

Graph Databases

What kind of data?



facebook for developers | Products | Docs | Tools & Support | News | Case Studies | Search | Log In

All Docs | Docs / Sharing / Open Graph Stories / On This Page

Open Graph Stories

Post rich, structured stories from your app using a strongly typed API.

- Open Graph Stories in iOS** Publish Open Graph stories from your iOS app.
- Open Graph Stories in Android** Publish Open Graph stories from your Android app.
- Open Graph Stories on Web** Publish Open Graph stories from your website or web app.

How It Works

People use stories to share the things they're doing, the people they're doing them with and the places where they happen. Let people share stories about your app on Facebook through a structured, strongly typed API.

To publish Open Graph stories with the Share dialog, you do not need to implement Facebook Login or ask for additional permissions. For more information, see [Share Dialog](#).

If you create a custom sharing UI to publishing Open Graph stories, you need to implement [Facebook Login](#) and request the `publish_actions` permission from people using your app. This also means you need to submit your app for review, see [Login Review](#). Both of these steps help make sure people using your app are aware of the action they are taking by sharing and helps provide a high-quality experience.

If you create custom Open Graph actions and objects for your app, you should also submit your app for review to ensure your new objects, actions and capabilities associated with actions comply with Facebook policies. See [App Review](#).

Open Graph stories have four basic elements:

- Actor** - The person who posts the story
- App** - Every story includes attribution to the app that created it

Give Feedback | English (US)

What kind of data?

The screenshot shows the 'Google Inside Search' interface. At the top, the Google logo is followed by 'Inside Search'. A navigation bar includes links for Home, How Search Works, Tips & Tricks, Features, Search Stories, Playground, Blog, and Help. The main content area features a dark background with a network of nodes and lines representing the Knowledge Graph. A large blue circle highlights a portrait of Leonardo da Vinci. To the right, a search result for 'Leonardo da Vinci' is displayed, showing a portrait and a list of related images: 'Ginevra de' Benci' (1478), 'The Virgin' (1508), and 'Adoration of the M...' (1481). Below the images, a 'Feedback' link is visible. The search result text for Leonardo da Vinci includes: 'Leonardo di ser Piero da Vinci was an Italian Renaissance polymath: painter, sculptor, architect, musician, scientist, mathematician, engineer, inventor, anatomist, geologist, cartographer, botanist, and writer. Wikipedia'. Other details listed are: 'Born: April 15, 1452, Anchiano', 'Died: May 2, 1519, Clos Lucé', 'Buried: Château d'Amboise', 'Parents: Caterina da Vinci, Piero da Vinci', and 'Structures: Vebjem Sand Da Vinci Project'. Two call-to-action buttons are present: a blue one with a white arrow pointing right, and a grey one with a white arrow pointing right.

Google Inside Search

Home How Search Works Tips & Tricks **Features** Search Stories Playground Blog Help


The Knowledge Graph

Learn more about one of the key breakthroughs behind the future of search.

See it in action

Discover answers to questions you never thought to ask, and explore collections and lists.


What kind of data?

Google 

Tudo [Imagens](#) [Notícias](#) [Vídeos](#) [Mapas](#) [Mais](#) [Ferramentas de pesquisa](#)

Cerca de 87 800 000 resultados (0,48 segundos)

Nas notícias

 **Cristiano Ronaldo é o segundo com mais seguidores online**
Jornal de Notícias - há 10 horas
O futebolista internacional português Cristiano Ronaldo é a segunda pessoa com maior ...

Cristiano Ronaldo é a segunda pessoa com mais seguidores nas redes sociais
Observador - há 11 horas

2ª Cara do Poder: Cristiano Ronaldo
TVI24 - há 9 horas

[Mais notícias de cristiano ronaldo](#)


Cristiano Ronaldo - Welcome to my Official Website
www.cristianoronaldo.com/ Traduzir esta página
Hi, my name is Cristiano Ronaldo. Can I take a selfie with you?

Cristiano Ronaldo – Wikipédia, a enciclopédia livre
https://pt.wikipedia.org/wiki/Cristiano_Ronaldo
Cristiano Ronaldo dos Santos Aveiro OIH • GOIH (Santo António, Funchal, 5 de fevereiro de 1985) é um futebolista português que joga como extremo-esquerdo ...

Cristiano Ronaldo | Facebook
<https://pt-pt.facebook.com/Cristiano/>
Cristiano Ronaldo, Madrid, 116.875.864 gostos · 5.671.162 falam sobre isto. Welcome to the OFFICIAL Facebook page of Cristiano Ronaldo....

Cristiano Ronaldo | Facebook
<https://www.facebook.com/Cristiano> Traduzir esta página
Welcome to the OFFICIAL Facebook page of Cristiano Ronaldo. [twitter.com/cristiano](#) · [instagram.com/cristiano](#) · [youtube.com/user/cristianoronaldo](#).

Cristiano Ronaldo (@cristiano) • Instagram photos and videos
<https://www.instagram.com/cristiano/> Traduzir esta página



Cristiano Ronaldo

Futebolista

Cristiano Ronaldo dos Santos Aveiro é um futebolista português que joga como extremo-esquerdo, ponta-de-lança e médio-esquerdo no Real Madrid e na Seleção Portuguesa, onde é capitão. [Wikipédia](#)

Nascimento: 5 de fevereiro de 1985 (31 anos), Hospital Dr. Nélio Mendonça, Funchal

Altura: 1,85 m

Salário: 32 milhões EUR (2016)


Filho: Cristiano Ronaldo Jr.

Filiação: Maria Dolores dos Santos Aveiro, José Diniz Aveiro

Irmãos: Liliana Catia Aveiro, Elma Aveiro, Hugo Aveiro

Pesquisas relacionadas

Ver mais de 15



[Feedback](#)

Where does the data come from ?

The image shows a screenshot of the Wikidata main page. A network diagram is overlaid on the page, consisting of nodes and connecting lines. The nodes are labeled with terms like 'open', 'multilingual', 'free', and 'collab'. The main content area features a large grey box with the text 'Welcome to Wikidata' and 'the free knowledge base with 20,081,351 data items that anyone can edit'. Below this, there are navigation links: 'Introduction • Project Chat • Community Portal • Help'. The page is divided into three main sections: 'Welcome!', 'Learn about data', and 'Get Involved'. The 'Learn about data' section includes a paragraph about data literacy and two images: a globe and a ruler. The 'Get Involved' section is partially visible at the bottom. The left sidebar contains various navigation and utility links. The top navigation bar includes language options, user status, and search.

English Not logged in Talk Contributions Create account Log in

Main Page Discussion Read View source View history Search

WIKIDATA

open

collab

multilingual

free

Welcome to Wikidata

the free knowledge base with 20,081,351 data items that anyone can edit

Introduction • Project Chat • Community Portal • Help

Welcome!



Wikidata is a free and open knowledge base that can be read and edited by both humans and machines.

Wikidata acts as central storage for the **structured data** of its Wikimedia sister projects including Wikipedia, Wikivoyage, Wikisource, and others.


Wikidata also provides support to many other sites and services beyond just Wikimedia projects! The content of Wikidata is available under a free license, exported using standard formats, and can be interlinked to other open data sets on the linked data web.

Learn about data

New to the wonderful world of data? Develop and improve your data literacy through content designed to get you up to speed and feeling comfortable with the fundamentals in no time.



item: *Earth (Q2)* property: *highest point (P610)*



Get Involved

Main page
Community portal
Project chat
Create a new item
Item by title
Recent changes
Random item
Query Service
Nearby
Help
Donate

Print/export
Create a book
Download as PDF
Printable version

In other projects
Wikimedia Commons
Meta-Wiki
Wikispecies
Wikibooks
Wikinews
Wikipedia
Wikiquote
Wikisource
Wikiversity
Wikivoyage

Tools
What links here
Related changes
Special pages
Permanent link
Page information
Wikidata item

In Wikipedia
Qafár af
Anouaa

WikiData

- **Wikidata** is a free, collaborative, multilingual, secondary database, collecting structured data to provide support for Wikipedia, Wikimedia Commons, the other wikis of the [Wikimedia movement](#), and to anyone in the world.
 - **Multilingual.** Editing, consuming, browsing, and reusing the data is fully multilingual. Data entered in any language is immediately available in all other languages. Editing in any language is possible and encouraged.
 - **A secondary database.** Wikidata records not just statements, but also their sources, and connections to other databases. This reflects the diversity of knowledge available and supports the notion of verifiability.

WikiData

The image shows a WikiData entry for Douglas Adams (Q42) with various components annotated. The entry includes a label, a description, a list of statements, and a list of references. Annotations point to specific parts of the entry, such as the label, description, property, value, qualifiers, rank, statement group, opened references, and collapsed reference.

label — Douglas Adams (Q42) — **item identifier**

description — English writer and humorist
Douglas Noël Adams | Douglas Noel Adams — **aliases**
▶ In more languages

Statements

property — educated at — **value** — St John's College

qualifiers — end time: 1974
academic major: English literature
academic degree: Bachelor of Arts
start time: 1971

rank — 2 references

statement group — **opened references**

stated in	Encyclopædia Britannica Online
reference URL	http://www.nndb.com/people/731/000023662/
original language of work	English
retrieved	7 December 2013
publisher	NNDB
title	Douglas Adams (English)

+ add reference

collapsed reference — Brentwood School

end time	1970
start time	1959

▶ 0 references

+ add (statement)

WikiData

The image shows a WikiData entry for Douglas Adams (Q42). The entry includes a label, a description, and a list of statements. Annotations point to various parts of the interface:

- label**: Douglas Adams (Q42)
- item identifier**: (Q42)
- description**: English writer and humorist
Douglas Noël Adams | Douglas Noel Adams
▶ In more languages
- aliases**: Douglas Noël Adams | Douglas Noel Adams
- property**: educated at
- value**: St John's College
- rank**: Douglas Adams
- statement group**: A group of statements including an opened reference and a collapsed reference.

Item	Property	Value
Q42	P69	Q691283
Douglas Adams	educated at	St John's College

opened references

reference URL	http://www.nndb.com/people/731/000023662/
original language of work	English
retrieved	7 December 2013
publisher	NNDB
title	Douglas Adams (English)

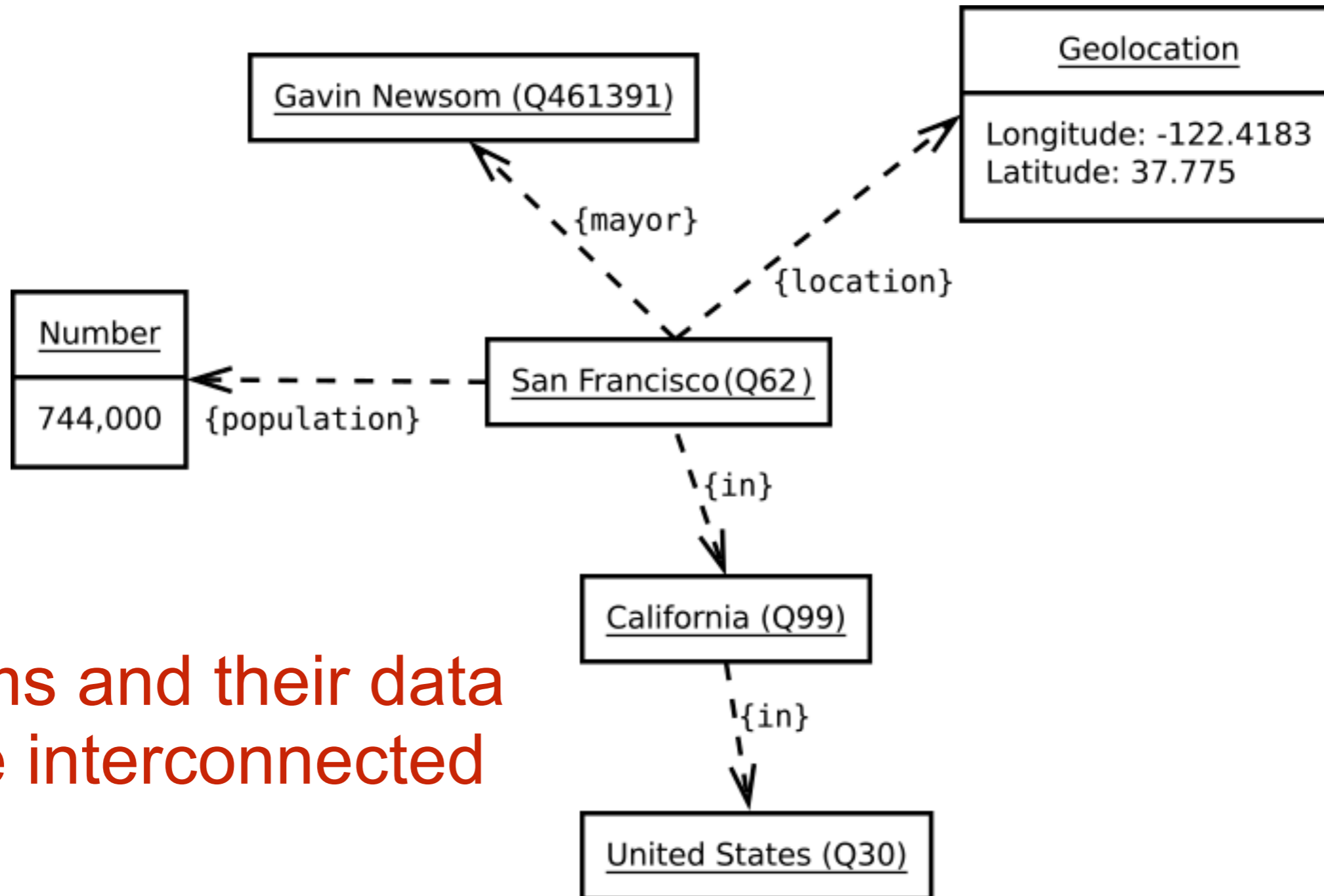
+ add reference

collapsed reference

▶ 0 references

+ add (statement)

WikiData



Items and their data
are interconnected

The Semantic Web Vision

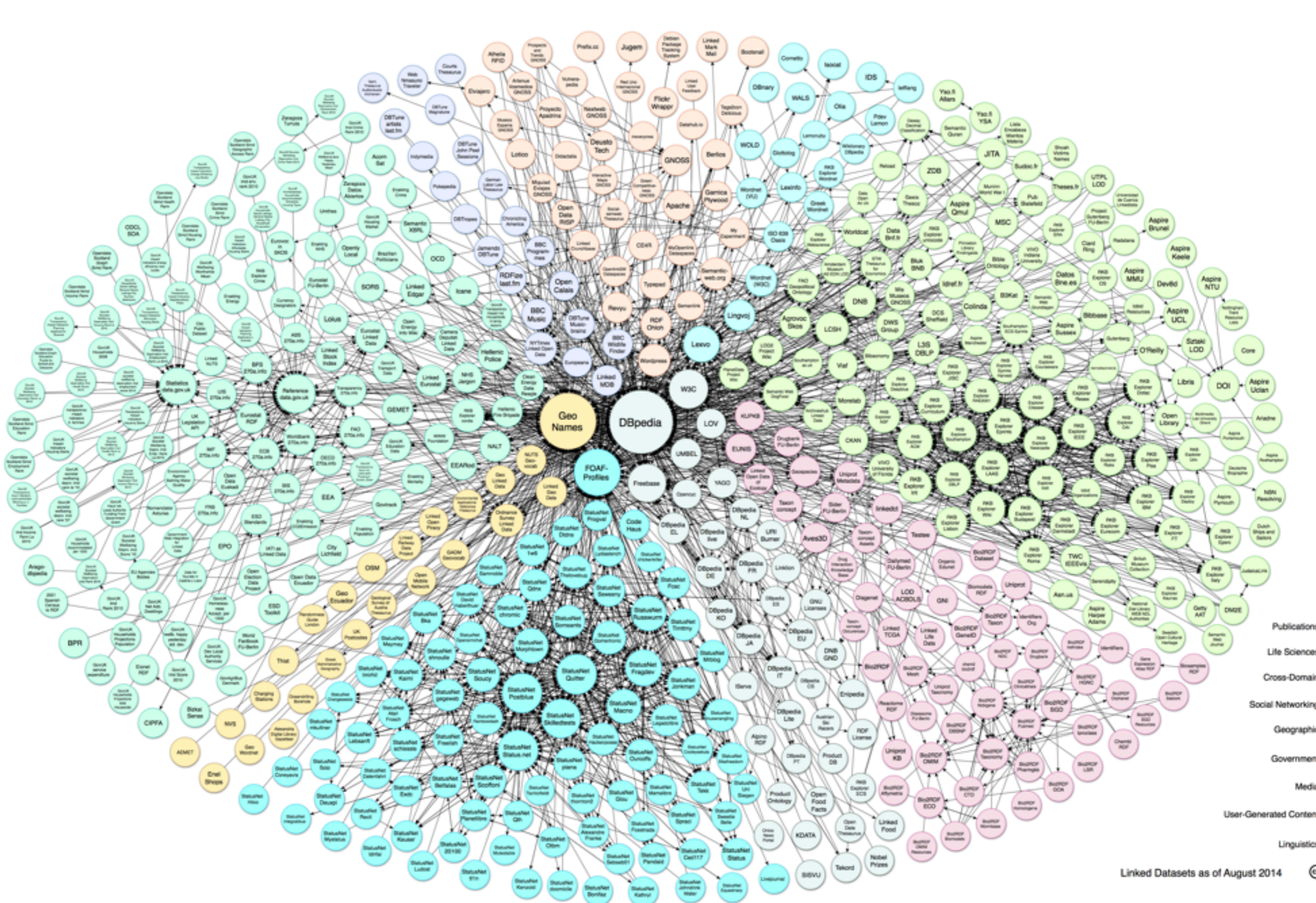
“I have a dream for the Web [in which computers] become capable of analyzing all the data on the Web – the content, links, and transactions between people and computers. A ‘Semantic Web’, which should make this possible, has yet to emerge, but when it does, the day-to-day mechanisms of trade, bureaucracy and our daily lives will be handled by machines talking to machines. The ‘intelligent agents’ people have touted for ages will finally materialize.”

Tim Berners-Lee, 1999



Linked Open Data Project

- An open initiative project
- “Exposing, sharing, and connecting pieces of data, information, and knowledge on the Semantic Web using URIs and RDF.”
- Large number of datasets with connections between them
- Billions of triples, millions of links!

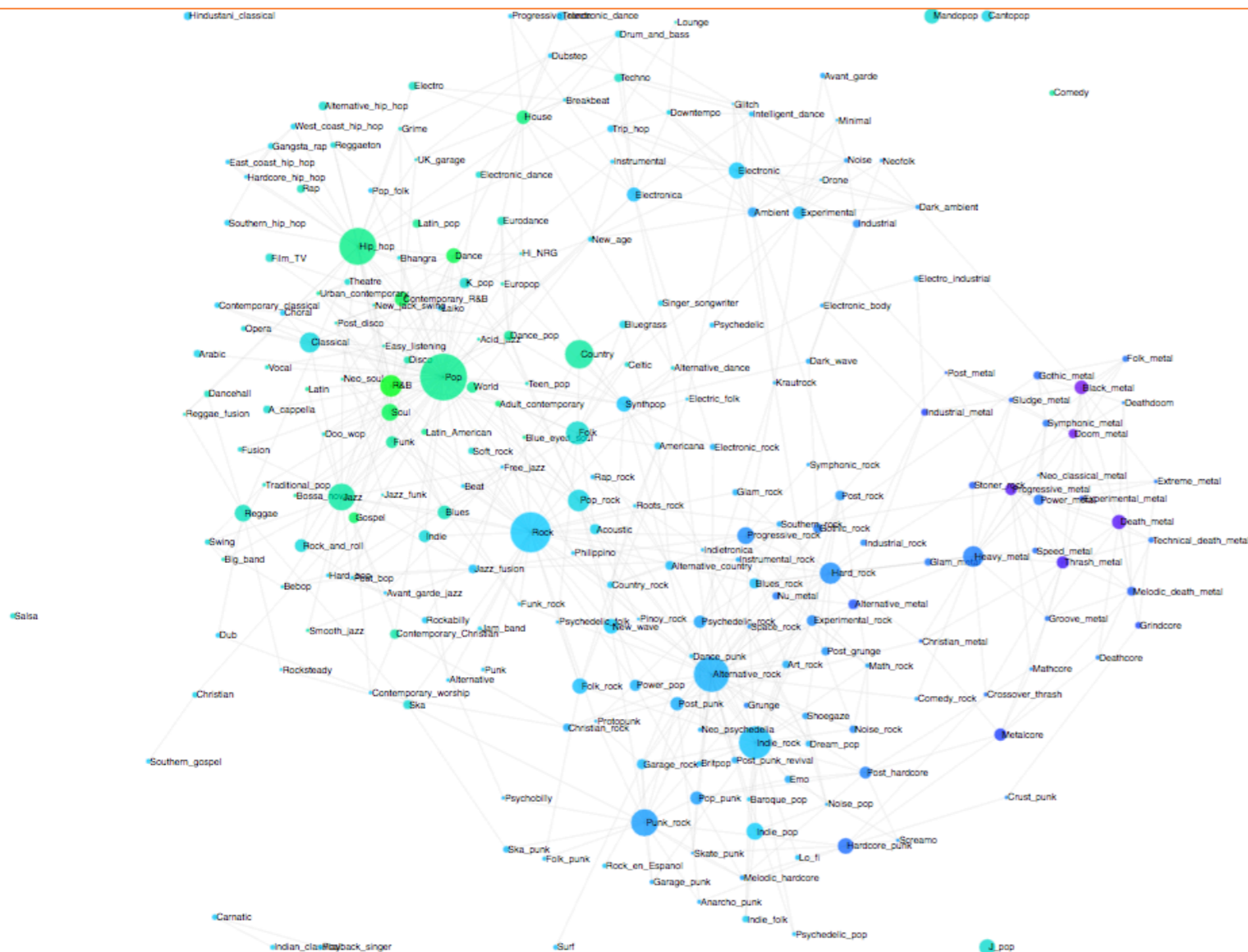


Linked Datasets as of August 2014

Linked Open Data

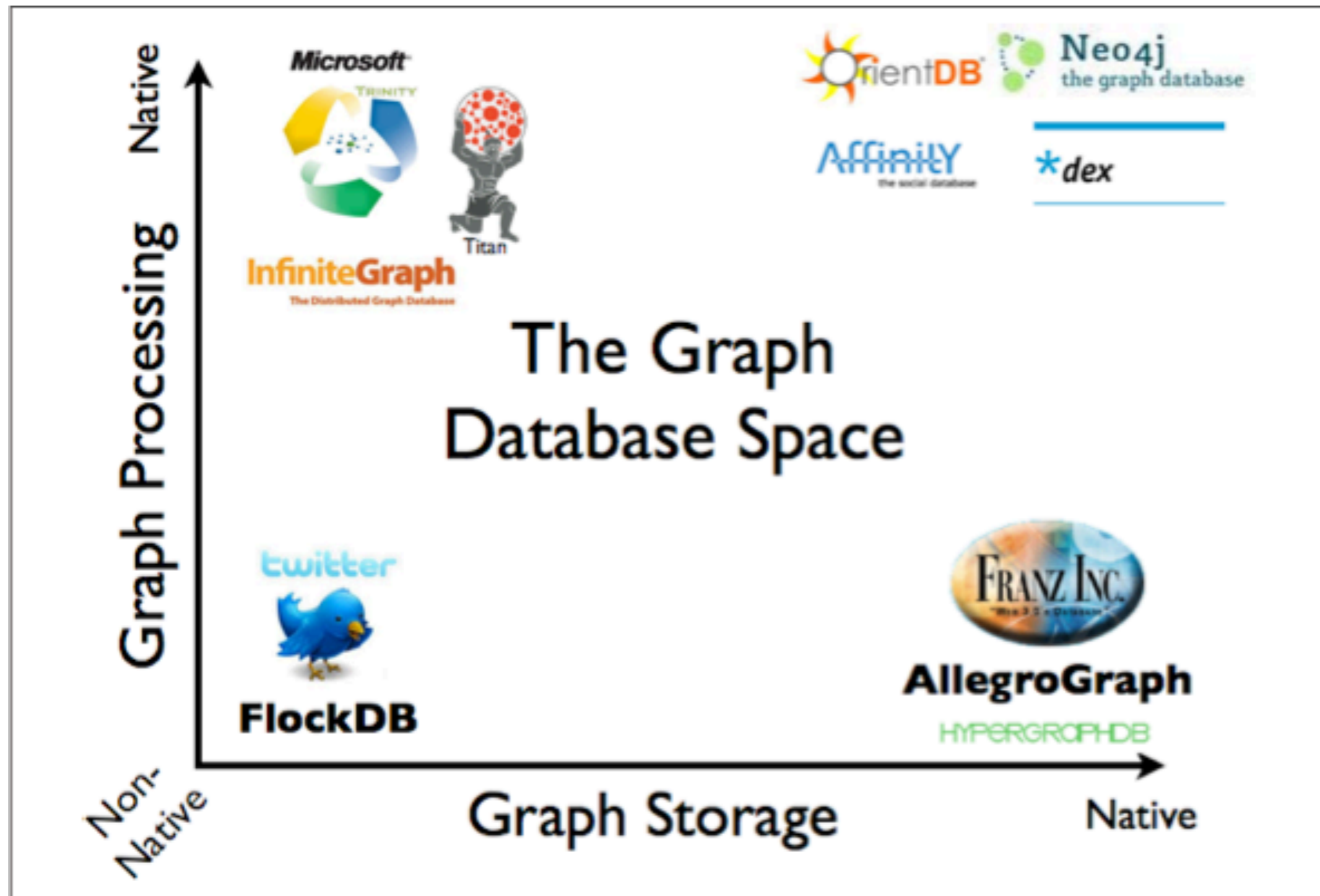
Music Genre Map

Genre frequency (circle size) and layout (genres that label the same bands are linked) for bands pulled from dbpedia



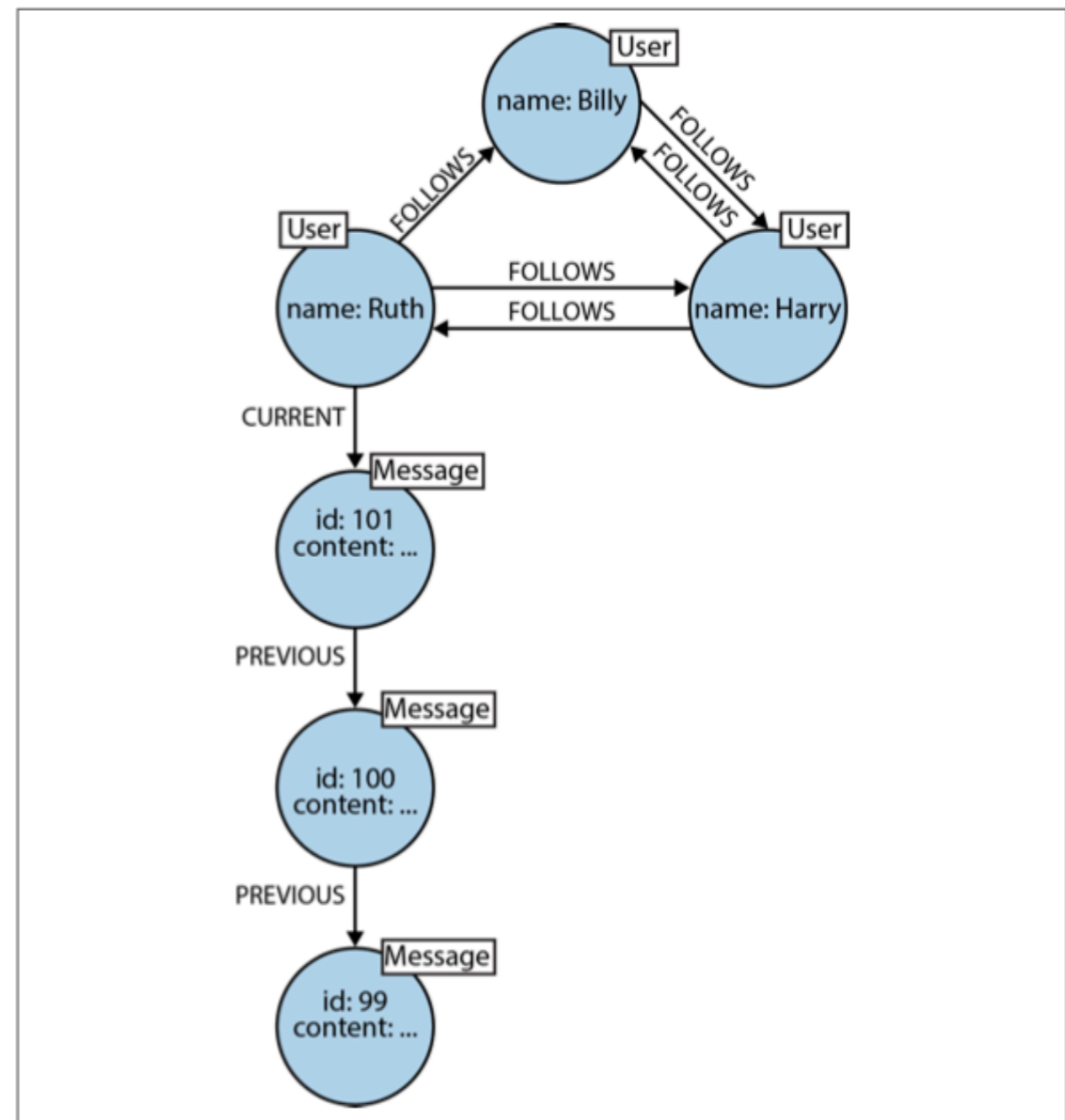
- Data are from 37k bands pulled from dbpedia.org
- d3.js code adapted from [here](#)
- Genres labeling fewer than 50 bands are omitted
- Genre labels that co-occurred (beyond a threshold I keep changing) are linked, closer links = more correspondence
- Circle area is proportional to amount of band labeling that genre label accomplished (such that a genre label for a band with two genre labels would count half as much as a genre label for a band with one label)
- Circle color is based on [unidimensional scaling](#) of the genre correspondence matrix (basically, color should ROUGHLY correspond to graph location)

Graph Database Space



Labeled Property Graphs

- Formed by nodes and relationships
- Nodes can contain properties (key-value pairs)
- Nodes can be labeled with one or more labels
- Relationships are named and directed, and always have a start and end node.
- Relationships can also contain properties.



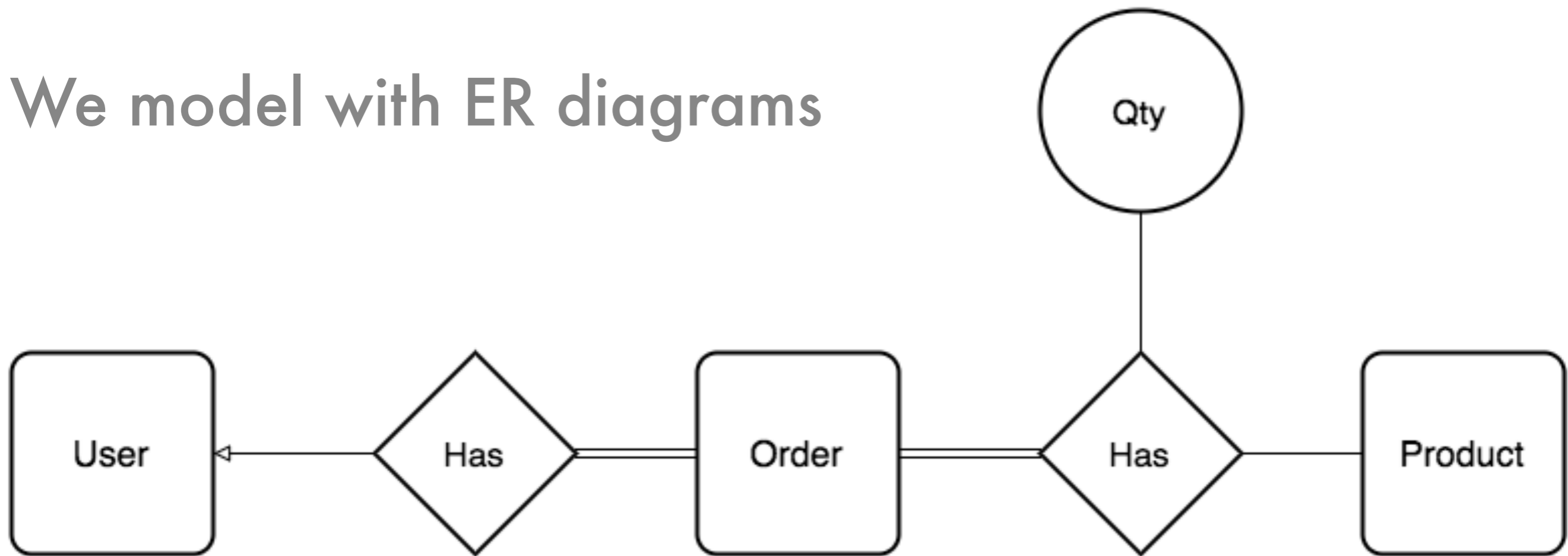
RDBMs lack relationships

- Relational Database Systems store data in:

TABLES

RDBMs lack relationships

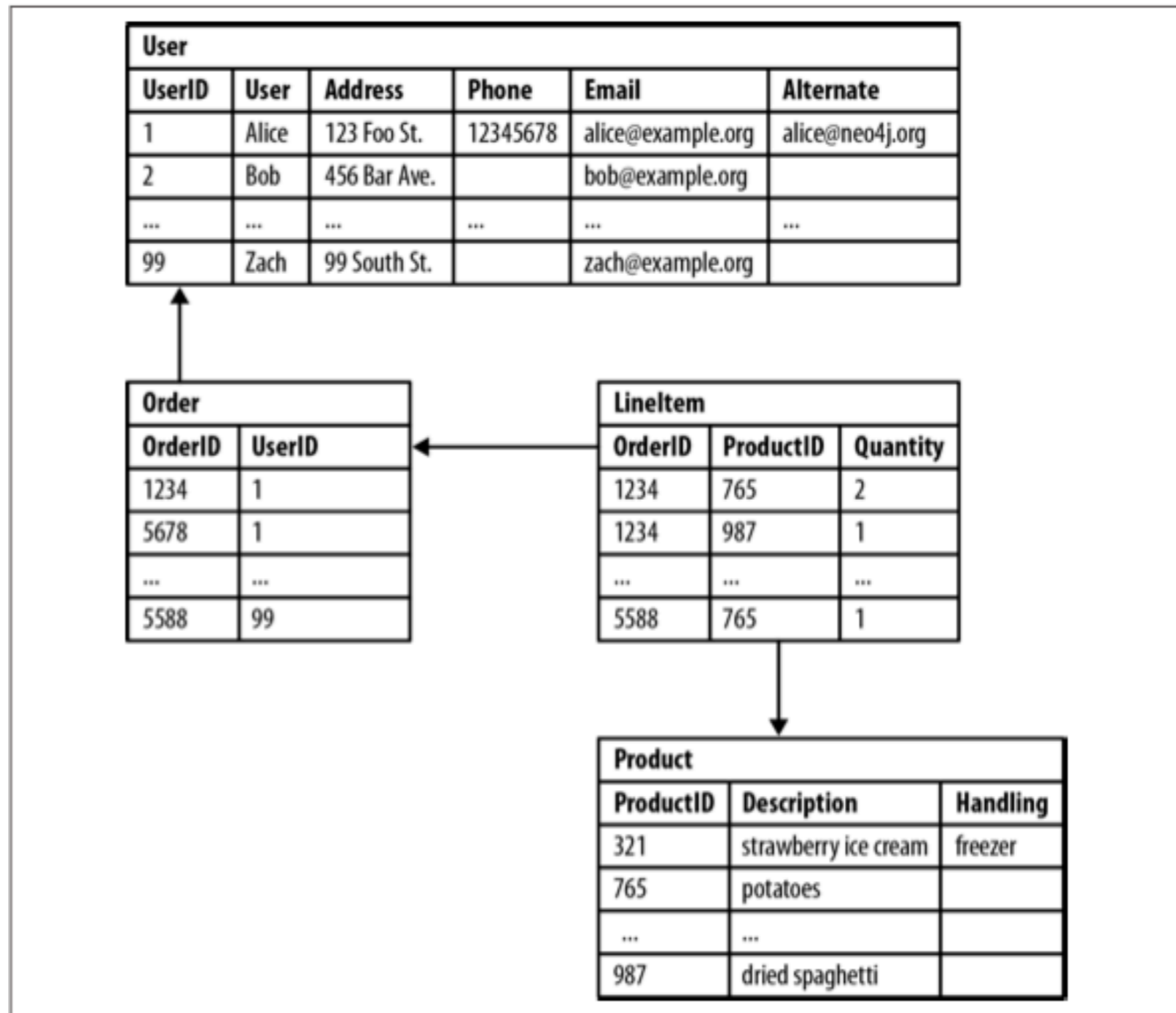
We model with ER diagrams



RDBMs lack relationships

We implement
with tables
and foreign
keys

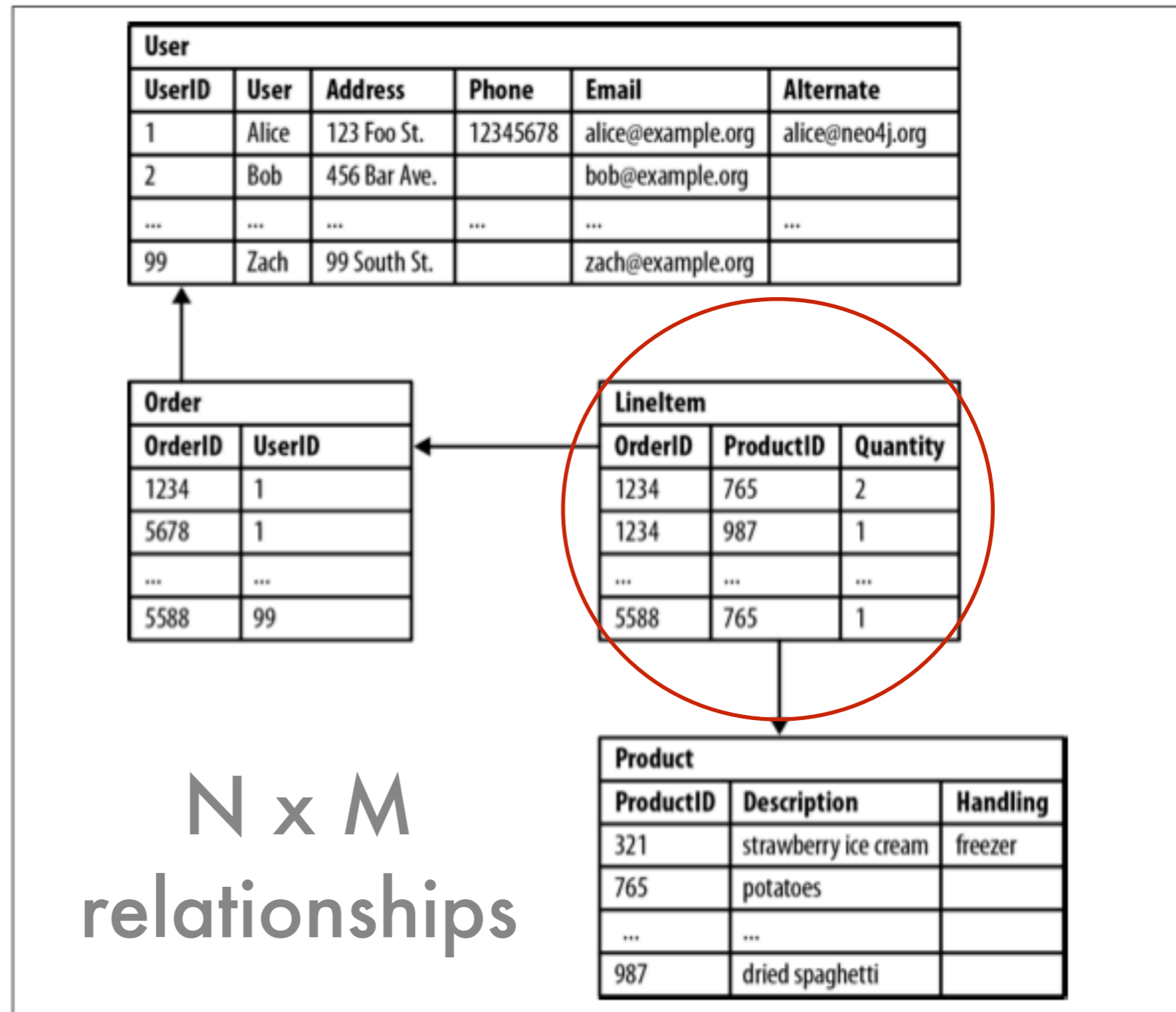
(everything are
tables)



RDBMs lack relationships

We implement
with tables
and foreign
keys

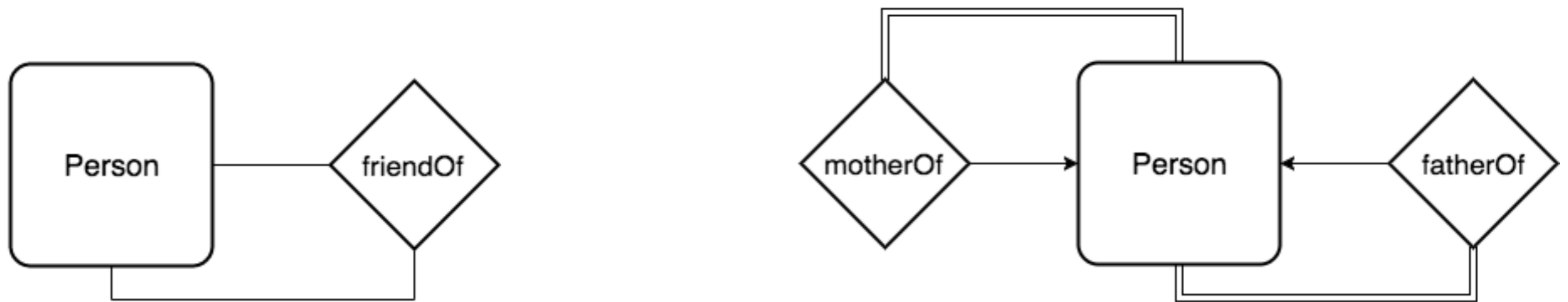
(everything are
tables)



Major issues with RDBMs

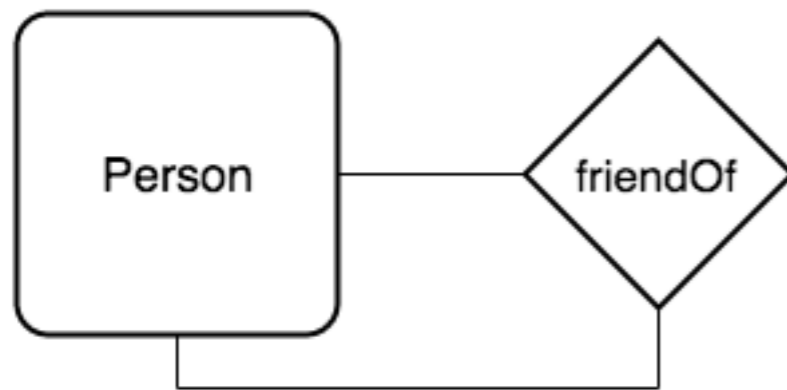
- When normalizing / optimizing the table schema entities get together with relationships resulting in accidental complexity:
 - Mixtures Business Data with Foreign Key Metadata
- Foreign keys add extra overhead at:
 - Development time
 - Production time
- Sparse tables with nullable columns require special extra code to check and handle (and queries are more difficult)
- Joins involving several tables might be necessary just to get simple information (e.g. what a customer bought?)
- Reciprocal queries can be costly (what products did a customer buy today?)
- Interconnected domains and recursive queries are hard or not supported

Representing connections



How would you represent this in a relational database?

Querying a social network



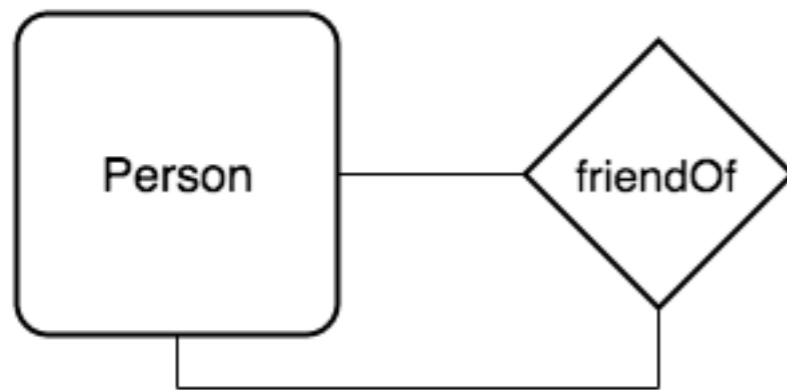
Person	
ID	Person
1	Alice
2	Bob
...	...
99	Zach

→

PersonFriend	
PersonID	FriendID
1	2
2	1
2	99
...	...
99	1

- Bob's friends
- Who is friends with Bob
- Alice's friends of friends

Bob's friends



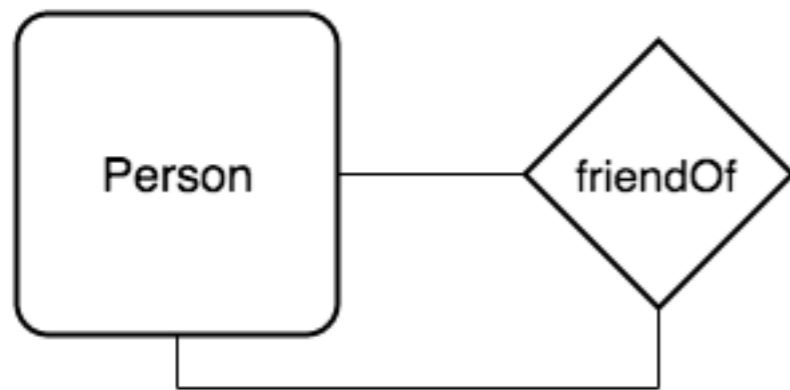
Person	
ID	Person
1	Alice
2	Bob
...	...
99	Zach

→

PersonFriend	
PersonID	FriendID
1	2
2	1
2	99
...	...
99	1

```
SELECT p1.Person
FROM Person p1 INNER JOIN PersonFriend
  ON ( PersonFriend.PersonID = p1.ID )
INNER JOIN Person p2
  ON ( PersonFriend.FriendID = p2.ID )
WHERE p1.Person = 'Bob'
```


Who is friends with Bob?



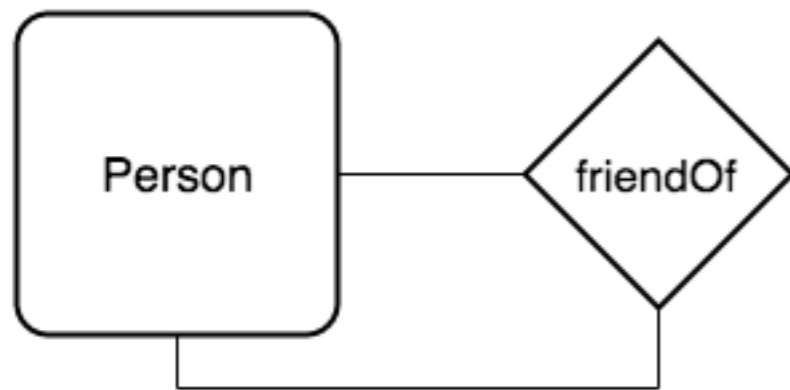
Person	
ID	Person
1	Alice
2	Bob
...	...
99	Zach

→

PersonFriend	
PersonID	FriendID
1	2
2	1
2	99
...	...
99	1

```
SELECT p1.Person
FROM Person p1 INNER JOIN PersonFriend
  ON ( PersonFriend.FriendID = p1.ID )
INNER JOIN Person p2
  ON ( PersonFriend.PersonID = p2.ID )
WHERE p2.Person = 'Bob'
```

Alice's friends-of-friends



Person	
ID	Person
1	Alice
2	Bob
...	...
99	Zach

→

PersonFriend	
PersonID	FriendID
1	2
2	1
2	99
...	...
99	1

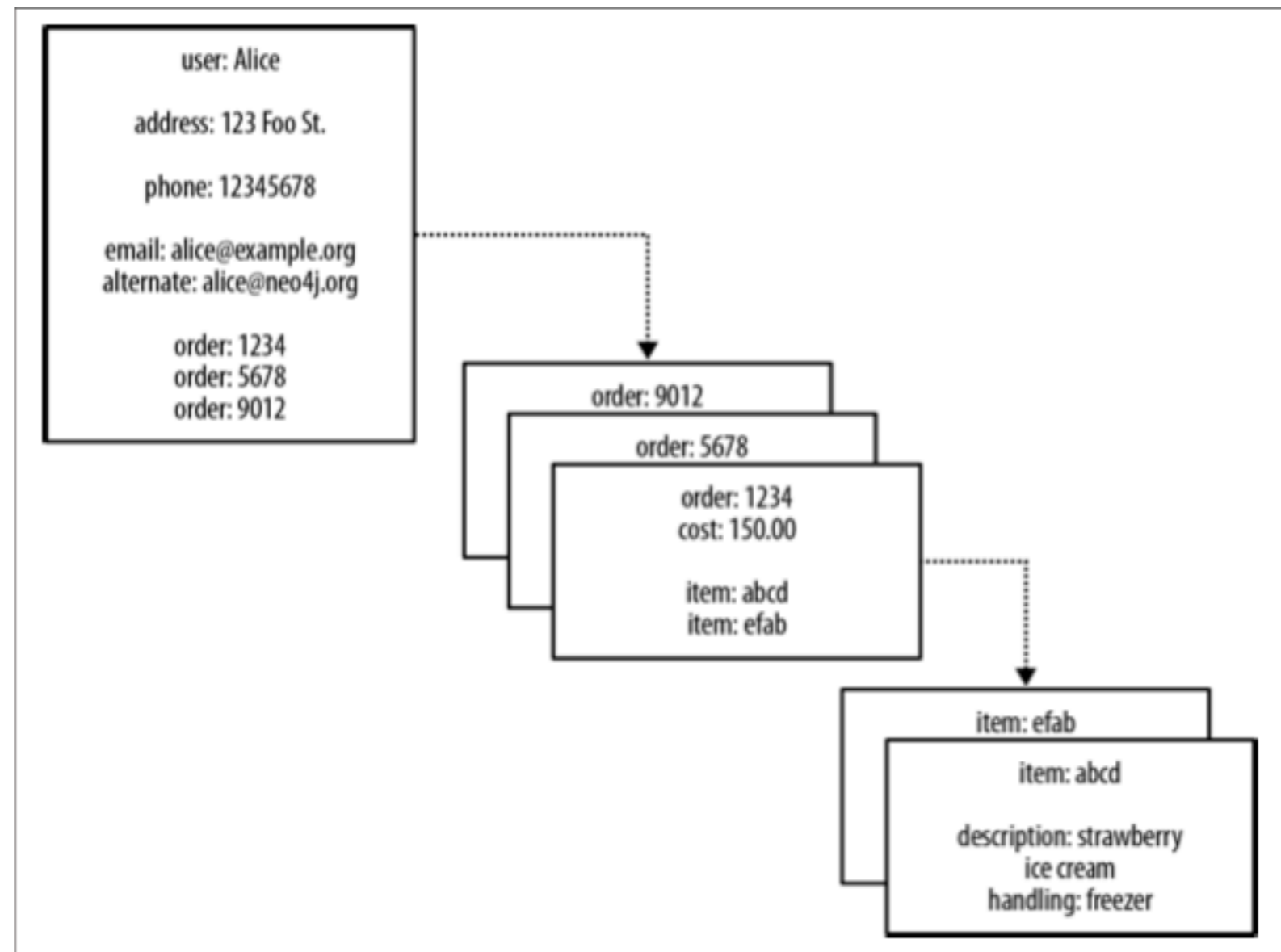
```
SELECT p1.Person AS PERSON, p2.Person AS FRIEND_OF_FRIEND
FROM PersonFriend pf1 JOIN Person p1
  ON ( pf1.PersonID = p1.ID )
JOIN PersonFriend pf2
  ON ( pf2.PersonID = pf1.FriendID )
JOIN Person p2
  ON ( pf2.FriendID = p2.ID)
WHERE p1.Person = 'Alice' AND pf2.FriendID <> p1.ID
```

NOSQL lack relationships

- Most NoSQL databases lack mechanisms to make connections among data, at least without significant penalty at query time that are not appropriate for real-time services
- Document stores usually resort to embedded structures that can be quite efficient for several applications, but do lack the flexibility required by general graph databases and require extensive care for keeping consistency
- Graph databases distinguish themselves by index-free adjacency lists that allow better performance when traversing graph structure.

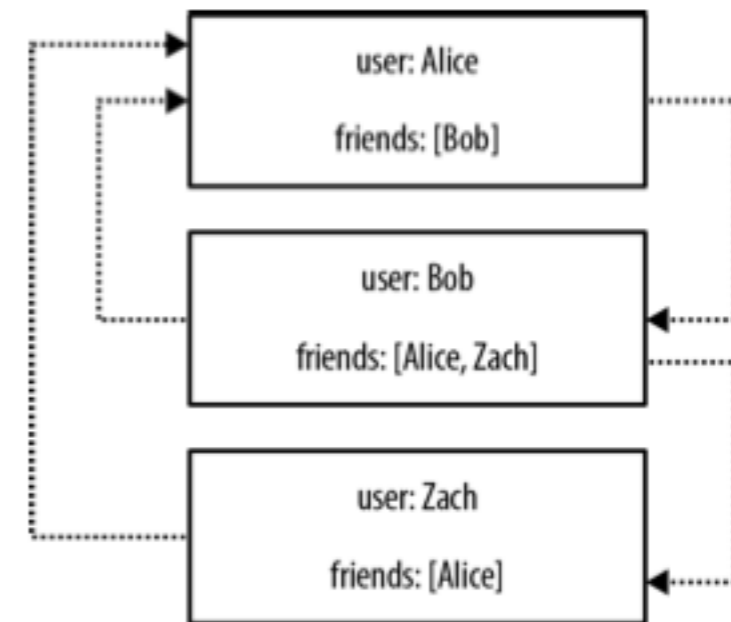
Linking data in NOSQL

- Simulation of foreign keys
- However, usually no support for consistency
- Recent versions of NOSQL major databases support some form of indexing to optimize querying



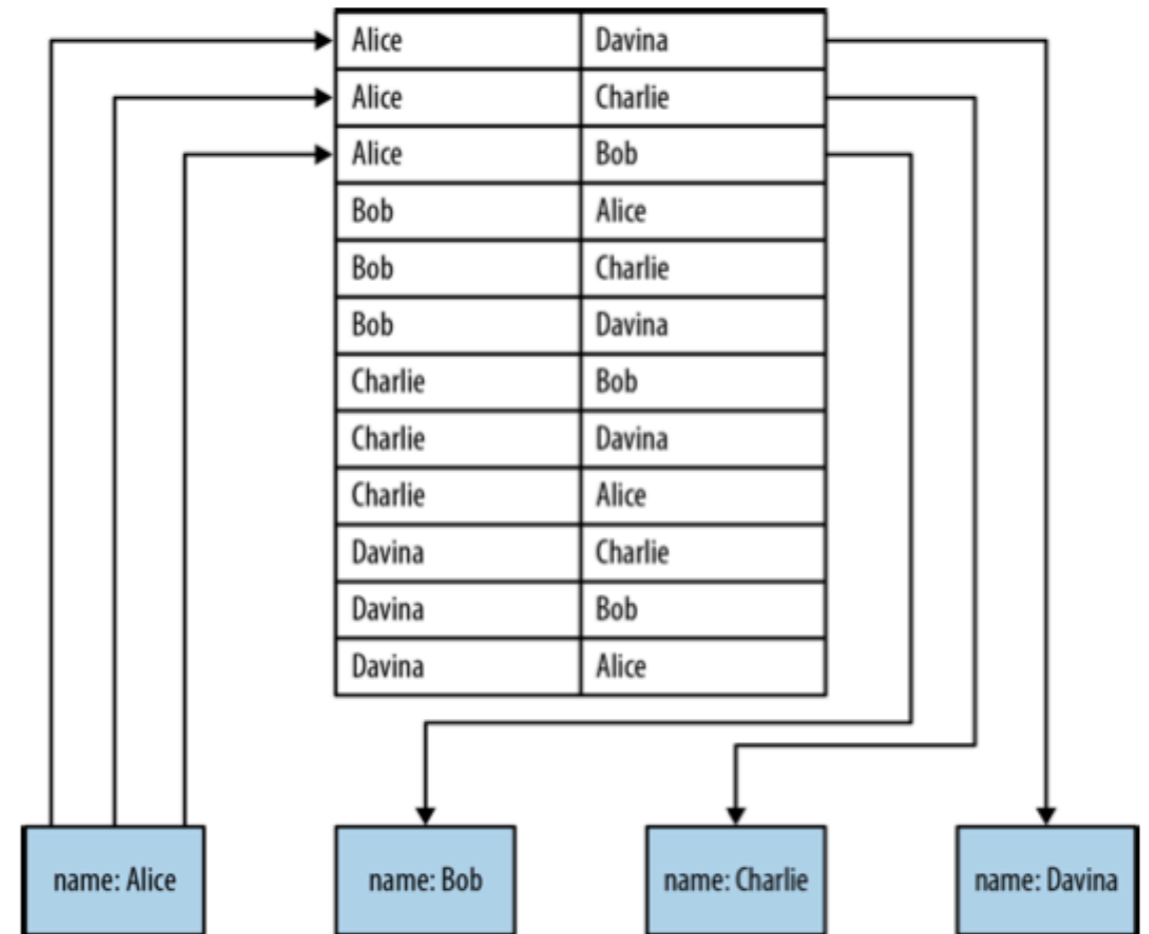
Linking data in NOSQL

- Explicitly store connections in the value
- If needed, we could add a reverse direction “friend_of”
- How to represent as an “embedded document” ?
- How to perform efficient traversals?

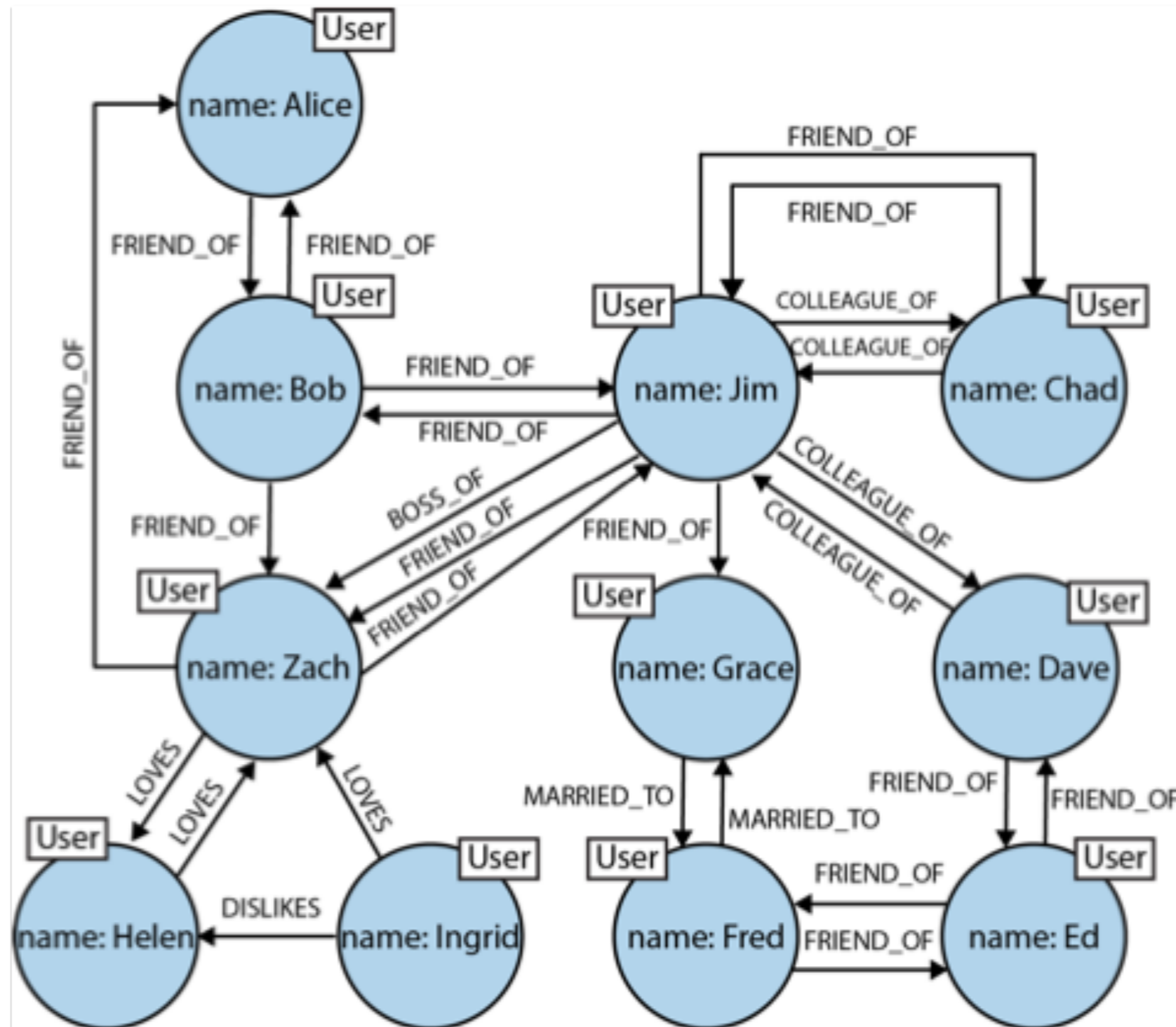


Graphs in RDBMs

- Usually, we have a table (or tables) for representing nodes preferably with a numeric primary key
- Usually, we have a table (or tables) for representing edges / properties / relationships with a composite primary key



Graph DBs Embrace Relationships



Native Graphs of Neo4j

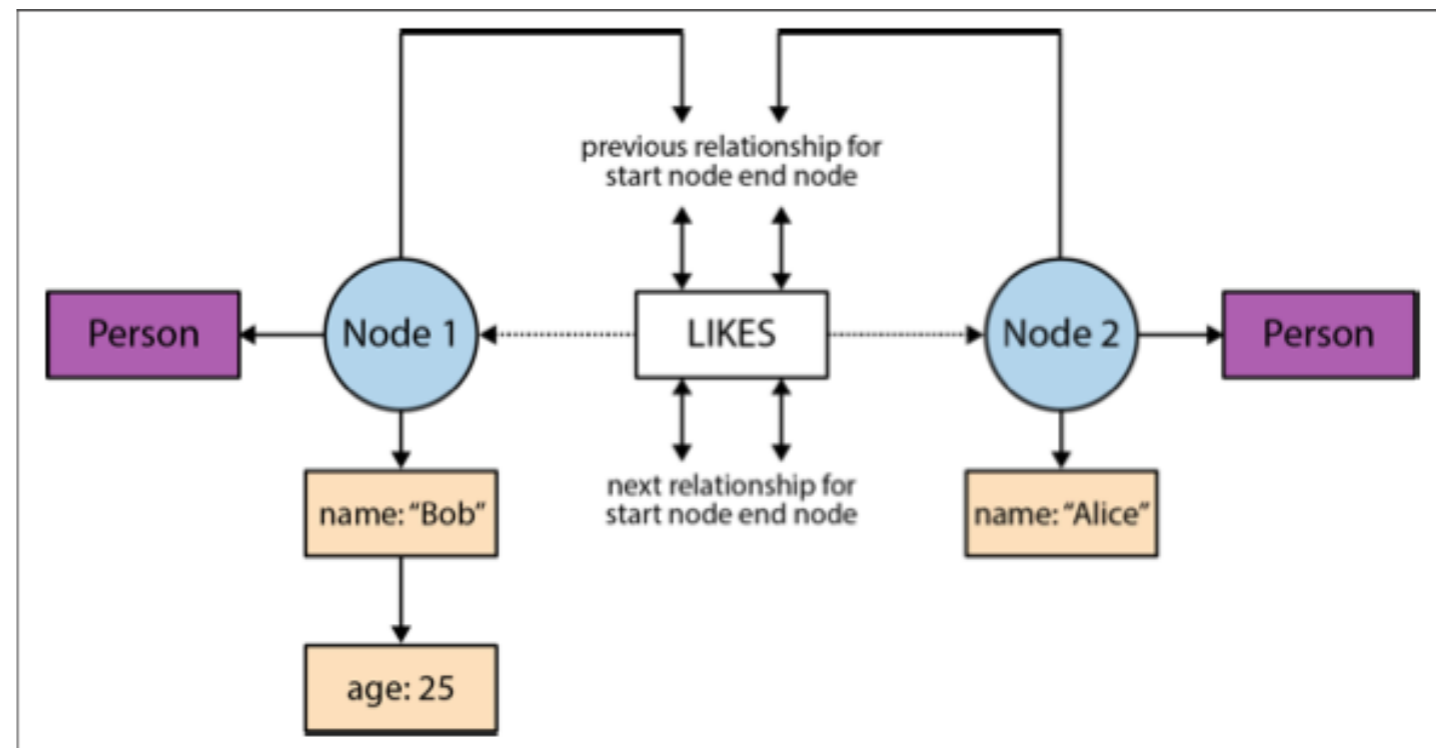
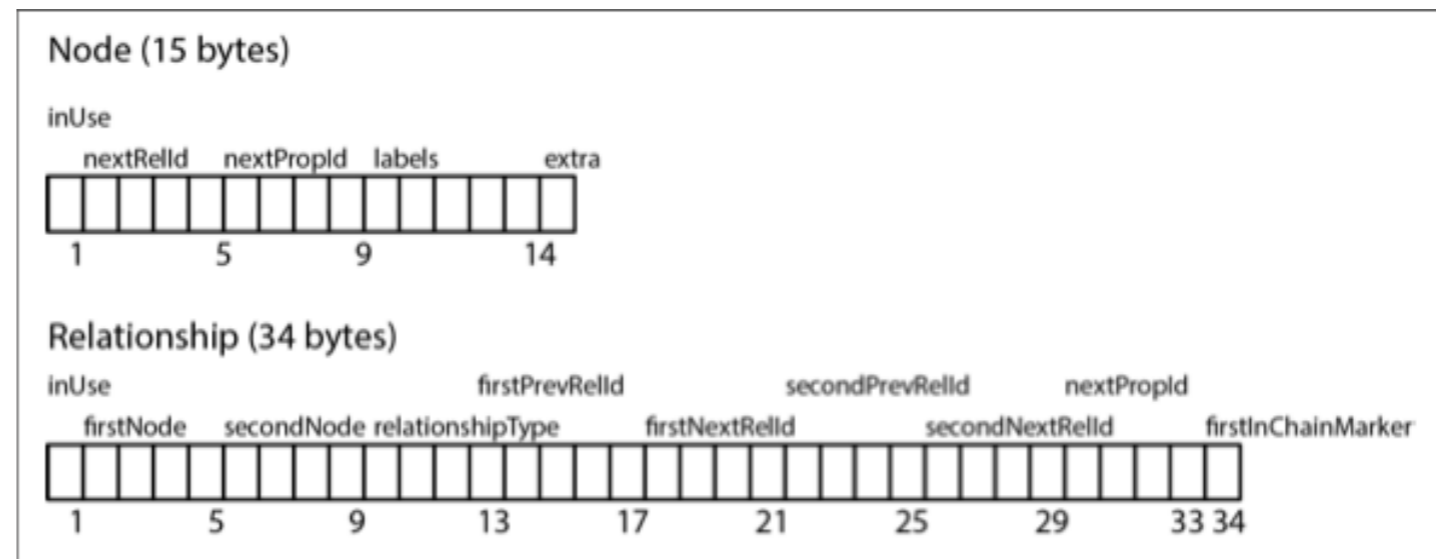
Table 2-1. Finding extended friends in a relational database versus efficient finding in Neo4j

Depth	RDBMS execution time(s)	Neo4j execution time(s)	Records returned
2	0.016	0.01	~2500
3	30.267	0.168	~110,000
4	1543.505	1.359	~600,000
5	Unfinished	2.132	~800,000

Graph DBs use index-free adjacency to ensure that traversing connected data is extremely rapid.

Native Graphs of Neo4j

- Information is stored in several files, most of them having fixed-length records
- The most important are **nodestore** and **relationshipstore**
- Using fixed-length records allows $O(1)$ retrieval of records



Data Modeling

with Graphs

Graph Models

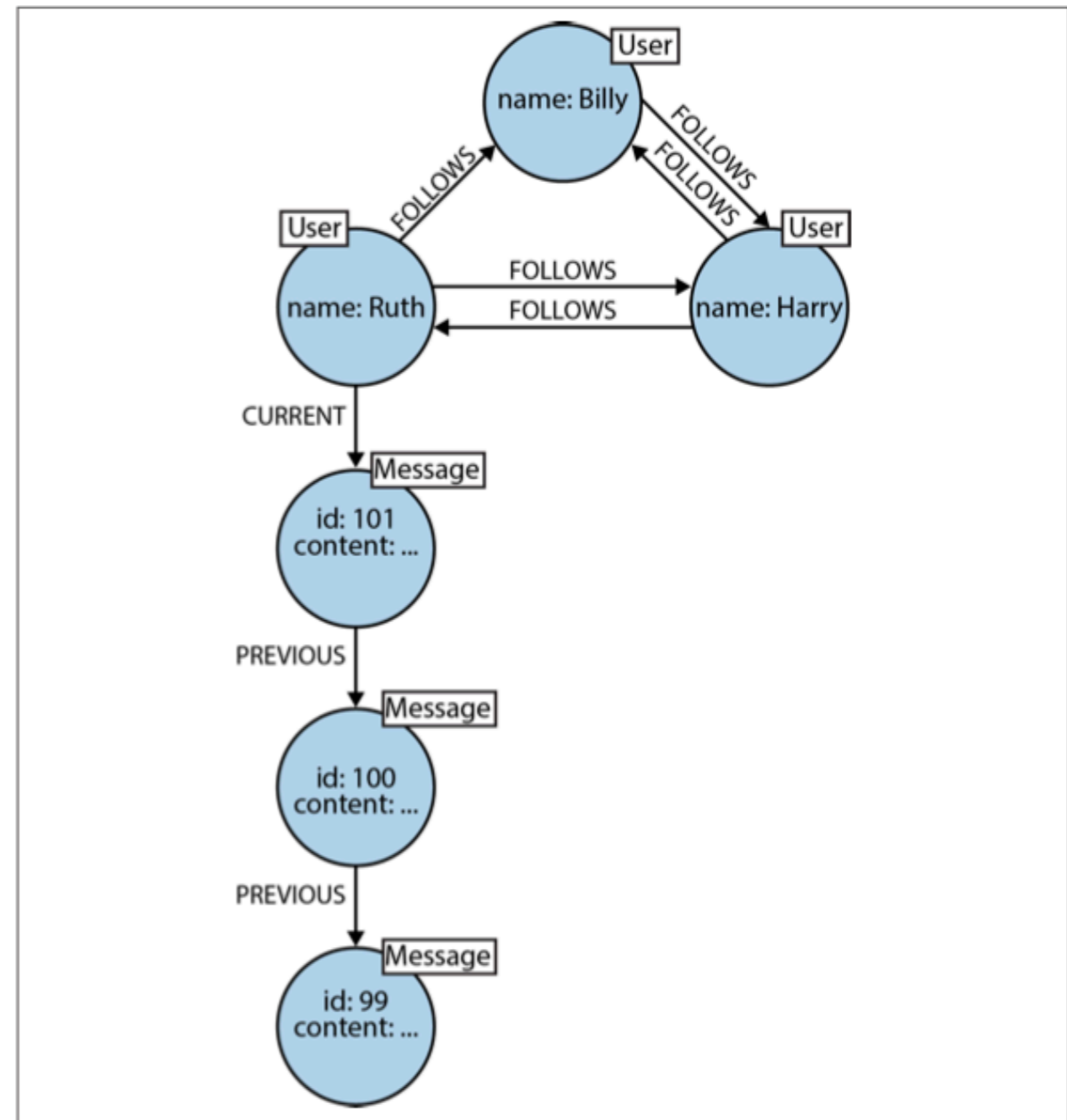
- Models are abstractions of reality to represent particular aspects of the world, usually guided by specific objectives, goals, motivations, and in the case of Computer Science the problem we are trying to solve
- Computer Science uses several models:
 - Physics like classical mechanisms used in games
 - Entity-relationship Models and Relational Model
 - UML Models (which are in fact another kind of Graph Model with a lot of extra features)
 - Graph Models (e.g. for representing networks)
 - Logic Representations (e.g. for Mathematics)
 - Category theory (a really abstract model of mathematics...)

Why Graphs are Cool

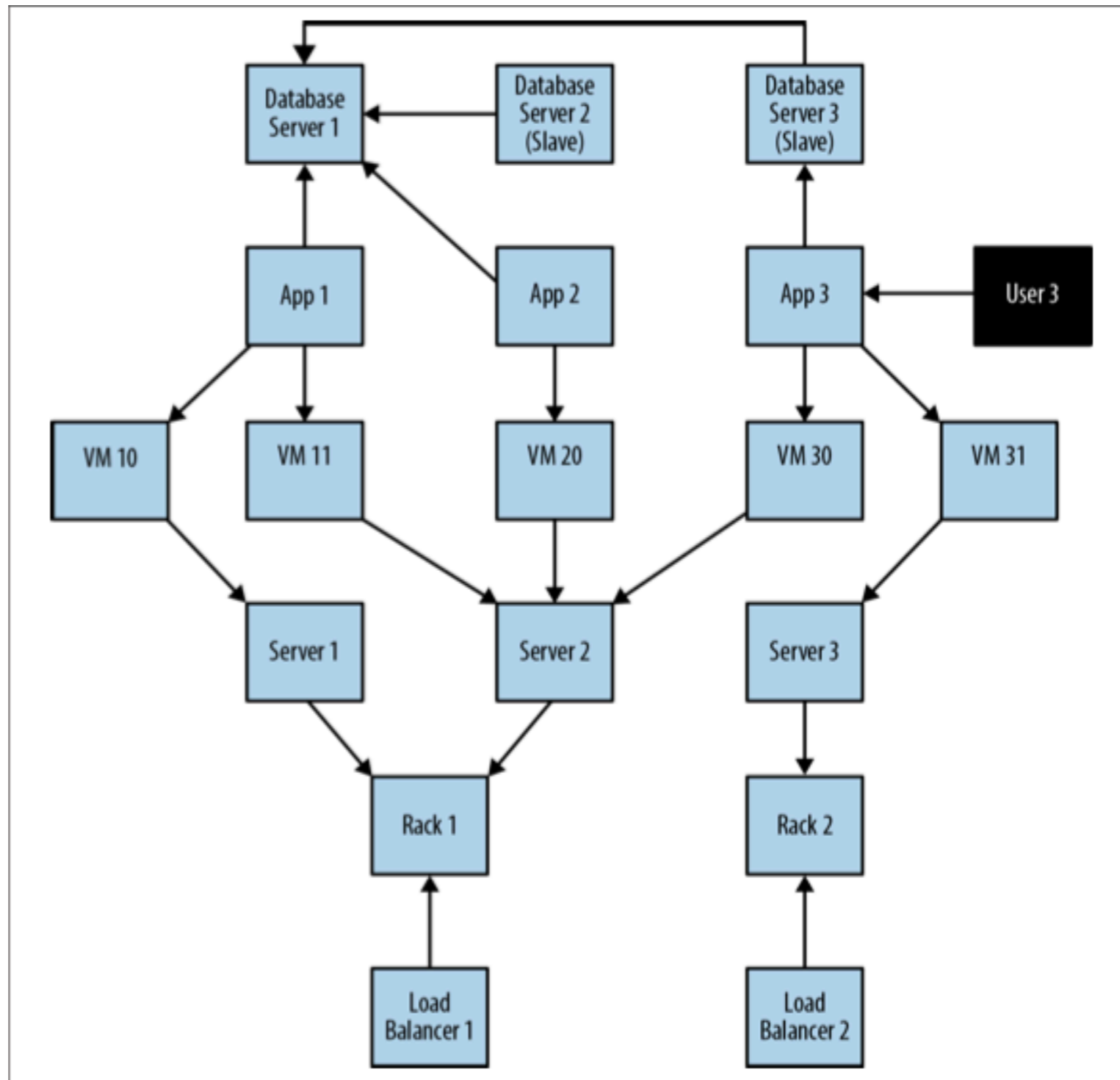
- Humans often communicate by drawing diagrams, using shapes and arrows (look at your preferred slide making applications)
- Graphs are the abstract representation of these diagrams, reducing the impedance mismatch between “analysis” and “implementation”, and are neat to present!
- Properties of graphs are very well-studied in mathematics and computer science
- A lot of problems are already encoded in graphs with appropriate algorithms to solve them (e.g. recall A^* for solving search-problems in A.I., or your car GPS)
- Graphs are VERY expressive...

Labeled Property Graphs

- Formed by **nodes** and **relationships**, **properties**, and **labels**
- Nodes can contain **properties** (key-value pairs)
- Nodes can be tagged with one or more labels
- Relationships are named and directed, and always have a start and end node.
- Relationships can also contain properties.



A data center



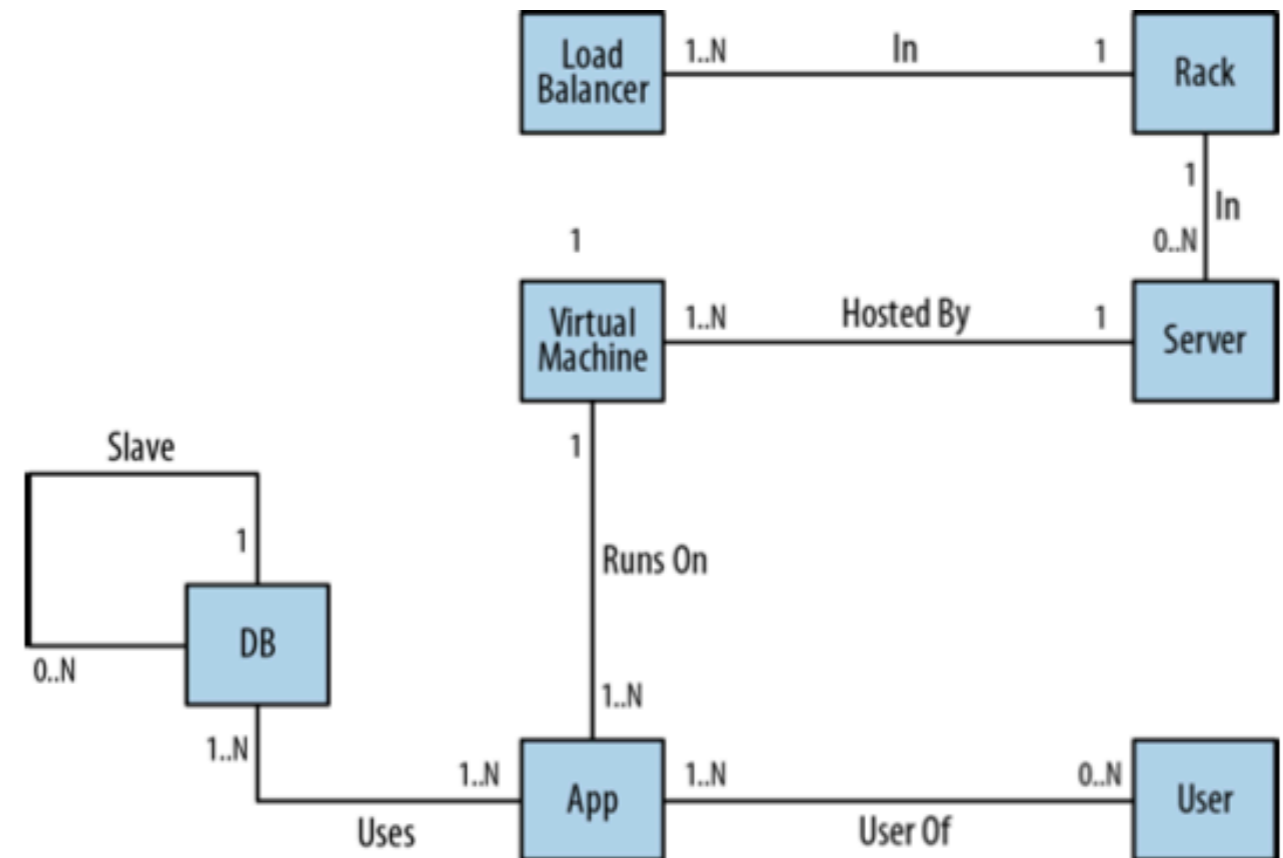
A data center

As the operators of such a system, we have two primary concerns:

- Ongoing provision of functionality to meet (or exceed) a service-level agreement, including the ability to perform forward-looking analyses to determine single points of failure, and retrospective analyses to rapidly determine the cause of any customer complaints regarding the availability of service.
- Billing for resources consumed, including the cost of hardware, virtualization, network provisioning, and even the costs of software development and operations (since these are simply logical extensions of the system we see here).

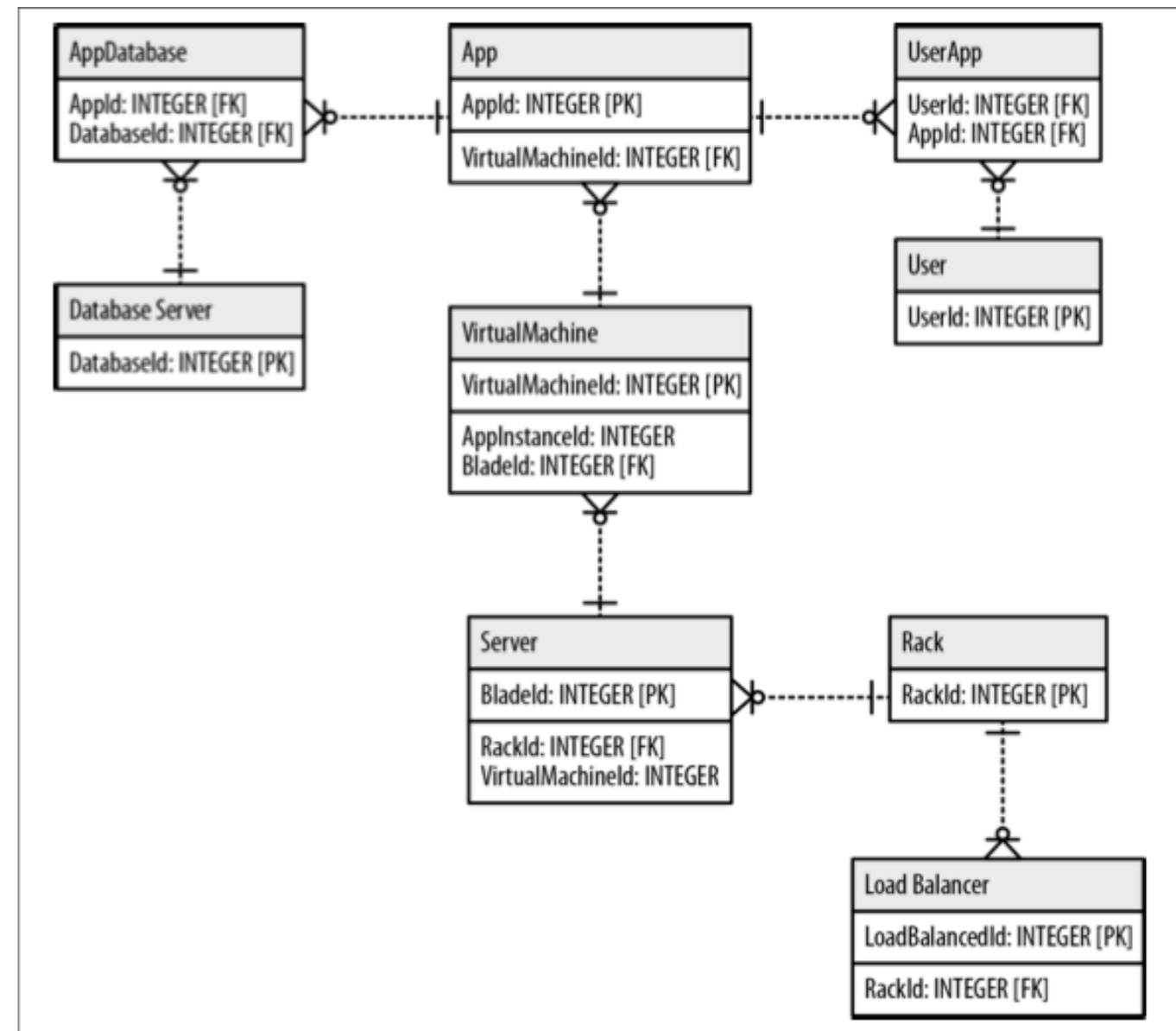
A data center ER model

- ER diagrams are graphs
- Relationships are unidirectional (implicit in the name)
- Other languages do support some extra-features (e.g. UML):
 - Directionality
 - N-ary relations
 - Inheritance
 - Association classes



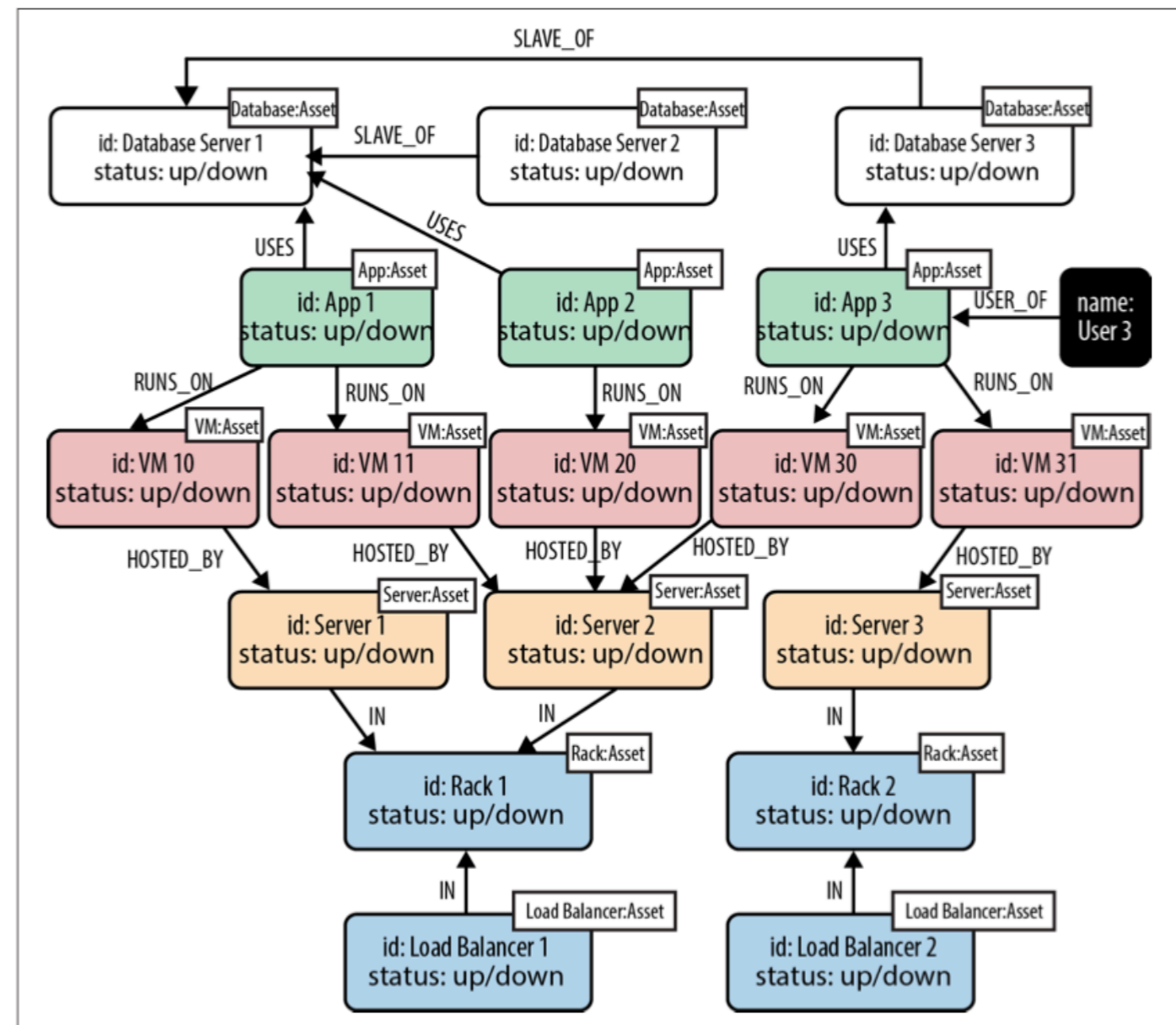
A Data Center Table Schema

- Not clear which tables represent entities or relationships
- Foreign keys are introduced to capture relationships
- Denormalization sometimes required to optimize queries
- Difficult to evolve without code rewriting and significant risks



A Data Center Graph Model

- Enrich the diagrams with:
 - Roles as Labels
 - Attributes as Properties
 - Connections as Relationships
- Test the model by
 - Checking if graph “reads well”
 - Describing use-case queries
- Avoid antipatterns:
 - Don't encode entities into relationships
 - Do not conflate



Checking the model

- When a user reports a problem, we can limit the physical fault-finding to problematic network elements between the user and the application and the application and its dependencies

```
MATCH (user:User)-[*1..5]-(asset:Asset)
WHERE user.name = 'User 3' AND asset.status = 'down'
RETURN DISTINCT asset
```

- The MATCH clause here describes a *variable length path* between one and five relationships long. The relationships are unnamed and undirected (there's no colon or relationship name between the square brackets, and no arrowtip to indicate direction).

Checking the model

```
MATCH (user:User)-[*1..5]-(asset:Asset)
WHERE user.name = 'User 3' AND asset.status = 'down'
RETURN DISTINCT asset
```

```
(user)-[:USER_OF]->(app)
(user)-[:USER_OF]->(app)-[:USES]->(database)
(user)-[:USER_OF]->(app)-[:USES]->(database)-[:SLAVE_OF]->(another-database)
(user)-[:USER_OF]->(app)-[:RUNS_ON]->(vm)
(user)-[:USER_OF]->(app)-[:RUNS_ON]->(vm)-[:HOSTED_BY]->(server)
(user)-[:USER_OF]->(app)-[:RUNS_ON]->(vm)-[:HOSTED_BY]->(server)-[:IN]->(rack)
(user)-[:USER_OF]->(app)-[:RUNS_ON]->(vm)-[:HOSTED_BY]->(server)-[:IN]->(rack)
  <-[:IN]-(load-balancer)
```


Modeling guidelines

- Describe the Model in Terms of the Application's Needs (queries to perform)
- Nodes for Things, Relationships for Structure
- Fine-Grained versus Generic Relationships
- Use well-known patterns:
 - Model Facts as Nodes
 - Represent Complex Value Types as Nodes
 - Patterns for Geospatial and Time data

Nodes or relationships?

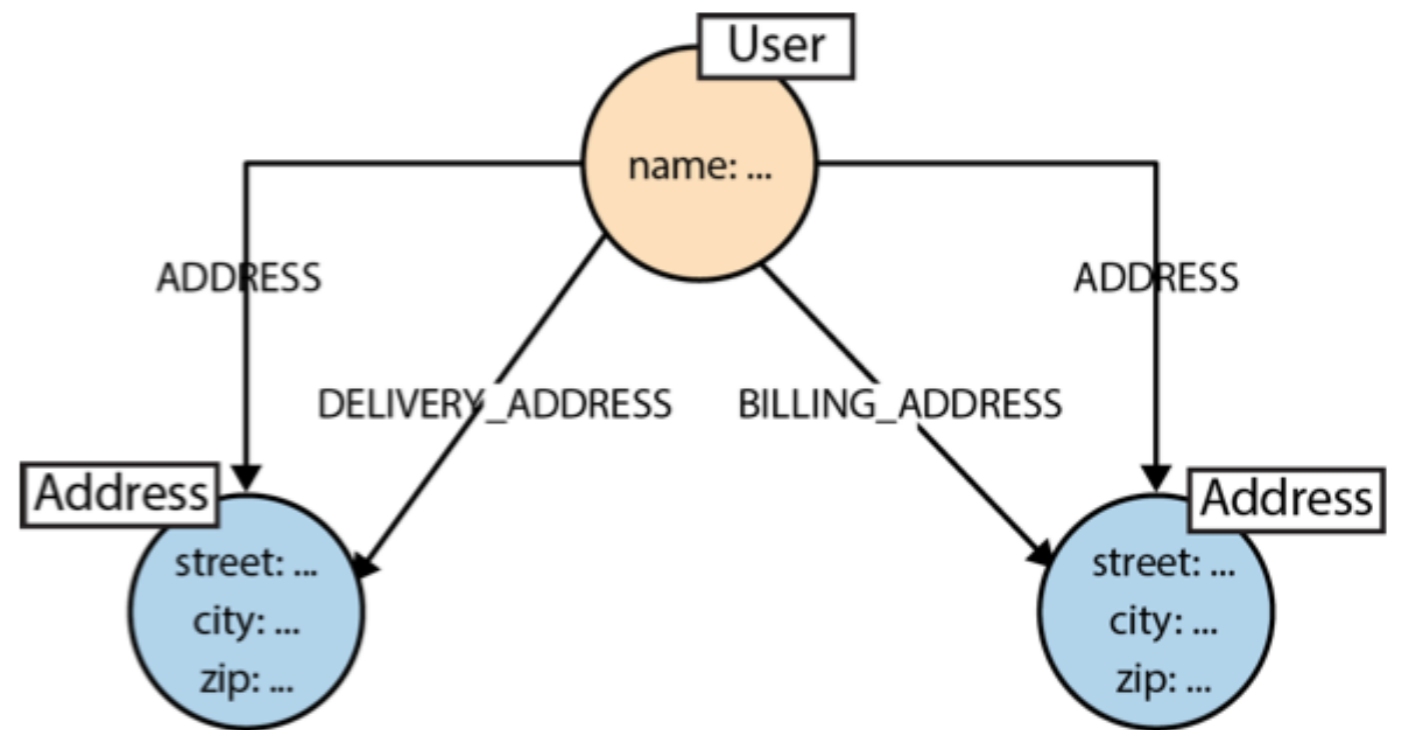
- Nodes represent **entities** — the *things* in the domain that are of interest to, and which can be labeled and grouped.
- Use node properties to represent **entity attributes**, plus any necessary **entity metadata**, such as timestamps, version numbers, etc.
- Represent a **fact** as a separate node with connections to each of the entities engaged in that fact. The intermediate node represents the outcome of an interaction between two or more entities.

Nodes or relationships?

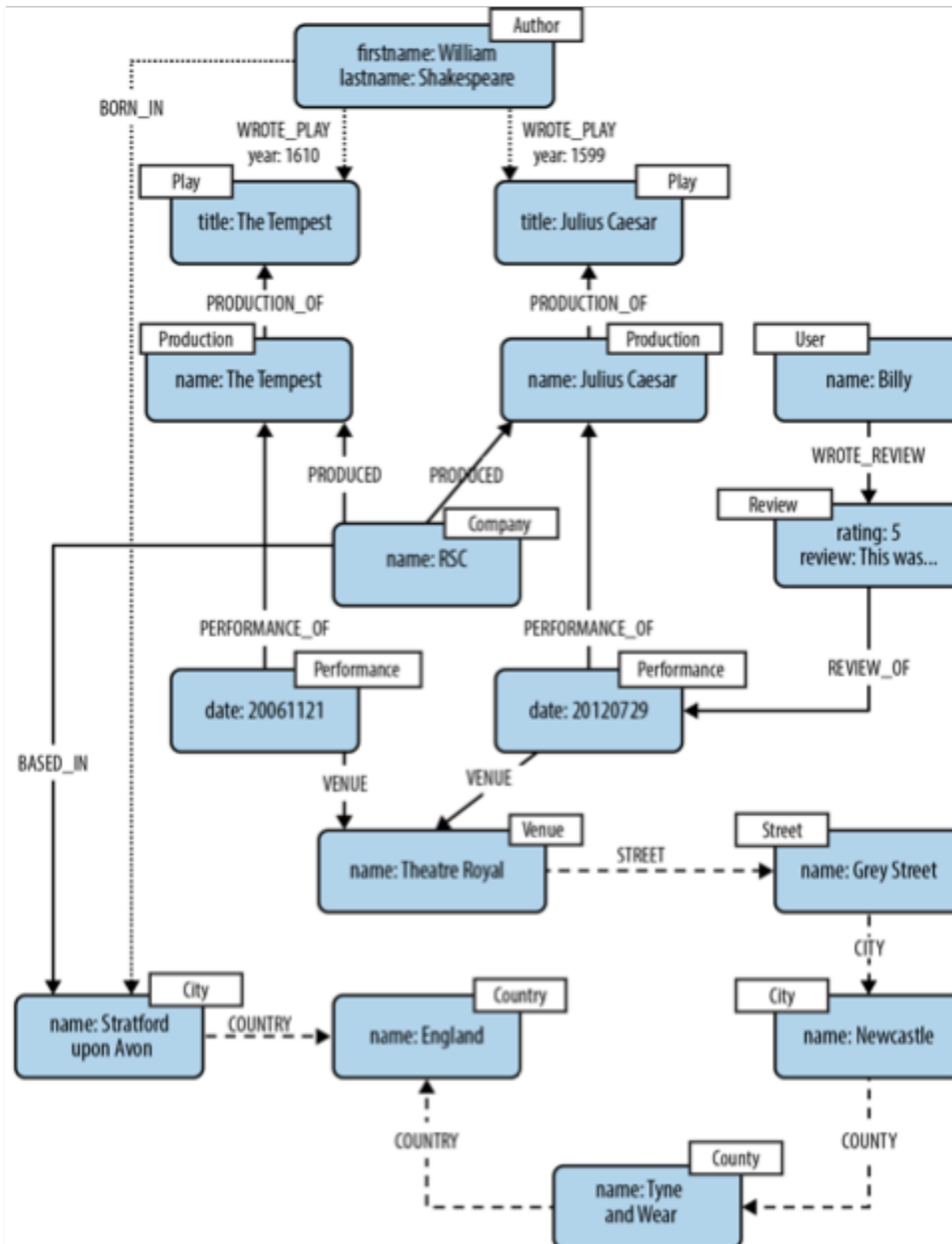
- Relationships express the *connections* between entities and to establish semantic context for each entity, thereby structuring the domain.
- Use **relationship direction** to further clarify relationship semantics. Many relationships are asymmetrical, which is why relationships in a property graph are always directed. For **bidirectional relationships**, we should make our queries ignore direction, rather than using two relationships.
- Use **relationship properties** to express the strength, weight, or quality of a relationship, plus any necessary relationship metadata, such as timestamps, version numbers, etc.

Relationship Grain

- Fine-grained relationships whenever closed set of names (e.g. addresses)
- Use of “generic relationships” with properties to distinguish them make queries slower
- Sometimes needed generic and fine-grained relationships (more on this later on the course)

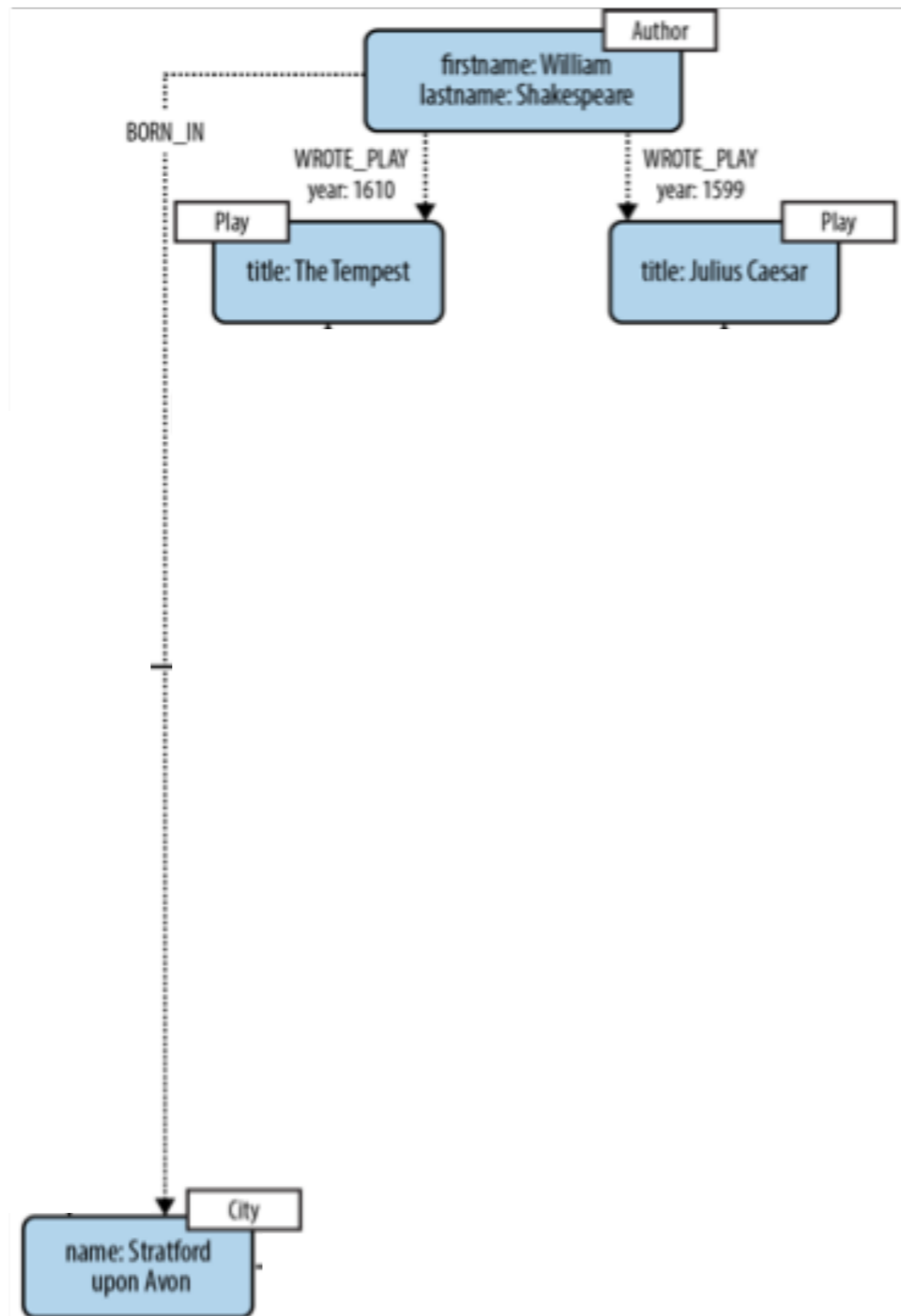


Cross-Domain Models



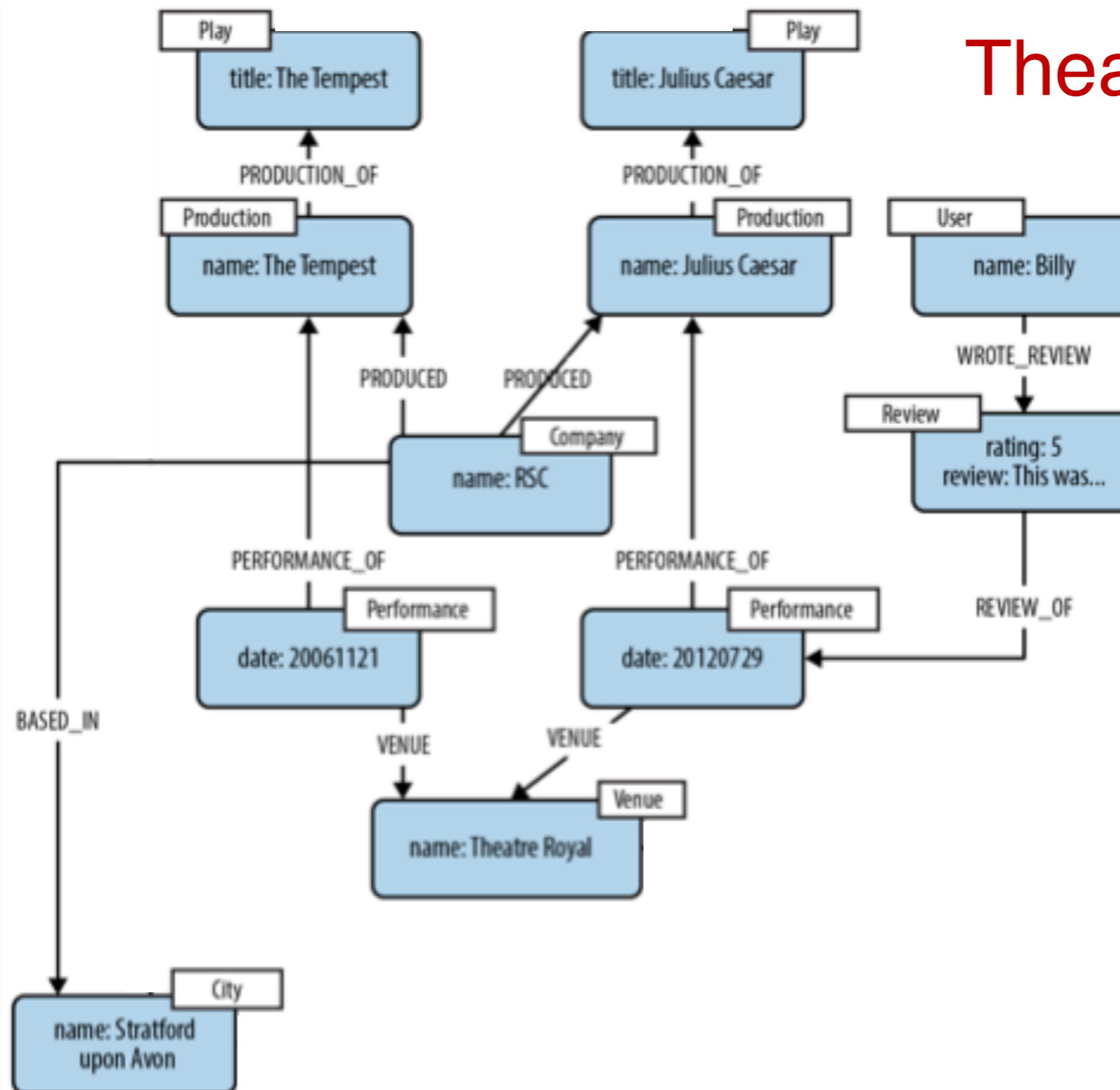
Relationships help both partition a graph into separate domains *and* connect those domains.

Cross-Domain Models



Literary domain

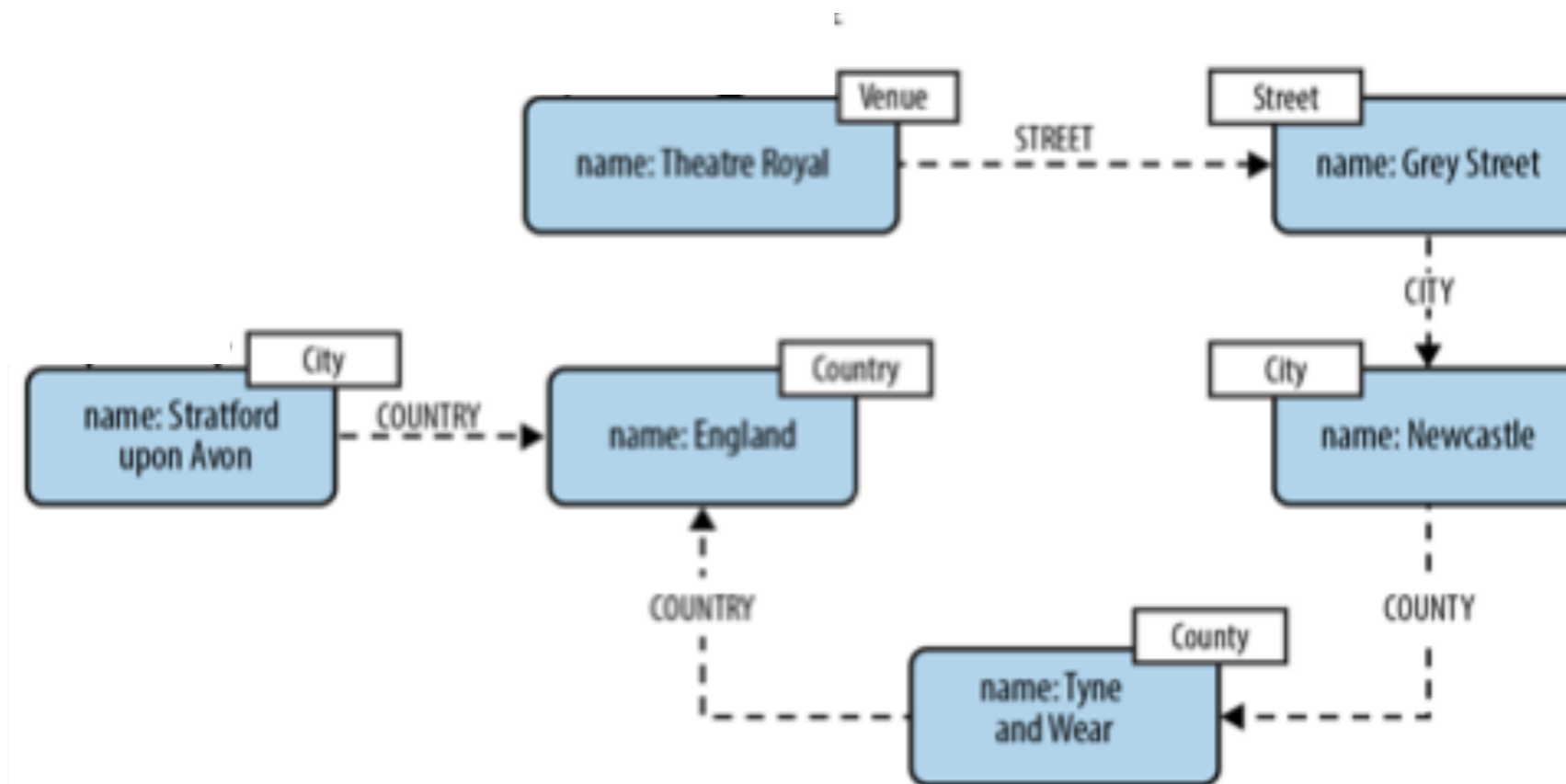
Cross-Domain Models



Theatrical domain

Cross-Domain Models

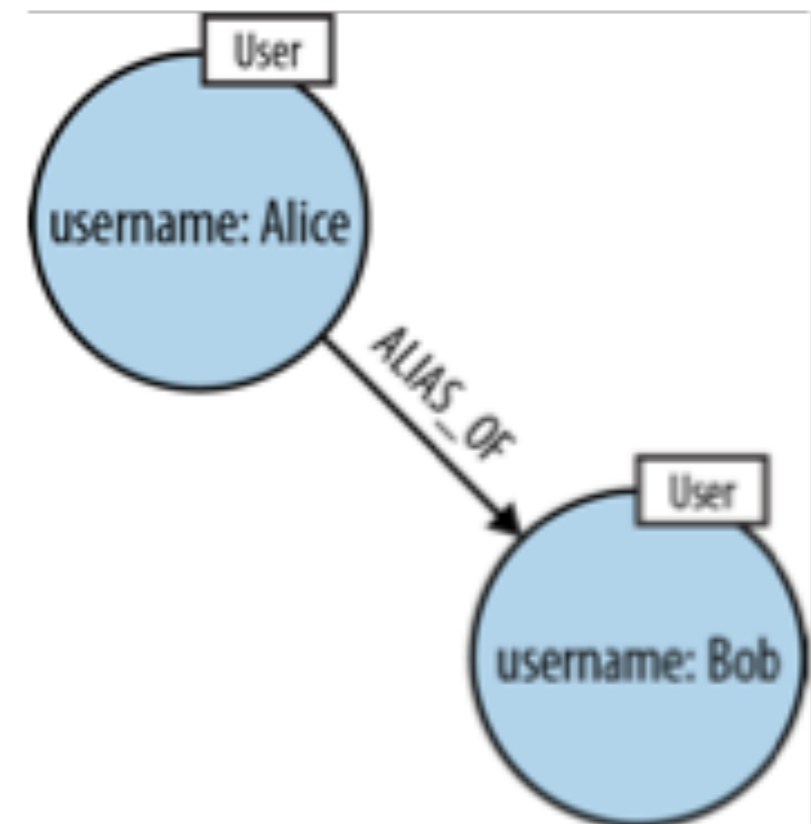
Geospatial domain



Common Modeling Pitfalls

- Model communications – Email

```
CREATE (alice:User {username:'Alice'}),  
      (bob:User {username:'Bob'}),  
      (charlie:User {username:'Charlie'}),  
      (davina:User {username:'Davina'}),  
      (edward:User {username:'Edward'}),  
      (alice)-[:ALIAS_OF]->(bob)
```

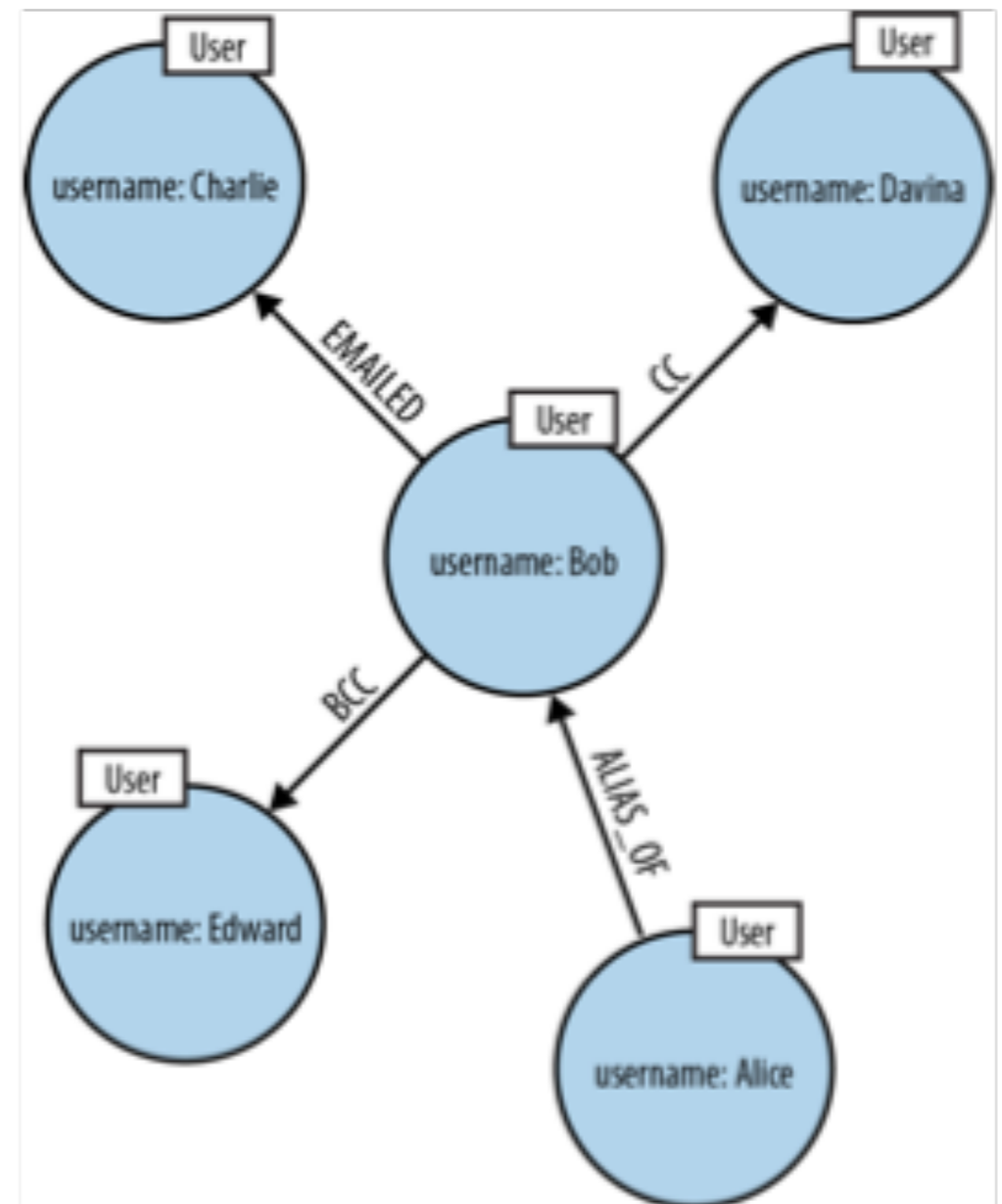


Common Modeling Pitfalls

- Join the users together through the emails they've exchanged

```
MATCH (bob:User {username:'Bob'}),  
      (charlie:User {username:'Charlie'}),  
      (davina:User {username:'Davina'}),  
      (edward:User {username:'Edward'})  
CREATE (bob)-[:EMAILED]->(charlie),  
       (bob)-[:CC]->(davina),  
       (bob)-[:BCC]->(edward)
```

the most critical element
of the data, the actual
email, is missing



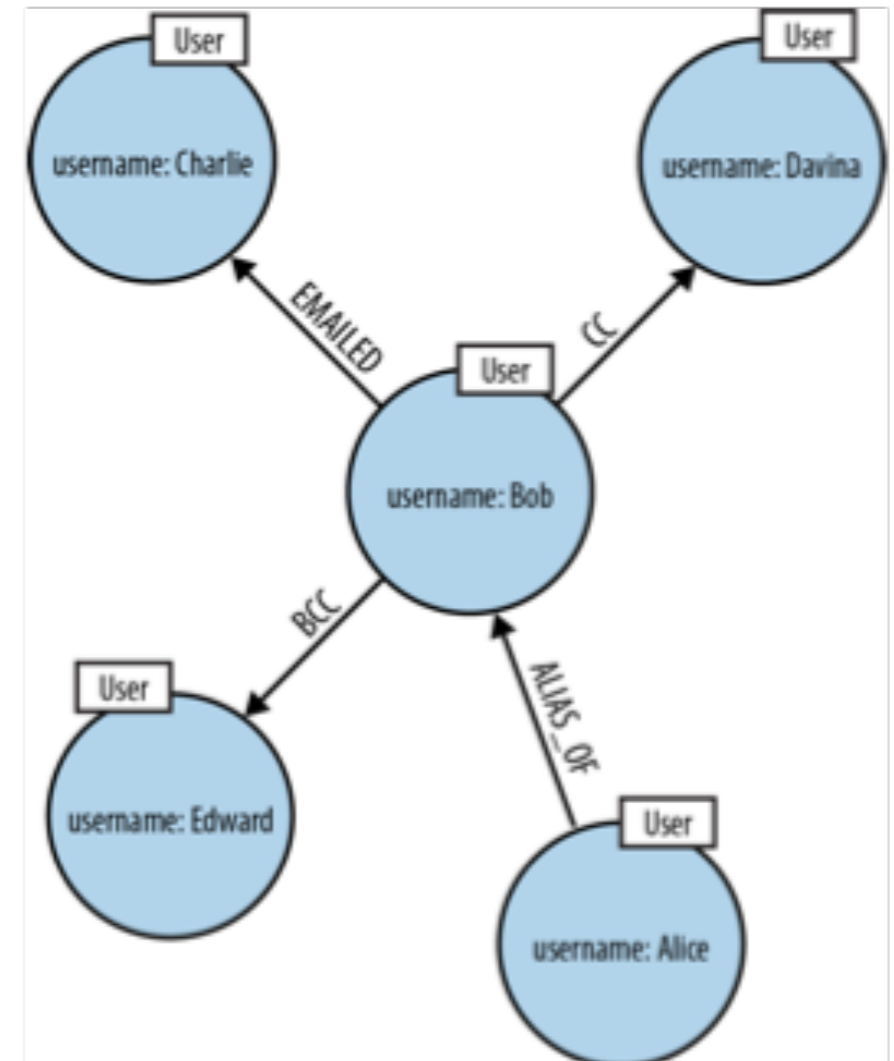
Common Modeling Pitfalls

- This query returns the EMAILED relationships between Bob and Charlie (there will likely be one for each email that Bob has sent to Charlie)

```
MATCH (bob:User {username:'Bob'})-[e:EMAILED]->
      (charlie:User {username:'Charlie'})
RETURN e
```

- This tells us that emails have been exchanged, but it tells us nothing about the emails themselves:

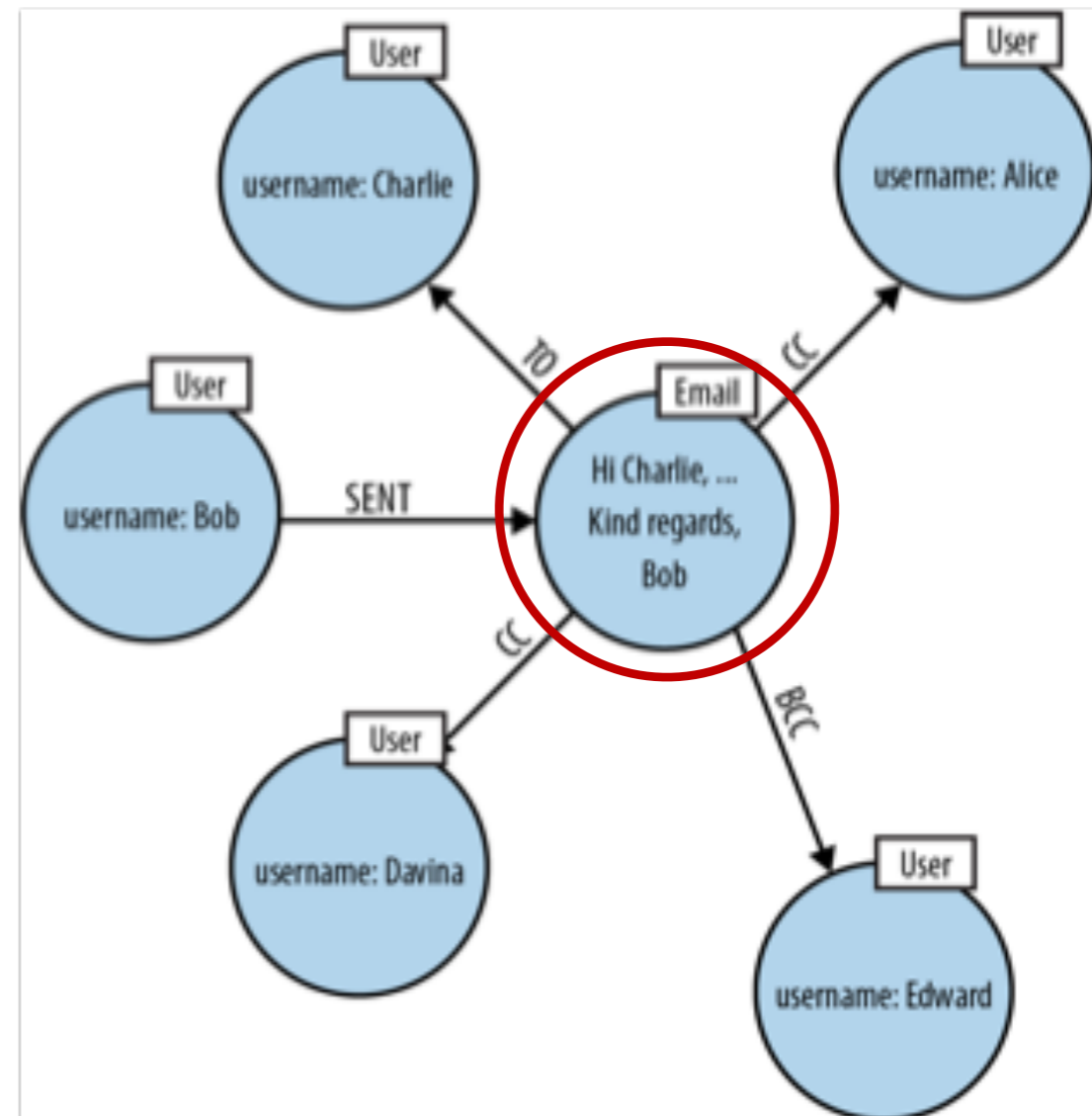
```
+-----+
| e      |
+-----+
| :EMAILED[1] {} |
+-----+
1 row
```



Common Modeling Pitfalls

- We need to insert **email nodes** to represent the real emails exchanged within the business

```
CREATE (email_1:Email {id:'1', content:'Hi Charlie, ... Kind regards, Bob'}),  
      (bob)-[:SENT]->(email_1),  
      (email_1)-[:TO]->(charlie),  
      (email_1)-[:CC]->(davina),  
      (email_1)-[:CC]->(alice),  
      (email_1)-[:BCC]->(edward)
```



Common Modeling Pitfalls

- We need to insert **email nodes** to represent the real emails exchanged within the business

```
CREATE (email_1:Email {id:'1', content:'email contents'}),
      (bob)-[:SENT]->(email_1),
      (email_1)-[:TO]->(charlie),
      (email_1)-[:CC]->(davina),
      (email_1)-[:CC]->(alice),
      (email_1)-[:BCC]->(edward);
```

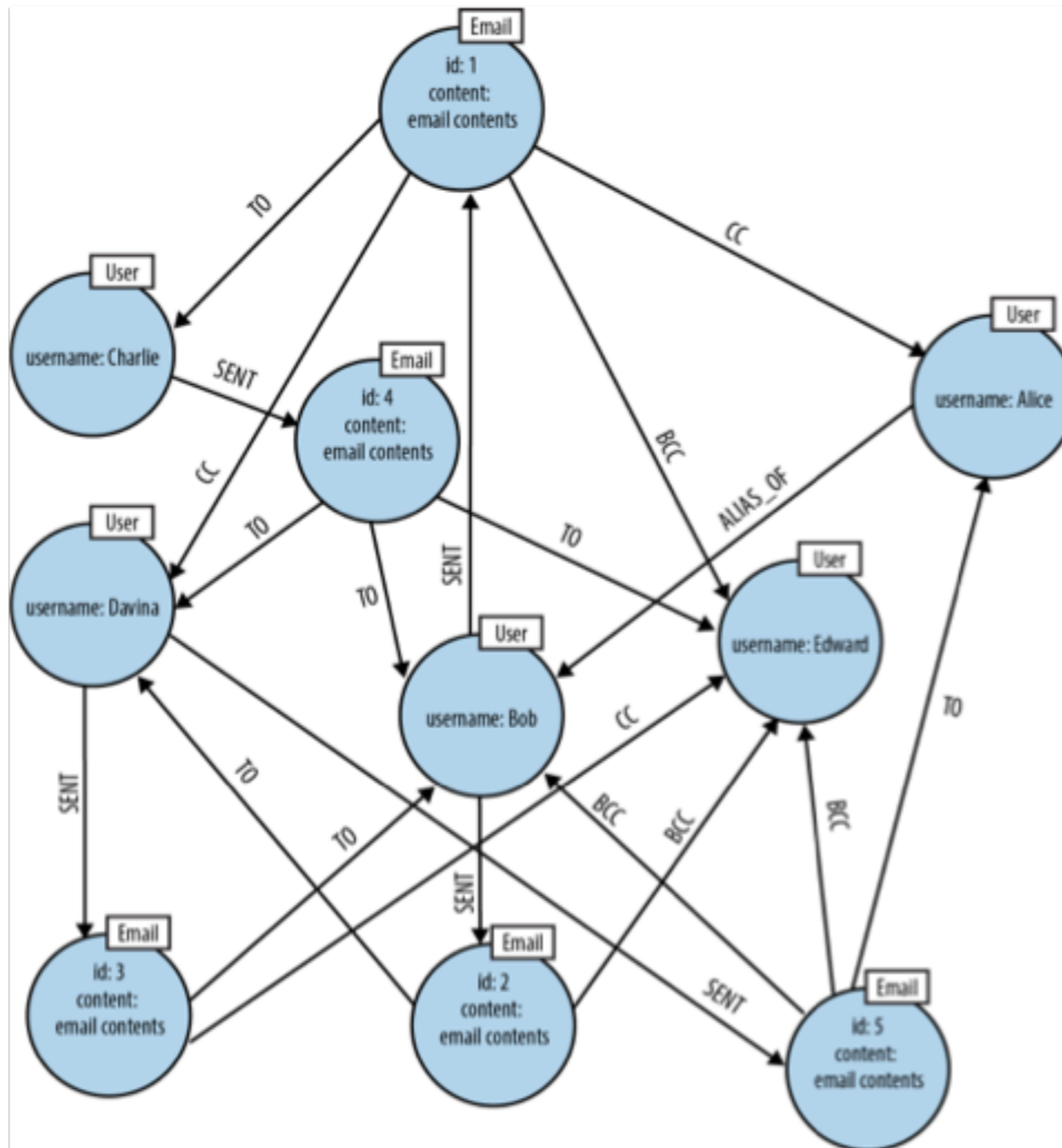
```
CREATE (email_2:Email {id:'2', content:'email contents'}),
      (bob)-[:SENT]->(email_2),
      (email_2)-[:TO]->(davina),
      (email_2)-[:BCC]->(edward);
```

```
CREATE (email_3:Email {id:'3', content:'email contents'}),
      (davina)-[:SENT]->(email_3),
      (email_3)-[:TO]->(bob),
      (email_3)-[:CC]->(edward);
```

```
CREATE (email_4:Email {id:'4', content:'email contents'}),
      (charlie)-[:SENT]->(email_4),
      (email_4)-[:TO]->(bob),
      (email_4)-[:TO]->(davina),
      (email_4)-[:TO]->(edward);
```

```
CREATE (email_5:Email {id:'5', content:'email contents'}),
      (davina)-[:SENT]->(email_5),
      (email_5)-[:TO]->(alice),
      (email_5)-[:BCC]->(bob),
      (email_5)-[:BCC]->(edward);
```

Common Modeling Pitfalls



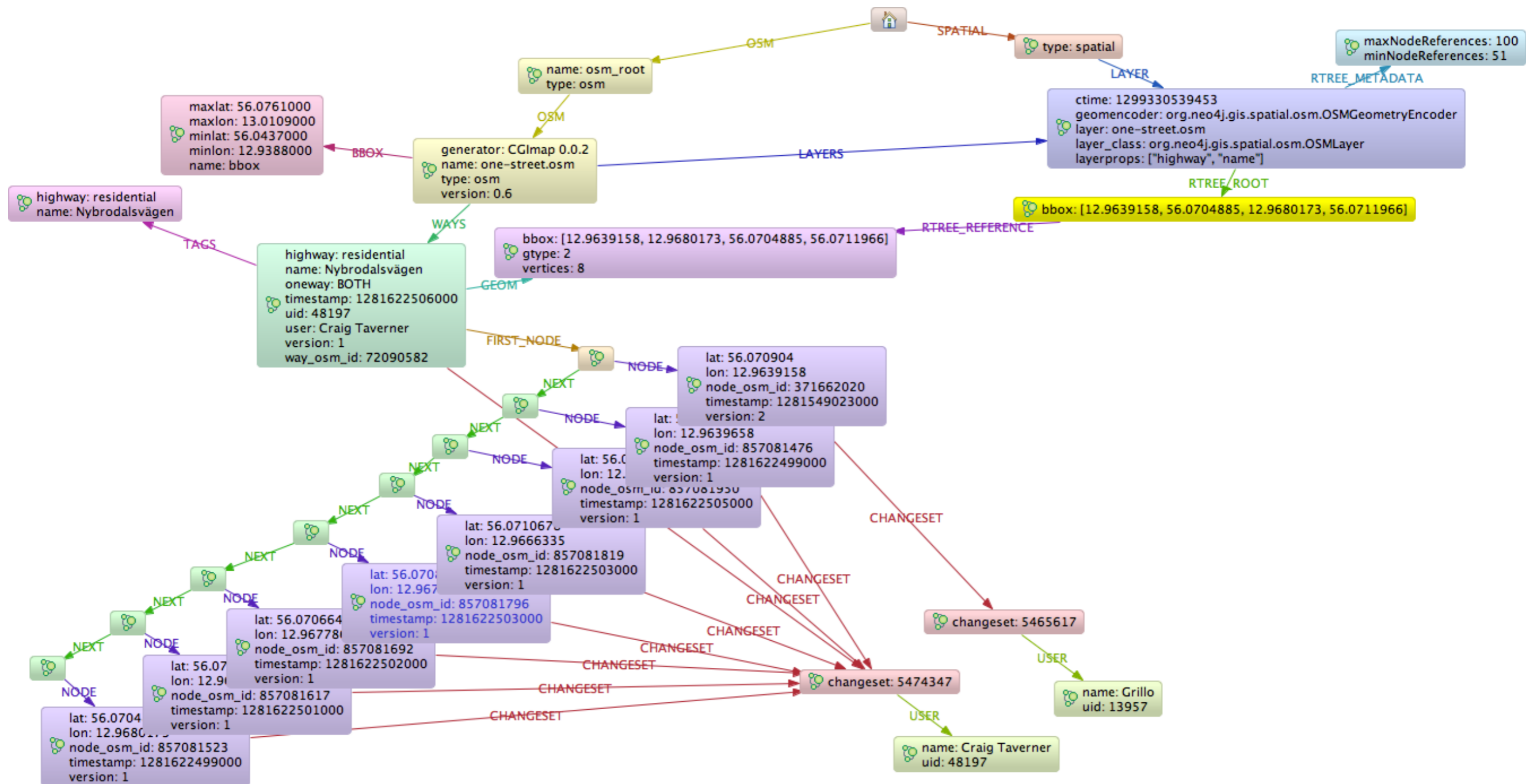
Common Modeling Pitfalls

- Retrieve all the emails that Bob has sent where he's CC'd one of his own aliases.

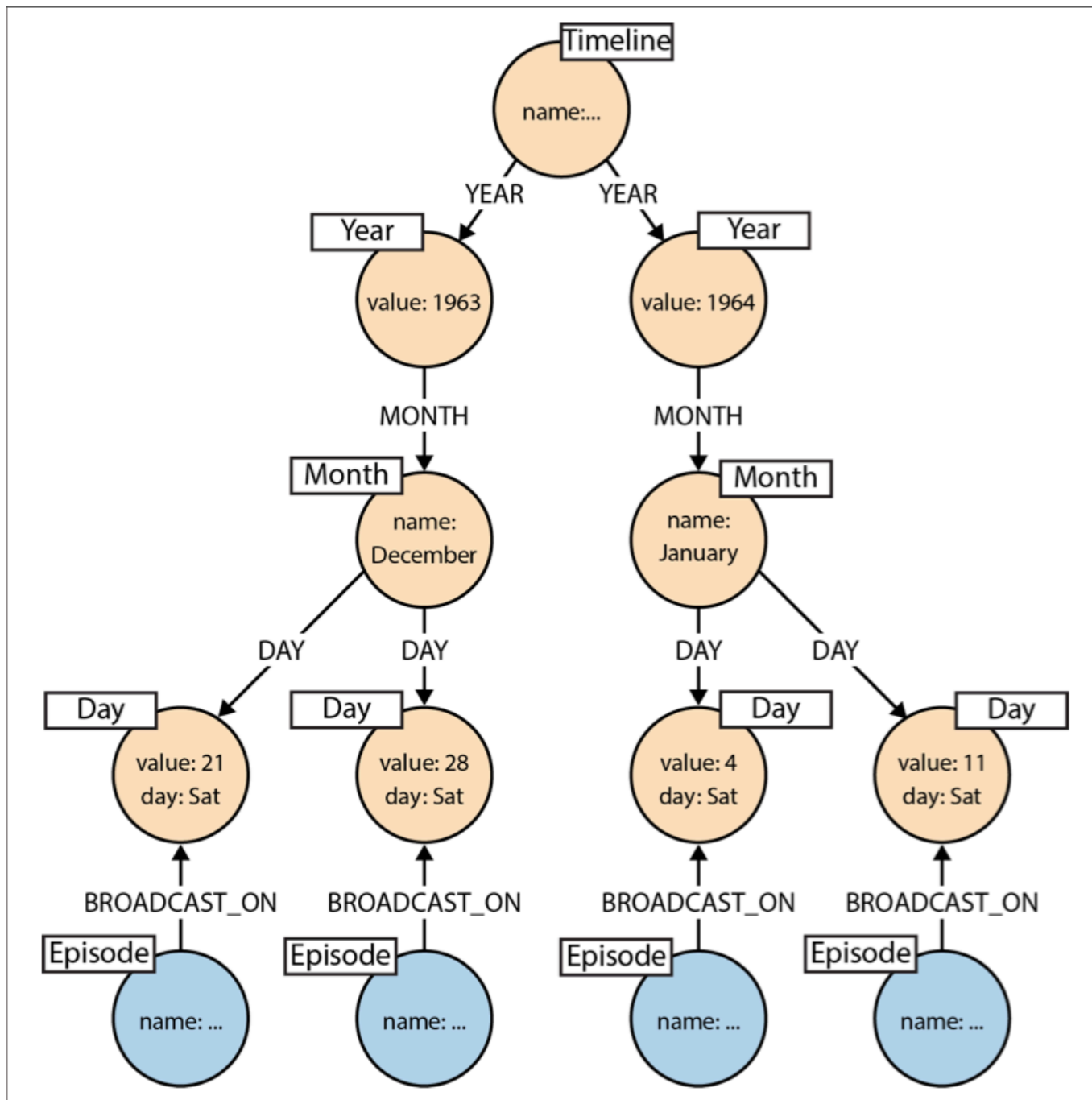
```
MATCH (bob:User {username:'Bob'})-[:SENT]->(email)-[:CC]->(alias),
      (alias)-[:ALIAS_OF]->(bob)
RETURN email.id
```

```
+-----+
| email |
+-----+
| Node[6]{id:"1",content:"email contents"} |
+-----+
1 row
```

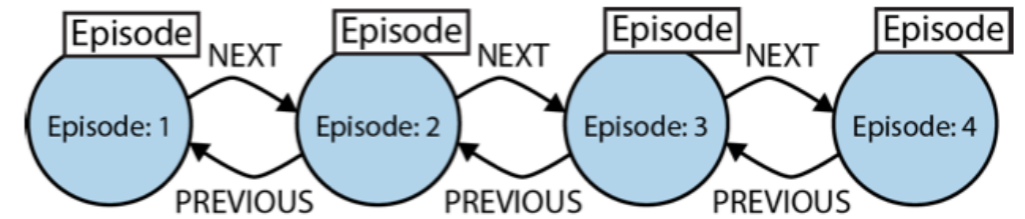

Geospatial data



Time



Timelines



Linked lists

Querying Graphs

Neo4j Cypher Query Language

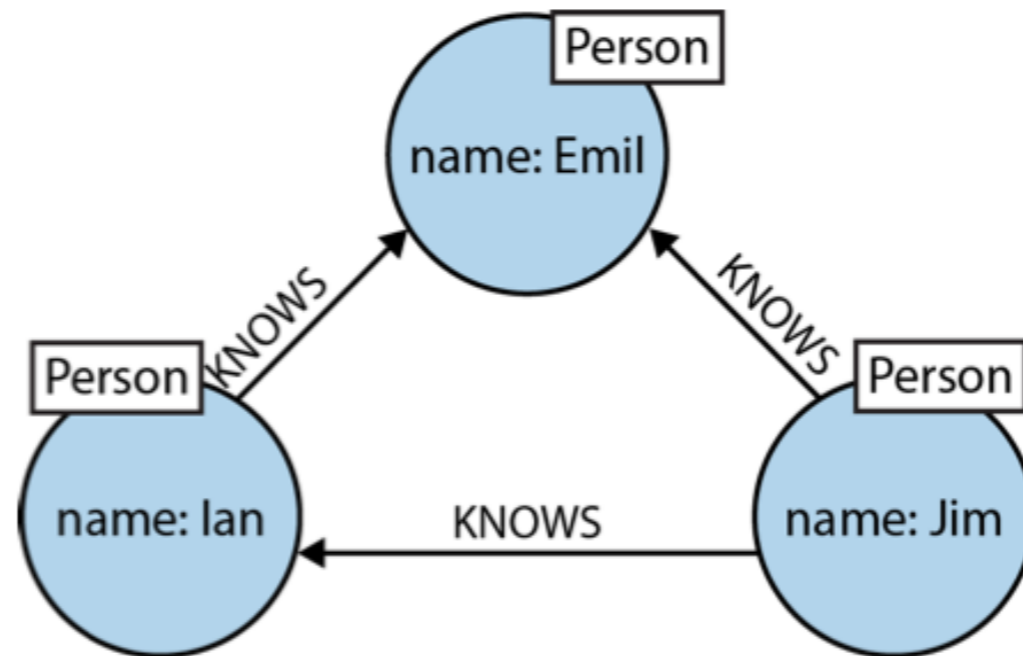
Cypher

- The Query Language of Neo4j, with an easy to use and natural syntax
- Uses patterns to match and traverse the graph
- Typically, one uses anchor points to start querying (maybe using indexes to find them easily) and navigate from these anchor points
- MATCH ... WHERE ... RETURN instructions

Other Query Languages

- **Cypher** is the Query Language of Neo4j, with an easy to use and natural syntax
- **SPARQL** - the RDF query language
- **Gremlin** - imperative, path-based query language

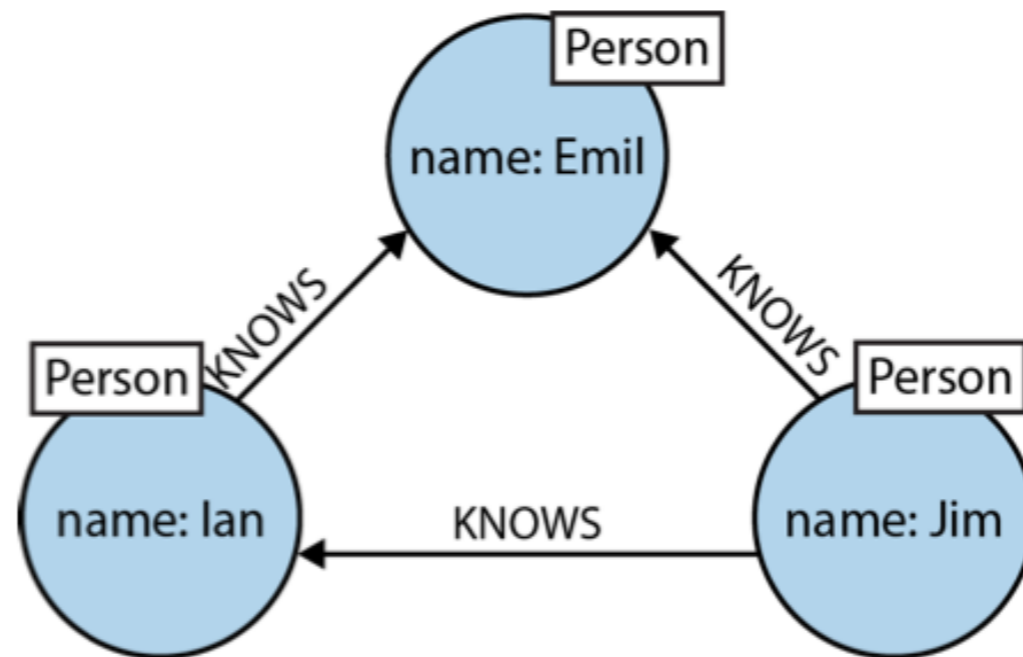
ASCII art graph patterns



`(e) <-[:KNOWS]-(j)-[:KNOWS]->(i)-[:KNOWS]->(e)`

```
(emil:Person {name: 'Emil'})  
<-[:KNOWS]-(jim:Person {name: 'Jim'})  
-[:KNOWS]->(ian:Person {name: 'Ian'})  
-[:KNOWS]->(emil)
```

MATCH and RETURN



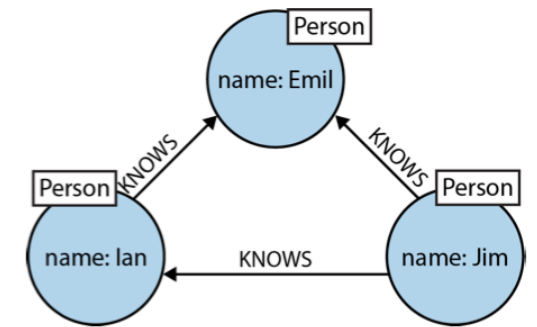
Find the mutual friends of a user named Jim:

MATCH (a:Person {name: 'Jim'}) -[:KNOWS]->(b) -[:KNOWS]->(c),
(a) -[:KNOWS]->(c)

RETURN b, c

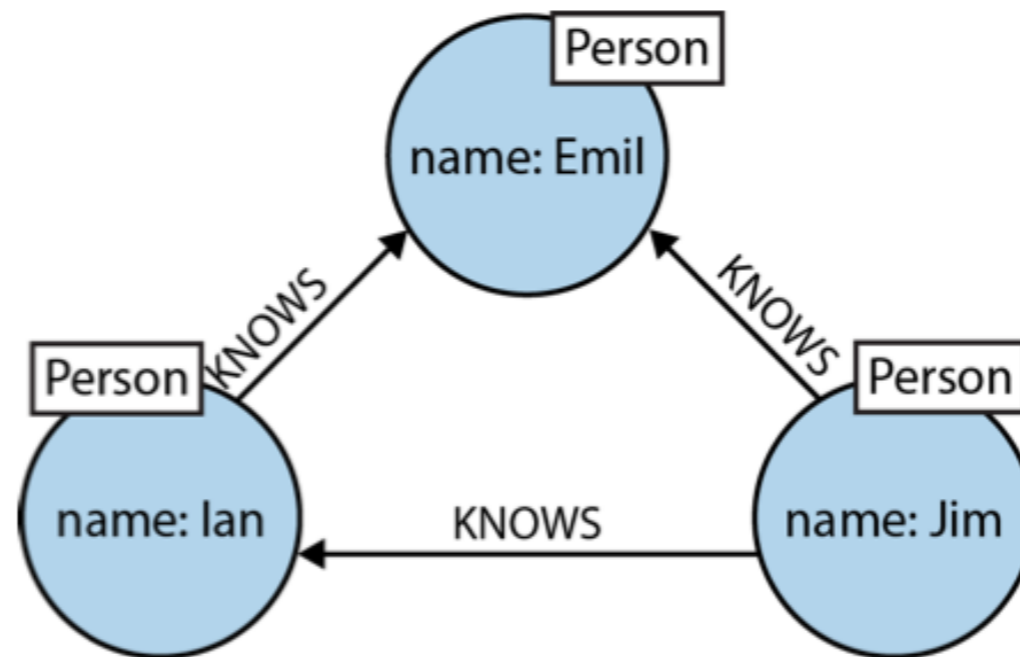
MATCH and RETURN

```
MATCH (a:Person {name:'Jim'})-[:KNOWS]->(b)-[:KNOWS]->(c),  
      (a)-[:KNOWS]->(c)  
RETURN b, c
```



1. looking for a node labeled **Person** with a name property whose value is **Jim**. The return value from this lookup is bound to the identifier **a**
2. A simple pattern
 $(a)-[:KNOWS]->(b)-[:KNOWS]->(c), (a)-[:KNOWS]->(c)$
that describes a path comprising three nodes
3. Cypher then matches the remainder of the pattern to the graph immediately surrounding this anchor point. As it does so, it discovers nodes to bind to the other identifiers. While **a** will always be anchored to **Jim**, **b** and **c** will be bound to a sequence of nodes as the query executes.

WHERE clause



Find the mutual friends of a user named Jim:

```
MATCH (a:Person)-[:KNOWS]->(b)-[:KNOWS]->(c), (a)-[:KNOWS]->(c)  
WHERE a.name = 'Jim'  
RETURN b, c
```

```
MATCH (a:Person {name:'Jim'})-[:KNOWS]->(b)-[:KNOWS]->(c),  
      (a)-[:KNOWS]->(c)  
RETURN b, c
```

Other Clauses

CREATE *and* CREATE UNIQUE

Create nodes and relationships.

MERGE

Ensures that the supplied pattern exists in the graph, either by reusing existing nodes and relationships that match the supplied predicates, or by creating new nodes and relationships.

DELETE

Removes nodes, relationships, and properties.

SET

Sets property values.

FOREACH

Performs an updating action for each element in a list.

UNION

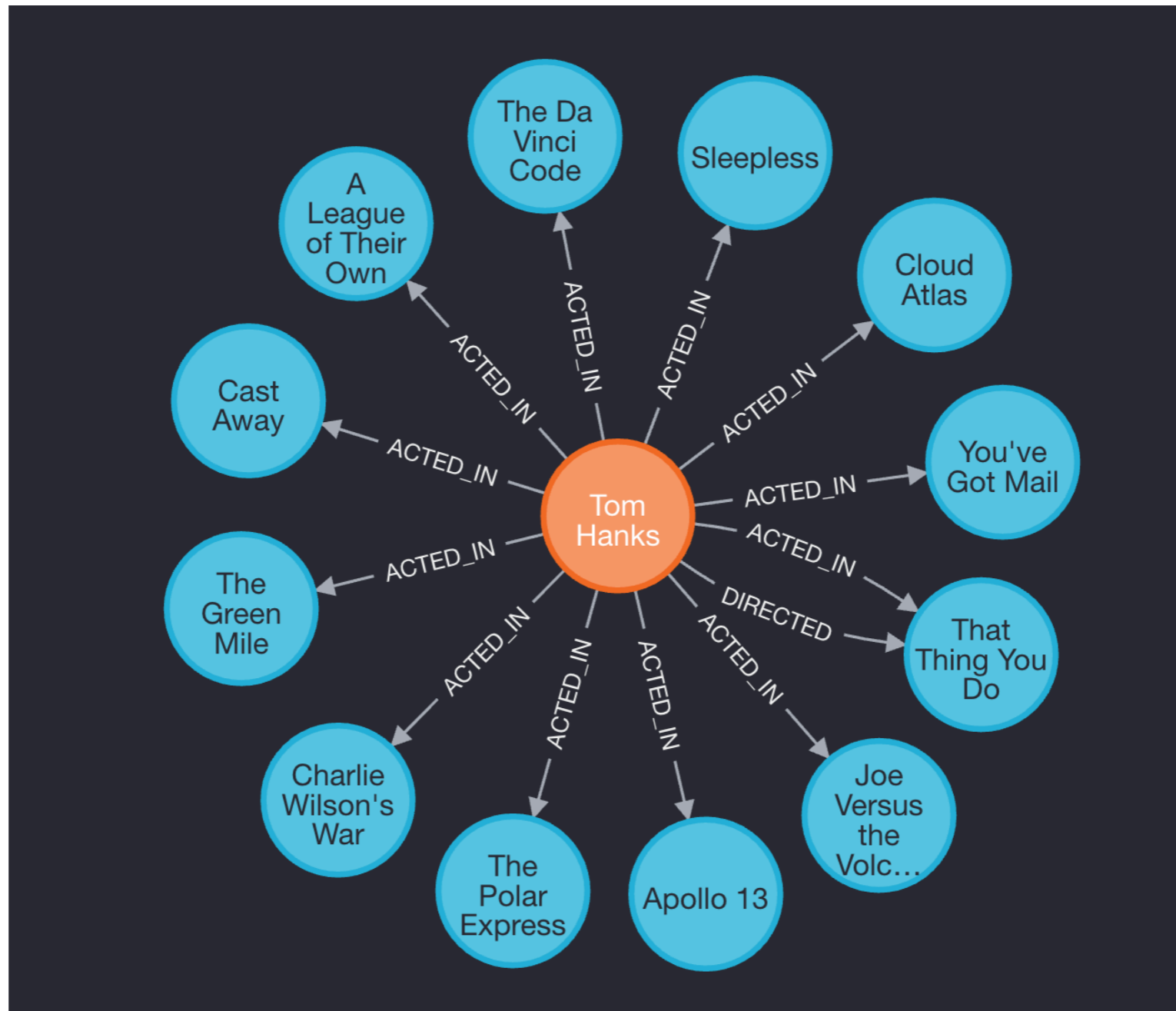
Merges results from two or more queries.

WITH

Chains subsequent query parts and forwards results from one to the next. Similar to piping commands in Unix.

Playing with Neo4J

```
MATCH (p:Person)-->(m:Movie) WHERE p.name = 'Tom Hanks' RETURN p, m
```



Playing with Neo4J

```
MATCH (p:Person)-->(m:Movie) WHERE p.name = 'Tom Hanks' RETURN p, m
```

"p"	"m"
{"name":"Tom Hanks","born":1956}	{"tagline":"At odds in life... in love on-line.,"title":"You've Got M ail","released":1998}
{"name":"Tom Hanks","born":1956}	{"tagline":"Houston, we have a problem.,"title":"Apollo 13","released ":1995}
{"name":"Tom Hanks","born":1956}	{"tagline":"A story of love, lava and burning desire.,"title":"Joe Ve rsus the Volcano","released":1990}
{"name":"Tom Hanks","born":1956}	{"tagline":"In every life there comes a time when that thing you dream becomes that thing you do","title":"That Thing You Do","released":199 6}
{"name":"Tom Hanks","born":1956}	{"tagline":"Everything is connected","title":"Cloud Atlas","released": 2012}
{"name":"Tom Hanks","born":1956}	{"tagline":"Break The Codes","title":"The Da Vinci Code","released":20 06}
{"name":"Tom Hanks","born":1956}	{"tagline":"What if someone you never met, someone you never saw, some one you never knew was the only someone for you?","title":"Sleepless i n Seattle","released":1993}
{"name":"Tom Hanks","born":1956}	{"tagline":"Once in a lifetime you get a chance to do something differ ent.,"title":"A League of Their Own","released":1992}
{"name":"Tom Hanks","born":1956}	{"tagline":"Walk a mile you'll never forget.,"title":"The Green Mile" ,"released":1999}
{"name":"Tom Hanks","born":1956}	{"tagline":"A stiff drink. A little mascara. A lot of nerve. Who said they couldn't bring down the Soviet empire.,"title":"Charlie Wilson's War","released":2007}
{"name":"Tom Hanks","born":1956}	{"tagline":"At the edge of the world, his journey begins.,"title":"Ca st Away","released":2000}
{"name":"Tom Hanks","born":1956}	{"tagline":"In every life there comes a time when that thing you dream becomes that thing you do","title":"That Thing You Do","released":199 6}
{"name":"Tom Hanks","born":1956}	{"tagline":"This Holiday Season... Believe","title":"The Polar Express", "released":2004}

MATCH (p:Person)-->(m:Movie) **WHERE** p.name = 'Tom Hanks' **RETURN** p, m

"p"	"m"
{"name":"Tom Hanks","born":1956}	{"tagline":"At odds in life... in love on-line.","title":"You've Got M ail","released":1998}
{"name":"Tom Hanks","born":1956}	{"tagline":"Houston, we have a problem.","title":"Apollo 13","released ":1995}
{"name":"Tom Hanks","born":1956}	{"tagline":"A story of love, lava and burning desire.","title":"Joe Ve rsus the Volcano","released":1990}
{"name":"Tom Hanks","born":1956}	{"tagline":"In every life there comes a time when that thing you dream becomes that thing you do","title":"That Thing You Do","released":199 6}
{"name":"Tom Hanks","born":1956}	{"tagline":"Everything is connected","title":"Cloud Atlas","released": 2012}
{"name":"Tom Hanks","born":1956}	{"tagline":"Break The Codes","title":"The Da Vinci Code","released":20 06}
{"name":"Tom Hanks","born":1956}	{"tagline":"What if someone you never met, someone you never saw, some one you never knew was the only someone for you?","title":"Sleepless i n Seattle","released":1993}
{"name":"Tom Hanks","born":1956}	{"tagline":"Once in a lifetime you get a chance to do something differ ent.","title":"A League of Their Own","released":1992}
{"name":"Tom Hanks","born":1956}	{"tagline":"Walk a mile you'll never forget.","title":"The Green Mile" ,"released":1999}
{"name":"Tom Hanks","born":1956}	{"tagline":"A stiff drink. A little mascara. A lot of nerve. Who said they couldn't bring down the Soviet empire.","title":"Charlie Wilson's War","released":2007}
{"name":"Tom Hanks","born":1956}	{"tagline":"At the edge of the world, his journey begins.","title":"Ca st Away","released":2000}
{"name":"Tom Hanks","born":1956}	{"tagline":"In every life there comes a time when that thing you dream becomes that thing you do","title":"That Thing You Do","released":199 6}
{"name":"Tom Hanks","born":1956}	{"tagline":"This Holiday Season... Believe","title":"The Polar Express", "released":2004}

Playing with Neo4J

```
MATCH (p:Person)-->(m:Movie) WHERE p.name = 'Tom Hanks' RETURN p, m
```

p		m
1	<pre>{ "identity": 71, "labels": ["Person"], "properties": { "name": "Tom Hanks", "born": 1956 } }</pre>	<pre>{ "identity": 67, "labels": ["Movie"], "properties": { "tagline": "At odds in life ... in love on-line.", "title": "You've Got Mail", "released": 1998 } }</pre>
2	<pre>{ "identity": 71, "labels": ["Person"], "properties": { "name": "Tom Hanks", "born": 1956 } }</pre>	<pre>{ "identity": 142, "labels": ["Movie"], "properties": { "tagline": "Houston, we have a problem.", "title": "Apollo 13", "released": 1995 } }</pre>

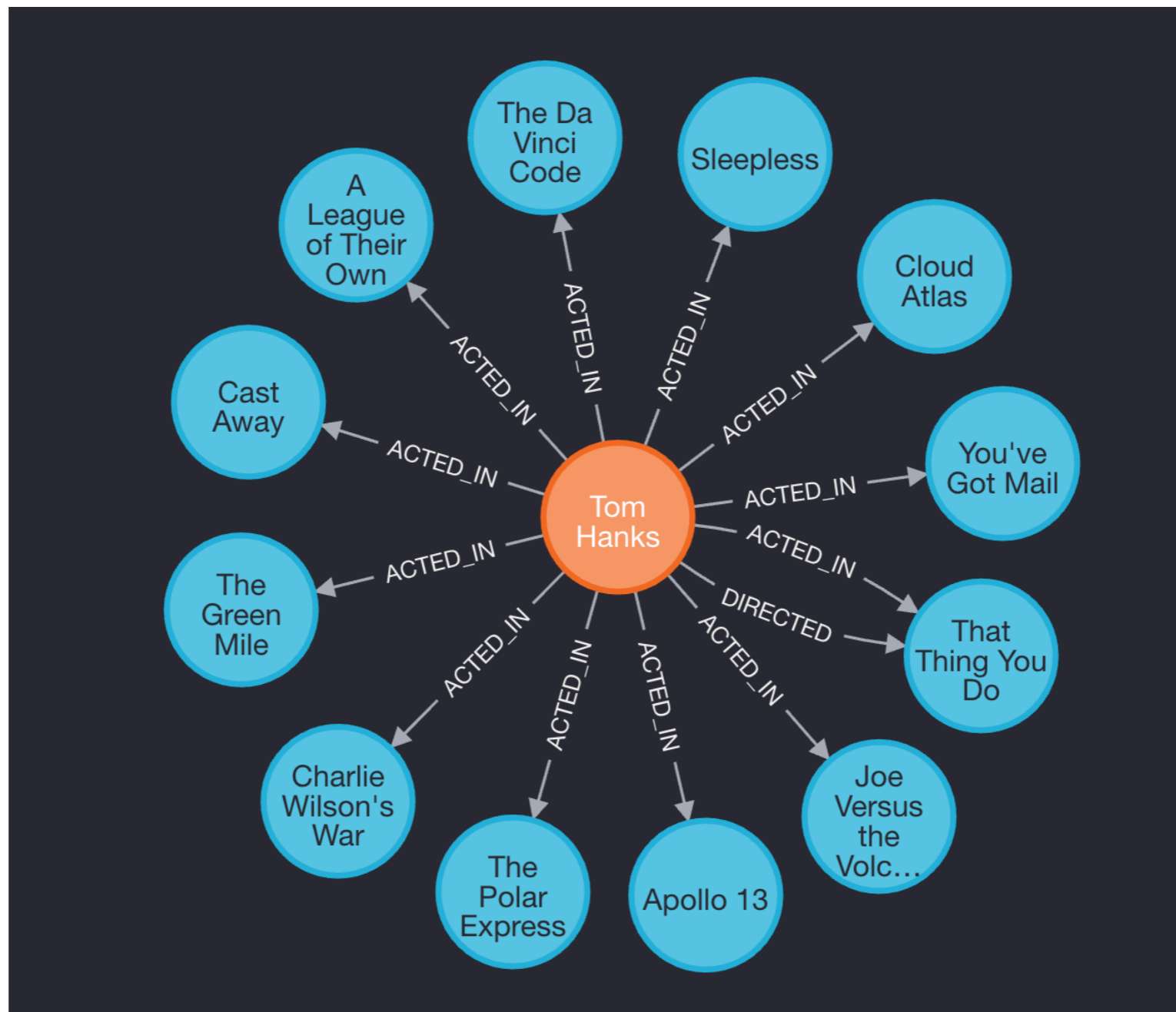
Playing with Neo4J

```
MATCH (p:Person)-->(m:Movie) WHERE p.name = 'Tom Hanks' RETURN m.title
```

	m.title
1	"You've Got Mail"
2	"Apollo 13"
3	"Joe Versus the Volcano"
4	"That Thing You Do"
5	"Cloud Atlas"
6	"The Da Vinci Code"

Playing with Neo4J

```
MATCH (p:Person)-[rel]->(m:Movie) WHERE p.name = 'Tom Hanks' RETURN p,rel, m
```



Playing with Neo4J

```
MATCH (p:Person)-[rel]->(m:Movie) WHERE p.name = 'Tom Hanks' RETURN p,rel, m
```

p	rel	m
1 <pre>{ "identity": 71, "labels": ["Person"], "properties": { "name": "Tom Hanks", "born": 1956 } }</pre>	<pre>{ "identity": 84, "start": 71, "end": 67, "type": "ACTED_IN", "properties": { "roles": ["Joe Fox"] } }</pre>	<pre>{ "identity": 67, "labels": ["Movie"], "properties": { "tagline": "At odds in life... in love on-line.", "title": "You've Got Mail", "released": 1998 } }</pre>

Playing with Neo4J

```
MATCH (p:Person)-[rel]->(m:Movie) WHERE p.name = 'Tom Hanks' RETURN p,rel, m
```

"p"	"rel"	"m"
{"name":"Tom Hanks","born":1956}	{"roles":["Joe Fox"]}	{"tagline":"At odds in life... in love on-line.", "title":"You've Got Mail","released":1998}
{"name":"Tom Hanks","born":1956}	{"roles":["Jim Lovell"]}	{"tagline":"Houston, we have a problem.", "title":"Apollo 13","released":1995}
{"name":"Tom Hanks","born":1956}	{"roles":["Joe Banks"]}	{"tagline":"A story of love, lava and burning desire.", "title":"Joe Versus the Volcano","released":1990}
{"name":"Tom Hanks","born":1956}	{"roles":["Mr. White"]}	{"tagline":"In every life there comes a time when that thing you dream becomes that thing you do", "title":"That Thing You Do","released":1996}
{"name":"Tom Hanks","born":1956}	{"roles":["Zachry","Dr. Henry Goose","Isaac Sachs","Dermot Hoggins"]}	{"tagline":"Everything is connected", "title":"Cloud Atlas","released":2012}
{"name":"Tom Hanks","born":1956}	{"roles":["Dr. Robert Langdon"]}	{"tagline":"Break The Codes", "title":"The Da Vinci Code","released":2006}

More info

- <https://neo4j.com/docs/>
- <https://neo4j.com/docs/pdf/cypher-refcard-4.3.pdf>
- <https://neo4j.com/videos/>