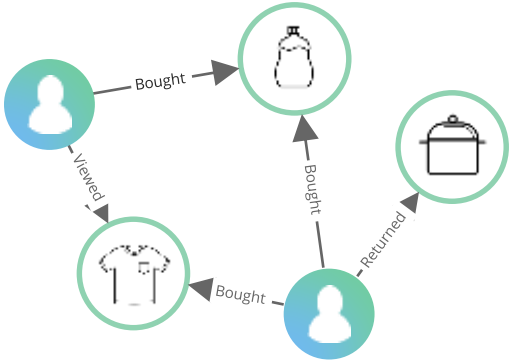
neo4j

Connections in Data are as Valuable as the Data Itself



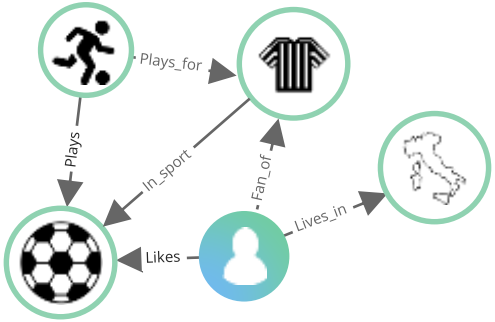
Networks of People

E.g., Employees, Customers, Suppliers, Partners, Influencers



Transaction Networks

E.g., Risk management, Supply chain, Payments



Knowledge Networks

E.g., Enterprise content, Domain specific content, eCommerce content

neo4j

Analyzing the FinCEN Files with Neo4j

ICIJ INTERNATIONAL CONSORTIUM of INVESTIGATIVE JOURNALISTS

neo4j

The FinCEN files?

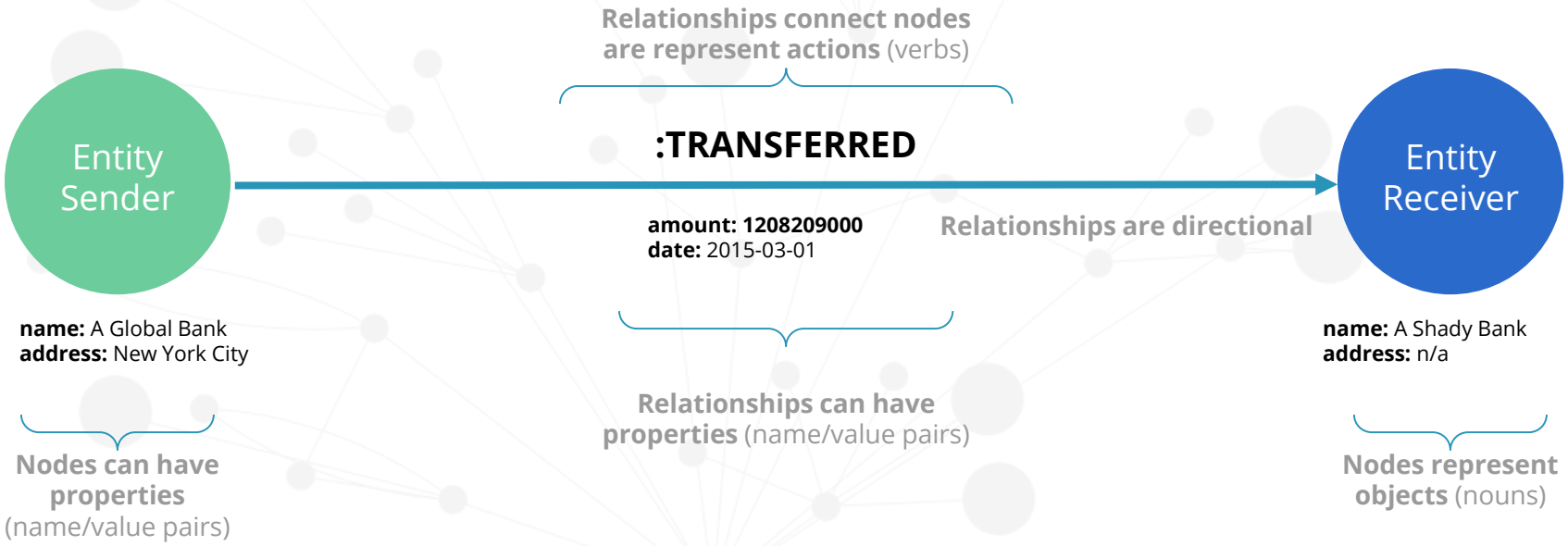
obtained and published **Suspicious Activity Reports** (SARs) submitted by global financial institutions to the

Suspicious Activity Reports

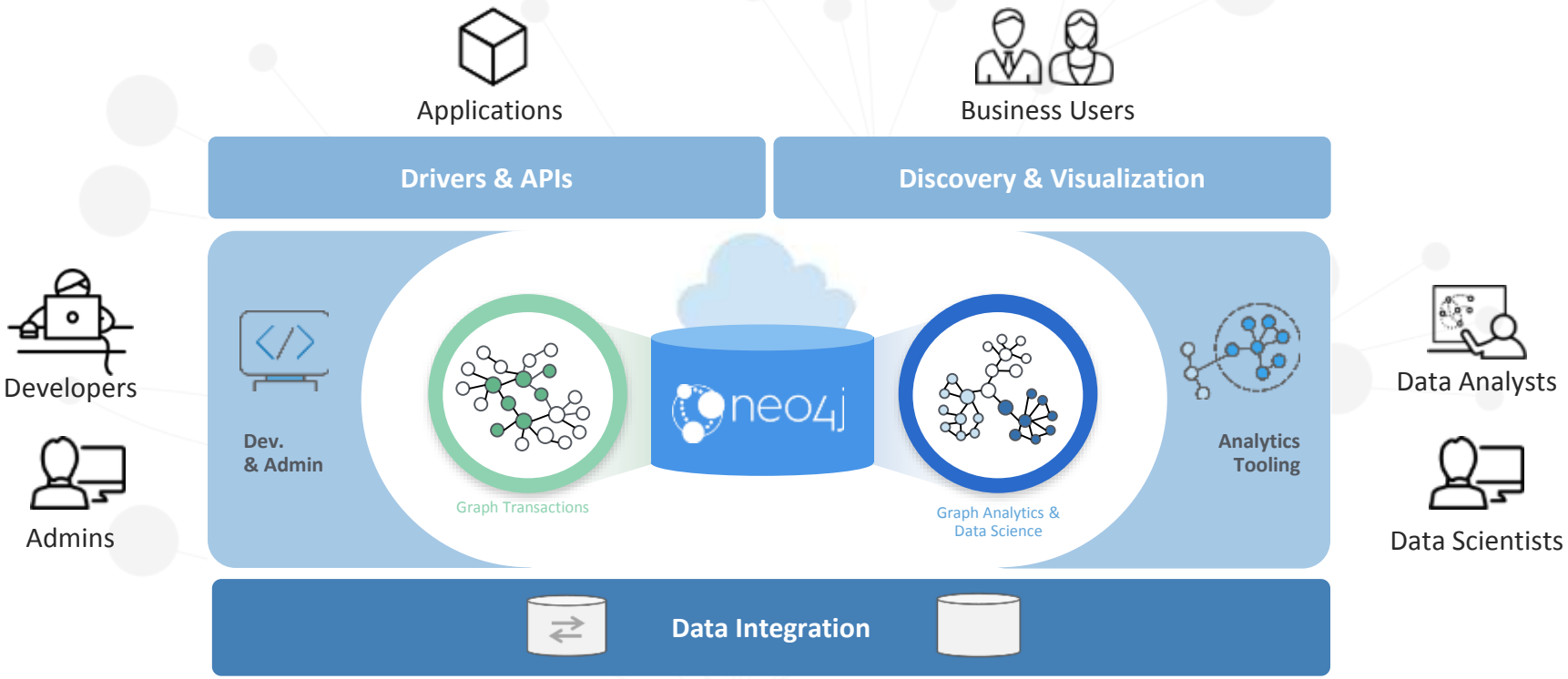
- Insider trading
- **Transactions linked to money laundering**, terrorism financing or other crimes.
- Odd dealings, also involving shell companies
- Transactions by individuals known or suspected to have links to criminal or terrorist organizations, or on sanction lists

These activities are required to be reported WITHIN 30-60 days. In the FinCEN files - this is rarely the case.

Property Graph - Simply Powerful



Native Graph Technology for Applications & Analytics



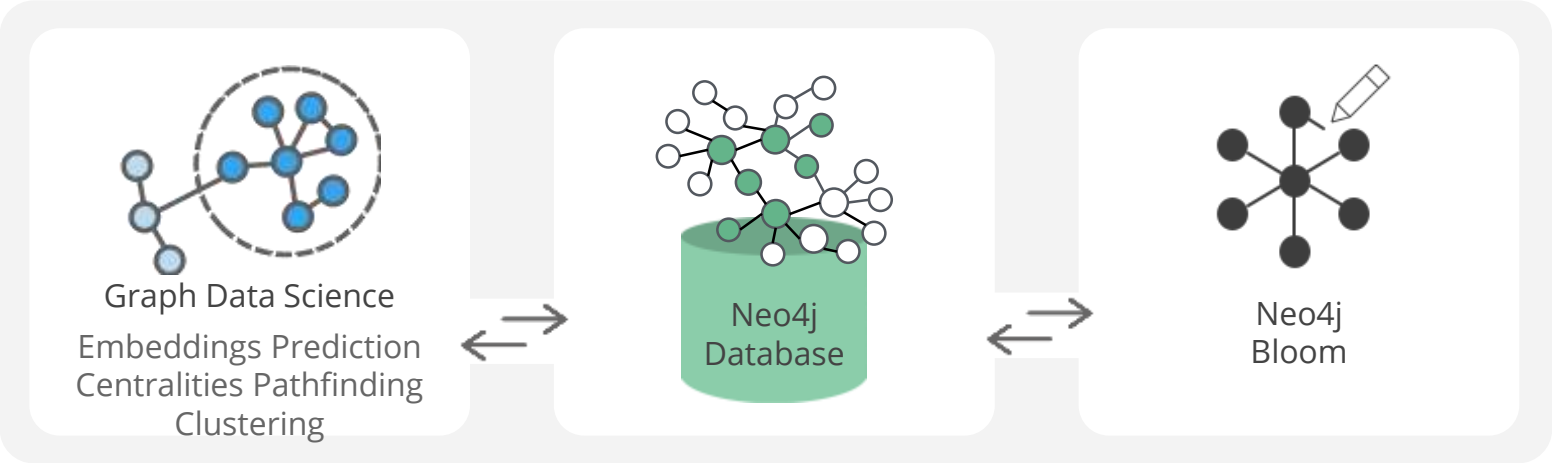
neo4j for Graph Data Science™

neo4j.com/graph-data-science

Scalable Graph Algorithms & Analytics Workspace

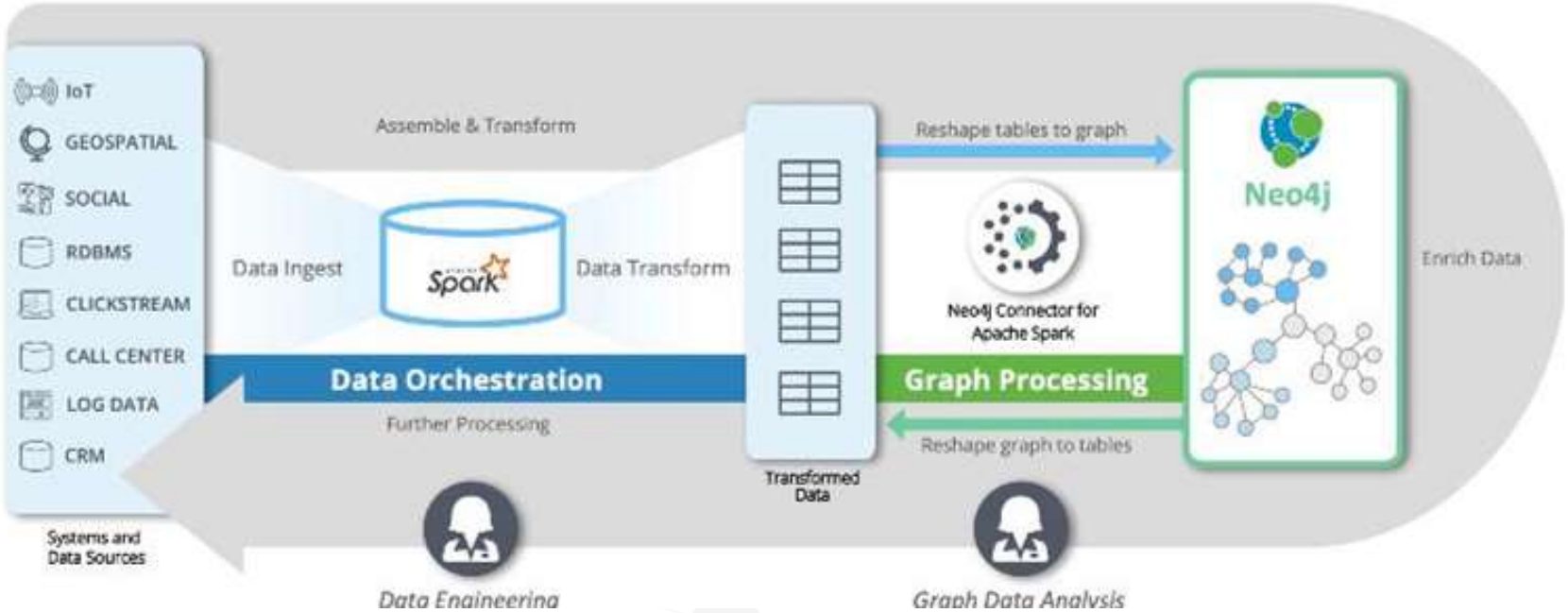
Native Graph Creation & Persistence

Visual Graph Exploration & Prototyping



Practical **Integrated** **Intuitive**

Big Picture


IoT
GEOSPATIAL
SOCIAL
RDBMS
CLICKSTREAM
CALL CENTER
LOG DATA
CRM
Systems and Data Sources
Assemble & Transform
Data Ingest
Spark
Data Transform
Data Orchestration
Further Processing
Transformed Data
Reshape tables to graph
Neo4j Connector for Apache Spark
Graph Processing
Reshape graph to tables
Neo4j
Enrich Data
Data Engineering
Graph Data Analysis

Simplest Examples (Python)

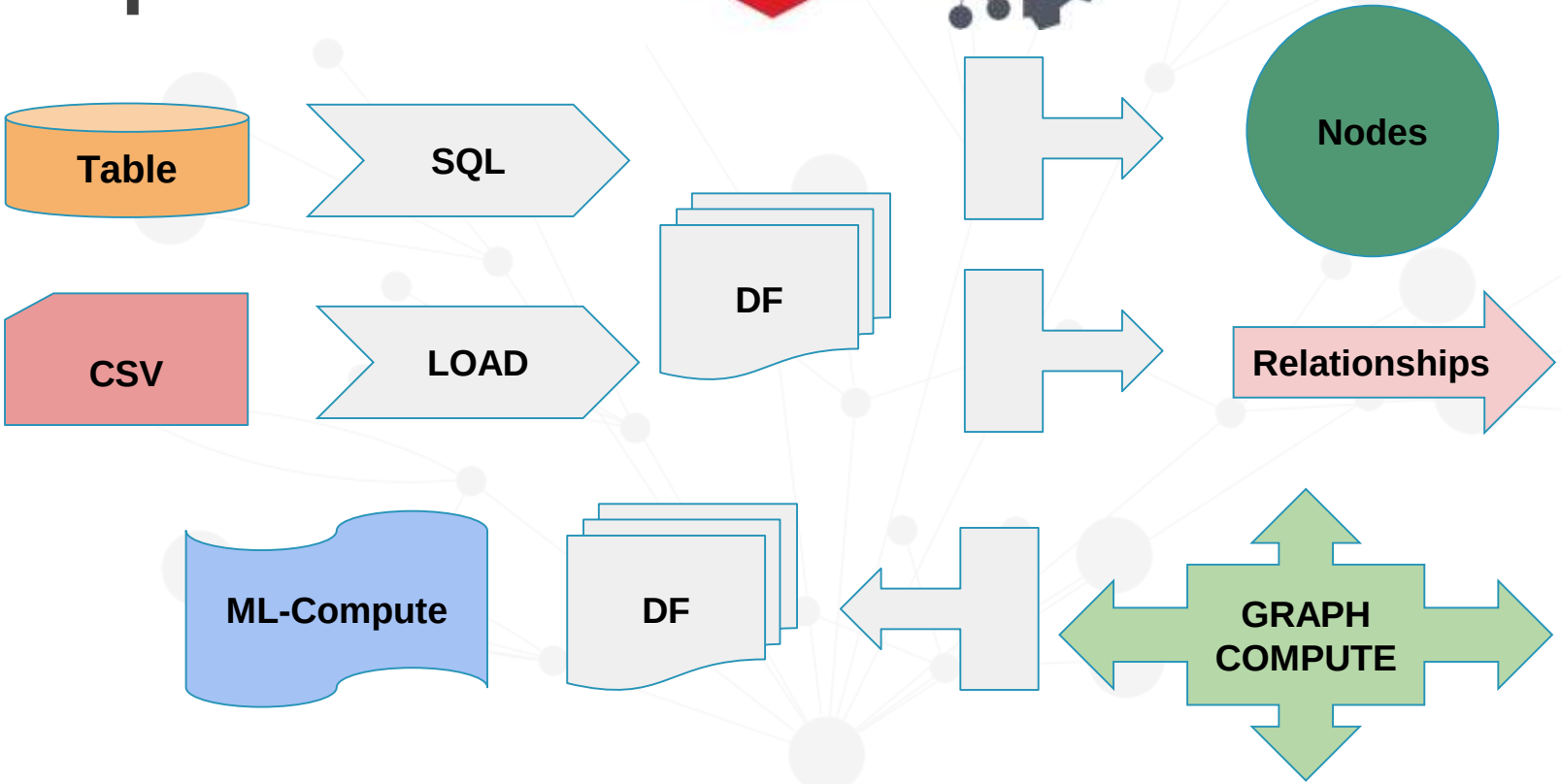
Read

```
spark.read.format("org.neo4j.spark.DataSource")
  .option("url", "bolt://localhost:7687")
  .option("labels", "Entity").load()


# Custom Cypher Query
spark.read.format("org.neo4j.spark.DataSource")
  .option("url", "...")
  .option("query",
          "MATCH (n:Entity) RETURN n.id as id")
  .load().show(10)
```

Write

```
df.write.format("org.neo4j.spark.DataSource")
  .mode("Overwrite")
  .option("url", "...")
  .option("labels", ":Entity").save()

# Custom Cypher Query
df.write.format("org.neo4j.spark.DataSource")
  .option("url", "...")
  .option("query",
          "CREATE (:Entity {id: event.id})")
  .save()
```

Demo Pipeline



Demo

Source CSV Data (Filings)

```python
import urllib.request

file = "download_transactions_map.csv"
url = "https://raw.githubusercontent.com/neo4j-graph-examples/fincen/main/import/" + file

dbutils.fs.put(file, str(urllib.request.urlopen(url).read(), encoding="utf-8"), True)
csv = spark.read.csv(file, header=True, inferSchema=True)
display(csv.take(10))
```

(6) Spark Jobs

csv: pyspark.sql.dataframe.DataFrame = [id: integer, icij_sar_id: integer ... 14 more fields]

	id	icij_sar_id	filer_org_name_id	filer_org_name	begin_date	end_date
1	223254	3297	the-bank-of-new-york-mellon-corp	The Bank of New York Mellon Corp.	Mar 25, 2015	Sep 25, 2015
2	223255	3297	the-bank-of-new-york-mellon-corp	The Bank of New York Mellon Corp.	Mar 30, 2015	Sep 25, 2015
3	223258	2924	the-bank-of-new-york-mellon-corp	The Bank of New York Mellon Corp.	Jul 5, 2012	Jul 5, 2012
4	223259	2924	the-bank-of-new-york-mellon-corp	The Bank of New York Mellon Corp.	Jun 20, 2012	Jun 20, 2012
5	223260	2924	the-bank-of-new-york-mellon-corp	The Bank of New York Mellon Corp.	May 31, 2012	May 31, 2012
6	223261	2924	the-bank-of-new-york-mellon-corp	The Bank of New York Mellon Corp.	May 29, 2012	May 29, 2012
7	223262	2924	the-bank-of-new-york-mellon-corp	The Bank of New York Mellon Corp.	May 29, 2012	May 29, 2012
8	223263	2924	the-bank-of-new-york-mellon-corp	The Bank of New York Mellon Corp.	May 22, 2012	May 22, 2012

Showing all 10 rows.

Save Graph

```
banks = csv.select("originator_bank_id").withColumnRenamed("originator_bank_id","id").union( \
        csv.select("beneficiary_bank_id").withColumnRenamed("beneficiary_bank_id","id")).distinct()

banks.write.format("org.neo4j.spark.DataSource") \
  .mode("Overwrite") \
  .option("url", dbutils.widgets.get("url")
  .option("authentication.basic.username",
  .option("authentication.basic.password",
  .option("node.keys", "id") \
  .option("labels", ":Entity").save()
```

▸ (1) Spark Jobs

▸ banks: pyspark.sql.dataframe.DataFrame = [id: string]

Command took 8.75 seconds -- by michael@neo4j.com at 11

Cmd 5

```
from pyspark.sql import functions as func

transactions = csv.withColumnRenamed("originator_bank_id","source") \
                  .withColumnRenamed("beneficiary_bank_id","target") \
                  .groupBy("source","target").agg(func.sum("amount_transactions")) \
                  .withColumnRenamed("sum(amount_transactions)","amount")

display(transactions.take(10))
```

Show result

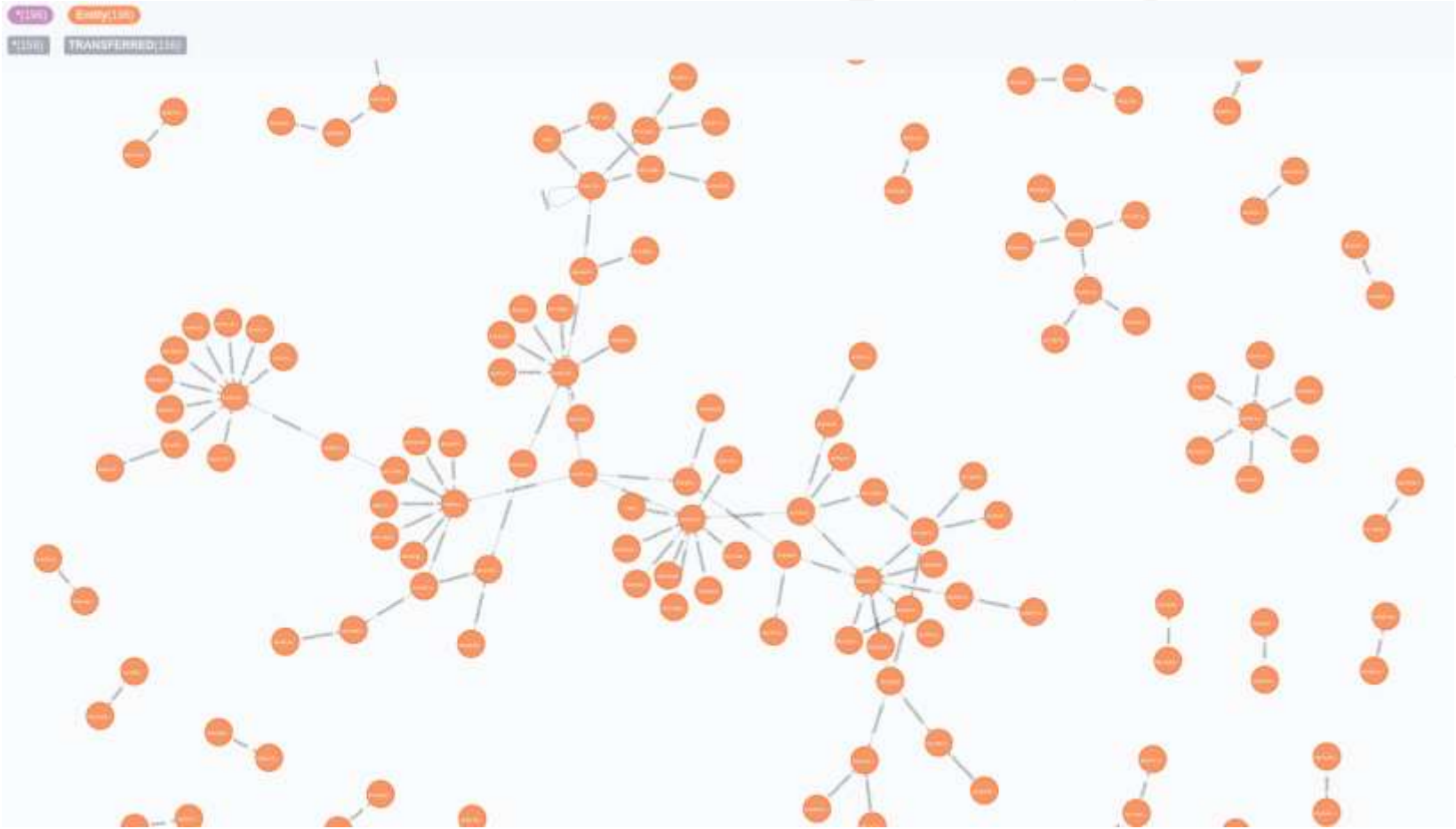
Cmd 6

```
# transactions = spark.sql("select originator_bank_id as source, beneficiary_bank_id as target, sum(amount_transactions)
as amount from download_transactions_map_csv group by source,target")

transactions.repartition(1).write.format("org.neo4j.spark.DataSource") \
.mode("Overwrite") \
.option("url", dbutils.widgets.get("url")) \
.option("authentication.basic.username", dbutils.widgets.get("user")) \
.option("authentication.basic.password", dbutils.widgets.get("password")) \
.option("relationship", "TRANSFERRED") \
.option("relationship.properties","amount") \
.option("relationship.save.strategy", "keys") \
.option("relationship.source.labels", ":Entity") \
.option("relationship.source.node.keys", "source:id") \
.option("relationship.source.save.mode", "Match") \
.option("relationship.target.labels", ":Entity") \
.option("relationship.target.node.keys", "target:id") \
.option("relationship.target.save.mode", "Match").save()
```

▸ (1) Spark Jobs

Command took 11.67 seconds -- by michael@neo4j.com at 11/10/2020, 1:20:40 PM on gdelt2

Show Graph



Check Graph

Cmd 7

```
topReceivers = spark.read.format("org.neo4j.spark.DataSource") \
  .option("url", dbutils.widgets.get("url")) \
  .option("authentication.basic.username", dbutils.widgets.get("user")) \
  .option("authentication.basic.password", dbutils.widgets.get("password")) \
  .option("query", \
  "MATCH (s:Entity)-[tx:TRANSFERRED]->(t:Entity) RETURN s.id, t.id, sum(tx.amount) as total ORDER BY total DESC") \
  .load()

display(topReceivers)
```

▸ (1) Spark Jobs

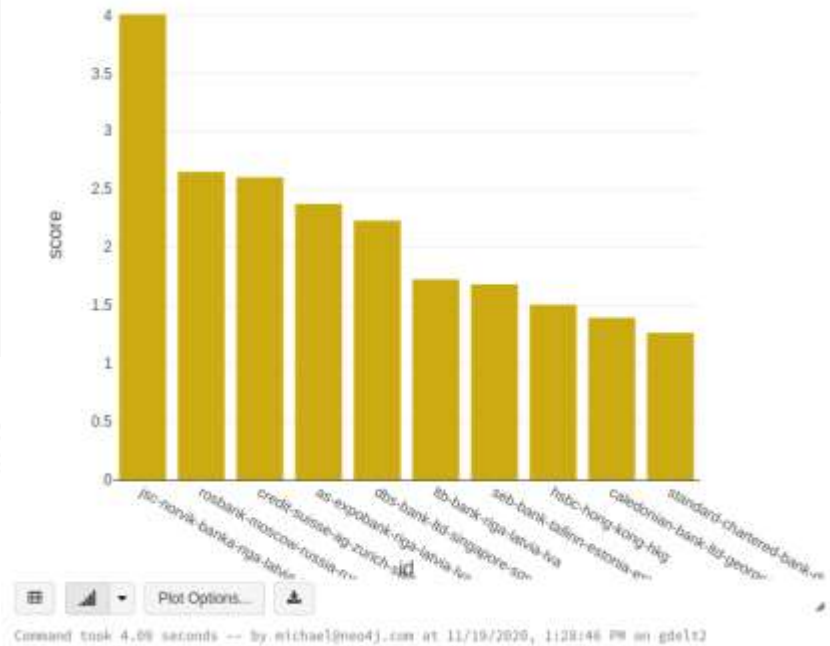
▸ topReceivers: pyspark.sql.dataframe.DataFrame = [s.id: string, total: double ... 1 more fields]

	s.id	total	t.id
1	amsterdam-trade-bank-nv	2747023352.12	rosbank-moscow-russia-rus
2	rigensis-bank-as	1201172347.2800002	ing-netherland-nv-netherlands-nld
3	ing-netherland-nv	1199219653.33	rigensis-bank-as-latvia-lva
4	jpmorgan-chase-bank	1082748112.02	deutsche-bank-ag-london-branch-gbr
5	as-expobank	888040730.6	credit-suisse-ag-zurich-switzerland-che
6	caledonian-bank-ltd	828938267.4300001	hongkong-and-shanghai-banking-corp-hong-kong-hkg
7	gazprombank	802656006.76	jp-morgan-us-usa
8	as-expobank	769834900	bank-sovuz-moscow-russia-rus

Showing the first 1000 rows.

Compute Graph - Centralities

```
pagerank = """
CALL gds.pageRank.stream(
 { nodeProjection: 'Entity',relationshipProjection: 'TRANSFERRED',relationshipProperties: 'amount', relationshipWeightProperty: 'amount' })
YIELD nodeId, score
RETURN gds.util.asNode(nodeId).id AS id, score AS score order by score desc
"""

ranks = spark.read.format("org.neo4j.spark.DataSource") \
  .option("url", dbutils.widgets.get("url")) \
  .option("authentication.basic.username", dbutils.widgets.get("user")) \
  .option("authentication.basic.password", dbutils.widgets.get("password")) \
  .option("query.count", 100) \
  .option("partitions", 1) \
  .option("schema.strategy", "string") \
  .option("query", pagerank) \
  .load().take(10)

display(ranks, ["id","score"])
```



Compute Graph - Clustering/Embeddings

```
clusters = spark.read.format("org.neo4j.spark.DataSource") \
  .option("url", dbutils.widgets.get("url")) \
  .option("authentication.basic.username", dbutils.widgets.get("user")) \
  .option("authentication.basic.password", dbutils.widgets.get("password")) \
  .option("query.count", 100) \
  .option("query", """
CALL gds.louvain.stream(
 { nodeProjection: 'Entity',relationshipProjection: 'TRANSFERRED',relationshipProperties: 'amount', relationshipWeightProperty: 'amount',
YIELD nodeId, intermediateCommunityIds as clusters
RETURN gds.util.asNode(nodeId).id, clusters
  """) \
  .load()

display(clusters.take(10))
```

(4) Spark Jobs

clusters: pyspark.sql.dataframe.DataFrame = [gds.util.asNode(nodeId).id: string, clusters: array]

	gds.util.asNode(nodeId).id	clusters
1	hua-nan-commercial-bank-ltd	[1898, 2086]
2	oversea-chinese-banking-corp	[88, 88]
3	standard-bank-plc-london	[211, 211]
4	harris-na	[1992, 2106]
5	zions-first-national-bank-salt-lake-city-ut-usa	[4, 4]
6	canara-bank-new-delhi-india-ind	[5, 2164]
7	dms-bank-trust-ltd-cayman-islands-cym	[6, 6]
8	petrocommerce-oisc-bank-moscow-russia-rus	[7, 7]

Showing all 10 rows.

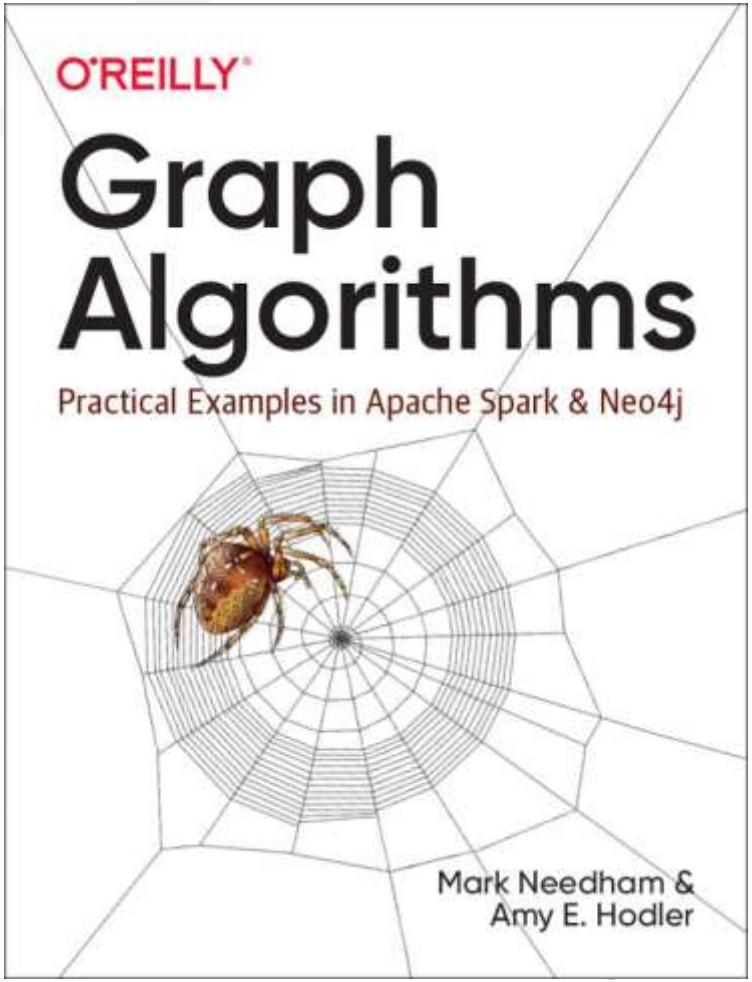
Next:

Graph-Embeddings
KNN Similarity
Train ML-Models

neo4j.com/graph-data-science

Free O'Reilly Book

neo4j.com/books



Connector: neo4j.com/developer/spark

Sandbox: neo4j.com/try-neo4j

Notebook: github.com/jexp/fincen

Thank You!

Questions?

neo4j