Introduction to Graph Databases

Chicago Graph Database Meet-Up

Matt DeMarco

About Me

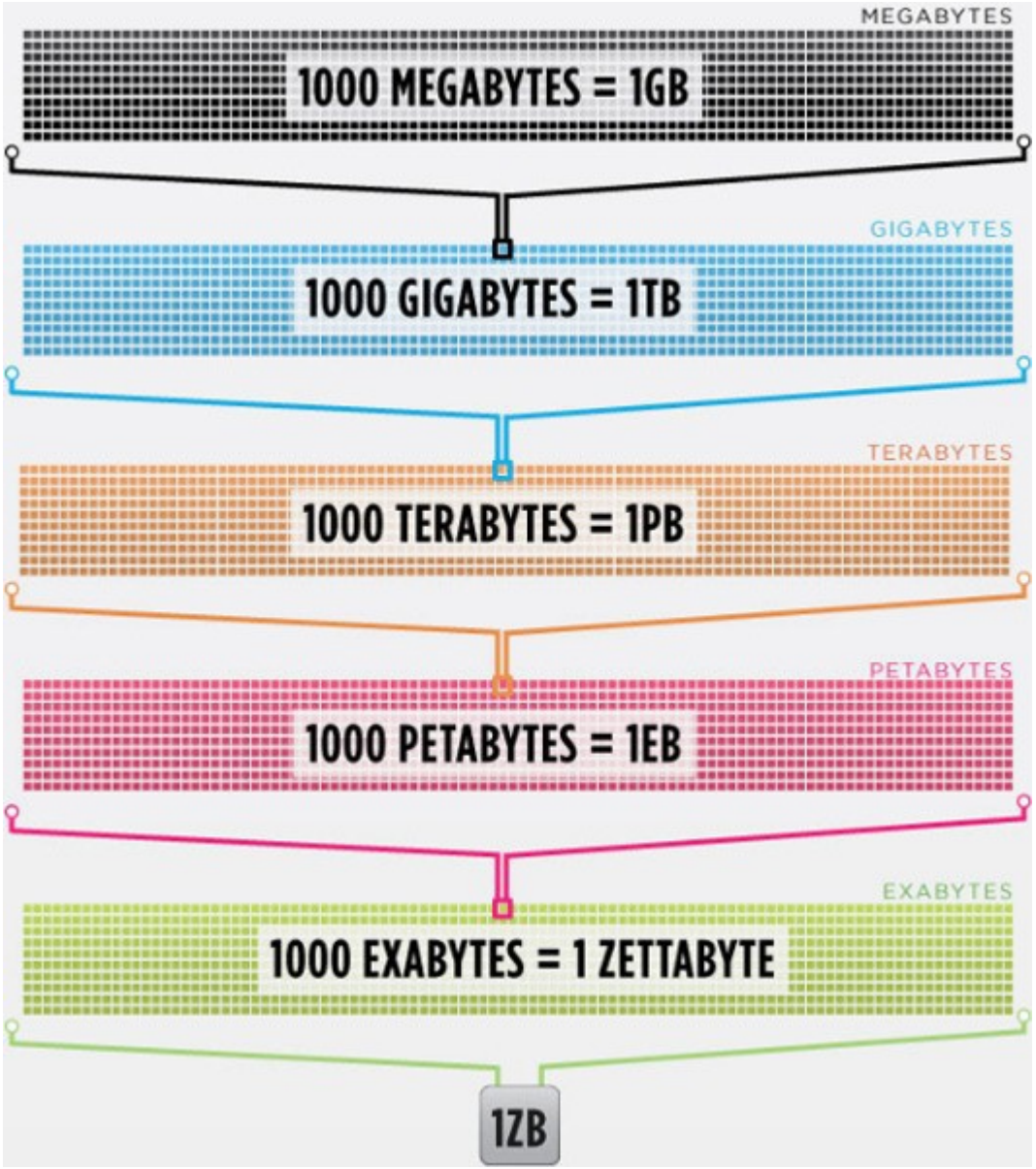
Built the Neography Gem (Ruby Wrapper to the Neo4j REST API)
Playing with Neo4j since 10/2009

- My Blog: http://maxdemarzi.com
- Find me on Twitter: @maxdemarzi
- Email me: maxdemarzi@gmail.com
- GitHub: http://github.com/maxdemarzi

Agenda

- Trends in Data
- NOSQL
- What is a Graph?
- What is a Graph Database?
- What is Neo4j?

Trends in Data



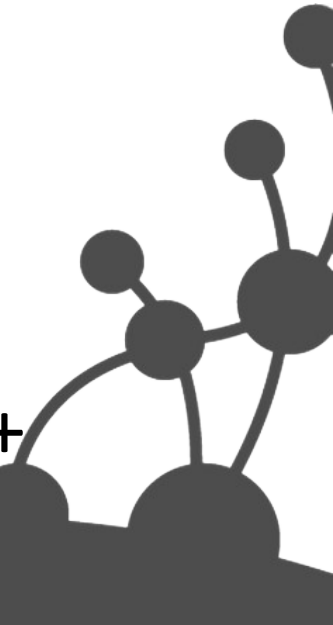
Data is getting bigger:

"Every 2 days we create as much information as we did up to 2003"

– Eric Schmidt, Google

Data is more connected:

- Text (content)
- HyperText (added pointers)
- RSS (joined those pointers)
- Blogs (added pingbacks)
- Tagging (grouped related data)
- RDF (described connected data)
- GGG (content + pointers + relationships + descriptions)

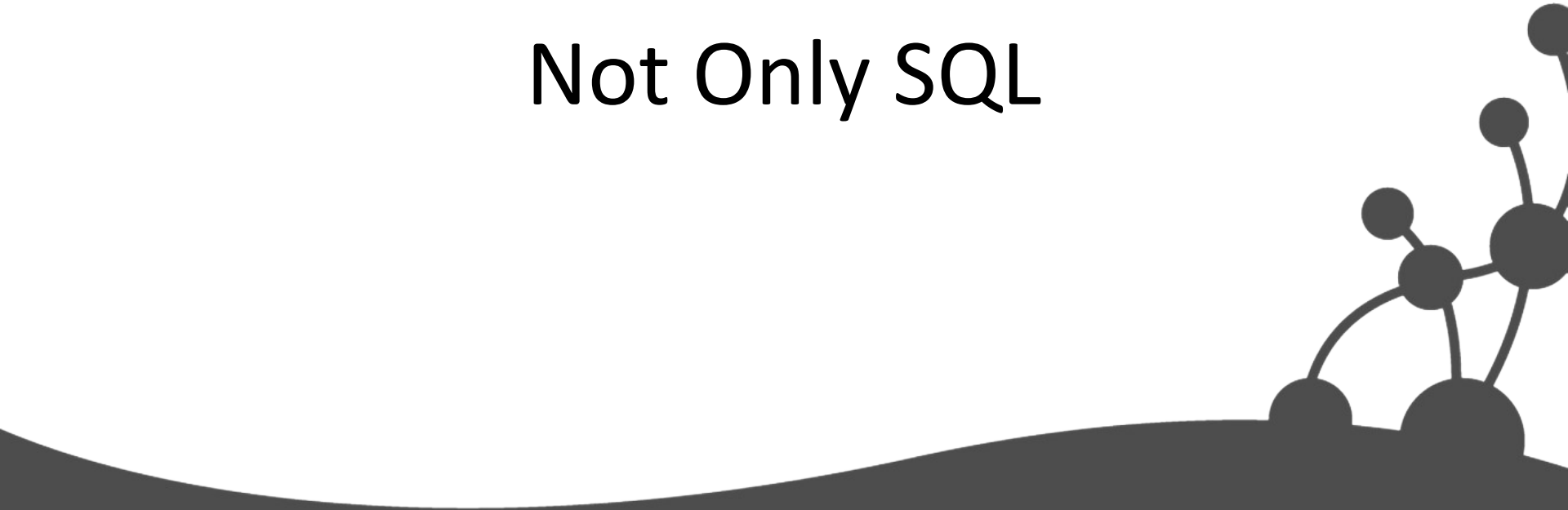


Data is more Semi-Structured:

- If you tried to collect all the data of every movie ever made, how would you model it?
- Actors, Characters, Locations, Dates, Costs, Ratings, Showings, Ticket Sales, etc.

NOSQL

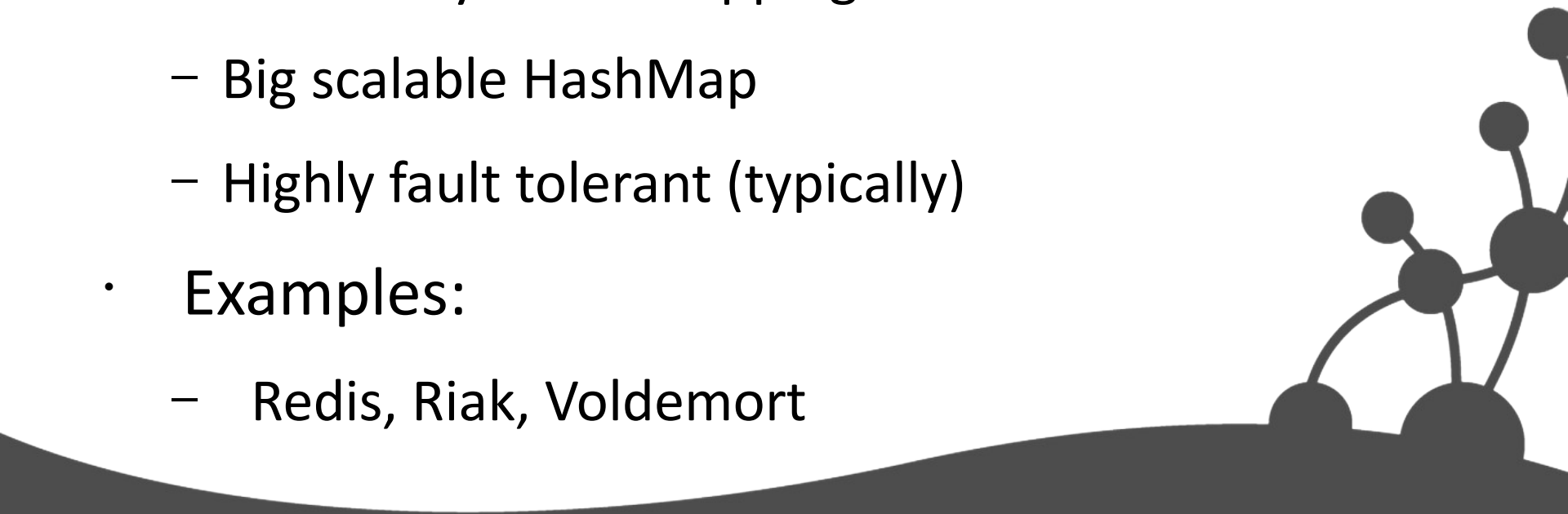
Not Only SQL



Less than 10% of the NOSQL Vendors

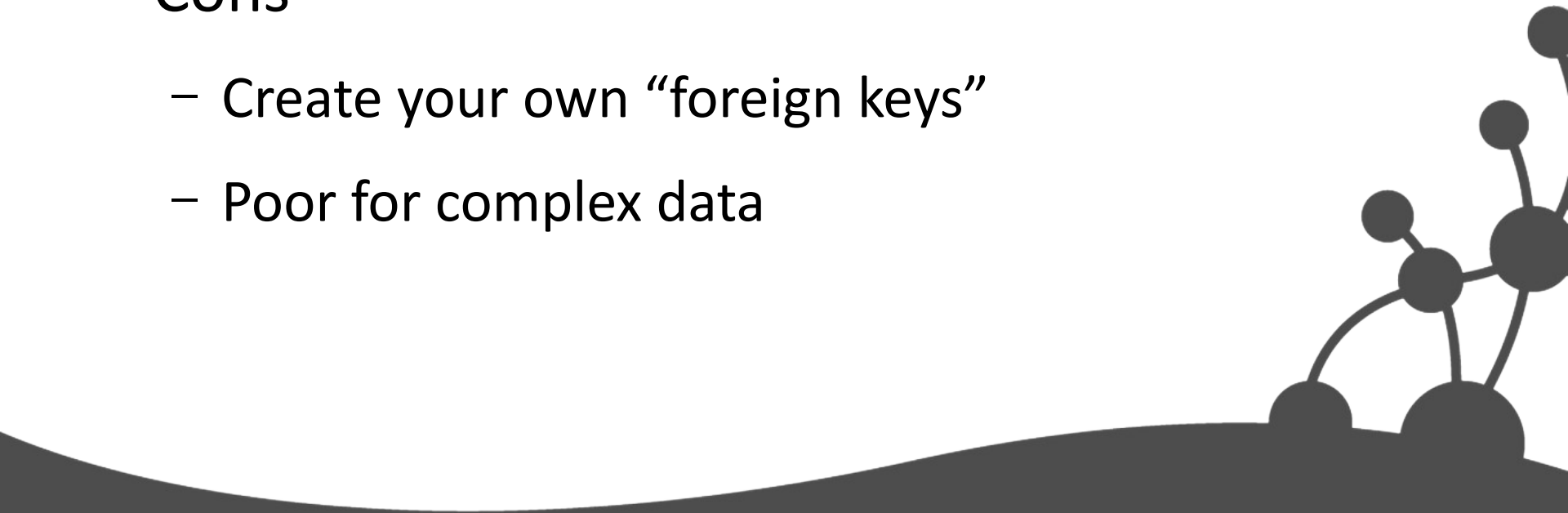
Key Value Stores

- Most Based on **Dynamo**: Amazon Highly Available Key-Value Store
- Data Model:
  - Global key-value mapping
  - Big scalable HashMap
  - Highly fault tolerant (typically)
- Examples:
  - Redis, Riak, Voldemort



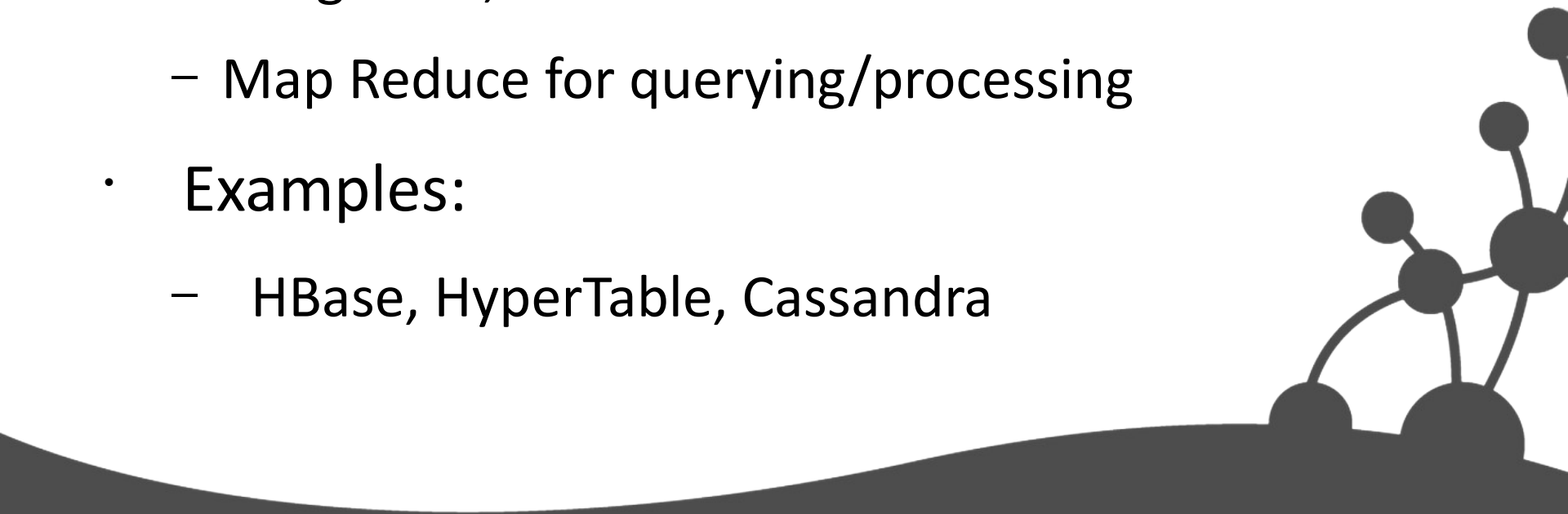
Key Value Stores: Pros and Cons

- Pros:
  - Simple data model
  - Scalable
- Cons
  - Create your own “foreign keys”
  - Poor for complex data



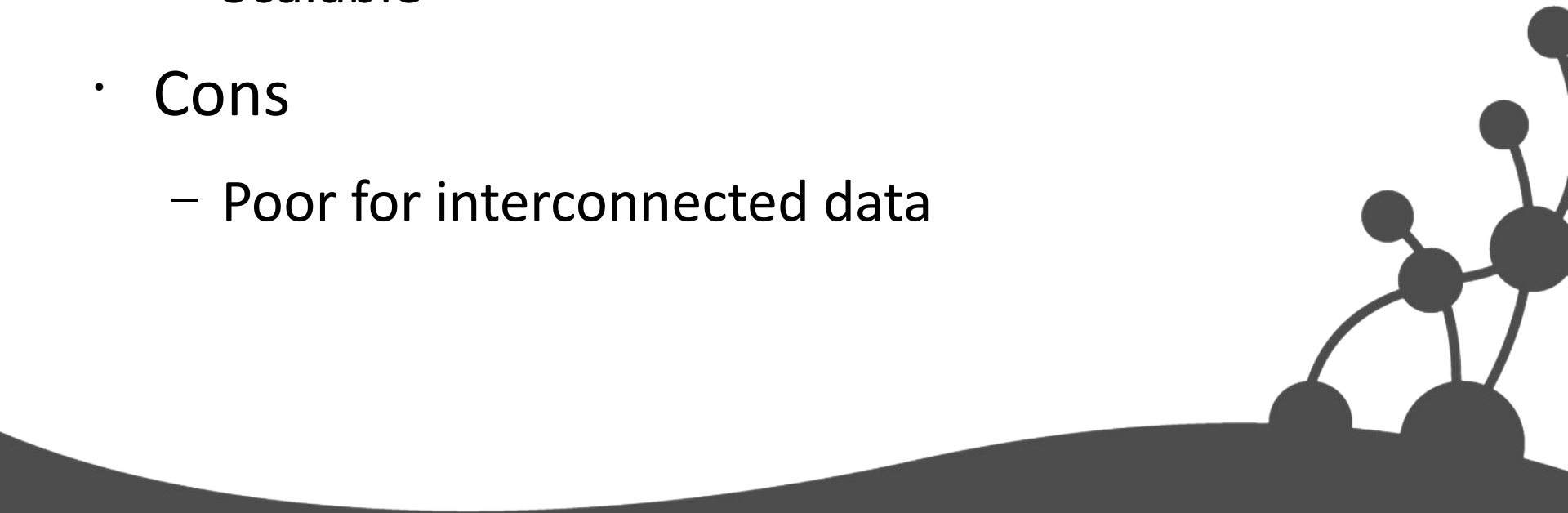
Column Family

- Most Based on **BigTable**: Google's Distributed Storage System for Structured Data
- Data Model:
  - A big table, with column families
  - Map Reduce for querying/processing
- Examples:
  - HBase, HyperTable, Cassandra



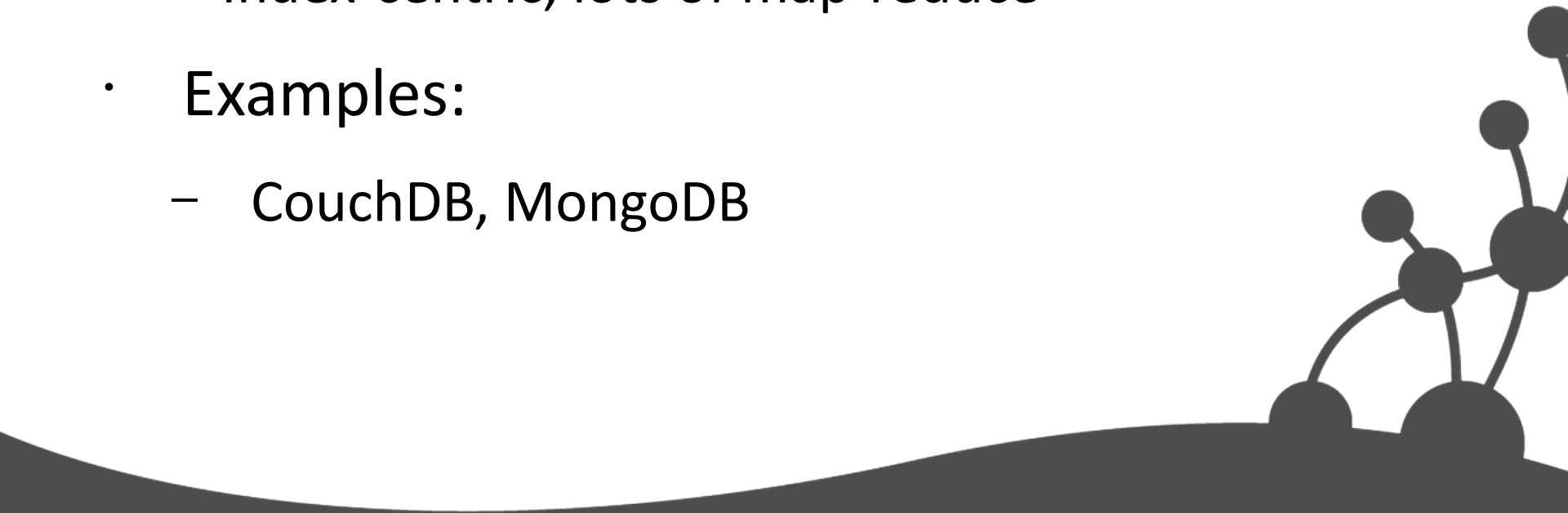
Column Family: Pros and Cons

- Pros:
  - Supports Simi-Structured Data
  - Naturally Indexed (columns)
  - Scalable
- Cons
  - Poor for interconnected data



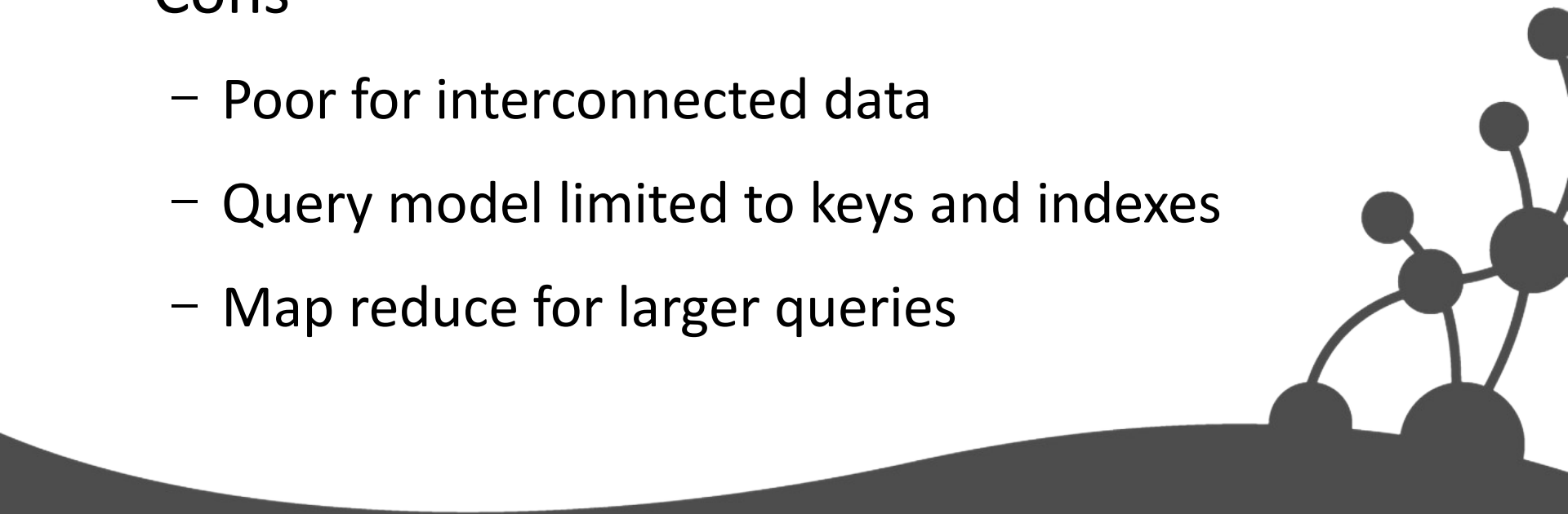
Document Databases

- Data Model:
  - A collection of documents
  - A document is a key value collection
  - Index-centric, lots of map-reduce
- Examples:
  - CouchDB, MongoDB



Document Databases: Pros and Cons

- Pros:
  - Simple, powerful data model
  - Scalable
- Cons
  - Poor for interconnected data
  - Query model limited to keys and indexes
  - Map reduce for larger queries

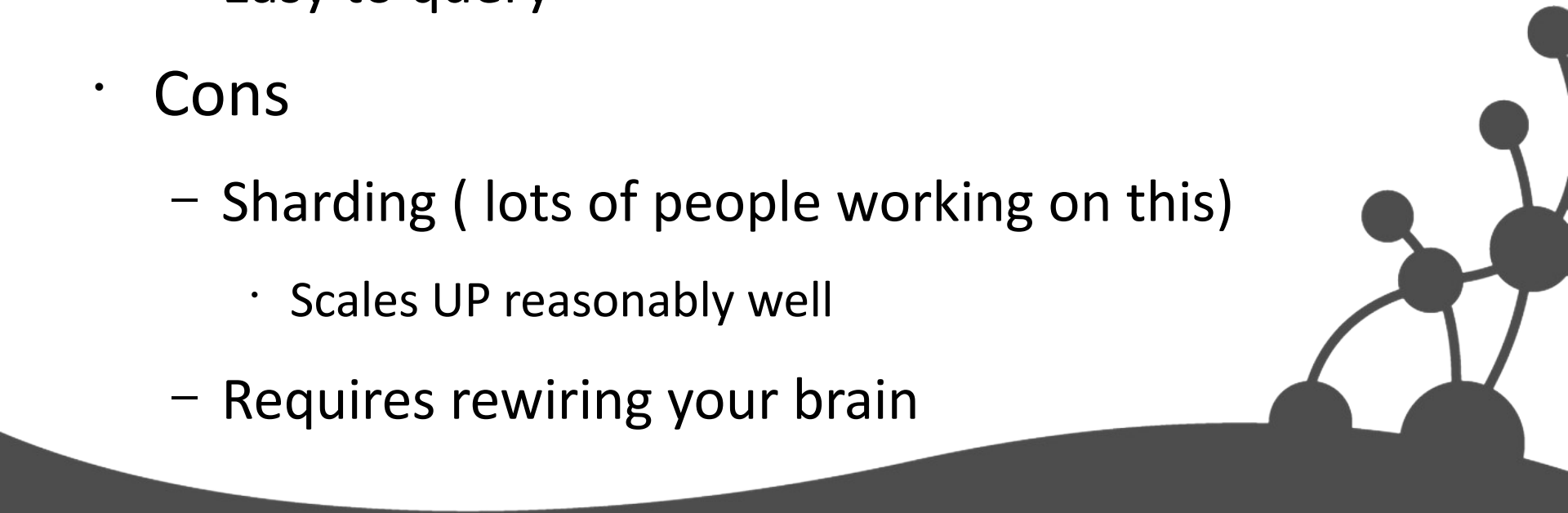


Graph Databases

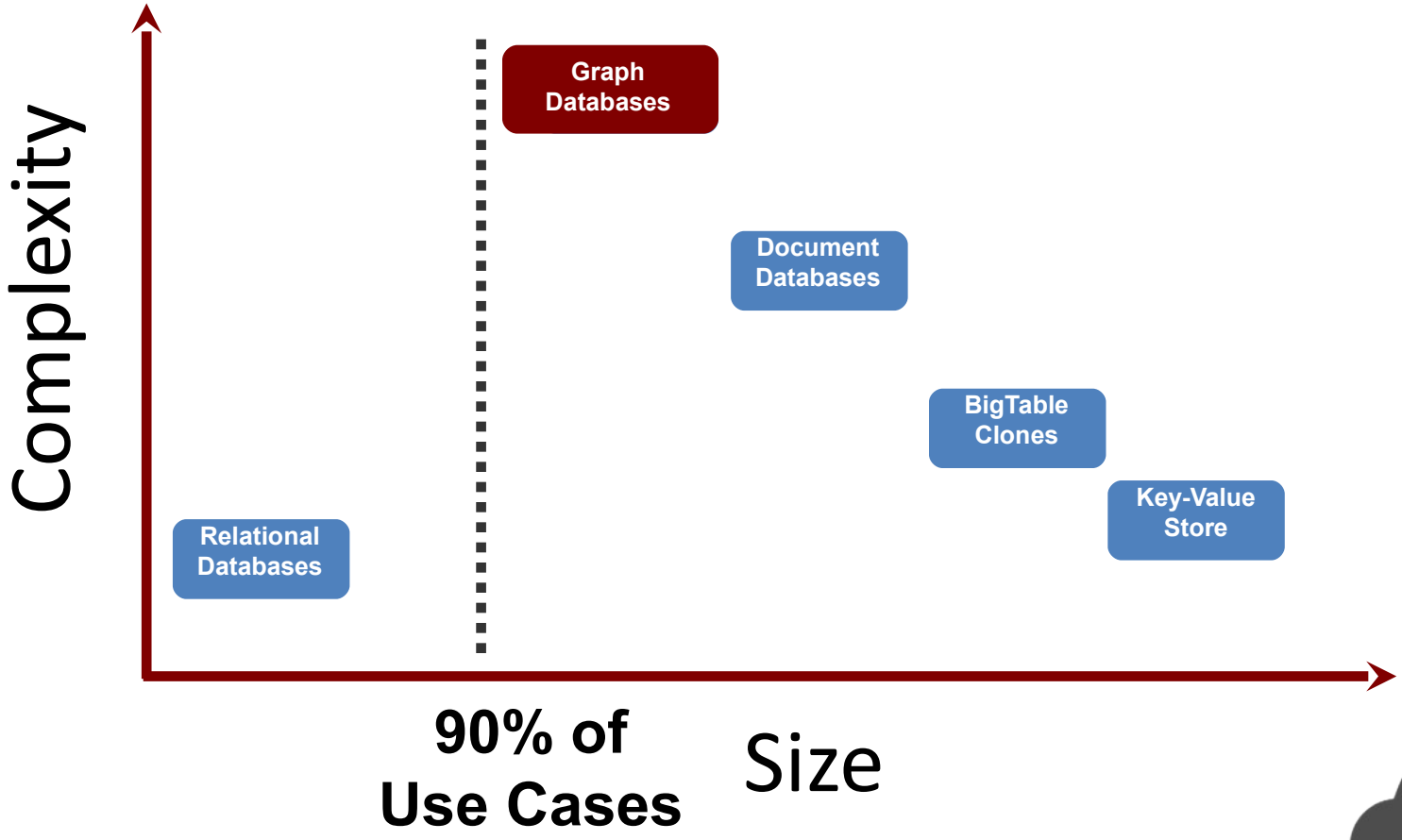
- Data Model:
  - Nodes and Relationships
- Examples:
  - Neo4j, OrientDB, InfiniteGraph, AllegroGraph

Graph Databases: Pros and Cons

- Pros:
  - Powerful data model, as general as RDBMS
  - Connected data locally indexed
  - Easy to query
- Cons
  - Sharding (lots of people working on this)
    - Scales UP reasonably well
  - Requires rewiring your brain



Living in a NOSQL World



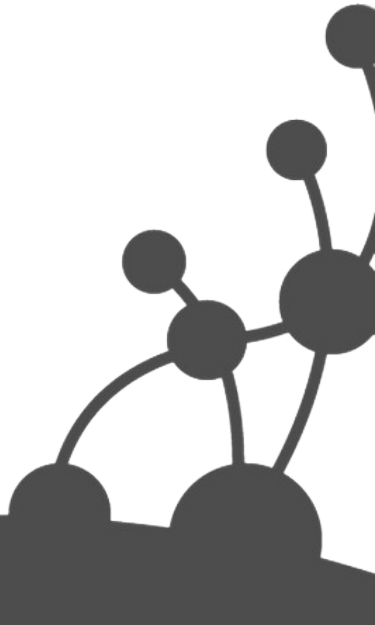
What is a Graph?

What is a Graph?

- An abstract representation of a set of objects where some pairs are connected by links.

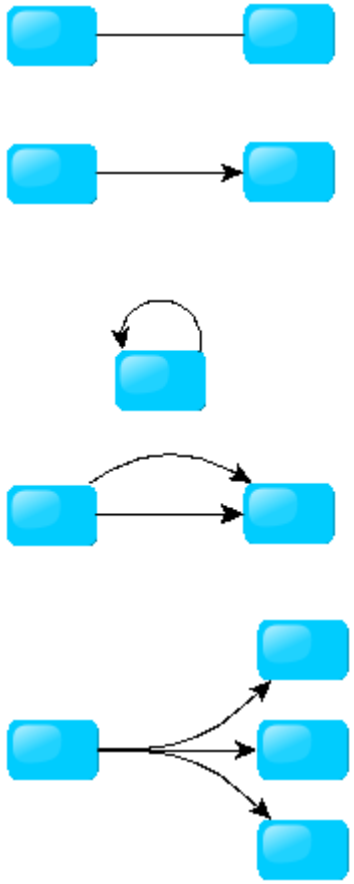
Object (Vertex, Node)

Link (Edge, Arc, Relationship)



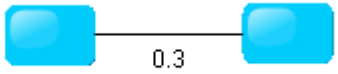
Different Kinds of Graphs

- Undirected Graph
- Directed Graph
- Pseudo Graph
- Multi Graph
- Hyper Graph

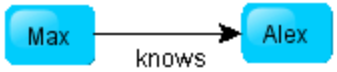


More Kinds of Graphs

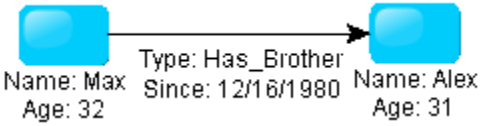
- Weighted Graph



- Labeled Graph

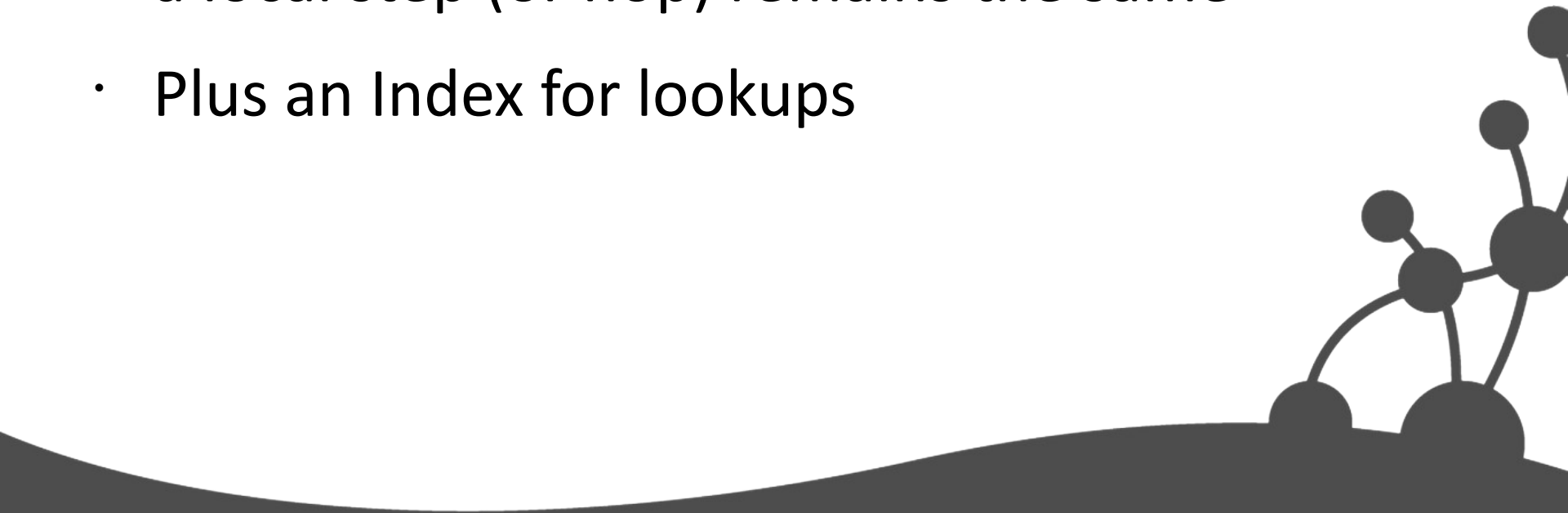


- Property Graph



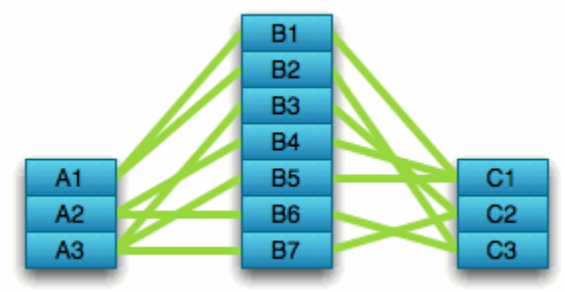
What is a Graph Database?

- A database with an explicit graph structure
- Each node knows its adjacent nodes
- As the number of nodes increases, the cost of a local step (or hop) remains the same
- Plus an Index for lookups

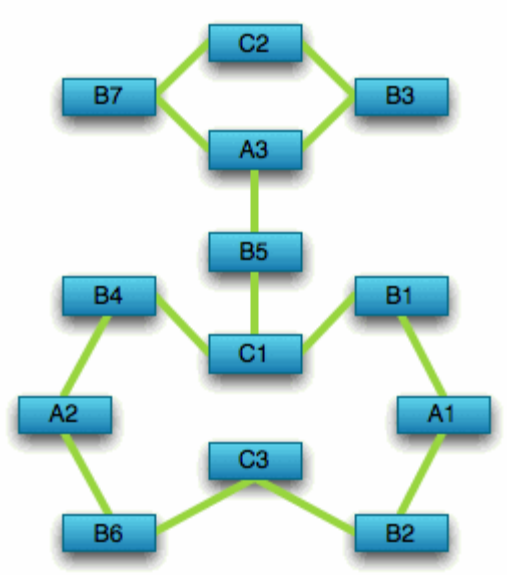


Compared to Relational Databases

Optimized for aggregation



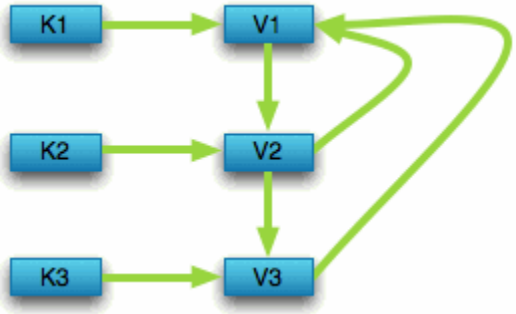
Optimized for connections



Compared to Key Value Stores

Optimized for simple look-ups

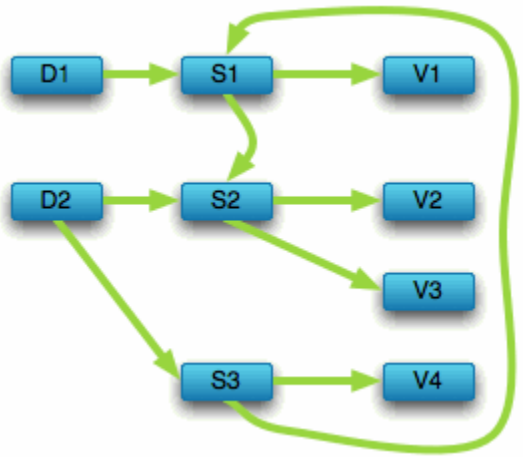
Optimized for traversing connected data



Compared to Key Value Stores

Optimized for “trees” of data

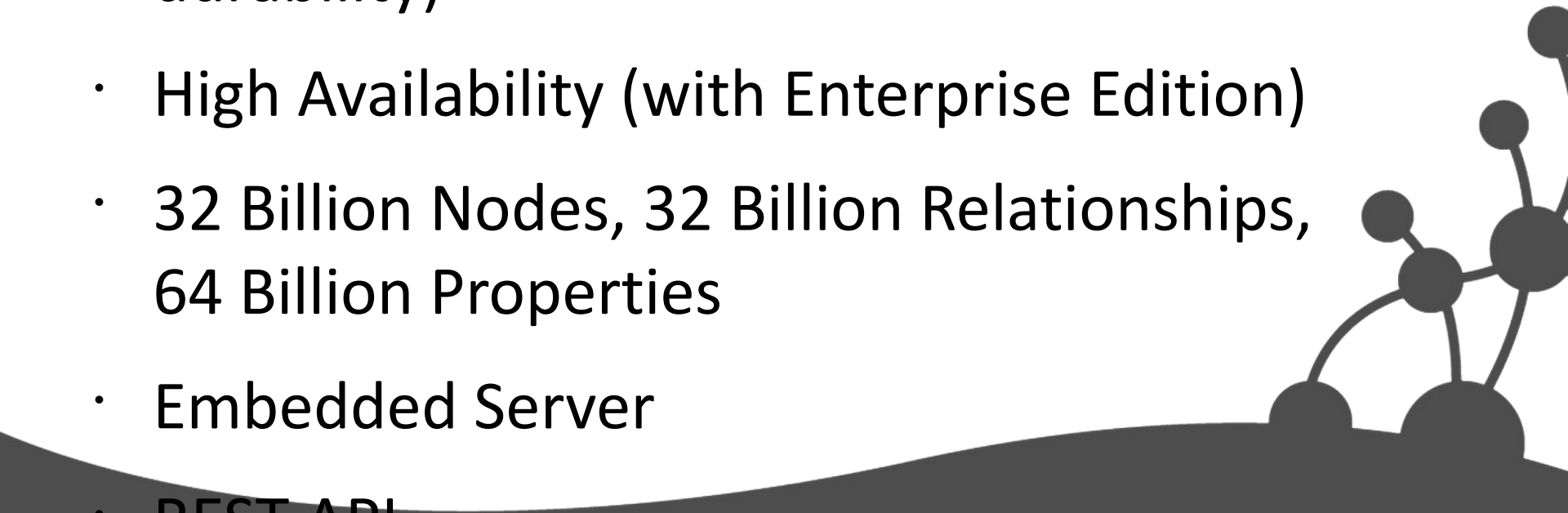
Optimized for seeing the forest and the trees, and the branches, and the trunks



What is Neo4j?

What is Neo4j?

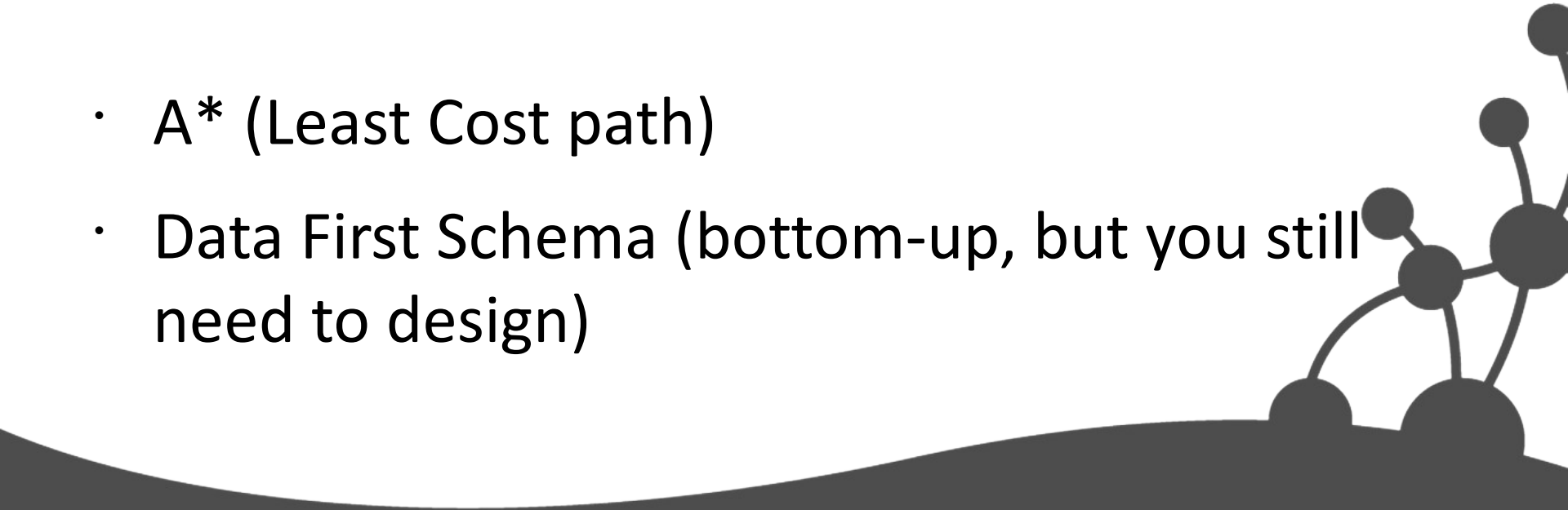
- A Graph Database + Lucene Index
- Property Graph
- Full ACID (atomicity, consistency, isolation, durability)
- High Availability (with Enterprise Edition)
- 32 Billion Nodes, 32 Billion Relationships, 64 Billion Properties
- Embedded Server
- REST API



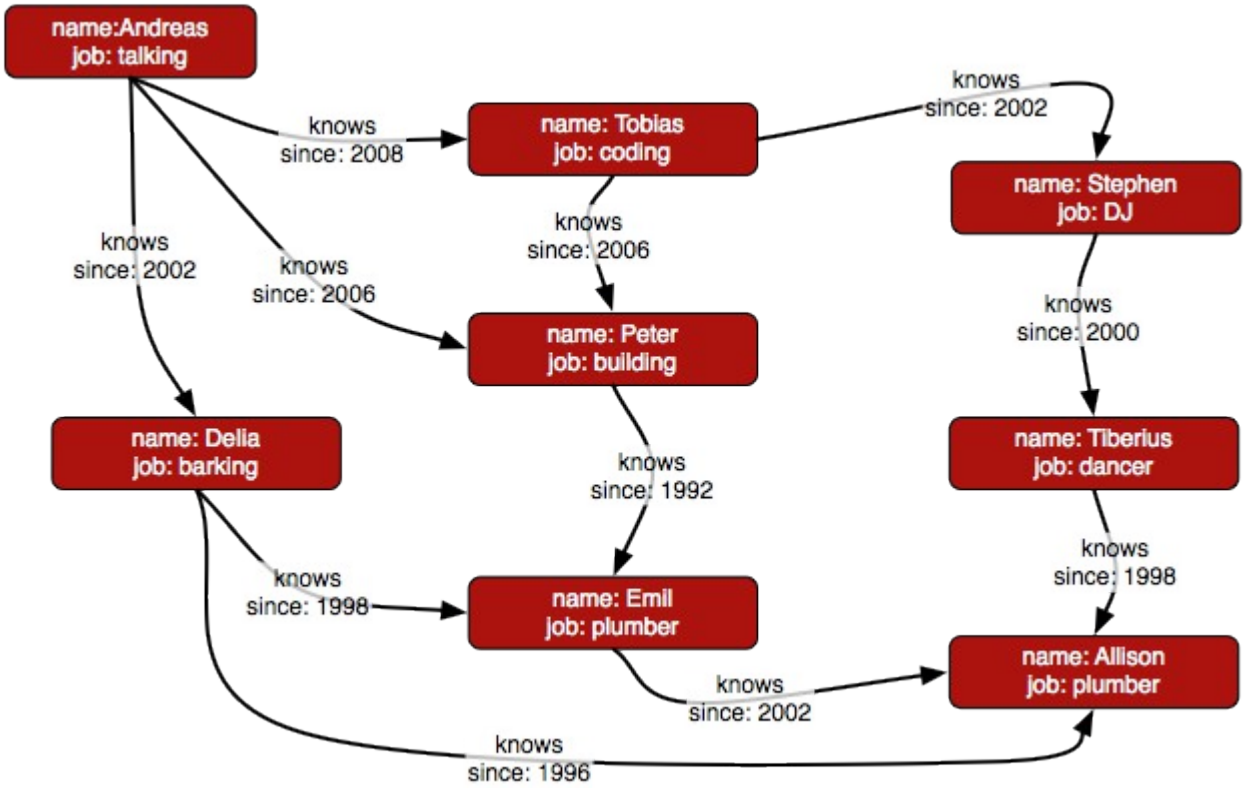
Good For

- Highly connected data (social networks)
- Recommendations (e-commerce)
- Path Finding (how do I know you?)

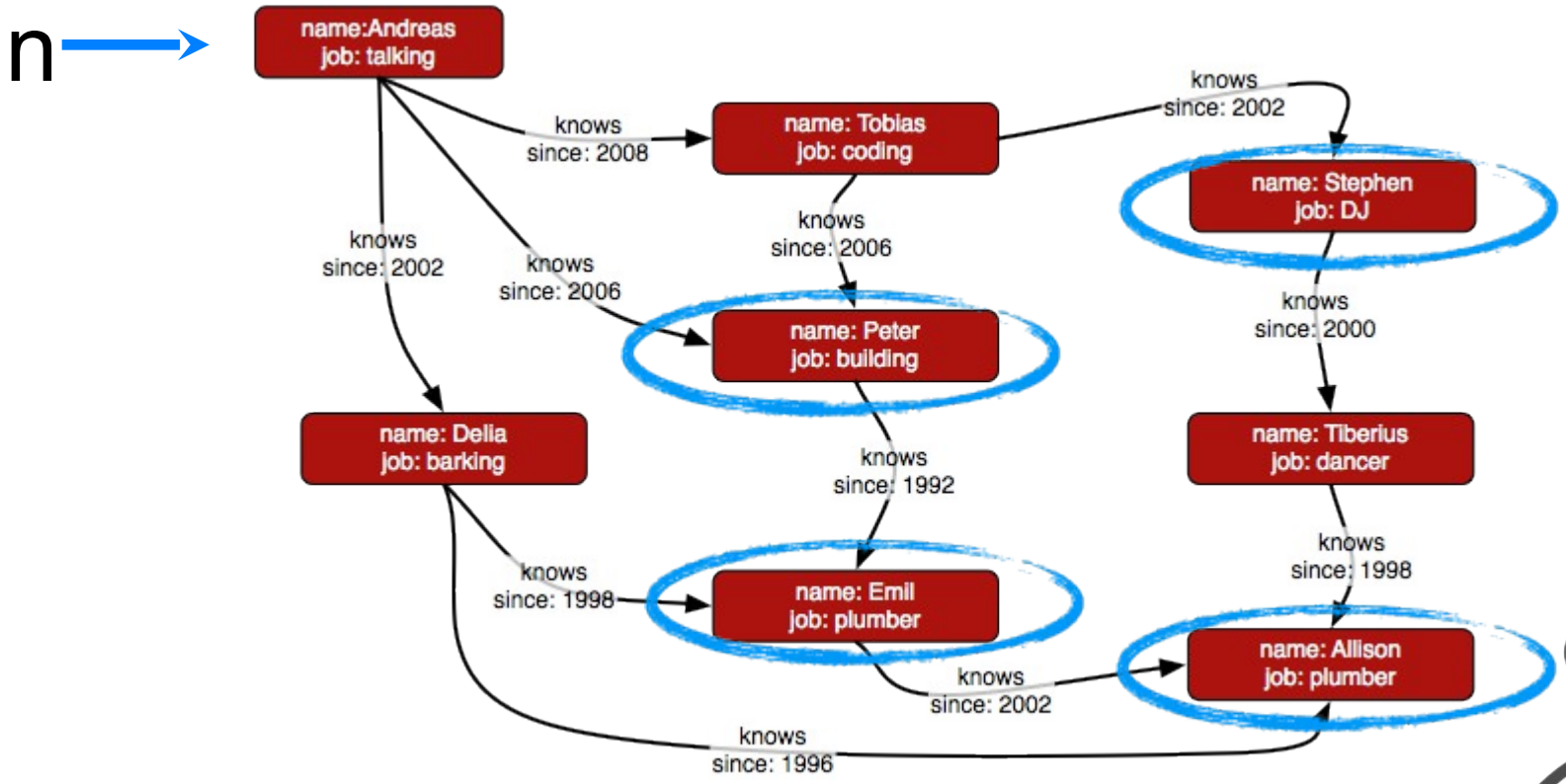
- A* (Least Cost path)
- Data First Schema (bottom-up, but you still need to design)



Property Graph



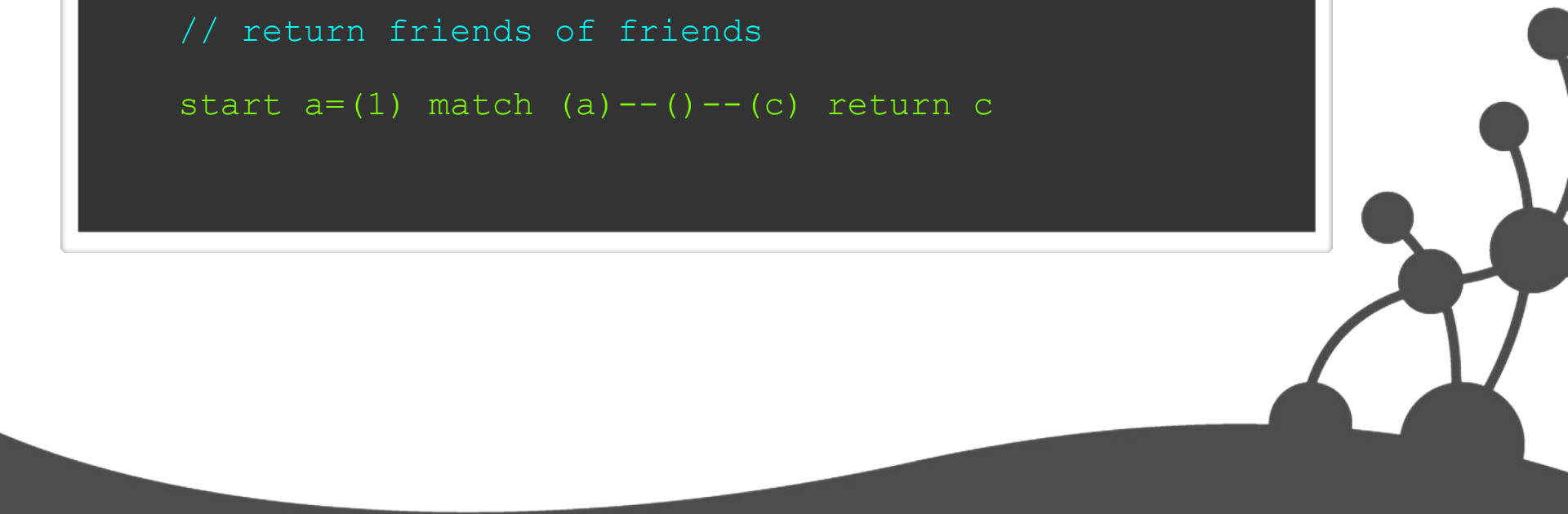
```
// then traverse to find results
start n=(people-index, name, "Andreas")
match (n)--()--(foaf) return foaf
```



Cypher

Pattern Matching Query Language (like SQL for graphs)

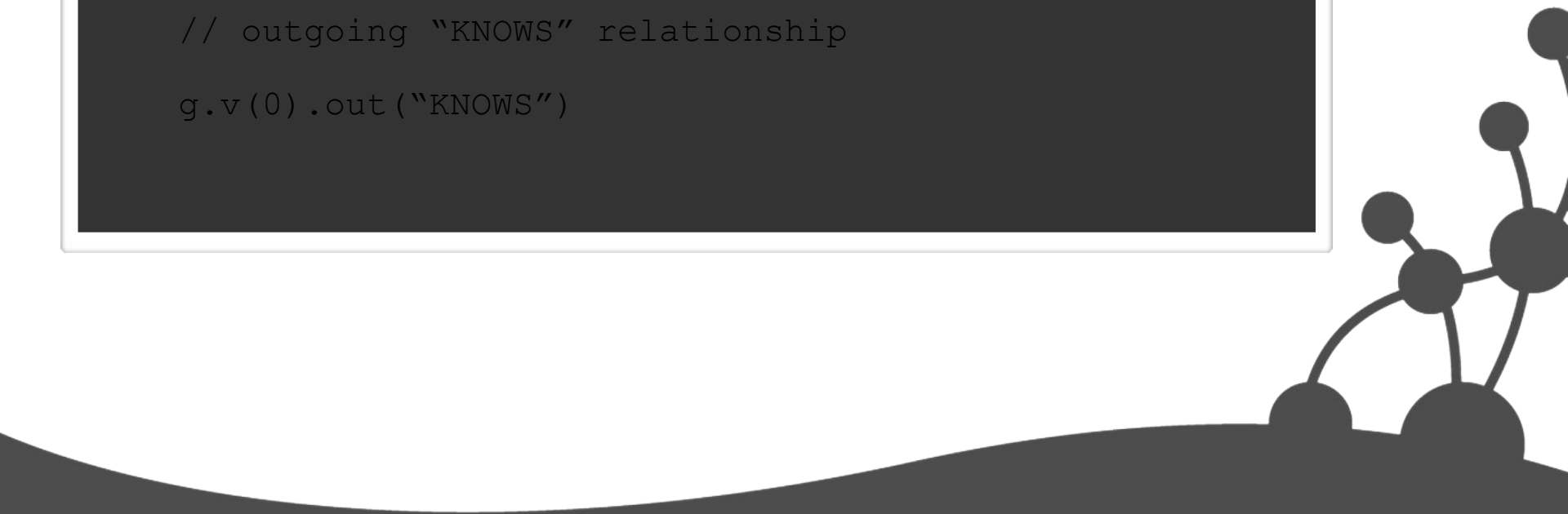
```
// get node 0

start a=(0) return a

// traverse from node 1

start a=(1) match (a)-->(b) return b

// return friends of friends

start a=(1) match (a)--()--(c) return c
```



Gremlin

A Graph Scripting DSL (groovy-based)

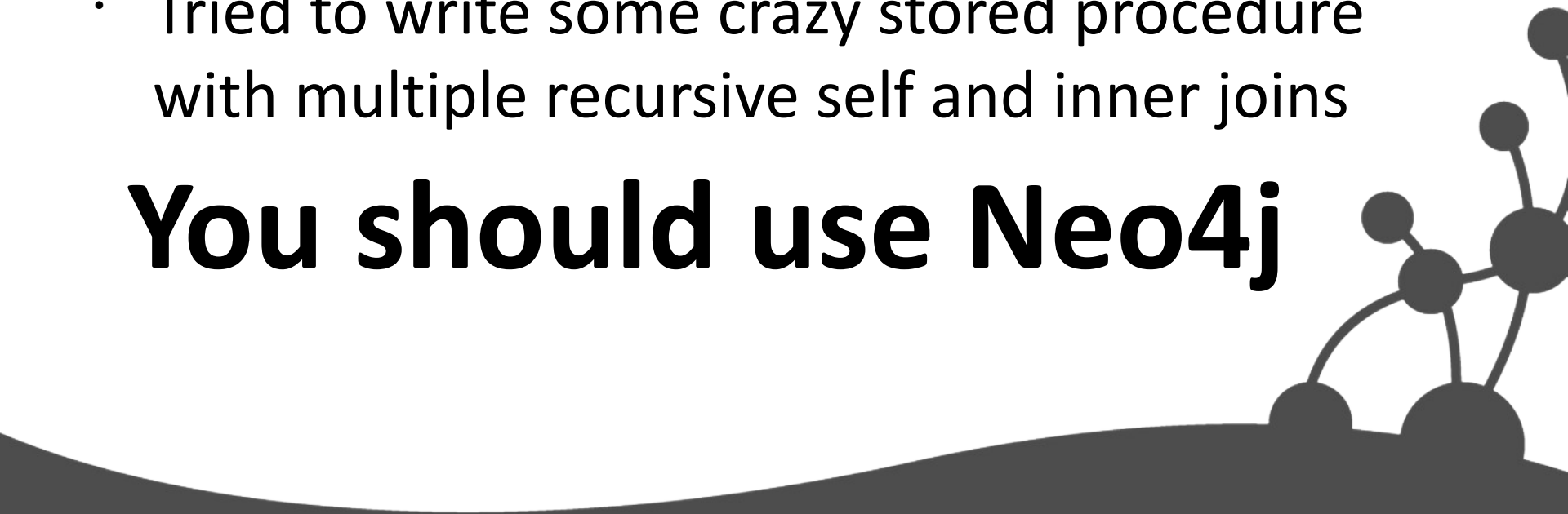
```
// get node 0

g.v(0)

// nodes with incoming relationship

g.v(0).in

// outgoing "KNOWS" relationship

g.v(0).out("KNOWS")
```

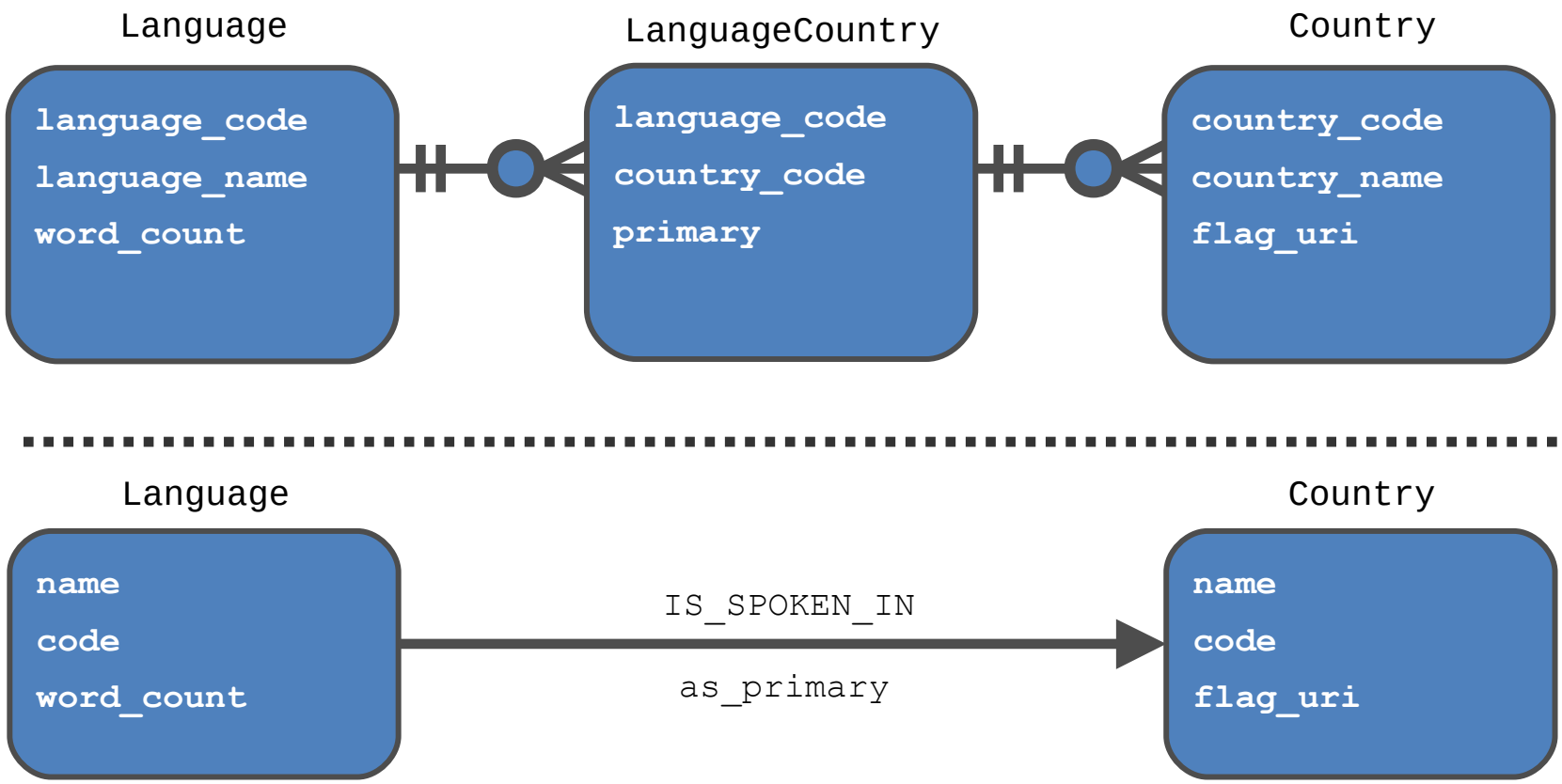


If you've ever

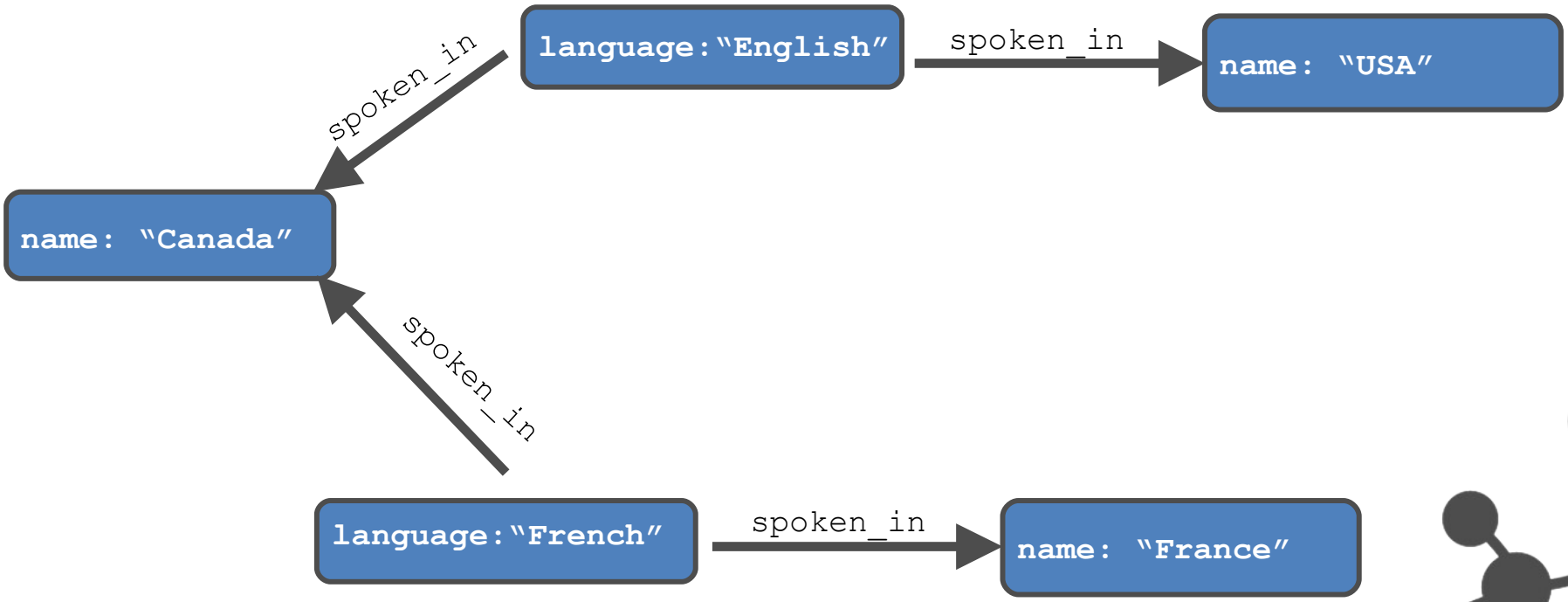
- Joined more than 7 tables together
- Modeled a graph in a table
- Written a recursive CTE
- Tried to write some crazy stored procedure with multiple recursive self and inner joins

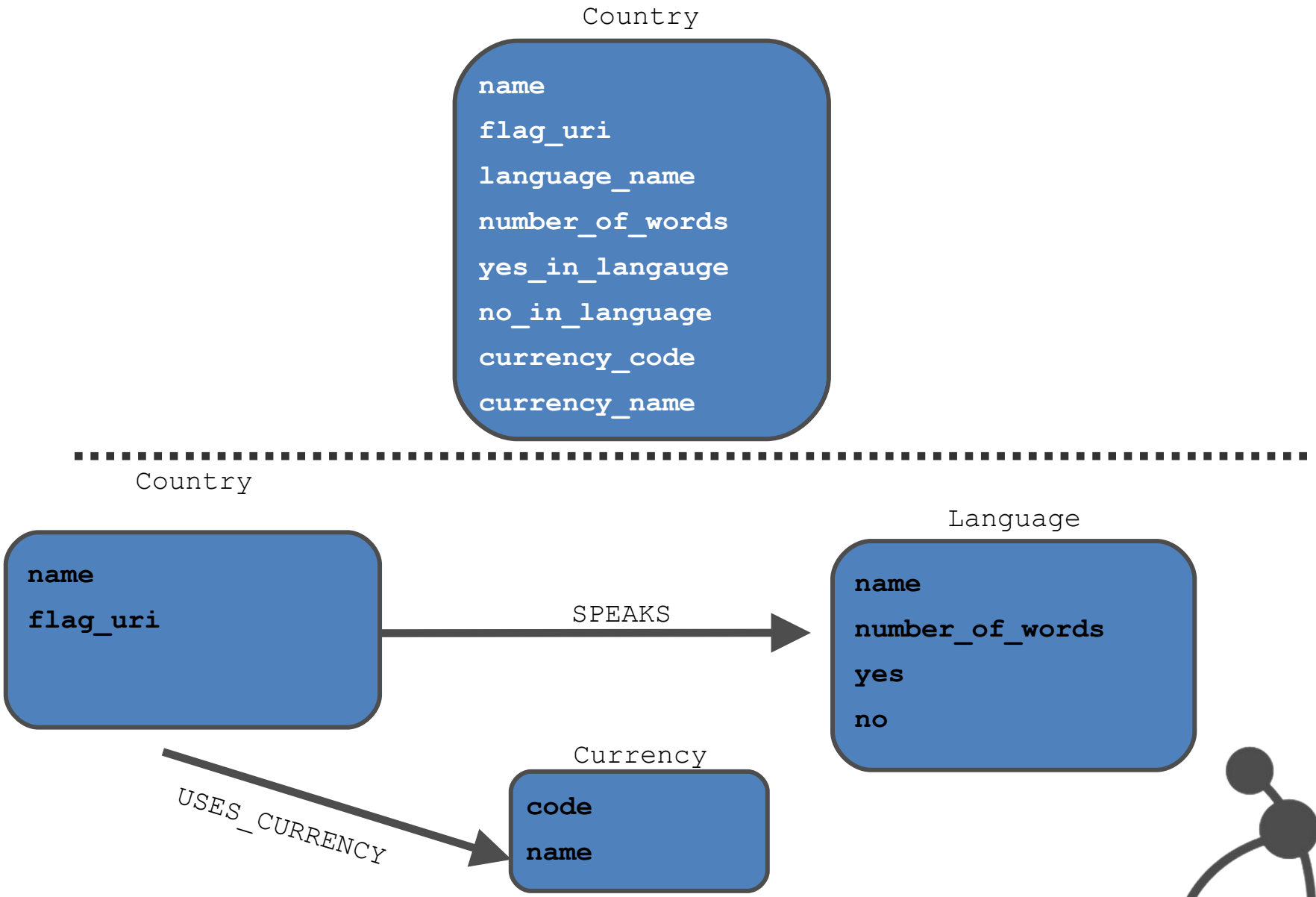
You should use Neo4j



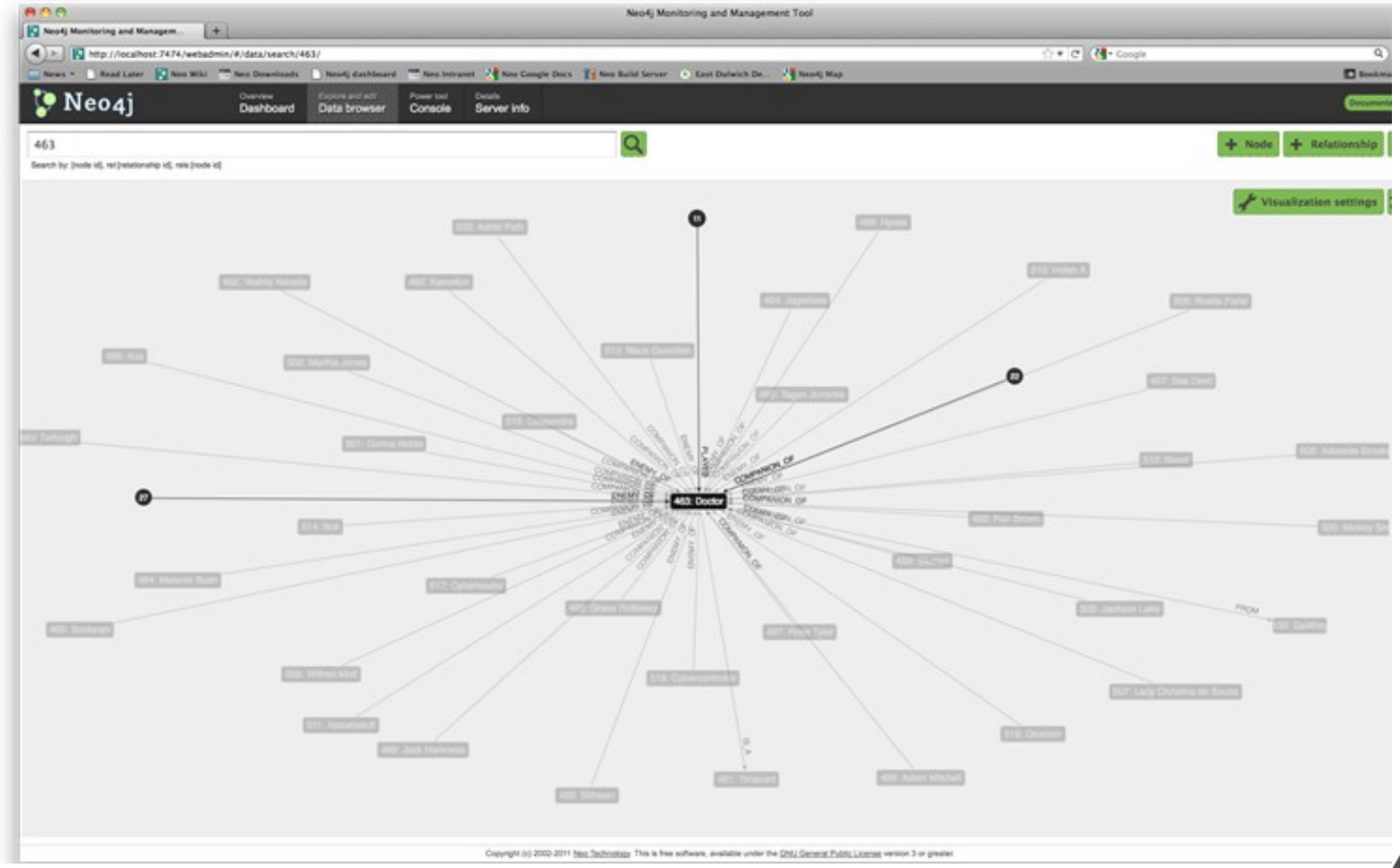

Language
language_code
language_name
word_count
LanguageCountry
language_code
country_code
primary
Country
country_code
country_name
flag_uri
Language
name
code
word_count
IS_SPOKEN_IN
as_primary
Country
name
code
flag_uri

```
name: "Canada"
languages_spoken: "[ 'English', 'French' ]"
```

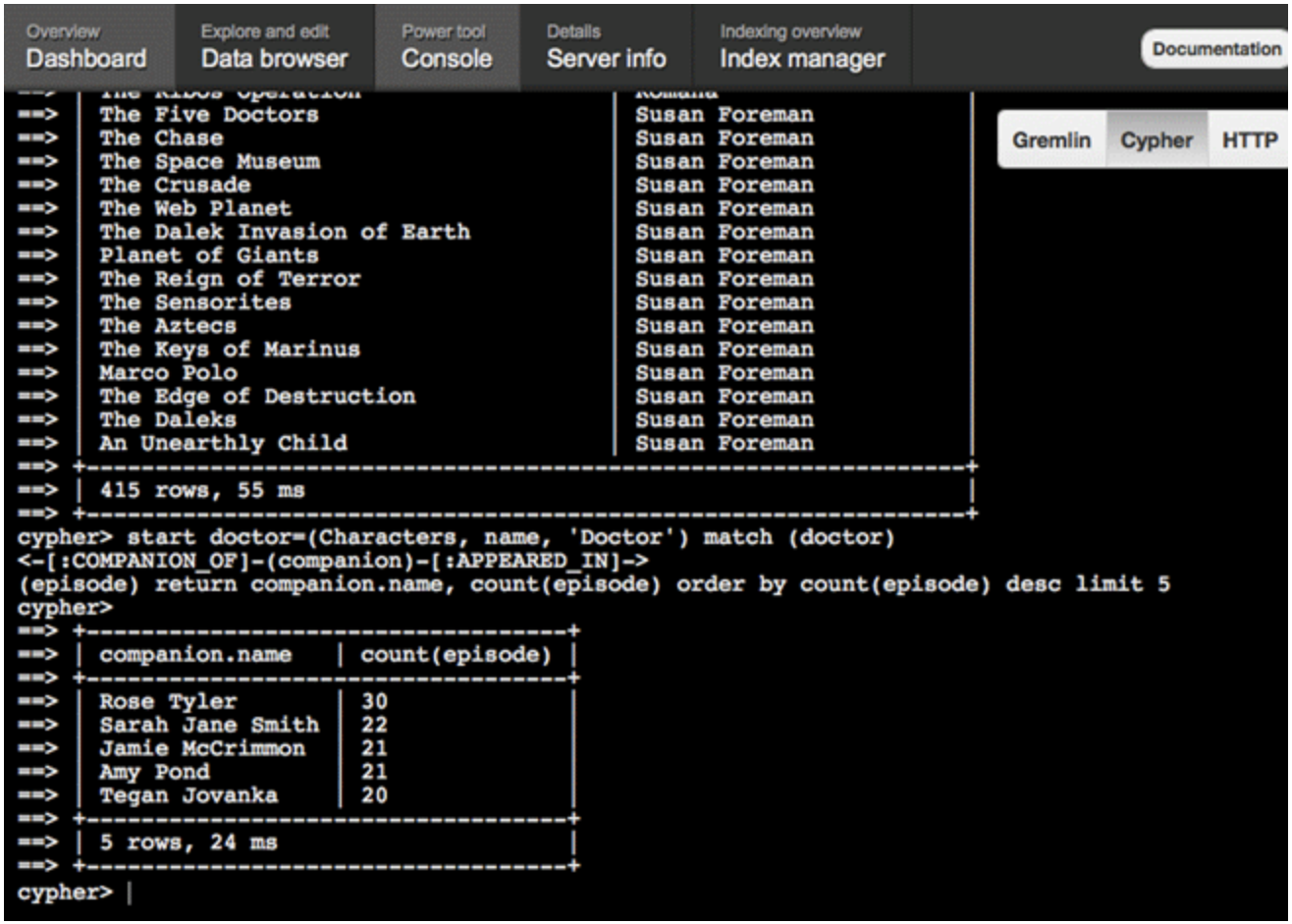



Country
name
flag_uri
language_name
number_of_words
yes_in_langauge
no_in_language
currency_code
currency_name
Country
name
flag_uri
SPEAKS
Language
name
number_of_words
yes
no
USES_CURRENCY
Currency
code
name

Neo4j Data Browser



Neo4j Console

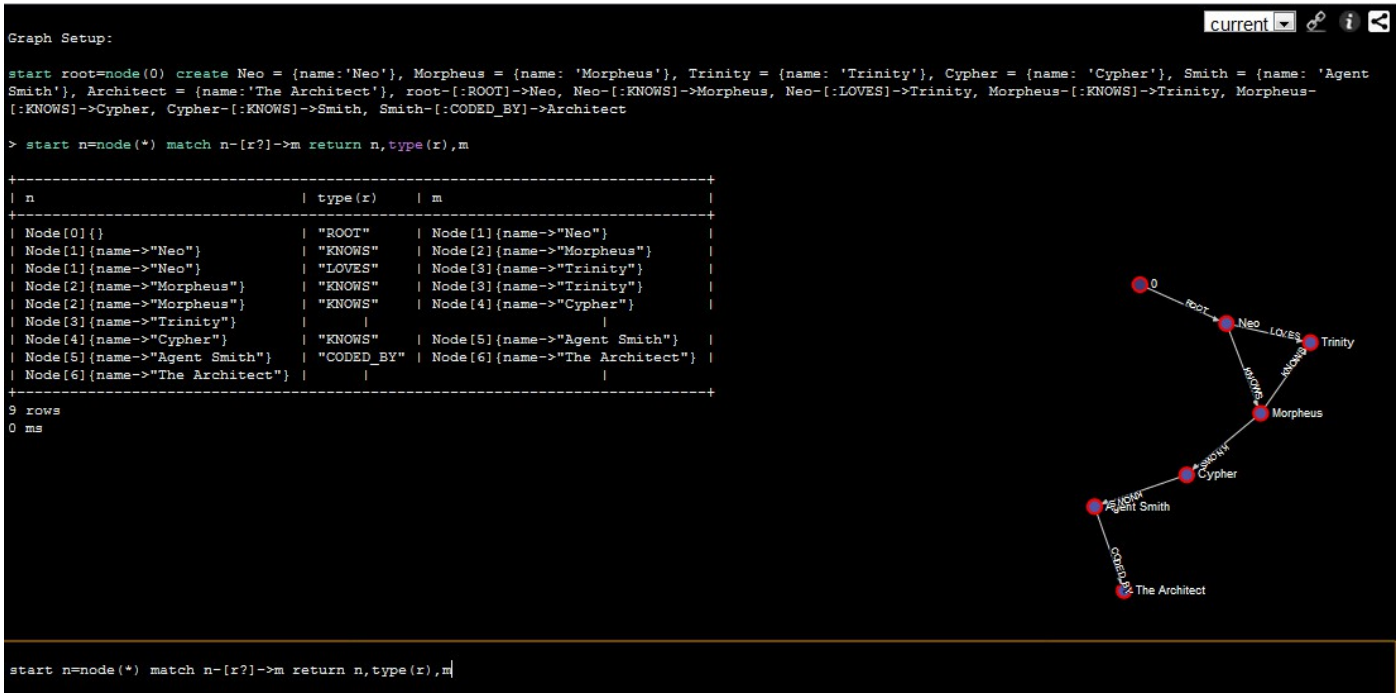


console.neo4j.org

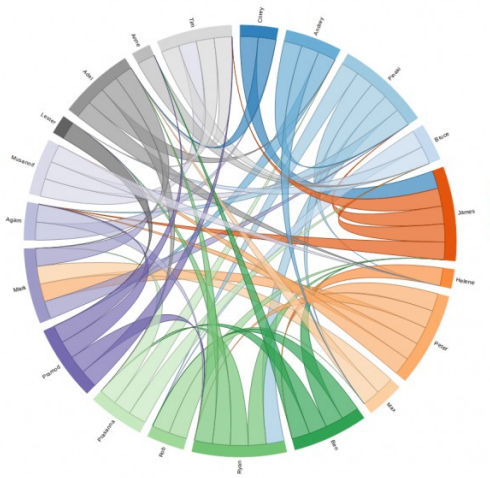
Try it right now:

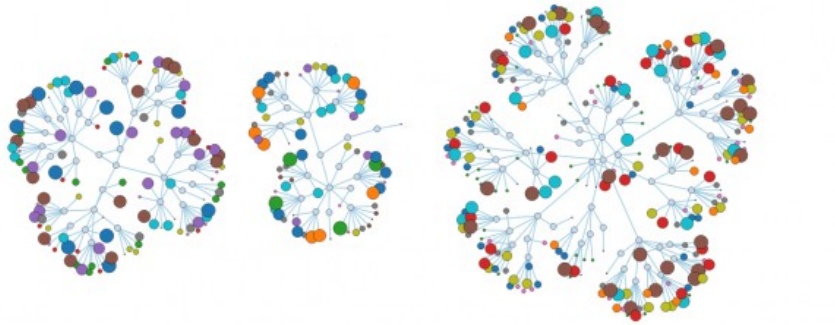
start n=node(*) match n-[r:LOVES]->m return n, type(r), m

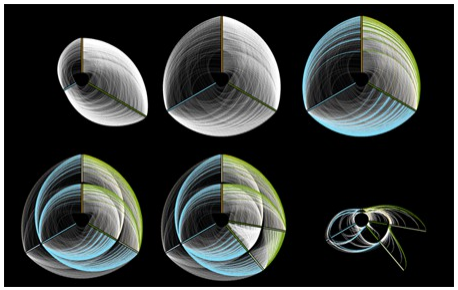
Notice the two nodes in red, they are your result set.

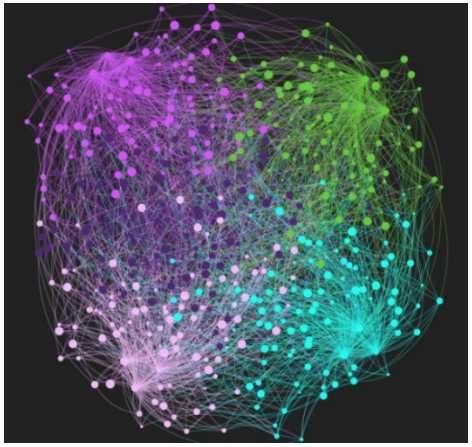


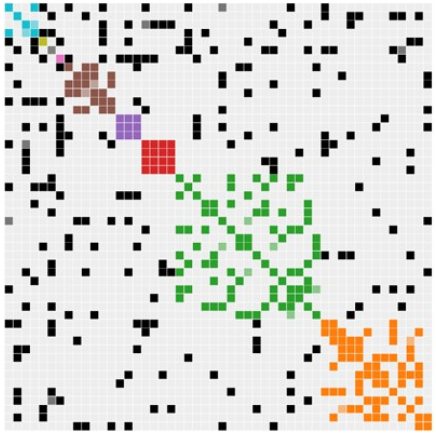
What does a Graph look like?

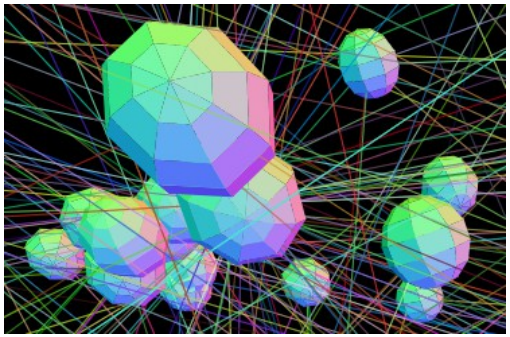












Questions?

Thank you!

http://maxdemarzi.com