Graph Databases will change your (freakin') life

Elena Williams

PyCon(line)AU 2020

Hi,

I’m Elena. I am a web developer and I love python.
github.com/**elena**
twitter.com/**elequ**

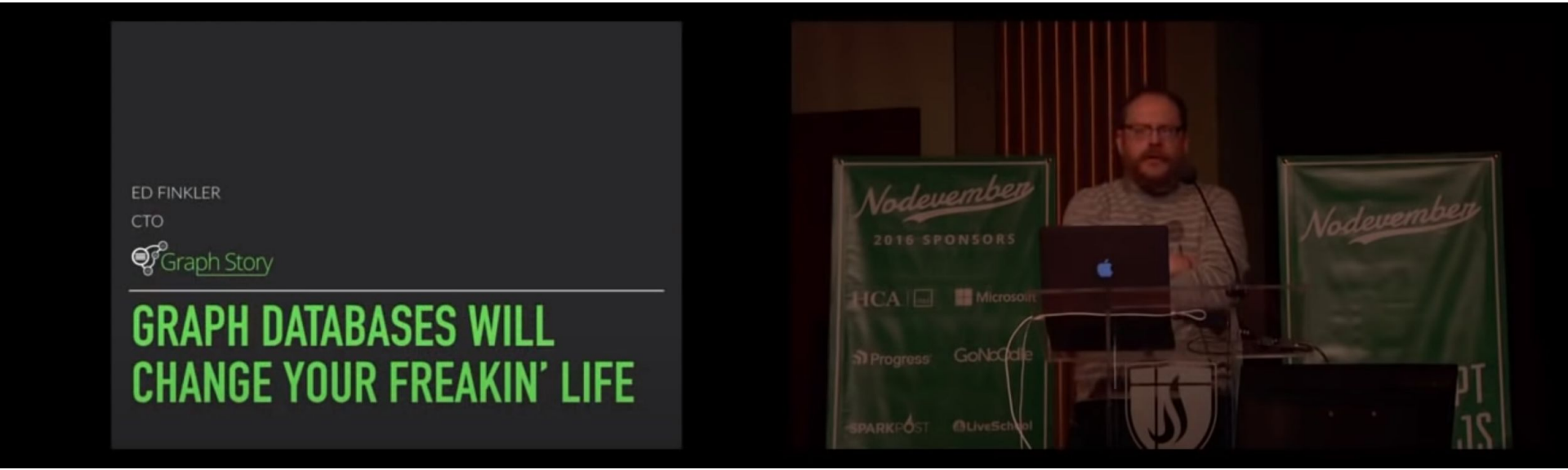
Proudly help organise
Canberra Python User Group

Join us on **Slack**
*github.com/**canberra-python***

10 - 12 September 2021
https://**2021**.pycon.org.**au**/
Please ***Volunteer***

Why?

Ed Finkler (**twitter**.com/**funkatron**)
*https://github.com/**OSMI**Help*



https://**github**.com
/**elena**
/**graph-fun**

Search or jump to… / Pull requests Issues Marketplace Explore

elena / graph-fun

Unwatch 1 Star 3 Fork 0

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

master 1 branch 0 tags Go to file Add file Code

elena Removing scratchy crufty-cruft and bail-out. Goodness. 680c63b on 18 Sep 2020 5 commits

media	Add talk materials and credits :)	7 months ago
README.md	More intuitive link action and pro CPUG propaganda	7 months ago
app.py	Add talk materials and credits :)	7 months ago
credits.md	TYPO!	7 months ago
derp.cypher	Add talk materials and credits :)	7 months ago
example.cypher	Removing scratchy crufty-cruft and bail-out. Goodness.	7 months ago
funkatron.png	Add talk materials and credits :)	7 months ago
notebook.ipynb	Add talk materials and credits :)	7 months ago
pyconline2020_graph-databases.png	Add talk materials and credits :)	7 months ago
requirements.txt	Add talk materials and credits :)	7 months ago

About

PyConlineAU 2020 Slides and Demos

Readme

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Languages

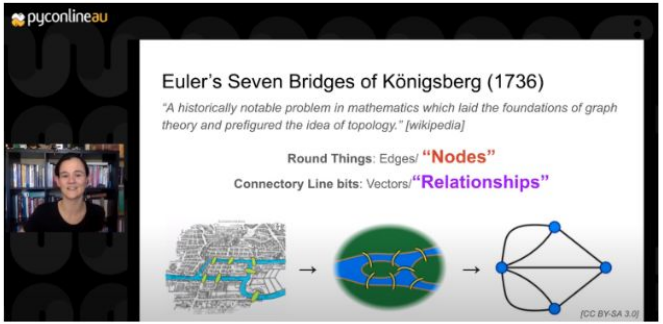
Jupyter Notebook 69.8%
Python 30.2%

README.md

PyConlineAU 2020 Talk

Graph Databases will Change Your Life

Full talk on Youtube:



Link to video: PyConlineAU 2020: Graph Databases will Change Your Life

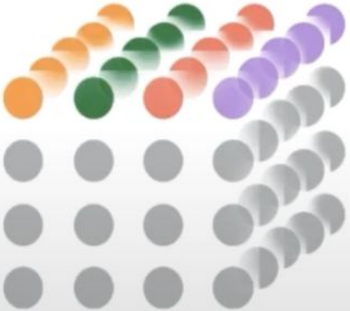
Goal

***Example** Graph DB:*

Actual “**Paradise Papers**” database
created by **ICIJ**
(International Consortium of Investigative Journalists)

Plan of Attack:

1. **Graphs** 101
2. **DBs** v. **Graph DBs**
3. **Usage** and **Applications**
4. **Demos**



Euler's Seven Bridges of Königsberg (1736)

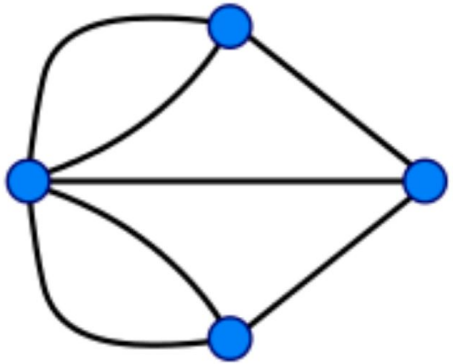
"A historically notable problem in mathematics which laid the foundations of graph theory and prefigured the idea of topology." [wikipedia]

Round Things: Edges/ **"Nodes"**

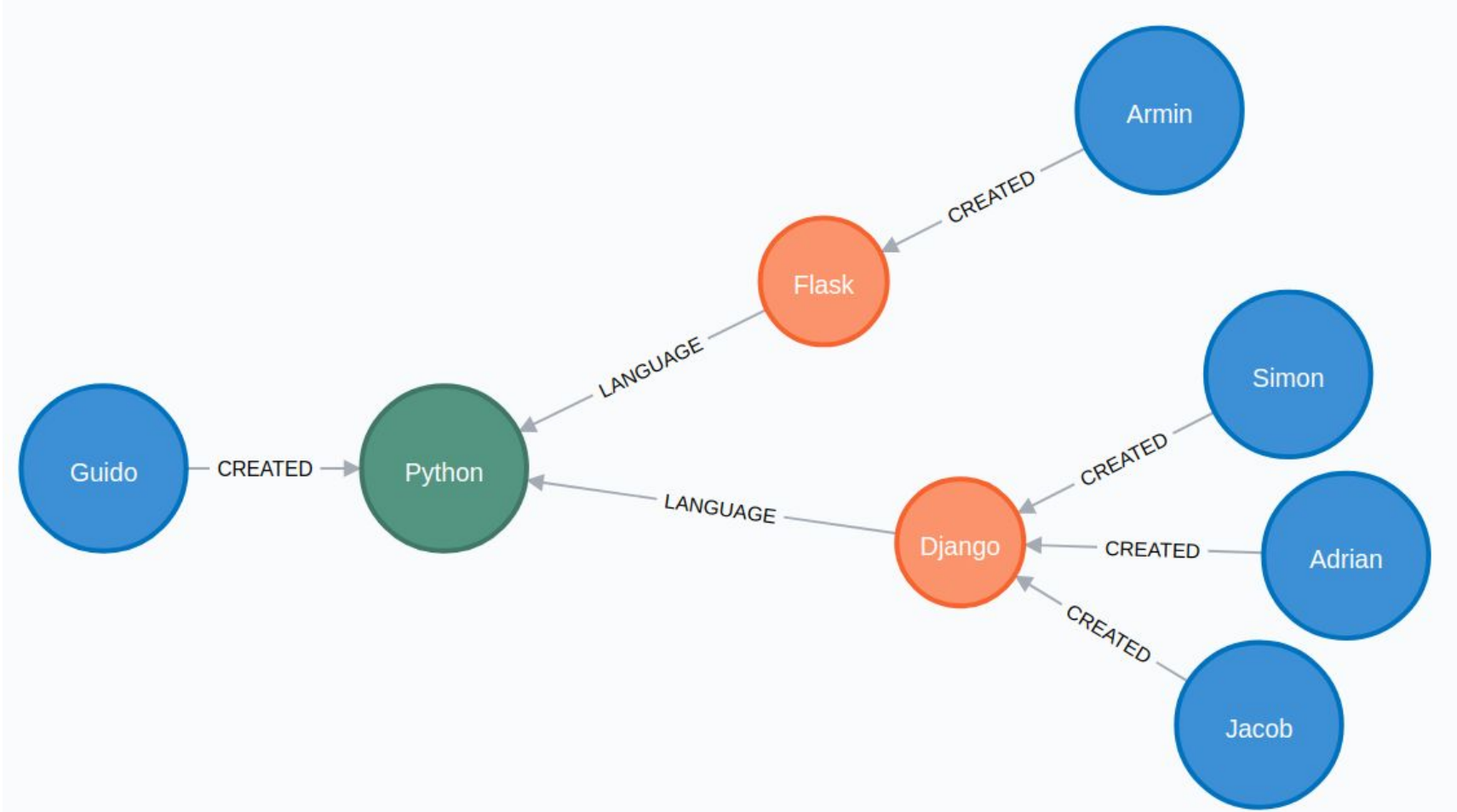
Connectory Line bits: Vectors/**"Relationships"**

Round Things: Edges/ **“Nodes”**

Connectory Line bits: Vectors/**“Relationships”**



Displaying 2 nodes, 1 relationships.



Displaying 8 nodes, 7 relationships.

:Person {
name: **Alice**
interests: ultra-marathons
}

:Department {
name: **IT Department**
}

:Person {
name: **Bob**
full_name: Robert Smith
interests: volleyball
}

:Person {
  name: Alice
  interests: ultra-marathons
}

:Department {
  name: IT Department
}

:Person {
  name: Bob
  full_name: Robert Smith
  interests: volleyball
}

```
class Person (models.Model):
  name = models.CharField(max_length=100)
  full_name = models.CharField(max_length=200)
  interests = models.TextField()


class WorksAt (models.Model):
  start_date = models.DateField()
  person = models.ForeignKey("Person", ... )
  ...
```

:Person {
name: **Alice**
interests: ultra-marathons
}

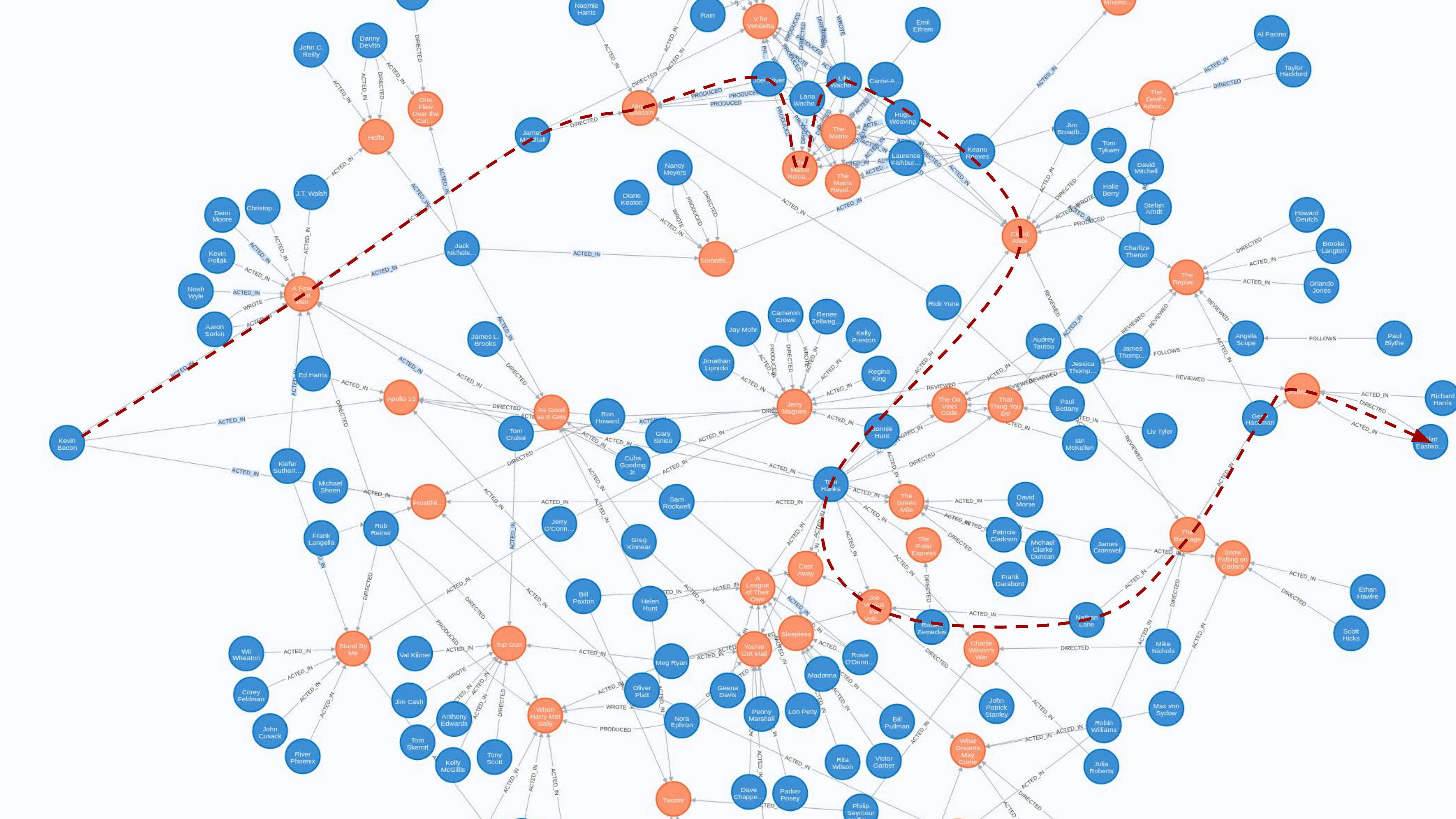
:Department {
name: **IT Department**
}

:Person {
name: **Bob**
full_name: Robert Smith
interests: volleyball
}


```
:Person:Engineer:Runner {
   name: Alice
   interests: ultra-marathons
}
```

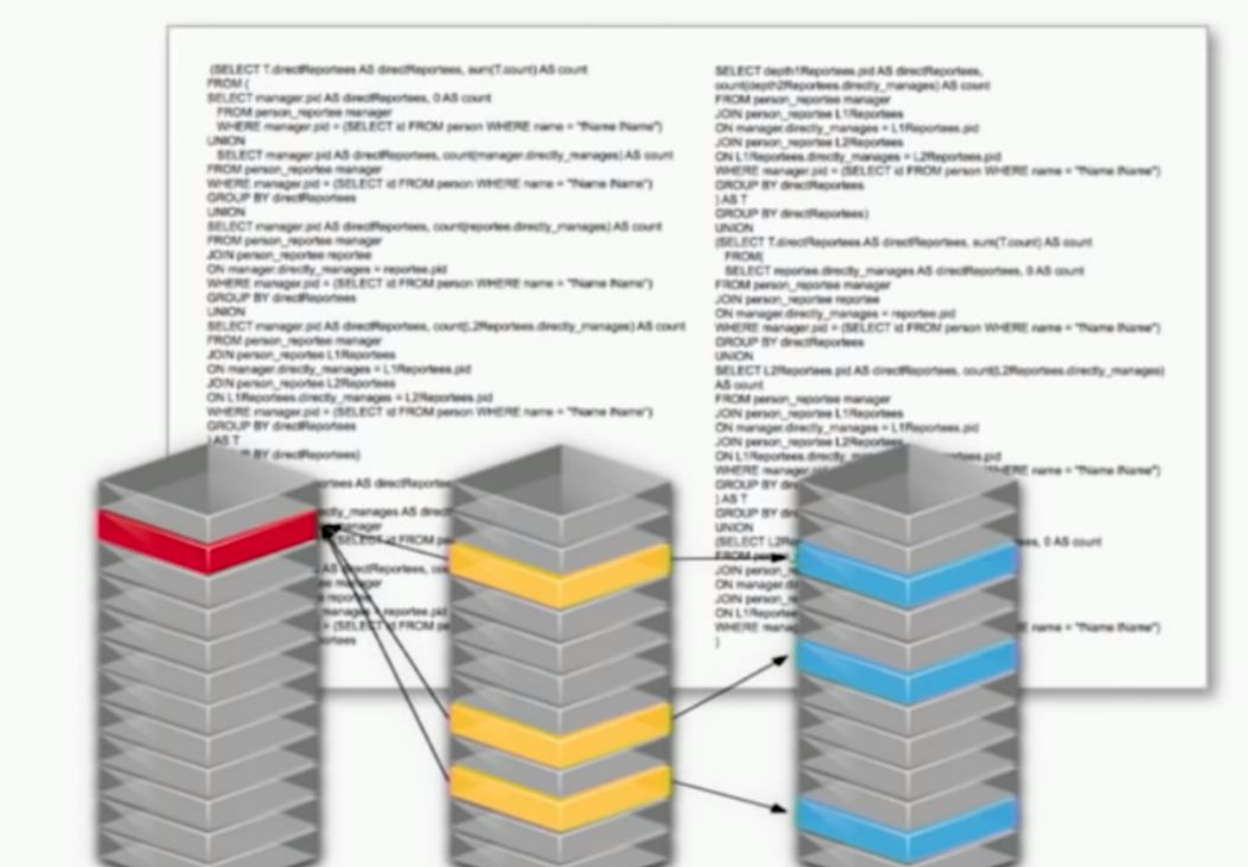
```
:Department:Group {
   name: IT Department
}
```

```
:Person {
   name: Bob
   full_name: Robert Smith
   interests: volleyball
}
```

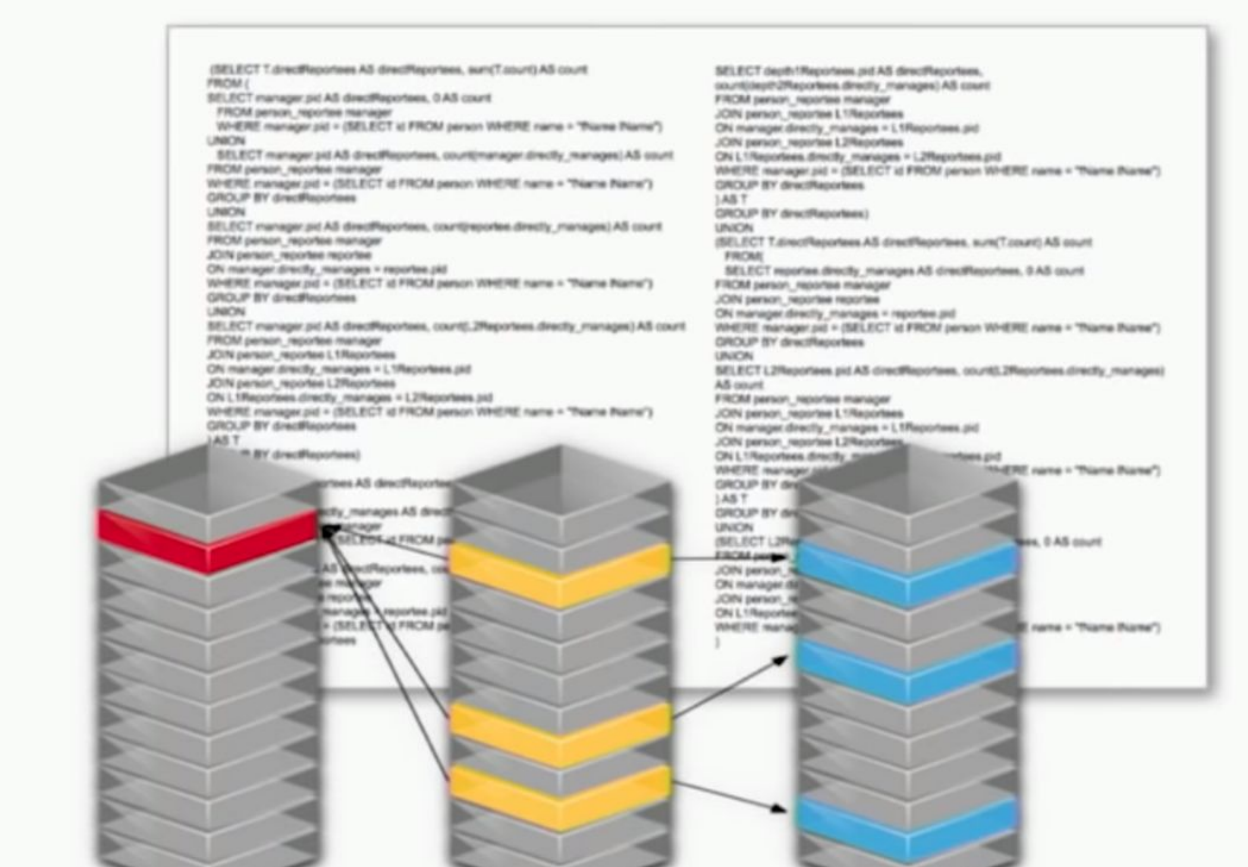


Kevin Bacon
A Few Good Men
Demi Moore
Christop...
Kevin Pollak
Noah Wyle
Aaron Sorkin
J.T. Walsh
John C. Reilly
Danny DeVito
Hoffa
One Flew Over the Cuc...
James Marshall
Jack Nicholson
Ninja Assassin
Naomie Harris
Rain
V for Vendetta
Joel Silver
Lana Wacho...
Lilly Wacho...
Carrie-A...
Hugo Weaving
The Matrix
The Matrix Reloa...
The Matrix Revol...
Laurence Fishbur...
Keanu Reeves
Emil Eifrem
Nancy Meyers
Diane Keaton
Somethi...
Cloud Atlas
Jim Broadb...
Tom Tykwer
David Mitchell
Halle Berry
Stefan Arndt
The Devil's Advoc...
Al Pacino
Taylor Hackford
Charlize Theron
Howard Deutch
Brooke Langton
Orlando Jones
The Replac...
Angela Scope
Paul Blythe
James Thomp...
Jessica Thomp...
Audrey Tautou
Rick Yune
Cameron Crowe
Renee Zellweg...
Jay Mohr
Kelly Preston
Jonathan Lipnicki
Regina King
Jerry Maguire
The Da Vinci Code
That Thing You Do
Paul Bettany
Ian McKellen
Liv Tyler
Gene Hackman
Richard Harris
Clint Eastwo...
James L. Brooks
Ed Harris
Apollo 13
As Good as It Gets
Ron Howard
Tom Cruise
Gary Sinise
Cuba Gooding Jr.
Bonnie Hunt
Tom Hanks
The Green Mile
David Morse
Patricia Clarkson
Michael Clarke Duncan
James Cromwell
Frank Darabont
The Polar Express
Kiefer Sutherl...
Michael Sheen
Frost/Ni...
Sam Rockwell
Frank Langella
Rob Reiner
Jerry O'Conn...
Greg Kinnear
Cast Away
A League of Their Own
Joe Versus the Volc...
Robert Zemeckis
Nathan Lane
The Birdcage
Snow Falling on Cedars
Ethan Hawke
Scott Hicks
Mike Nichols
Bill Paxton
Helen Hunt
Sleepless...
You've Got Mail
Rosie O'Donn...
Charlie Wilson's War
Wil Wheaton
Stand By Me
Corey Feldman
John Cusack
River Phoenix
Val Kilmer
Top Gun
Jim Cash
Anthony Edwards
Tom Skerritt
Kelly McGillis
Tony Scott
When Harry Met Sally
Meg Ryan
Oliver Platt
Nora Ephron
Geena Davis
Madonna
Penny Marshall
Lori Petty
Bill Pullman
Rita Wilson
Victor Garber
John Patrick Stanley
What Dreams May Come
Robin Williams
Julia Roberts
Max von Sydow
Dave Chappe...
Parker Posey
Philip Seymour
Twister
ACTED_IN
DIRECTED
PRODUCED
WROTE
REVIEWED
FOLLOWS

Relational Databases (RDBMS) aka **SQL**



“Relational” **Databases** (RDBMS) aka **SQL**



“Relational”

Are RDBMS (aka **SQL**) **good with relationships?**

Are RDBMS (aka **SQL**) **good with relationships?**

- Can’t do **complex models**

  Literally small-dimensional tables.

- Can’t **scale joins** that efficiently

  B-Tree Index: $O(n\ log(n))$, data grows 10x = speed halves

  more data → more slow

- SQL was built on **SET theory**

  not graph theory, ie: relationships are really only by *coincidence*.

- Can’t easily **change schema**

Are RDBMS (aka **SQL**) **good with relationships?**

- Can’t do **complex models**

  Literally small-dimensional tables.

- Can’t **scale joins** that efficiently

  B-Tree Index: $O(n\ log(n))$, data grows 10x = speed halves

  more data → more slow

- SQL was built on **SET theory**

  not graph theory, ie: relationships are really only by *coincidence*.

- Can’t easily **change schema**

Are RDBMS (aka **SQL**) **good with relationships?**

- Can’t do **complex models**

  Literally small-dimensional tables.

- Can’t **scale joins** that efficiently

  B-Tree Index: $O(n\ log(n))$, data grows 10x = speed halves

  more data → more slow

- SQL was built on **SET theory**

  not graph theory, ie: relationships are really only by *coincidence*.

- Can’t easily **change schema**

Are RDBMS (aka **SQL**) **good with relationships?**

- Can’t do **complex models**

  Literally small-dimensional tables.

- Can’t **scale joins** that efficiently

  B-Tree Index: $O(n\ log(n))$, data grows 10x = speed halves

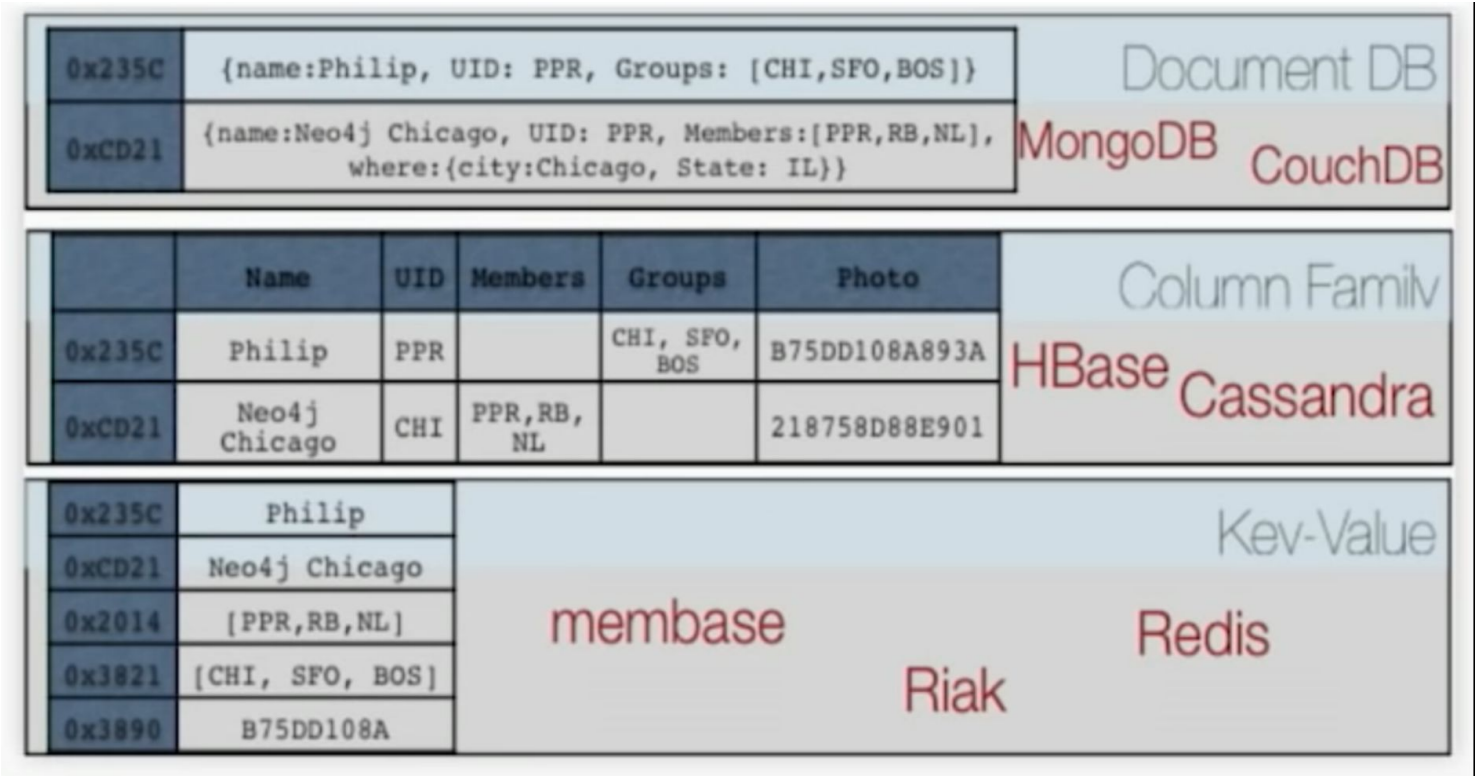
  more data → more slow

- SQL was built on **SET theory**

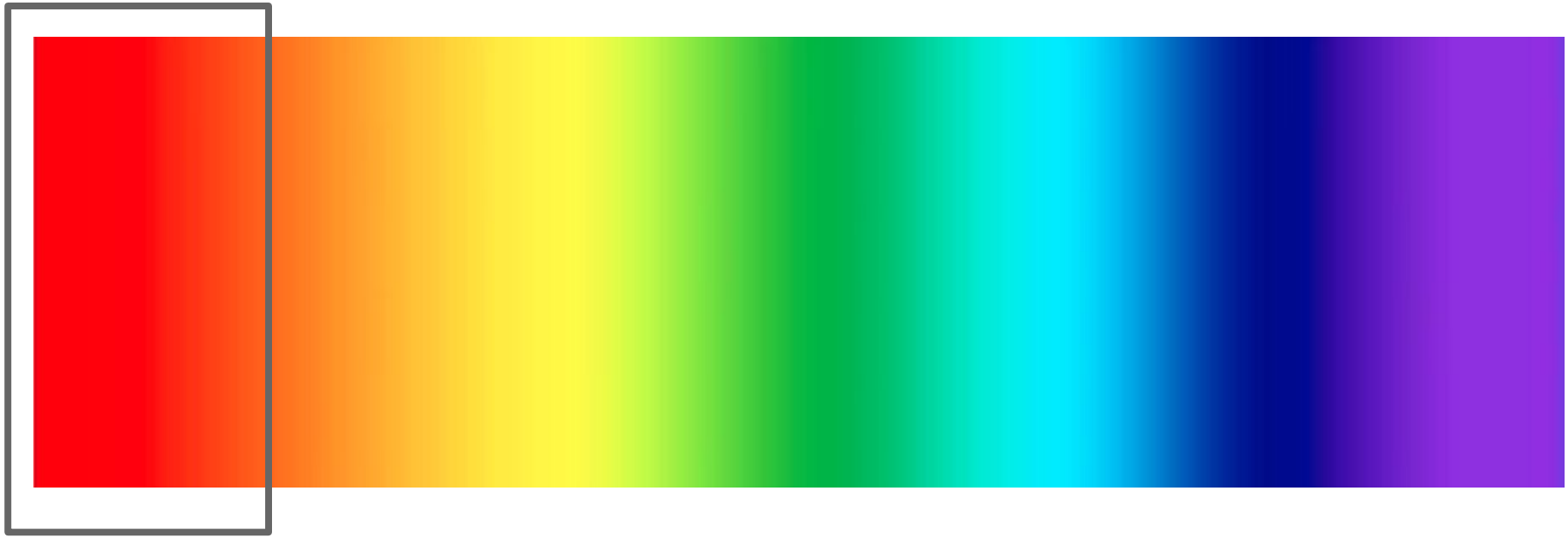
  not graph theory, ie: relationships are really only by *coincidence*.

- Can’t easily **change schema**

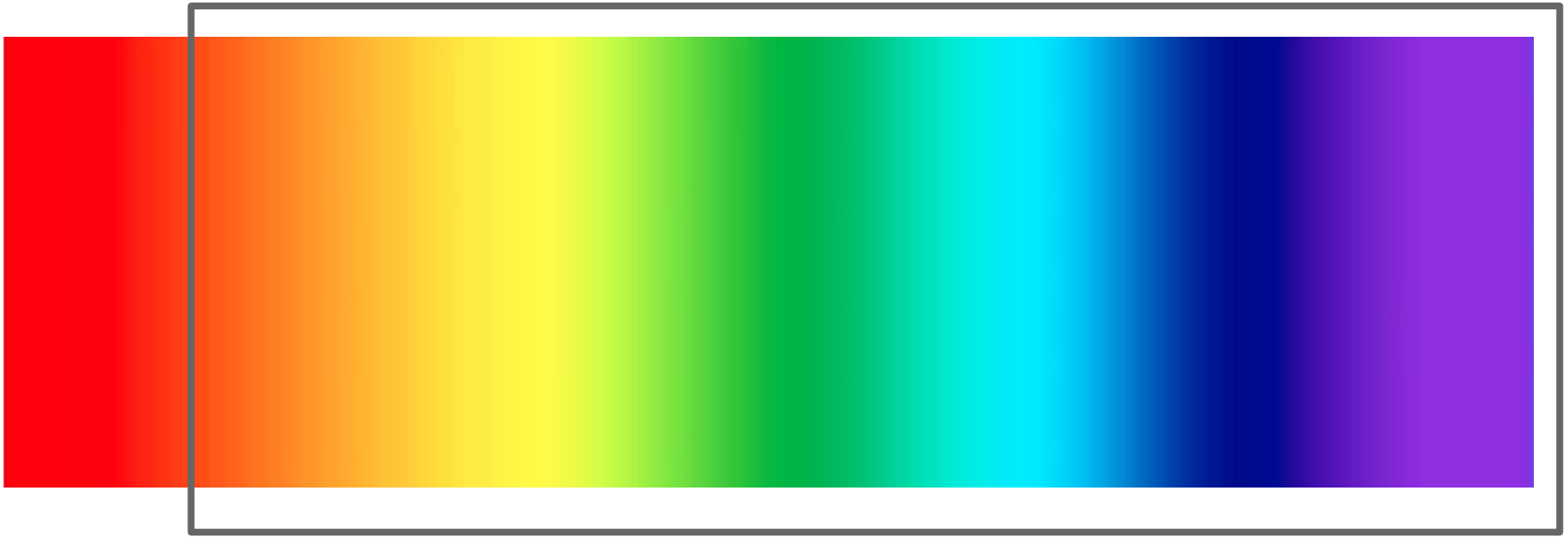
NoSQL


0x235C
{name:Philip, UID: PPR, Groups: [CHI,SFO,BOS]}
0xCD21
{name:Neo4j Chicago, UID: PPR, Members:[PPR,RB,NL], where:{city:Chicago, State: IL}}
Document DB
MongoDB
CouchDB
Name
UID
Members
Groups
Photo
0x235C
Philip
PPR
CHI, SFO, BOS
B75DD108A893A
0xCD21
Neo4j Chicago
CHI
PPR,RB, NL
218758D88E901
Column Family
HBase
Cassandra
0x235C
Philip
0xCD21
Neo4j Chicago
0x2014
[PPR,RB,NL]
0x3821
[CHI, SFO, BOS]
0x3890
B75DD108A
Kev-Value
membase
Redis
Riak

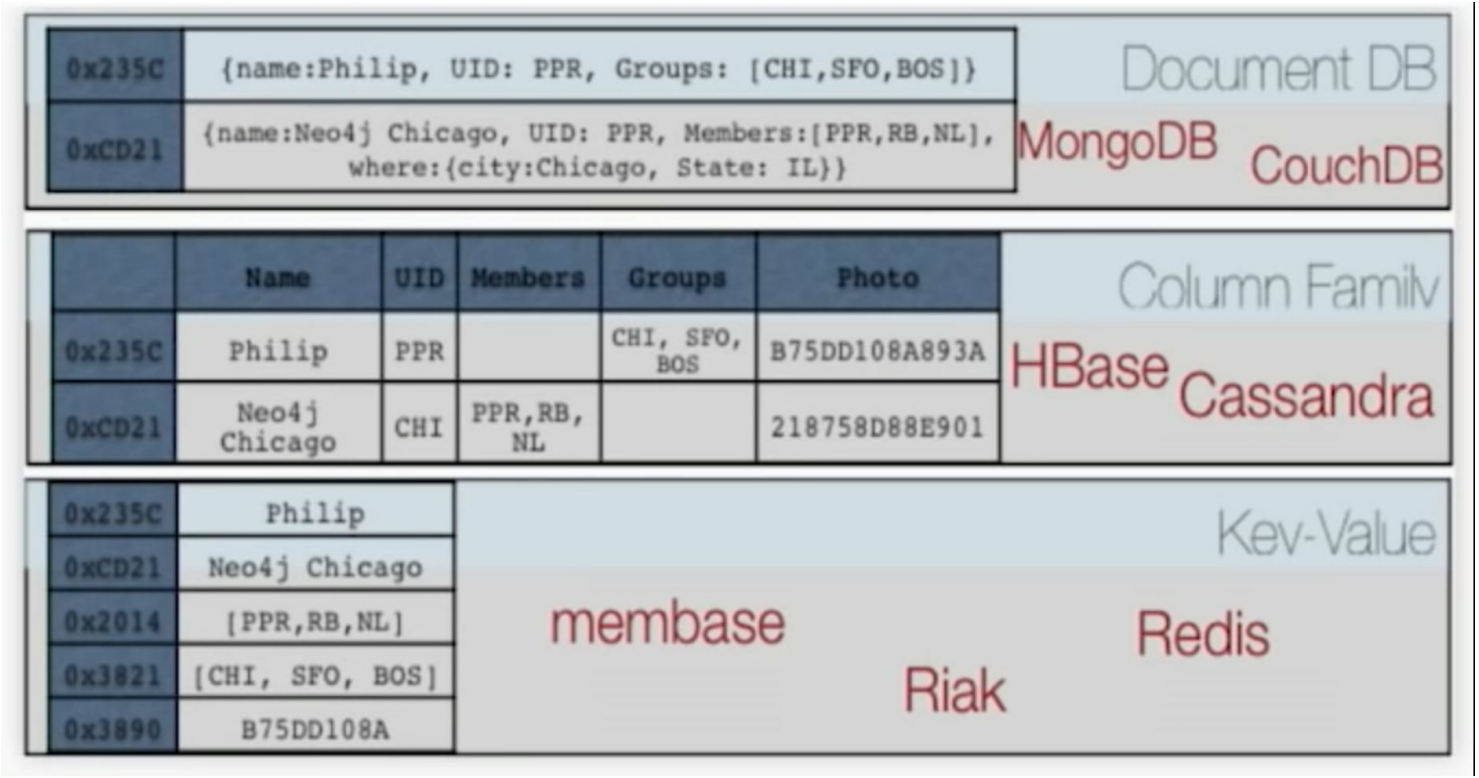
SQL / *Red*



NoSQL / ***Not****Red*



NoSQL *(some families)*



**don't even get me started on RDF*

Are NoSQL DBs good with relationships?

Are NoSQL DBs **good with relationships?**

- Can’t do **complex models**
- Can’t **scale joins** efficiently
- “joins” are not easy to **query**
- Not **ACID**

  (“eventually consistent” == “eventually corrupt”)

Are NoSQL DBs **good with relationships?**

- Can’t do **complex models**
- Can’t **scale joins** efficiently
- “joins” are not easy to **query**
- Not **ACID**

  (“eventually consistent” == “eventually corrupt”)

Are NoSQL DBs **good with relationships?**

- Can’t do **complex models**
- Can’t **scale joins** efficiently
- “joins” are not easy to **query**
- Not **ACID**

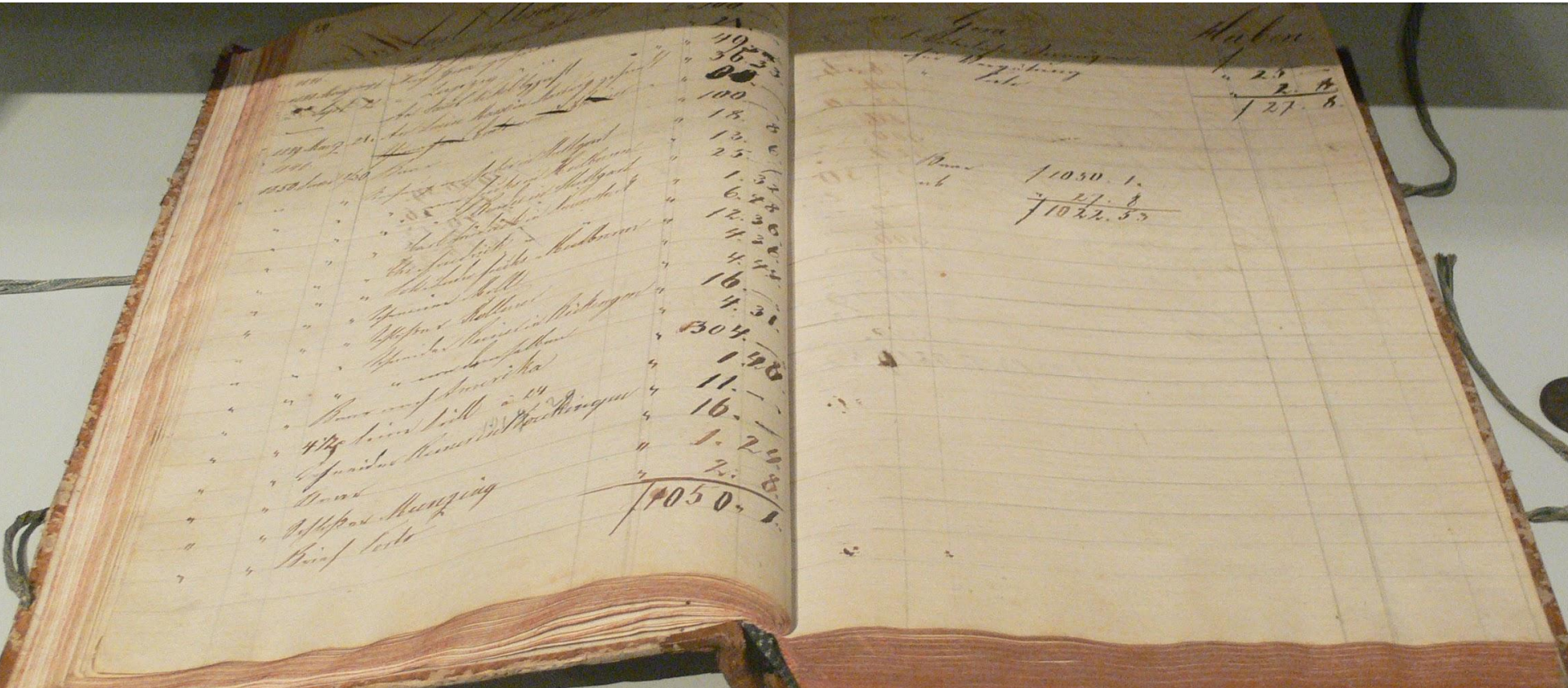
  (“eventually consistent” == “eventually corrupt”)

Are NoSQL DBs **good with relationships?**

- Can’t do **complex models**
- Can’t **scale joins** efficiently
- “joins” are not easy to **query**
- Not **ACID**

  (“eventually consistent” == “eventually corrupt”)

What Graph Databases do **NOT** do well



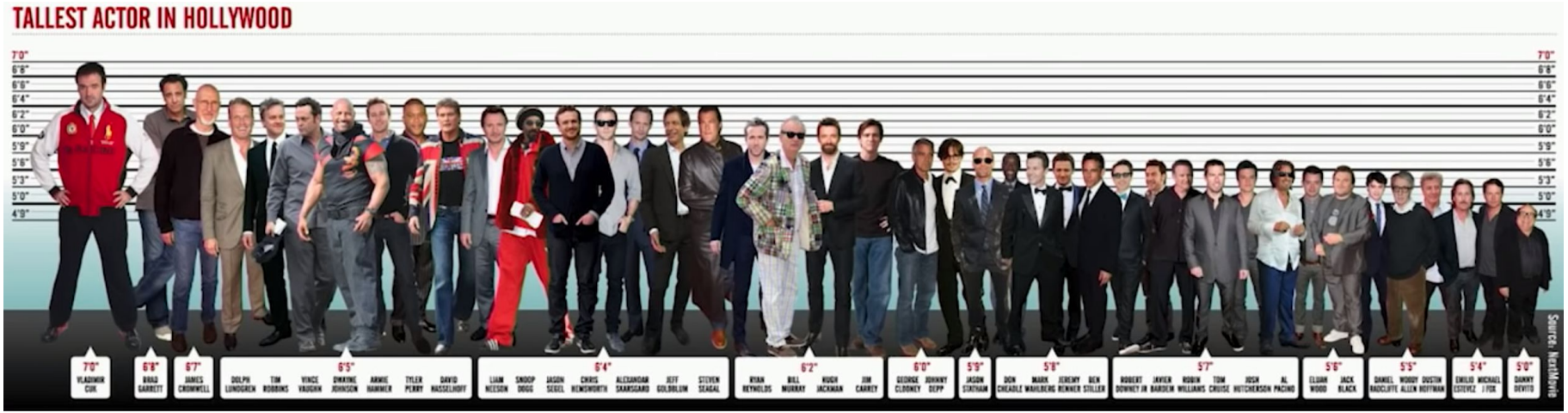
Accounting? Averages? … **No!**

	A	B	C
1			
2			
3			
4			
5			
6			
7			
8			

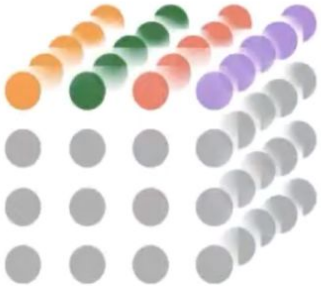
Use postgres. Or a spreadsheet.

Graphs are the incorrect tool.

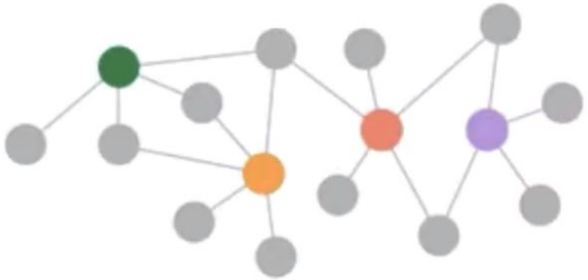
Operations on properties over lots of records.

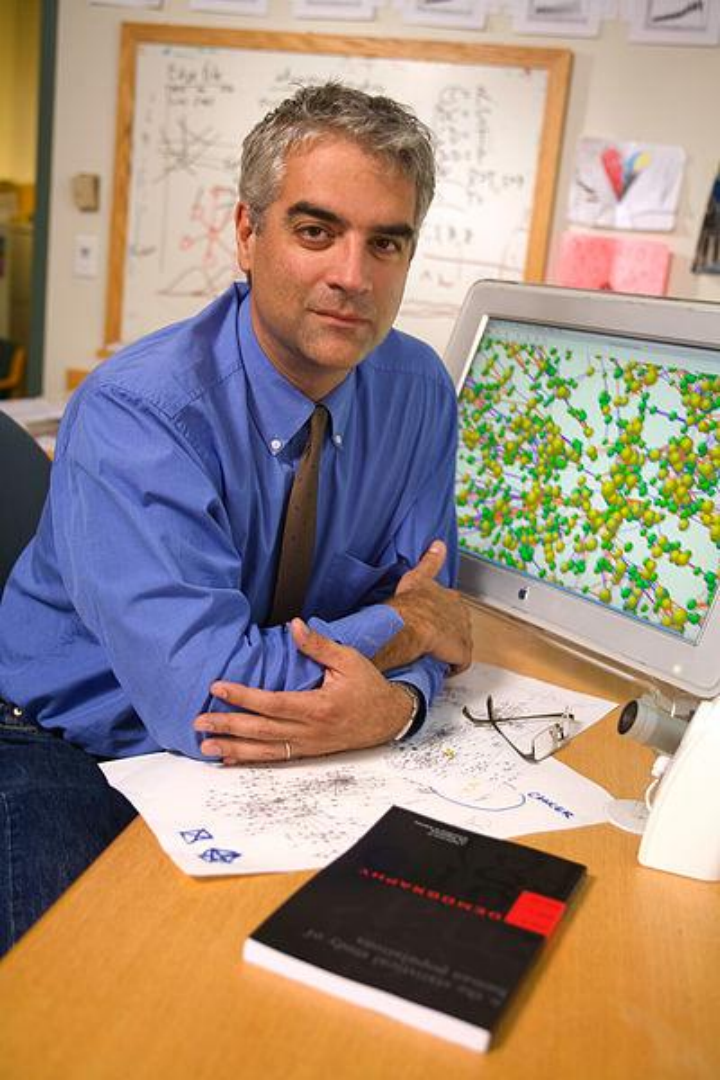


IT'S NOT WHAT YOU KNOW...



IT'S WHO YOU KNOW.





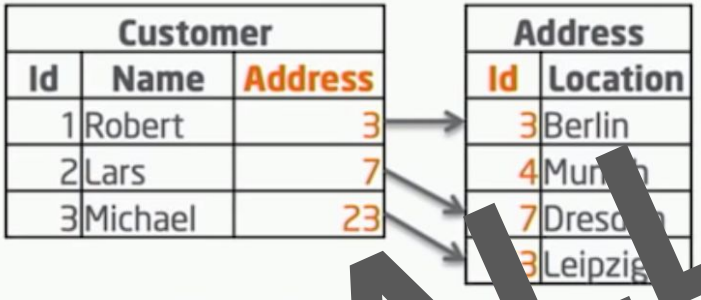
Nicholas A. Christakis is an American sociologist and physician known for his research on social networks.

h-index: 102

You get a **report** …

- w
- k in
- w …
- who are **interested** in **books** ...
- who's **friends** …
  - … have **f s** …
  - … who are om **S**
  - who are a **int**

SQL Join Hell (1)



Customer		
Id	Name	Address
1	Robert	3
2	Lars	7
3	Michael	23

Address	
Id	Location
3	Berlin
4	Mun
7	Dresd
3	Leipzig

1:1 Relationship

Id	Name
1	bert
2	Lars
3	Michael

Address		
Id	Customer	Location
3	1	Berlin
7	2	Dresden
8	2	New York
	3	Leipzig

1:n Relationship

Customer	
Id	Name
1	Robert
2	Lars
3	Michael

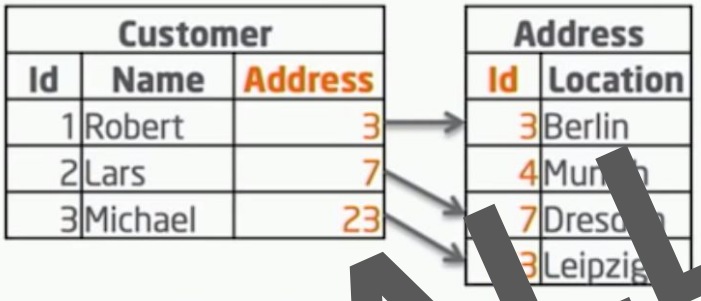
Id	
1	3
2	7
2	8
3	23

Address	
Id	Location
3	Berlin
7	Dresden
8	New York
23	Leipzig

m:n Relationship

ALL THE INDEXS

SQL Join Hell (1)



Customer		
Id	Name	Address
1	Robert	3
2	Lars	7
3	Michael	23

Address	
Id	Location
3	Berlin
4	Mun
7	Dresd
3	Leipzig

1:1 Relationship

Id	Name
1	bert
2	Lars
3	Michael

Address		
Id	Customer	Location
3	1	Berlin
7	2	Dresden
8	2	New York
23	3	Leipzig

1:n Relationship

Customer	
Id	Name
1	Robert
2	Lars
3	Michael

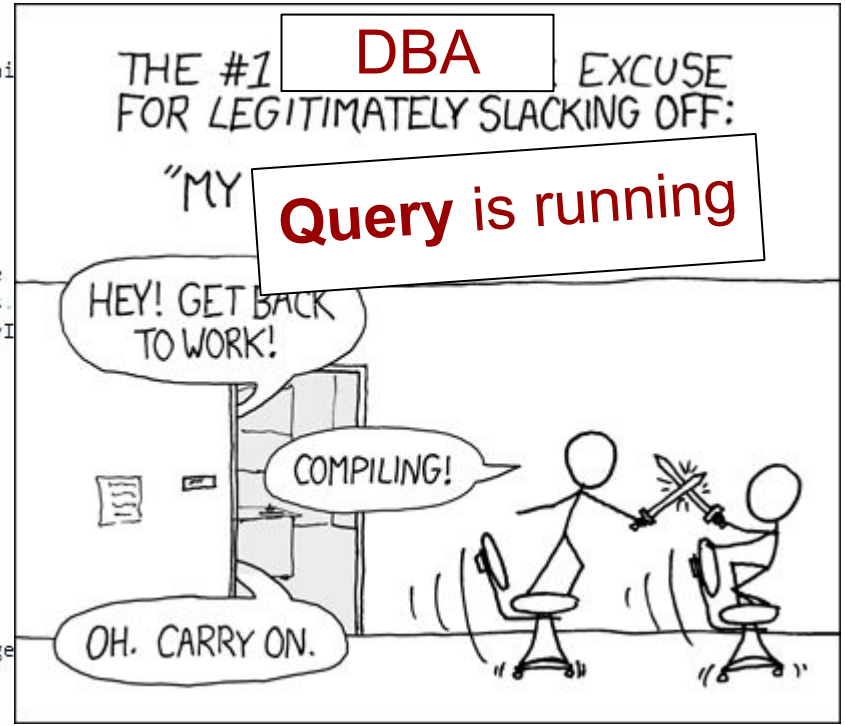
Cu	Ad
1	3
2	7
2	8
3	23

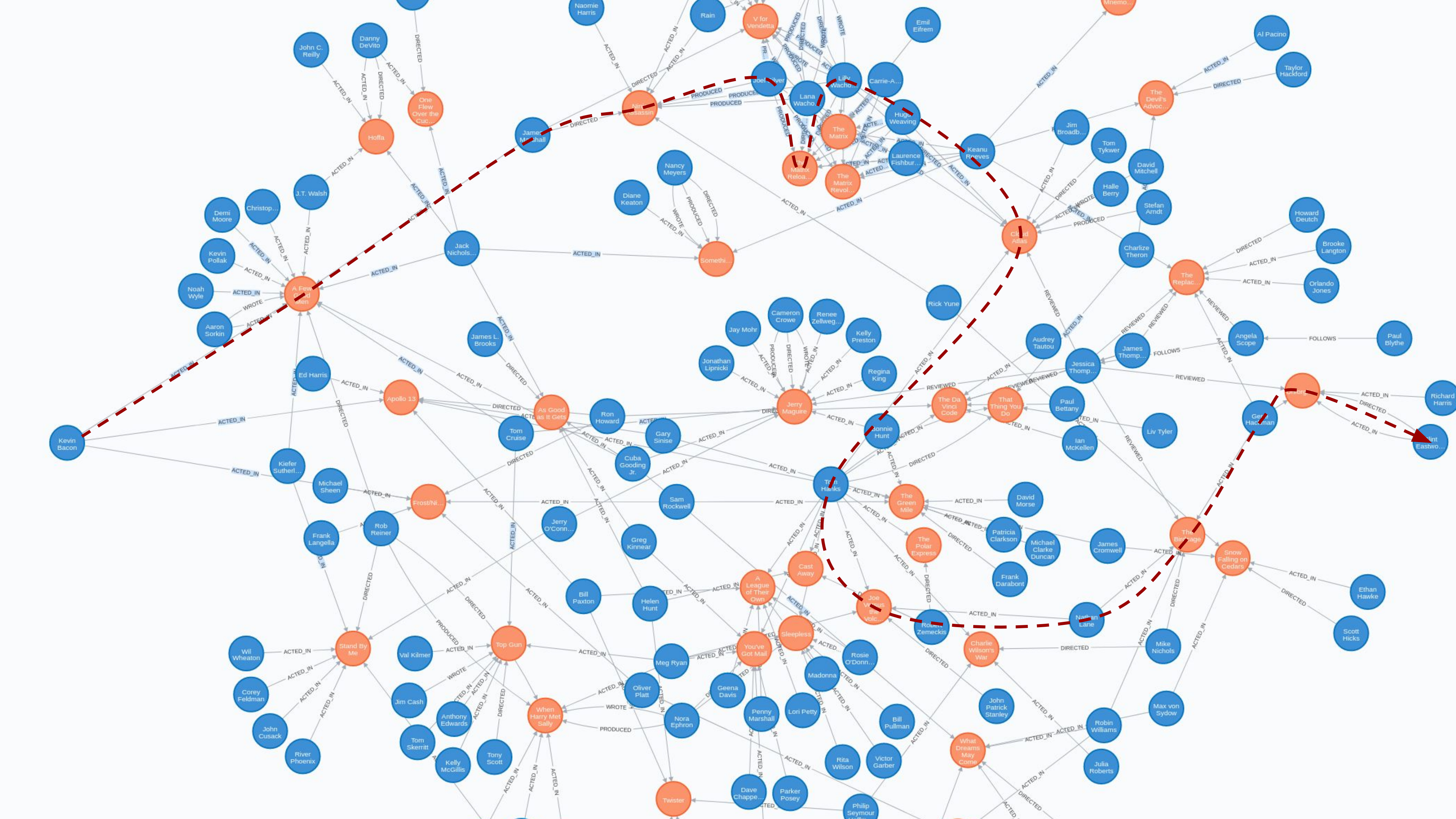
Address	
Id	Location
3	Berlin
7	Dresden
8	New York
23	Leipzig

m:n Relationship

ALL THE INDICES

```
SELECT p.Name AS ProductName,
NonDiscountSales = (OrderQty * UnitPrice),
Discounts = ((OrderQty * UnitPrice) * UnitPriceDiscount)
FROM Production.Product AS p
INNER JOIN Sales.SalesOrderDetail AS sod
ON p.ProductID = sod.ProductID
ORDER BY ProductName DESC;

SELECT 'Total income is', ((OrderQty * Uni
p.Name AS ProductName
FROM Production.Product AS p
INNER JOIN Sales.SalesOrderDetail AS sod
ON p.ProductID = sod.ProductID
ORDER BY ProductName ASC;

SELECT DISTINCT pp.LastName, pp.FirstName
FROM Person.Person pp JOIN HumanResources.
ON e.BusinessEntityID = pp.BusinessEntityI
(SELECT SalesPersonID
FROM Sales.SalesOrderHeader
WHERE SalesOrderID IN
(SELECT SalesOrderID
FROM Sales.SalesOrderDetail
WHERE ProductID IN
(SELECT ProductID
FROM Production.Product p
WHERE ProductNumber = 'BK-M68B-42')));

SELECT ProductID, AVG(OrderQty) AS Average
FROM Sales.SalesOrderDetail
GROUP BY ProductID
HAVING SUM(LineTotal) > $1000000.00
AND AVG(OrderQty) < 3;

SELECT pp.FirstName, pp.LastName, e.NationalIDNumber
FROM HumanResources.Employee AS e WITH (INDEX(AK_Employee_NationalIDNumber))
JOIN Person.Person AS pp on e.BusinessEntityID = pp.BusinessEntityID
WHERE LastName = 'Johnson';
```



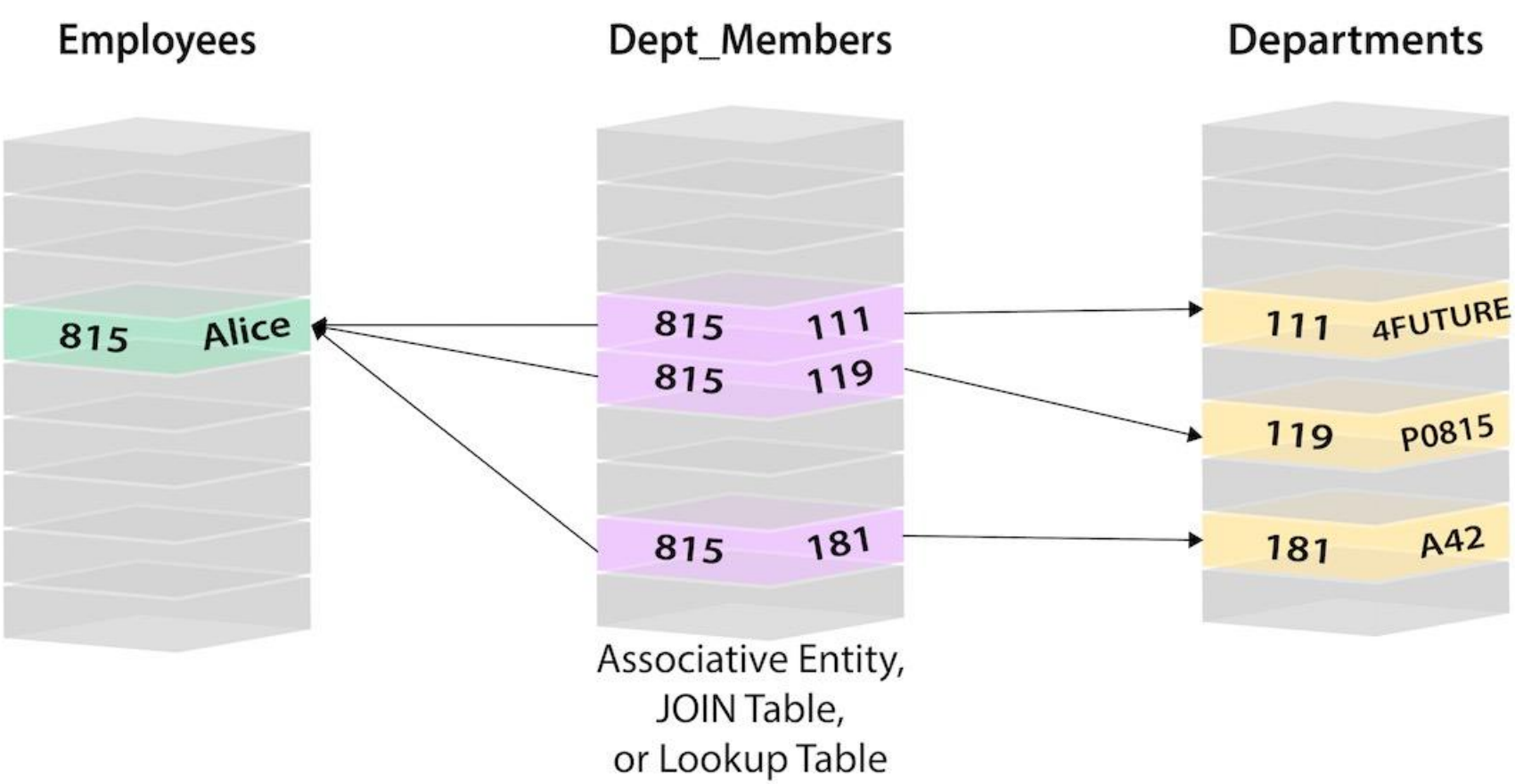

Naomie Harris
Rain
V for Vendetta
Emil Eifrem
Al Pacino
Taylor Hackford
John C. Reilly
Danny DeVito
One Flew Over the Cuc...
Hoffa
Ninja Assassin
Lana Wacho...
Carrie-A...
Hugo Weaving
The Matrix
The Matrix Reloa...
The Matrix Revol...
Laurence Fishbur...
Keanu Reeves
The Devil's Advoc...
Jim Broadb...
Tom Tykwer
David Mitchell
Halle Berry
Stefan Arndt
Cloud Atlas
Nancy Meyers
Diane Keaton
Somethi...
J.T. Walsh
Demi Moore
Christop...
Kevin Pollak
Noah Wyle
Aaron Sorkin
A Few Good Men
Jack Nichols...
Howard Deutch
Brooke Langton
Orlando Jones
Charlize Theron
The Replac...
Rick Yune
Cameron Crowe
Renee Zellweg...
Jay Mohr
Kelly Preston
Jonathan Lipnicki
Regina King
James L. Brooks
Audrey Tautou
Jessica Thomp...
James Thomp...
Angela Scope
Paul Blythe
Ed Harris
Apollo 13
As Good as It Gets
Ron Howard
Jerry Maguire
The Da Vinci Code
That Thing You Do
Paul Bettany
Richard Harris
Kevin Bacon
Tom Cruise
Gary Sinise
Bonnie Hunt
Ian McKellen
Liv Tyler
Gene Hackman
Kiefer Sutherl...
Michael Sheen
Cuba Gooding Jr.
Tom Hanks
Sam Rockwell
The Green Mile
David Morse
Frost/Ni...
Frank Langella
Rob Reiner
Jerry O'Conn...
Greg Kinnear
Patricia Clarkson
Michael Clarke Duncan
James Cromwell
The Polar Express
Frank Darabont
Cast Away
A League of Their Own
Snow Falling on Cedars
Ethan Hawke
Scott Hicks
Bill Paxton
Helen Hunt
Robert Zemeckis
Nathan Lane
Wil Wheaton
Stand By Me
Val Kilmer
Top Gun
Meg Ryan
You've Got Mail
Sleepless...
Rosie O'Donn...
Charlie Wilson's War
Mike Nichols
Corey Feldman
Jim Cash
Oliver Platt
Geena Davis
Madonna
John Cusack
Anthony Edwards
When Harry Met Sally
Nora Ephron
Penny Marshall
Lori Petty
Bill Pullman
John Patrick Stanley
Robin Williams
Max von Sydow
River Phoenix
Tom Skerritt
Kelly McGillis
Tony Scott
Rita Wilson
Victor Garber
What Dreams May Come
Julia Roberts
Dave Chappe...
Parker Posey
Twister
Philip Seymour

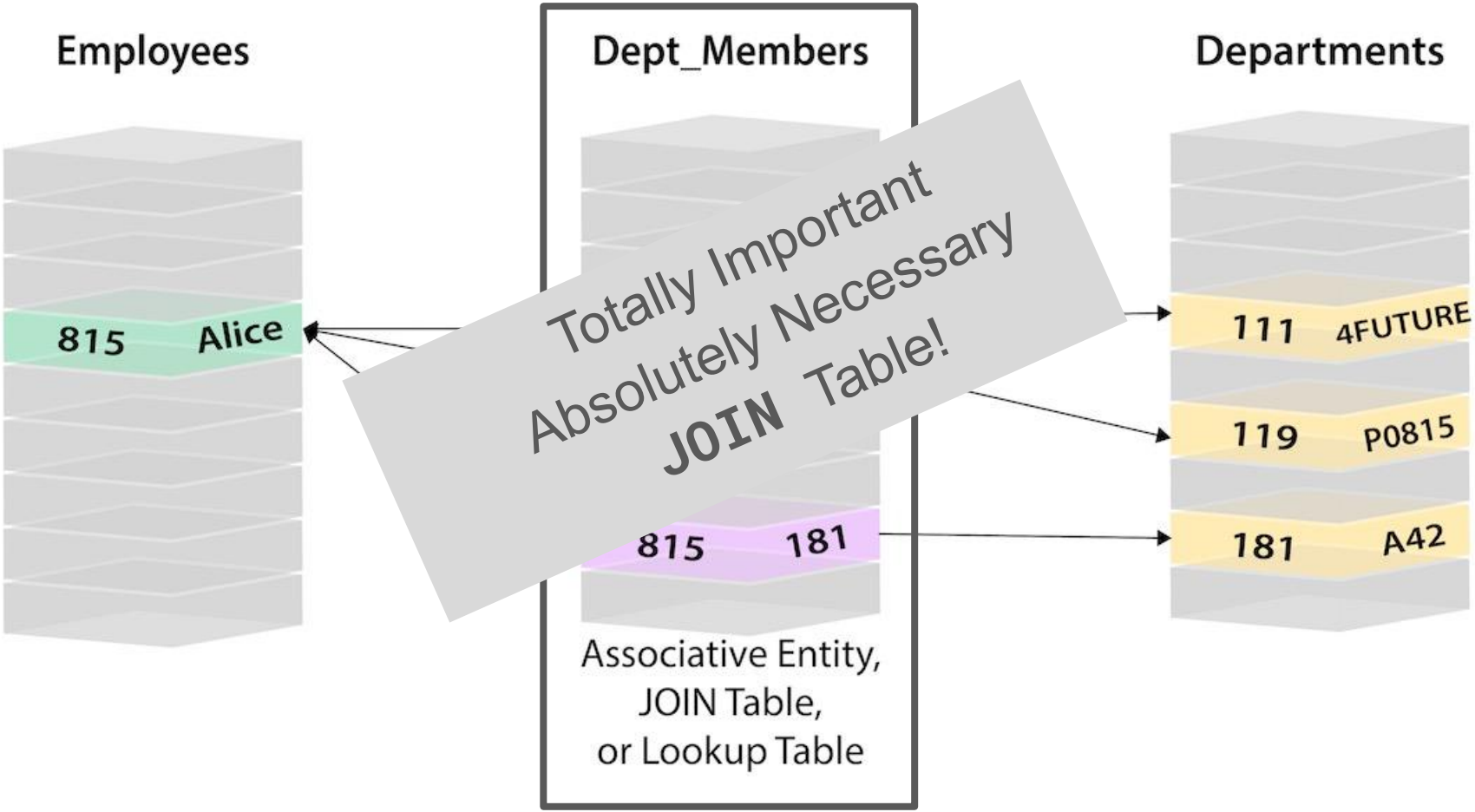
“ … literally **1000s** of times ***faster*** than our prior MySQL solution, with queries that require **10** to **100** times ***less code***, providing functionality that was previous ***impossible***.”

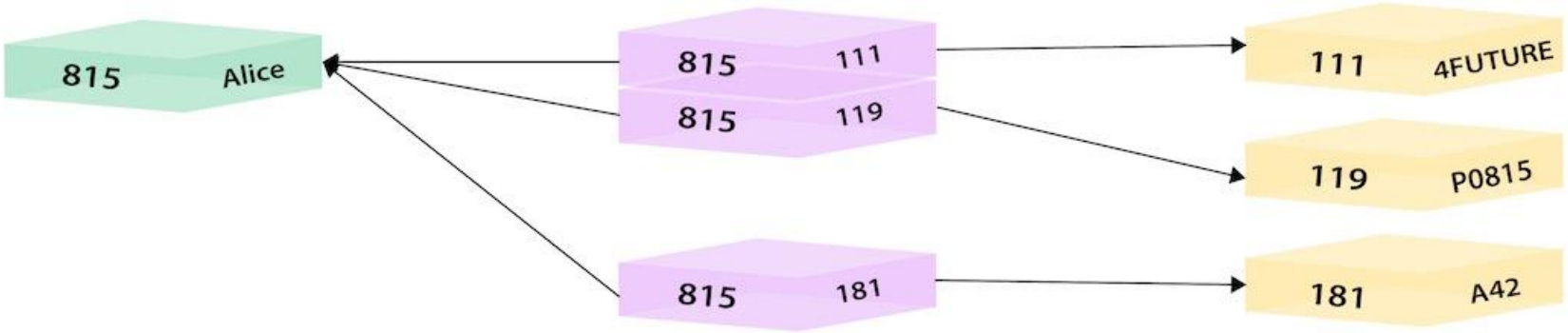
Volker Pacher
Senior Engineer

SQL Query

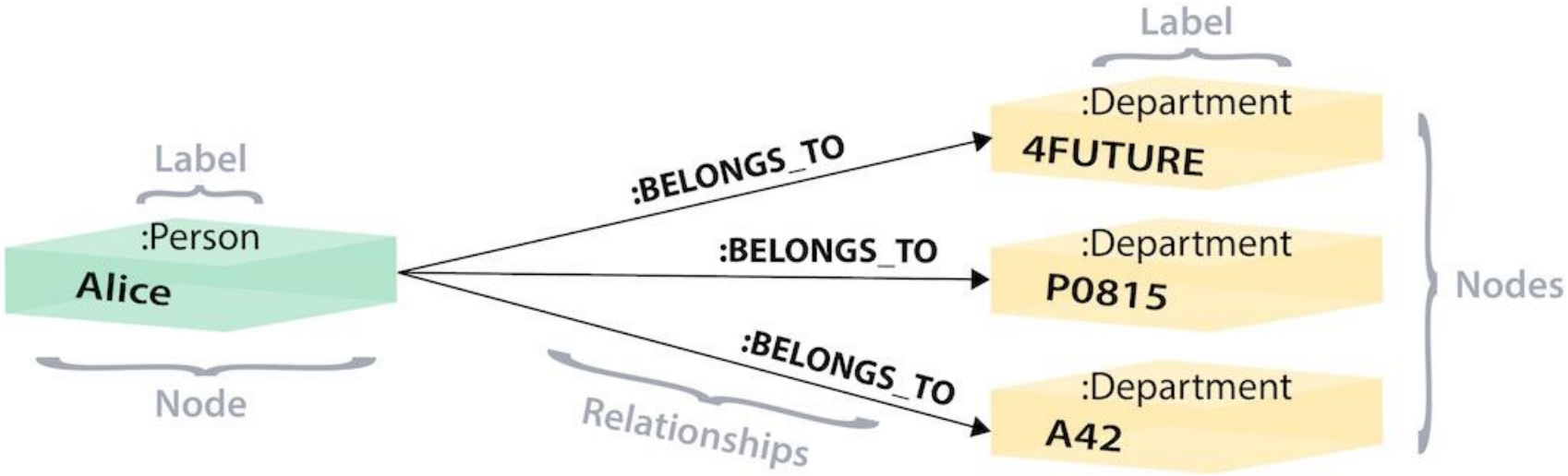
```
SELECT name FROM Person
LEFT JOIN Person_Department
  ON Person.Id = Person_Department.PersonId
LEFT JOIN Department
  ON Department.Id =
Person_Department.DepartmentId
WHERE Department.name = "IT Department"
```


Employees
Dept_Members
Departments
815
Alice
815
111
815
119
815
181
111
4FUTURE
119
P0815
181
A42
Associative Entity,
JOIN Table,
or Lookup Table


Employees
Dept_Members
Departments
815 Alice
111 4FUTURE
119 P0815
181 A42
815 181
Totally Important
Absolutely Necessary
JOIN Table!
Associative Entity,
JOIN Table,
or Lookup Table


815 Alice
815 111
815 119
815 181
111 4FUTURE
119 P0815
181 A42


815
Alice
815
815
181
111
4FUTURE
119
P0815
181
A42


Label
:Person
Alice
Node
:BELONGS_TO
:BELONGS_TO
:BELONGS_TO
Relationships
Label
:Department
4FUTURE
:Department
P0815
:Department
A42
Nodes

GQL (cypher) Query

```
MATCH
  (people:Person)
  -[:WORKS_AT]->
  (dept:Dept {name: "IT Department"})

RETURN people.name
```

GQL (cypher) Query

```
MATCH
  (people:Person)
  -[:WORKS_AT]->
  (dept:Dept)

WHERE dept.name = "IT Department"
RETURN people.name
```

SQL Query

```
SELECT name FROM Person
LEFT JOIN Person_Department
  ON Person.Id =
Person_Department.PersonId
LEFT JOIN Department
  ON Department.Id =
Person_Department.DepartmentId
WHERE Department.name =
"IT Department"
```

GQL (cypher) Query

```
MATCH
  (people:Person)
  -[:WORKS_AT]->
  (dept:Dept {name:"IT Department"})

RETURN people.name
```

Django Query

```
Person.objects.filter(department__name="IT Department")
```

Django Query

```
Person.objects.filter(department__name="IT Department").query
```

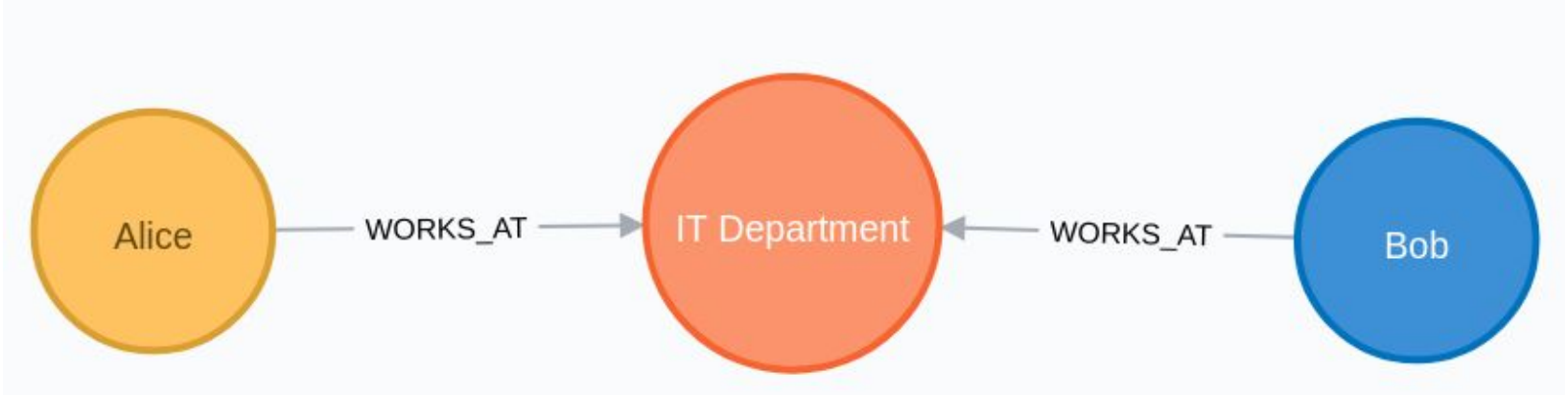
```
SELECT "staff_person"."id" FROM "staff_person"
INNER JOIN "staff_person_staff" ON ("staff_person"."id" =
"staff_person_staff"."person_id") INNER JOIN "staff_department" ON
("staff_person_staff"."department_id" = "staff_department"."id")
WHERE "staff_department"."name" = IT Department
```

GQL (cypher) Query

```
MATCH (people:Person)-[:WORKS_AT]->(dept:Department)
WHERE dept.name = "IT Department"
RETURN people.name
```

:Person {
`name`: **Alice**
`interests`: [ultra-marathons, LARPing]
`best_100km`: 18:47:19
`preferred_larp_system`: L5r
`l5r_main_char`: A-Bomb the Mighty
`l5r_character_type`: Seeker of Enlightenment
`l5r_main_skill_group`: Scholar Skills
`l5r_preferred_weapon`: Kusarigama
`l5r_preferred_clan`: Phoenix Clan

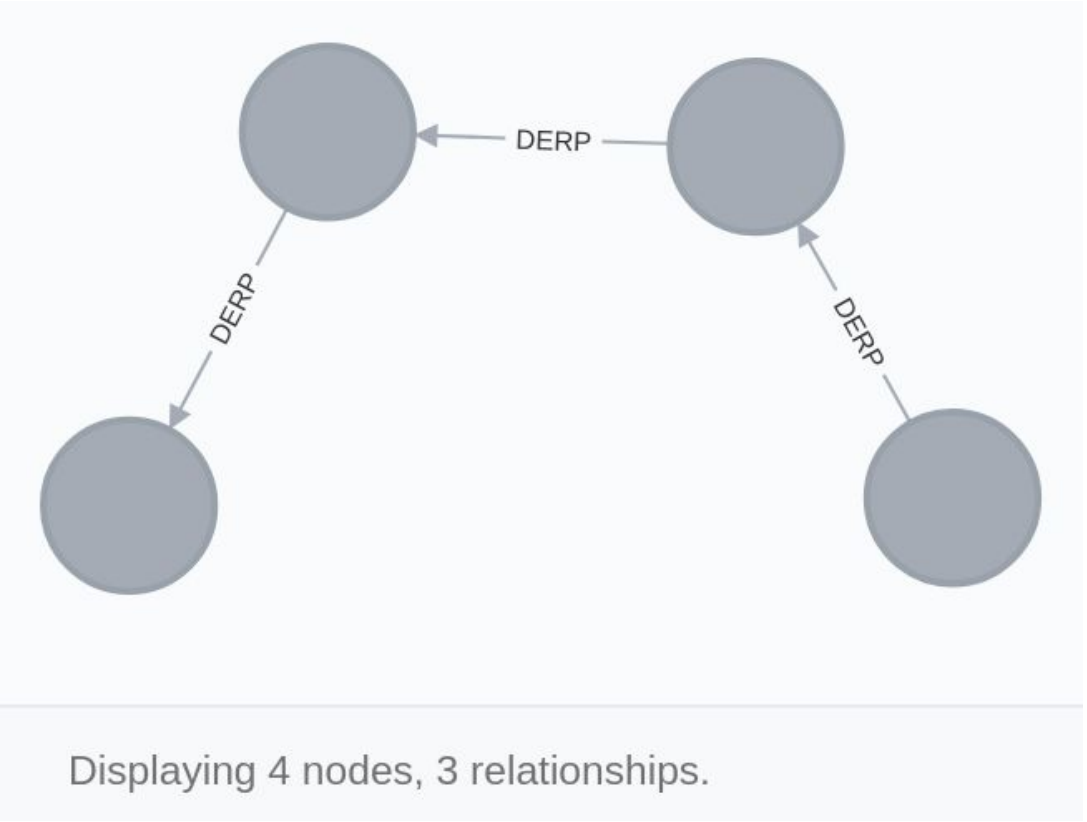
:Person {
`name`: **Bob**
`full_name`: Robert Perry Smith
`interests`: volleyball
}



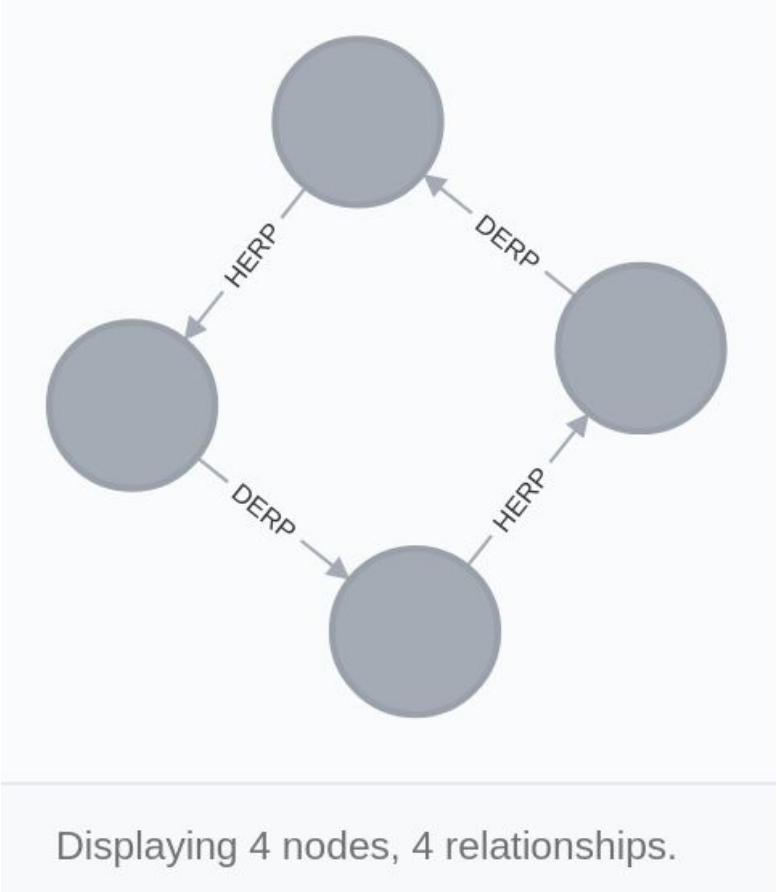
:Engineer:Runner:AllRoundLegend {
  name: **Alice**
  interests: [ultra-marathons, LARPing]
  best_100km: 18:47:19
  preferred_larp_system: L5r
  l5r_main_char: A-Bomb the Mighty
  l5r_character_type: Seeker of Enlightenment
  l5r_main_skill_group: Scholar Skills
  l5r_preferred_weapon: Kusarigama
  l5r_preferred_clan: Phoenix Clan

:Person {
  name: **Bob**
  full_name: Robert Perry Smith
  interests: volleyball
}

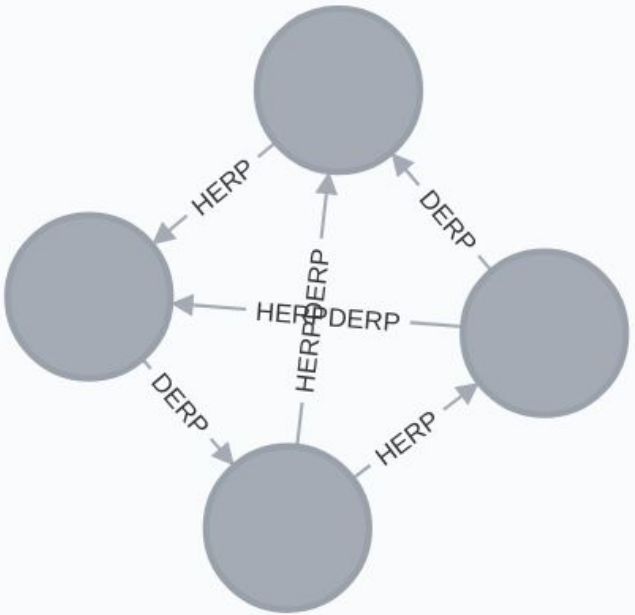
```
CREATE (i)
CREATE (j)
CREATE (k)
CREATE (l)

CREATE
(i)-[:DERP]->
(j)-[:DERP]->
(k)-[:DERP]->(l)

RETURN i, j, k, l
```



```
CREATE (i)
CREATE (j)
CREATE (k)
CREATE (l)

CREATE
(i)-[:HERP]->
(j)-[:DERP]->
(k)-[:HERP]->
(l)-[:DERP]->(i)

RETURN i, j, k, l
```

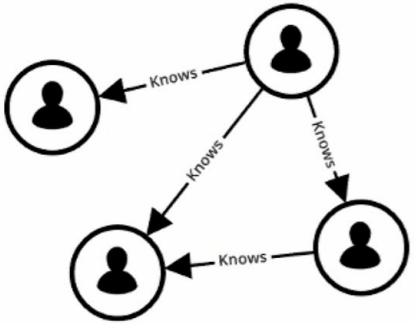


```
CREATE (i)
CREATE (j)
CREATE (k)
CREATE (l)

CREATE
(i)-[:HERP]->(j)-[:DERP]->
(k)-[:HERP]->(l)-[:DERP]->(i)
CREATE (i)-[:HERPDERP]->(k)
CREATE (j)-[:HERPDERP]->(l)

RETURN i, j, k, l
```



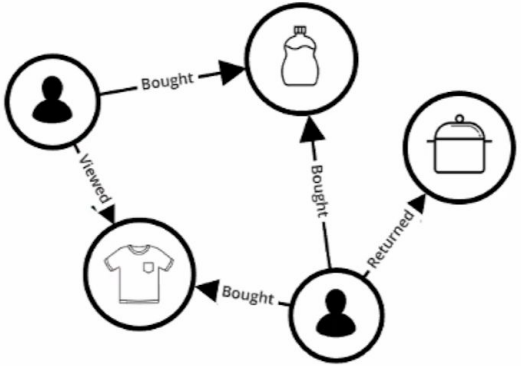
Displaying 4 nodes, 6 relationships.

What Graph Databases **DO**, er, do well



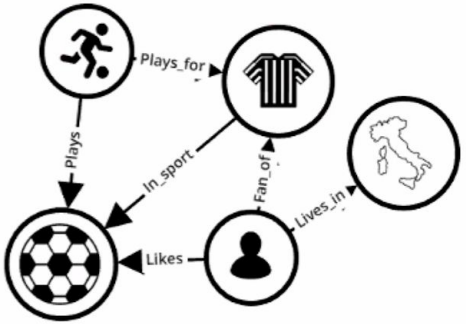
Networks of People

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



Business Processes

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



Knowledge Networks

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

What Graph Databases **DO**, er, do well

Financial Services

Drug Discovery

Recommendations

Customer Segmentation

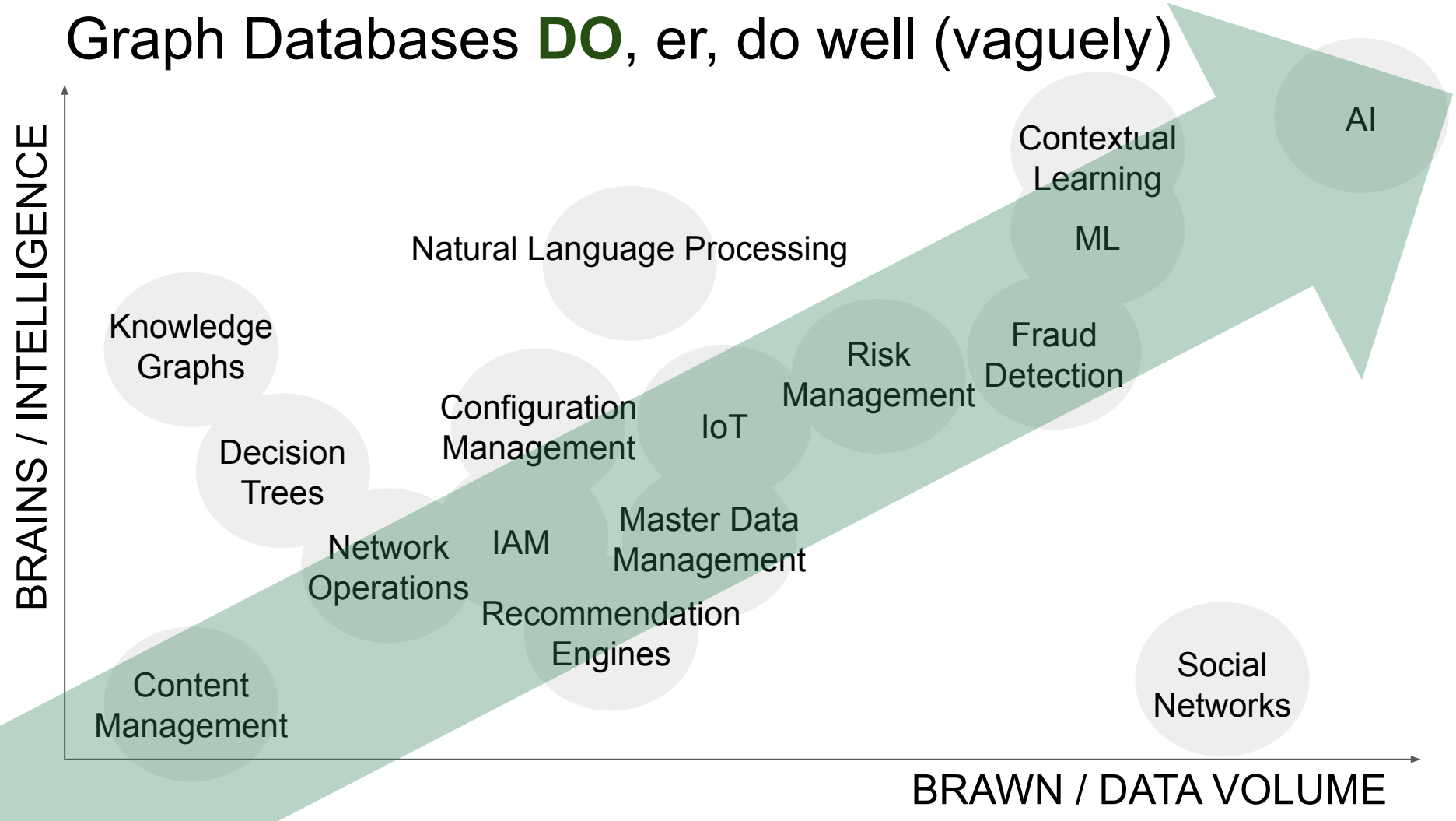
Cybersecurity

Churn Prediction

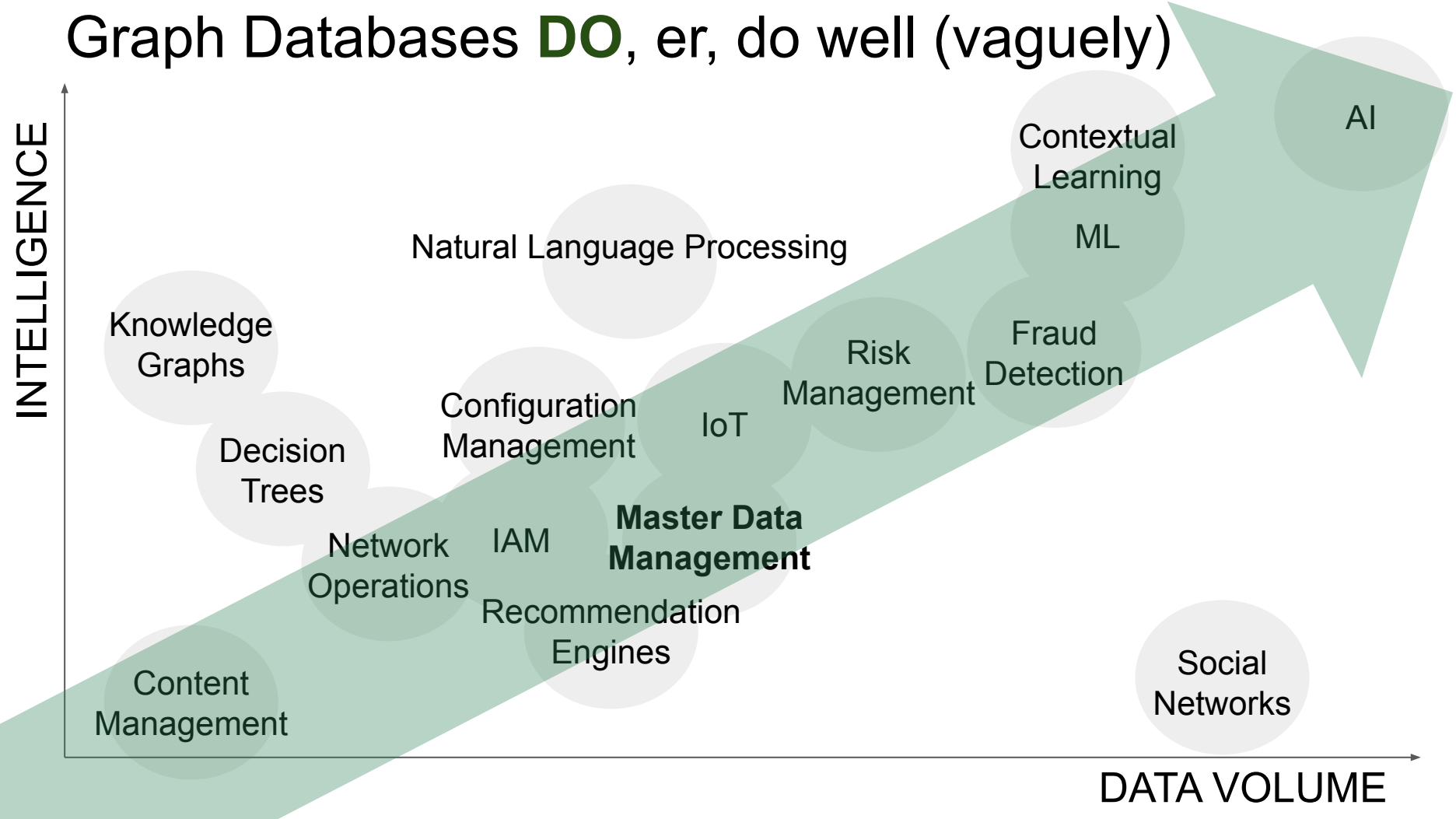
Search/MDM

Predictive Maintenance

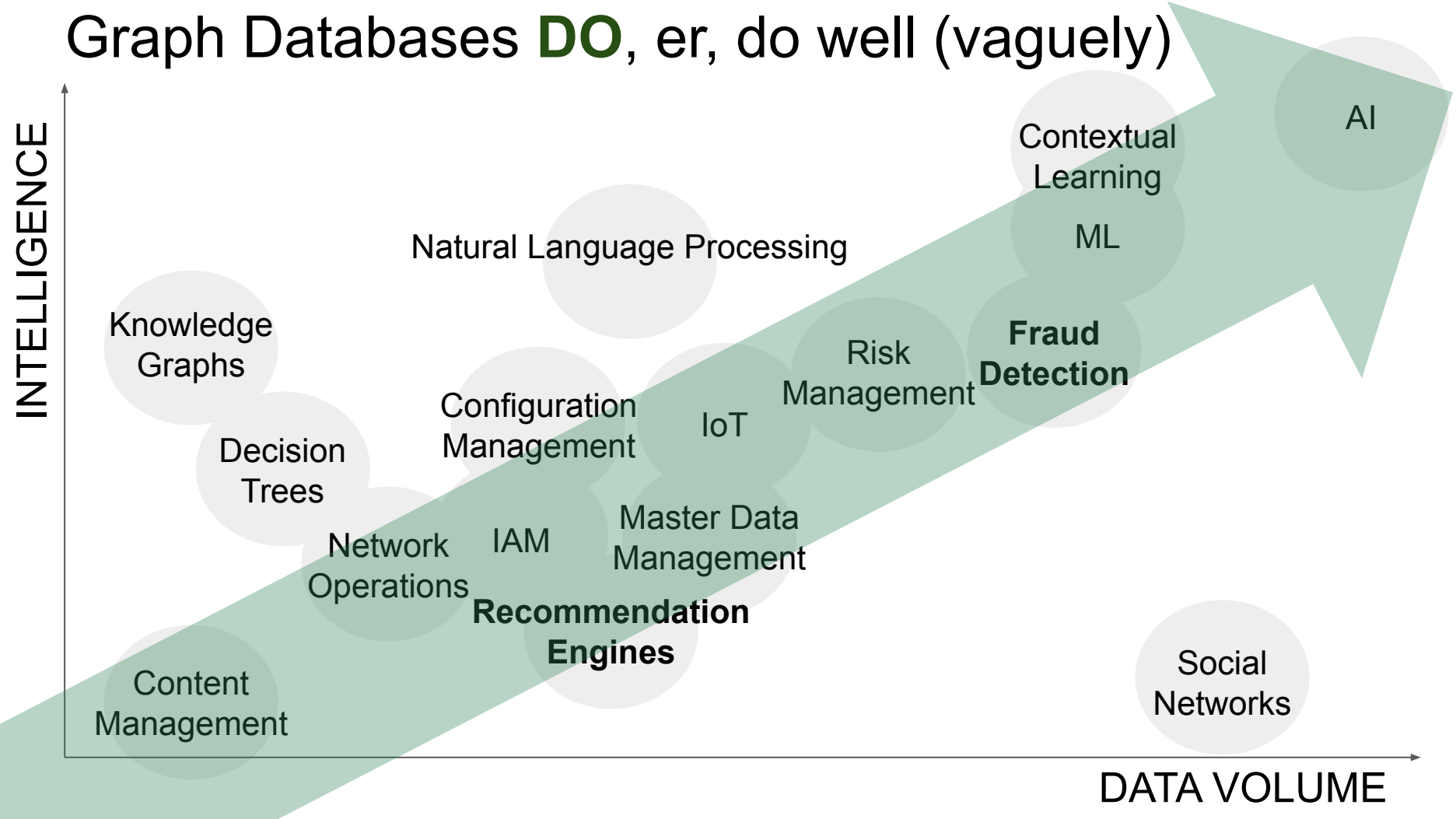
Graph Databases **DO**, er, do well (vaguely)



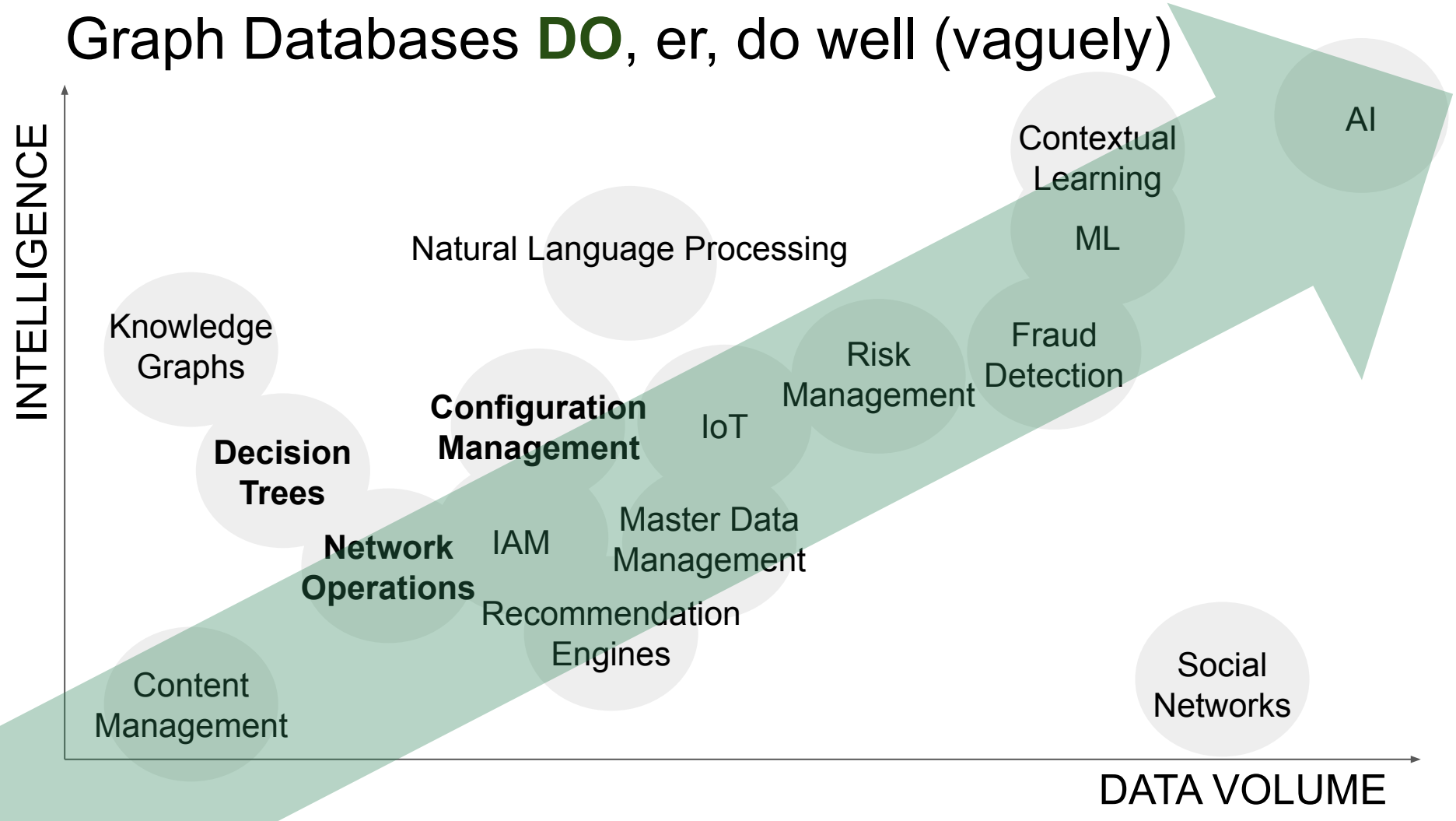
Graph Databases **DO**, er, do well (vaguely)



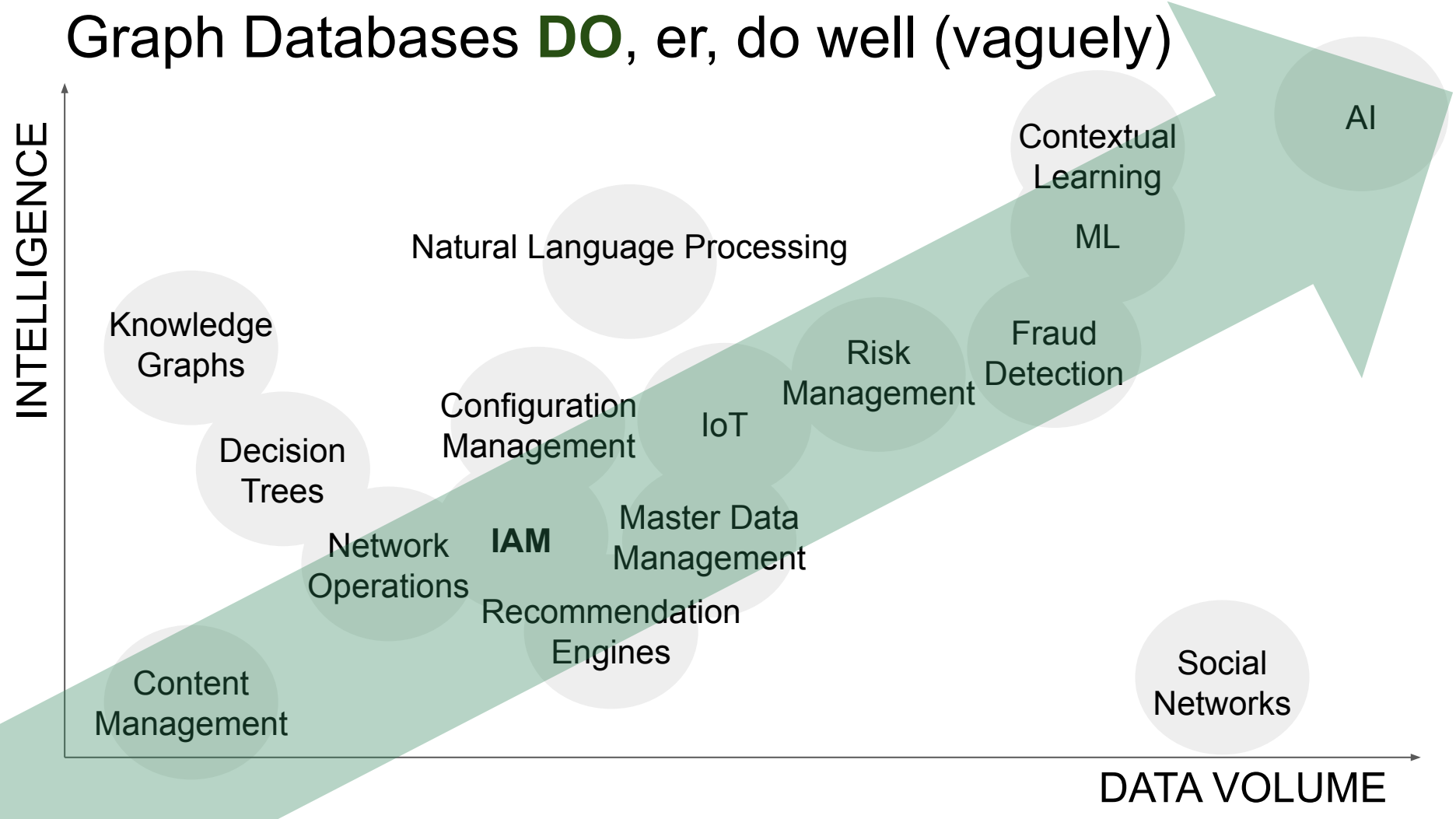
Graph Databases **DO**, er, do well (vaguely)



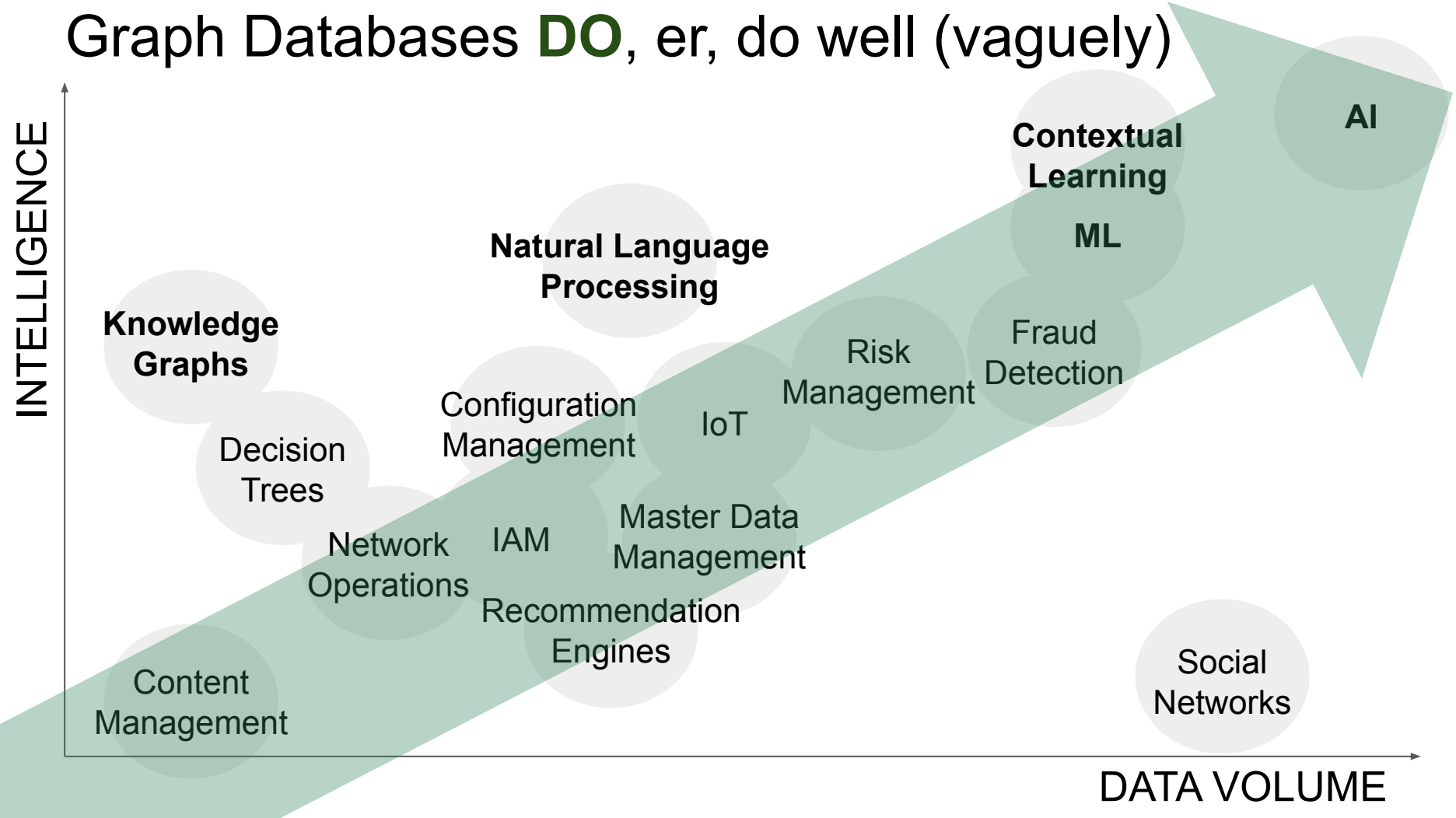
Graph Databases **DO**, er, do well (vaguely)



Graph Databases **DO**, er, do well (vaguely)



Graph Databases **DO**, er, do well (vaguely)



Open Source Graph DBs

Labelled Property Graph Databases

In active development.

There are many proprietary ones also.
There are new Graph DBs being created.

vs. Persistent data

Great. Enough talking about code ...

DEMO

Great. Enough talking about code ...

Let’s make a ...
Recommendations Engine

http://localhost:**7474/browser/**

alt-tab dawg ...

Plan of Attack:

1. **Connect** to our DB
2. **Load** in our presentation data
3. **Query**
4. make **Flask** app

alt-tab dawg ...

Recommendations Engine recap

Many Algorithms:

- Shared Identifiers (**Connected Components**)
- Influence/Volumes (**Page Rank**)
- Community Interactions (say 6 hops) (**Louvain**)
- Known Troublemaker (**Jaccard**)

Recommendations Engine recap

Many Algorithms:

- Shared Identifiers (**Connected Components**)
- Influence/Volumes (**Page Rank**)
- Community Interactions (say 6 hops) (**Louvain**)
- Known Troublemaker (**Jaccard**)

Recommendations Engine recap

Many Algorithms:

- Shared Identifiers (**Connected Components**)
- Influence/Volumes (**Page Rank**)
- Community Interactions (say 6 hops) (**Louvain**)
- Known Troublemaker (**Jaccard**)

Fraud Detection

Many Algorithms:

- Shared identifiers (**Connected Components**)
- Influence/Volumes (**Page Rank**)
- Community Interactions (say 6 hops) (**Louvain**)
- Known Troublemaker (**Jaccard**)

Fraud Detection

Many Algorithms:

- Shared identifiers (**Connected Components**)
- Influence/Volumes (**Page Rank**)
- Community Interactions (say 6 hops) (**Louvain**)
- Known Troublemaker (**Jaccard**)

Recommendations Engine

Many Algorithms:

- Shared identifiers (**Connected Components**)
- Influence/Volumes (**Page Rank**)
- Community Interactions (say 6 hops) (**Louvain**)
- Known Influencer (**Jaccard**)

So many more ...

DEMOS

“graphgists”

Please:
Write more introductory materials!
Document your progress!

Share and enjoy Graph DB fun.

github.com
/**elena**
/**graph-fun**

Join us on ***Slack***
links at:
github.com
/**canberra-python**

Actual "**Paradise Papers**" database
created by **ICIJ**
(International Consortium of Investigative Journalists)

Hi, I’m Elena. I am a web developer and I love python.
github.com/**elena/graph-fun**
twitter.com/**elequ**

Proudly help organise
Canberra Python User Group

Join us on **Slack**
*github.com/**canberra-python***

10 - 12 September 2021
https://**2021**.pycon.org.**au**/
Please ***Volunteer***

Hi,

I’m Elena.

github.com/**elena**

twitter.com/**elequ**

Proudly help organise

Canberra Python User Group

with: Jonah Sullivan, Mike Leonard, Zac Hatfield-Dodds

Check us out on **Meetup**

Join our Online Hacktoberfest

tl&dr;

Graph Databases are vastly more efficient ($O(n)$ v. $O(n\ log(n))$) than other database architectures for getting relations in **massive datasets** where you'd need to do **query-time index lookups** through **many joins**.

As a trade-off: they are less efficient at aggregation.

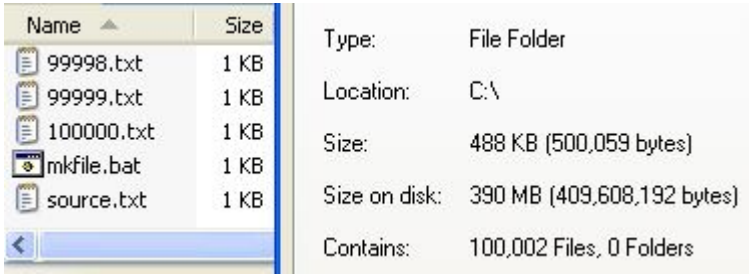
This useful for application in: financial systems, telecommunication networks, logistics and distribution, retail and data science generally.

SQL statement

```
SELECT name FROM Person
LEFT JOIN Address
  ON Person.Id = Person_Address.PersonId
  WHERE Address.city = 'canberra' COLLATE
SQL_Latin1_General_CP1_CI_A
LEFT JOIN Person_Sale
  ON Person.Id = Person_Basket.PersonId
LEFT JOIN Basket
  ON Basket.Id = Person_Basket.BasketId
LEFT JOIN Item
  ON Item.Id = Item_Basket.ItemId
  WHERE Item.name = "Widget"
LEFT JOIN PersonInterest
  ON Person.Id = Person_Interest.PersonId
  WHERE Interest.name = "books" COLLATE
SQL_Latin1_General_CP1_CI_A
LEFT JOIN Friend
  ON Person.Id = Person_Friend.PersonId
LEFT JOIN Address
  ON PersonFriend.Id = Person_Address.PersonFriendId
LEFT JOIN Address
  ON Person.Id = Person_Address.PersonId
  WHERE Address.city = "sydney" COLLATE
SQL_Latin1_General_CP1_CI_A
```

GQL (cypher) statement

```
MATCH (p:Person) WHERE p.city = "(?i)canberra"
MATCH (f:Person) WHERE f.city = "(?i)sydney"
MATCH (f)-[:LIKES]-(:Book)
MATCH (p)-[:LIKES]-(:Book)
WITH f, p
MATCH (f)-[:FRIENDS]-(p) WITH p
MATCH (p)-[:BOUGHT]-(s:Sale)-[]-(:Widget)
WHERE s.date = $yesterday, s.promo_code = $code
RETURN p
```




0 | 1