An Overview Of Neo4j

And The Property Graph Model

Berkeley, CS294, Nov 2015

Emil Eifrem
emil@neotechnology.com
@emileifrem
#neo4j

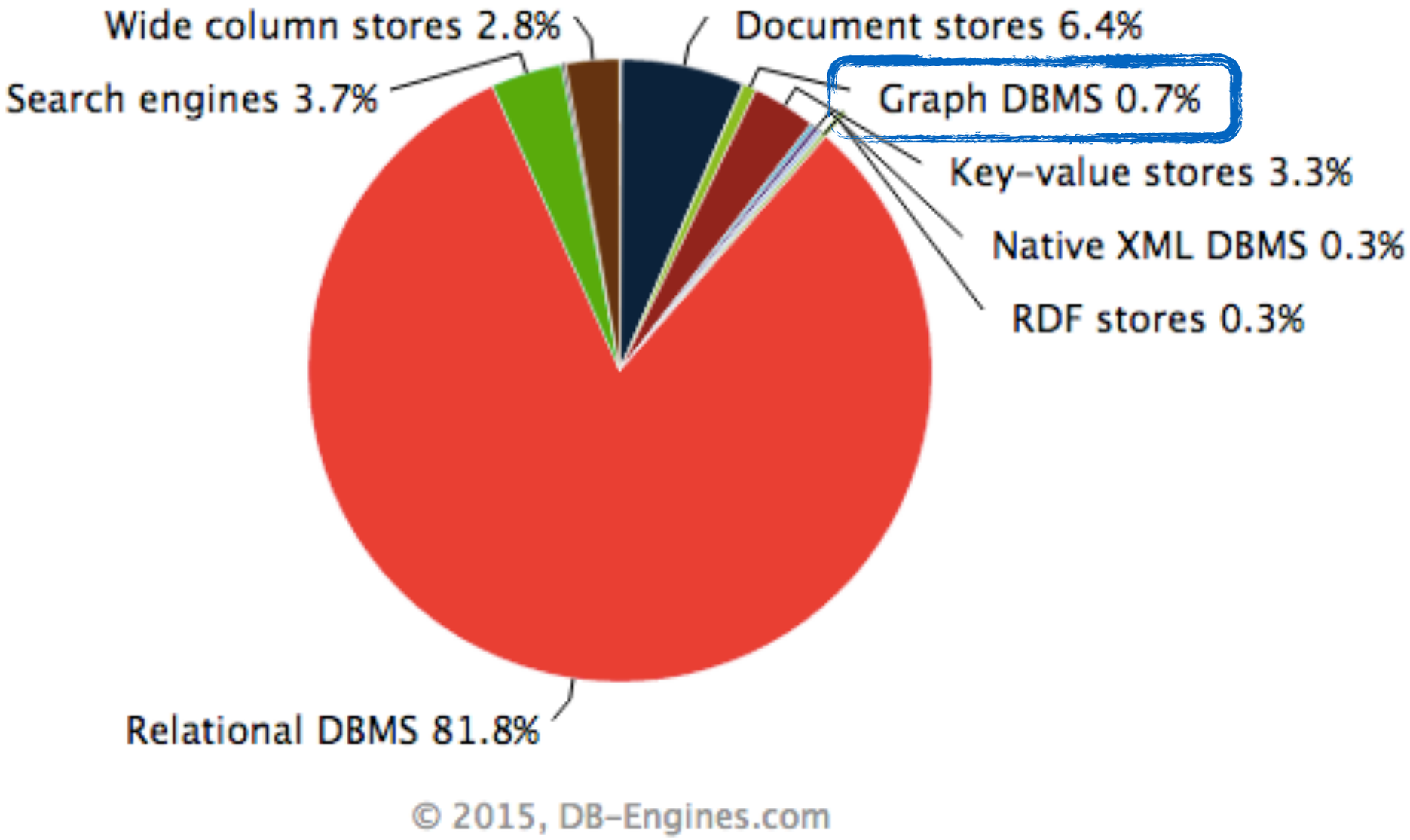
Agenda

1. Why Care About Graphs?
2. The Graph Ecosystem
    - Operational Graph Technologies
    - Analytic Graph Technologies
3. Neo4j: Past & Present

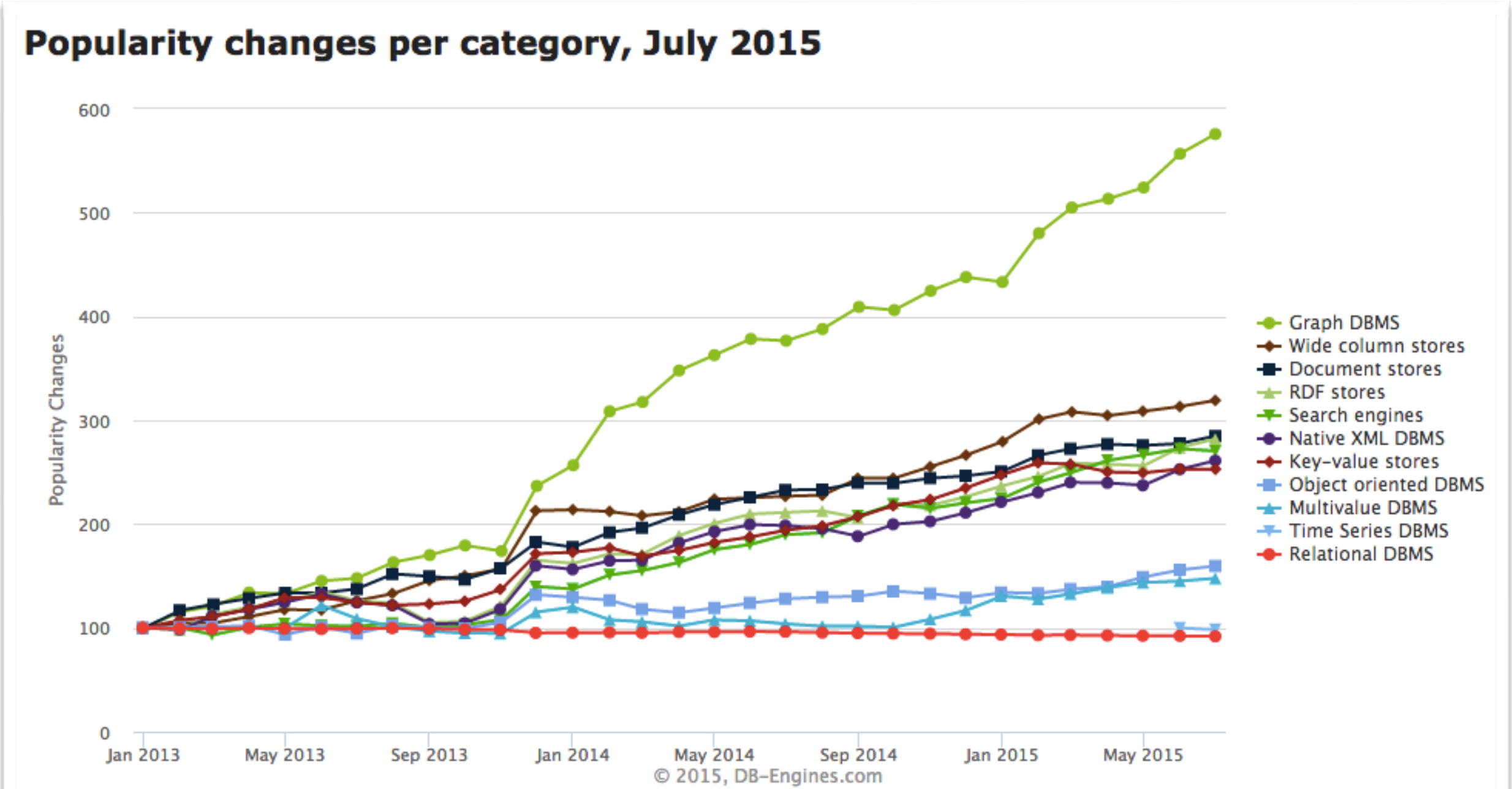
Warning: Vendor Bias

1. Why Care About Graphs?

Reality Check



However…


Popularity changes per category, July 2015
600
500
400
300
200
100
0
Popularity Changes
Jan 2013
May 2013
Sep 2013
Jan 2014
May 2014
Sep 2014
Jan 2015
May 2015
Graph DBMS
Wide column stores
Document stores
RDF stores
Search engines
Native XML DBMS
Key-value stores
Object oriented DBMS
Multivalue DBMS
Time Series DBMS
Relational DBMS
© 2015, DB-Engines.com

What Analysts Say

"Graph analysis is possibly the **single most effective competitive differentiator** for organizations pursuing data-driven operations and decisions after the design of data capture."

Gartner

https://www.gartner.com/doc/2852717/it-market-clock-database-management

What Analysts Say

"Graph analysis is the **true killer app** for Big Data."

"Forrester estimates that **over 25% of enterprises** will be using graph databases by 2017"

http://blogs.forrester.com/james_kobielus/11-12-19-the_year_ahead_in_big_data_big_cool_new_stuff_looms_large
http://www.forrester.com/go?objectid=RES106801

In The Real World

2. The Graph Ecosystem

The Graph Ecosystem:

1. Operational Graph Technologies (“OLTP”)

2. Analytic Graph Technologies (“OLAP”)

Graph Databases

A **graph database management system** is an online ("real-time") database management system with CRUD methods that expose a graph data model[1]

- Graph databases may or may not have:
  - **Native graph processing**, including *index-free adjacency* to facilitate traversals
  - **Native graph storage** engine, i.e. written from the ground up to manage graph data

1] Robinson et al. *Graph Databases*. O'Reilly, 2013. p. 5. ISBN-10: 1449356265

Graph Database Bias

Developers

Applications

Anatomy of a Graph Database Deployment

Application

python nodeJS Ruby Microsoft .net php Java

Language Drivers

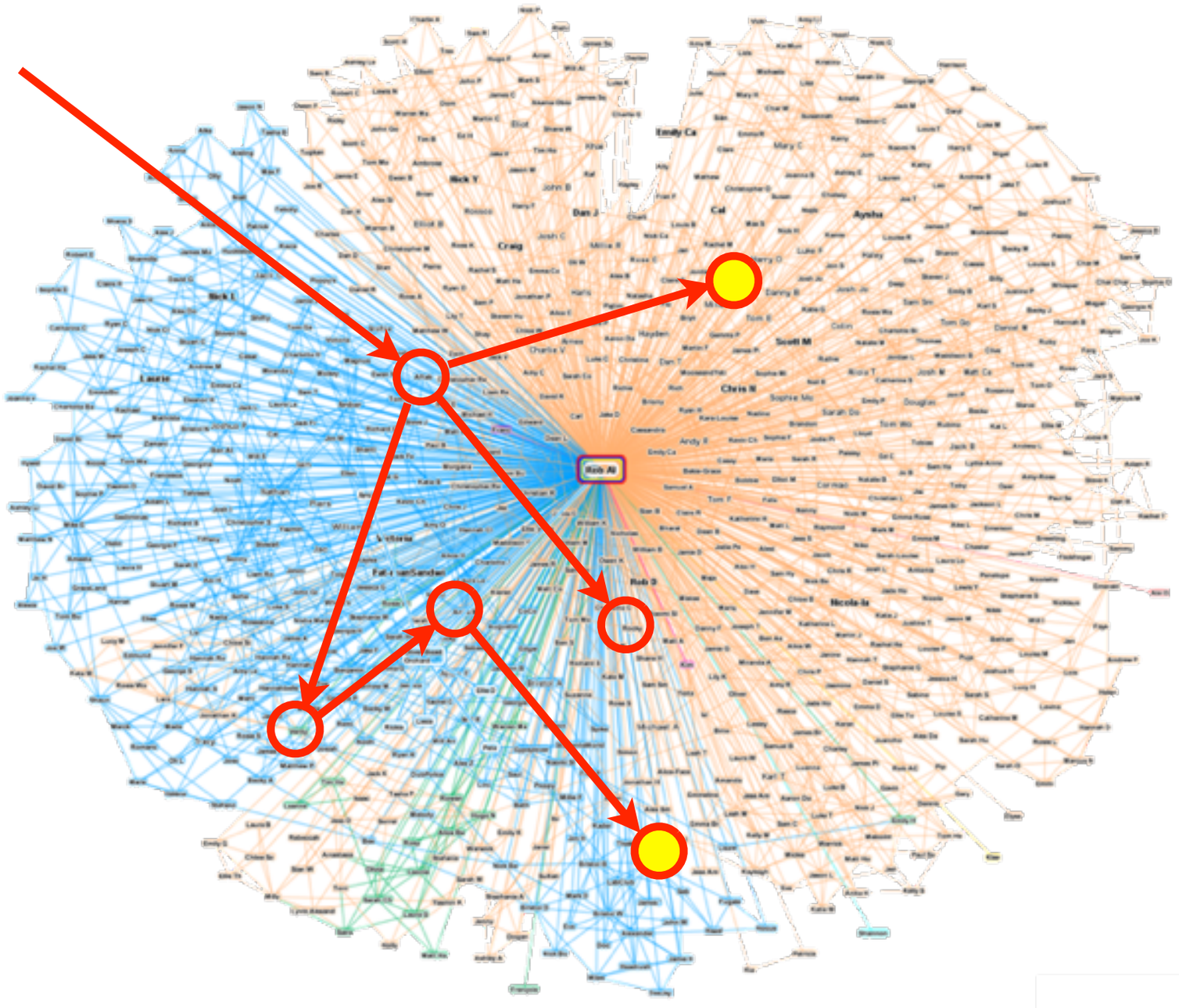
```
MATCH (:Person { name:“Dan”} ) -[:LOVES]-> (:Person { name:“Ann”} )
```

Graph Query Language

Graph DBMS

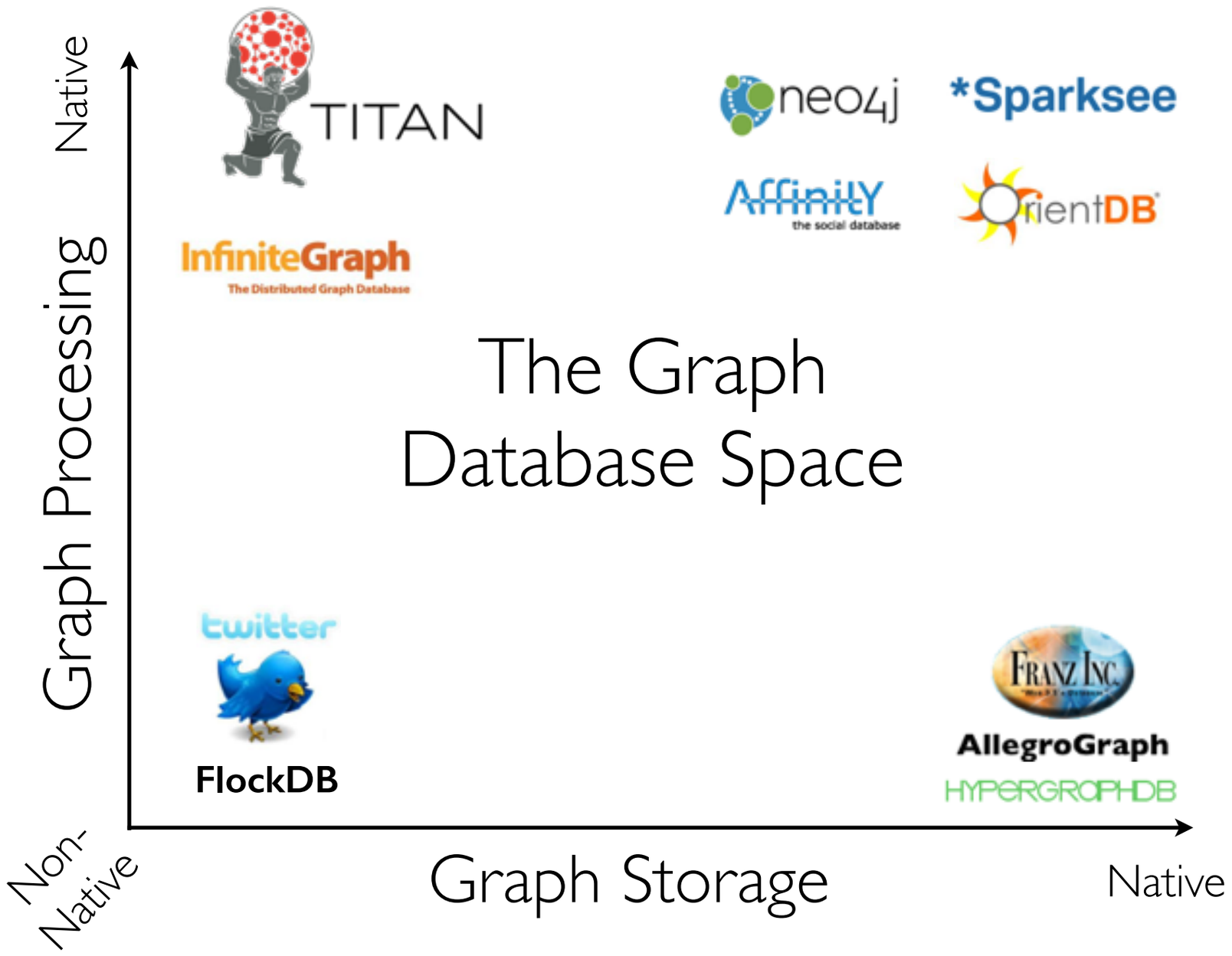
Graph Local Queries

Sweet Spot for Graph Databases



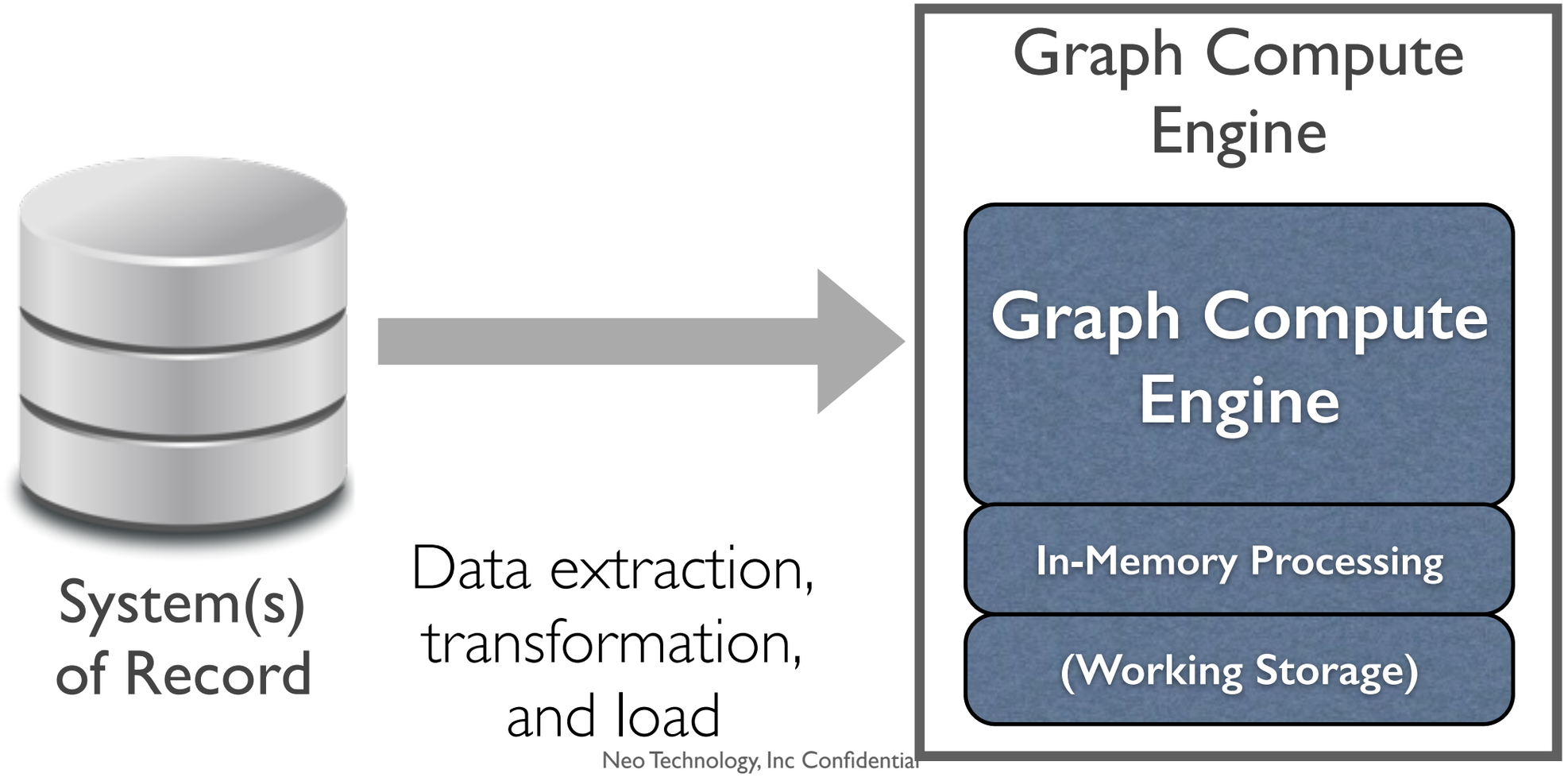
e.g. Recommendations, Friend-of-Friend, Shortest Path, Arbitrary-Depth Queries

The Graph Database Space



Graph Compute Engine

Processing engines that enable graph global computational algorithms to be run against large data sets



Graph Compute Bias

Data scientists

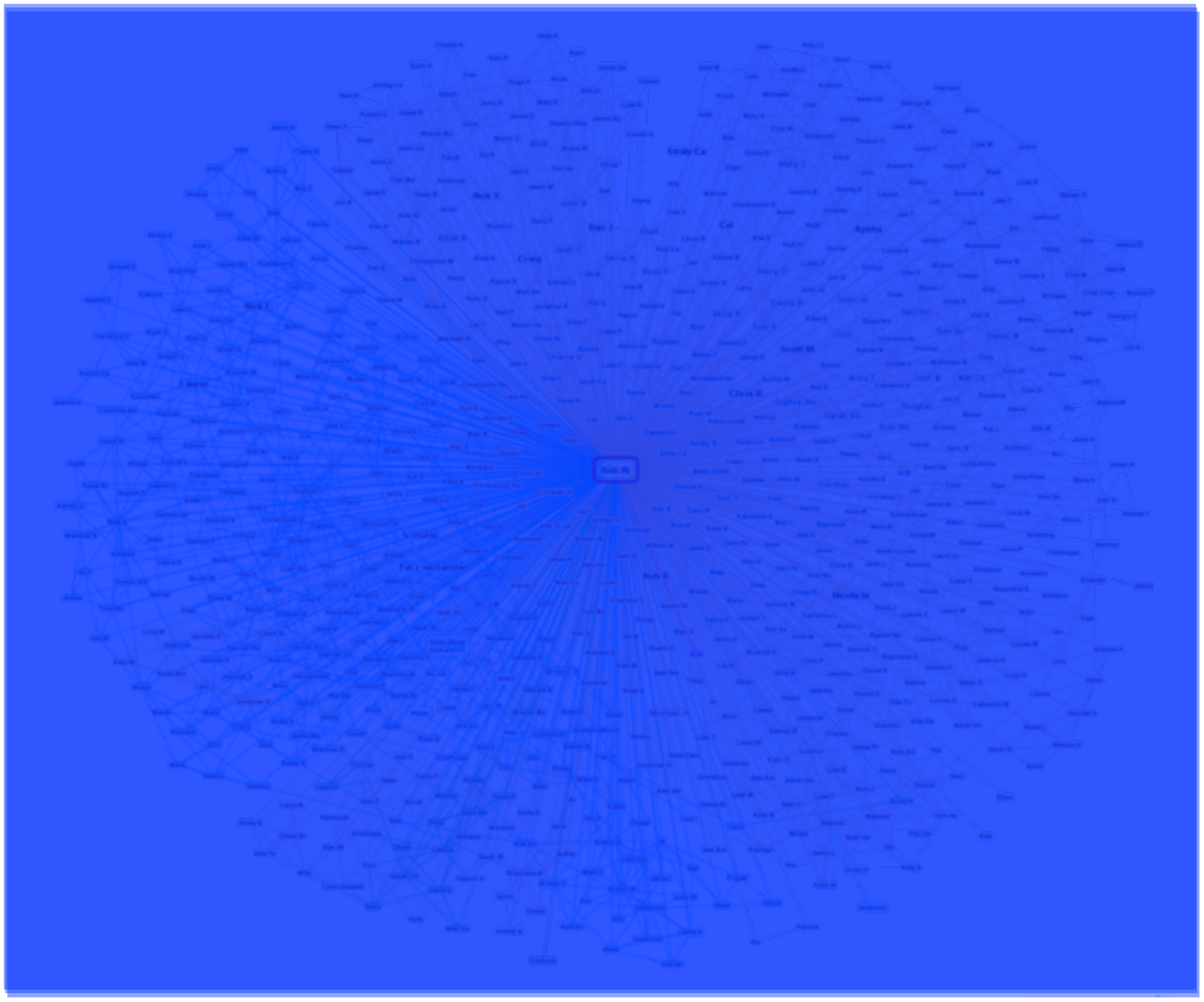
End-user reports

Graph Global Queries

Sweet Spot for Graph Compute Engines

neo technology
graphs are everywhere

How many restaurants, on average, has each person liked?



Graph Compute Engines

Two patterns / sub-categories:

- Single Image - typically in-memory and single machine
- Partitioned - spread across multiple machines, sometimes using the "Bulk Synchronous Parallel Model" from Google Pregel

Graph Compute Engine

Partitioned Examples

- Apache project based on Hadoop
- Bulk Synchronous Processing Model (Pregel Clone)
- Released in 2012

- *"The OSS Project formerly known as GraphLab"*
- Distributes relationships vs. nodes
- Developed at CMU with funding from DARPA, Intel, et al. & VC

- Bundled as part of Spark (first class citizen)
- Well integrated with the rest of the Spark ecosystem (streaming, etc)

Graph Compute Engine

Single Image Examples

Cassovary

- OSS Project led by Twitter
- (No longer!) Used by Twitter for large-scale graph mining (uses daily export from FlockDB system of record)

GraphChi

- GraphLab Spinoff
- Similar order-of-magnitude performance as GraphLab on a Mac Mini

YarcData uRiKA

- Graph compute appliance launched by Cray in Feb 2012
- Built to discover unforeseen relationships in the graph

3. Neo4j: Past & Present

<demo/>

Example HR Query (using SQL)

*"*Find all direct reports and how many they manage, up to 3 levels down*"

Example HR Query (using SQL)

```
 (SELECT T.directReportees AS directReportees, sum(T.count) AS count
FROM (
SELECT manager.pid AS directReportees, 0 AS count
   FROM person_reportee manager
   WHERE manager.pid = (SELECT id FROM person WHERE name = "fName lName")
UNION
   SELECT manager.pid AS directReportees, count(manager.directly_manages) AS count
FROM person_reportee manager
WHERE manager.pid = (SELECT id FROM person WHERE name = "fName lName")
GROUP BY directReportees
UNION
SELECT manager.pid AS directReportees, count(reportee.directly_manages) AS count
FROM person_reportee manager
JOIN person_reportee reportee
ON manager.directly_manages = reportee.pid
WHERE manager.pid = (SELECT id FROM person WHERE name = "fName lName")
GROUP BY directReportees
UNION
SELECT manager.pid AS directReportees, count(L2Reportees.directly_manages) AS count
FROM person_reportee manager
JOIN person_reportee L1Reportees
ON manager.directly_manages = L1Reportees.pid
JOIN person_reportee L2Reportees
ON L1Reportees.directly_manages = L2Reportees.pid
WHERE manager.pid = (SELECT id FROM person WHERE name = "fName lName")
GROUP BY directReportees
) AS T
GROUP BY directReportees)
UNION
(SELECT T.directReportees AS directReportees, sum(T.count) AS count
FROM (
SELECT manager.directly_manages AS directReportees, 0 AS count
FROM person_reportee manager
WHERE manager.pid = (SELECT id FROM person WHERE name = "fName lName")
UNION
SELECT reportee.pid AS directReportees, count(reportee.directly_manages) AS count
FROM person_reportee manager
JOIN person_reportee reportee
ON manager.directly_manages = reportee.pid
WHERE manager.pid = (SELECT id FROM person WHERE name = "fName lName")
GROUP BY directReportees
UNION
```

(continued from previous page...)

```
SELECT depth1Reportees.pid AS directReportees,
count(depth2Reportees.directly_manages) AS count
FROM person_reportee manager
JOIN person_reportee L1Reportees
ON manager.directly_manages = L1Reportees.pid
JOIN person_reportee L2Reportees
ON L1Reportees.directly_manages = L2Reportees.pid
WHERE manager.pid = (SELECT id FROM person WHERE name = "fName lName")
GROUP BY directReportees
) AS T
GROUP BY directReportees)
UNION
(SELECT T.directReportees AS directReportees, sum(T.count) AS count
   FROM(
   SELECT reportee.directly_manages AS directReportees, 0 AS count
FROM person_reportee manager
JOIN person_reportee reportee
ON manager.directly_manages = reportee.pid
WHERE manager.pid = (SELECT id FROM person WHERE name = "fName lName")
GROUP BY directReportees
UNION
SELECT L2Reportees.pid AS directReportees, count(L2Reportees.directly_manages) AS
count
FROM person_reportee manager
JOIN person_reportee L1Reportees
ON manager.directly_manages = L1Reportees.pid
JOIN person_reportee L2Reportees
ON L1Reportees.directly_manages = L2Reportees.pid
WHERE manager.pid = (SELECT id FROM person WHERE name = "fName lName")
GROUP BY directReportees
) AS T
GROUP BY directReportees)
UNION
(SELECT L2Reportees.directly_manages AS directReportees, 0 AS count
FROM person_reportee manager
JOIN person_reportee L1Reportees
ON manager.directly_manages = L1Reportees.pid
JOIN person_reportee L2Reportees
ON L1Reportees.directly_manages = L2Reportees.pid
WHERE manager.pid = (SELECT id FROM person WHERE name = "fName lName")
)
```

*“*Find all direct reports and how many they manage, up to 3 levels down*”

Same Query in Cypher

```
MATCH (boss)-[:MANAGES*0..3]->(sub),
      (sub)-[:MANAGES*1..3]->(report)
WHERE boss.name = “John Doe”
RETURN sub.name AS Subordinate, count(report) AS Total
```

*“Find all direct reports and how many they manage, up to 3 levels down”

Real World(tm) Performance

"Our Neo4j solution is literally **thousands of times**

faster than the prior MySQL solution,

with queries that require **10-100 times less code**."

- Volker Pacher, Senior Developer eBay

openCypher

- open implementation of Cypher
  - announced two weeks ago
  - supported by Oracle, Neo4j (& there's even an AMPLab project!)

http://opencypher.org

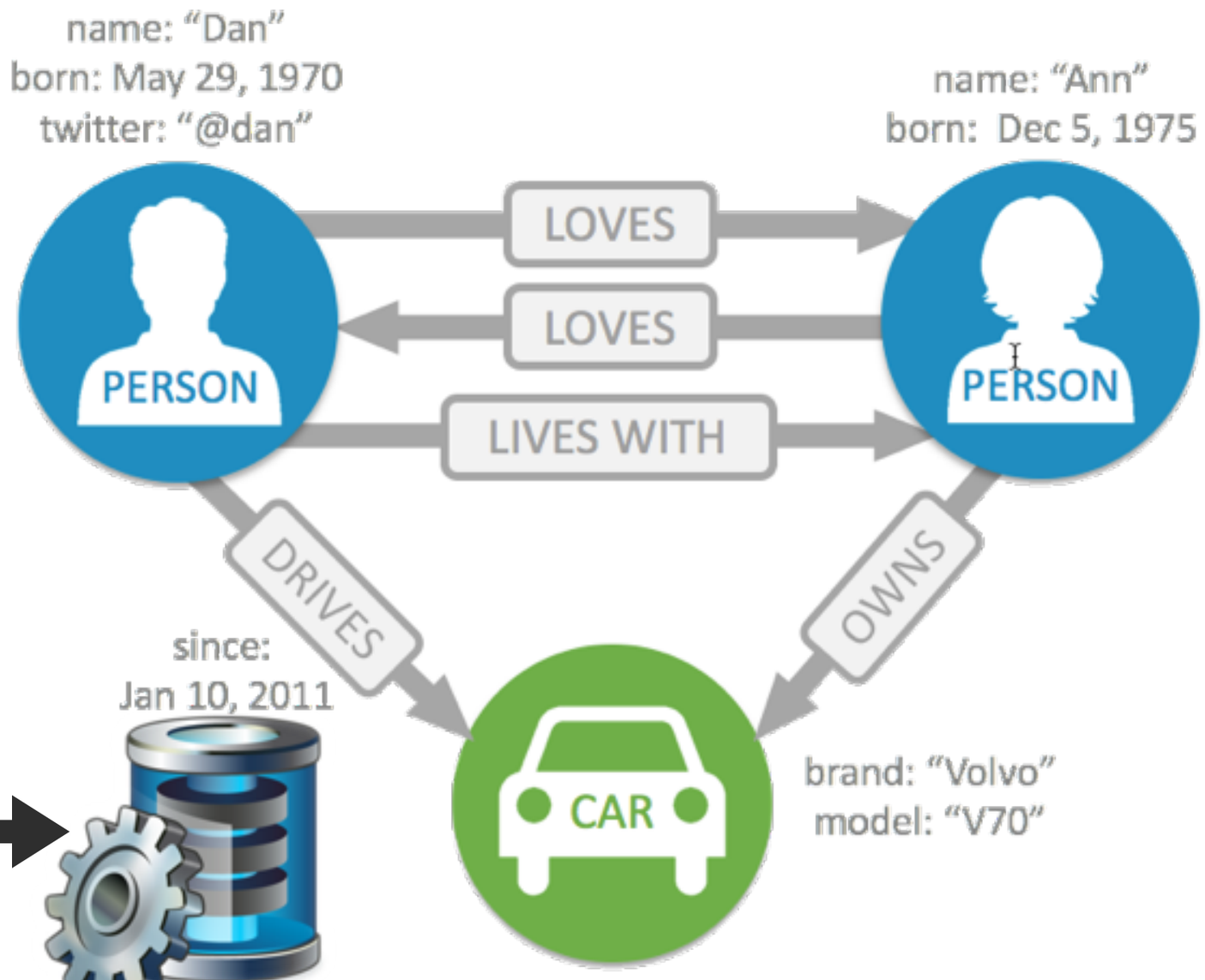
teh end (sic)

stay connected

Appendix

Popular Graph Models

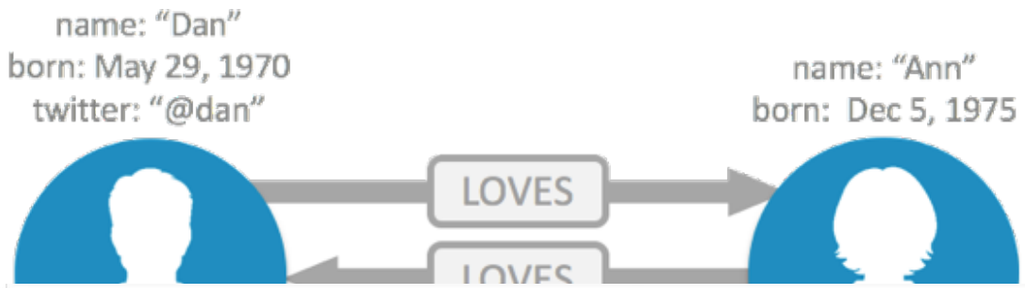
Property Graph



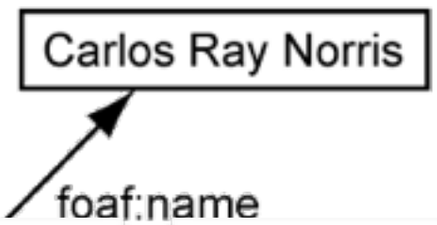
Enterprise Applications

Popular Graph Models

Property Graph

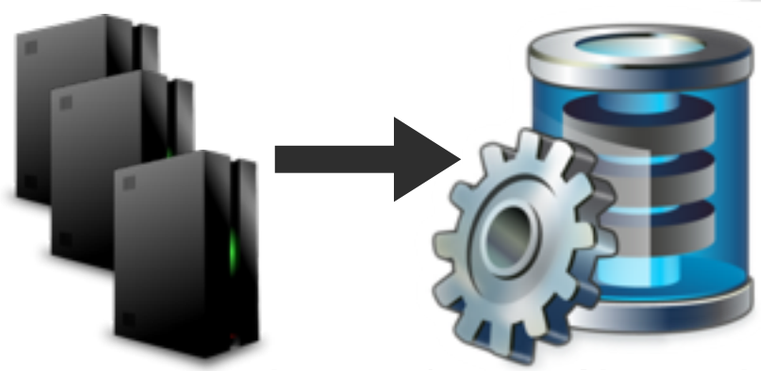


RDF



Rank Jul 2015	Rank Jun 2015	Rank Jul 2014	DBMS	Database Model	Score Jul 2015	Score Jun 2015	Score Jul 2014
1.	1.	1.	Neo4j	Graph DBMS	31.34	+0.84	+9.10
2.	2.	↑ 3.	OrientDB	Multi-model	4.46	+0.23	+2.76
3.	3.	↓ 2.	Titan	Graph DBMS	3.89	+0.24	+1.88
4.	4.	↑ 6.	ArangoDB	Multi-model	1.29	+0.09	+1.06
5.	5.	5.	Giraph	Graph DBMS	1.03	+0.11	+0.63

Rank Jul 2015	Rank Jun 2015	Rank Jul 2014	DBMS	Database Model	Score Jul 2015	Score Jun 2015	Score Jul 2014
1.	1.	1.	MarkLogic	Multi-model	11.13	+0.36	+2.93
2.	2.	2.	Virtuoso	Multi-model	3.06	+0.05	+0.67
3.	3.	↑ 4.	Jena	RDF store	1.71	+0.06	+0.50
4.	4.	↓ 3.	Sesame	RDF store	1.66	+0.09	+0.34
5.	5.	5.	AllegroGraph	RDF store	0.74	-0.03	+0.18

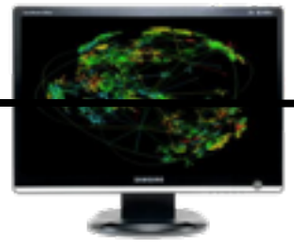


Enterprise Applications

Example Graph Database Deployment

neotechnology
graphs are everywhere

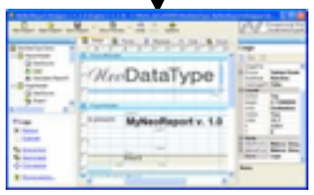
End User



Ad-hoc visual navigation & discovery

Graph Visualization

Graph-Dashboards & Ad-hoc Analysis



Reporting

Application

Other Databases

Bulk Analytic Infrastructure

(e.g. Graph Compute Engine)

ETL

ETL

Graph Database Cluster

Graph Mining & Aggregation

Ad-Hoc Analysis

Data Storage & Business Rules Execution

Data Scientist