CC5212-1

Procesamiento Masivo de Datos

Otoño 2018

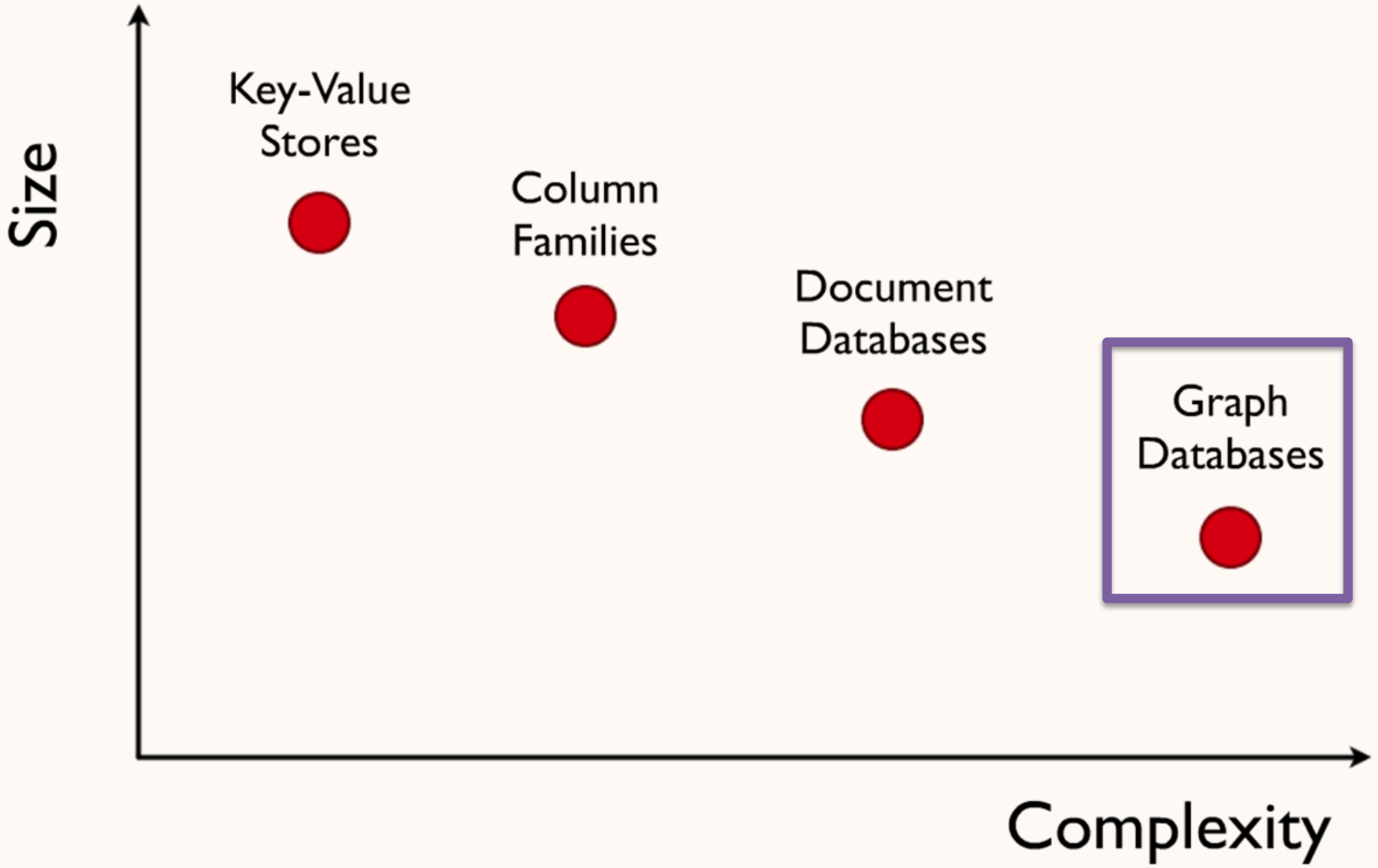
Lecture 10

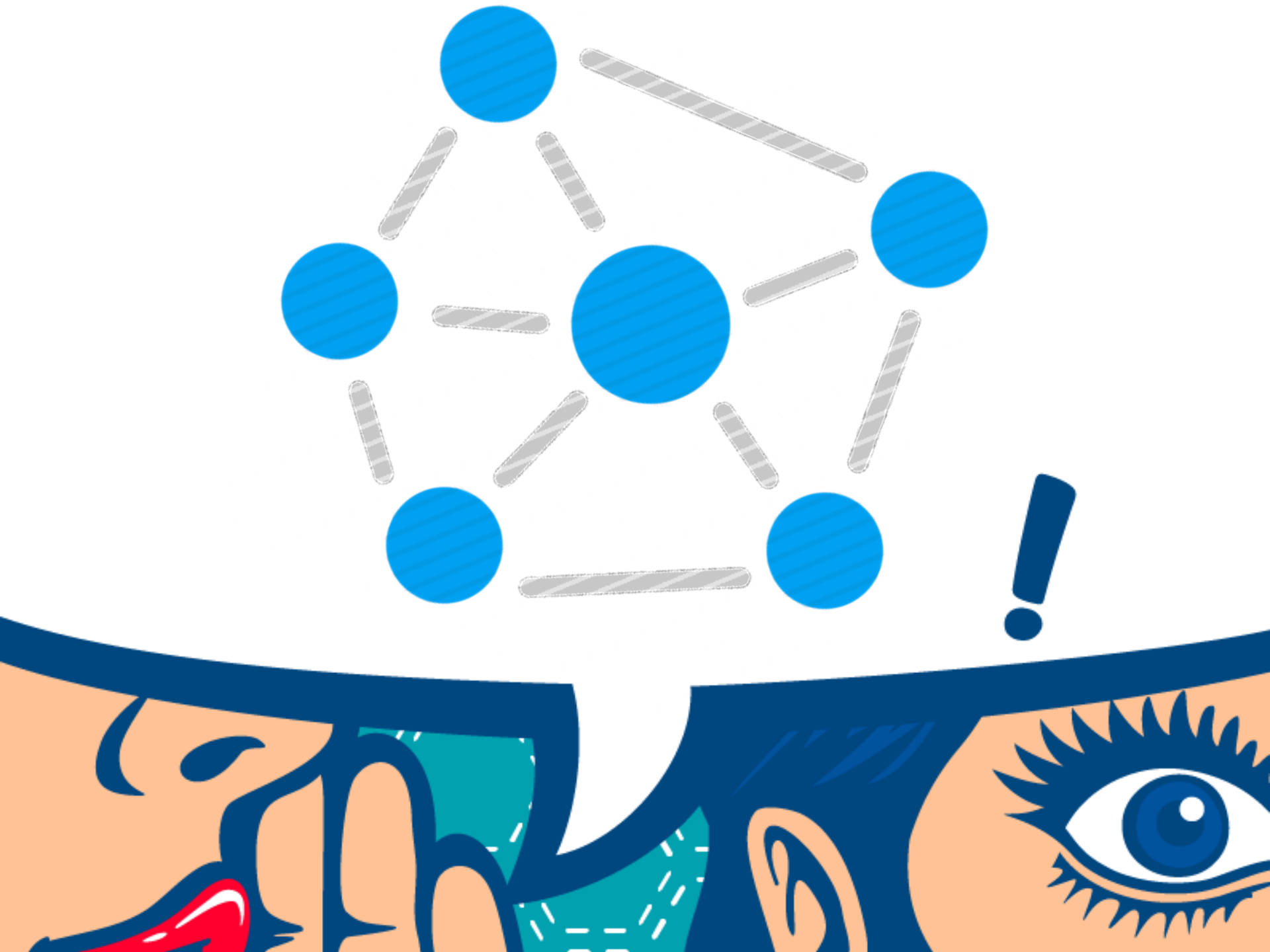
NoSQL: Neo4J

Aidan Hogan

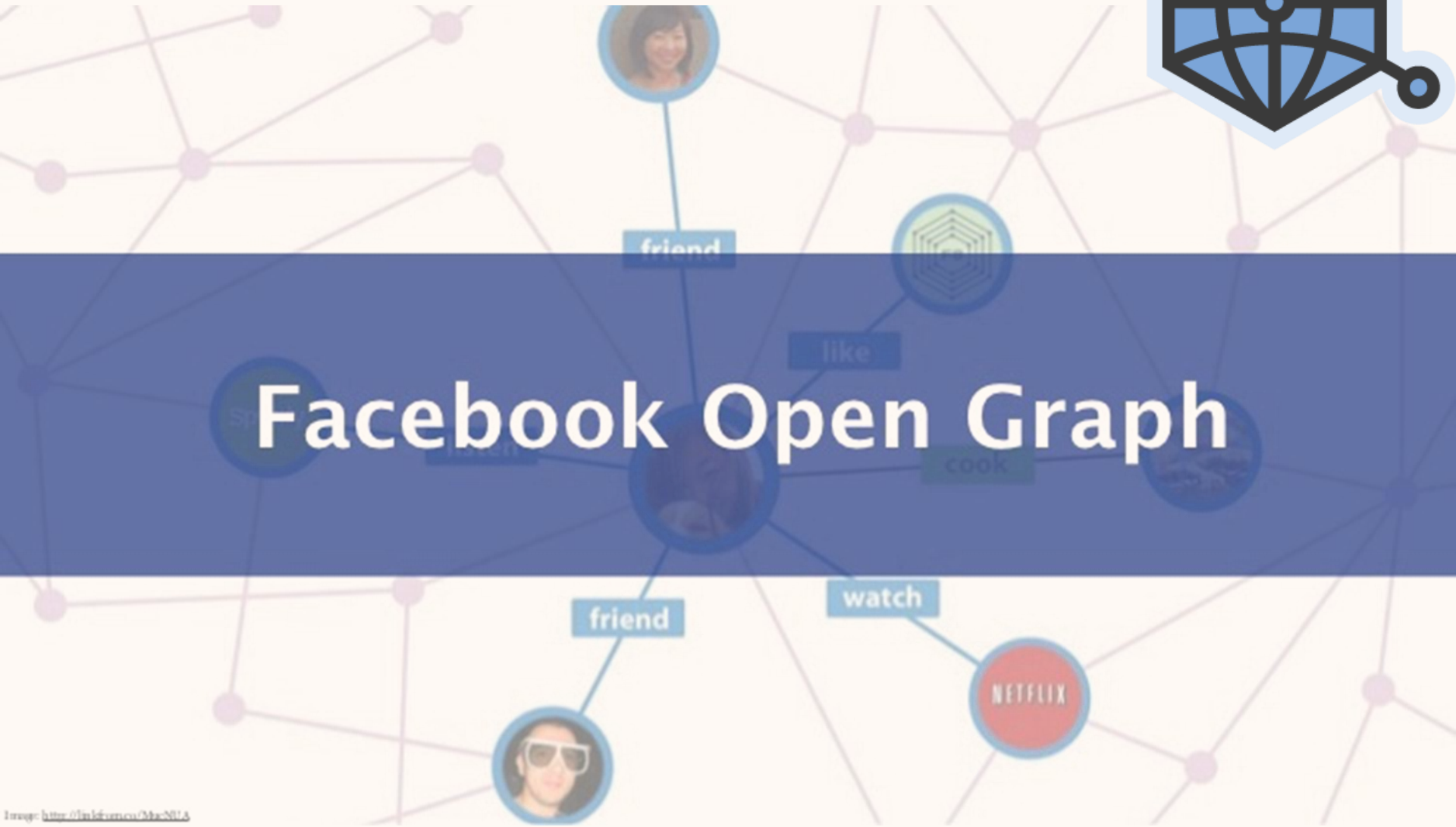
aidhog@gmail.com

NoSQL

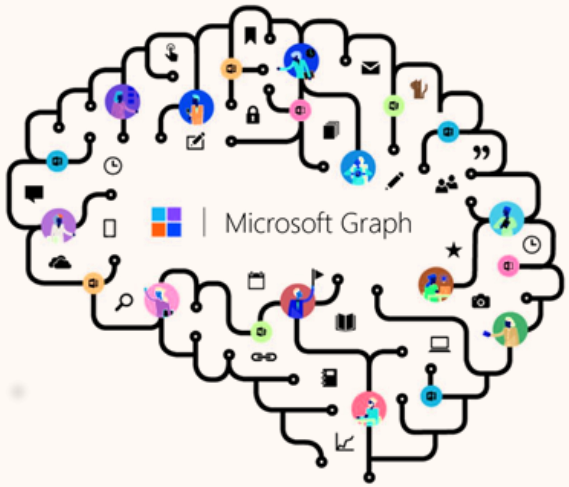

Size
Key-Value Stores
Column Families
Document Databases
Graph Databases
Complexity




Google
Knowledge
Graph

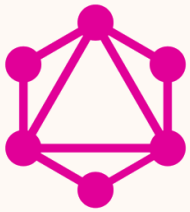


Facebook Open Graph



graph.microsoft.com

Thinking in Graphs



GraphQL

It's Graphs All the Way Down *

> With GraphQL, you model your business domain as a graph

Graphs are powerful tools for modeling many real-world phenomena because they resemble our natural mental models and verbal descriptions of the underlying process. With GraphQL, you model your business domain as a graph by defining a schema; within your schema, you define different types of nodes and how they connect/relate to one another. On the client, this creates a pattern similar to Object-Oriented Programming: types that reference other types. On the server, since GraphQL only defines the interface, you have the freedom to use it with any backend (new or legacy!).

Shared Language

> Naming things is a hard but important part of building intuitive APIs

Think of your GraphQL schema as an expressive shared language for your team and your users. To build a good schema, examine the everyday language you use to describe your business. For example, let's try to describe an email app in plain english:

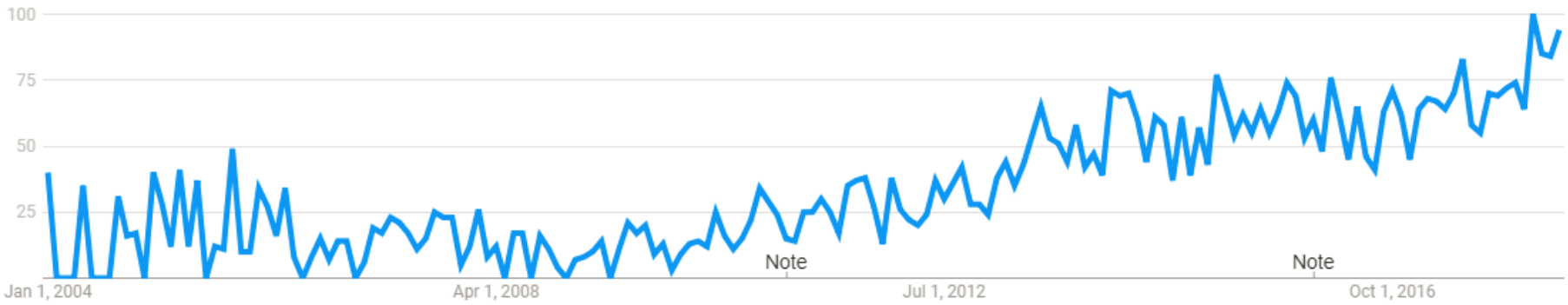

Amazon Neptune
Fast, reliable graph database built for the cloud
Get started with Amazon Neptune

Graph database
Topic

+ Compare

United States | 2004 - present | All categories | Web Search

Interest over time



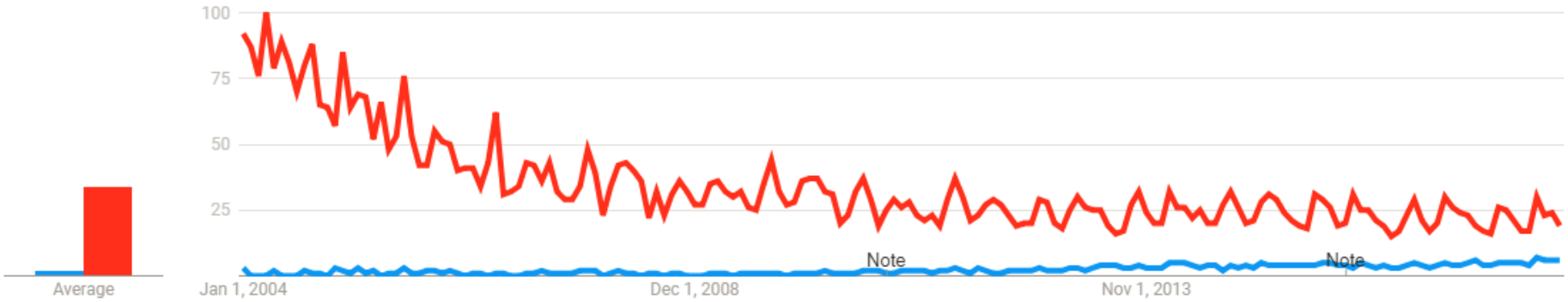
Graph database
Topic

Relational database
Topic

+ Add comparison

United States | 2004 - present | All categories | Web Search

Interest over time

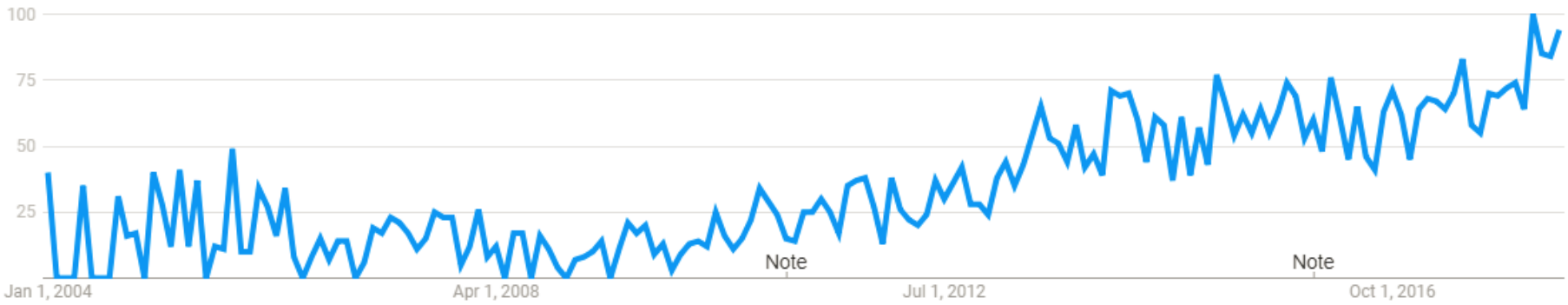


Graph database
Topic

+ Compare

United States | 2004 - present | All categories | Web Search

Interest over time



343 systems in ranking, July 2018

Rank Jul 2018	Rank Jun 2018	Rank Jul 2017	DBMS	Database Model	Score Jul 2018	Score Jun 2018	Score Jul 2017
1.	1.	1.	Oracle	Relational DBMS	1277.79	-33.47	-97.09
2.	2.	2.	MySQL	Relational DBMS	1196.07	-37.62	-153.04
3.	3.	3.	Microsoft SQL Server	Relational DBMS	1053.41	-34.32	-172.59
4.	4.	4.	PostgreSQL	Relational DBMS	405.81	-4.86	+36.37
5.	5.	5.	MongoDB	Document store	350.33	+6.54	+17.56
6.	6.	6.	DB2	Relational DBMS	186.20	+0.56	-5.05
7.	7.	↑ 9.	Redis	Key-value store	139.91	+3.61	+18.40
8.	8.	↑ 10.	Elasticsearch	Search engine	136.22	+5.18	+20.25
9.	9.	↓ 7.	Microsoft Access	Relational DBMS	132.58	+1.59	+6.45
10.	10.	↓ 8.	Cassandra	Wide column store	121.06	+1.84	-3.07
11.	11.	11.	SQLite	Relational DBMS	115.28	+1.02	+1.41
12.	12.	12.	Teradata	Relational DBMS	78.22	+2.45	-0.14
13.	↑ 14.	↑ 16.	Splunk	Search engine	69.24	+3.46	+8.94
14.	↓ 13.	↑ 18.	MariaDB	Relational DBMS	67.51	+1.67	+13.15
15.	↑ 16.	↓ 13.	SAP Adaptive Server	Relational DBMS	62.12	+0.64	-4.79
16.	↓ 15.	↓ 14.	Solr	Search engine	61.52	-0.55	-4.51
17.	17.	↓ 15.	HBase	Wide column store	60.77	+1.07	-2.85
18.	18.	↑ 20.	Hive	Relational DBMS	57.63	+0.30	+11.42
19.	19.	↓ 17.	FileMaker	Relational DBMS	56.39	+0.21	-2.26
20.	20.	↓ 19.	SAP HANA	Relational DBMS	51.60	+2.25	+3.65
21.	21.	↑ 22.	Amazon DynamoDB	Multi-model	49.63	+3.84	+13.17
22.	22.	↓ 21.	Neo4j	Graph DBMS	41.88	-0.09	+3.36
23.	23.	↑ 24.	Memcached	Key-value store	33.88	+0.07	+5.35
24.	24.	↓ 23.	Couchbase	Document store	33.07	+0.62	+0.06
25.	↑ 26.	↑ 26.	Microsoft Azure SQL Database	Relational DBMS	26.84	+0.55	+4.55
26.	↓ 25.	↓ 25.	Informix	Relational DBMS	26.59	+0.03	-1.08
27.	27.	↑ 28.	Vertica	Relational DBMS	20.82	-0.34	-0.97
28.	28.	↑ 30.	Firebird	Relational DBMS	20.66	+0.27	+1.67
29.	29.	↓ 27.	CouchDB	Document store	19.50	-0.70	-2.65
30.	30.	↑ 41.	Microsoft Azure Cosmos DB	Multi-model	19.45	+0.25	+11.74

http://db-engines.com/en/ranking

DB-Engines Ranking of Graph DBMS

The DB-Engines Ranking ranks database management systems according to their popularity. The ranking is updated monthly.

This is a partial list of the complete ranking showing only graph DBMS.

Read more about the method of calculating the scores.

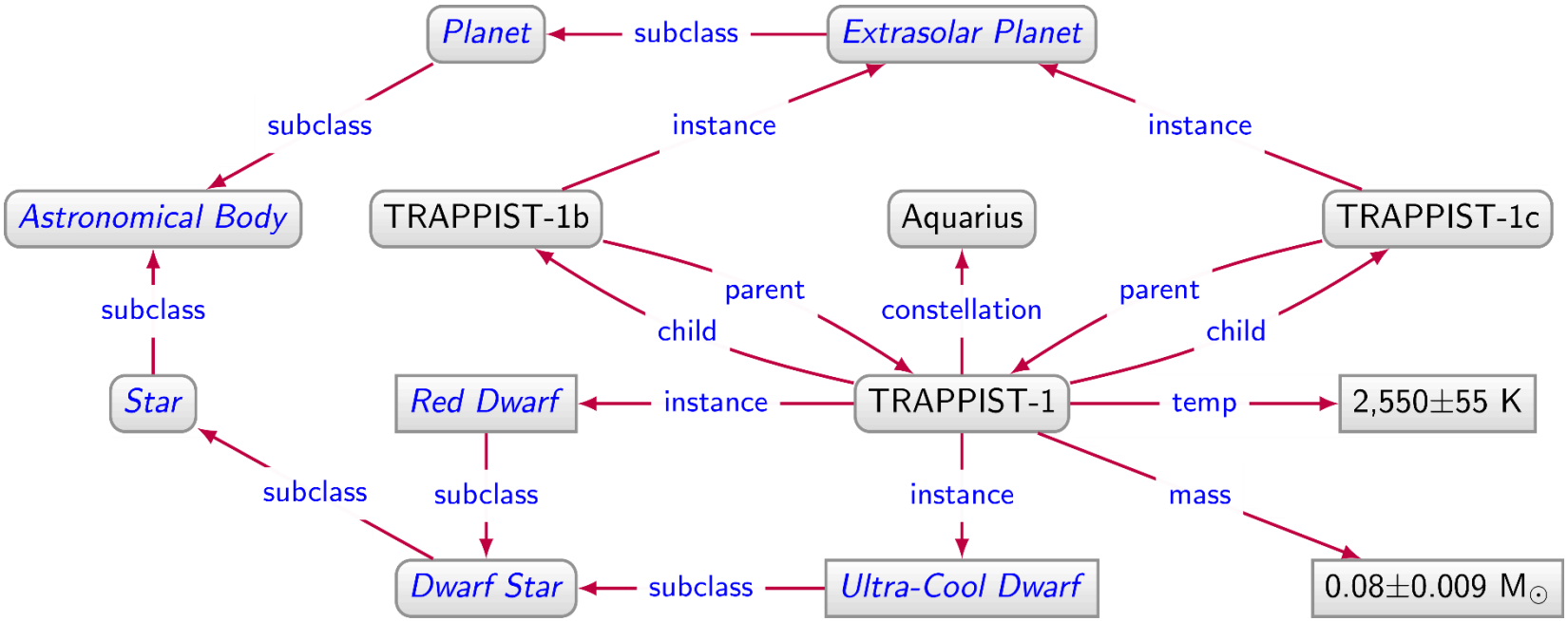
trend chart

31 systems in ranking, May 2018

Rank May 2018	Rank Apr 2018	Rank May 2017	DBMS	Database Model	Score May 2018	Score Apr 2018	Score May 2017
1.	1.	1.	Neo4j	Graph DBMS	40.58	-0.32	+4.44
2.	2.	↑ 4.	Microsoft Azure Cosmos DB	Multi-model	17.54	+0.35	+12.70
3.	3.		Datastax Enterprise	Multi-model	7.38	-0.09	
4.	4.	↓ 2.	OrientDB	Multi-model	5.25	-0.39	-0.49
5.	5.	5.	ArangoDB	Multi-model	3.70	-0.10	+0.75
6.	6.	6.	Virtuoso	Multi-model	1.79	-0.01	-0.27
7.	7.	7.	Giraph	Graph DBMS	0.98	-0.06	-0.11
8.	8.		Amazon Neptune	Multi-model	0.71	+0.02	
9.	9.	↓ 8.	AllegroGraph	Multi-model	0.58	+0.00	-0.02
10.	10.	↓ 9.	Stardog	Multi-model	0.51	-0.02	+0.00
11.	11.	↓ 10.	GraphDB	Multi-model	0.46	-0.00	-0.04
12.	↑ 14.	↑ 19.	JanusGraph	Graph DBMS	0.41	+0.12	+0.29
13.	↓ 12.	↑ 16.	Graph Engine	Multi-model	0.36	-0.04	+0.18
14.	↓ 13.	↓ 11.	Sqrrl	Multi-model	0.33	-0.06	-0.13
15.	15.	↑ 22.	Sparksee	Graph DBMS	0.19	-0.02	+0.14
16.	16.		TigerGraph	Graph DBMS	0.17	-0.01	
17.	↑ 20.	↓ 14.	Blazegraph	Multi-model	0.14	+0.01	-0.13
18.	18.	↓ 12.	Dgraph	Graph DBMS	0.14	+0.00	-0.15
19.	↓ 17.	↓ 17.	HyperGraphDB	Graph DBMS	0.14	-0.01	-0.02
20.	↓ 19.	↓ 15.	FlockDB	Graph DBMS	0.13	+0.00	-0.06
21.	↑ 23.	↓ 13.	InfiniteGraph	Graph DBMS	0.13	+0.02	-0.15
22.	22.	↓ 20.	FaunaDB	Multi-model	0.11	+0.00	+0.05
23.	↑ 24.	23.	VelocityDB	Multi-model	0.10	+0.02	+0.06
24.	↓ 21.	↓ 18.	InfoGrid	Graph DBMS	0.10	-0.02	-0.03
25.	↑ 26.	25.	AgensGraph	Multi-model	0.04	+0.01	+0.03

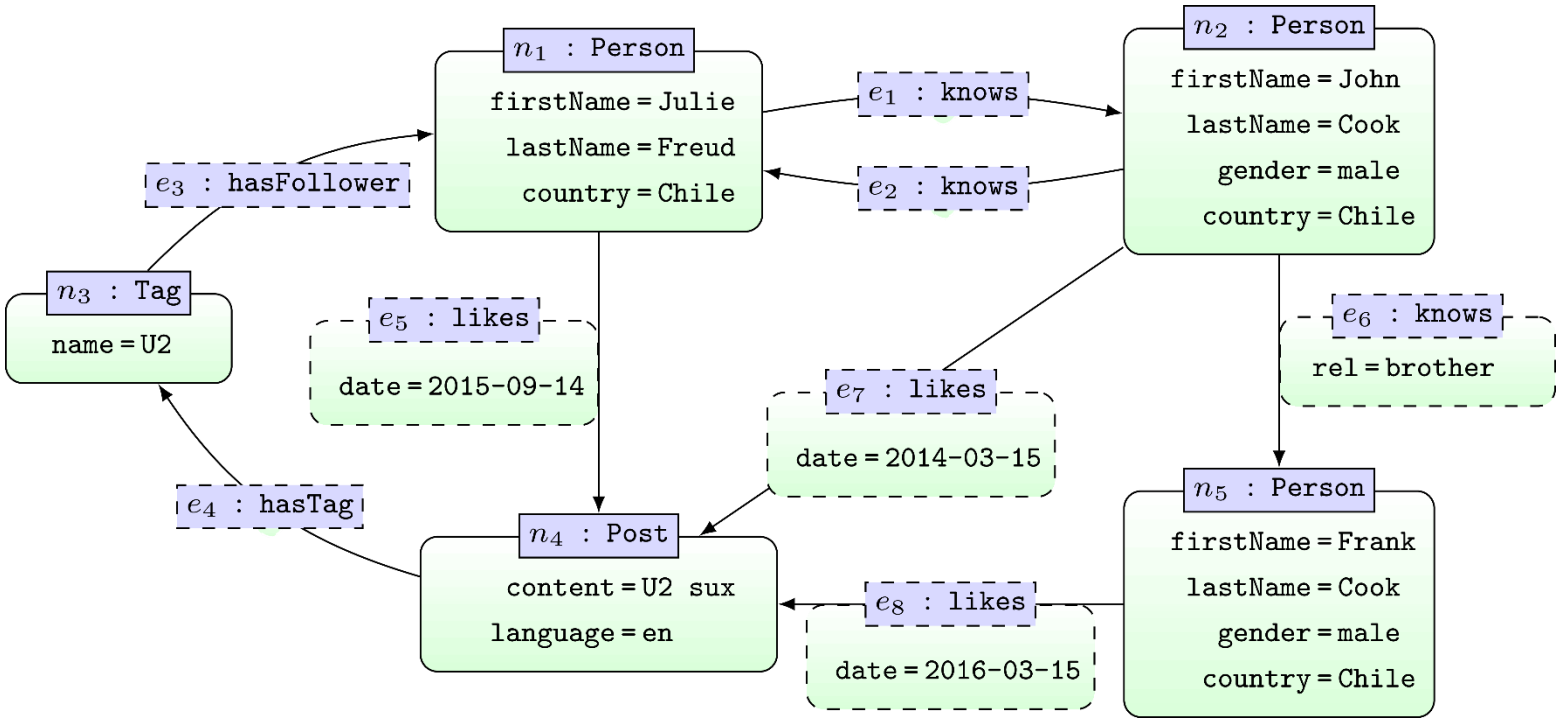
What is a Graph Database?

Directed Edge-labelled Graph




```
SELECT ?const (COUNT(DISTINCT ?body) AS ?num)
WHERE {
  ?body :instance/:subclass* :AstronomicalBody .
  ?body :parent?/:constellation ?const .
}
GROUP BY ?const
ORDER BY DESC(?num)
```

Property Graph




```
MATCH (x1:Person {firstName:"Julie"})-[:knows*]->(x2:Person)
MATCH (x2)-[:likes]->()-[:hasTag]->()-[:hasFollower]->(x1)
RETURN x2.firstName
```

Why do we need Graph Databases?

Why do we need Graph Databases?

Flexibility

Relational Databases …

Relational Databases ...

Debit

account	comment	date	time	amount	total	id
7873698669	Initial deposit	2020-21-01	20:02:02	300000	300000	TRCXGU8JSHD
7873698669	C0°0°L Designs	2020-02-06	09:15:33	50000	325000	TRCCIA2J8A0

Credit

account	comment	date	time	amount	total	id
7873698669	Electricity	2020-02-02	20:00:01	8200	291800	TRCJASJDA9A
7873698669	Heat	2020-02-02	20:00:02	600	291200	TRC81KAQWAS
7873698669	Moviestar	2020-02-02	20:00:03	16200	275000	TRCK8J7JA8D
7873698669	ATM	2020-02-08	16:05:02	100000	225000	TRCPM8A45AD

Account

number	rut	type	total_clp	total_usd
7873698669	32.000.273-K	Current	225000	344,94

Client

rut	name	phone	address
32.000.273-K	Kelvin	+56976698463	Campo de Hielo Sur, Depto 273

Exchange

c1	c2	value
CLP	USD	0,0001533
USD	CLP	652,2750000

Planets / Relational Database

Planets / Relational Database

Planet
name
Mercury
Venus
Earth
Mars
Jupiter
Saturn
Uranus
Neptune
Pluto

Planets / Relational Database

Planet	
name	**dist**
Mercury	
Venus	
Earth	1.00
Mars	
Jupiter	
Saturn	
Uranus	
Neptune	
Pluto	

Planets / Relational Database

Planet

name	**dist**
Mercury	0.39
Venus	0.72
Earth	1.00
Mars	1.52
Jupiter	
Saturn	
Uranus	
Neptune	
Pluto	49.31

Planets / Relational Database

Planet

name	dist	radius
Mercury	0.39	0.38
Venus	0.72	
Earth	1.00	1.00
Mars	1.52	0.53
Jupiter		10.97
Saturn	9.54	
Uranus	19.19	3.98
Neptune		
Pluto	49.31	

Planets / Relational Database

Planet

name	dist	radius	grav	days	years	temp	ring
Mercury	0.39	0.38	2.8	58.646	0.241	440	false
Venus	0.72	0.95	8.9	-243.019	0.615	730	false
Earth	1.00	1.00	9.8	0.997	1.000	288	false
Mars	1.52	0.53	3.7	1.026	1.880	186	false
Jupiter	5.20	10.97	22.9	0.414	11.862	152	true
Saturn	9.54	9.14	9.1	0.444	29.447	134	true
Uranus	19.19	3.98	7.8	-0.719	84.017	76	true
Neptune	30.07	3.86	11.0	0.671	164.791	53	true
Pluto	49.31	0.19	0.063	6.39	248.000	44	false

Planets / Relational Database

Planet

name	dist	radius	grav	days	years	temp	ring
Mercury	0.39	0.38	2.8	58.646	0.241	440	false
Venus	0.72	0.95	8.9	-243.019	0.615	730	false
Earth	1.00	1.00	9.8	0.997	1.000	288	false
Mars	1.52	0.53	3.7	1.026	1.880	186	false
Jupiter	5.20	10.97	22.9	0.414	11.862	152	true
Saturn	9.54	9.14	9.1	0.444	29.447	134	true
Uranus	19.19	3.98	7.8	-0.719	84.017	76	true
Neptune	30.07	3.86	11.0	0.671	164.791	53	true
Pluto	49.31	0.19	0.063	6.39	248.000	44	false

Planets / Relational Database

Planet

name	dist	radius	grav	days	years	temp	ring	moon
Mercury	0.39	0.38	2.8	58.646	0.241	440	false	⊥
Venus	0.72	0.95	8.9	-243.019	0.615	730	false	⊥
Earth	1.00	1.00	9.8	0.997	1.000	288	false	Luna
Mars	1.52	0.53	3.7	1.026	1.880	186	false	Phobos, Deimos
Jupiter	5.20	10.97	22.9	0.414	11.862	152	true	Callisto, Ganymede, ...
Saturn	9.54	9.14	9.1	0.444	29.447	134	true	Titan, Rhea, ...
Uranus	19.19	3.98	7.8	-0.719	84.017	76	true	Oberon, Titania, ...
Neptune	30.07	3.86	11.0	0.671	164.791	53	true	Triton, ...
Pluto	49.31	0.19	0.063	6.39	248.000	44	false	Charon

Planets / Relational Database

Planet

name	dist	radius	grav	days	years	temp	ring
Mercury	0.39	0.38	2.8	58.646	0.241	440	false
Venus	0.72	0.95	8.9	-243.019	0.615	730	false
Earth	1.00	1.00	9.8	0.997	1.000	288	false
Mars	1.52	0.53	3.7	1.026	1.880	186	false
Jupiter	5.20	10.97	22.9	0.414	11.862	152	true
Saturn	9.54	9.14	9.1	0.444	29.447	134	true
Uranus	19.19	3.98	7.8	-0.719	84.017	76	true
Neptune	30.07	3.86	11.0	0.671	164.791	53	true
Pluto	49.31	0.19	0.063	6.39	248.000	44	false

Moon

name	planet
Ganimedes	Jupiter
Calisto	Jupiter
Europa	Jupiter
Io	Jupiter
Titan	Saturn
Triton	Neptune
Luna	Terra
Oberon	Uranus
Charon	Pluto
...	...

Planets / Relational Database

Planet

name	dist	radius	grav	days	years	temp	ring
Mercury	0.39	0.38	2.8	58.646	0.241	440	false
Venus	0.72	0.95	8.9	-243.019	0.615	730	false
Earth	1.00	1.00	9.8	0.997	1.000	288	false
Mars	1.52	0.53	3.7	1.026	1.880	186	false
Jupiter	5.20	10.97	22.9	0.414	11.862	152	true
Saturn	9.54	9.14	9.1	0.444	29.447	134	true
Uranus	19.19	3.98	7.8	-0.719	84.017	76	true
Neptune	30.07	3.86	11.0	0.671	164.791	53	true
Pluto	49.31	0.19	0.063	6.39	248.000	44	false

Moon

name	planet	discoverer	year
Ganimedes	Jupiter	Galileo Galilei	1610
Calisto	Jupiter	Galileo Galilei	1610
Europa	Jupiter	Galileo Galilei	1610
Io	Jupiter	Galileo Galilei	1610
Titan	Saturn	Christiaan Huygens	1655
Triton	Neptune	William Lassell	1846
Luna	Terra	$\perp$	$\perp$
Oberon	Uranus	William Herschel	1787
Charon	Pluto	$\perp$	1978
...	...	...	...

Planets / Relational Database

Planet

name	dist	radius	grav	days	years	temp	ring
Mercury	0.39	0.38	2.8	58.646	0.241	440	false
Venus	0.72	0.95	8.9	-243.019	0.615	730	false
Earth	1.00	1.00	9.8	0.997	1.000	288	false
Mars	1.52	0.53	3.7	1.026	1.880	186	false
Jupiter	5.20	10.97	22.9	0.414	11.862	152	true
Saturn	9.54	9.14	9.1	0.444	29.447	134	true
Uranus	19.19	3.98	7.8	-0.719	84.017	76	true
Neptune	30.07	3.86	11.0	0.671	164.791	53	true
Pluto	49.31	0.19	0.063	6.39	248.000	44	false

Moon

name	planet
Ganimedes	Jupiter
Calisto	Jupiter
Europa	Jupiter
Io	Jupiter
Titan	Saturn
Triton	Neptune
Luna	Terra
Oberon	Uranus
Charon	Pluto
...	...

MoonDiscoverer

name	discoverer
Ganimedes	Galileo Galilei
Calisto	Galileo Galilei
Europa	Galileo Galilei
Io	Galileo Galilei
Titan	Christiaan Huygens
Triton	William Lassell
Oberon	William Herschel
...	...

MoonDiscYear

name	year
Ganimedes	1610
Calisto	1610
Europa	1610
Io	1610
Titan	1655
Triton	1846
Oberon	1787
Charon	1978
...	...

Planets / Relational Database

Planet

name	dist	radius	grav	days	years	temp	ring
Mercury	0.39	0.38	2.8	58.646	0.241	440	false
Venus	0.72	0.95	8.9	-243.019	0.615	730	false
Earth	1.00	1.00	9.8	0.997	1.000	288	false
Mars	1.52	0.53	3.7	1.026	1.880	186	false
Jupiter	5.20	10.97	22.9	0.414	11.862	152	true
Saturn	9.54	9.14	9.1	0.444	29.447	134	true
Uranus	19.19	3.98	7.8	-0.719	84.017	76	true
Neptune	30.07	3.86	11.0	0.671	164.791	53	true
Pluto	49.31	0.19	0.063	6.39	248.000	44	false

Moon

name	planet
Ganimedes	Jupiter
Calisto	Jupiter
Europa	Jupiter
Io	Jupiter
Titan	Saturn
Triton	Neptune
Luna	Terra
Oberon	Uranus
Charon	Pluto
...	...

MoonDiscoverer

name	discoverer
Ganimedes	Galileo Galilei
Calisto	Galileo Galilei
Europa	Galileo Galilei
Io	Galileo Galilei
Titan	Christiaan Huygens
Triton	William Lassell
Oberon	William Herschel
...	...

MoonDiscYear

name	year
Ganimedes	1610
Calisto	1610
Europa	1610
Io	1610
Titan	1655
Triton	1846
Oberon	1787
Charon	1978
...	...

Planets / Relational Database

Planet

name	dist	radius	grav	days	years	temp	ring
Mercury	0.39	0.38	2.8	58.646	0.241	440	false
Venus	0.72	0.95	8.9	-243.019	0.615	730	false
Earth	1.00	1.00	9.8	0.997	1.000	288	false
Mars	1.52	0.53	3.7	1.026	1.880	186	false
Jupiter	5.20	10.97	22.9	0.414	11.862	152	true
Saturn	9.54	9.14	9.1	0.444	29.447	134	true
Uranus	19.19	3.98	7.8	-0.719	84.017	76	true
Neptune	30.07	3.86	11.0	0.671	164.791	53	true
Pluto	49.31	0.19	0.063	6.39	248.000	44	false

Moon

name	P.name
Ganimedes	Jupiter
Calisto	Jupiter
Europa	Jupiter
Io	Jupiter
Titan	Saturn
Triton	Neptune
Luna	Earth
Oberon	Uranus
Charon	Pluto
...	...

MoonDiscoverer

name	discoverer
Ganimedes	Galileo Galilei
Calisto	Galileo Galilei
Europa	Galileo Galilei
Io	Galileo Galilei
Titan	Christiaan Huygens
Triton	William Lassell
Oberon	William Herschel
...	...

MoonDiscYear

name	year
Ganimedes	1610
Calisto	1610
Europa	1610
Io	1610
Titan	1655
Triton	1846
Oberon	1787
Charon	1978
...	...

Planets / Relational Database

Planet

name	dist	radius	grav	days	years	temp	ring
Mercury	0.39	0.38	2.8	58.646	0.241	440	false
Venus	0.72	0.95	8.9	-243.019	0.615	730	false
Earth	1.00	1.00	9.8	0.997	1.000	288	false
Mars	1.52	0.53	3.7	1.026	1.880	186	false
Jupiter	5.20	10.97	22.9	0.414	11.862	152	true
Saturn	9.54	9.14	9.1	0.444	29.447	134	true
Uranus	19.19	3.98	7.8	-0.719	84.017	76	true
Neptune	30.07	3.86	11.0	0.671	164.791	53	true
Pluto	49.31	0.19	0.063	6.39	248.000	44	false

Moon

name	P.name
Ganimedes	Jupiter
Calisto	Jupiter
Europa	Jupiter
Io	Jupiter
Titan	Saturn
Triton	Neptune
Luna	Earth
Oberon	Uranus
Charon	Pluto
...	...

MoonDiscoverer

name	discoverer
Ganimedes	Galileo Galilei
Calisto	Galileo Galilei
Europa	Galileo Galilei
Io	Galileo Galilei
Titan	Christiaan Huygens
Triton	William Lassell
Oberon	William Herschel
...	...

MoonDiscYear

name	year
Ganimedes	1610
Calisto	1610
Europa	1610
Io	1610
Titan	1655
Triton	1846
Oberon	1787
Charon	1978
...	...

Planets / Relational Database

Planet

name	dist	radius	grav	days	years	temp	ring
Mercury	0.39	0.38	2.8	58.646	0.241	440	false
Venus	0.72	0.95	8.9	-243.019	0.615	730	false
Earth	1.00	1.00	9.8	0.997	1.000	288	false
Mars	1.52	0.53	3.7	1.026	1.880	186	false
Jupiter	5.20	10.97	22.9	0.414	11.862	152	true
Saturn	9.54	9.14	9.1	0.444	29.447	134	true
Uranus	19.19	3.98	7.8	-0.719	84.017	76	true
Neptune	30.07	3.86	11.0	0.671	164.791	53	true

DwarfPlanet

name	dist	radius	grav	days	years	temp	ring
Pluto	49.31	0.19	0.063	6.39	248.000	44	false

Moon

name	P.name
Ganimedes	Jupiter
Calisto	Jupiter
Europa	Jupiter
Io	Jupiter
Titan	Saturn
Triton	Neptune
Luna	Earth
Oberon	Uranus
Charon	Pluto
...	...

MoonDiscoverer

name	discoverer
Ganimedes	Galileo Galilei
Calisto	Galileo Galilei
Europa	Galileo Galilei
Io	Galileo Galilei
Titan	Christiaan Huygens
Triton	William Lassell
Oberon	William Herschel
...	...

MoonDiscYear

name	year
Ganimedes	1610
Calisto	1610
Europa	1610
Io	1610
Titan	1655
Triton	1846
Oberon	1787
Charon	1978
...	...

Planets / Relational Database

Planet

name	dist	radius	grav	days	years	temp	ring
Mercury	0.39	0.38	2.8	58.646	0.241	440	false
Venus	0.72	0.95	8.9	-243.019	0.615	730	false
Earth	1.00	1.00	9.8	0.997	1.000	288	false
Mars	1.52	0.53	3.7	1.026	1.880	186	false
Jupiter	5.20	10.97	22.9	0.414	11.862	152	true
Saturn	9.54	9.14	9.1	0.444	29.447	134	true
Uranus	19.19	3.98	7.8	-0.719	84.017	76	true
Neptune	30.07	3.86	11.0	0.671	164.791	53	true

DwarfPlanet

name	dist	radius	grav	days	years	temp	ring
Pluto	49.31	0.19	0.063	6.39	248.000	44	false

Moon

name	P.name
Ganimedes	Jupiter
Calisto	Jupiter
Europa	Jupiter
Io	Jupiter
Titan	Saturn
Triton	Neptune
Luna	Earth
Oberon	Uranus
Charon	Pluto
...	...

MoonDiscoverer

name	discoverer
Ganimedes	Galileo Galilei
Calisto	Galileo Galilei
Europa	Galileo Galilei
Io	Galileo Galilei
Titan	Christiaan Huygens
Triton	William Lassell
Oberon	William Herschel
...	...

MoonDiscYear

name	year
Ganimedes	1610
Calisto	1610
Europa	1610
Io	1610
Titan	1655
Triton	1846
Oberon	1787
Charon	1978
...	...

Planets / Relational Database

Planet

name	dist	radius	grav	days	years	temp	ring
Mercury	0.39	0.38	2.8	58.646	0.241	440	false
Venus	0.72	0.95	8.9	-243.019	0.615	730	false
Earth	1.00	1.00	9.8	0.997	1.000	288	false
Mars	1.52	0.53	3.7	1.026	1.880	186	false
Jupiter	5.20	10.97	22.9	0.414	11.862	152	true
Saturn	9.54	9.14	9.1	0.444	29.447	134	true
Uranus	19.19	3.98	7.8	-0.719	84.017	76	true
Neptune	30.07	3.86	11.0	0.671	164.791	53	true

DwarfPlanet

name	dist	radius	grav	days	years	temp	ring
Pluto	49.31	0.19	0.063	6.39	248.000	44	false

Moon

name	parent
Ganimedes	Jupiter
Calisto	Jupiter
Europa	Jupiter
Io	Jupiter
Titan	Saturn
Triton	Neptune
Luna	Earth
Oberon	Uranus
Charon	Pluto
...	...

MoonDiscoverer

name	discoverer
Ganimedes	Galileo Galilei
Calisto	Galileo Galilei
Europa	Galileo Galilei
Io	Galileo Galilei
Titan	Christiaan Huygens
Triton	William Lassell
Oberon	William Herschel
...	...

MoonDiscYear

name	year
Ganimedes	1610
Calisto	1610
Europa	1610
Io	1610
Titan	1655
Triton	1846
Oberon	1787
Charon	1978
...	...

Planets / Relational Database

Planet

name	dist	radius	grav	days	years	temp	ring
Mercury	0.39	0.38	2.8	58.646	0.241	440	false
Venus	0.72	0.95	8.9	-243.019	0.615	730	false
Earth	1.00	1.00	9.8	0.997	1.000	288	false
Mars	1.52	0.53	3.7	1.026	1.880	186	false
Jupiter	5.20	10.97	22.9	0.414	11.862	152	true
Saturn	9.54	9.14	9.1	0.444	29.447	134	true
Uranus	19.19	3.98	7.8	-0.719	84.017	76	true
Neptune	30.07	3.86	11.0	0.671	164.791	53	true

DwarfPlanet

name	dist	radius	grav	days	years	temp	ring
Pluto	49.31	0.19	0.063	6.39	248.000	44	false

Moon

name	parent
Ganimedes	Jupiter
Calisto	Jupiter
Europa	Jupiter
Io	Jupiter
Titan	Saturn
Triton	Neptune
Luna	Earth
Oberon	Uranus
Charon	Pluto
...	...

MoonDiscoverer

name	discoverer
Ganimedes	Galileo Galilei
Calisto	Galileo Galilei
Europa	Galileo Galilei
Io	Galileo Galilei
Titan	Christiaan Huygens
Triton	William Lassell
Oberon	William Herschel
...	...

MoonDiscYear

name	year
Ganimedes	1610
Calisto	1610
Europa	1610
Io	1610
Titan	1655
Triton	1846
Oberon	1787
Charon	1978
...	...

Planets / Relational Database

Planet

name	dist	radius	grav	days	years	temp	ring
Mercury	0.39	0.38	2.8	58.646	0.241	440	false
Venus	0.72	0.95	8.9	-243.019	0.615	730	false
Earth	1.00	1.00	9.8	0.997	1.000	288	false
Mars	1.52	0.53	3.7	1.026	1.880	186	false
Jupiter	5.20	10.97	22.9	0.414	11.862	152	true
Saturn	9.54	9.14	9.1	0.444	29.447	134	true
Uranus	19.19	3.98	7.8	-0.719	84.017	76	true
Neptune	30.07	3.86	11.0	0.671	164.791	53	true

(ノ°□°)ノ ┻━┻

DwarfPlanet

name	dist	radius	grav	days	years	temp	ring
Pluto	49.31	0.19	0.063	6.39	248.000	44	false

Moon

name	parent
Ganimedes	Jupiter
Calisto	Jupiter
Europa	Jupiter
Io	Jupiter
Titan	Saturn
Triton	Neptune
Luna	Earth
Oberon	Uranus
Charon	Pluto
...	...

MoonDiscoverer

name	discoverer
Ganimedes	Galileo Galilei
Calisto	Galileo Galilei
Europa	Galileo Galilei
Io	Galileo Galilei
Titan	Christiaan Huygens
Triton	William Lassell
Oberon	William Herschel
...	...

MoonDiscYear

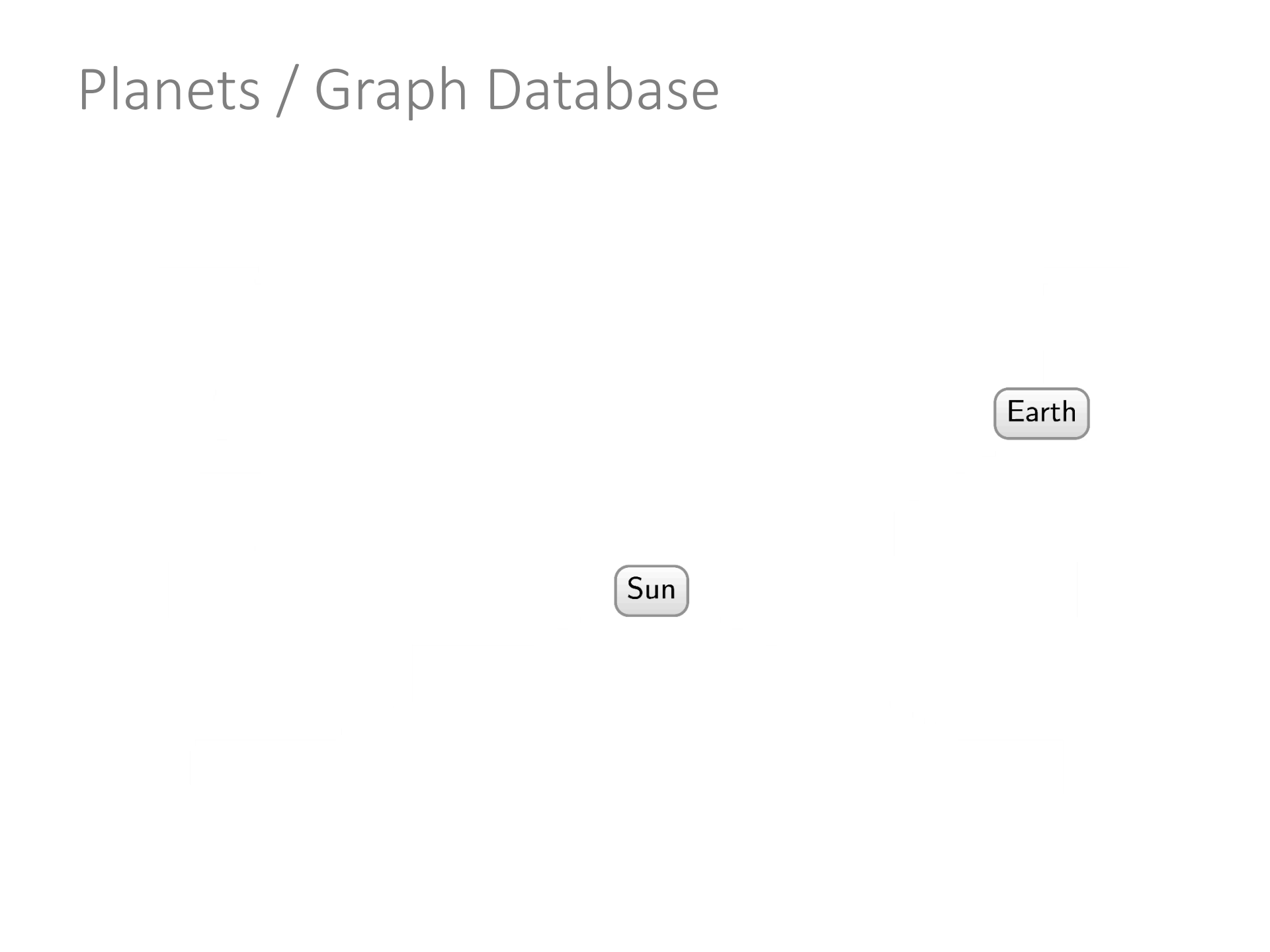
name	year
Ganimedes	1610
Calisto	1610
Europa	1610
Io	1610
Titan	1655
Triton	1846
Oberon	1787
Charon	1978
...	...

Planets / Graph Database

Planets / Graph Database

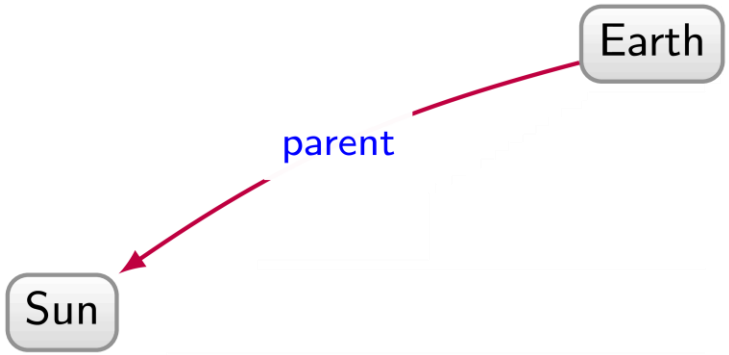
Planets / Graph Database

Planets / Graph Database

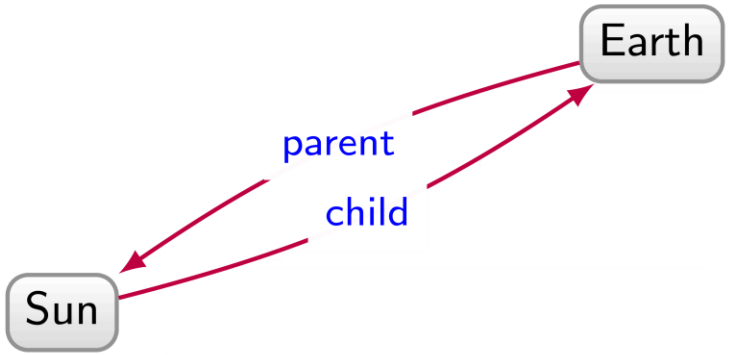


Planets / Graph Database

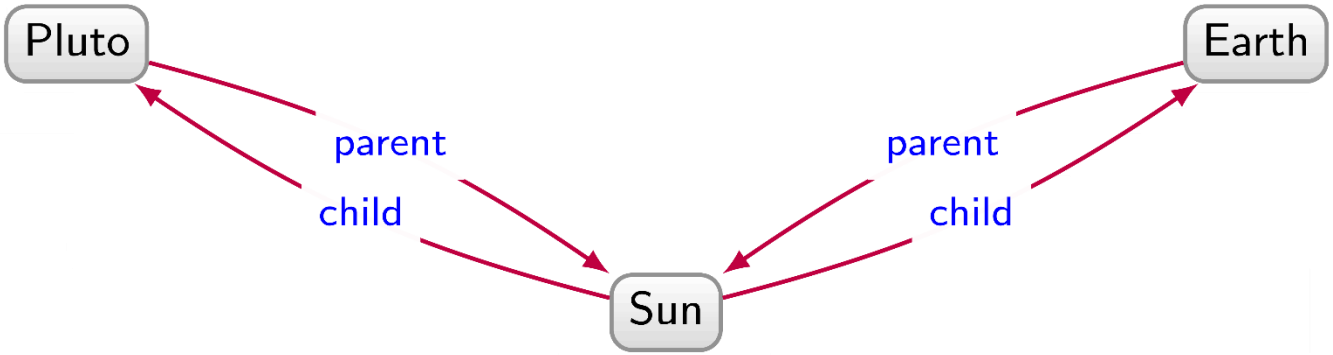
Planets / Graph Database



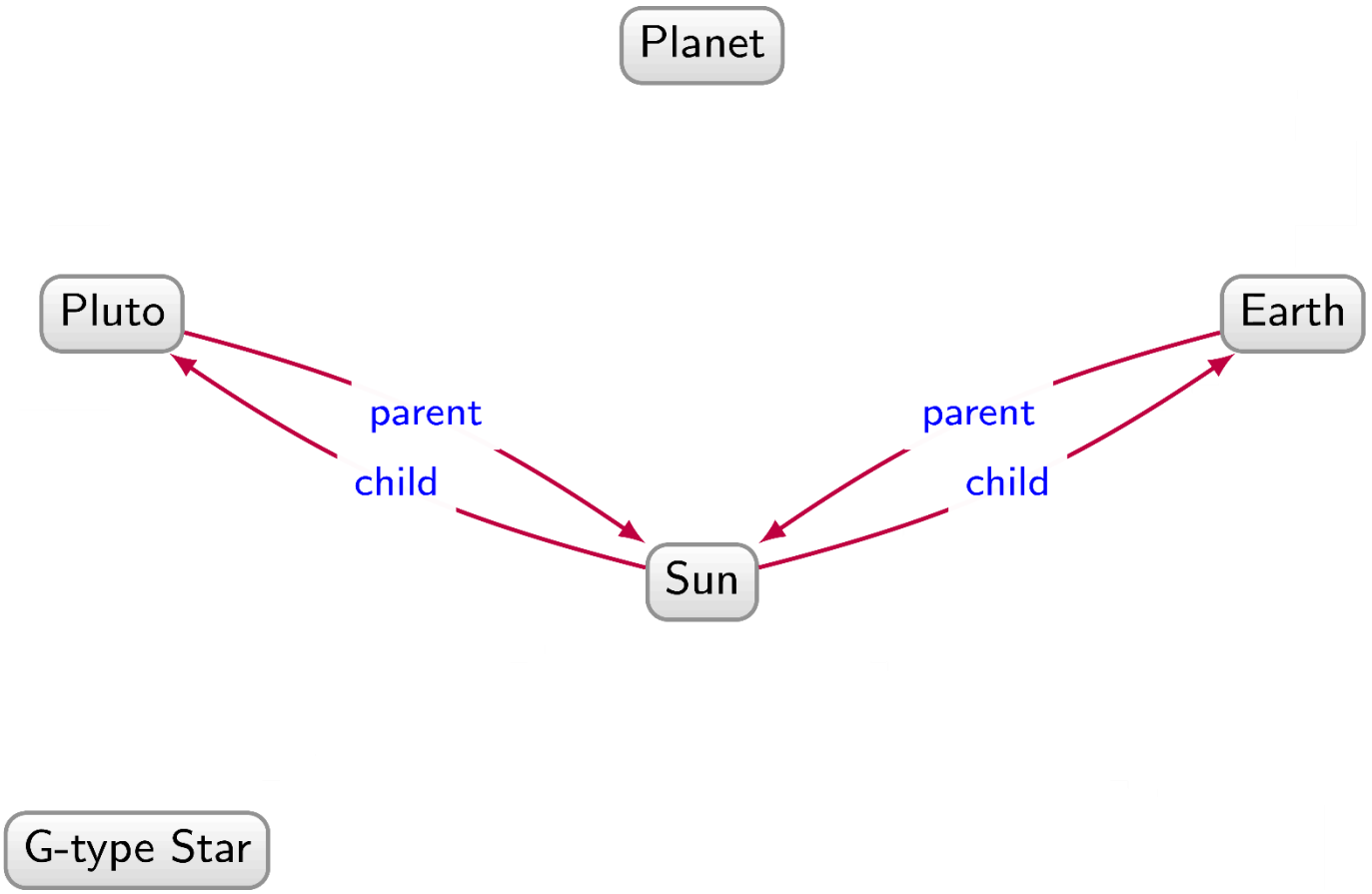
Planets / Graph Database



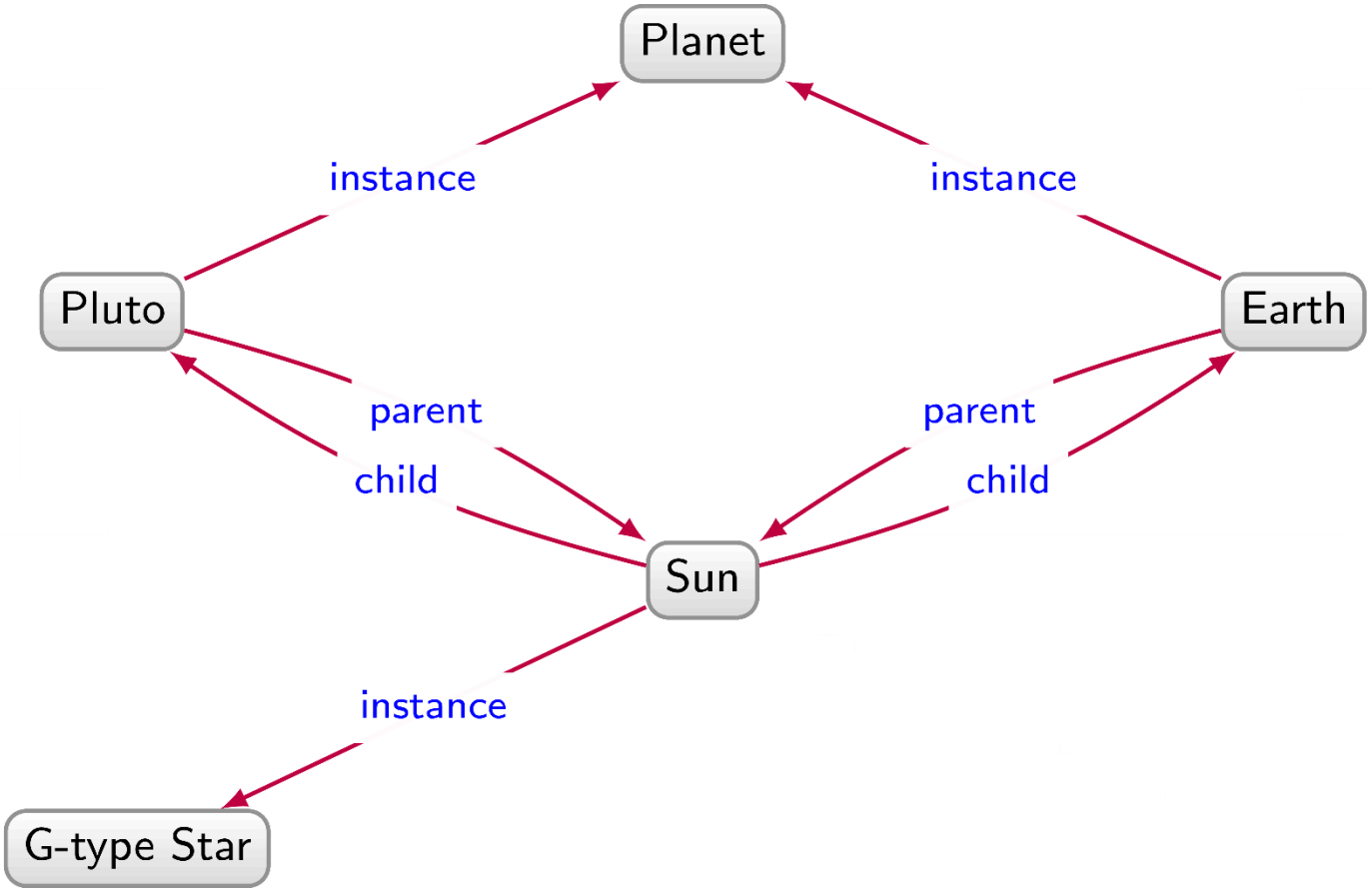
Planets / Graph Database



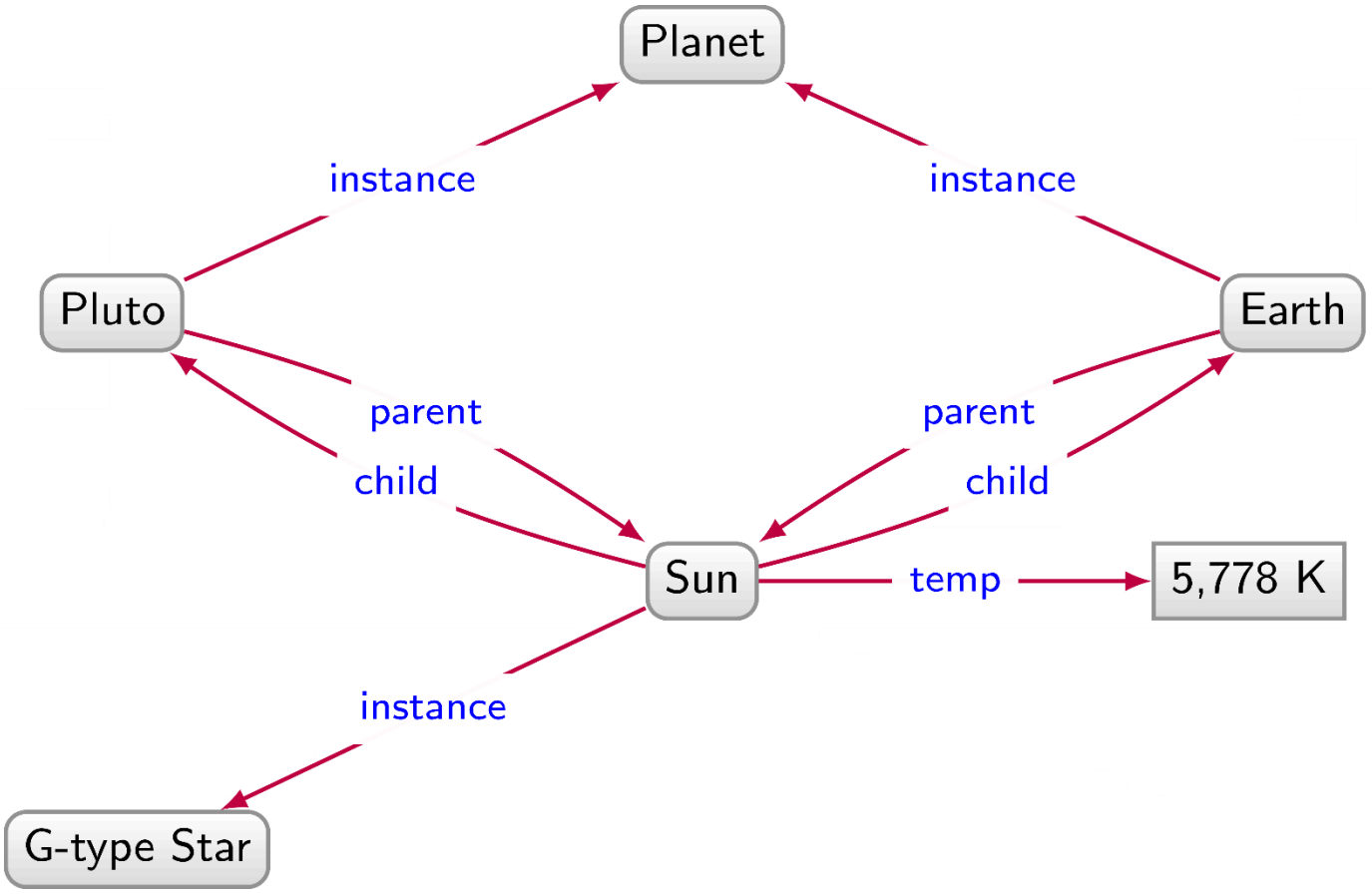
Planets / Graph Database



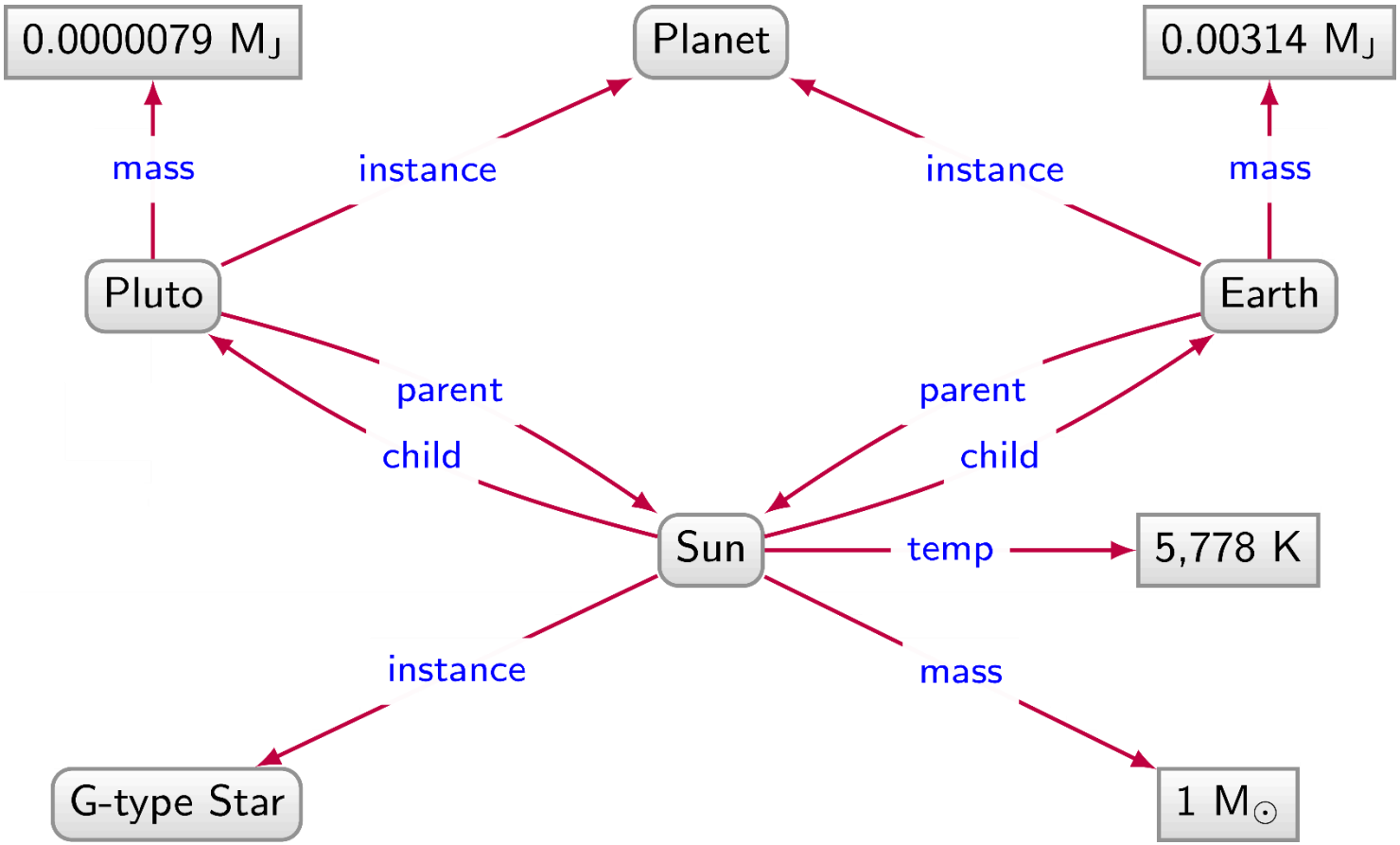
Planets / Graph Database



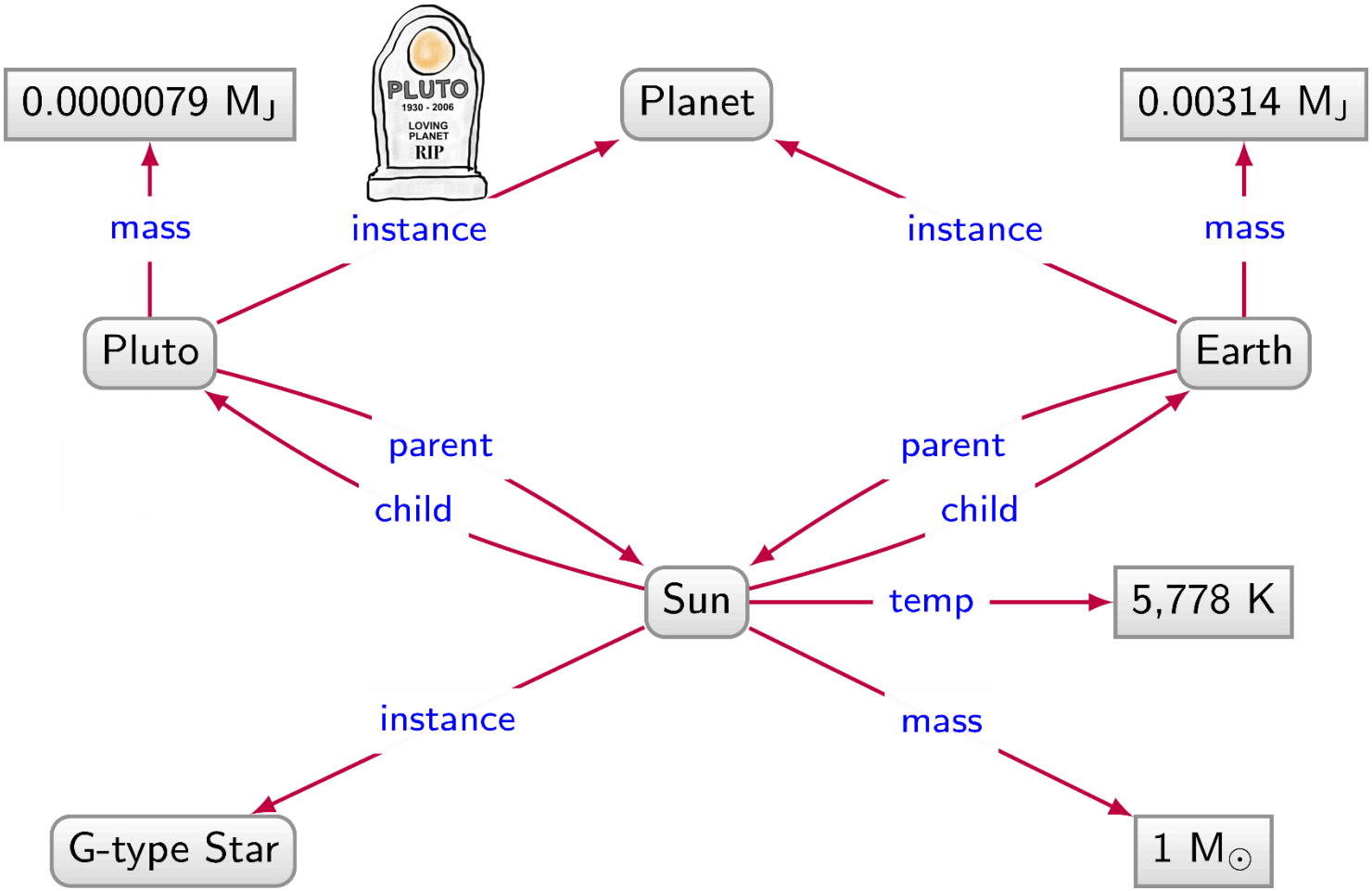
Planets / Graph Database



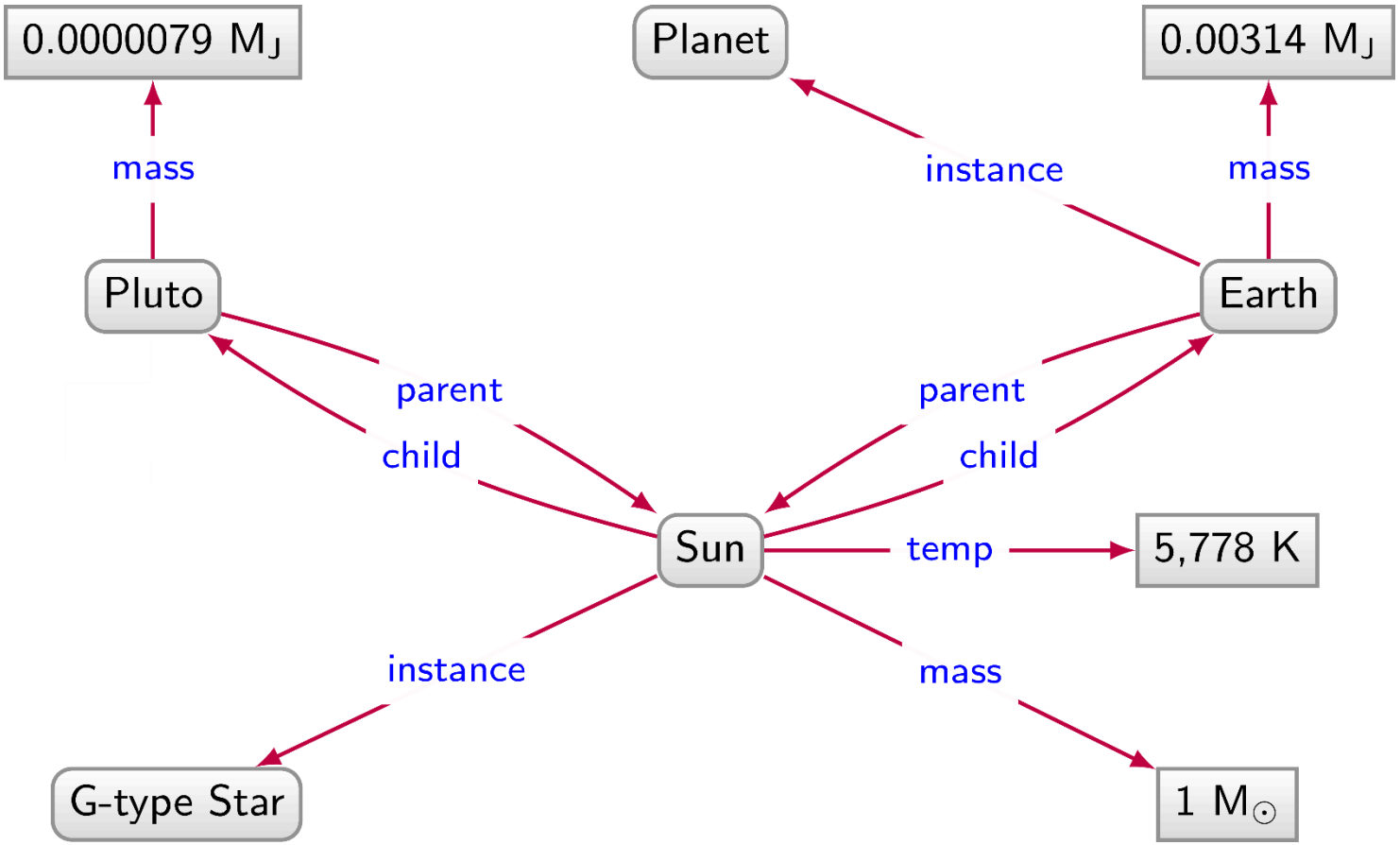
Planets / Graph Database



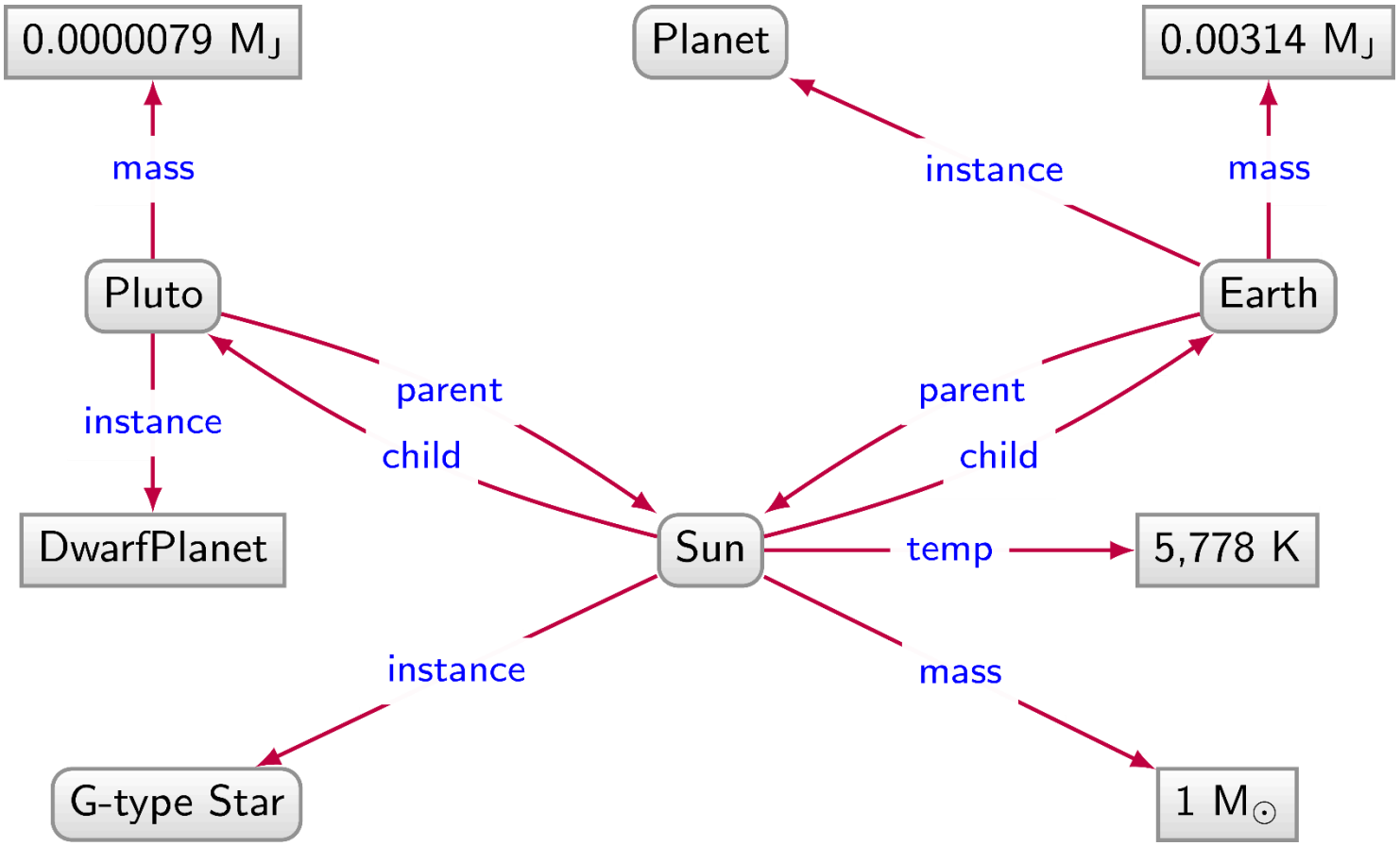
Planets / Graph Database



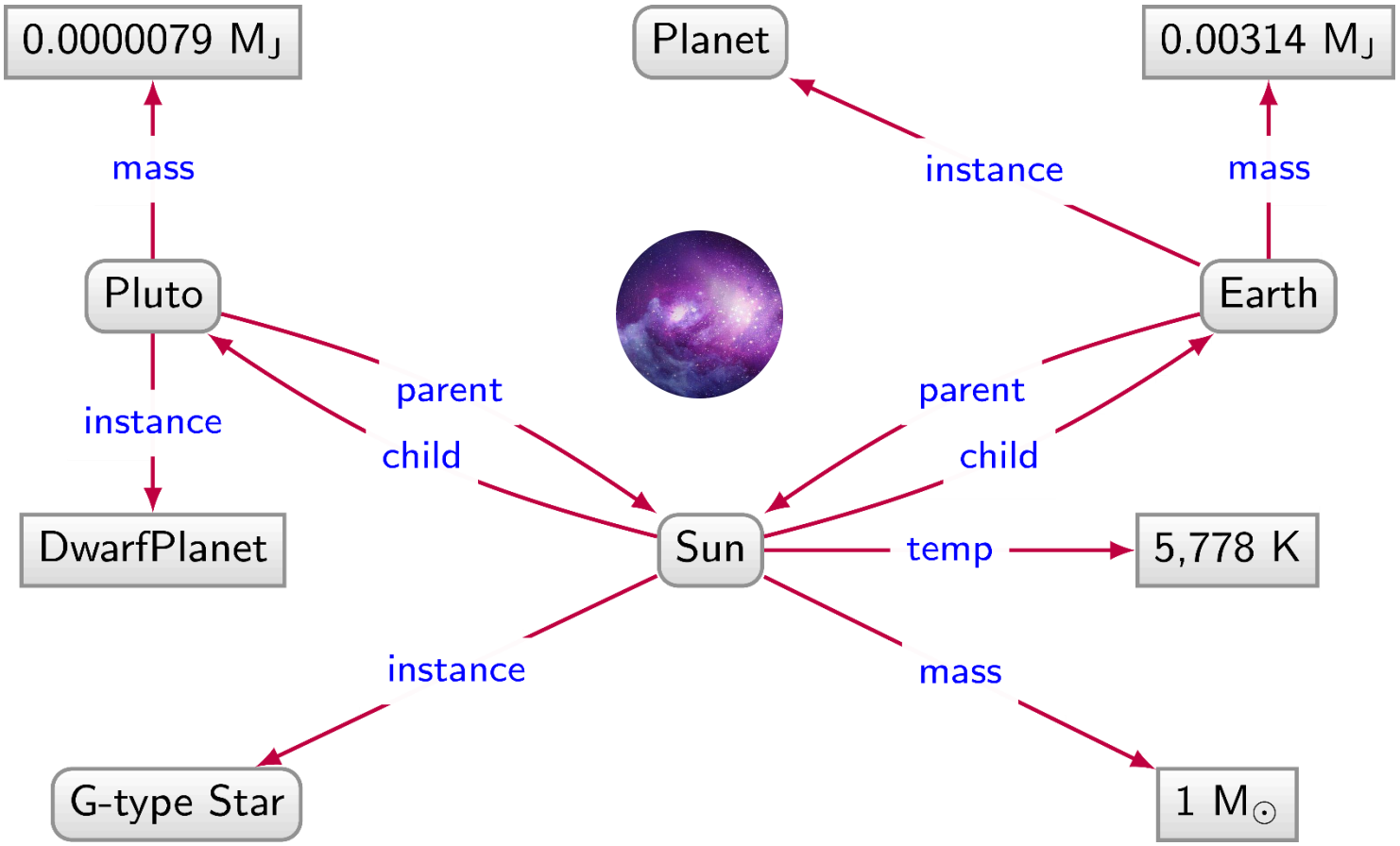
Planets / Graph Database



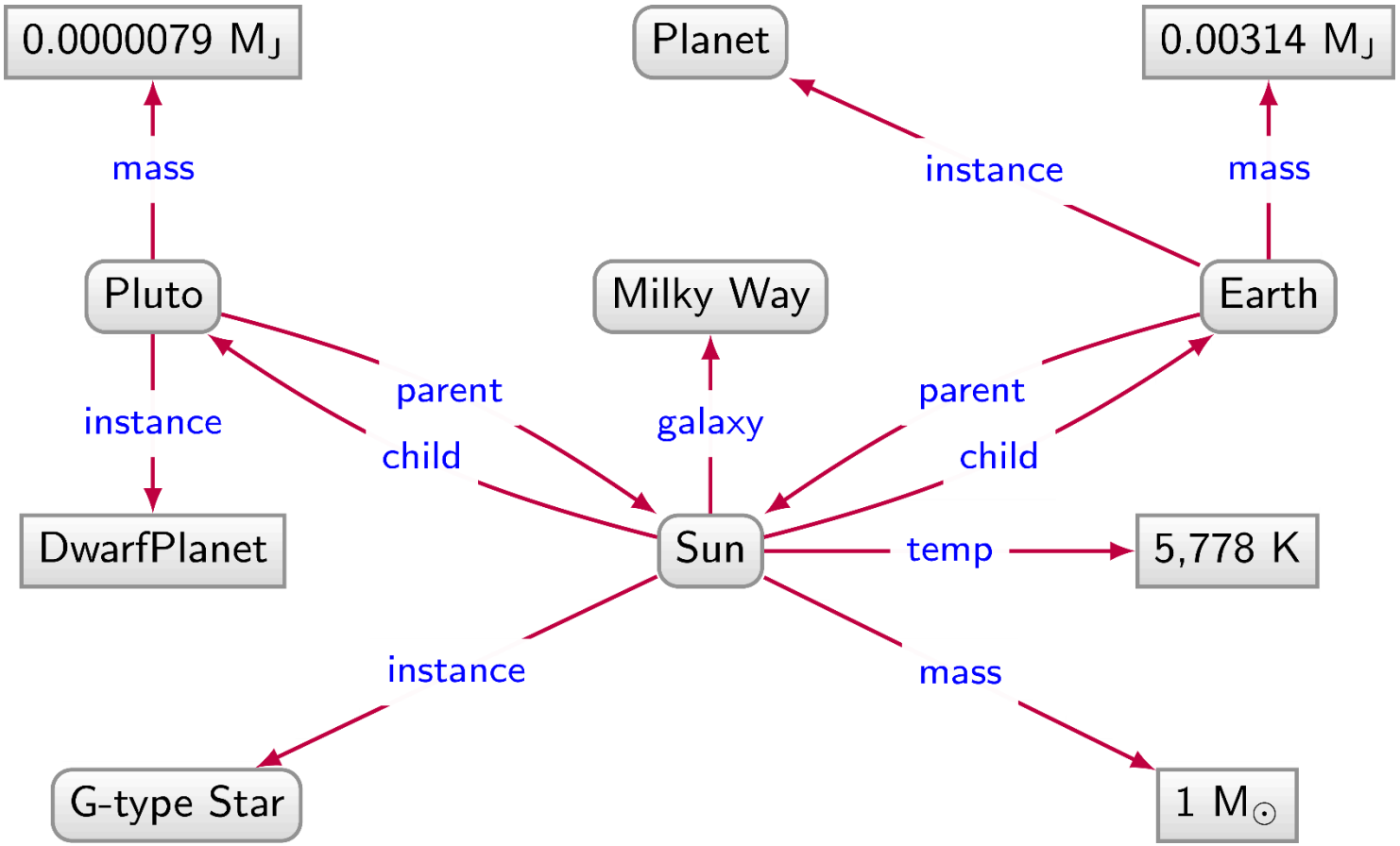
Planets / Graph Database



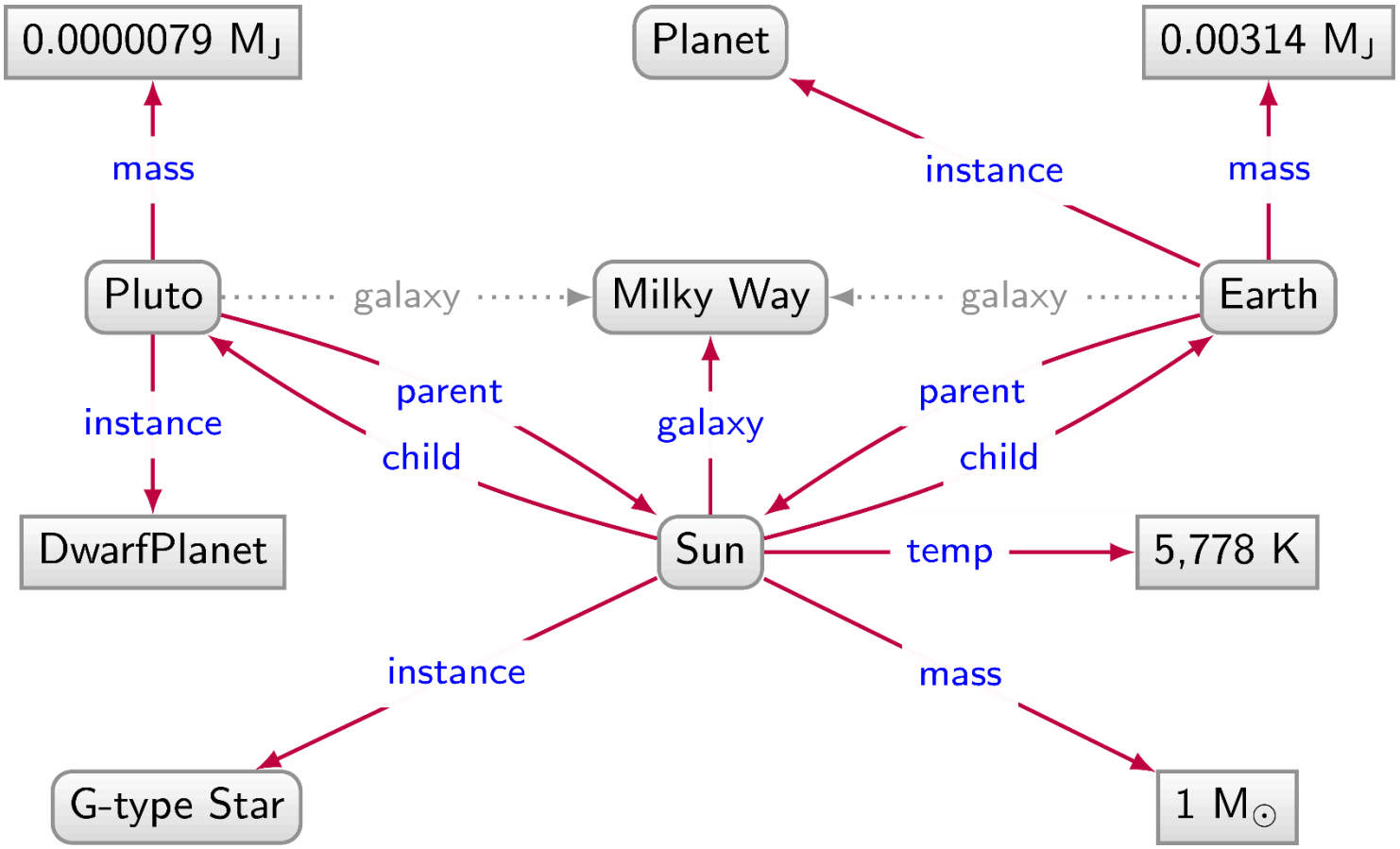
Planets / Graph Database



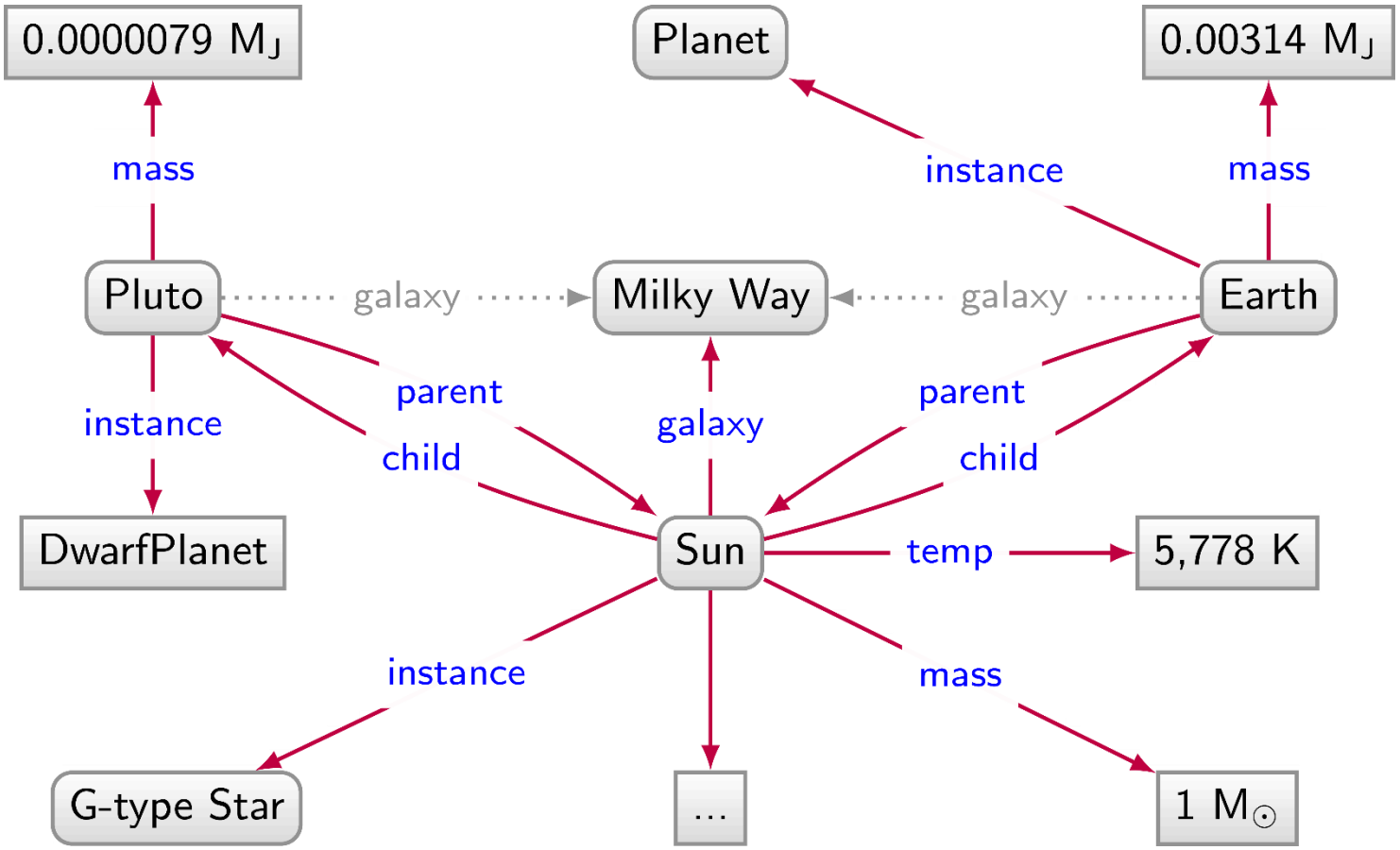
Planets / Graph Database



Planets / Graph Database



Planets / Graph Database

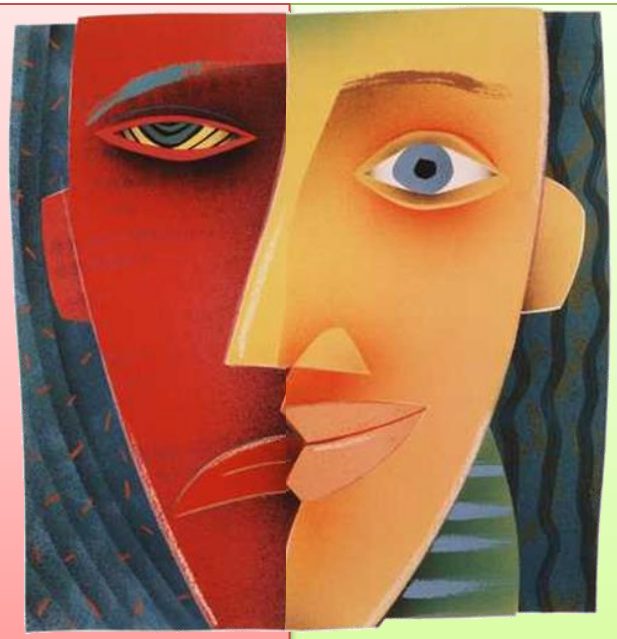


Relational databases: Pros and Cons

Planet

name	dist	radius	grav	days	years	temp	ring
Mercury	0.39	0.38	2.8	58.646	0.241	440	false
Venus	0.72	0.95	8.9	-243.019	0.615	730	false

We have to impose a
structure (schema)
from the start



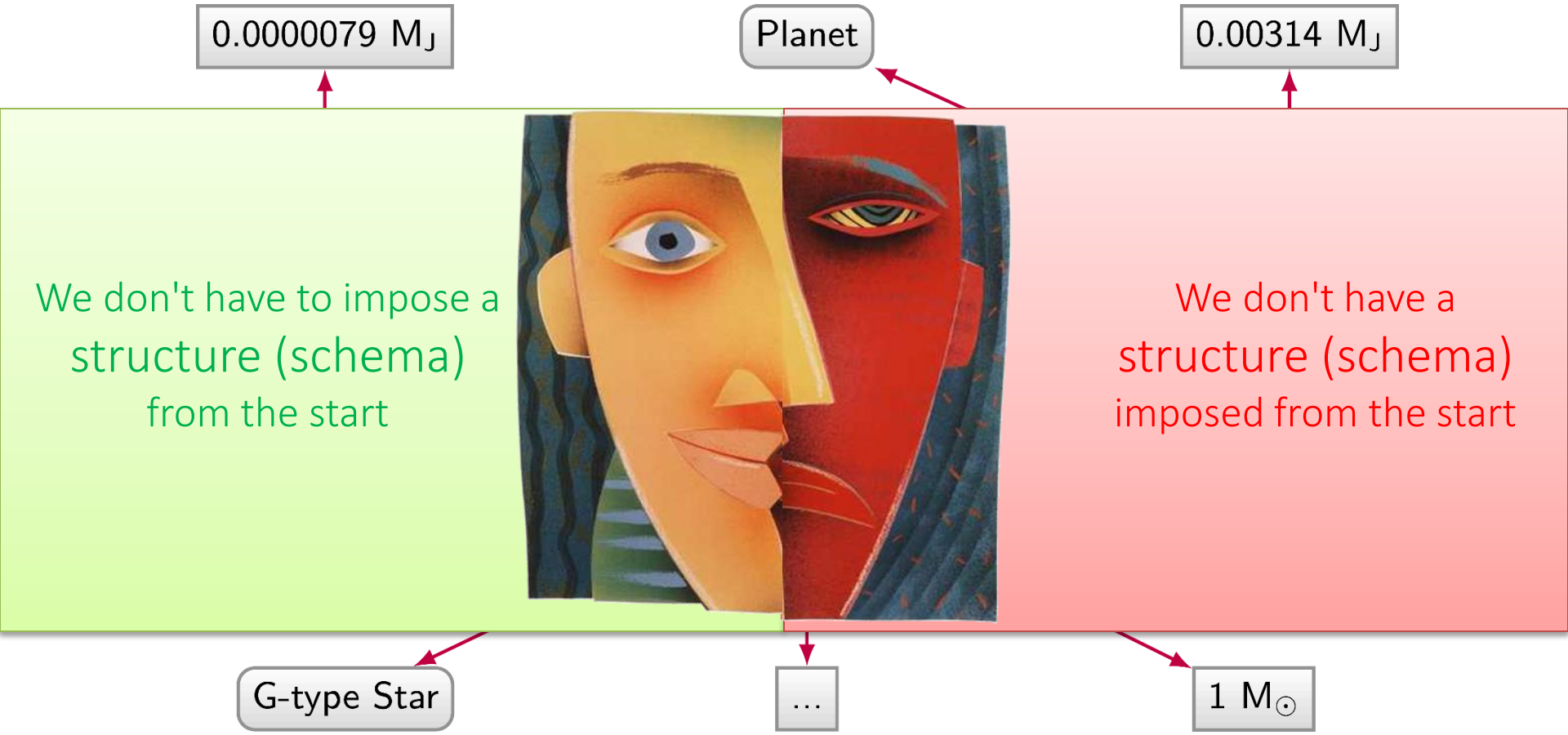
We have a
structure (schema)
imposed from the start

Europa	Jupiter
Io	Jupiter
Titan	Saturn
Triton	Neptune
Luna	Earth
Oberon	Uranus
Charon	Pluto
...	...

Europa	Galileo Galilei
Io	Galileo Galilei
Titan	Christiaan Huygens
Triton	William Lassell
Oberon	William Herschel
...	...

Europa	1610
Io	1610
Titan	1655
Triton	1846
Oberon	1787
Charon	1978
...	...

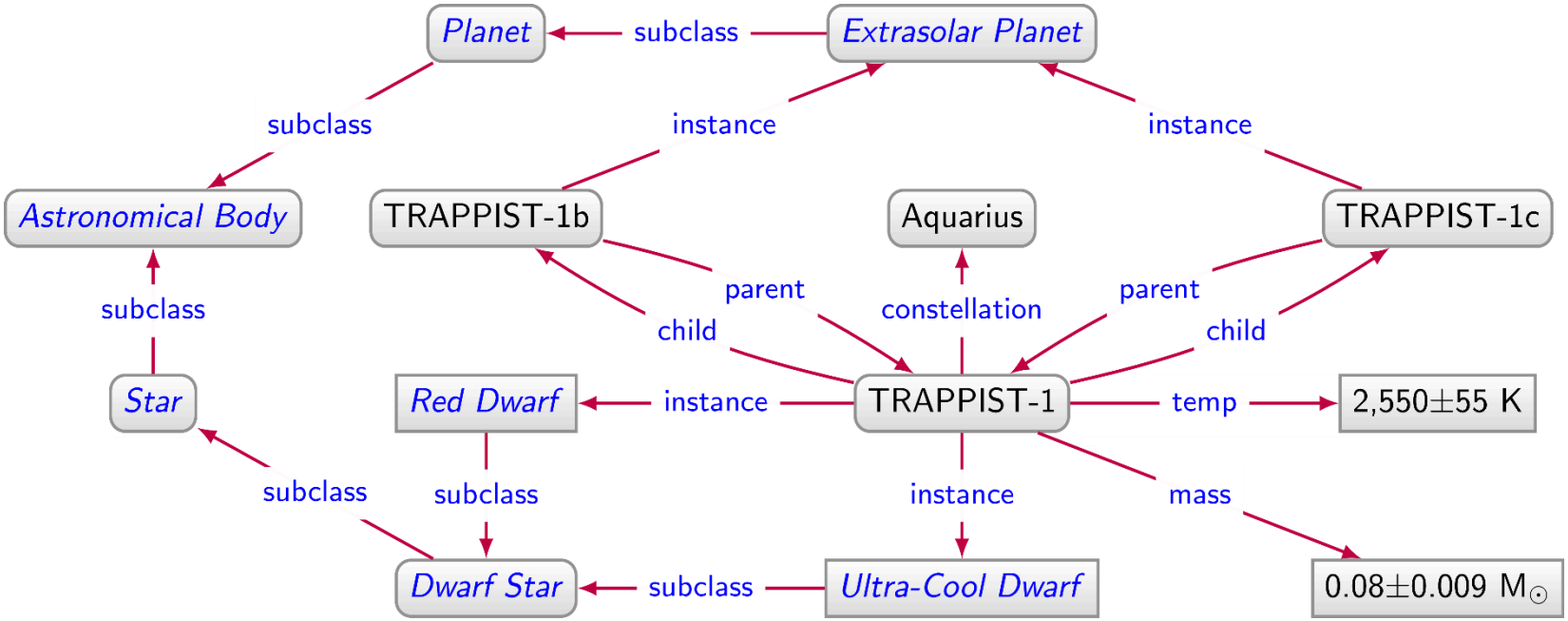
Graph Databases: Pros and Cons



Why do we need Graph Databases?

Path Queries

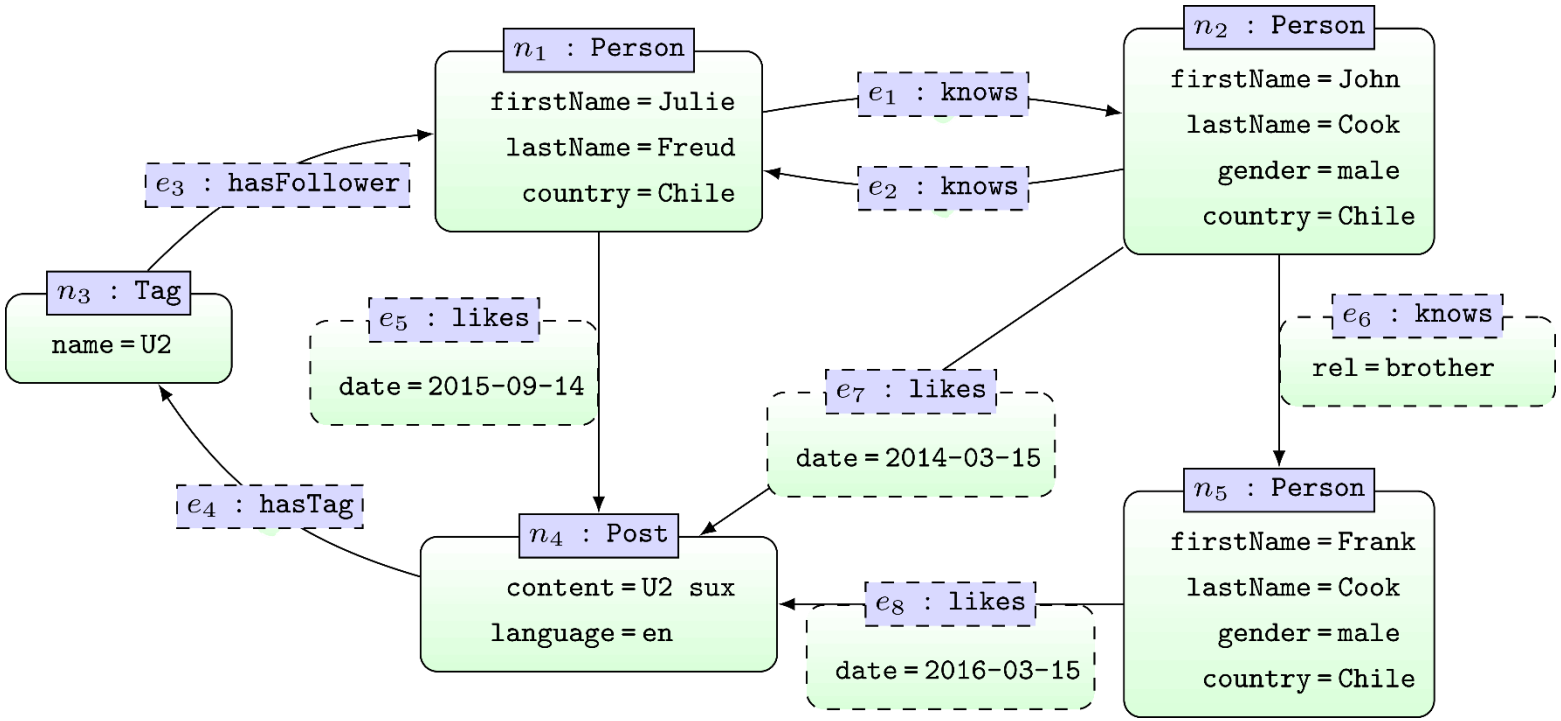
Directed Edge-labelled Graph




```
SELECT ?const (COUNT(DISTINCT ?body) AS ?num)
WHERE {
  ?body :instance/:subclass* :AstronomicalBody .
  ?body :parent?/:constellation ?const .
}
GROUP BY ?const
ORDER BY DESC(?num)
```

?const	?num
:Aquarius	3

Property Graph



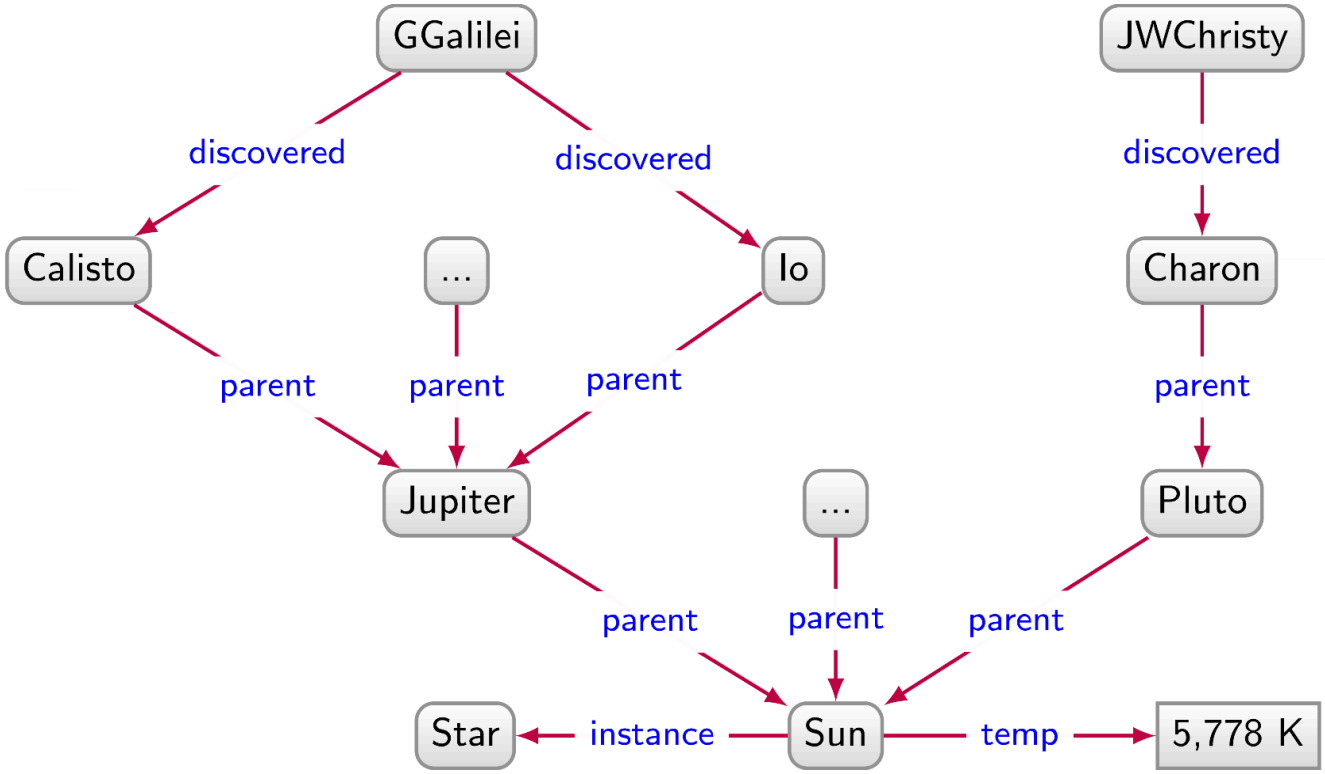

```
MATCH (x1:Person {firstName:"Julie"})-[:knows*]->(x2:Person)
MATCH (x2)-[:likes]->()-[:hasTag]->()-[:hasFollower]->(x1)
RETURN x2.firstName
```

???

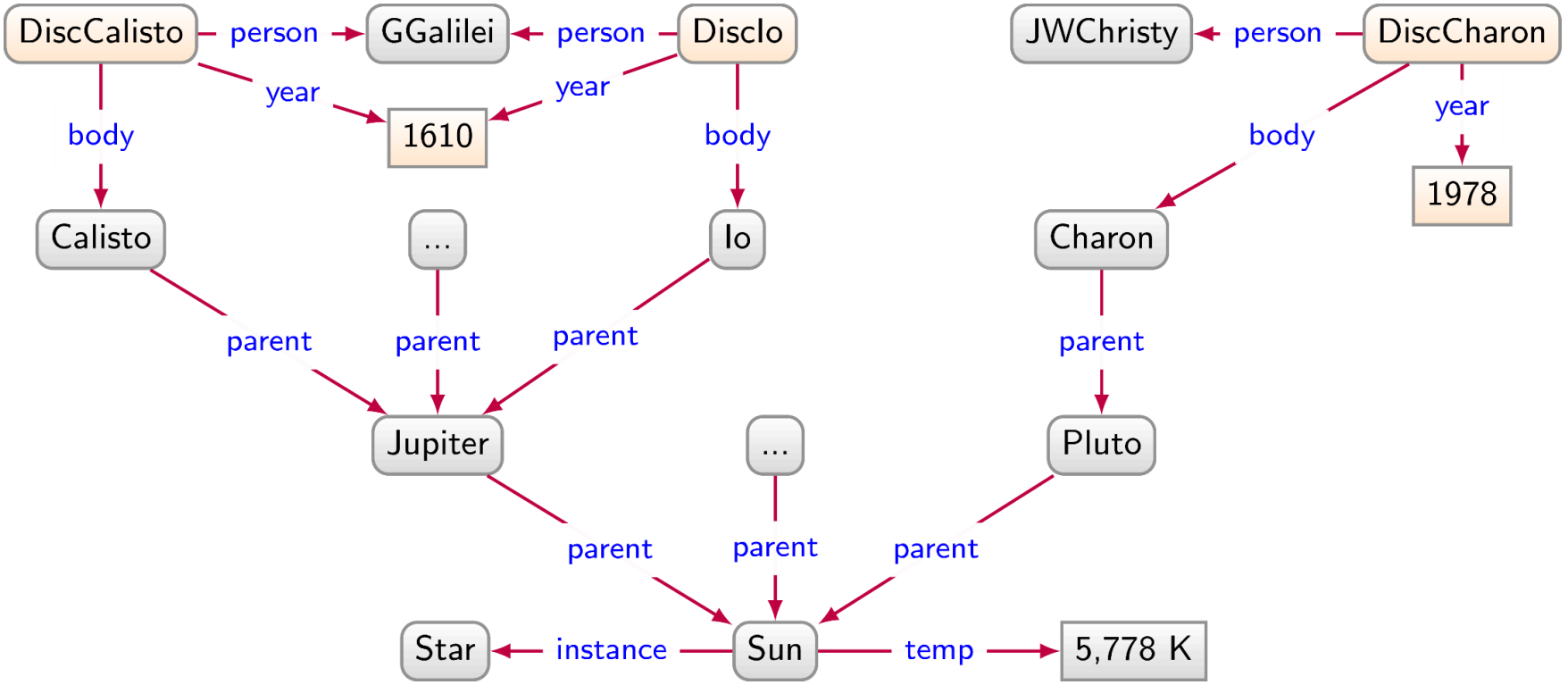
Why do we need Property Graphs?

Directed Labelled Graph

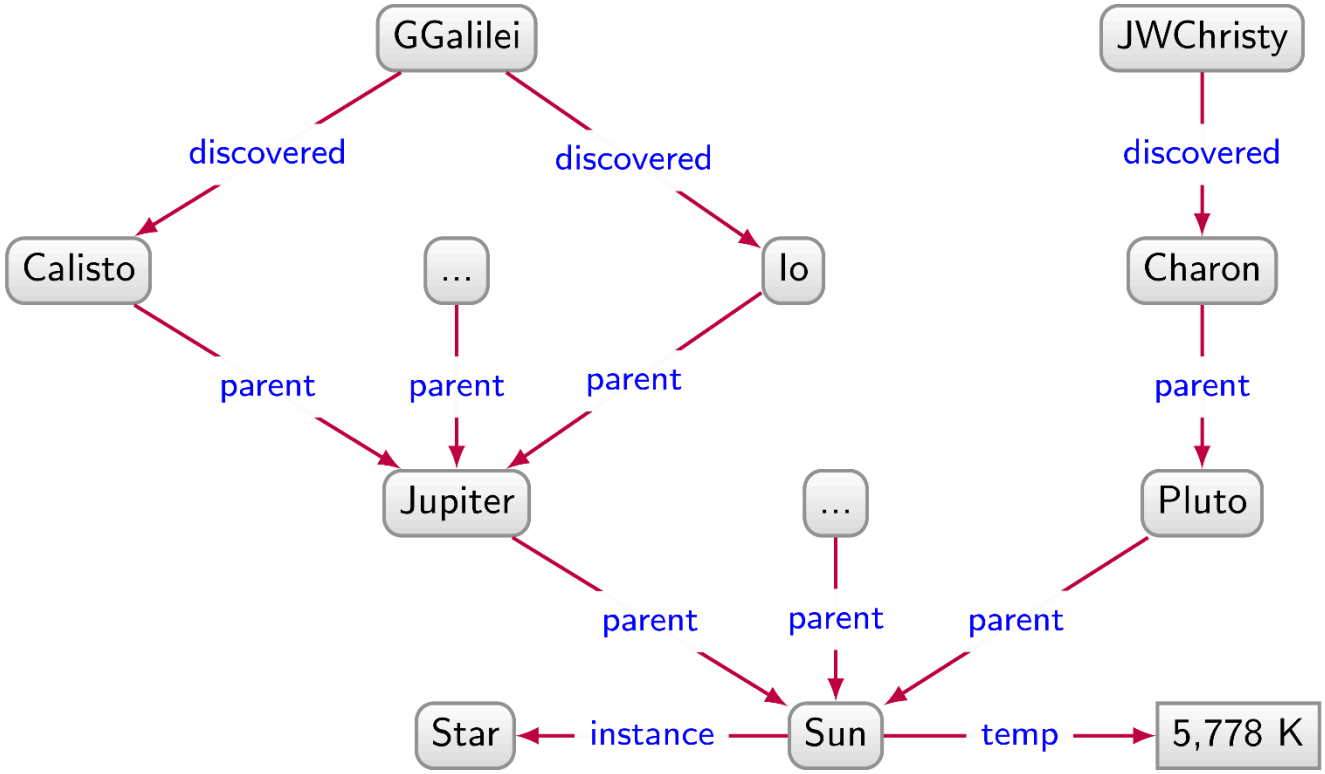
How can we say that Galileo Galilei discovered Calisto and Io in 1610 while James W. Christy discovered Charon in 1978?



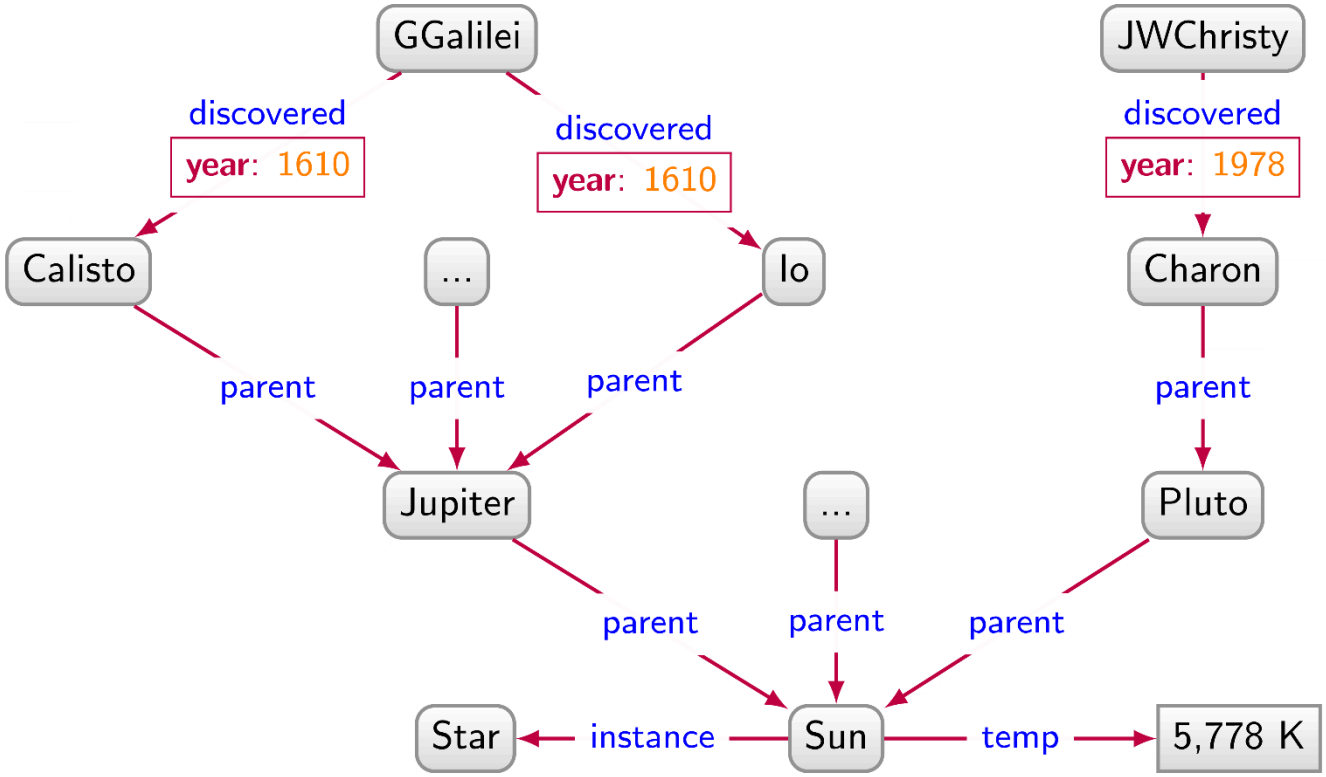
Directed Labelled Graph



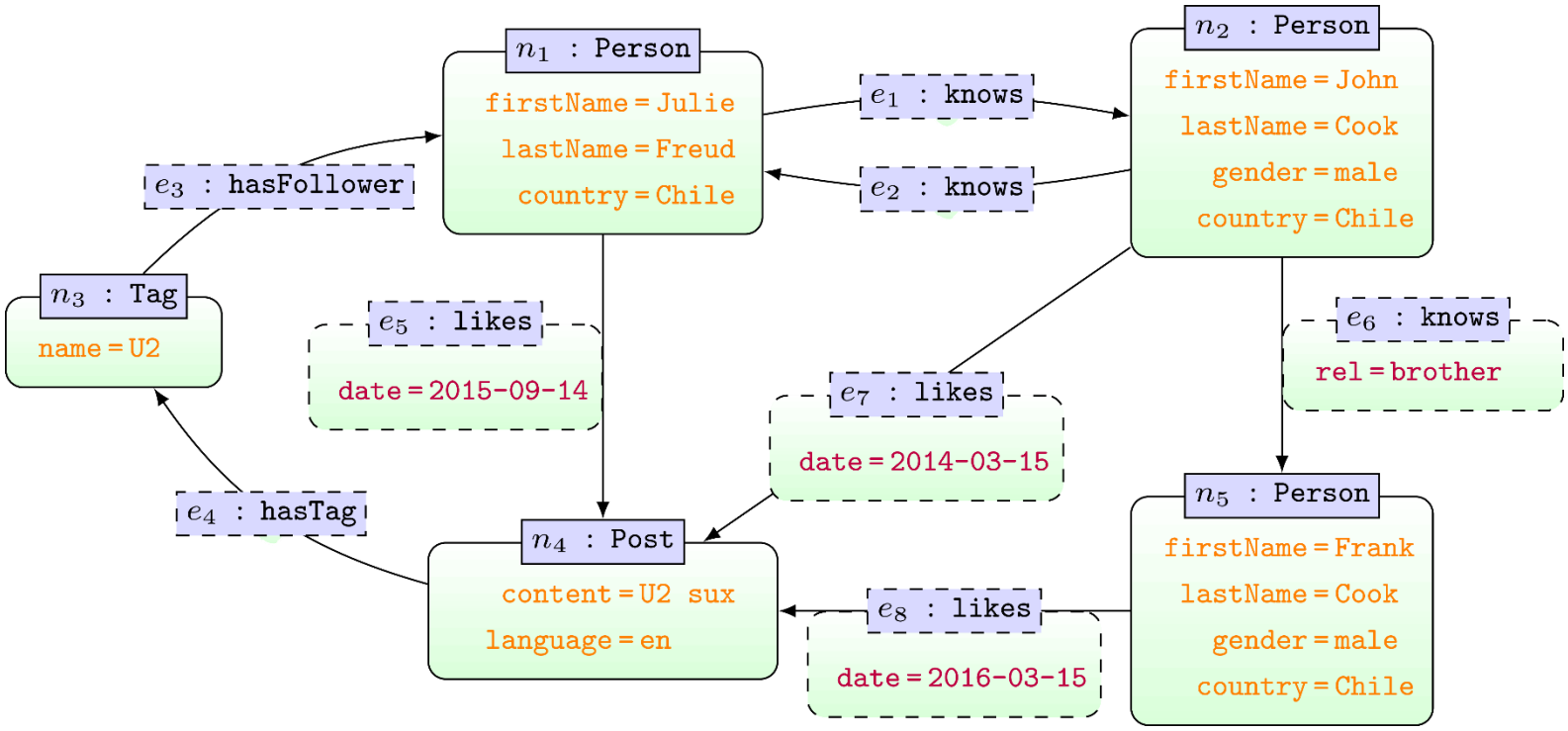
Wouldn't it have been nice to simply ...



Wouldn't it have been nice to simply ...

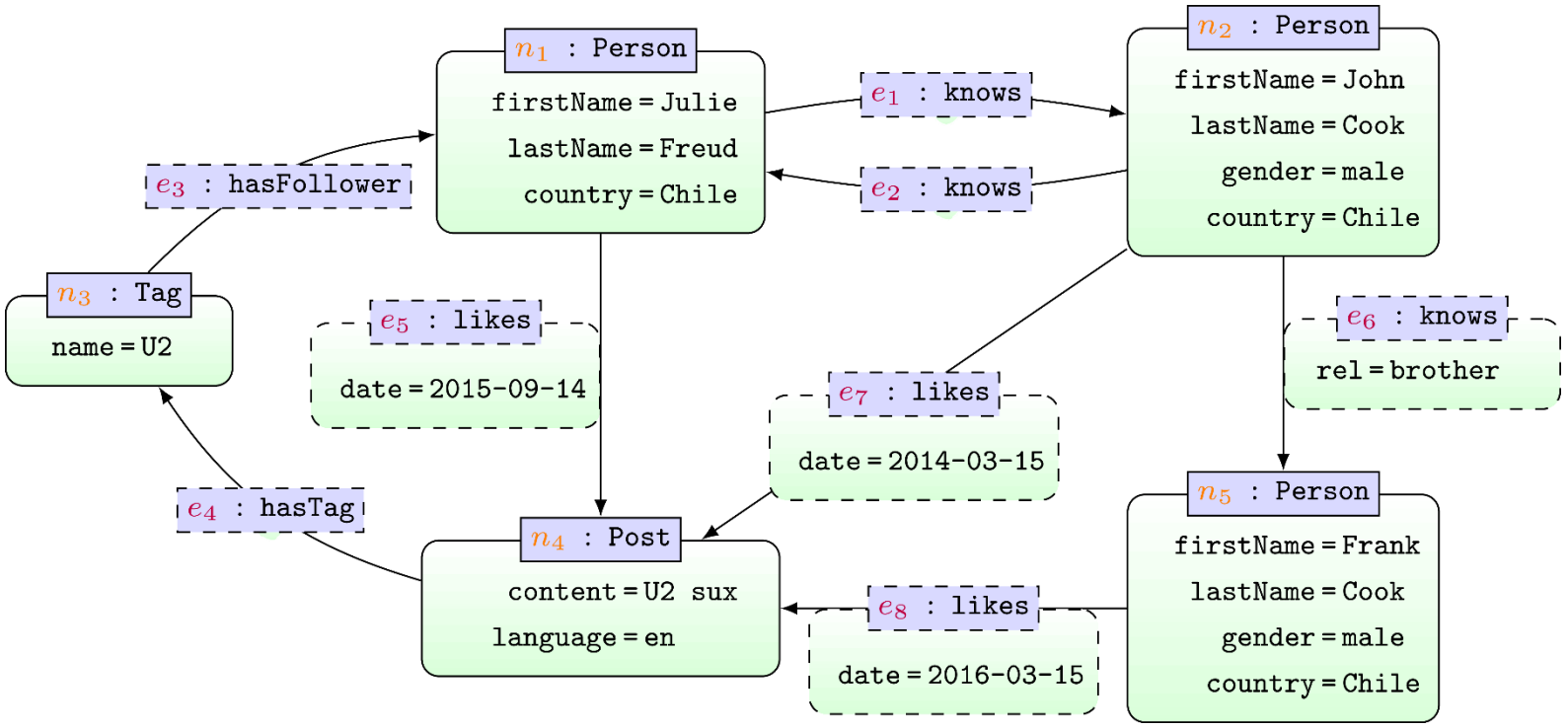


Property Graphs ...



... attributes on nodes and edges

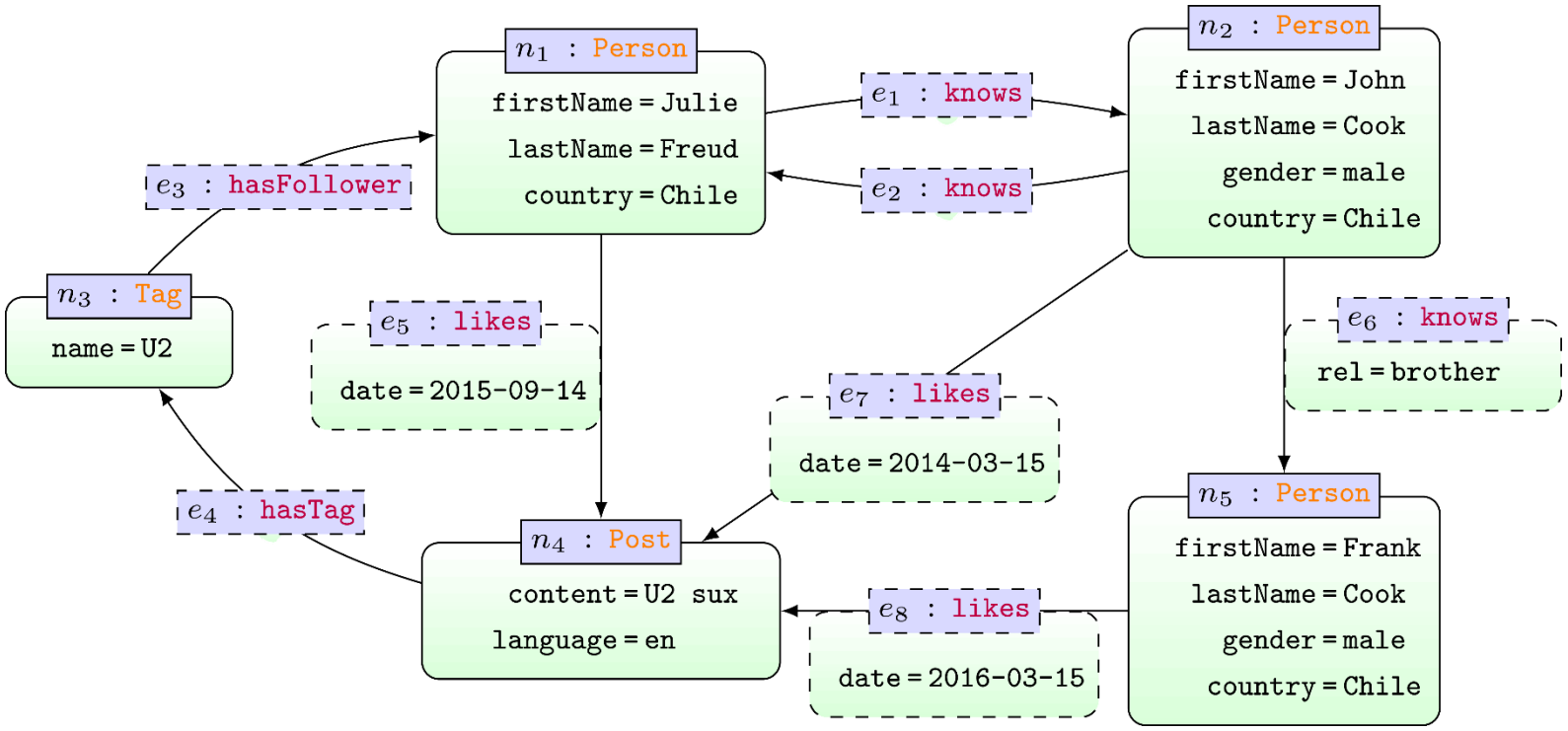
Property Graphs



... attributes on nodes and edges

... IDs on nodes and edges

Property Graphs



... attributes on nodes and edges

... IDs on nodes and edges

... labels on nodes and edges

Popular Graph Databases

DB-Engines Ranking of Graph DBMS

The DB-Engines Ranking ranks database management systems according to their popularity. The ranking is updated monthly.

trend chart

This is a partial list of the complete ranking showing only graph DBMS.

Read more about the method of calculating the scores.

31 systems in ranking, May 2018

Rank			DBMS	Database Model	Score		
May 2018	Apr 2018	May 2017			May 2018	Apr 2018	May 2017
1.	1.	1.	Neo4j	Graph DBMS	40.58	-0.32	+4.44
2.	2.	↑ 4.	Microsoft Azure Cosmos DB	Multi-model	17.54	+0.35	+12.70
3.	3.		Datastax Enterprise	Multi-model	7.38	-0.09	
4.	4.	↓ 2.	OrientDB	Multi-model	5.25	-0.39	-0.49
5.	5.	5.	ArangoDB	Multi-model	3.70	-0.10	+0.75
6.	6.	6.	Virtuoso	Multi-model	1.79	-0.01	-0.27
7.	7.	7.	Giraph	Graph DBMS	0.98	-0.06	-0.11
8.	8.		Amazon Neptune	Multi-model	0.71	+0.02	
9.	9.	↓ 8.	AllegroGraph	Multi-model	0.58	+0.00	-0.02
10.	10.	↓ 9.	Stardog	Multi-model	0.51	-0.02	+0.00
11.	11.	↓ 10.	GraphDB	Multi-model	0.46	-0.00	-0.04
12.	↑ 14.	↑ 19.	JanusGraph	Graph DBMS	0.41	+0.12	+0.29
13.	↓ 12.	↑ 16.	Graph Engine	Multi-model	0.36	-0.04	+0.18
14.	↓ 13.	↓ 11.	Sqrrl	Multi-model	0.33	-0.06	-0.13
15.	15.	↑ 22.	Sparksee	Graph DBMS	0.19	-0.02	+0.14
16.	16.		TigerGraph	Graph DBMS	0.17	-0.01	
17.	↑ 20.	↓ 14.	Blazegraph	Multi-model	0.14	+0.01	-0.13
18.	18.	↓ 12.	Dgraph	Graph DBMS	0.14	+0.00	-0.15
19.	↓ 17.	↓ 17.	HyperGraphDB	Graph DBMS	0.14	-0.01	-0.02
20.	↓ 19.	↓ 15.	FlockDB	Graph DBMS	0.13	+0.00	-0.06
21.	↑ 23.	↓ 13.	InfiniteGraph	Graph DBMS	0.13	+0.02	-0.15
22.	22.	↓ 20.	FaunaDB	Multi-model	0.11	+0.00	+0.05
23.	↑ 24.	23.	VelocityDB	Multi-model	0.10	+0.02	+0.06
24.	↓ 21.	↓ 18.	InfoGrid	Graph DBMS	0.10	-0.02	-0.03
25.	↑ 26.	25.	AgensGraph	Multi-model	0.04	+0.01	+0.03

DB-Engines Ranking

The DB-Engines Ranking ranks database management systems according to their popularity. The ranking is updated monthly.

Read more about the method of calculating the scores.

trend chart

342 systems in ranking, May 2018

Rank			DBMS	Database Model	Score		
May 2018	Apr 2018	May 2017			May 2018	Apr 2018	May 2017
1.	1.	1.	Oracle	Relational DBMS	1290.42	+0.63	-63.90
2.	2.	2.	MySQL	Relational DBMS	1223.34	-3.06	-116.69
3.	3.	3.	Microsoft SQL Server	Relational DBMS	1085.84	-9.67	-127.96
4.	4.	4.	PostgreSQL	Relational DBMS	400.90	+5.43	+34.99
5.	5.	5.	MongoDB	Document store	342.11	+0.70	+10.53
6.	6.	6.	DB2	Relational DBMS	185.61	-3.34	-3.23
7.	↑ 9.	↑ 9.	Redis	Key-value store	135.35	+5.24	+17.90
8.	↓ 7.	↓ 7.	Microsoft Access	Relational DBMS	133.11	+0.89	+3.24
9.	↓ 8.	↑ 11.	Elasticsearch	Search engine	130.44	-0.92	+21.62
10.	10.	↓ 8.	Cassandra	Wide column store	117.83	-1.26	-5.28
11.	11.	↓ 10.	SQLite	Relational DBMS	115.45	-0.53	-0.61
12.	12.	12.	Teradata	Relational DBMS	74.41	+0.74	-1.91
13.	13.	↑ 16.	Splunk	Search engine	65.09	+0.04	+8.40
14.	14.	↑ 18.	MariaDB	Relational DBMS	64.99	+0.44	+14.01
15.	15.	↓ 14.	Solr	Search engine	61.51	-1.70	-2.26
16.	16.	↓ 13.	SAP Adaptive Server	Relational DBMS	61.51	-0.12	-6.24
17.	17.	↓ 15.	HBase	Wide column store	59.95	+0.26	+0.44
18.	18.	↑ 20.	Hive	Relational DBMS	56.97	-0.43	+13.49
19.	19.	↓ 17.	FileMaker	Relational DBMS	54.67	-0.33	-1.81
20.	20.	↓ 19.	SAP HANA	Relational DBMS	48.37	-0.52	-0.68
21.	21.	↑ 22.	Amazon DynamoDB	Multi-model	44.19	+1.05	+10.99
22.	22.	↓ 21.	Neo4j	Graph DBMS	40.58	-0.32	+4.44
23.	23.	↑ 24.	Memcached	Key-value store	33.56	-0.23	+4.15
24.	24.	↓ 23.	Couchbase	Document store	32.41	+0.07	+0.16
25.	25.	25.	Informix	Relational DBMS	25.79	-0.82	-2.44

Neo4j

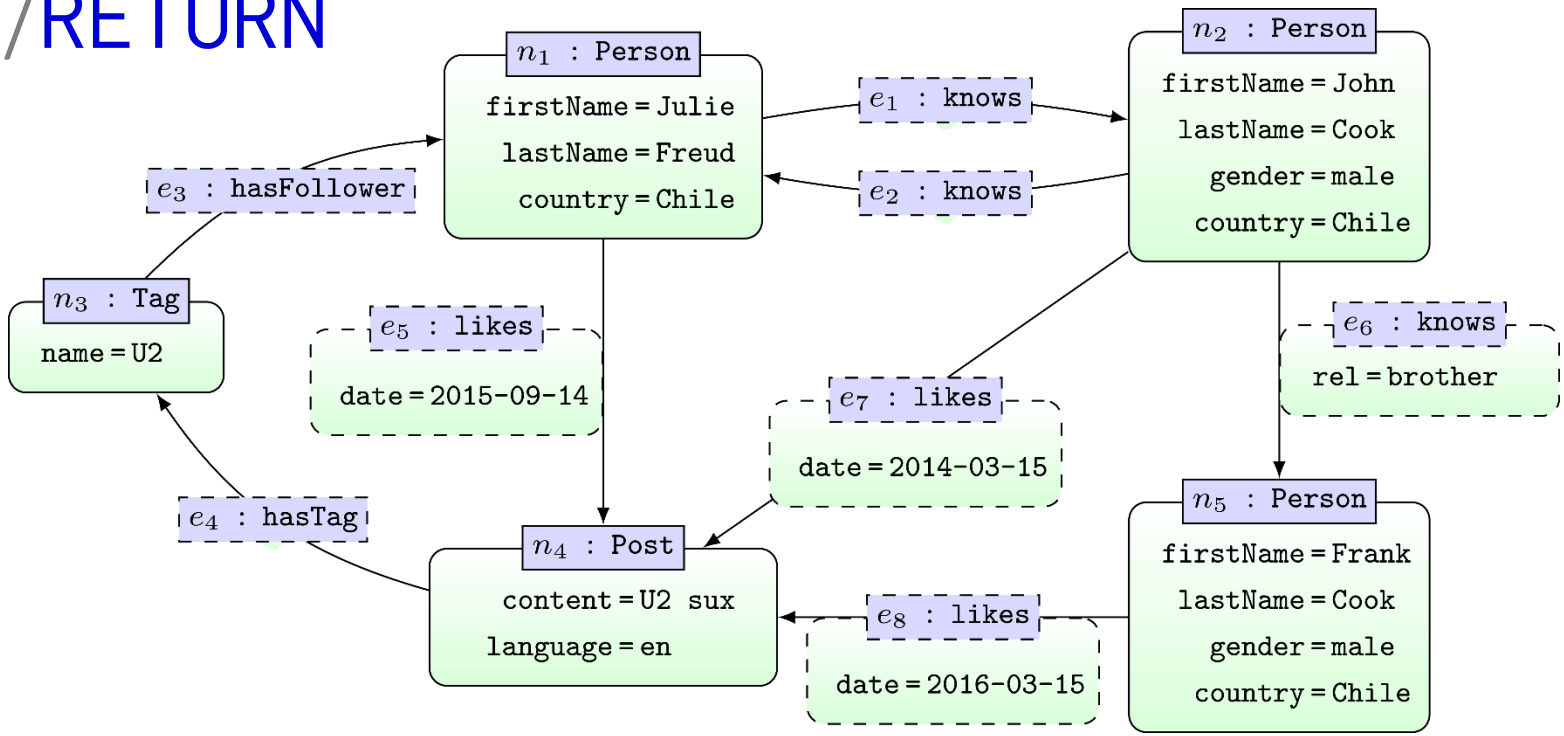
Neo4j Graph Database

- Data Model: Property Graphs
- Query Language: Cypher
- Scripting Language: Gremlin
- Licence: Open Source (Single Machine)
  Commercial (Cluster Edition)

Cypher:
MATCH/RETURN

https://neo4j.com/docs/developer-manual/3.4/cypher/clauses/match/

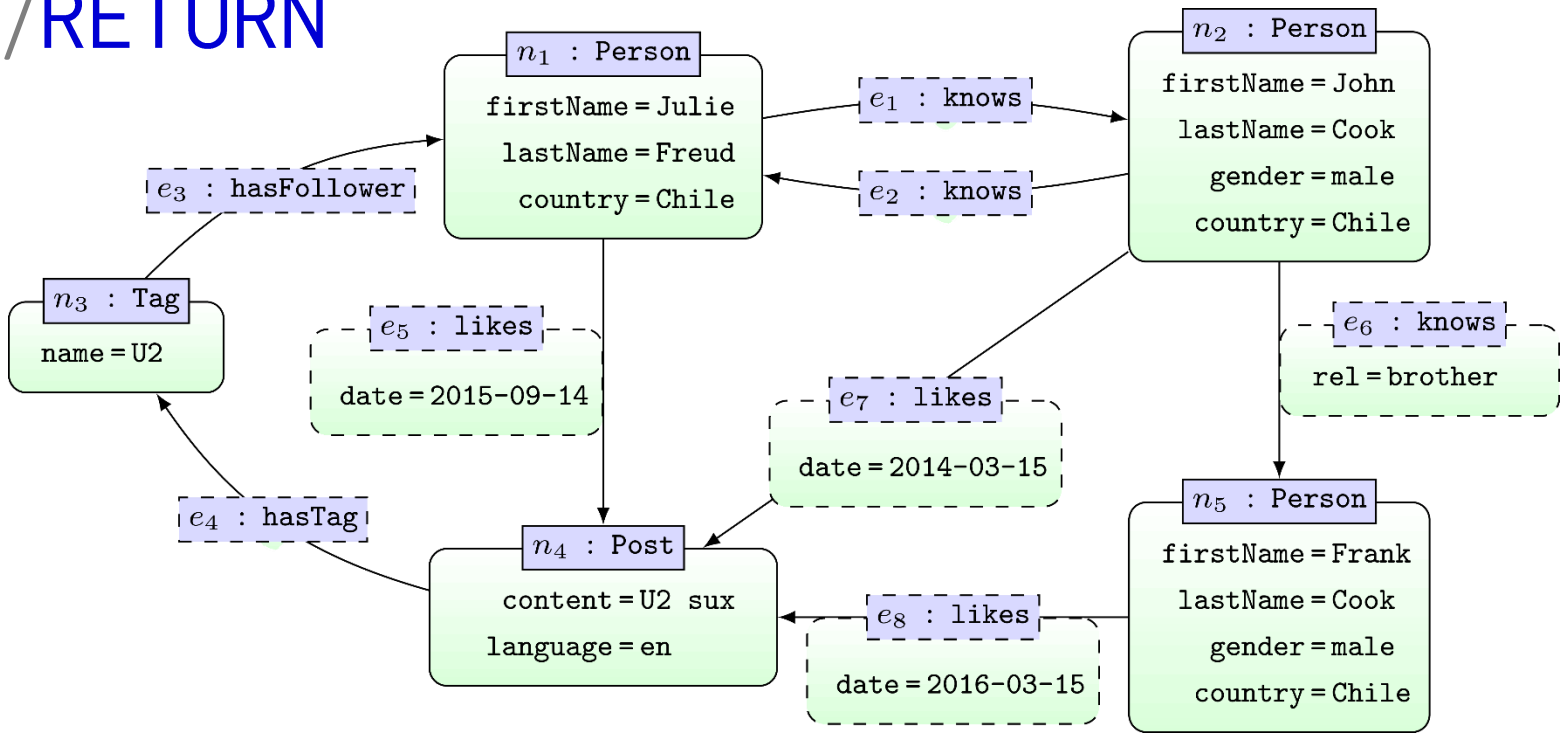
MATCH/RETURN




```
MATCH (x:Post)
RETURN x
```

x
(:Post {content: "U2 sux", language: "en"})

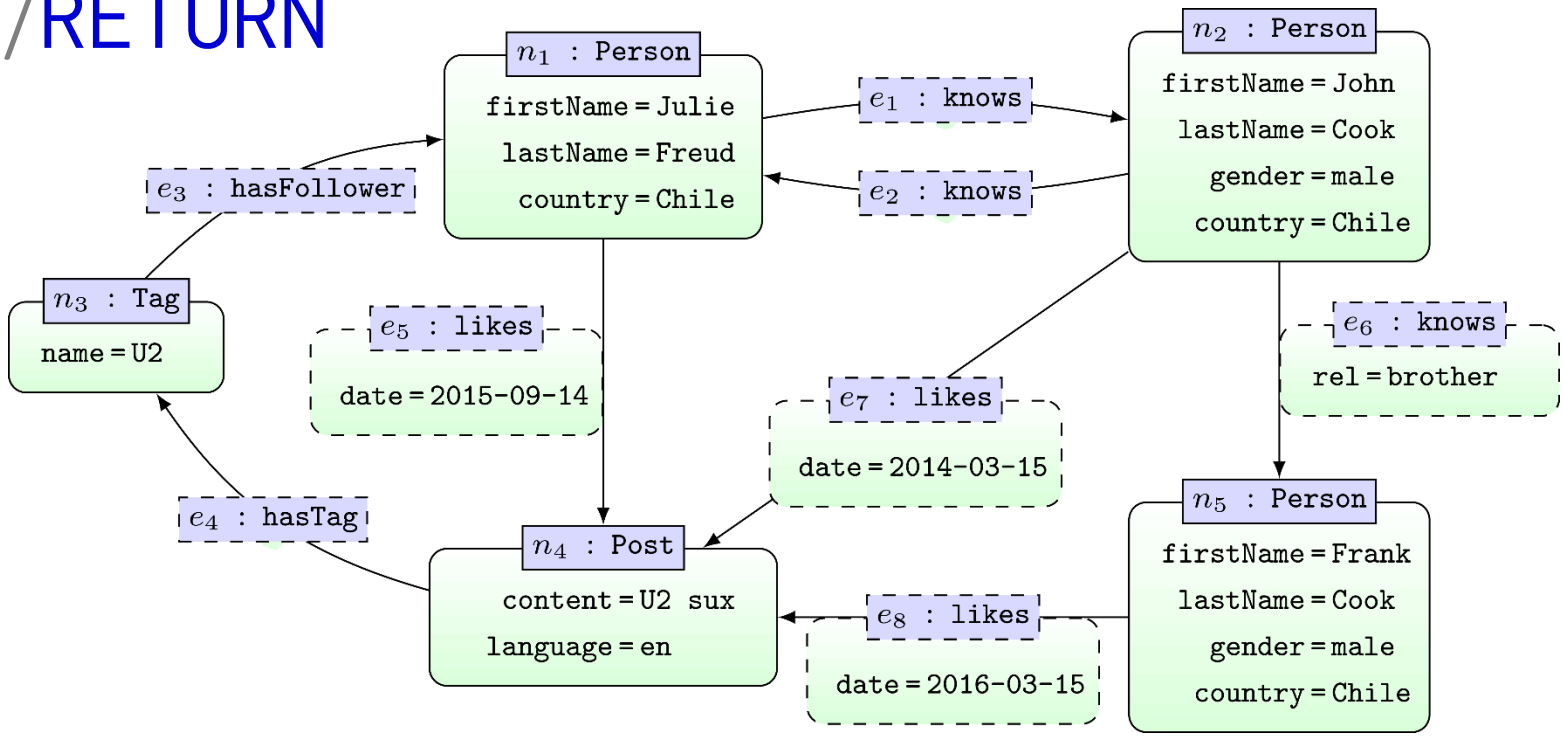
MATCH/RETURN




```
MATCH (x:Person)
RETURN x.firstName
```

x.firstName
Julie
John
Frank

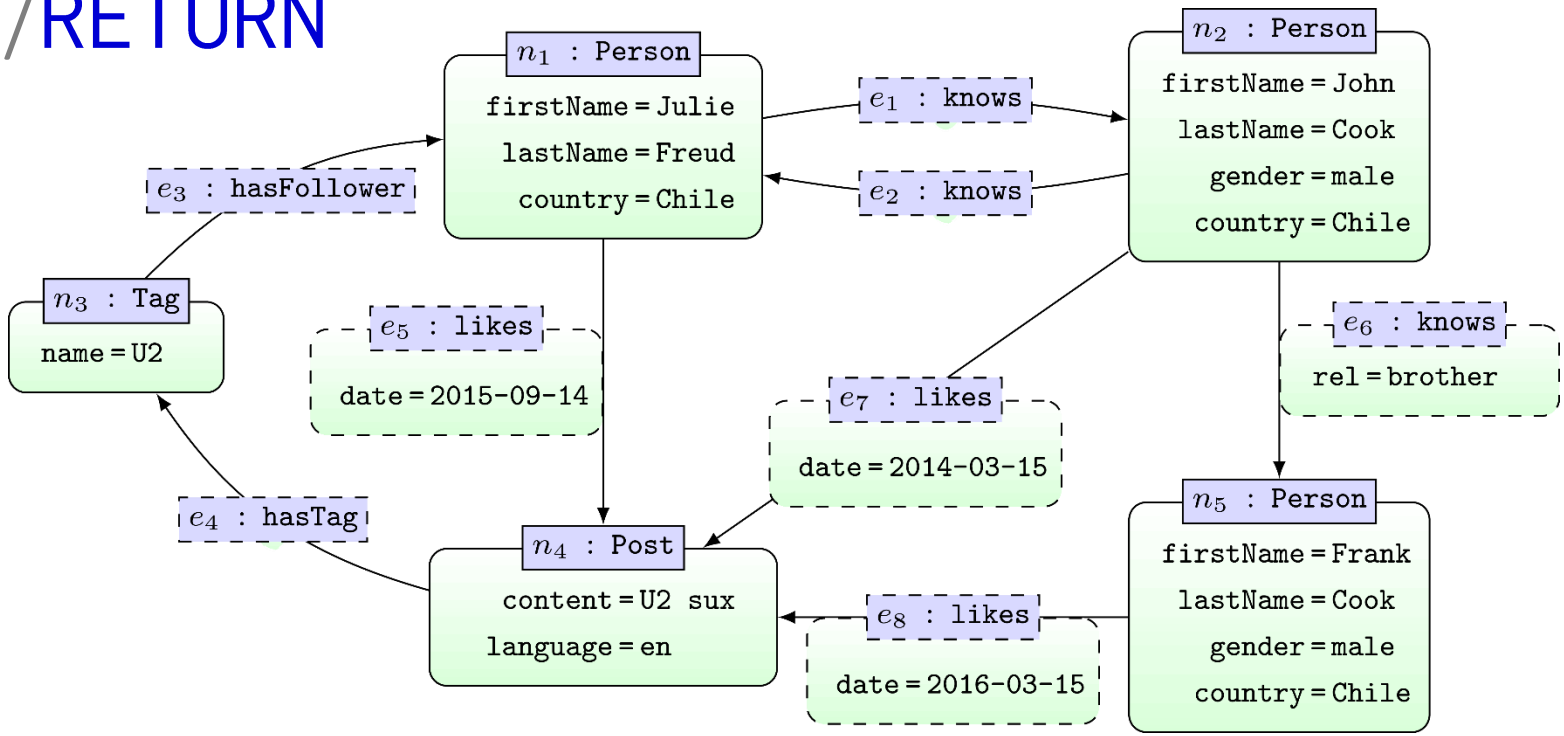
MATCH/RETURN




```
MATCH (x:Person {gender: "male", lastName: "Cook"})
RETURN x.firstName
```

x.firstName
John
Frank

MATCH/RETURN

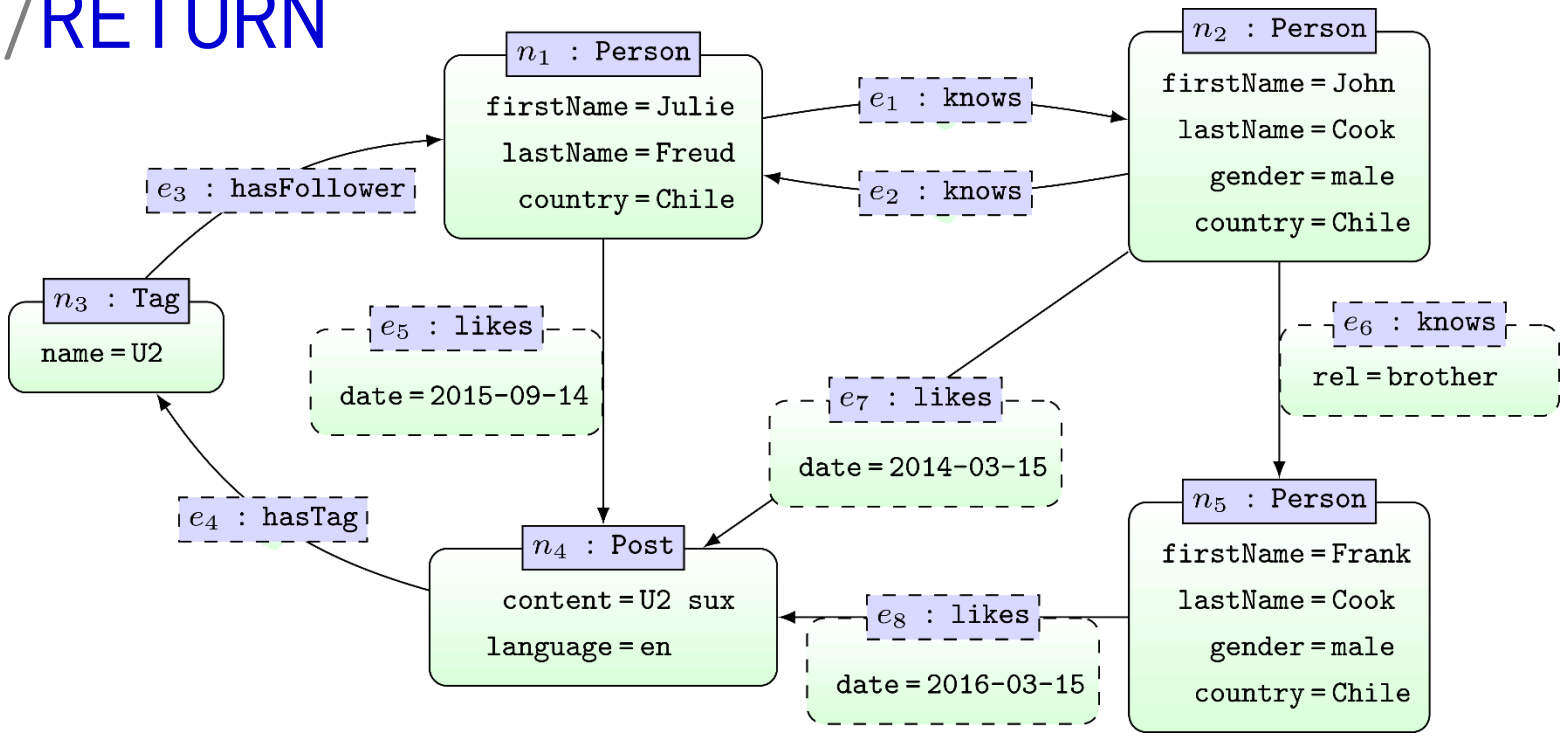



```
MATCH (x:Person)
RETURN x.firstName,x.gender
```

x.firstName	x.gender
Julie	
John	male
Frank	male

... matching nodes returned with blank attributes

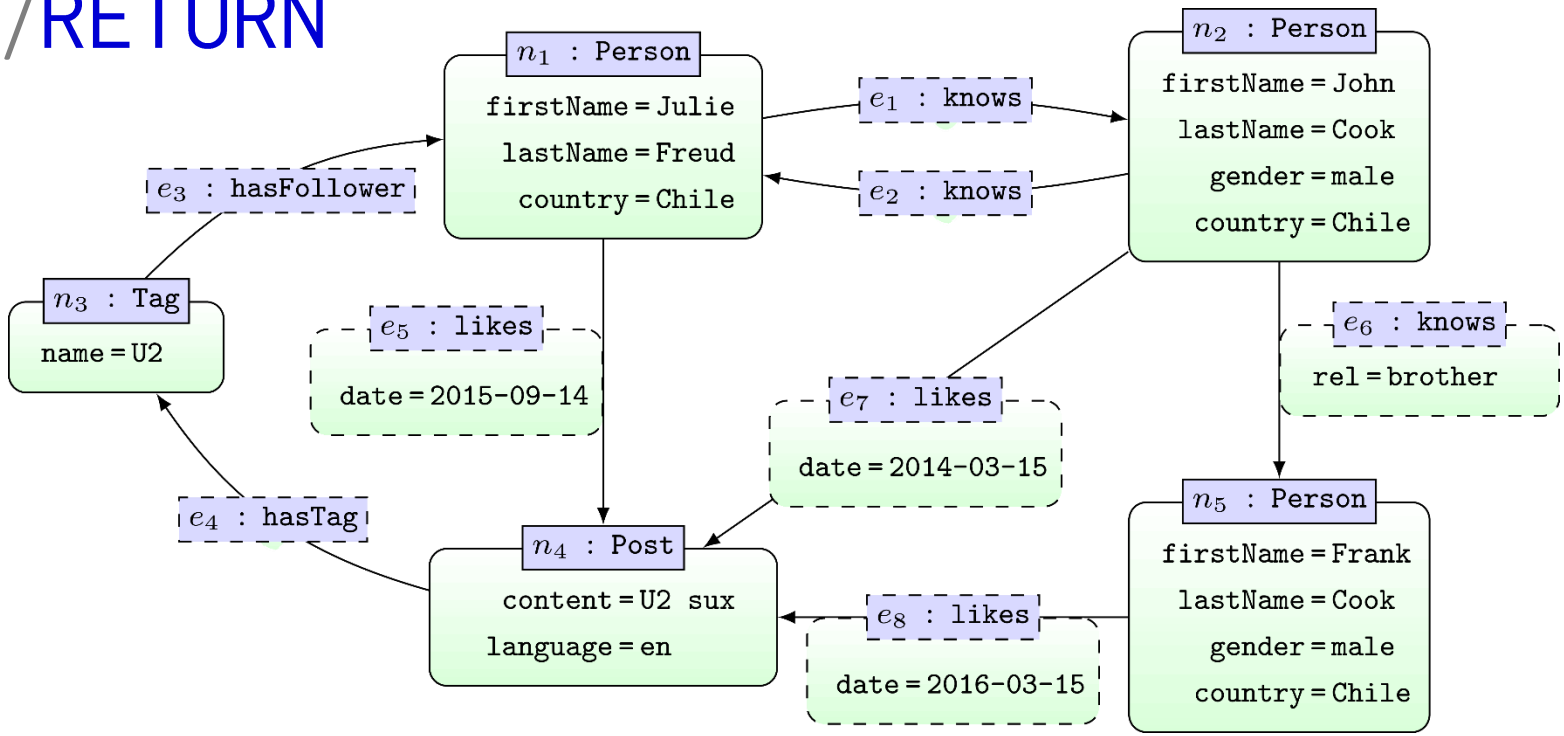
MATCH/RETURN




```
MATCH (x)
RETURN x.firstName,x.gender
```

x.firstName	x.gender
Julie	
John	male
Frank	male

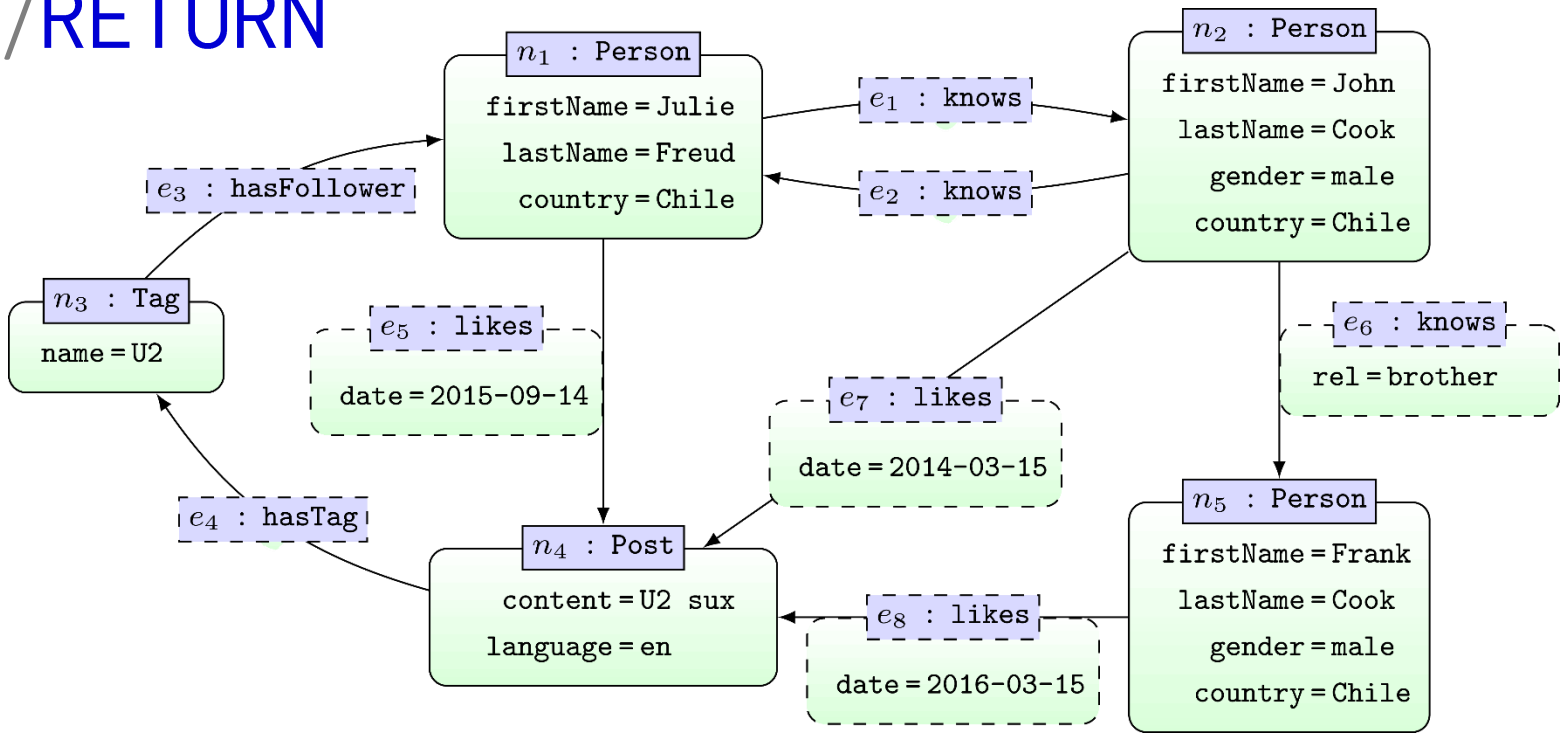
MATCH/RETURN




```
MATCH (:Person)-->(x:Person)
RETURN x.firstName
```

x.firstName
Julie
John
Frank

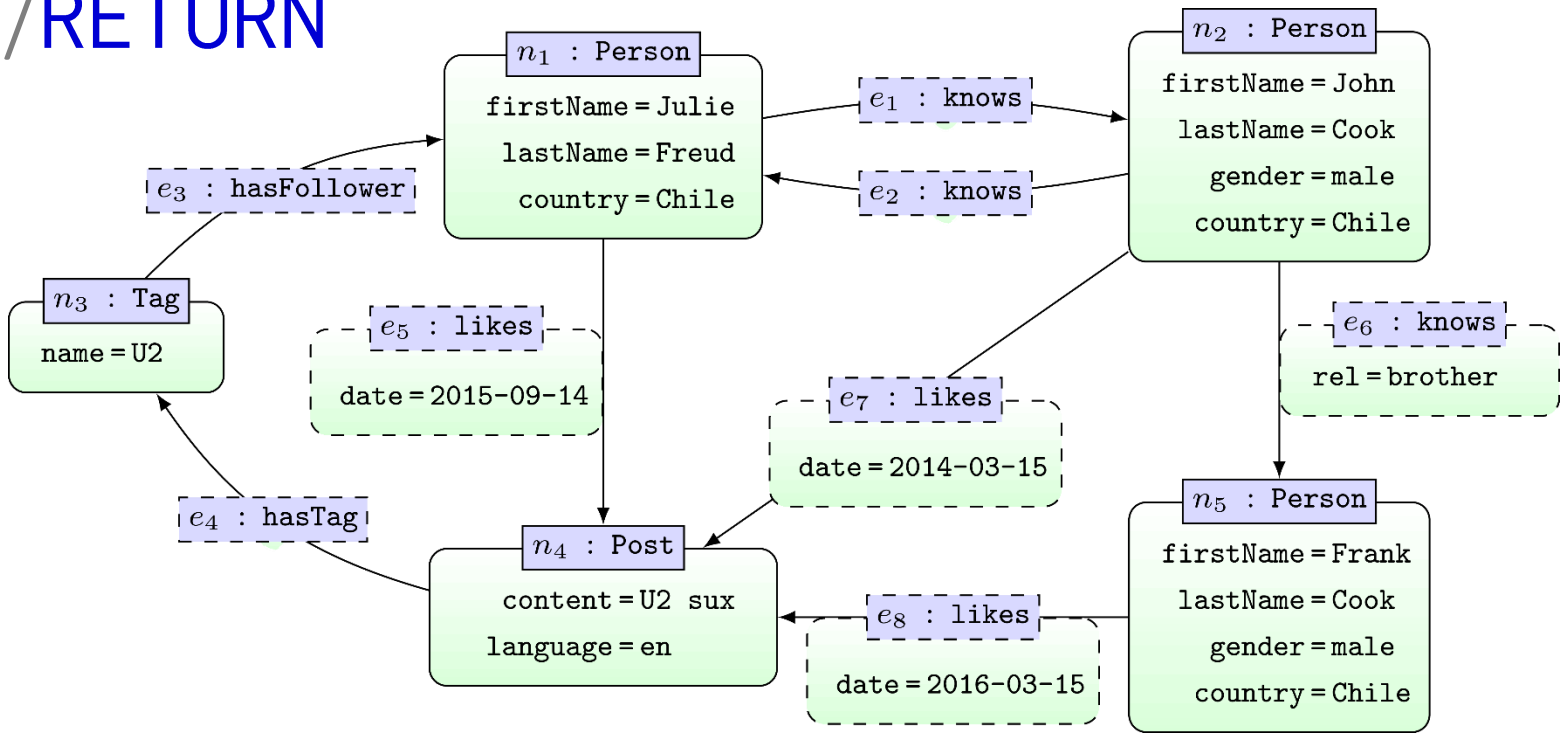
MATCH/RETURN




```
MATCH (x:Person)-->(:Person)
RETURN x.firstName
```

x.firstName
Julie
John

MATCH/RETURN

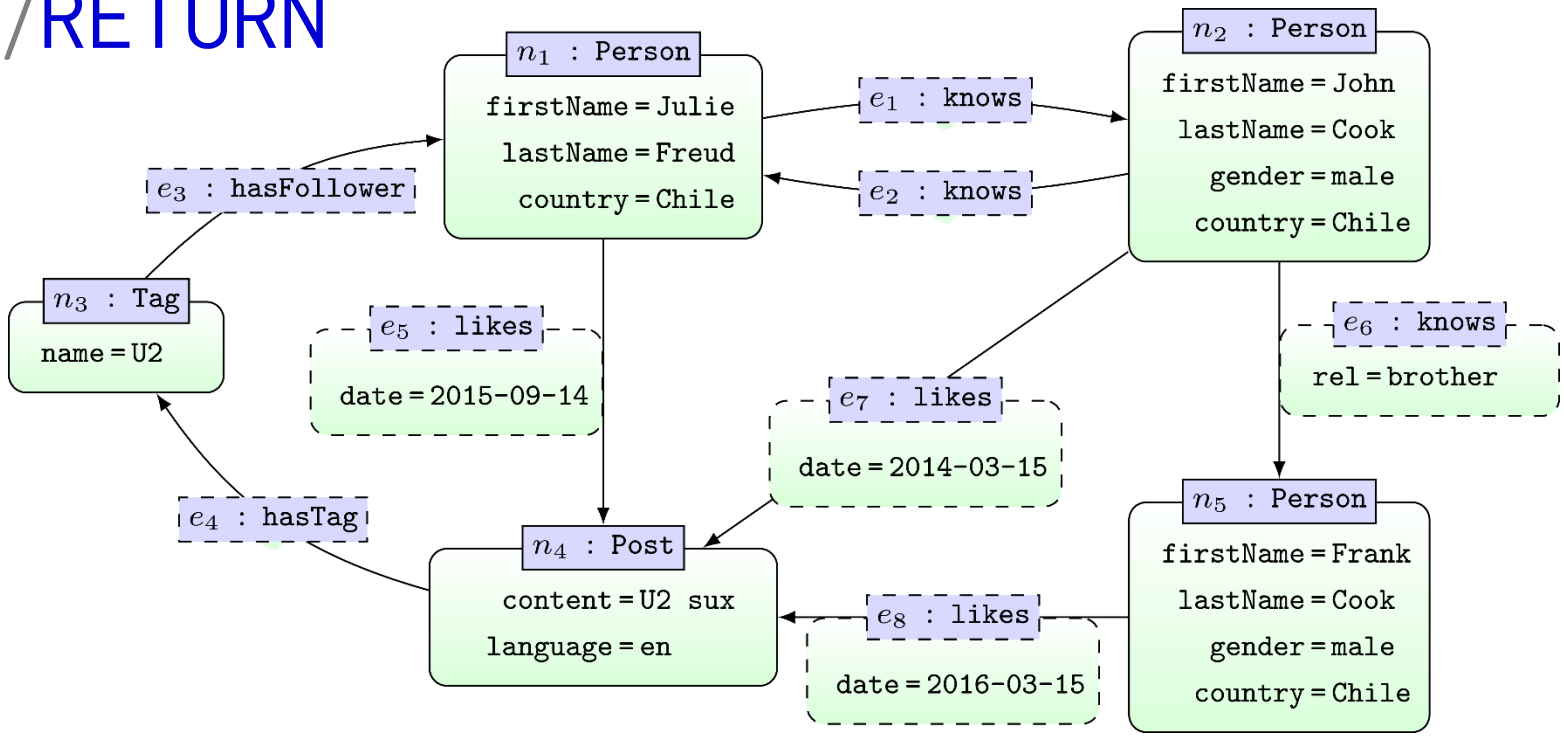



```
MATCH (x:Person)-->()
RETURN x.firstName
```

x.firstName
Julie
Julie
John
John
John
Frank

... multiplicity of results corresponds to number of matches

MATCH/RETURN

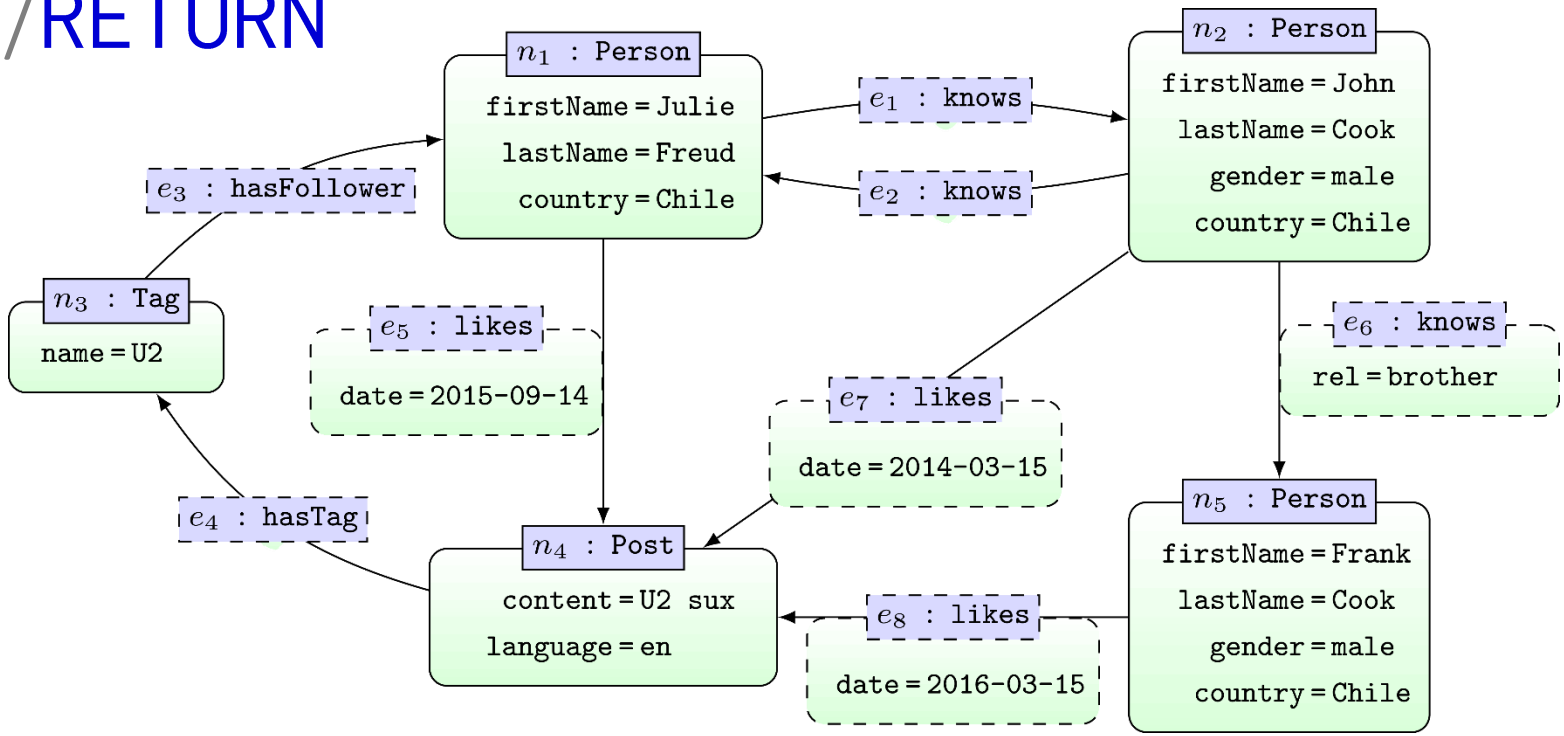



```
MATCH (x:Person)-->()
RETURN DISTINCT x.firstName
```

x.firstName
Julie
John
Frank

... RETURN DISTINCT removes duplicates

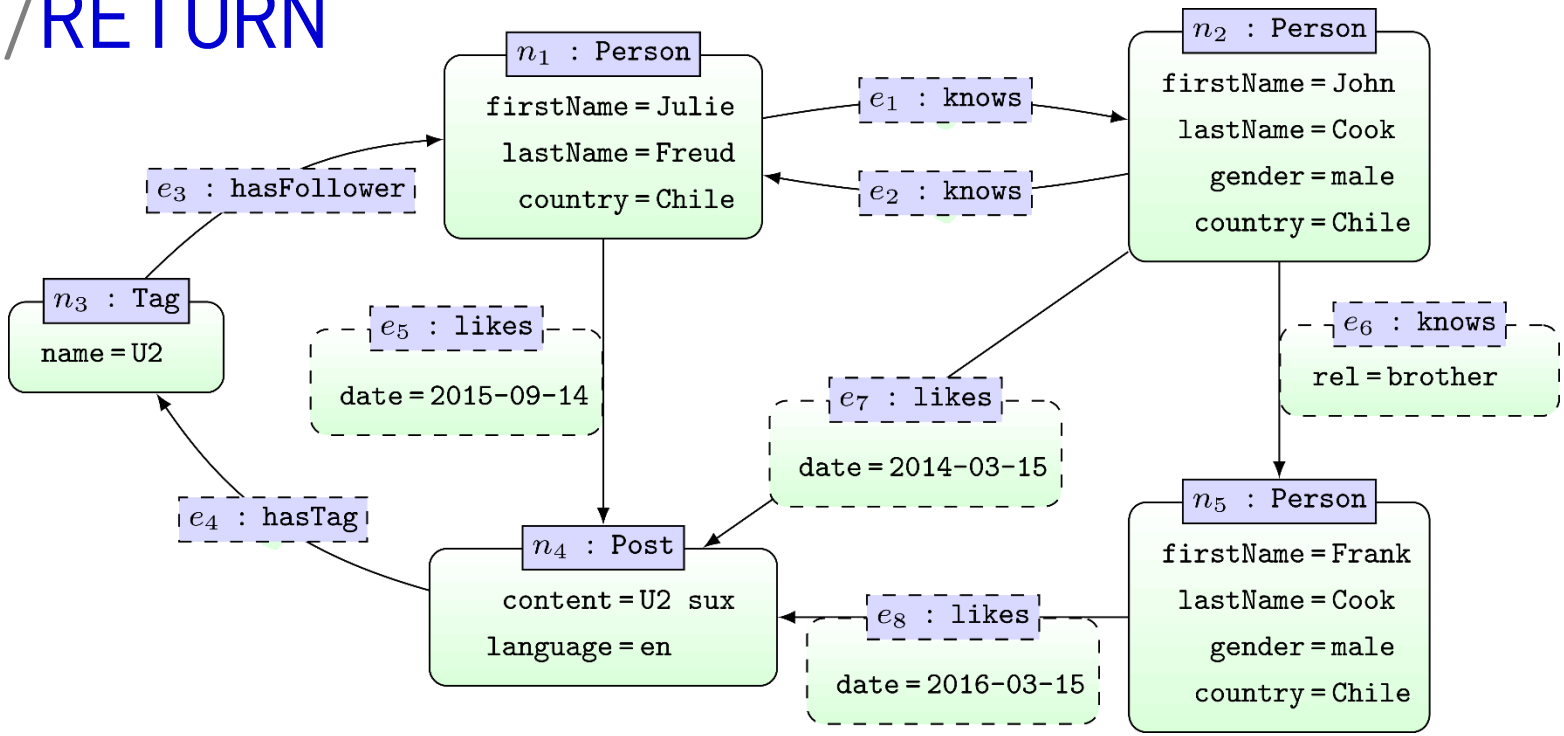
MATCH/RETURN




```
MATCH (x1:Person)-->(x2:Person)
RETURN x1.firstName,x2.firstname
```

x1.firstName	x2.firstName
Julie	John
John	Julie
John	Frank

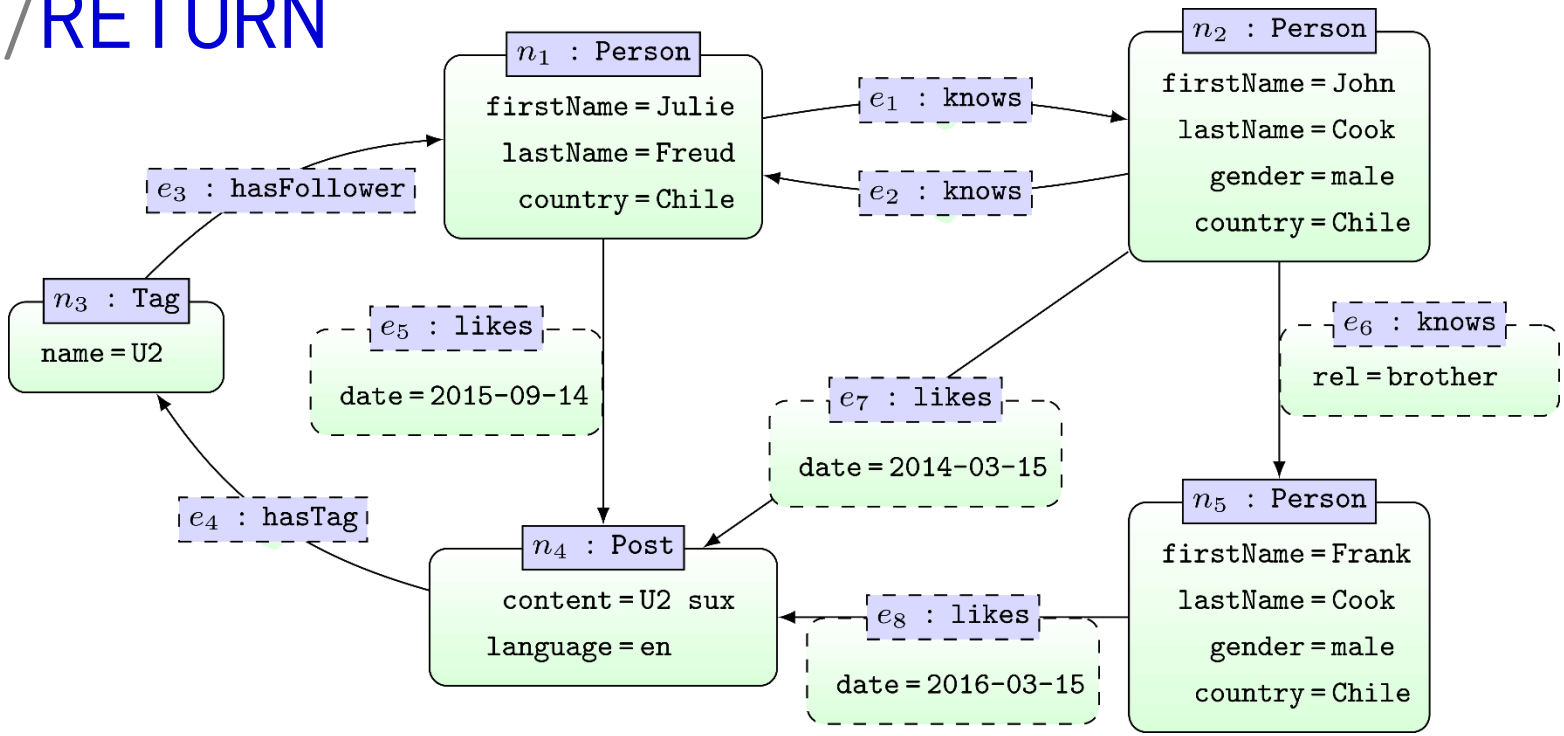
MATCH/RETURN




```
MATCH (x1:Person)-[r]->(x2:Person)
RETURN x1.firstName,x2.firstName,r.rel
```

x1.firstName	x2.firstName	r.rel
Julie	John	
John	Julie	
John	Frank	brother

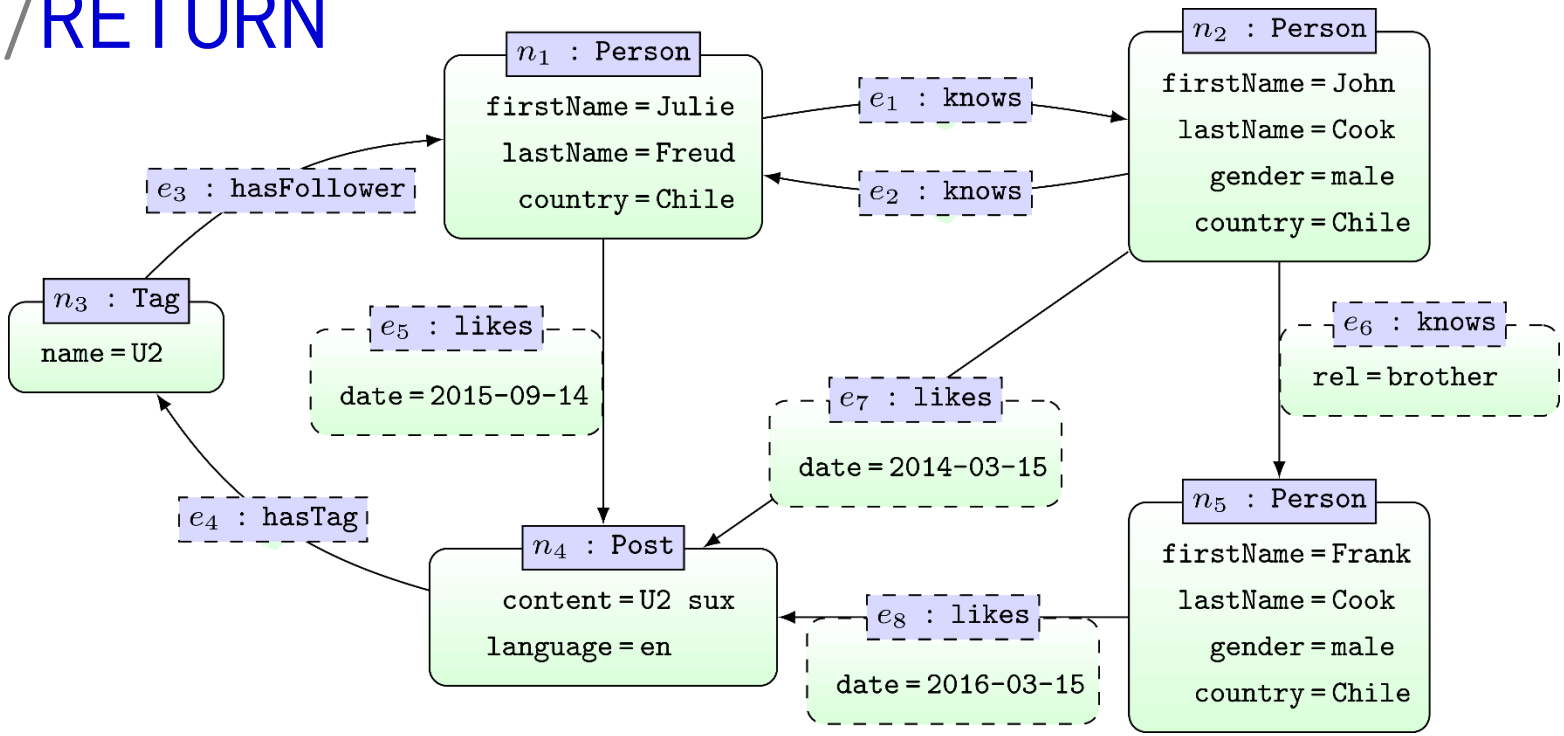
MATCH/RETURN




```
MATCH (x1:Person)-[r]->(x2:Person)
RETURN r
```

r
[:knows]
[:knows]
[:knows {rel: "brother"}]

MATCH/RETURN

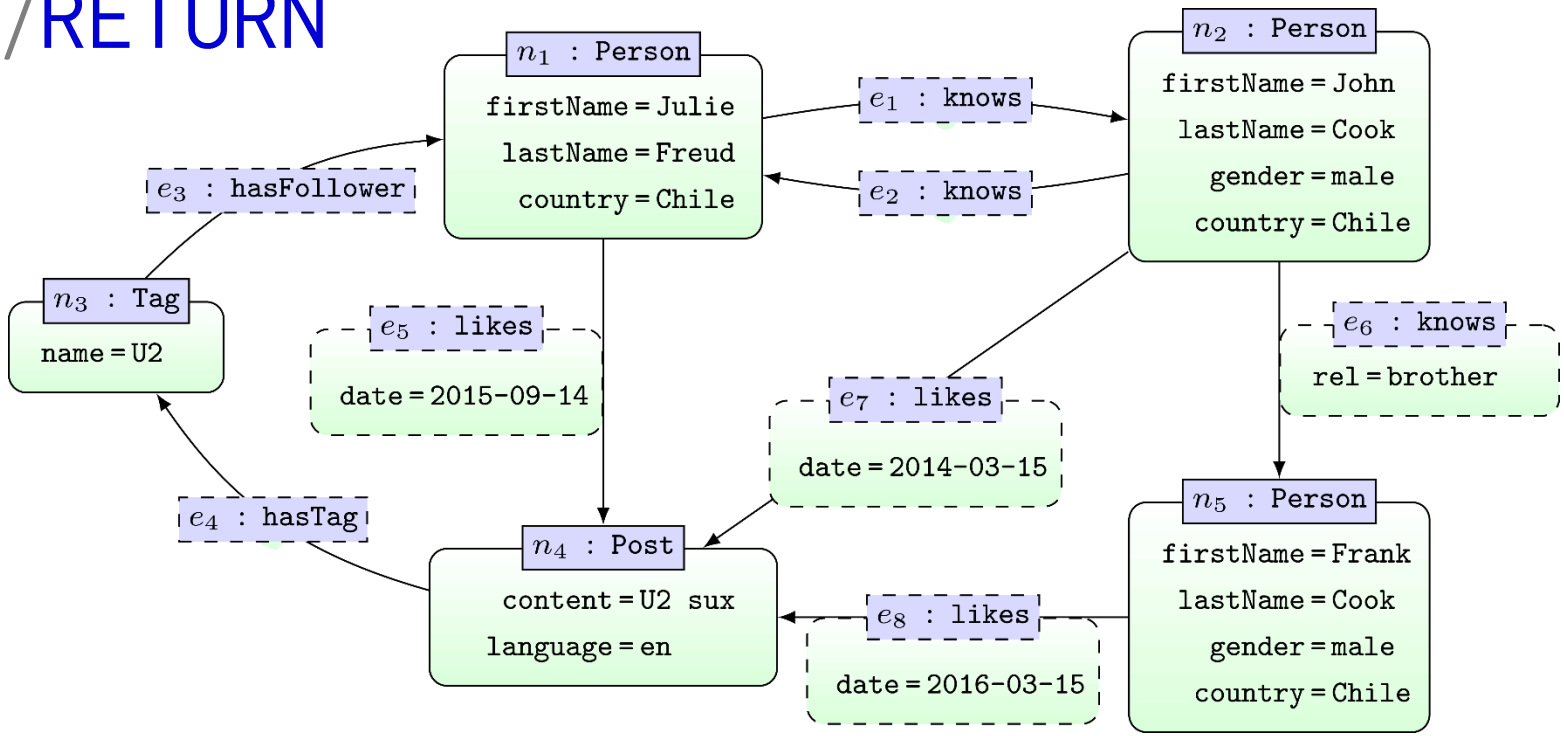



```
MATCH ()<-[:knows]-(y)-[:knows]->()
RETURN y.firstName
```

y.firstName
John
John

... MATCH will not match the same edge twice

MATCH/RETURN

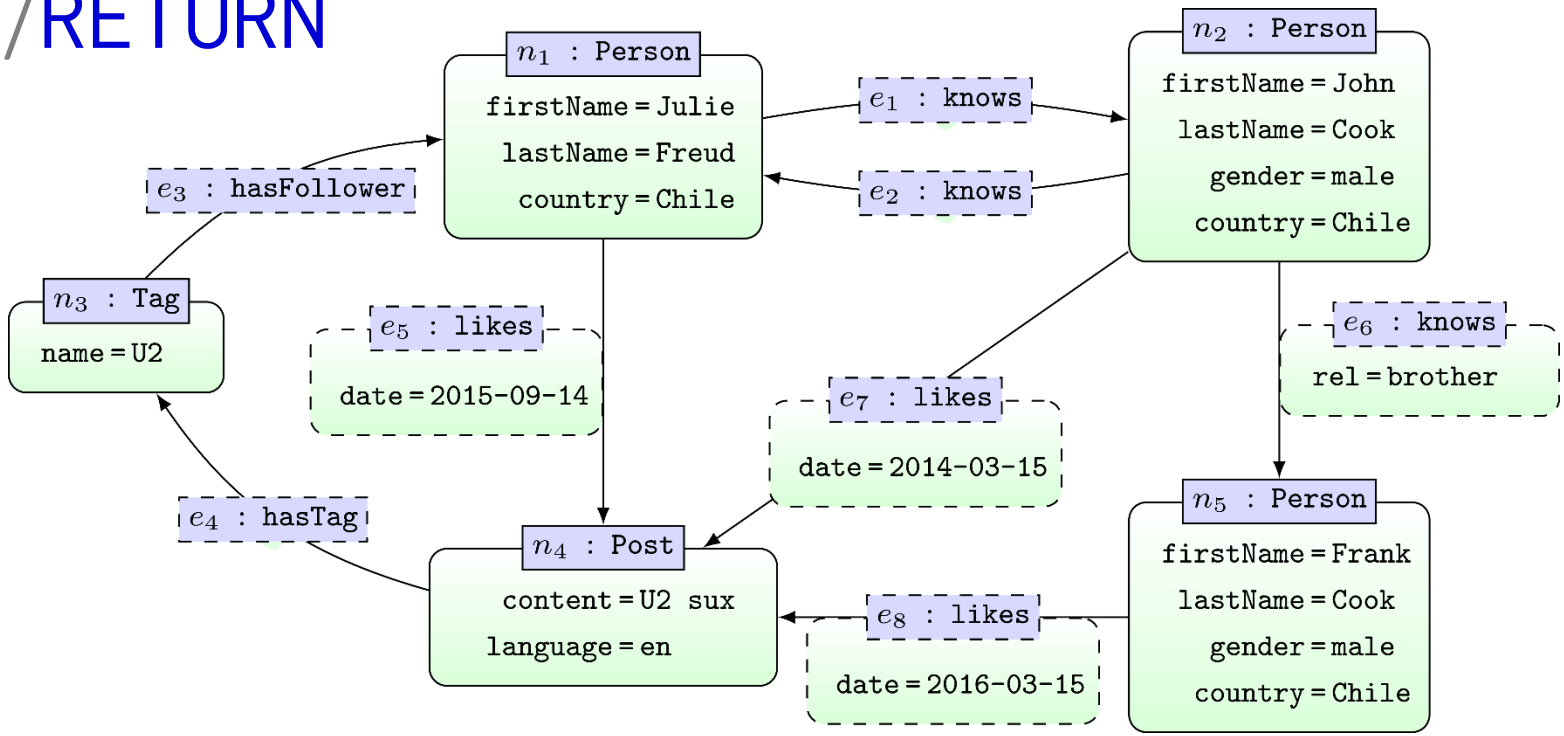



```
MATCH ()-[:knows]->(y)-[:knows]->()
RETURN y.firstName
```

y.firstName
Julie
John
John

... MATCH will match same node twice

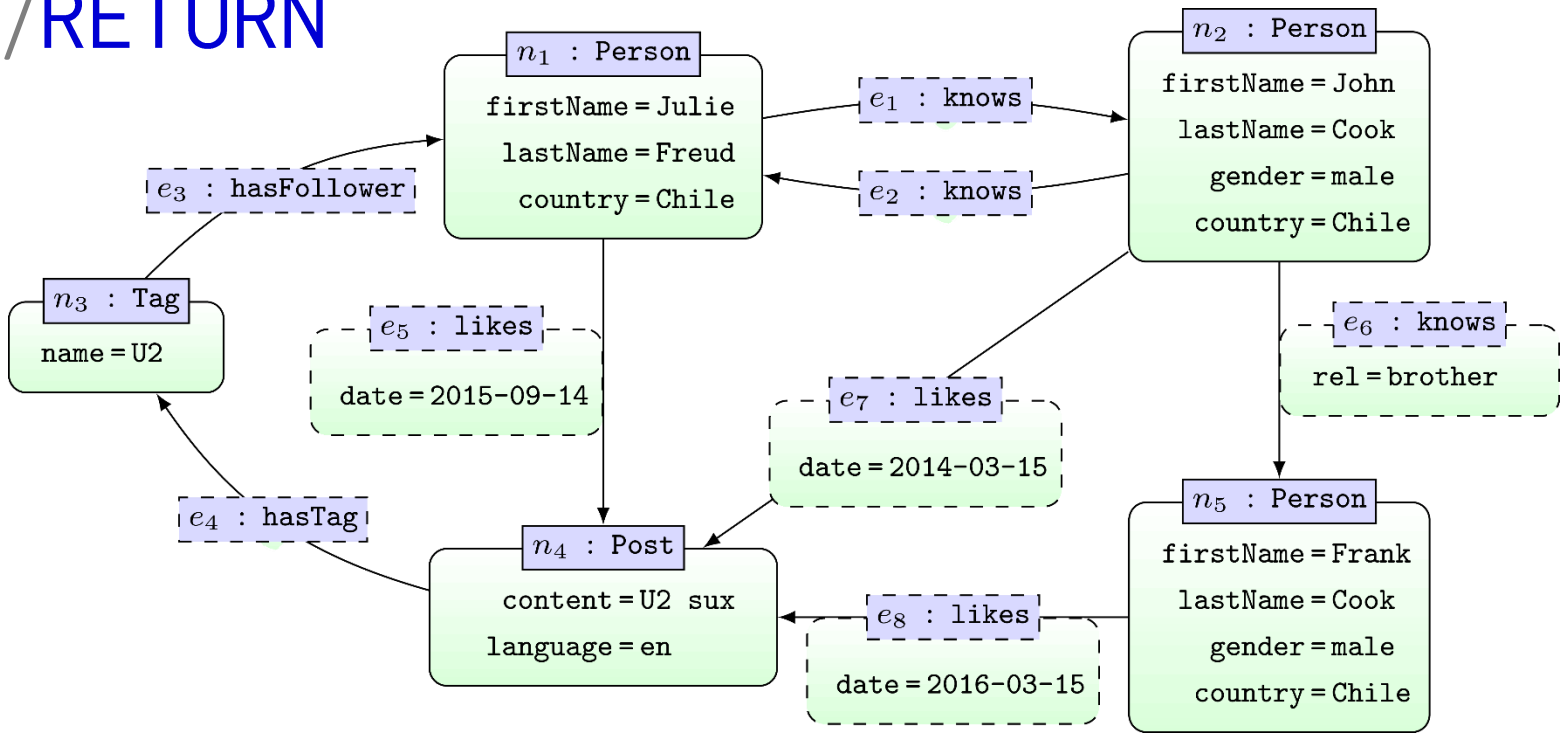
MATCH/RETURN




```
MATCH (x:Person)-->()-->()-->(x)
RETURN x.firstName
```

x.firstName
Julie

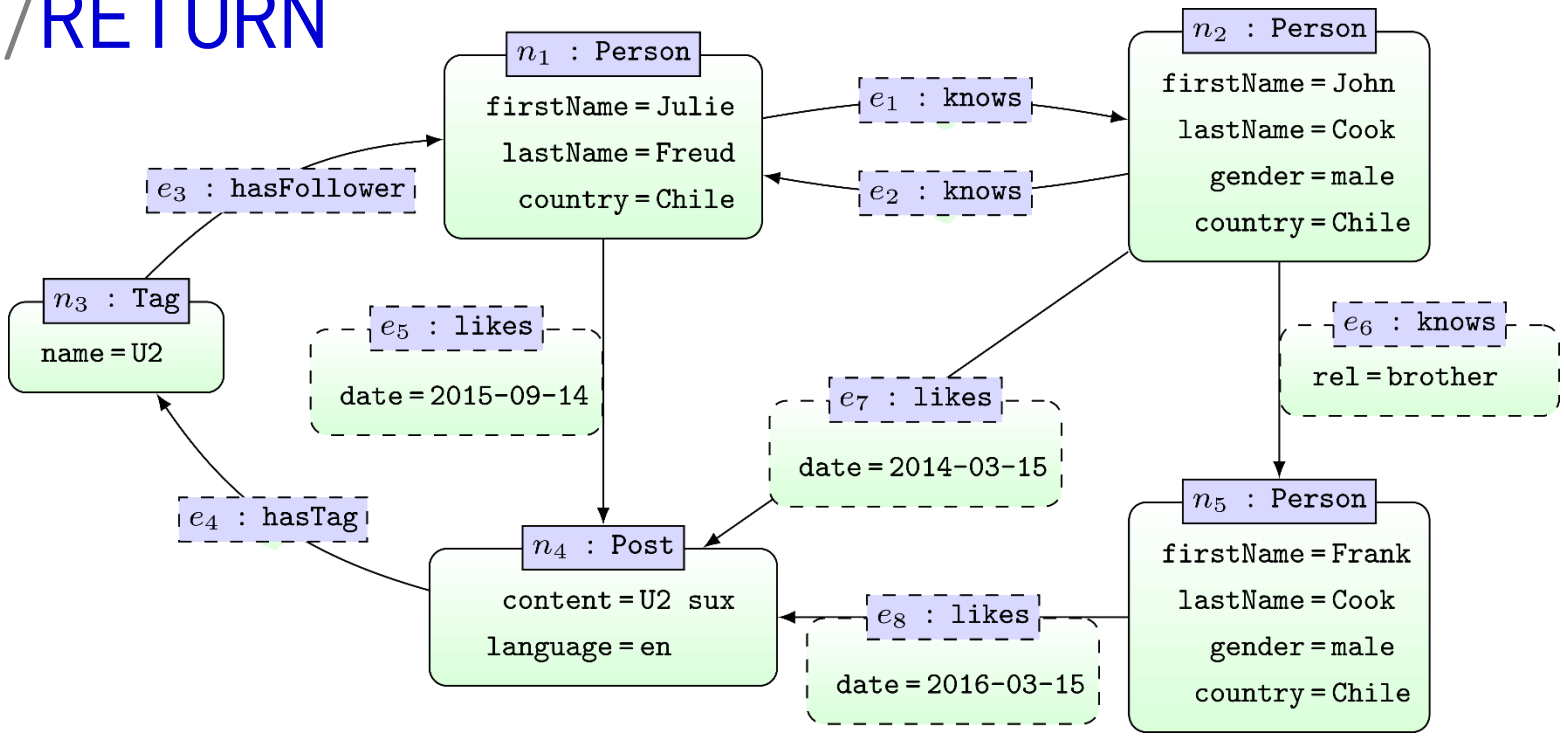
MATCH/RETURN




```
MATCH (x)-->(y)-->(x)
RETURN x.firstName
```

x.firstName
Julie
John

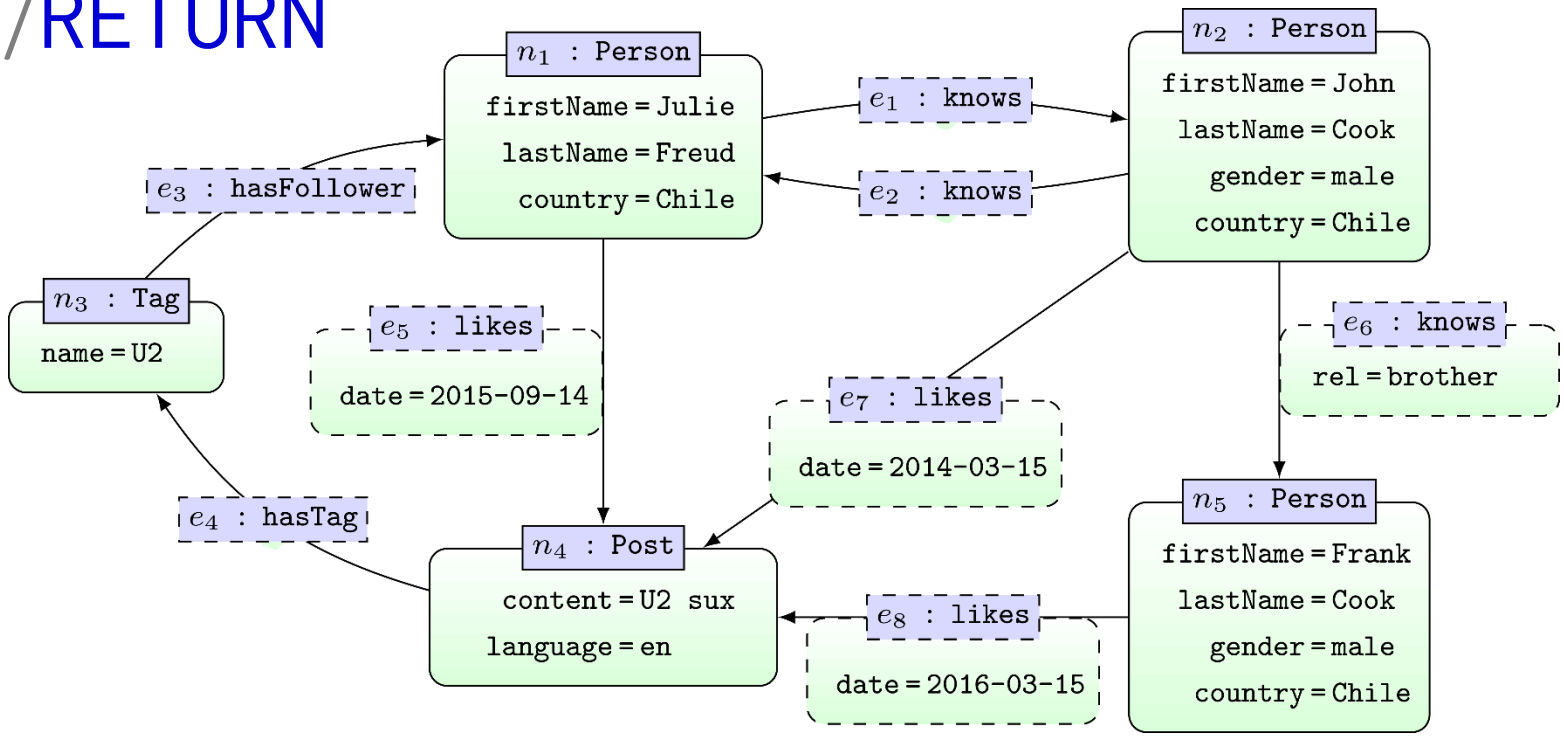
MATCH/RETURN




```
MATCH (x)-->(y)-->(x)-->(y)
RETURN x.firstName
```

x.firstName

MATCH/RETURN

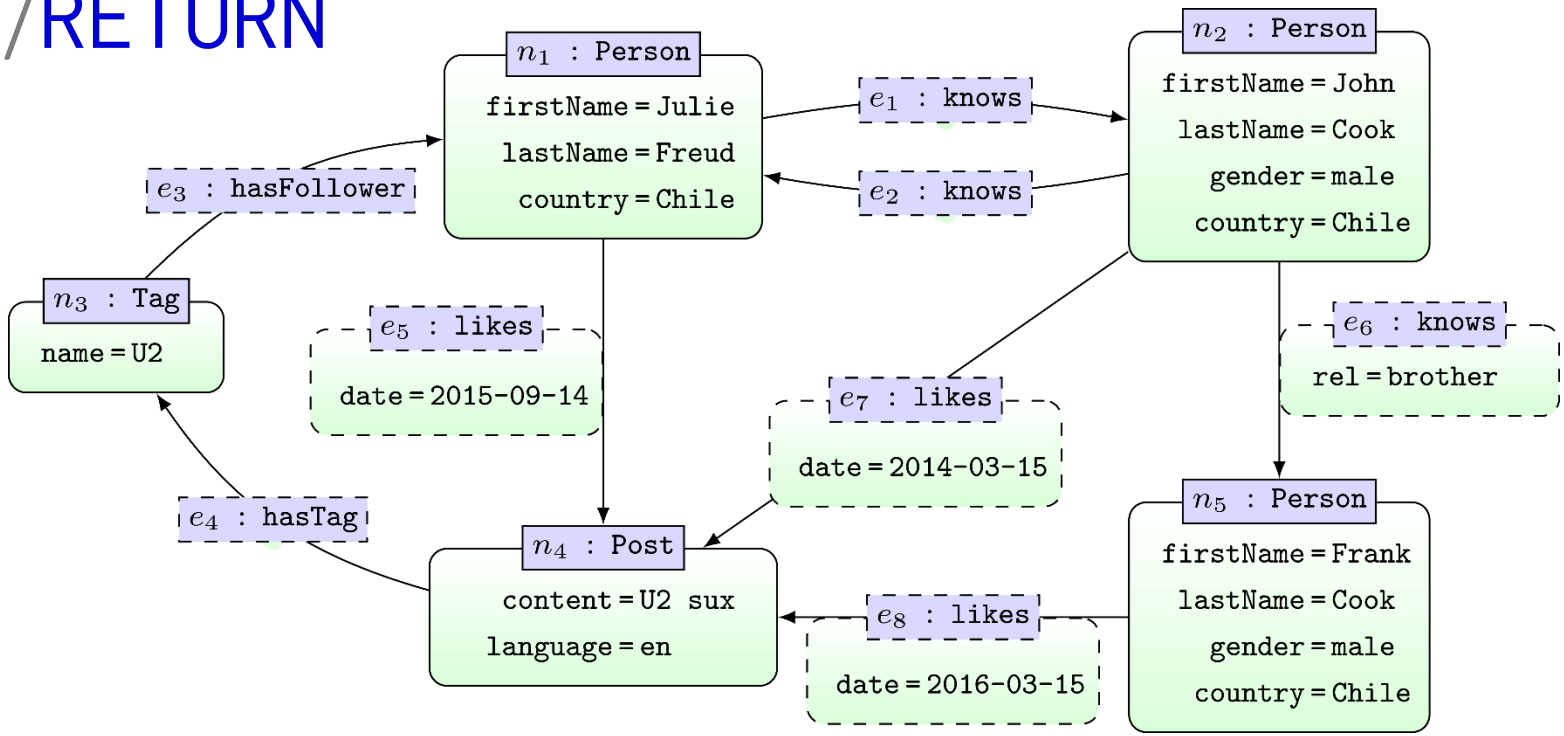



```
MATCH (x1)-[:likes]->(y)<-[:likes]-(x2)
RETURN x1.firstName AS n1, x2.firstName AS n2
```

n1	n2
Julie	John
John	Julie
John	Frank
Frank	John
Frank	Julie
Julie	Frank

... AS renames columns in results

MATCH/RETURN

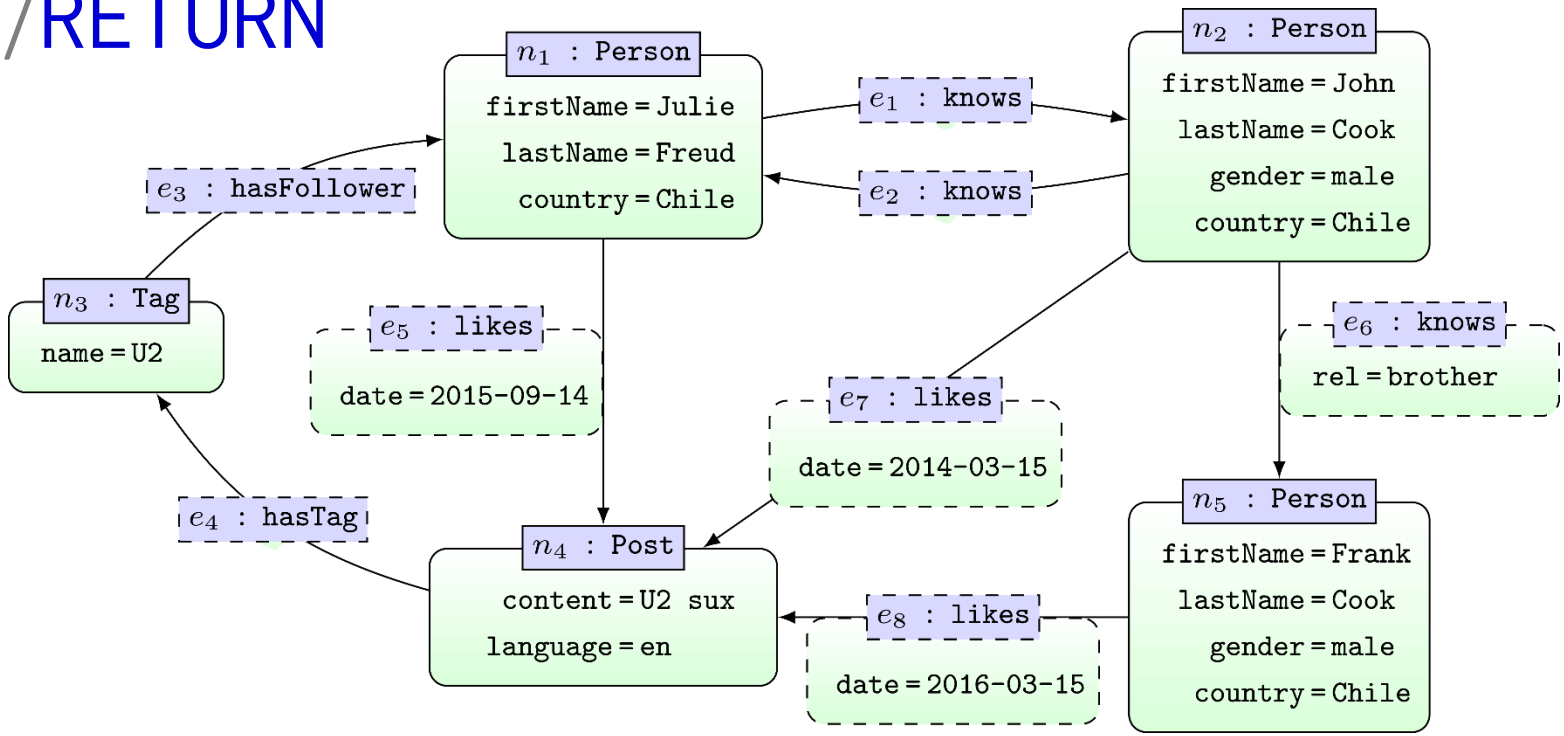



```
MATCH (x1)-[:likes]->(y)
MATCH (y)<-[:likes]-(x2)
RETURN x1.firstName AS n1, x2.firstName AS n2
```

n1	n2
Julie	John
John	Julie
Julie	Julie
John	Frank
...	...

... use multiple MATCH to match same edge multiple times

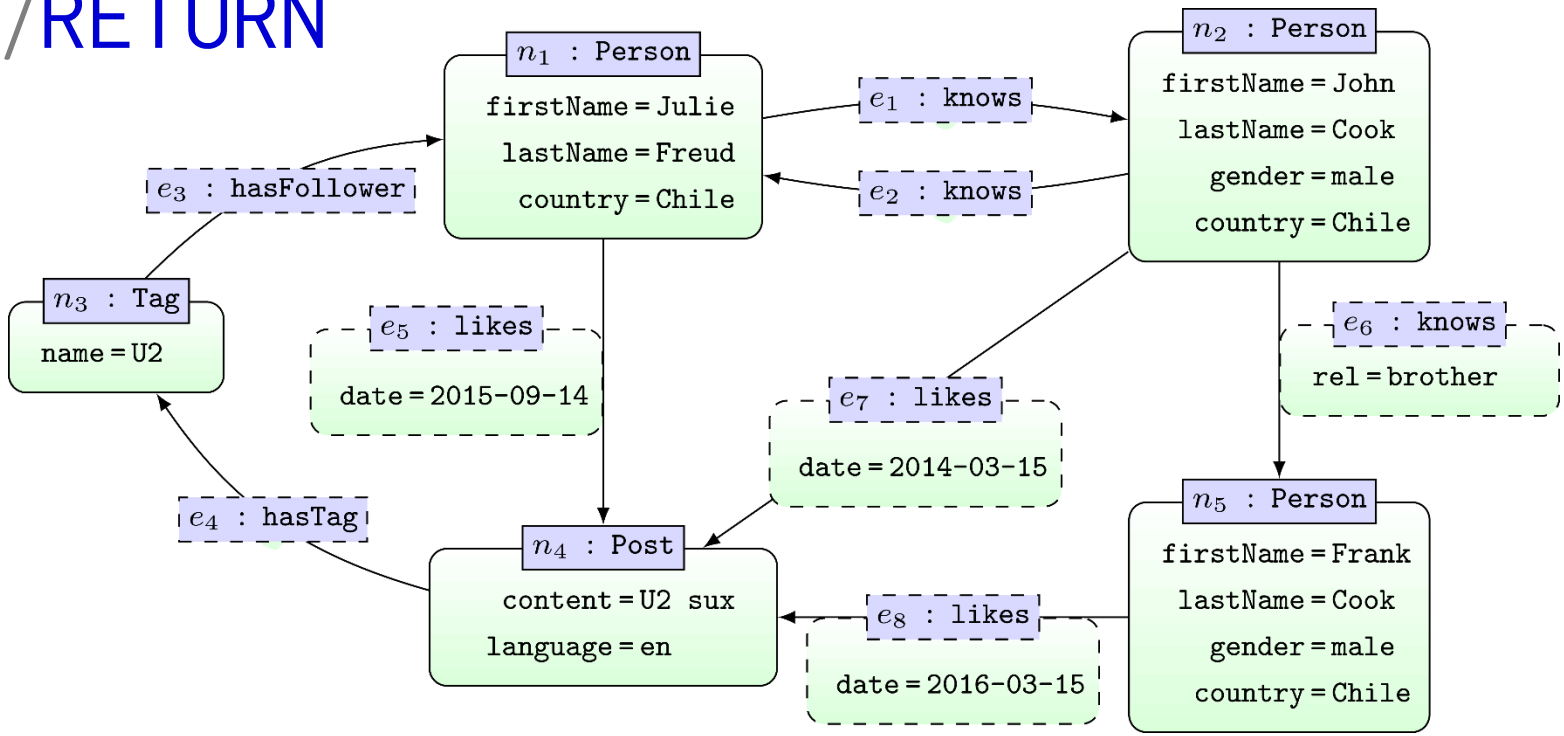
MATCH/RETURN




```
MATCH (x1)-[:likes]->(y)-[:hasTag]->(z),
   (x2)-[:likes]->(y)
RETURN z.name
```

z.name
U2
U2
U2
U2
U2
U2

MATCH/RETURN



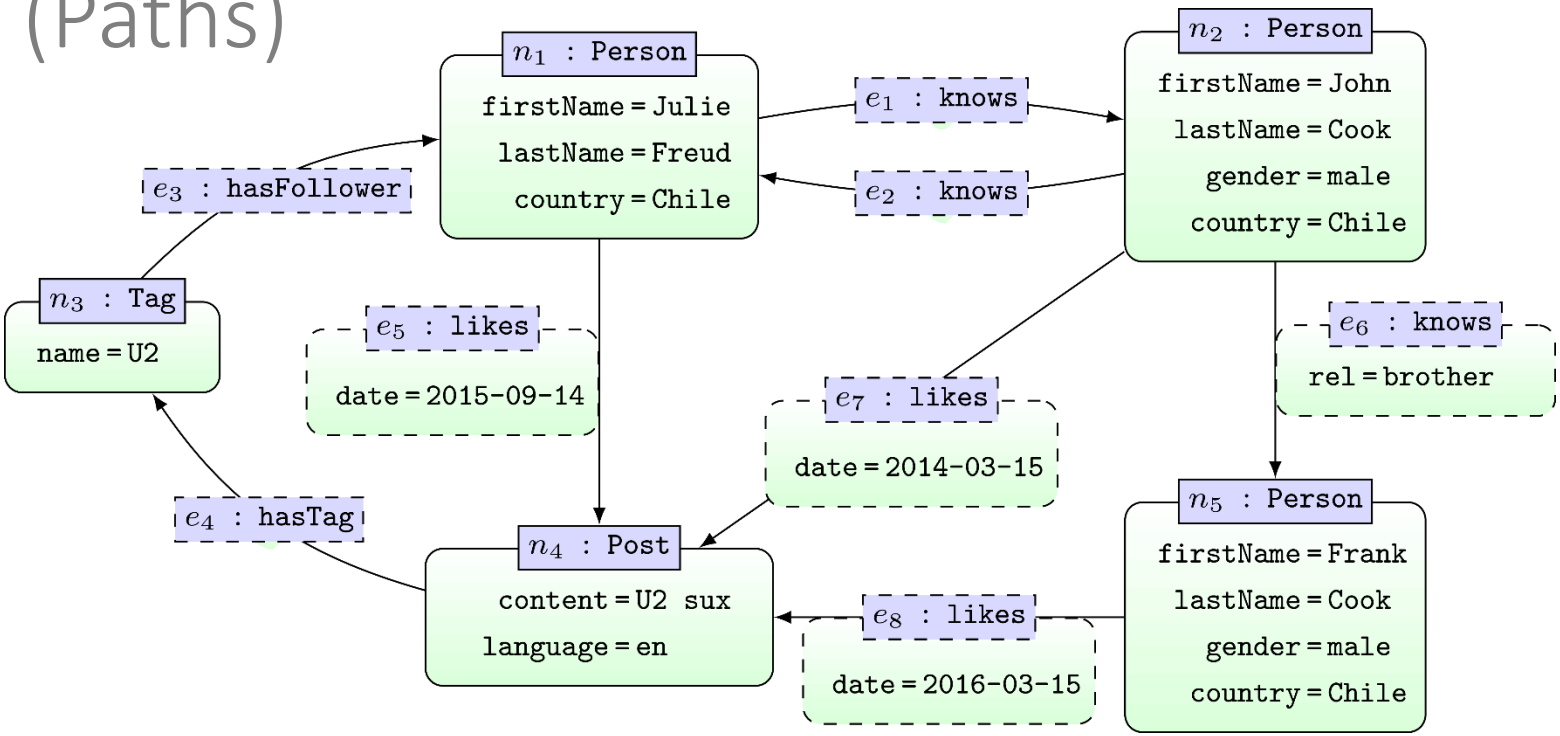

```
MATCH (x1)-[:likes]->(y)-[:hasTag]->(z)
MATCH (x2)-[:likes]->(y)
RETURN z.name
```

z.name
U2
U2
U2
U2
U2
U2
U2
U2
U2

Cypher:
MATCH (Paths)

https://neo4j.com/docs/developer-manual/3.4/cypher/clauses/where/

MATCH (Paths)

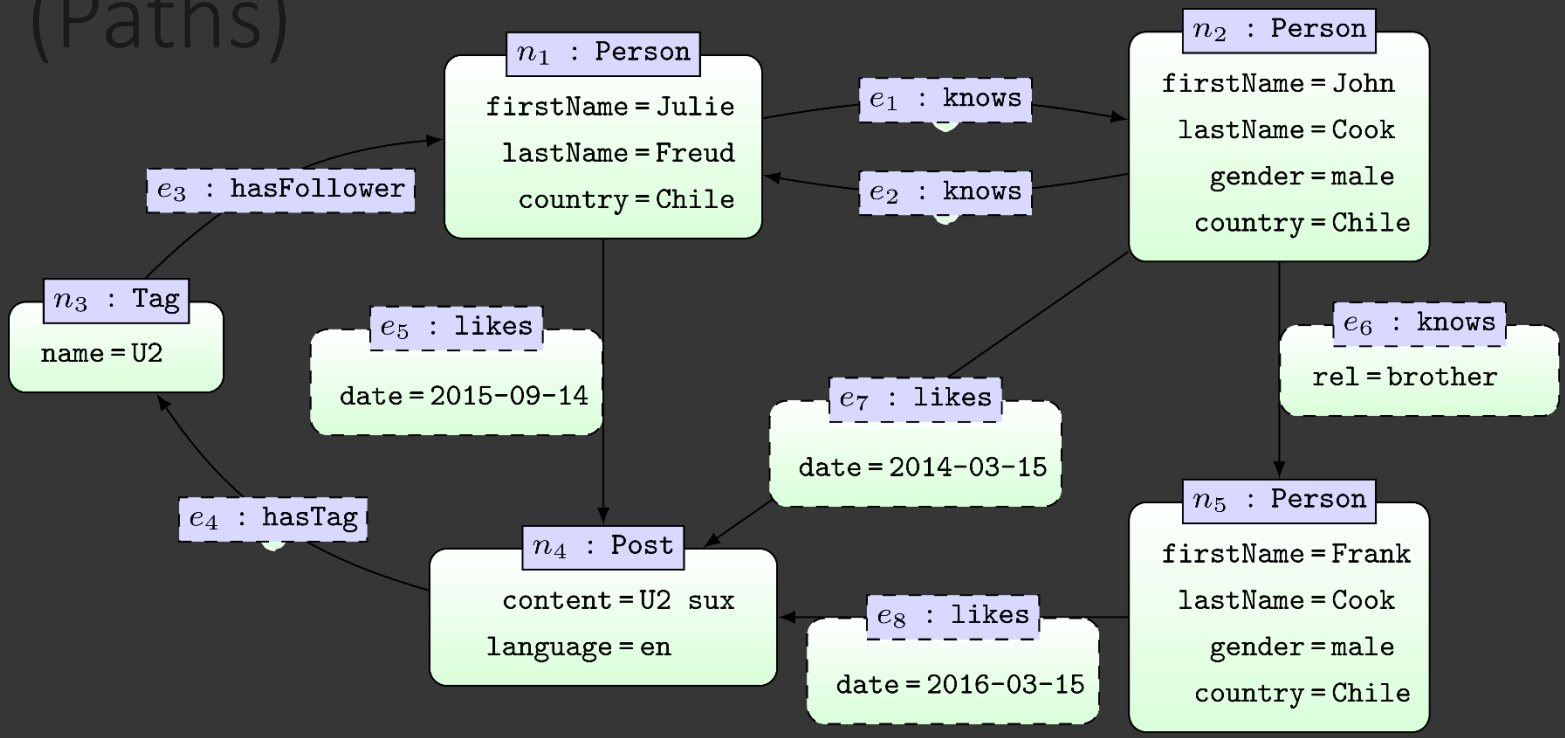



```
MATCH (x1)-[:knows*]->(x2)
RETURN x1.firstName, x2.firstName
```

x1.firstName	x2.firstName
Julie	John
Julie	Frank
Julie	Julie
John	Julie
John	John
John	Frank
John	Frank

... paths visit each edge at most once!

MATCH (Paths)



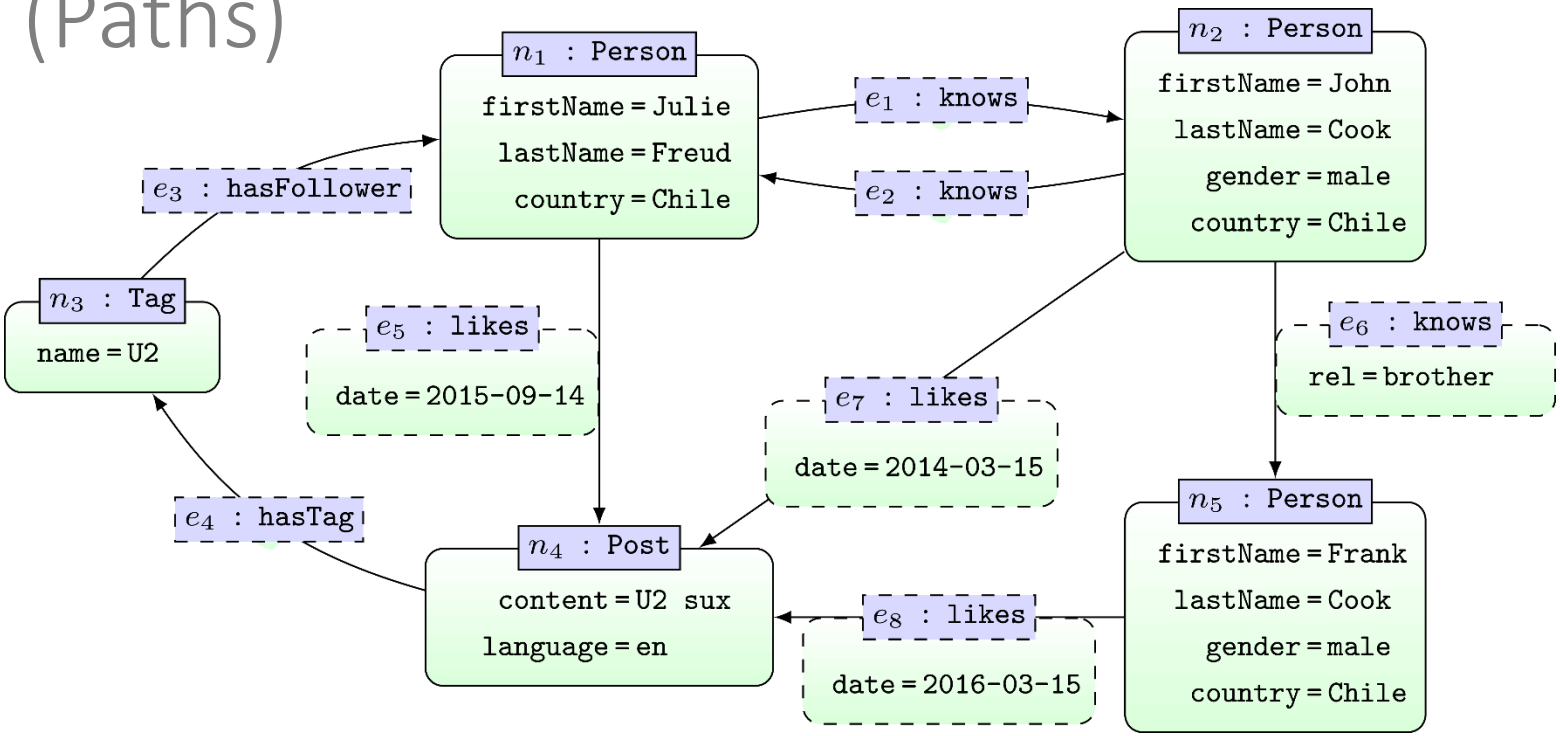

```
MATCH (x1)-[:knows*]->(x2)
RETURN x1.firstName, x2.firstName
```

x1.firstName	x2.firstName
Julie	John
Julie	Frank
Julie	Julie
John	John
John	Frank
John	Frank

Otherwise ... ... we could have infinite paths!

... paths visit each edge at most once!

MATCH (Paths)

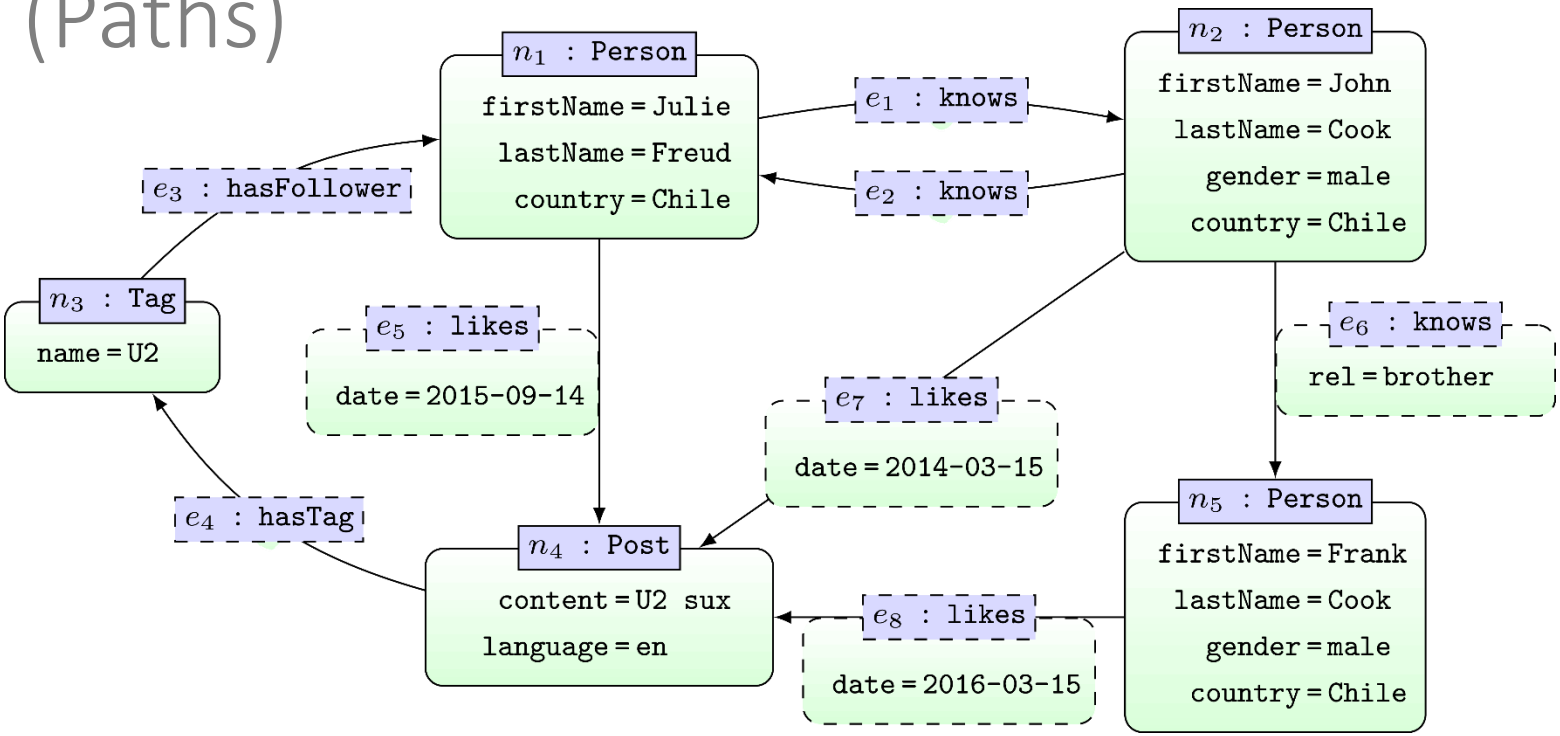



```
MATCH (x1)-[:knows*3]->(x2)
RETURN x1.firstName, x2.firstName
```

x1.firstName	x2.firstName
John	Frank

... can set minimum path length (no. of nodes visited)

MATCH (Paths)

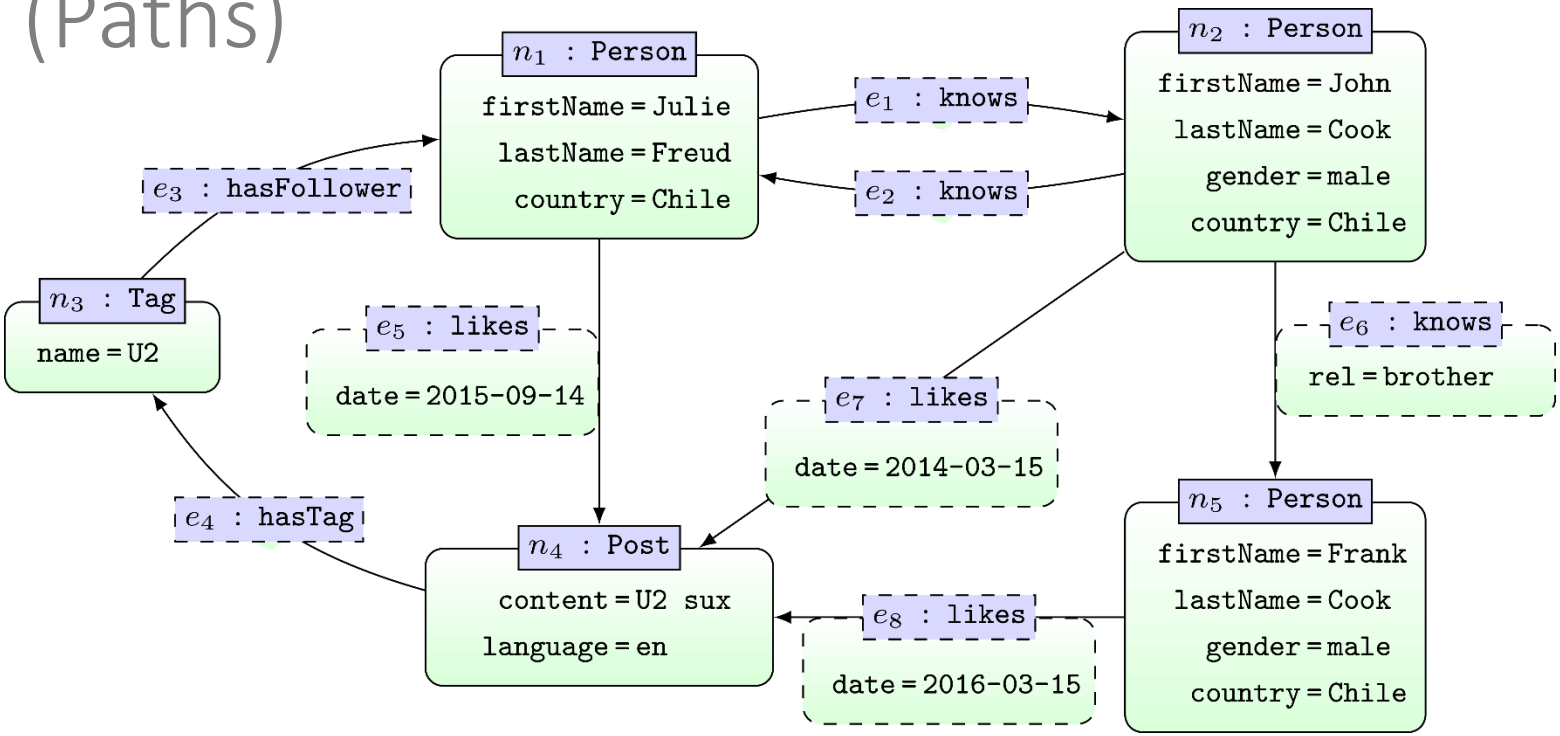



```
MATCH (x1)-[:knows*2..3]->(x2)
RETURN x1.firstName, x2.firstName
```

x1.firstName	x2.firstName
Julie	Frank
Julie	Julie
John	Frank
John	John

... or range of path length

MATCH (Paths)

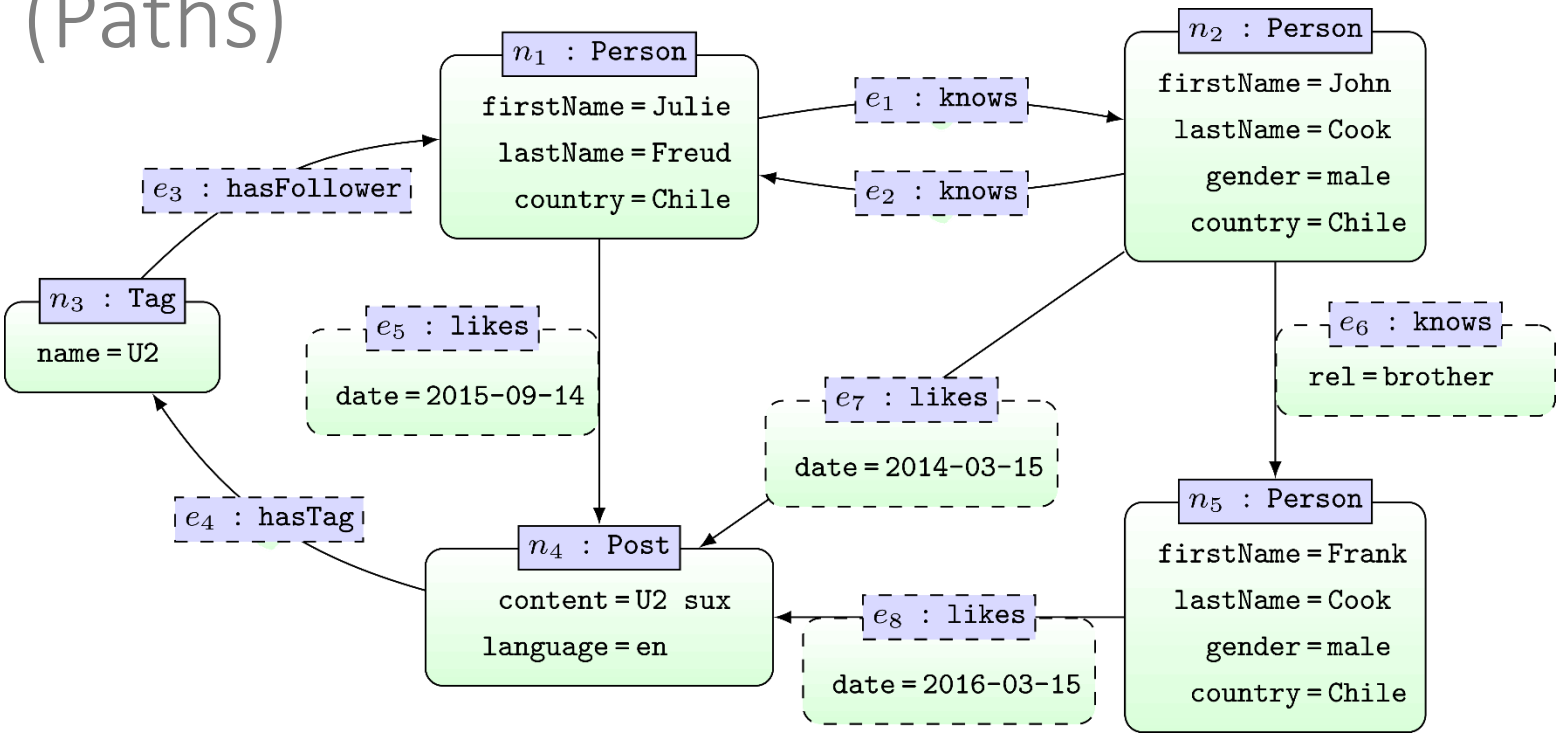



```
MATCH (x1)-[:knows*..2]->(x2)
RETURN x1.firstName, x2.firstName
```

x1.firstName	x2.firstName
Julie	John
Julie	Frank
Julie	Julie
John	Julie
John	John
John	Frank

... or maximum path length

MATCH (Paths)

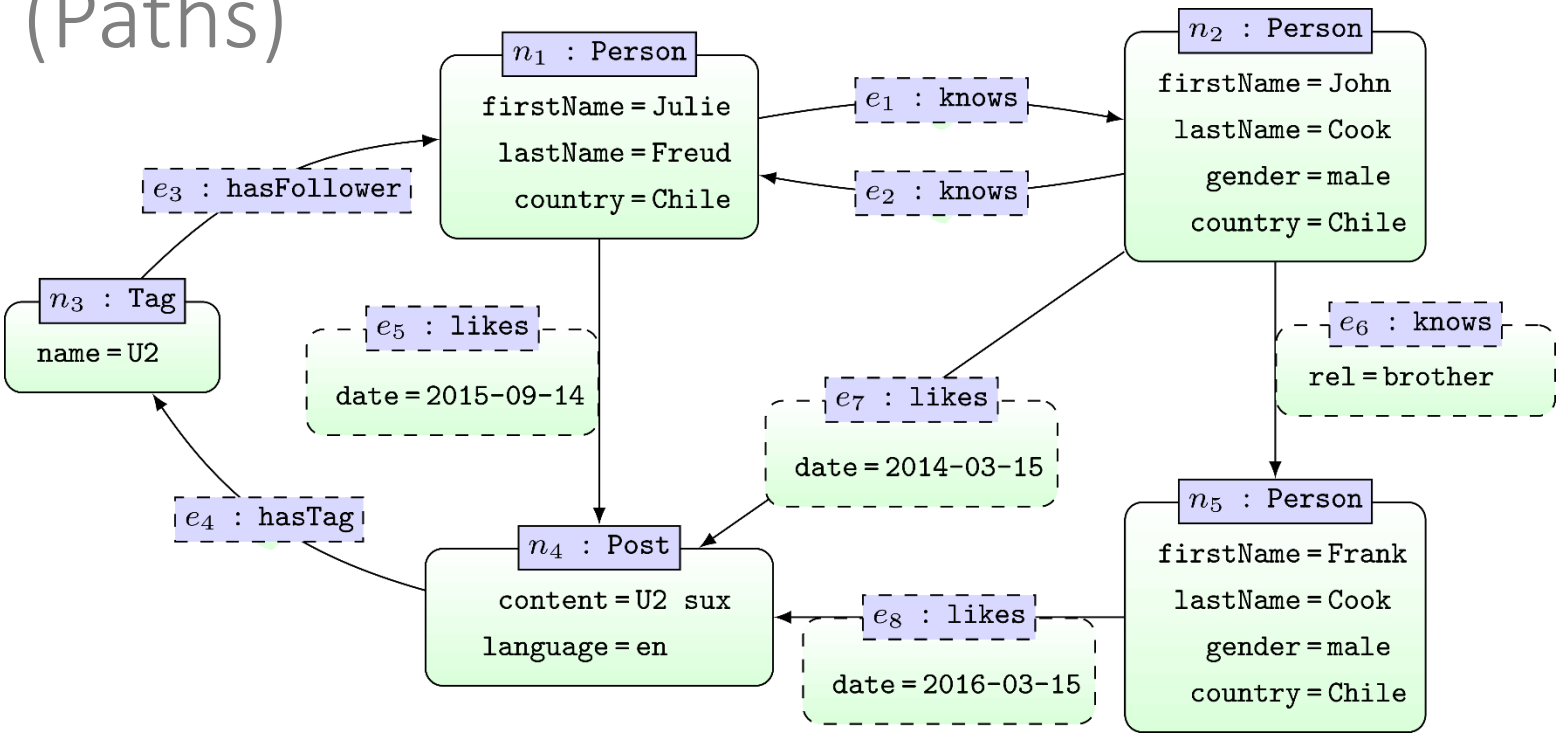



```
MATCH (x1)-[:knows*0..1]->(x2)
RETURN x1.firstName, x2.firstName
```

x1.firstName	x2.firstName
Julie	Julie
Julie	John
John	John
John	Julie
John	Frank
Frank	Frank

... 0-length path is the node itself; will match any node

MATCH (Paths)

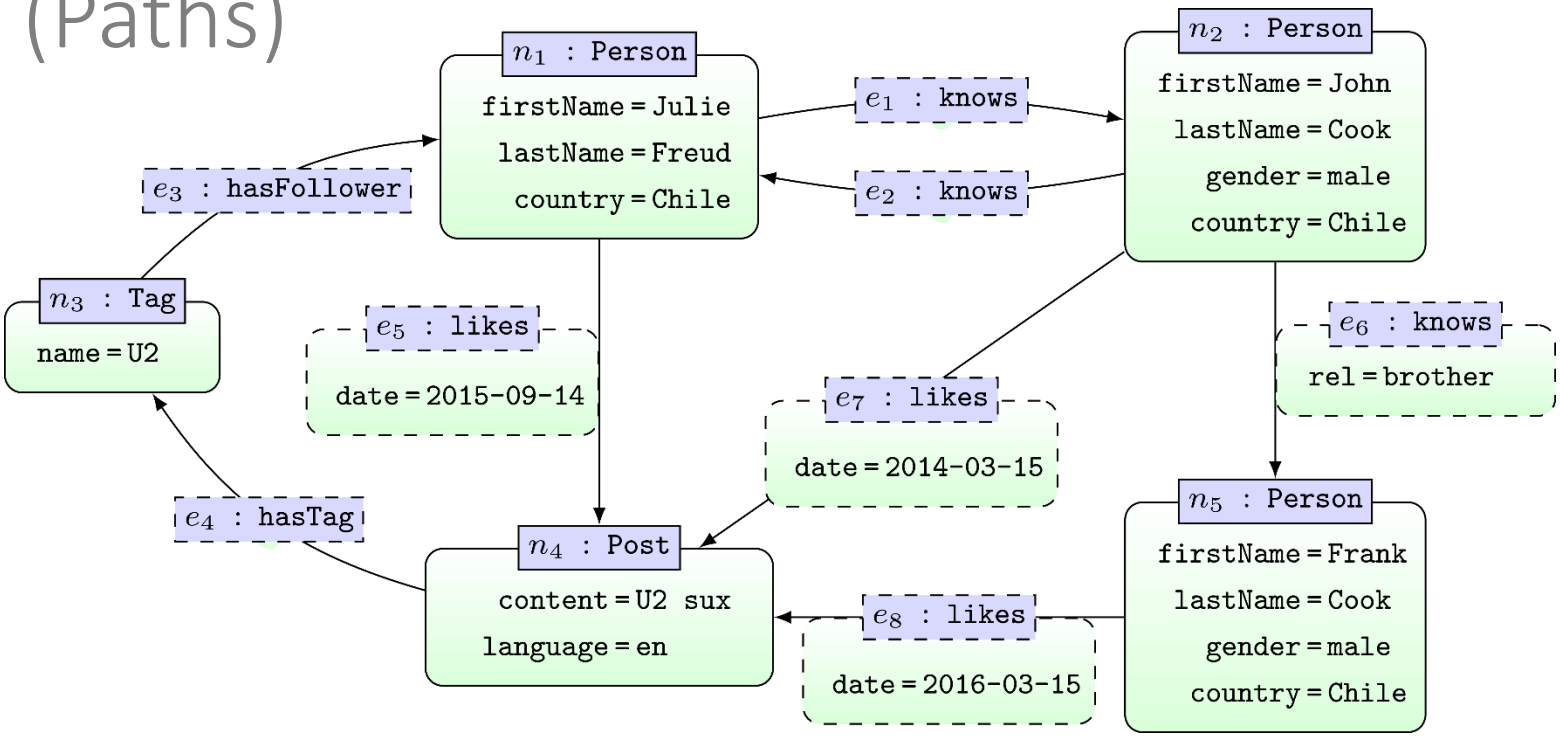



```
MATCH p = (x1)-[:knows*3]->(x2)
RETURN p
```

p
(:Person {firstName:"John",...})-[:knows]->(:Person {firstName:"Julie",...})-[:knows]->(:Person {firstName:"John",...})-[:knows rel:"brother"]->(:Person {firstName:"Frank",...})

... can return a full path

MATCH (Paths)

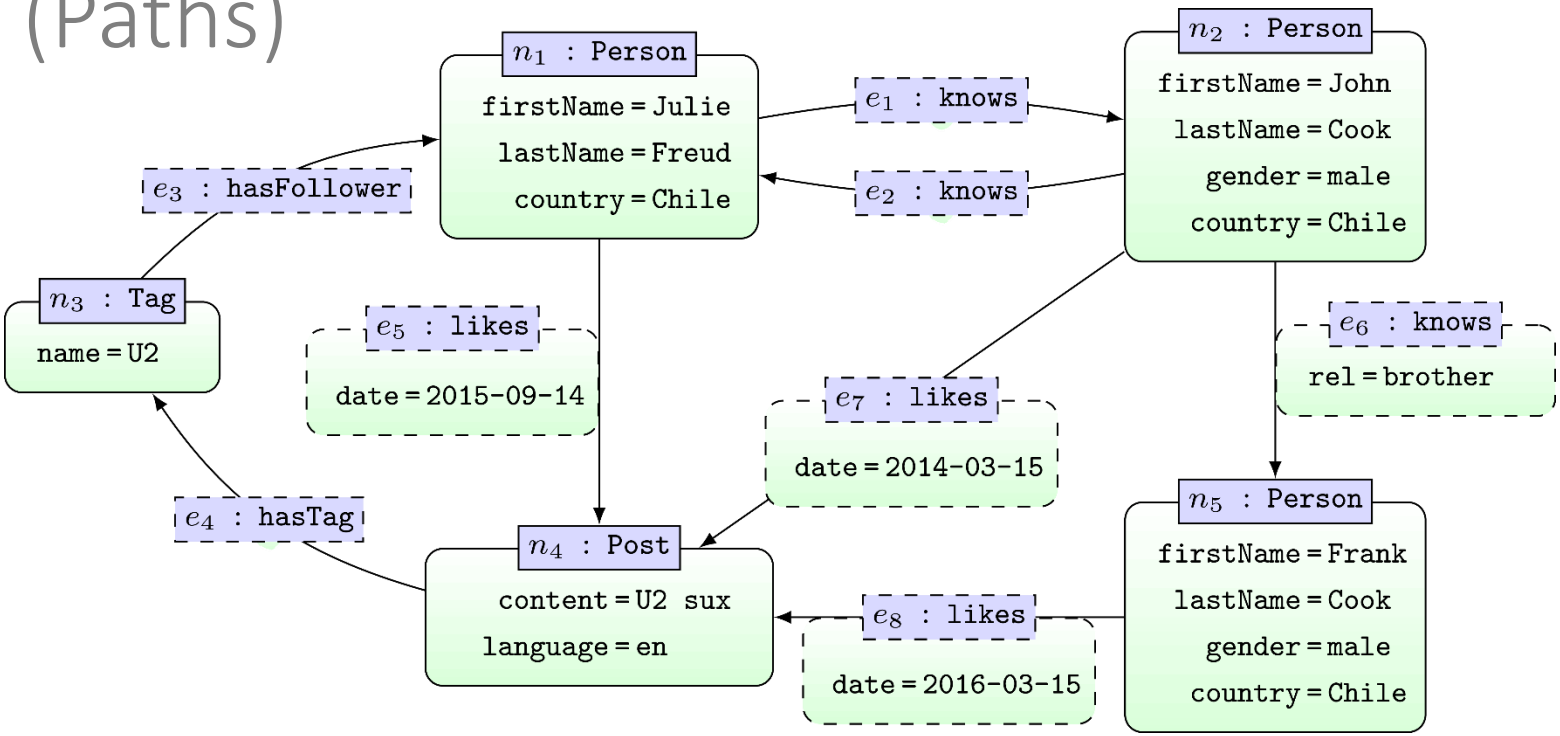



```
MATCH p = (x1)-[:knows*3]->(x2)
RETURN p
```

p
(:Person {firstName:"John",...})-[:knows]->(:Person {firstName:"Julie",...})-[:knows]->(:Person {firstName:"John",...})-[:knows rel:"brother"]->(:Person {firstName:"Frank",...})

... can return a full path

MATCH (Paths)

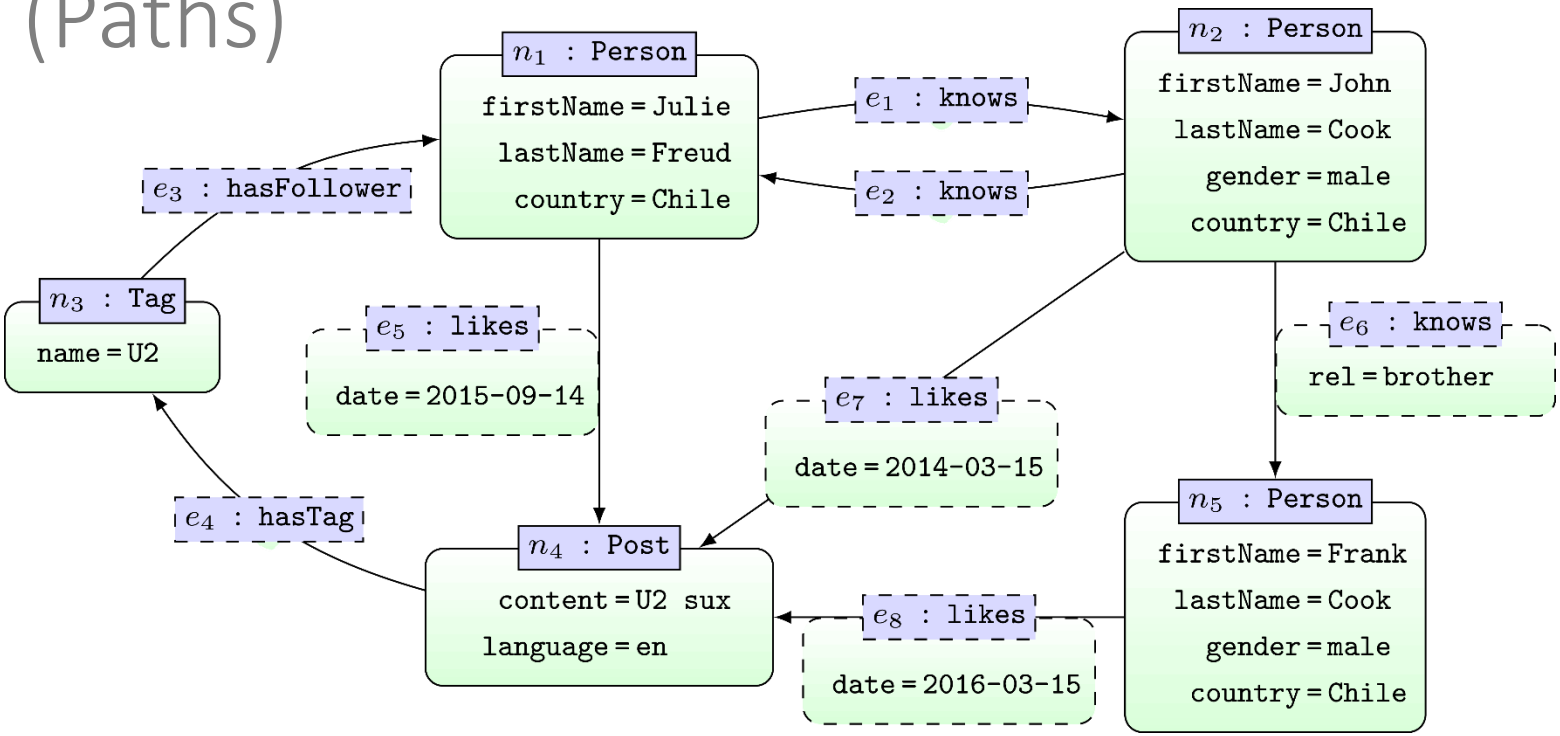



```
MATCH p = (x1)-[:knows*3]->(x2)
RETURN p
```

p
$n_2 \cdot e_2 \cdot n_1 \cdot e_1 \cdot n_2 \cdot e_6 \cdot n_5$

... can return a full path

MATCH (Paths)

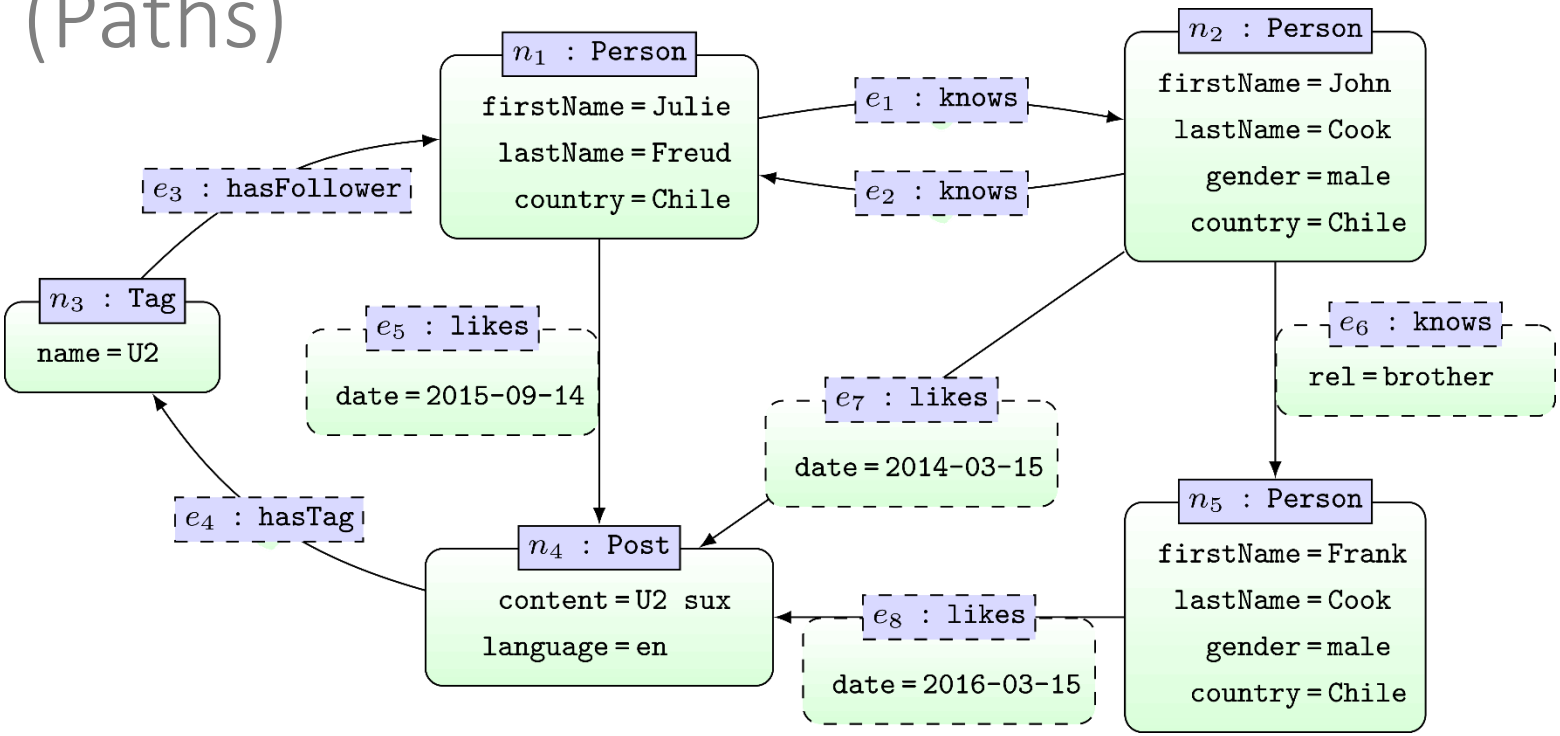



```
MATCH (f {firstName: 'Frank'}),
  (j {firstName: 'Julie'}),
  p = shortestPath((f)-[*]-(j))
RETURN p
```

p
$n_5 \cdot e_6 \cdot n_2 \cdot e_2 \cdot n_1$

... returns any shortest path (matching criteria)

MATCH (Paths)

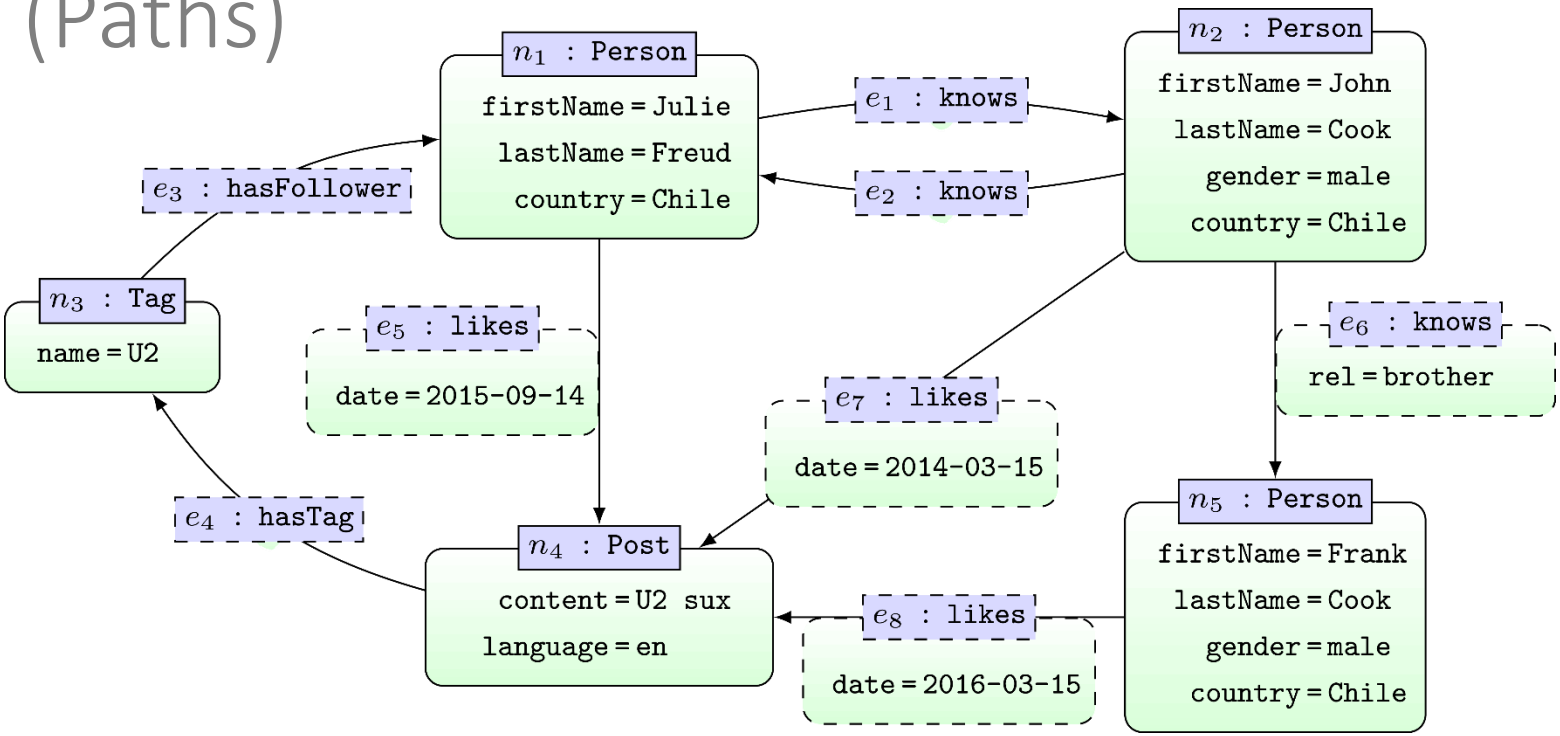



```
MATCH (f {firstName: 'Frank'}),
  (j {firstName: 'Julie'}),
  p = allShortestPaths((f)-[*]-(j))
RETURN p
```

p
$n_5 \cdot e_6 \cdot n_2 \cdot e_2 \cdot n_1$
$n_5 \cdot e_6 \cdot n_2 \cdot e_1 \cdot n_1$
$n_5 \cdot e_8 \cdot n_4 \cdot e_5 \cdot n_1$

... returns all shortest paths (matching criteria)

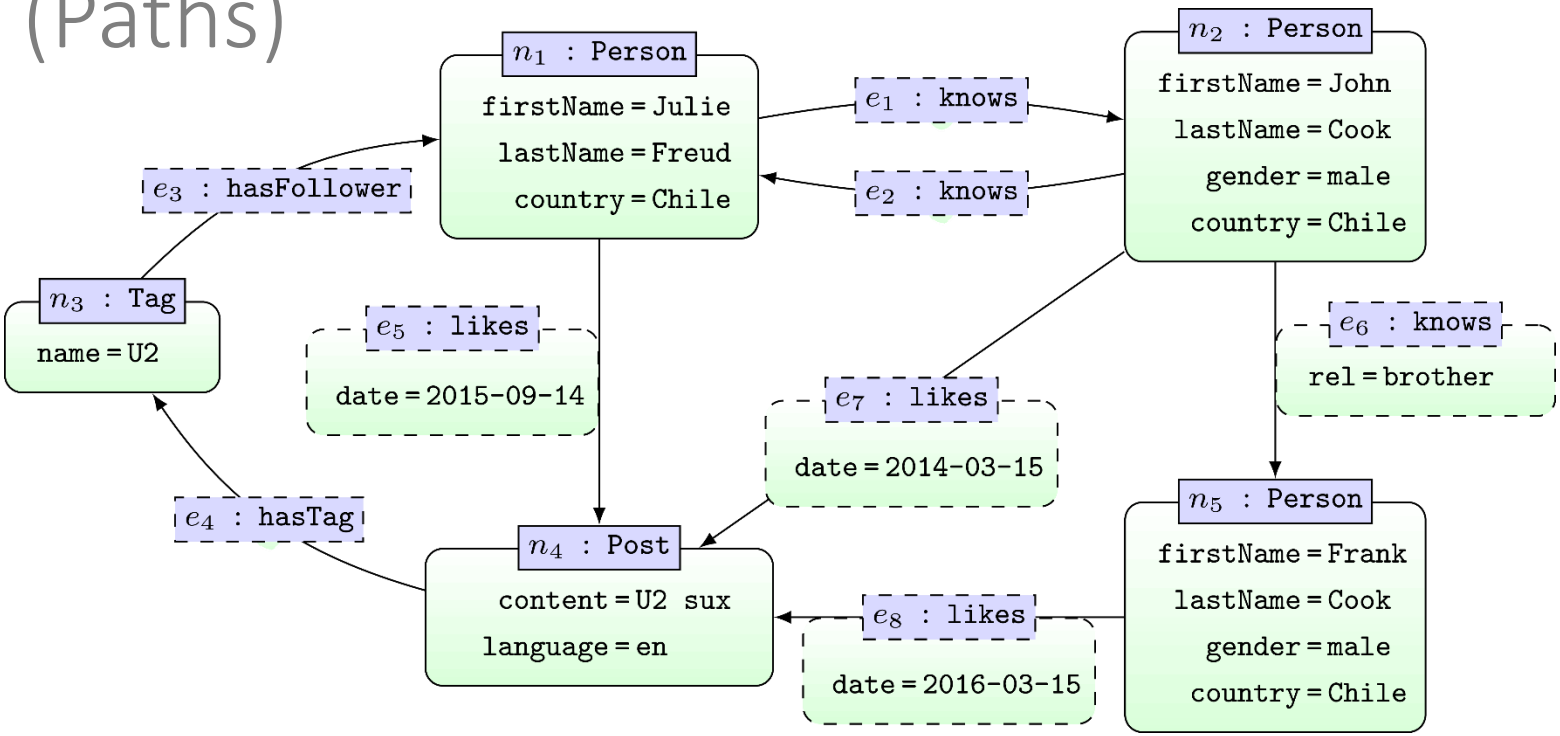
MATCH (Paths)




```
MATCH (f {firstName: 'Frank'}),
  (j {firstName: 'Julie'}),
  p = shortestPath((f)-[*]->(j))
RETURN p
```

p
$n_5 \cdot e_8 \cdot n_4 \cdot e_4 \cdot n_3 \cdot e_3 \cdot n_1$

MATCH (Paths)

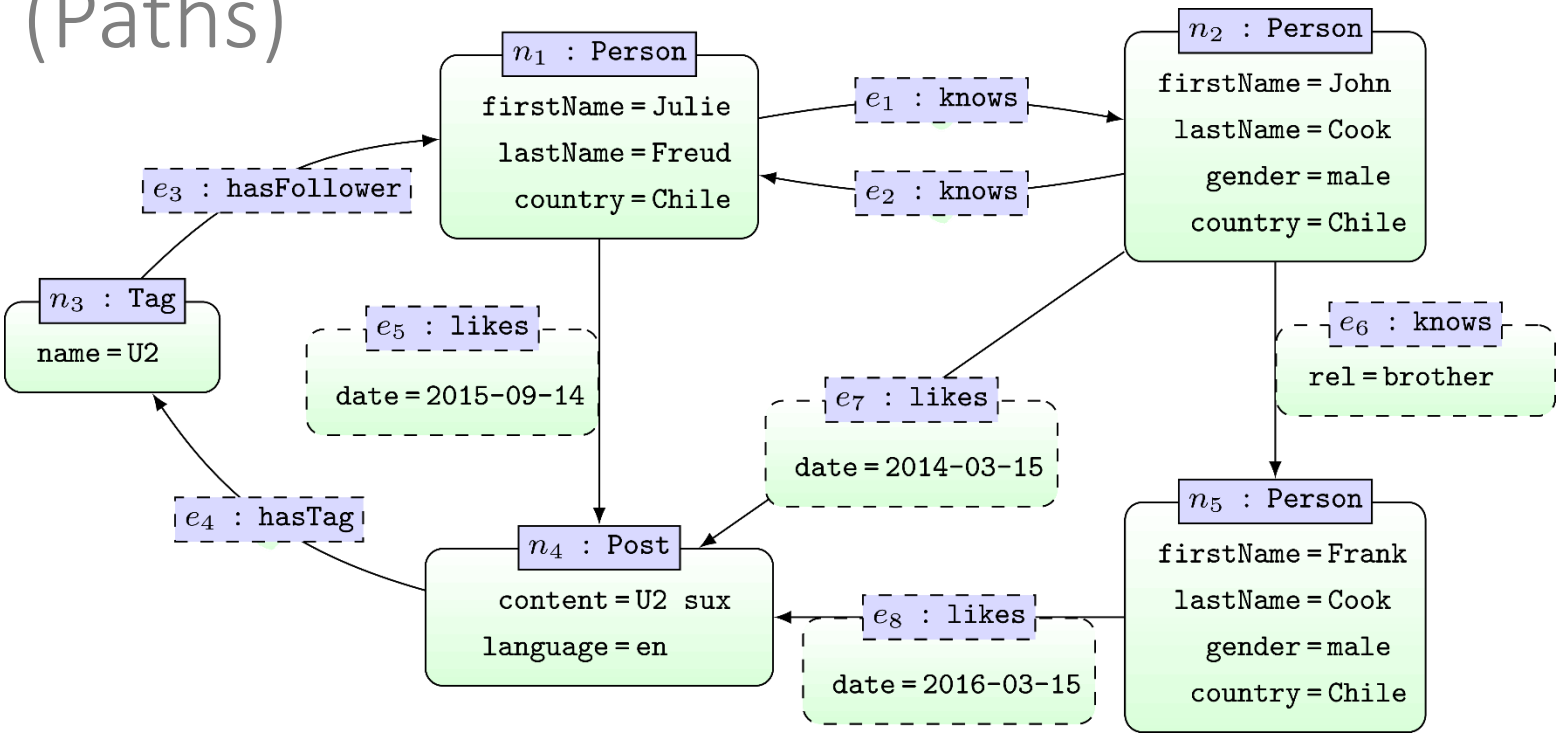



```
MATCH (c1 {lastName: 'Cook'}),
  (c2 {lastName: 'Freud'}),
  p = shortestPath((c1)-[*]->(c2))
RETURN p
```

p
$n_5 \cdot e_8 \cdot n_4 \cdot e_4 \cdot n_3 \cdot e_3 \cdot n_1$
$n_2 \cdot e_2 \cdot n_1$

... returns a shortest path for each matching pair of nodes

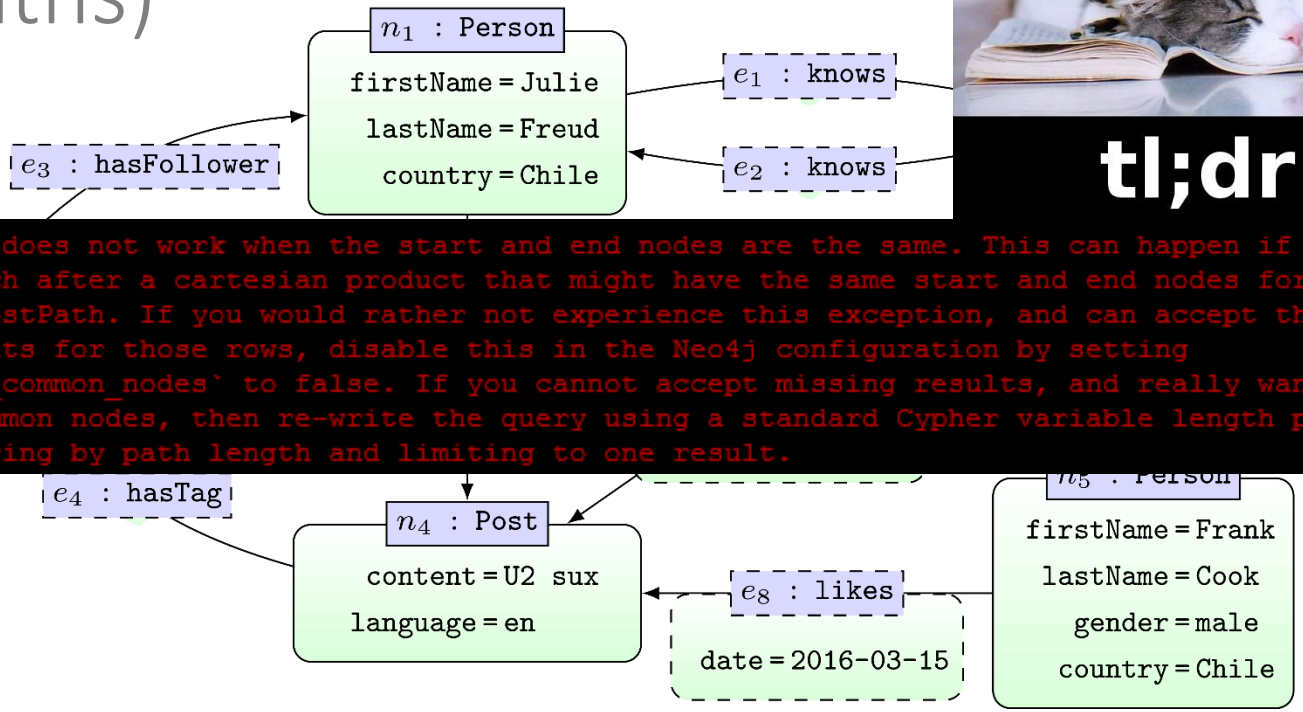
MATCH (Paths)




```
MATCH (c1 {lastName: 'Cook'}),
  (c2 {lastName: 'Cook'}),
  p = shortestPath((c1)-[*]->(c2))
RETURN p
```

MATCH (Paths)

tl;dr




```
The shortest path algorithm does not work when the start and end nodes are the same. This can happen if you
perform a shortestPath search after a cartesian product that might have the same start and end nodes for some
of the rows passed to shortestPath. If you would rather not experience this exception, and can accept the
possibility of missing results for those rows, disable this in the Neo4j configuration by setting
`cypher.forbid_shortestpath_common_nodes` to false. If you cannot accept missing results, and really want the
shortestPath between two common nodes, then re-write the query using a standard Cypher variable length pattern
expression followed by ordering by path length and limiting to one result.
```

```
MATCH (c1 {lastName: 'Cook'}),
  (c2 {lastName: 'Cook'}),
  p = shortestPath((c1)-[*]->(c2))
RETURN p
```

Cypher:

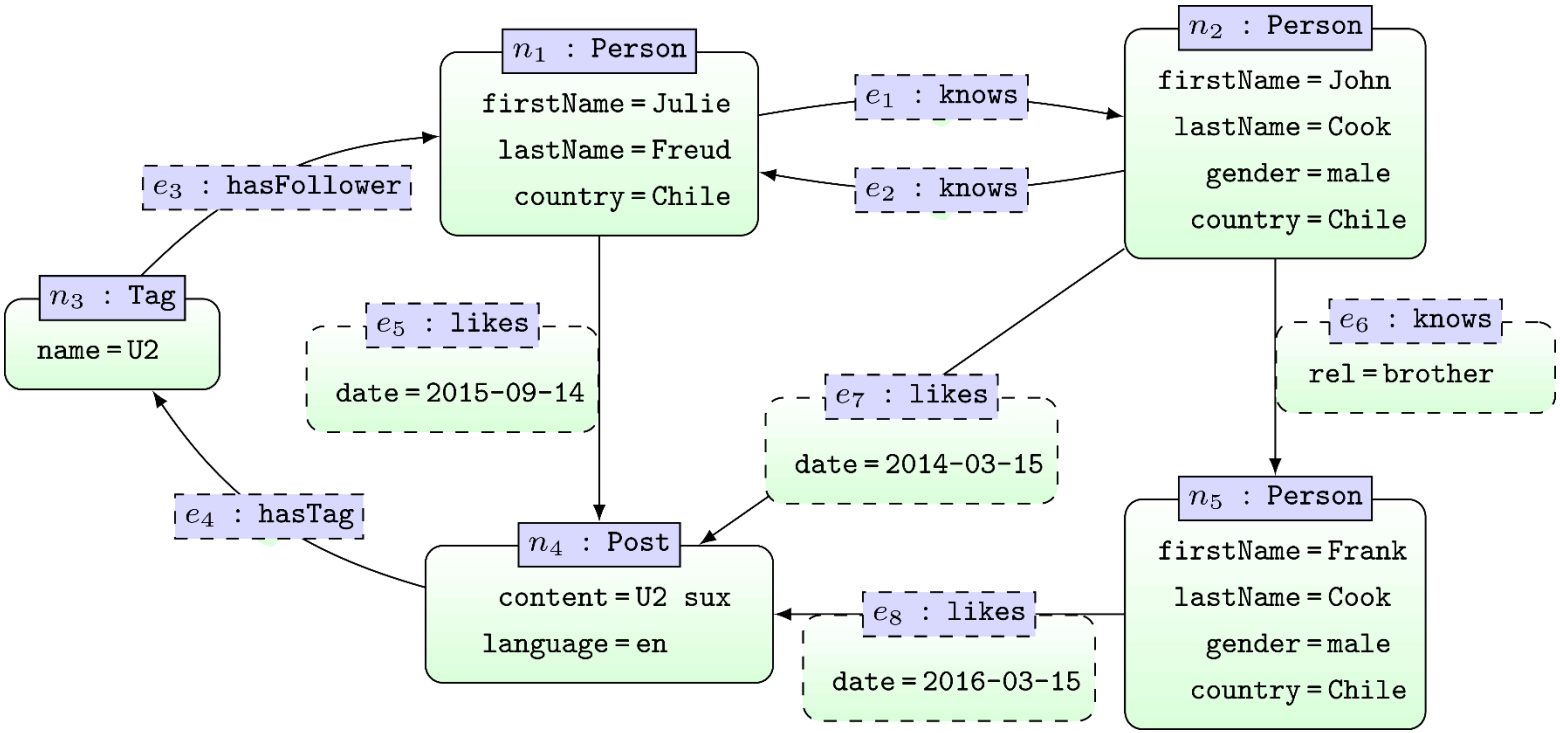
WHERE

https://neo4j.com/docs/developer-manual/3.4/cypher/clauses/where/

WHERE

- Boolean:
  - AND, OR, XOR, NOT
- (In)equalities:
  - <, >, <>, <=, >=
- Exists attribute property:
  - EXISTS
- Boolean:
  - STARTS WITH, ENDS WITH, CONTAINS, =~ (Regex)

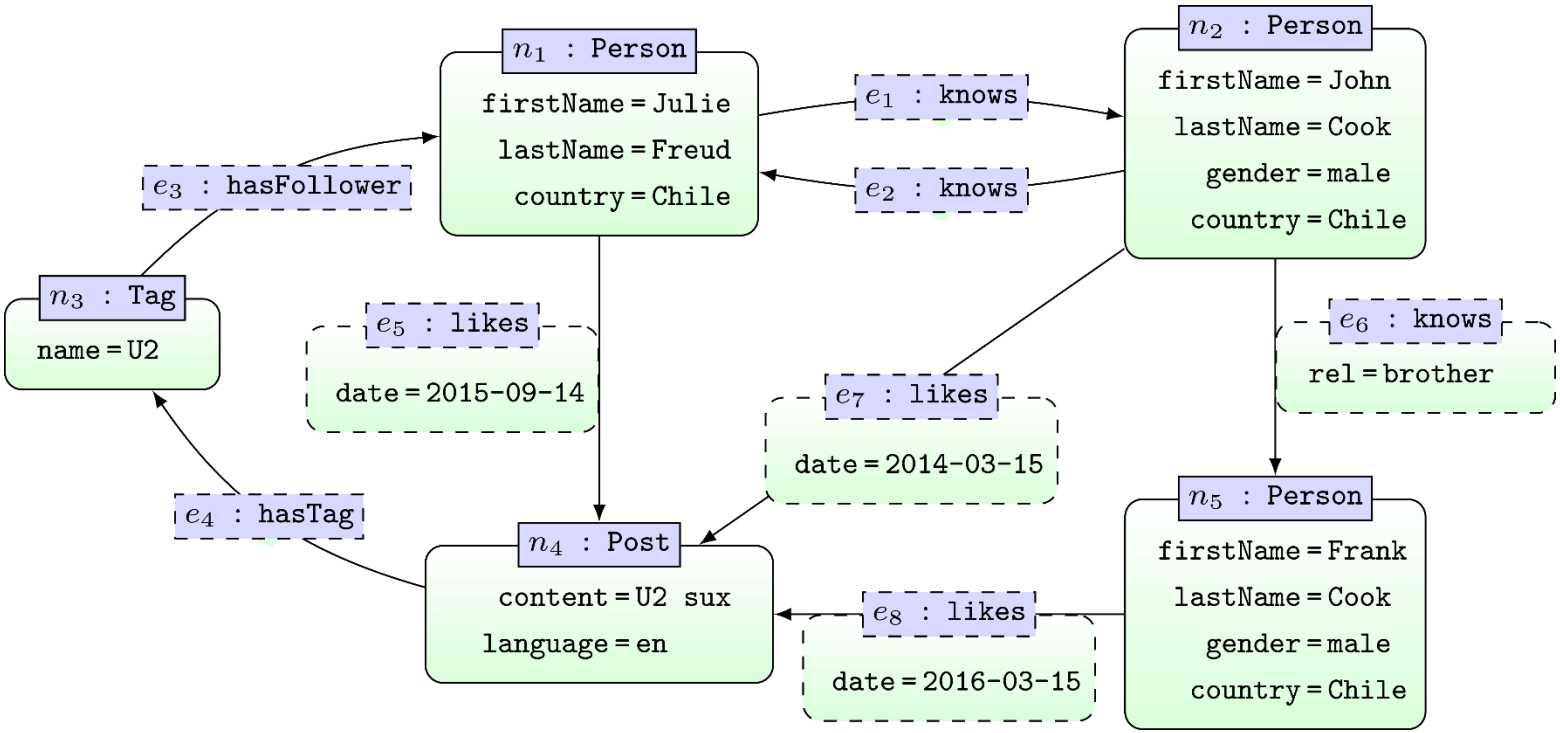
WHERE




```
MATCH (x)-[r:likes]->(y:Post)
WHERE r.date > '2010-01-01' AND r.date < '2015-01-01'
RETURN x.firstName
```

x.firstName
John

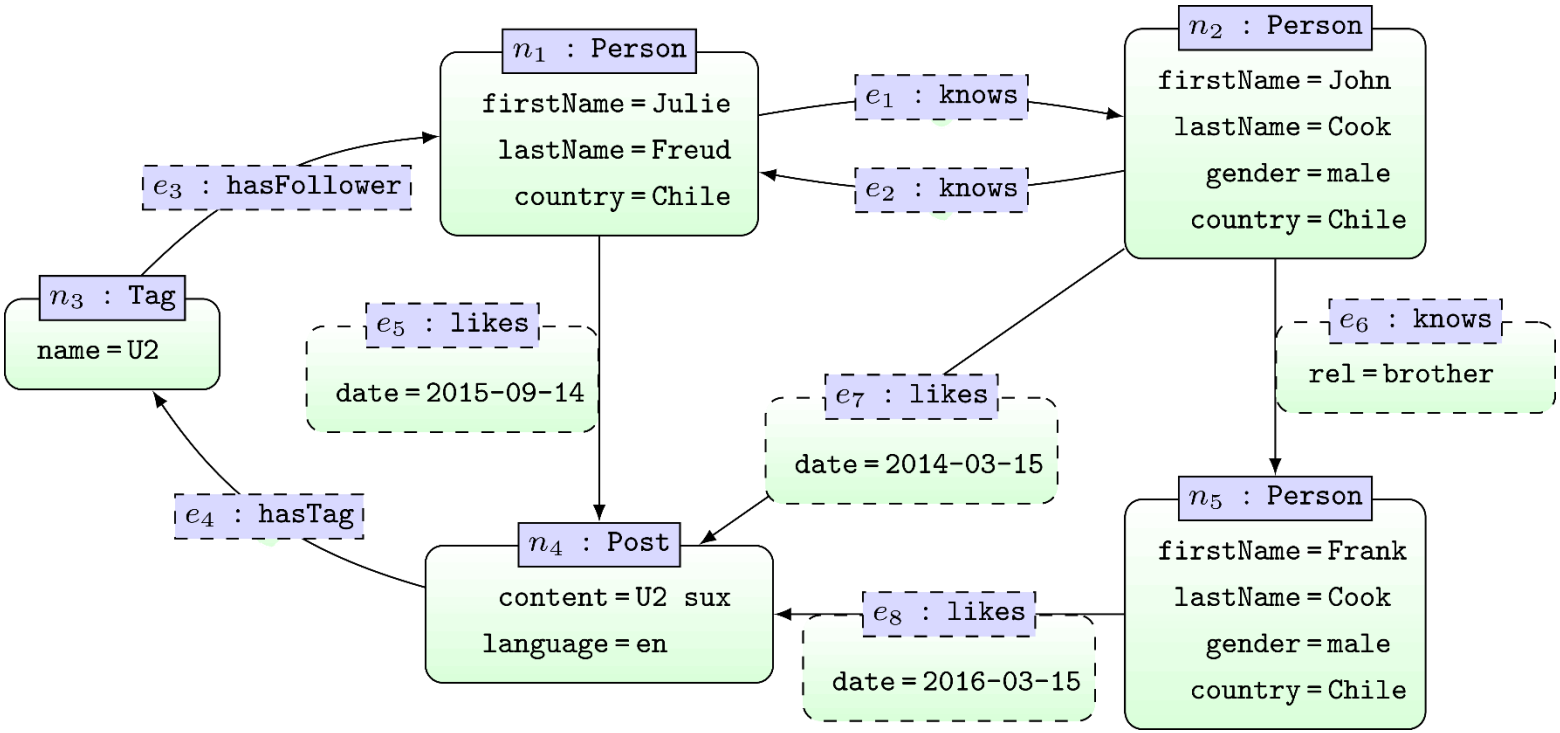
WHERE




```
MATCH (x)
WHERE EXISTS(x.gender)
RETURN x.firstName
```

x.firstName
John
Frank

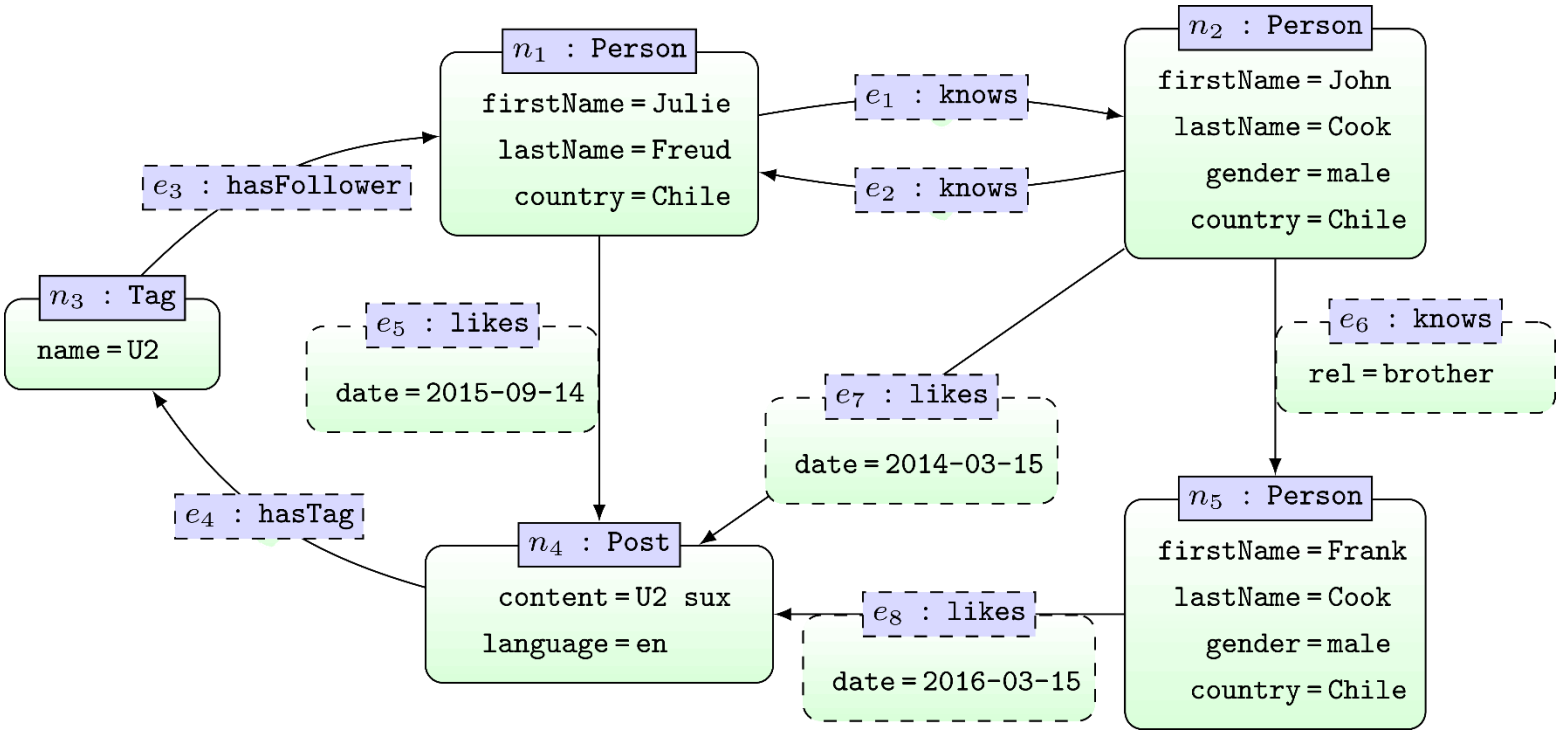
WHERE




```
MATCH (x)
WHERE x.firstName STARTS WITH 'J'
RETURN x.firstName
```

x.firstName
John
Julie

WHERE




```
MATCH (x)
WHERE x.name =~ '.*[0-9]'
RETURN x.name
```

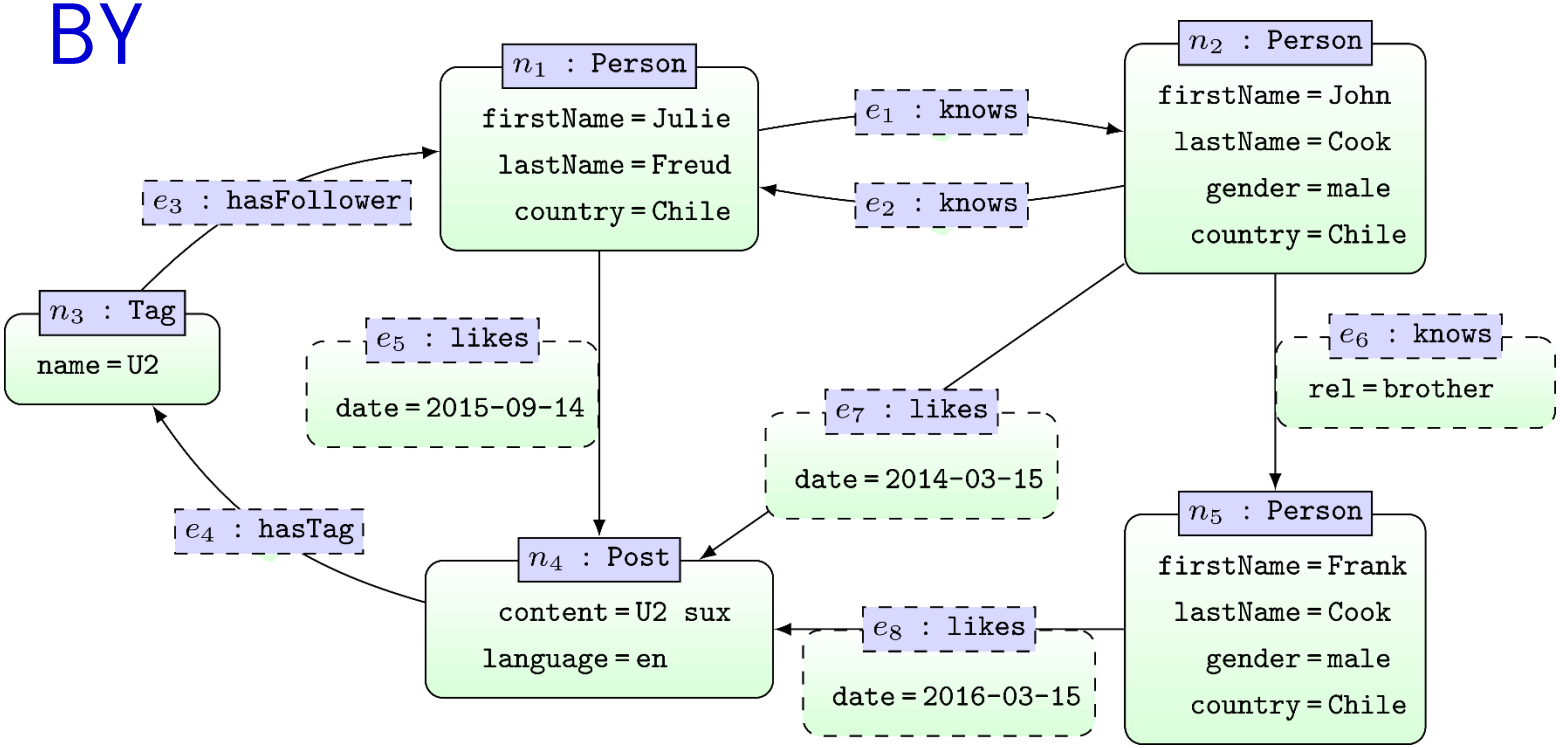
x.name
U2

Cypher:

ORDER BY/SKIP/LIMIT

https://neo4j.com/docs/developer-manual/3.4/cypher/clauses/order-by/
https://neo4j.com/docs/developer-manual/3.4/cypher/clauses/skip/
https://neo4j.com/docs/developer-manual/3.4/cypher/clauses/limit/

ORDER BY



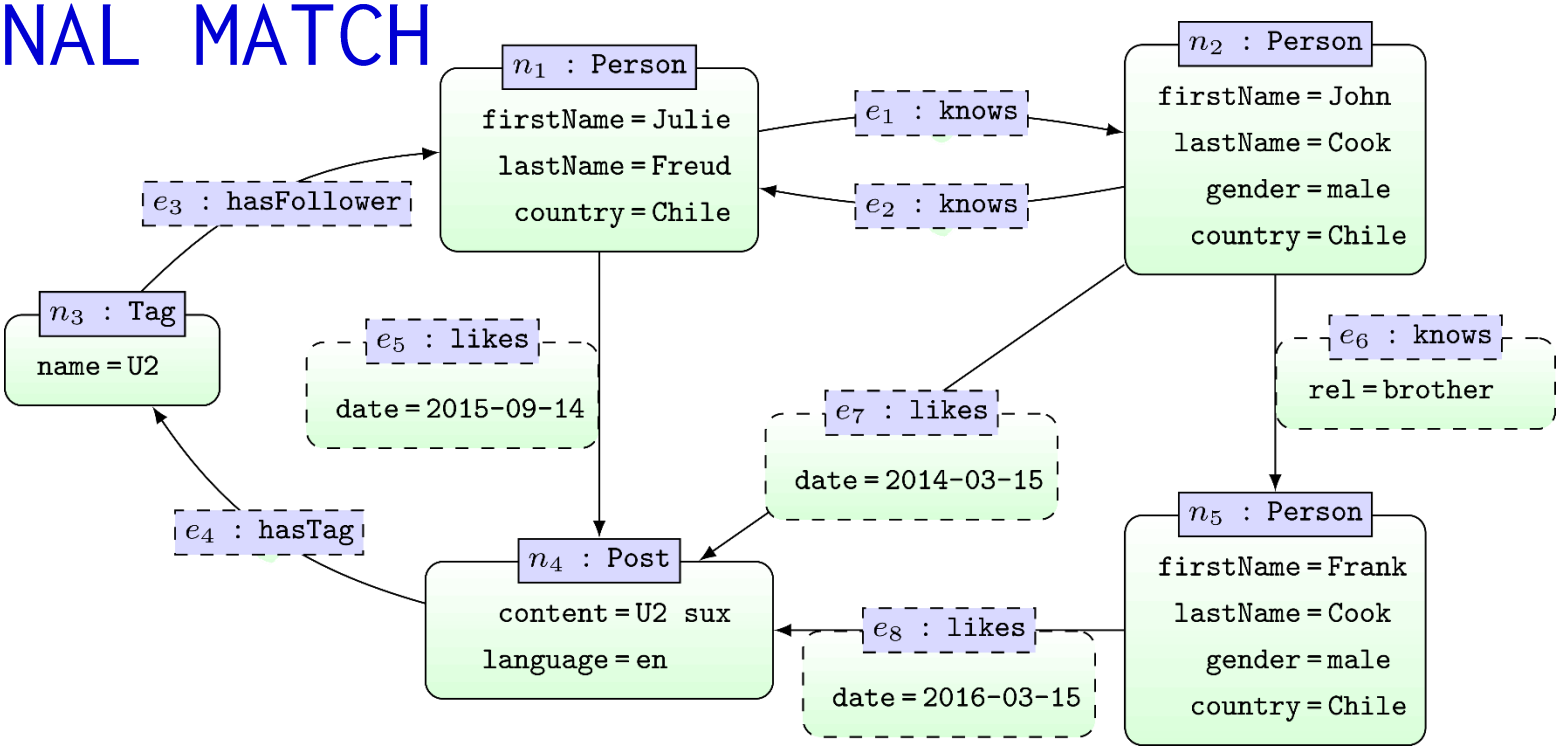

```
MATCH ()-[r:likes]->(p:Post)
RETURN r.date, p.content, p.language
ORDER BY p.content, r.date DESC
SKIP 1
LIMIT 1
```

r.date	p.content	p.language
2015-09-14	U2 sux	en

Cypher:
OPTIONAL MATCH

https://neo4j.com/docs/developer-manual/3.4/cypher/clauses/optional-match/

OPTIONAL MATCH




```
MATCH (x1)-[:knows]->(x2)
OPTIONAL MATCH (y)-[:hasFollower]->(x1)
RETURN x1.firstName,y.name
```

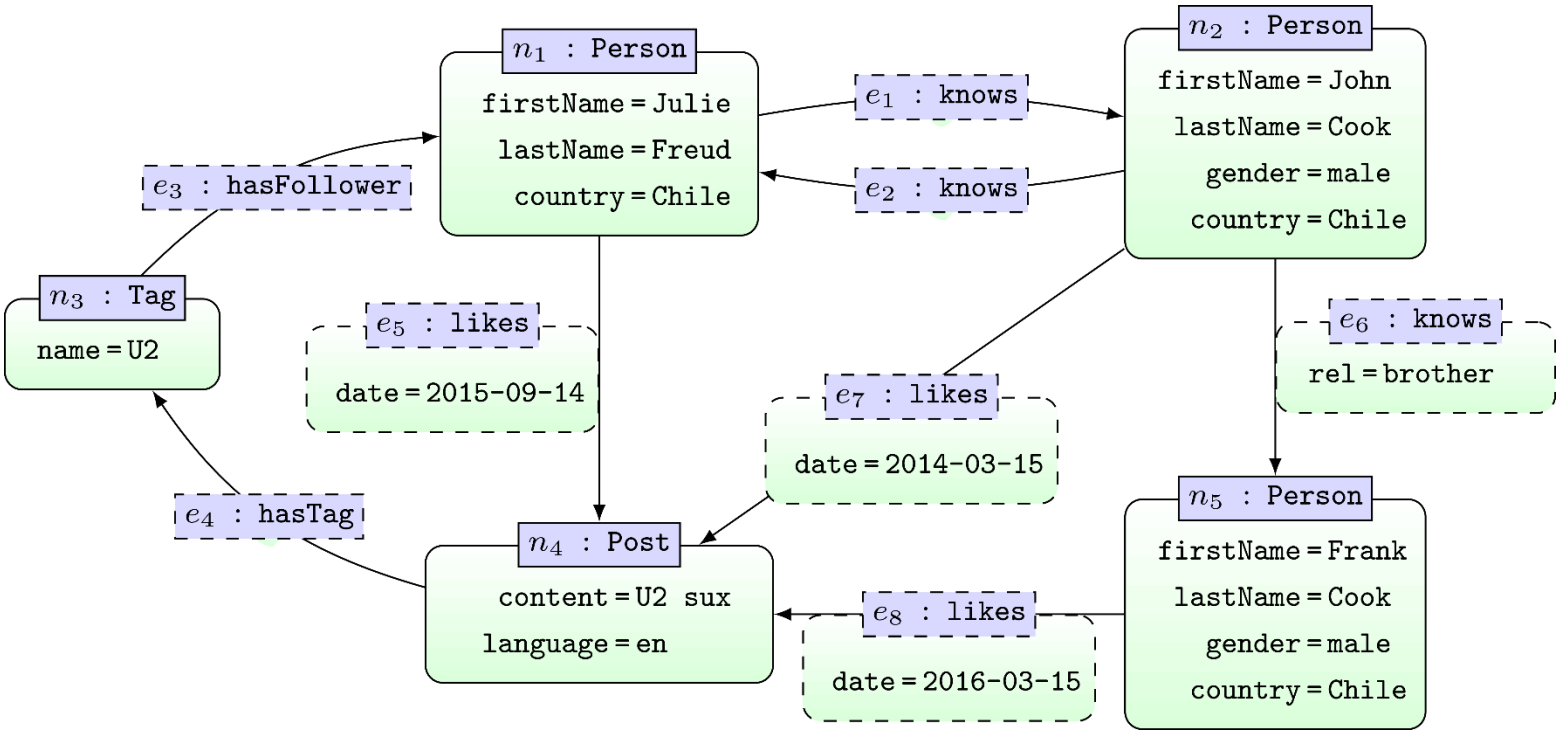
x1.firstName	y.name
Julie	U2
John	
John	

... OPTIONAL MATCH acts like a left join

Cypher:
UNION (ALL)

https://neo4j.com/docs/developer-manual/3.4/cypher/clauses/optional-match/

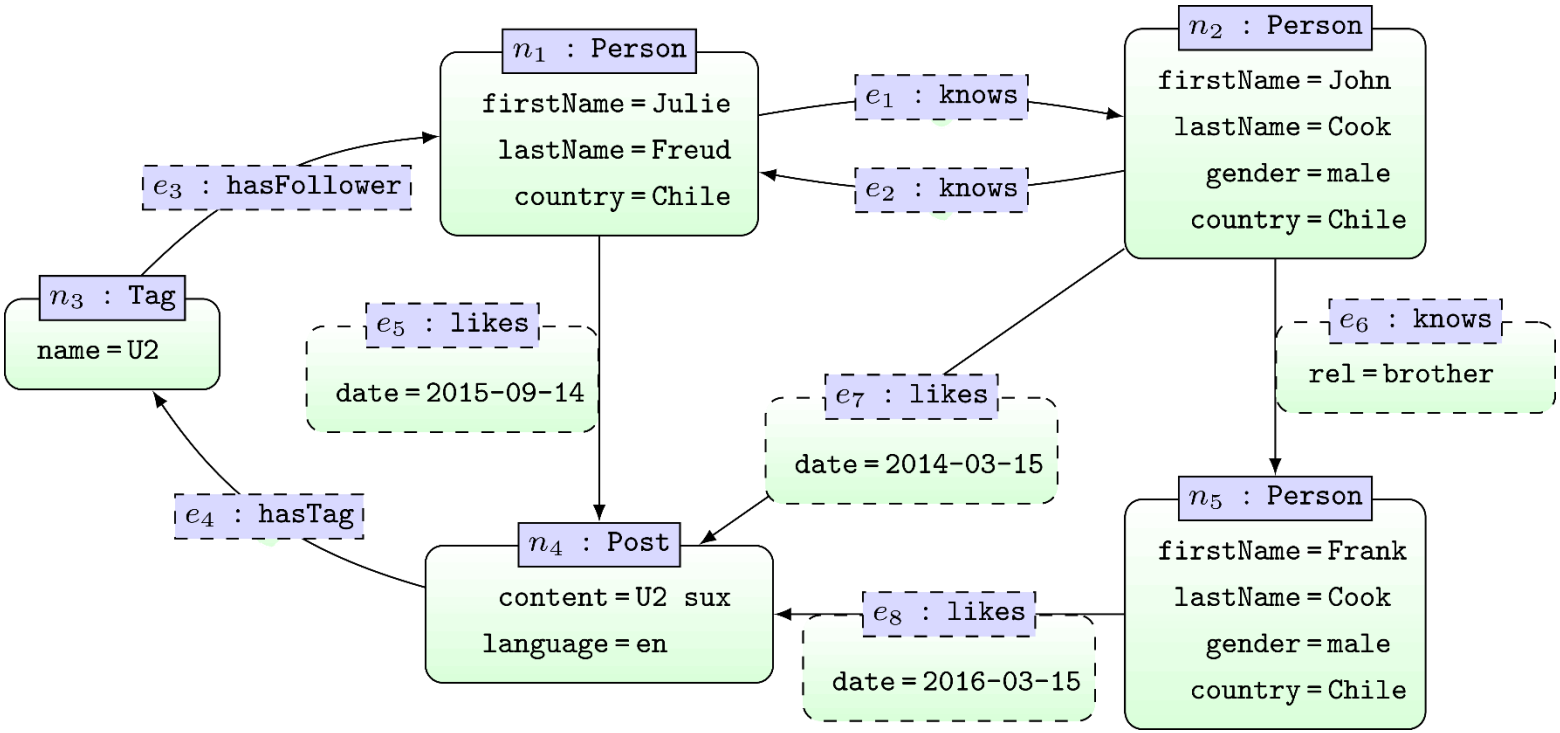
UNION




```
MATCH (x1)-[:knows]->(x2)
RETURN x1.firstName
UNION
MATCH (x1)-[:knows]->(x2)
RETURN x2.firstName
```

... column names have to be the same in the UNION

UNION

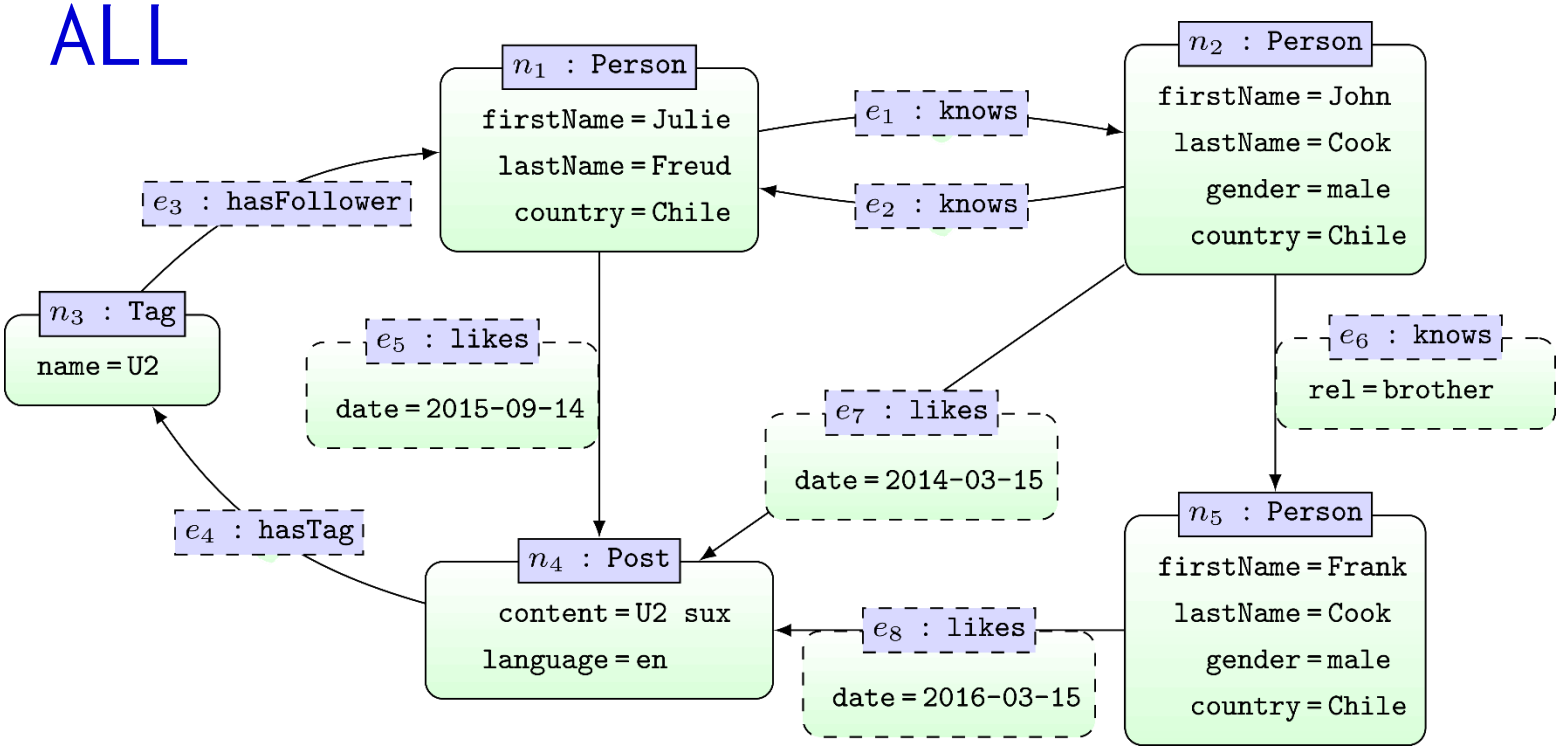



```
MATCH (x1)-[:knows]->(x2)
RETURN x1.firstName
UNION
MATCH (x2)-[:knows]->(x1)
RETURN x1.firstName
```

x1.firstName
Julie
John
Frank

... UNION applies set union

UNION ALL




```
MATCH (x1)-[:knows]->(x2)
RETURN x1.firstName
UNION ALL
MATCH (x2)-[:knows]->(x1)
RETURN x1.firstName
```

x1.firstName
Julie
Julie
John
John
John
Frank

... UNION ALL applies bag union

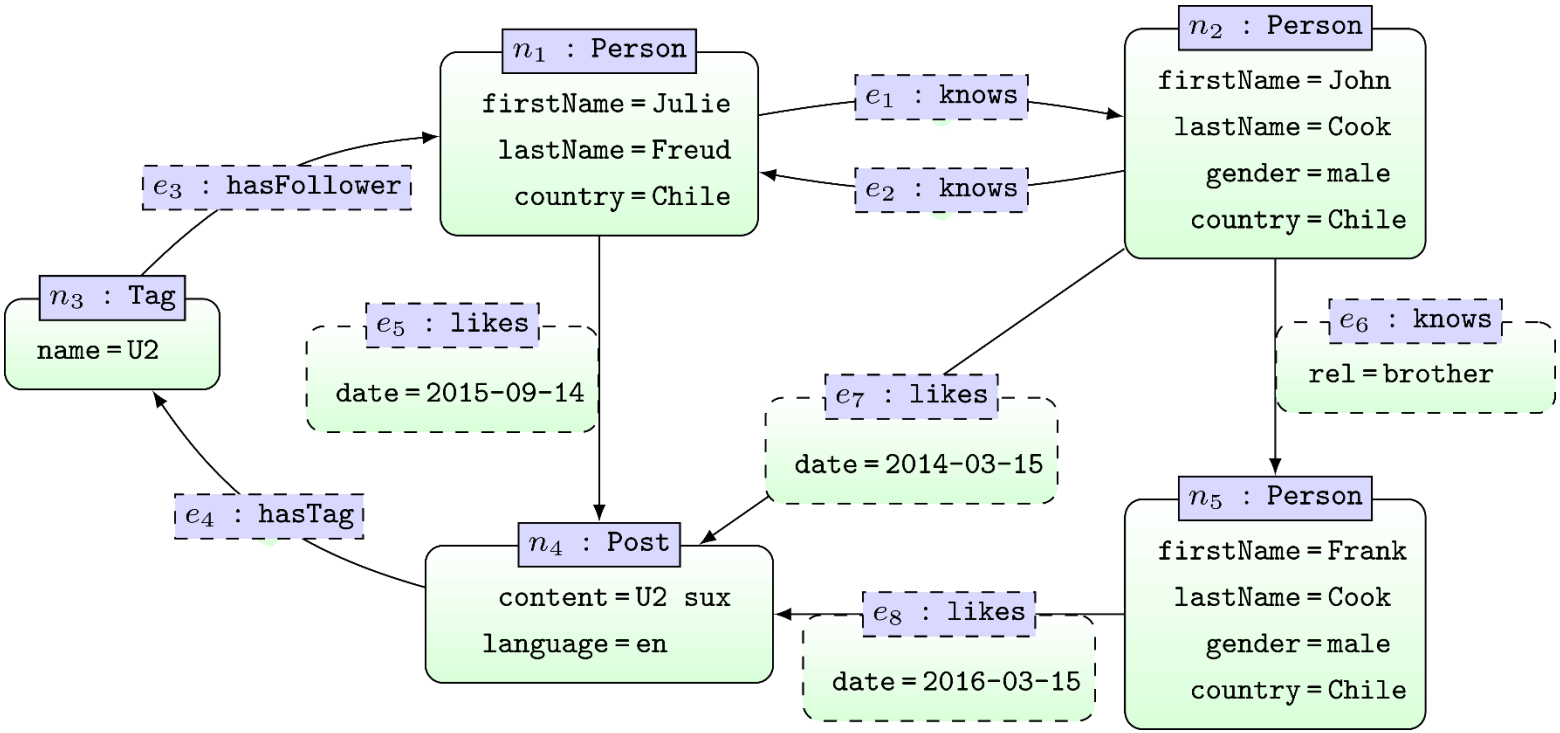
Cypher: Aggregation

https://neo4j.com/docs/developer-manual/current/cypher/functions/aggregating/

Aggregation

- `count`
- `max`/`min`
- `avg`
- `percentileCont`/`percentileDisc`
  - Computes percentile of some value w.r.t. some list
  - (*continuous*: interpolates / *discrete*: rounds)
- `stDev`/`stDevP`
  - Computes standard deviation (*sample*/*population*)

COUNT

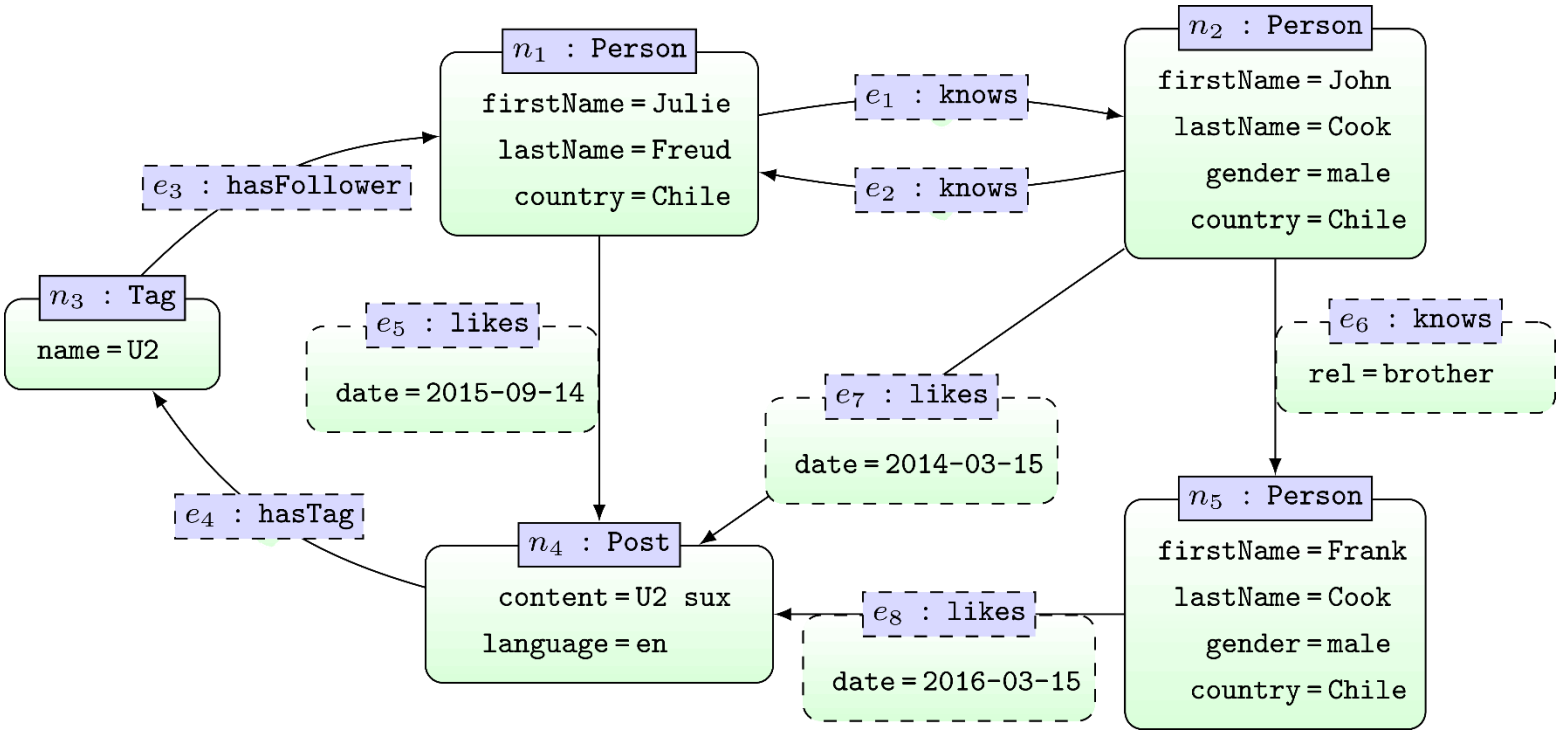



```
MATCH (x1)-[:knows*]->(x2)
RETURN x1.firstName, count(x2)
```

x1.firstName	count(x2)
Julie	3
John	4

... GROUP BY implicit on non-aggregated columns

COUNT

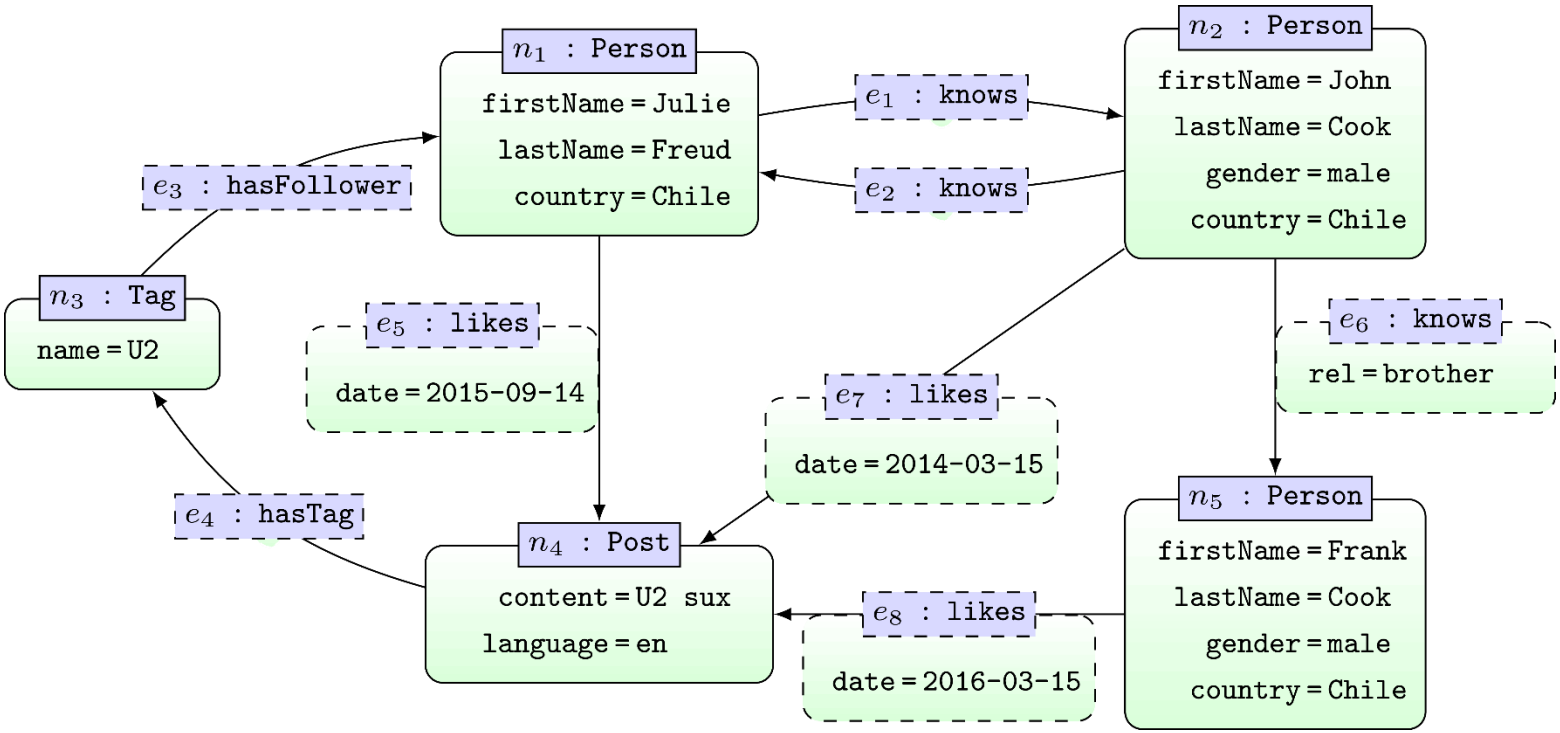



```
MATCH (x1)-[:knows*]->(x2)
RETURN DISTINCT x1.firstName, count(x2)
```

x1.firstName	count(x2)
Julie	3
John	4

... removes duplicate results, not count arguments

COUNT




```
MATCH (x1)-[:knows*]->(x2)
RETURN x1.firstName, count(distinct x2)
```

x1.firstName	count(x2)
Julie	3
John	3

... removes duplicate count arguments

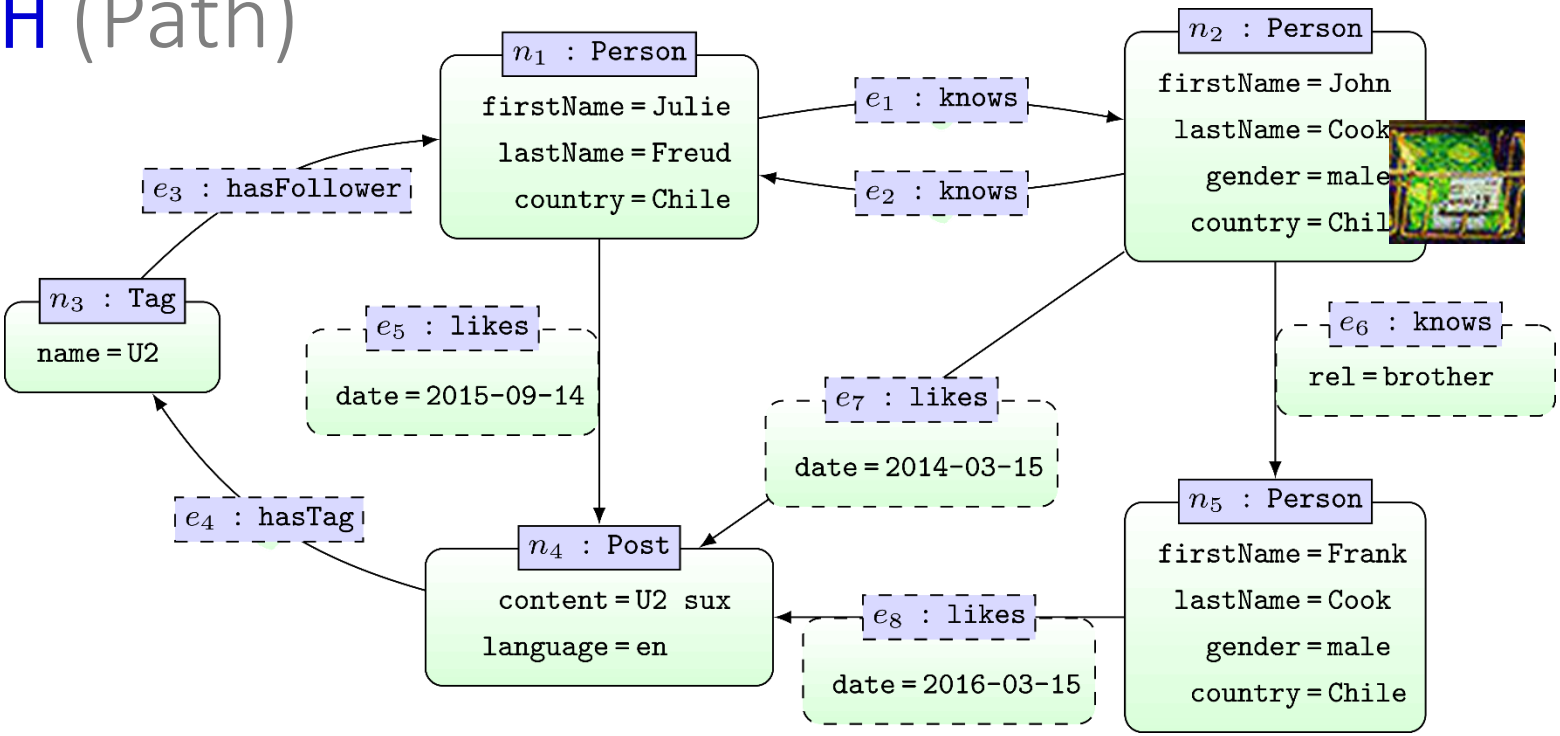
Cypher: Other Functions

https://neo4j.com/docs/developer-manual/current/cypher/functions/

https://neo4j.com/docs/developer-manual/current/cypher/functions/

https://neo4j.com/docs/developer-manual/current/cypher/functions/

LENGTH (Path)




```
MATCH (f {firstName: 'Frank'}),
  (j {firstName: 'Julie'}),
  p = shortestPath((f)-[*]->(j))
RETURN length(p)
```

length(p)
3

https://neo4j.com/docs/developer-manual/current/cypher/functions/

Cypher: Update Graphs

CREATE/REMOVE/...

Update graphs

- CREATE nodes and relationships
  - https://neo4j.com/docs/developer-manual/current/cypher/clauses/create/
- DELETE nodes and relationships
  - https://neo4j.com/docs/developer-manual/current/cypher/clauses/delete/
- DETACH DELETE nodes with relationships
  - https://neo4j.com/docs/developer-manual/current/cypher/clauses/delete/
- SET update labels and attributes
  - https://neo4j.com/docs/developer-manual/current/cypher/clauses/set/
- REMOVE remove labels and attributes
  - https://neo4j.com/docs/developer-manual/current/cypher/clauses/remove/

Update graphs

- Create the nodes we've seen

```
CREATE (:Person { firstName:'Julie', lastName:'Freud', country:'Chile' });
CREATE (:Person { firstName:'John', lastName:'Cook', country:'Chile', gender:'male' });
CREATE (:Tag { name:'U2' });
CREATE (:Post { content:'U2 sux', language:'en' });
CREATE (:Person { firstName:'Frank', lastName:'Cook', country:'Chile', gender:'male' });
```

- Create the edges (sample) we've seen

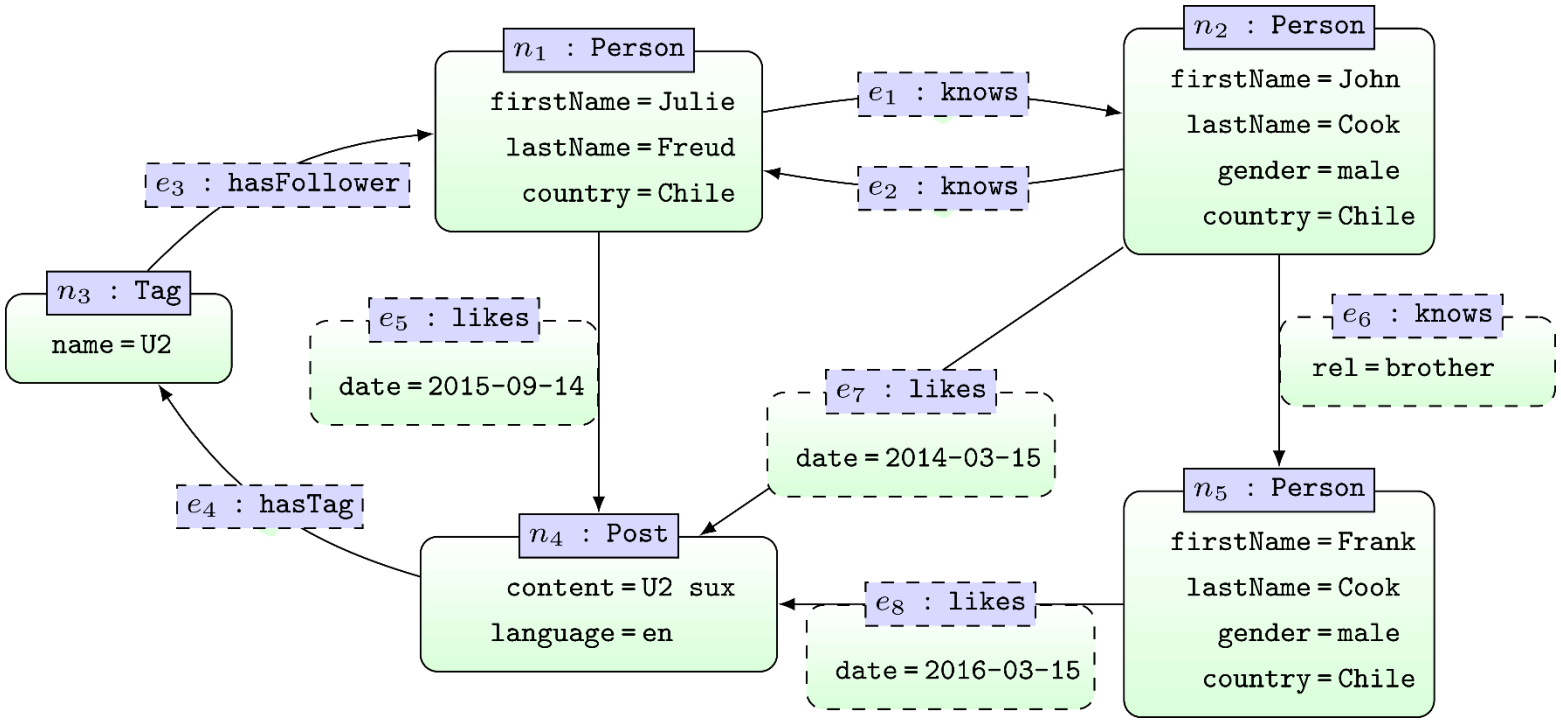
```
MATCH  (n1 { firstName:'Julie' }),(n2 { firstName:'John' }),(n3:Tag),(n4:Post),(n5 { firstName:'Frank' })
CREATE (n1)-[e1:knows]->(n2)
CREATE (n2)-[e2:knows]->(n1)
CREATE (n3)-[e3:hasFollower]->(n1)
CREATE (n4)-[e4:hasTag]->(n3)
CREATE (n1)-[e5:likes { date:'2015-09-14'}]->(n4)
CREATE (n2)-[e6:knows { rel:'brother'}]->(n5)
CREATE (n2)-[e7:likes { date:'2014-03-15'}]->(n4)
CREATE (n5)-[e8:likes { date:'2016-03-15'}]->(n4);...
```

- Drop all nodes and edges

```
MATCH (n) DETACH DELETE n;
```

/Core of Cypher

Property Graph




```
MATCH (x1:Person {firstName:"Julie"})-[:knows*]->(x2:Person)
MATCH (x2)-[:likes]->()-[:hasTag]->()-[:hasFollower]->(x1)
RETURN x2.firstName
```

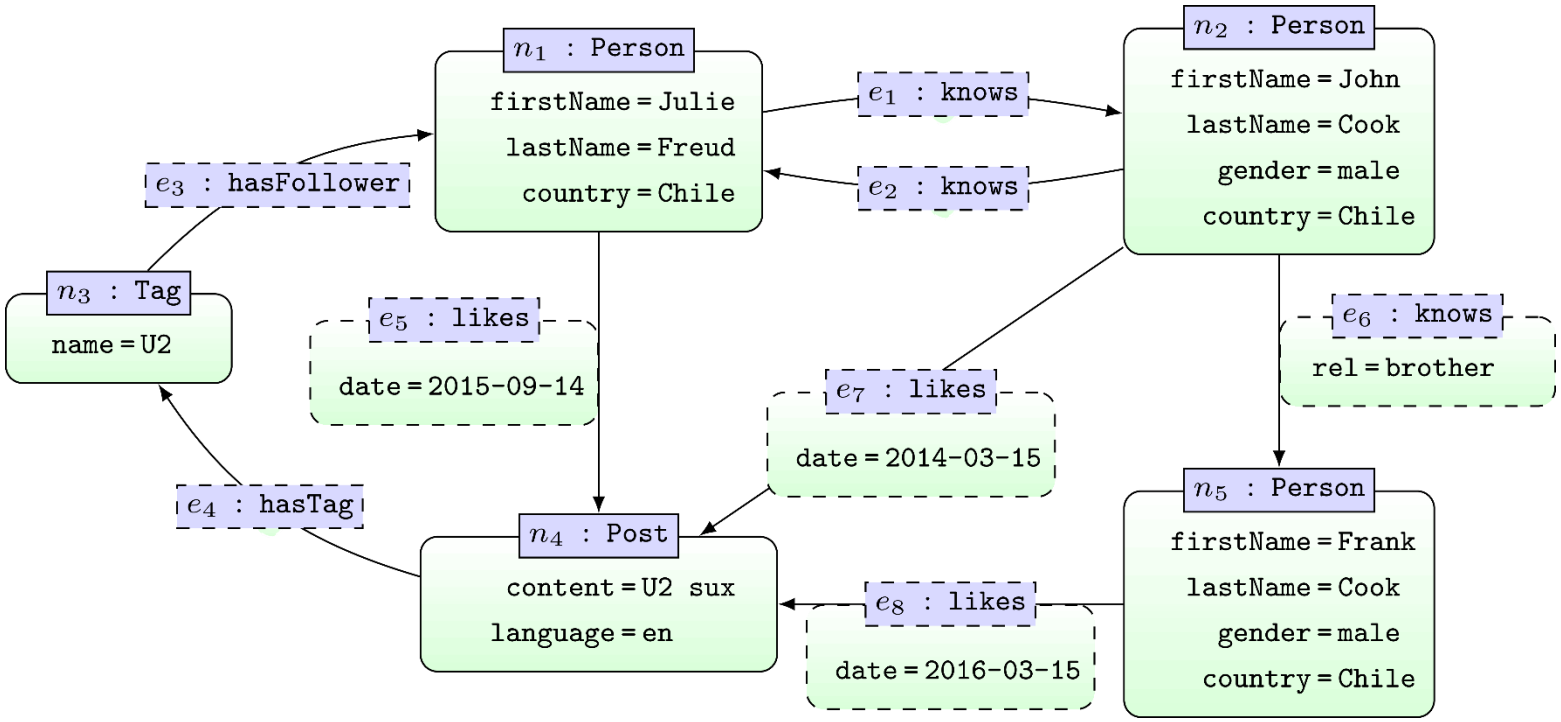
x2.firstName
Julie
John
Frank

/Core of Cypher
/Part of Neo4j

Neo4j Graph Database

- Data Model: Property Graphs
- Query Language: Cypher
- Scripting Language: Gremlin
- Licence: Open Source (Single Machine)
  Commercial (Cluster Edition)

Property Graph: Cypher

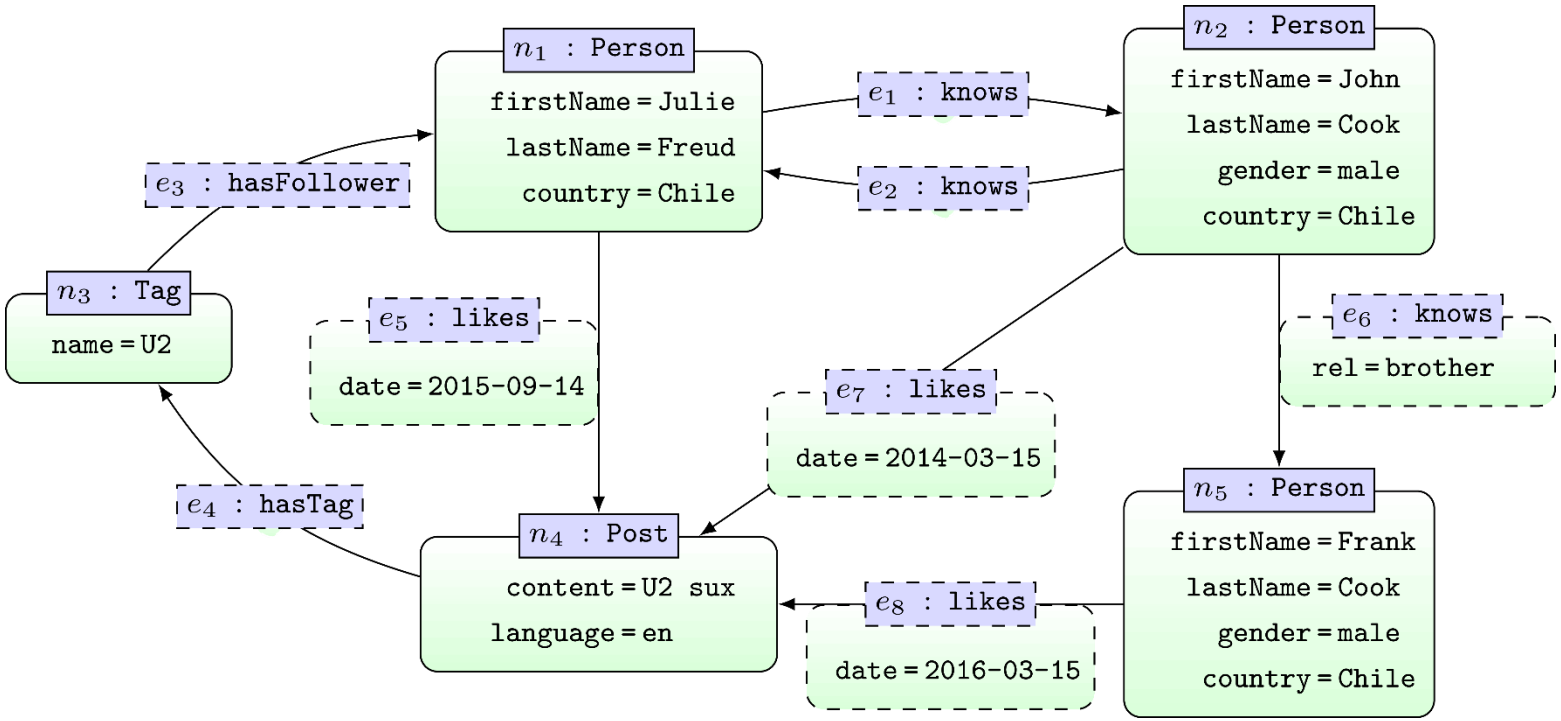



```
MATCH (x1:Person {firstName:"Julie"})-[:knows*]->(x2:Person)
MATCH (x2)-[:likes]->()-[:hasTag]->()-[:hasFollower]->(x1)
RETURN x2.firstName
```

x2.firstName
Julie
John
Frank

neo4j

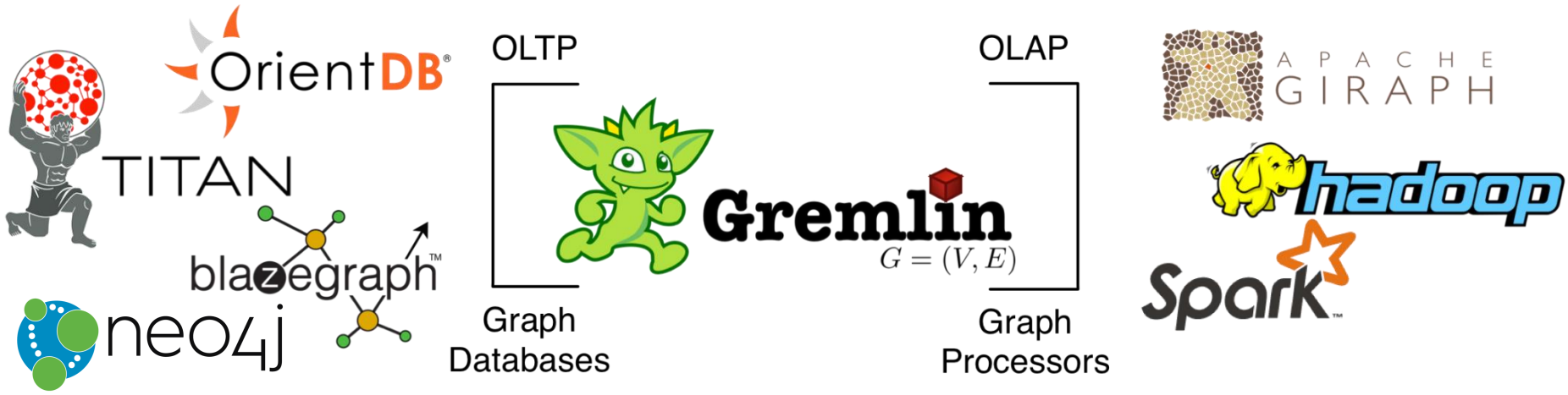
Property Graph: Gremlin




```
n1 = g.v.filter{firstName:"Julie"}.inV.outE('knows').loop()
n2 = n1.call().more.functions().until('done')
```

x2.firstName
Julie
John
Frank

Gremlin: Graph Queries + Processing



DB-Engines Ranking of Graph DBMS

The DB-Engines Ranking ranks database management systems according to their popularity. The ranking is updated monthly.

trend chart

This is a partial list of the complete ranking showing only graph DBMS.

Read more about the method of calculating the scores.

31 systems in ranking, May 2018

Rank			DBMS	Database Model	Score		
May 2018	Apr 2018	May 2017			May 2018	Apr 2018	May 2017
1.	1.	1.	Neo4j	Graph DBMS	40.58	-0.32	+4.44
2.	2.	↑ 4.	Microsoft Azure Cosmos DB	Multi-model	17.54	+0.35	+12.70
3.	3.		Datastax Enterprise	Multi-model	7.38	-0.09	
4.	4.	↓ 2.	OrientDB	Multi-model	5.25	-0.39	-0.49
5.	5.	5.	ArangoDB	Multi-model	3.70	-0.10	+0.75
6.	6.	6.	Virtuoso	Multi-model	1.79	-0.01	-0.27
7.	7.	7.	Giraph	Graph DBMS	0.98	-0.06	-0.11
8.	8.		Amazon Neptune	Multi-model	0.71	+0.02	
9.	9.	↓ 8.	AllegroGraph	Multi-model	0.58	+0.00	-0.02
10.	10.	↓ 9.	Stardog	Multi-model	0.51	-0.02	+0.00
11.	11.	↓ 10.	GraphDB	Multi-model	0.46	-0.00	-0.04
12.	↑ 14.	↑ 19.	JanusGraph	Graph DBMS	0.41	+0.12	+0.29
13.	↓ 12.	↑ 16.	Graph Engine	Multi-model	0.36	-0.04	+0.18
14.	↓ 13.	↓ 11.	Sqrrl	Multi-model	0.33	-0.06	-0.13
15.	15.	↑ 22.	Sparksee	Graph DBMS	0.19	-0.02	+0.14
16.	16.		TigerGraph	Graph DBMS	0.17	-0.01	
17.	↑ 20.	↓ 14.	Blazegraph	Multi-model	0.14	+0.01	-0.13
18.	18.	↓ 12.	Dgraph	Graph DBMS	0.14	+0.00	-0.15
19.	↓ 17.	↓ 17.	HyperGraphDB	Graph DBMS	0.14	-0.01	-0.02
20.	↓ 19.	↓ 15.	FlockDB	Graph DBMS	0.13	+0.00	-0.06
21.	↑ 23.	↓ 13.	InfiniteGraph	Graph DBMS	0.13	+0.02	-0.15
22.	22.	↓ 20.	FaunaDB	Multi-model	0.11	+0.00	+0.05
23.	↑ 24.	23.	VelocityDB	Multi-model	0.10	+0.02	+0.06
24.	↓ 21.	↓ 18.	InfoGrid	Graph DBMS	0.10	-0.02	-0.03
25.	↑ 26.	25.	AgensGraph	Multi-model	0.04	+0.01	+0.03

DB-Engines Ranking

The DB-Engines Ranking ranks database management systems according to their popularity. The ranking is updated monthly.

Read more about the method of calculating the scores.

trend chart

342 systems in ranking, May 2018

Rank			DBMS	Database Model	Score		
May 2018	Apr 2018	May 2017			May 2018	Apr 2018	May 2017
1.	1.	1.	Oracle	Relational DBMS	1290.42	+0.63	-63.90
2.	2.	2.	MySQL	Relational DBMS	1223.34	-3.06	-116.69
3.	3.	3.	Microsoft SQL Server	Relational DBMS	1085.84	-9.67	-127.96
4.	4.	4.	PostgreSQL	Relational DBMS	400.90	+5.43	+34.99
5.	5.	5.	MongoDB	Document store	342.11	+0.70	+10.53
6.	6.	6.	DB2	Relational DBMS	185.61	-3.34	-3.23
7.	↑ 9.	↑ 9.	Redis	Key-value store	135.35	+5.24	+17.90
8.	↓ 7.	↓ 7.	Microsoft Access	Relational DBMS	133.11	+0.89	+3.24
9.	↓ 8.	↑ 11.	Elasticsearch	Search engine	130.44	-0.92	+21.62
10.	10.	↓ 8.	Cassandra	Wide column store	117.83	-1.26	-5.28
11.	11.	↓ 10.	SQLite	Relational DBMS	115.45	-0.53	-0.61
12.	12.	12.	Teradata	Relational DBMS	74.41	+0.74	-1.91
13.	13.	↑ 16.	Splunk	Search engine	65.09	+0.04	+8.40
14.	14.	↑ 18.	MariaDB	Relational DBMS	64.99	+0.44	+14.01
15.	15.	↓ 14.	Solr	Search engine	61.51	-1.70	-2.26
16.	16.	↓ 13.	SAP Adaptive Server	Relational DBMS	61.51	-0.12	-6.24
17.	17.	↓ 15.	HBase	Wide column store	59.95	+0.26	+0.44
18.	18.	↑ 20.	Hive	Relational DBMS	56.97	-0.43	+13.49
19.	19.	↓ 17.	FileMaker	Relational DBMS	54.67	-0.33	-1.81
20.	20.	↓ 19.	SAP HANA	Relational DBMS	48.37	-0.52	-0.68
21.	21.	↑ 22.	Amazon DynamoDB	Multi-model	44.19	+1.05	+10.99
22.	22.	↓ 21.	Neo4j	Graph DBMS	40.58	-0.32	+4.44
23.	23.	↑ 24.	Memcached	Key-value store	33.56	-0.23	+4.15
24.	24.	↓ 23.	Couchbase	Document store	32.41	+0.07	+0.16
25.	25.	25.	Informix	Relational DBMS	25.79	-0.82	-2.44

Questions?