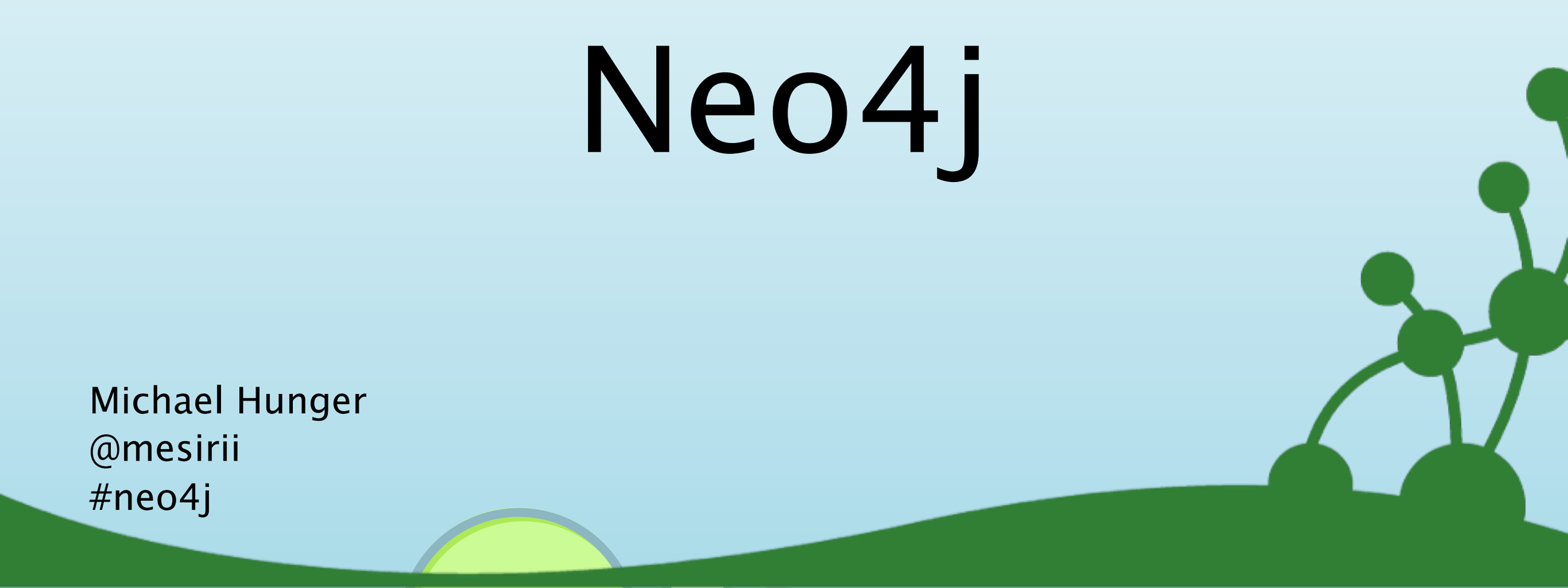
REALITY IS A GRAPH

EMBRACE IT

Introduction to Neo4j

Michael Hunger
@mesirii
#neo4j



Introduction to Neo4j

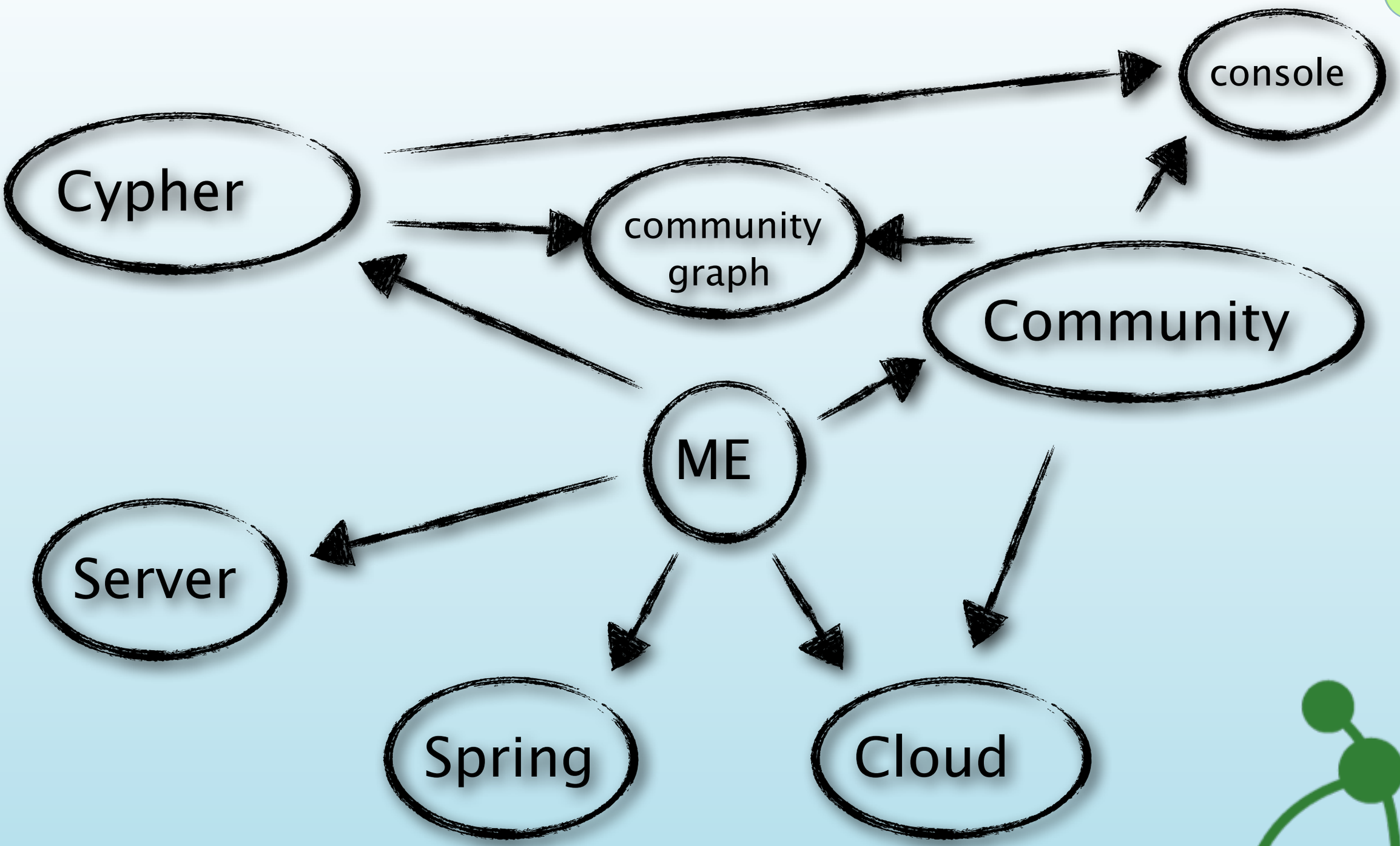
Michael Hunger
@mesirii
#neo4j

Neo4j

Introduction to Neo4j

Michael Hunger
@mesirii
#neo4j

(Michael) -[:WORKS_ON]-> (Neo4j)



The Path Forward

The Path Forward

1.No .. NO .. NOSQL

The Path Forward

1. No .. NO .. NOSQL
2. Why graphs?

The Path Forward

1. No .. NO .. NOSQL
2. Why graphs?
3. What's a graph database?

The Path Forward

1. No .. NO .. NOSQL
2. Why graphs?
3. What's a graph database?
4. Some things about Neo4j.

The Path Forward

1. No .. NO .. NOSQL
2. Why graphs?
3. What's a graph database?
4. Some things about Neo4j.
5. How do people use Neo4j?

NOSQL

NOSQL

Neo4j

Neo4j

What is NOSQL?

What is NOSQL?

What is NOSQL?

It's not "No to SQL"

What is NOSQL?

It's not "No to SQL"

It's not "Never SQL"

What is NOSQL?

It's not "No to SQL"

It's not "Never SQL"

It's "**Not Only SQL**"

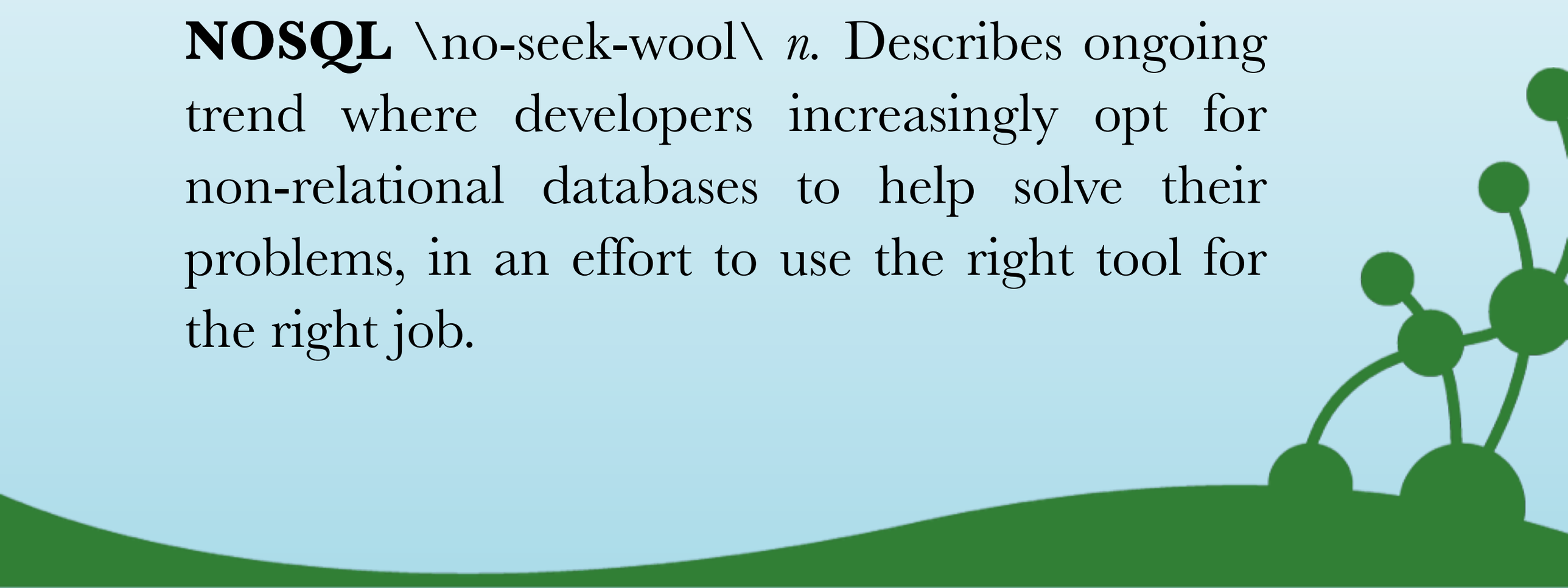
What is NOSQL?

It's not "No to SQL"

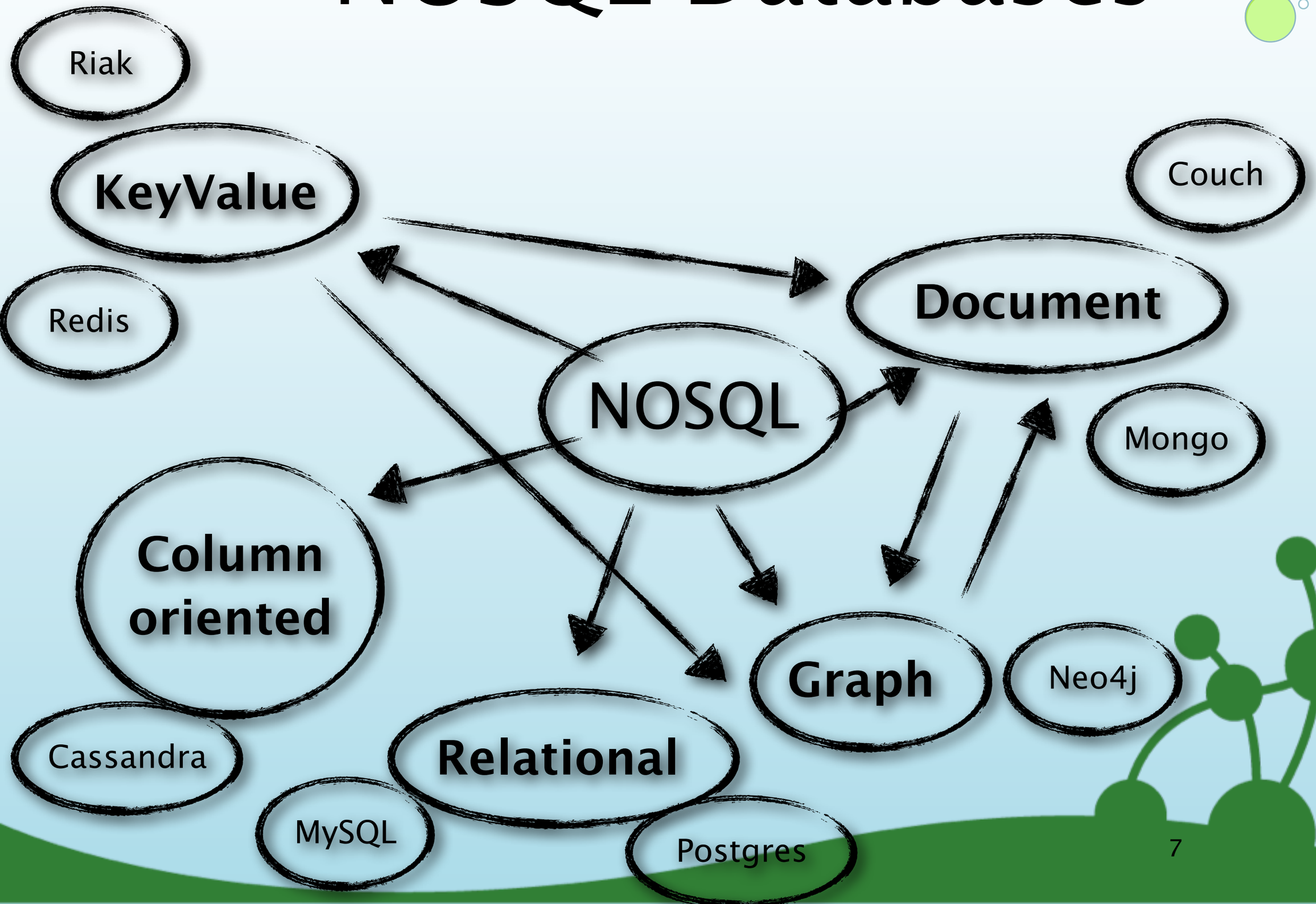
It's not "Never SQL"

It's "**Not Only SQL**"

NOSQL \no-seek-wool\ *n.* Describes ongoing trend where developers increasingly opt for non-relational databases to help solve their problems, in an effort to use the right tool for the right job.

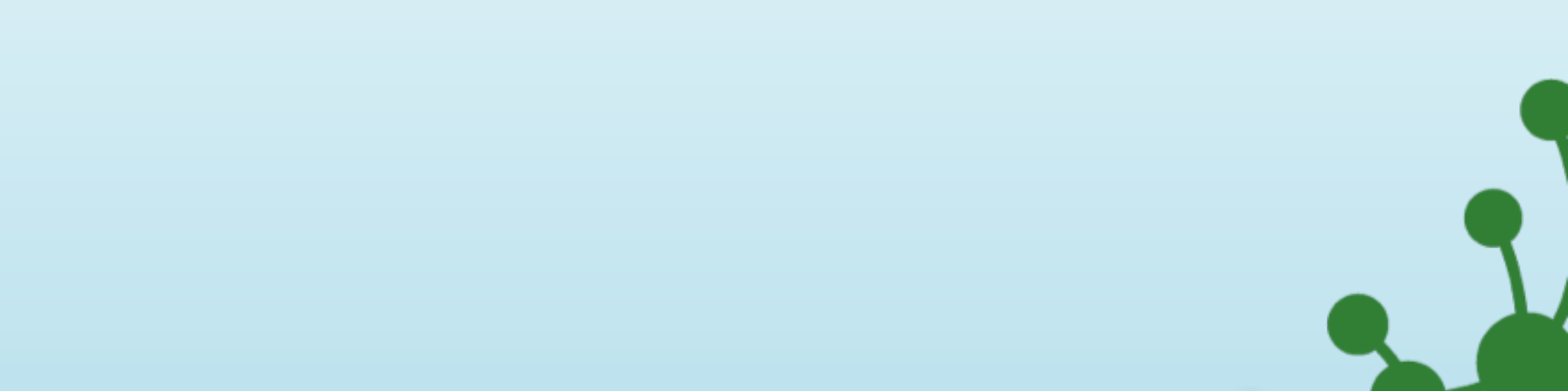


NOSQL Databases

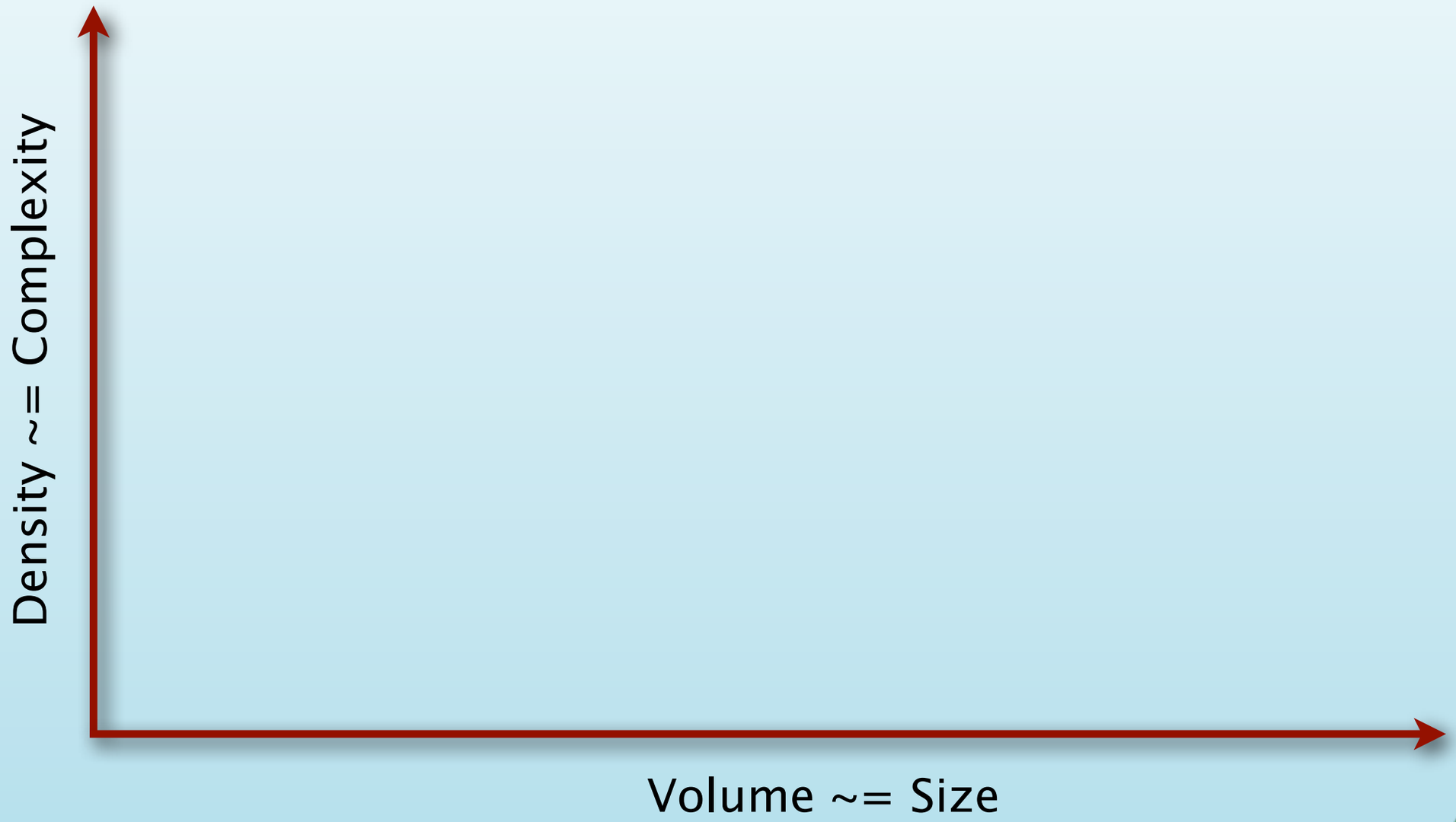


Living in a NOSQL World

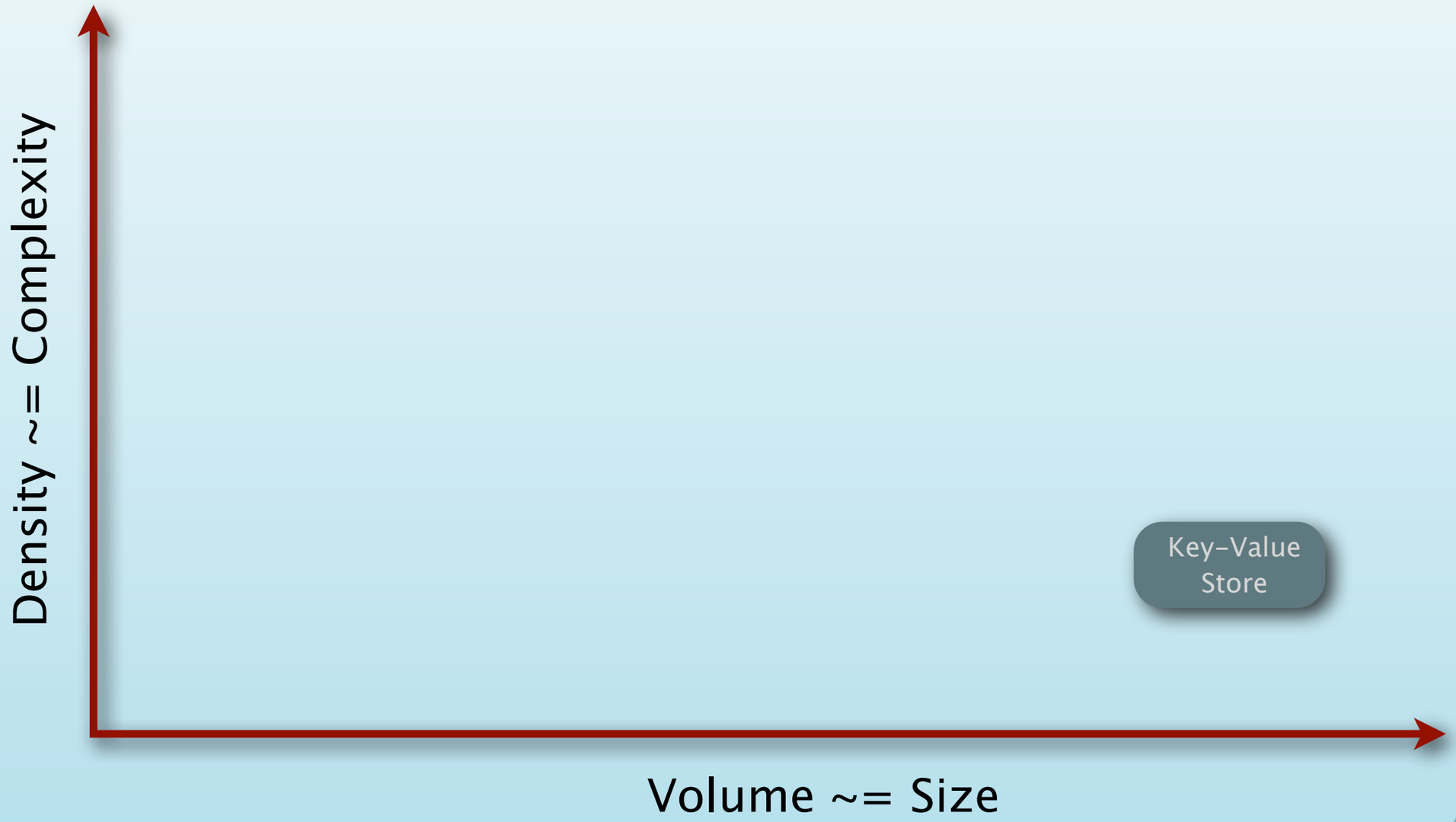
Living in a NOSQL World



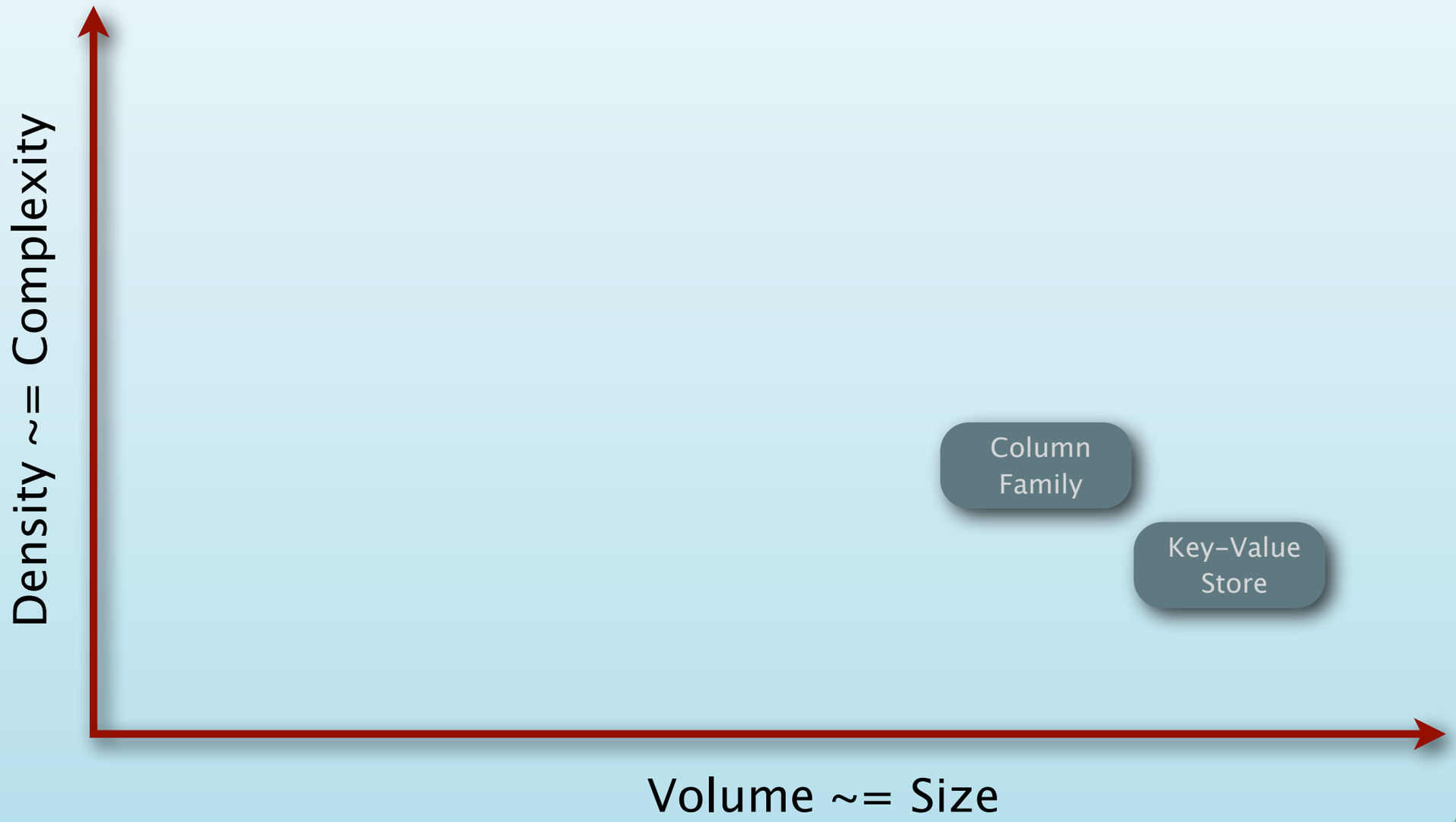
Living in a NOSQL World



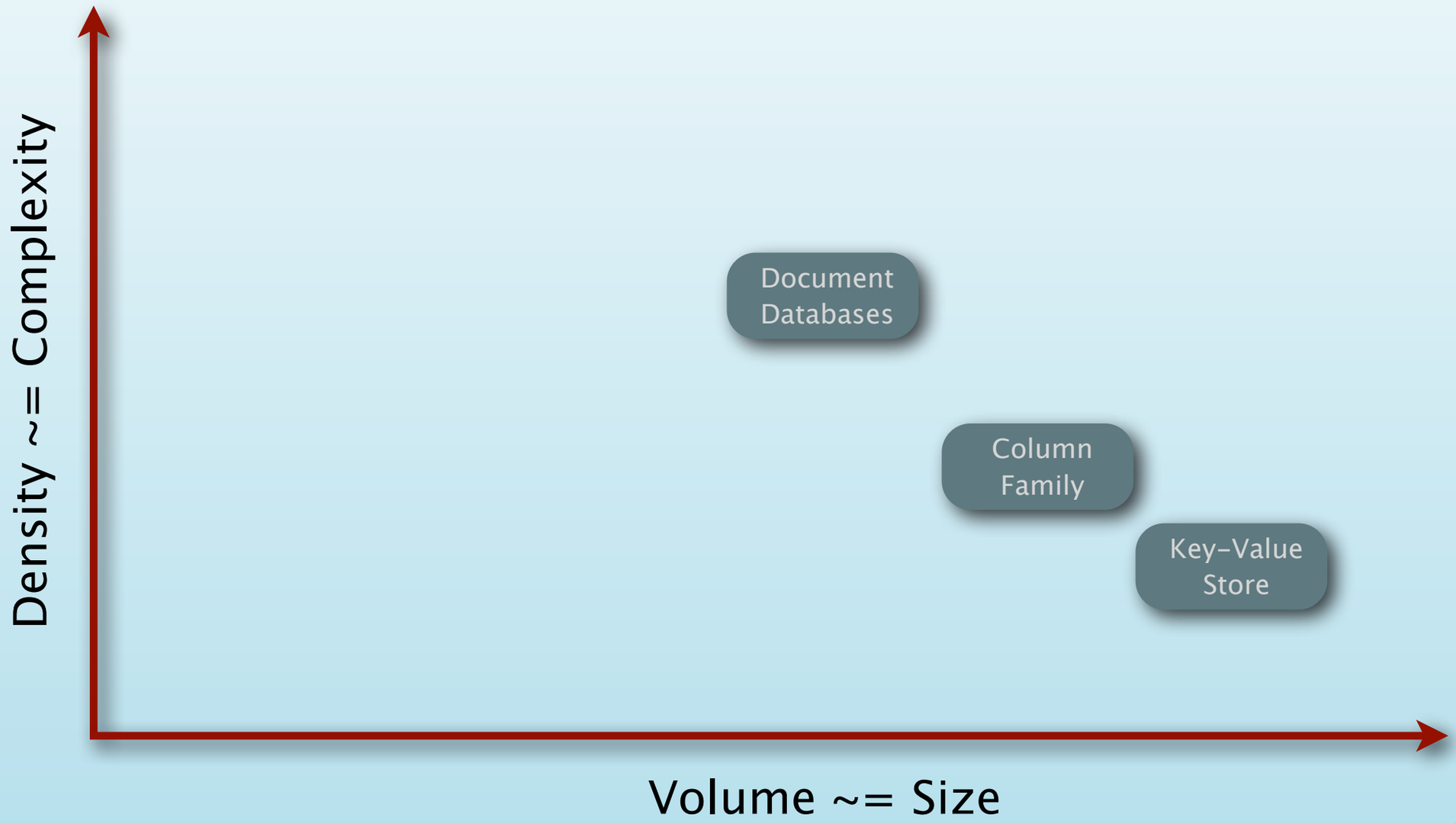
Living in a NOSQL World



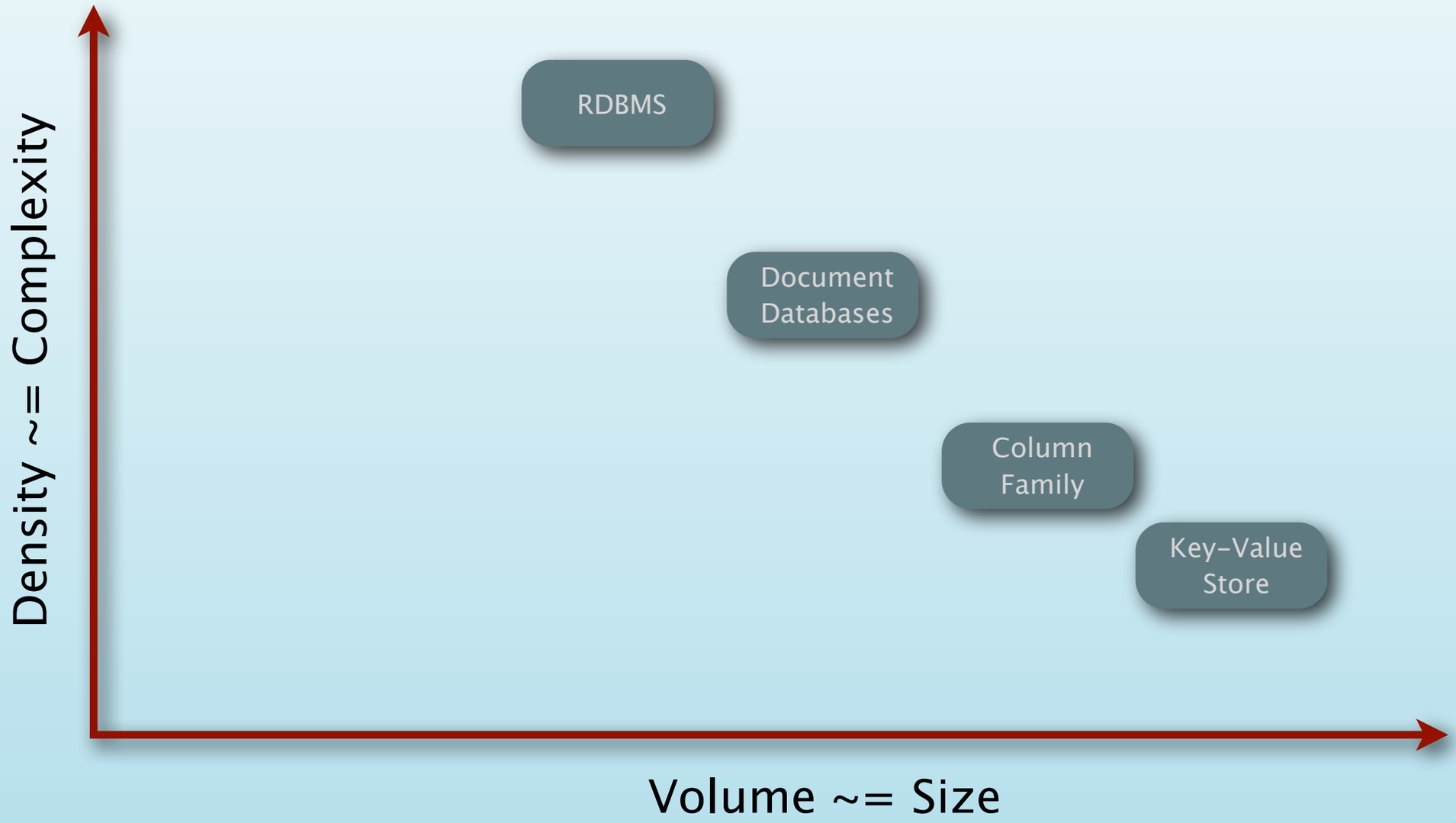
Living in a NOSQL World



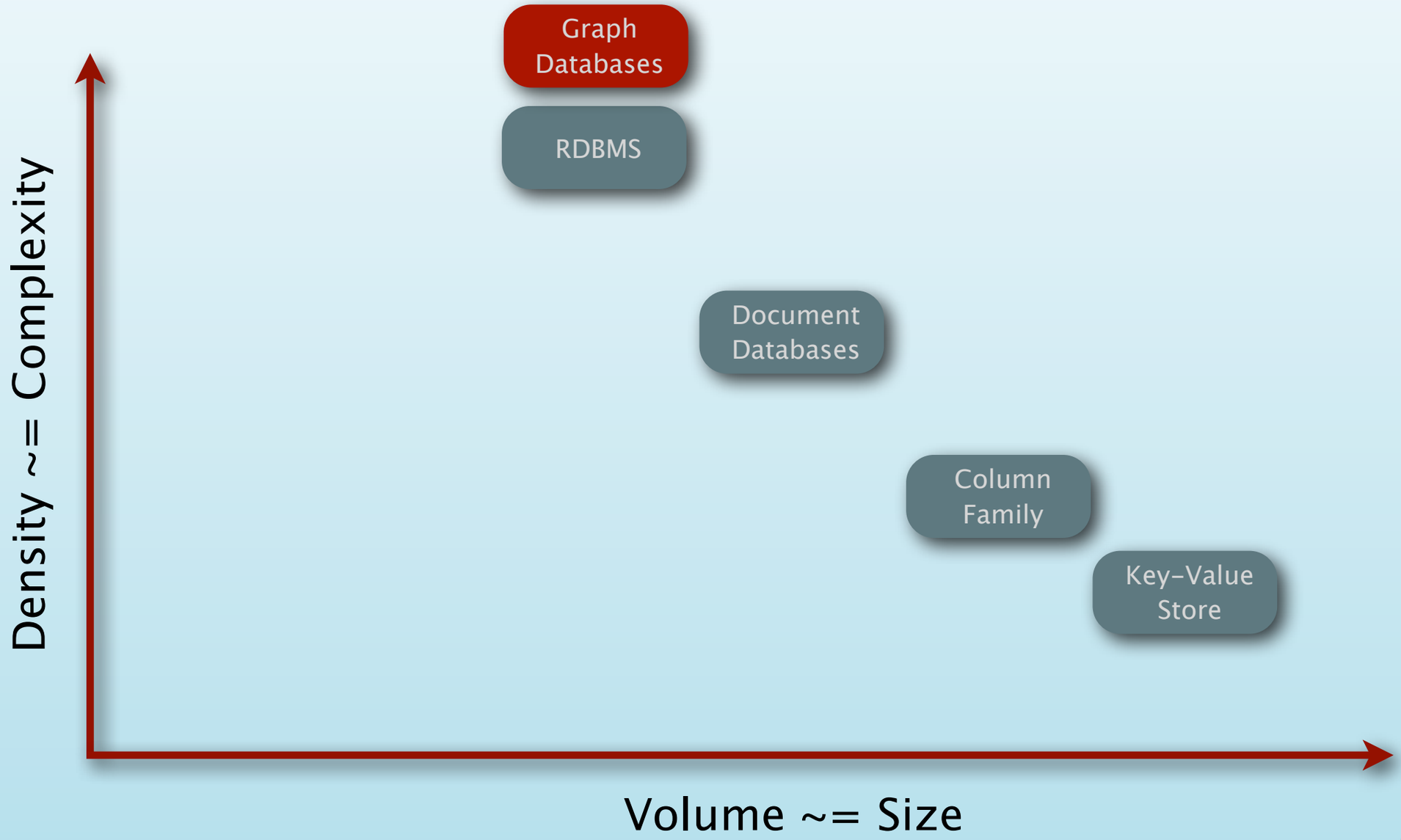
Living in a NOSQL World



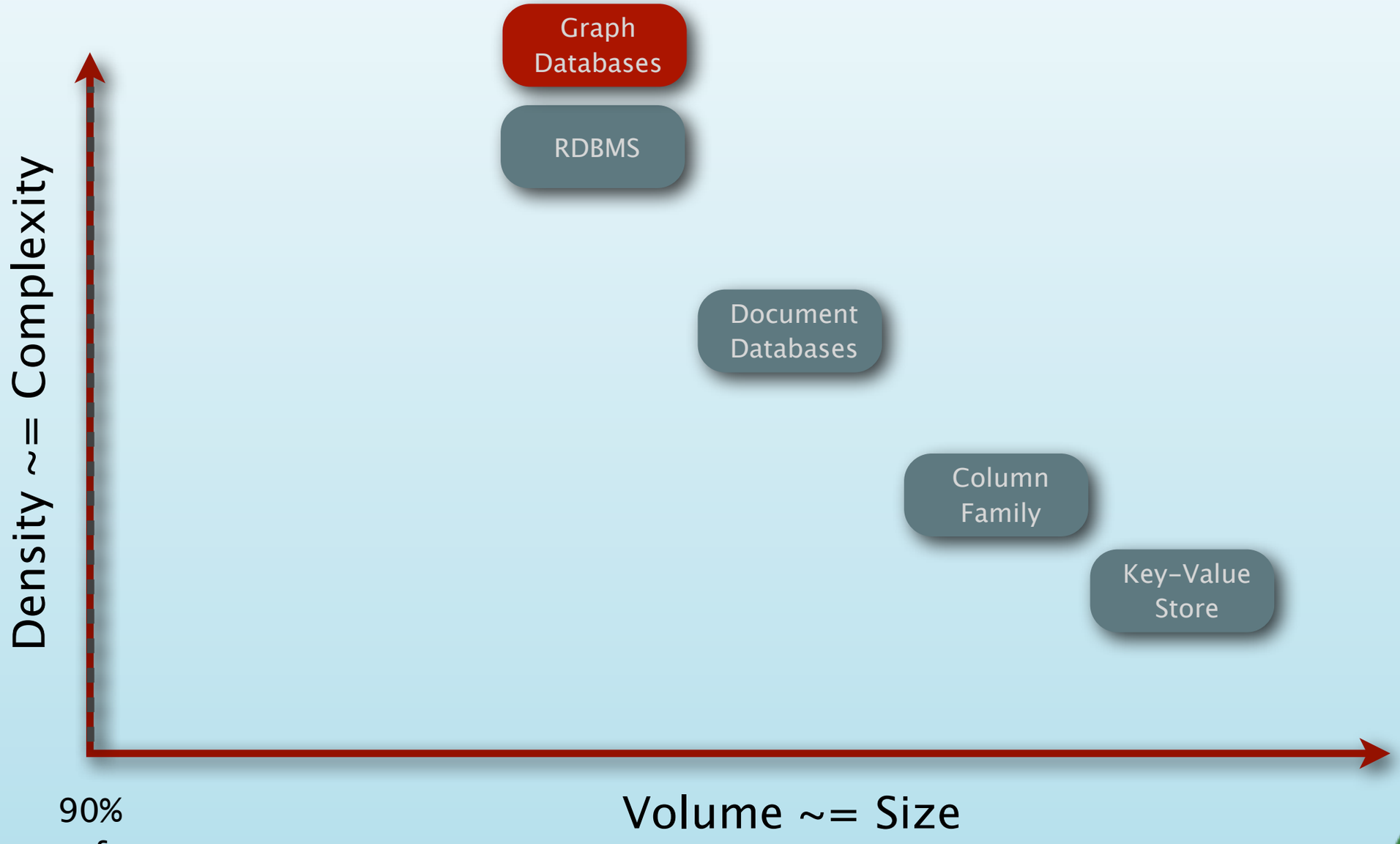
Living in a NOSQL World



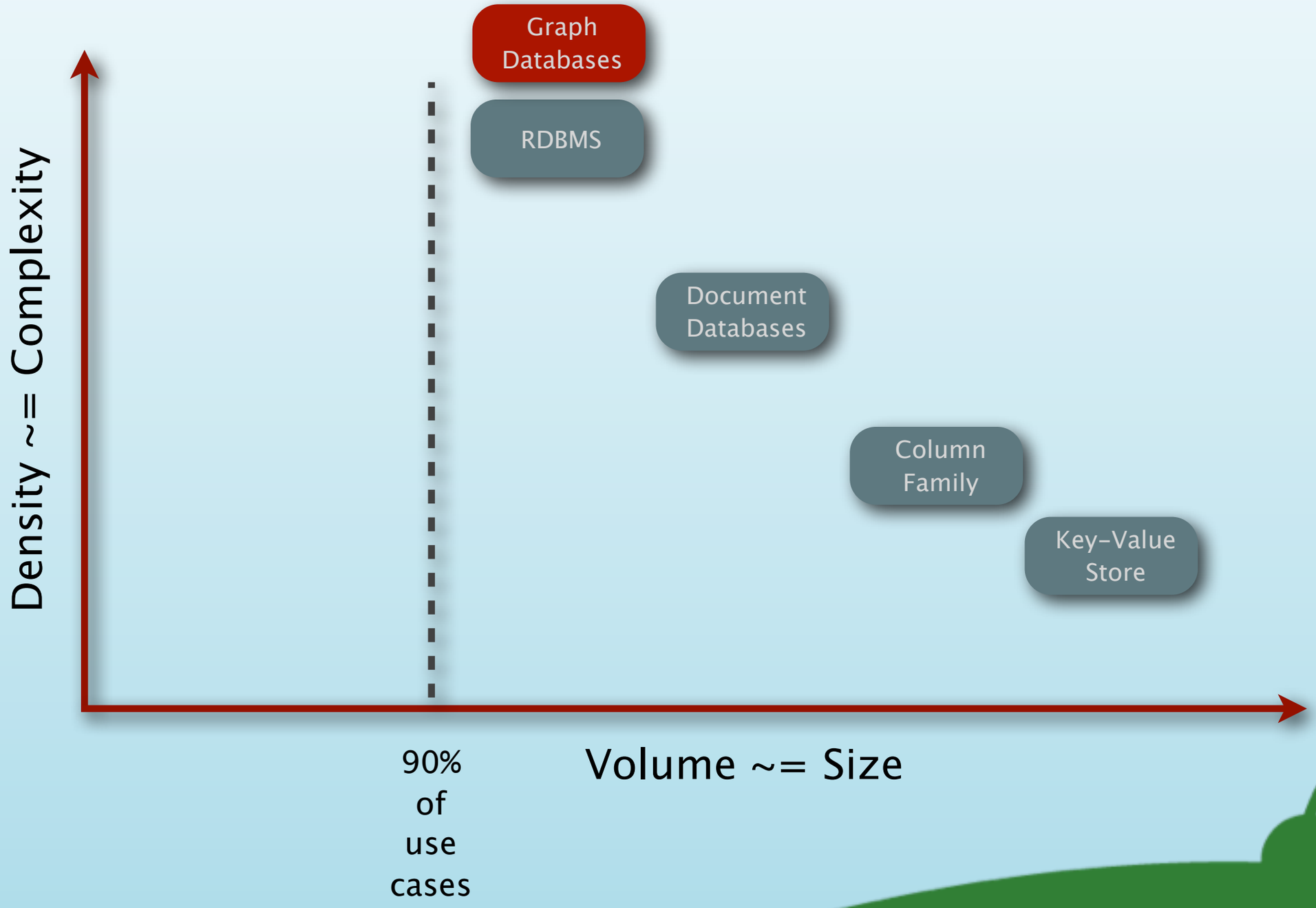
Living in a NOSQL World



Living in a NOSQL World



Living in a NOSQL World



Trends in BigData & NOSQL

1. increasing data size (big data)

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

2. increasingly connected data (graph data)

- for example, text documents to html

3. semi-structured data

- individualization of data, with common sub-set

4. architecture - a facade over multiple services

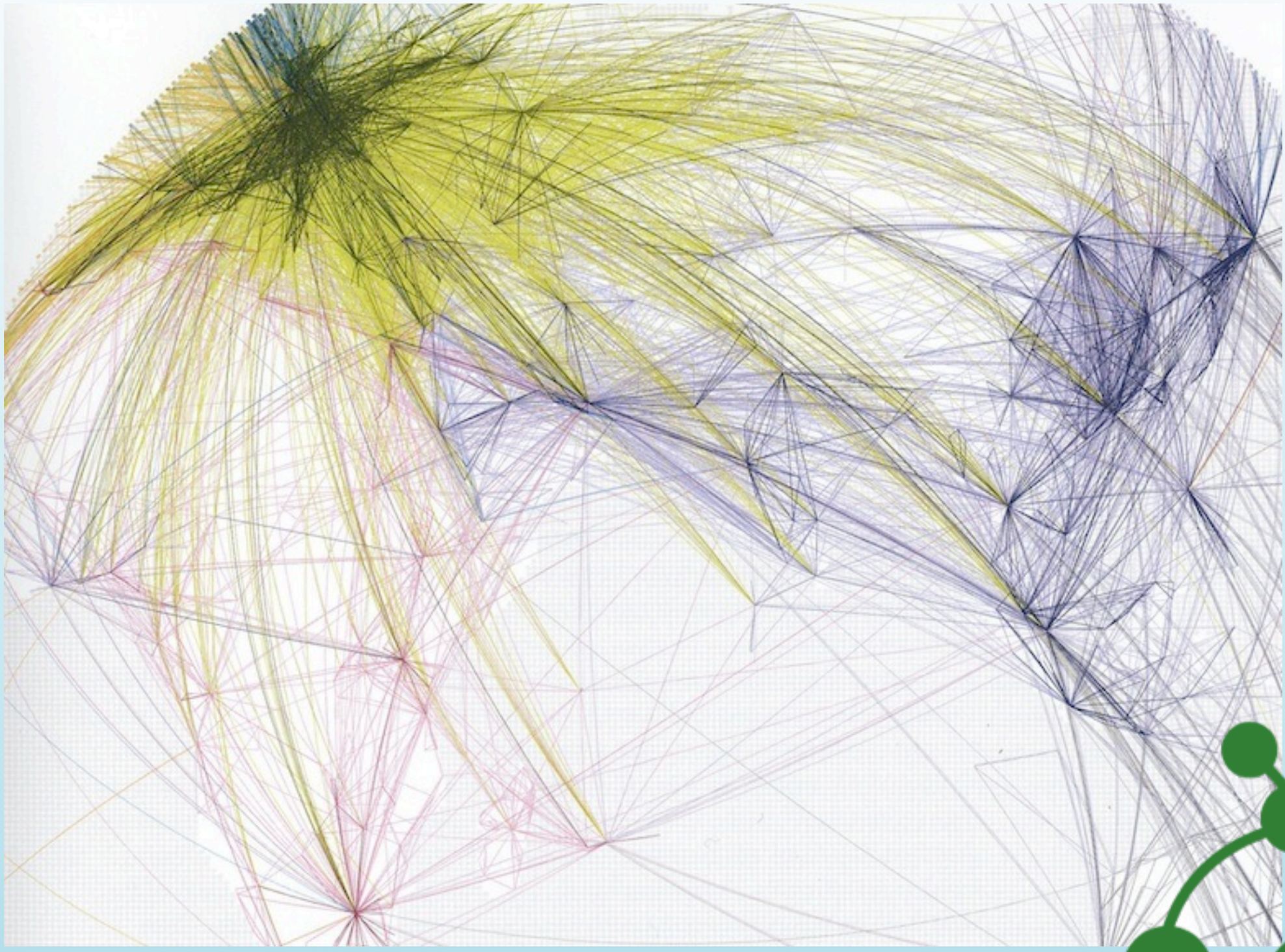
- from monolithic to modular, distributed applications

A Graph?

A Graph?

Yes, a graph

They are everywhere



Flight Patterns in Europe

Graphs Everywhere

- Relationships in
  - Politics, Economics, History, Science, Transportation
- Biology, Chemistry, Physics, Sociology
  - Body, Ecosphere, Reaction, Interactions
- Internet
  - Hardware, Software, Interaction
- Social Networks
  - Family, Friends
  - Work, Communities
  - Neighbours, Cities, Society

Good Relationships

- the world is rich, messy and related data
- relationships are as least as important as the things they connect
- Graphs = Whole > Σ parts
- complex interactions
- always changing, change of structures as well
- Graph: Relationships are part of the data
- RDBMS: Relationships part of the fixed schema

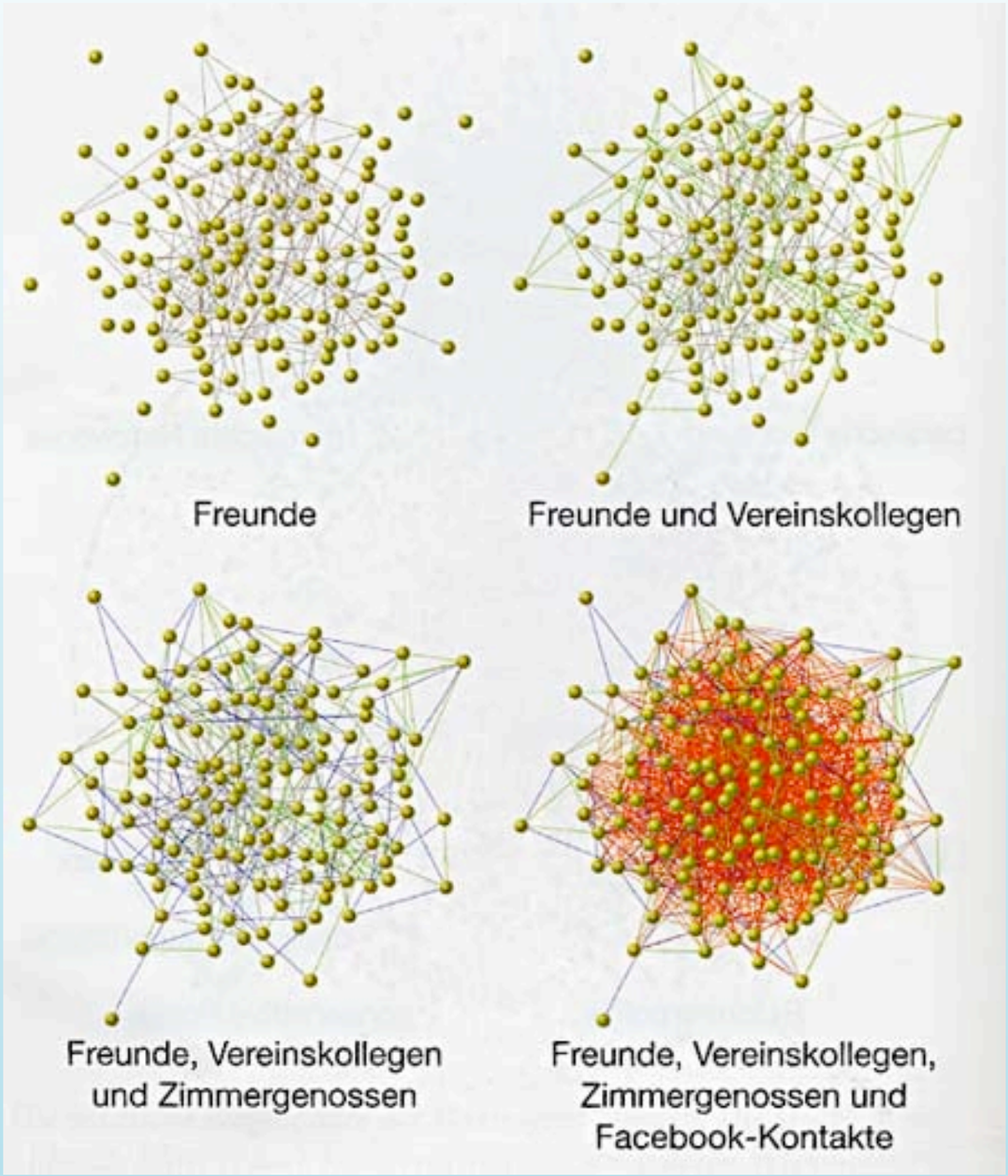
Questions and Answers

- Complex Questions
- Answers lie between the lines (things)
- Locality of the information
- Global searches / operations very expensive
- constant query time, regardless of data volume

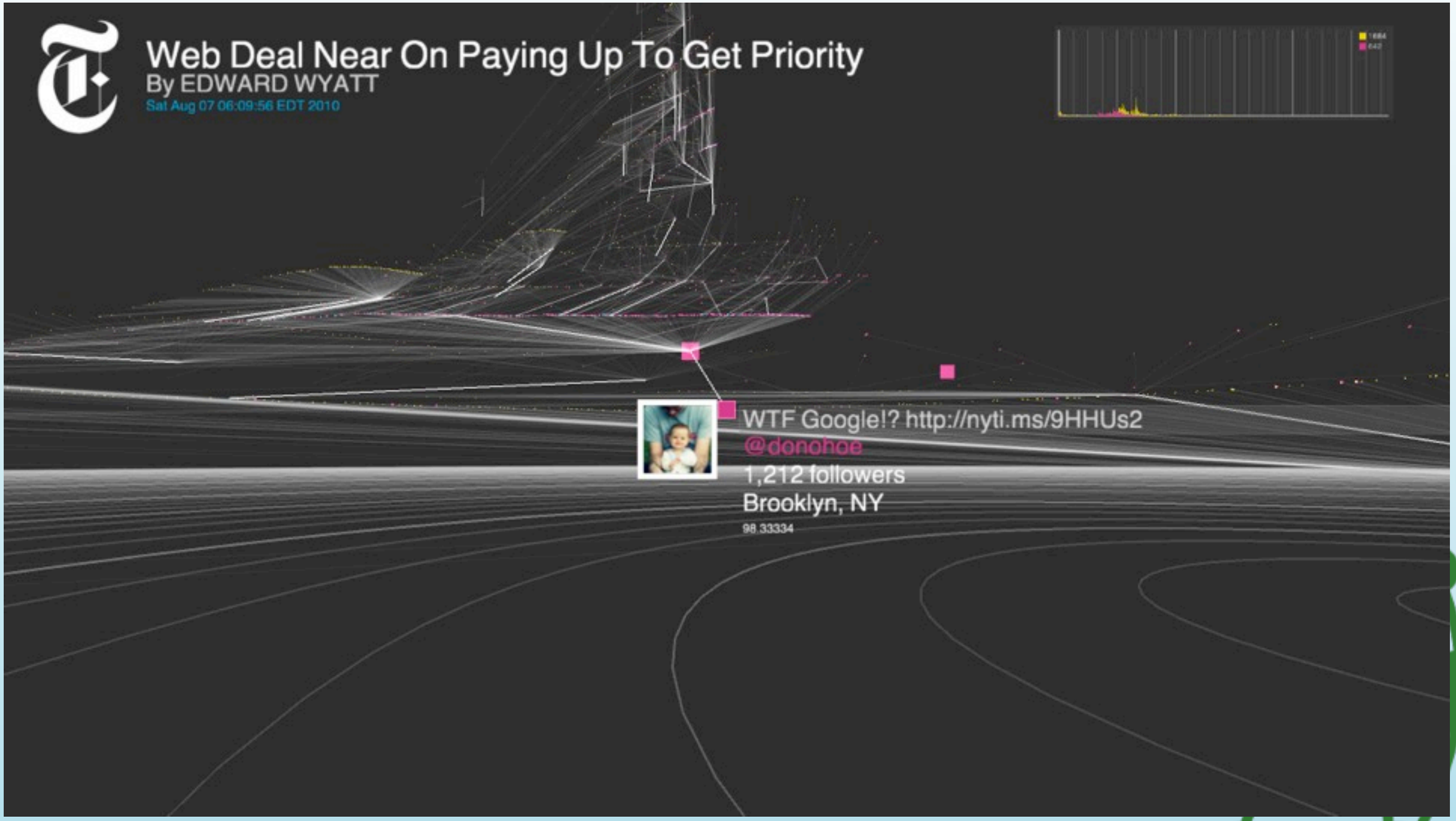
Categories ?

- Categories == Classes, Trees ?
- What if more than one category fits?
- Tags
- Categories vi relationships like „IS_A“
- any number, easy change
- „virtual“ Relationships - Traversals
- Category dynamically derived from queries

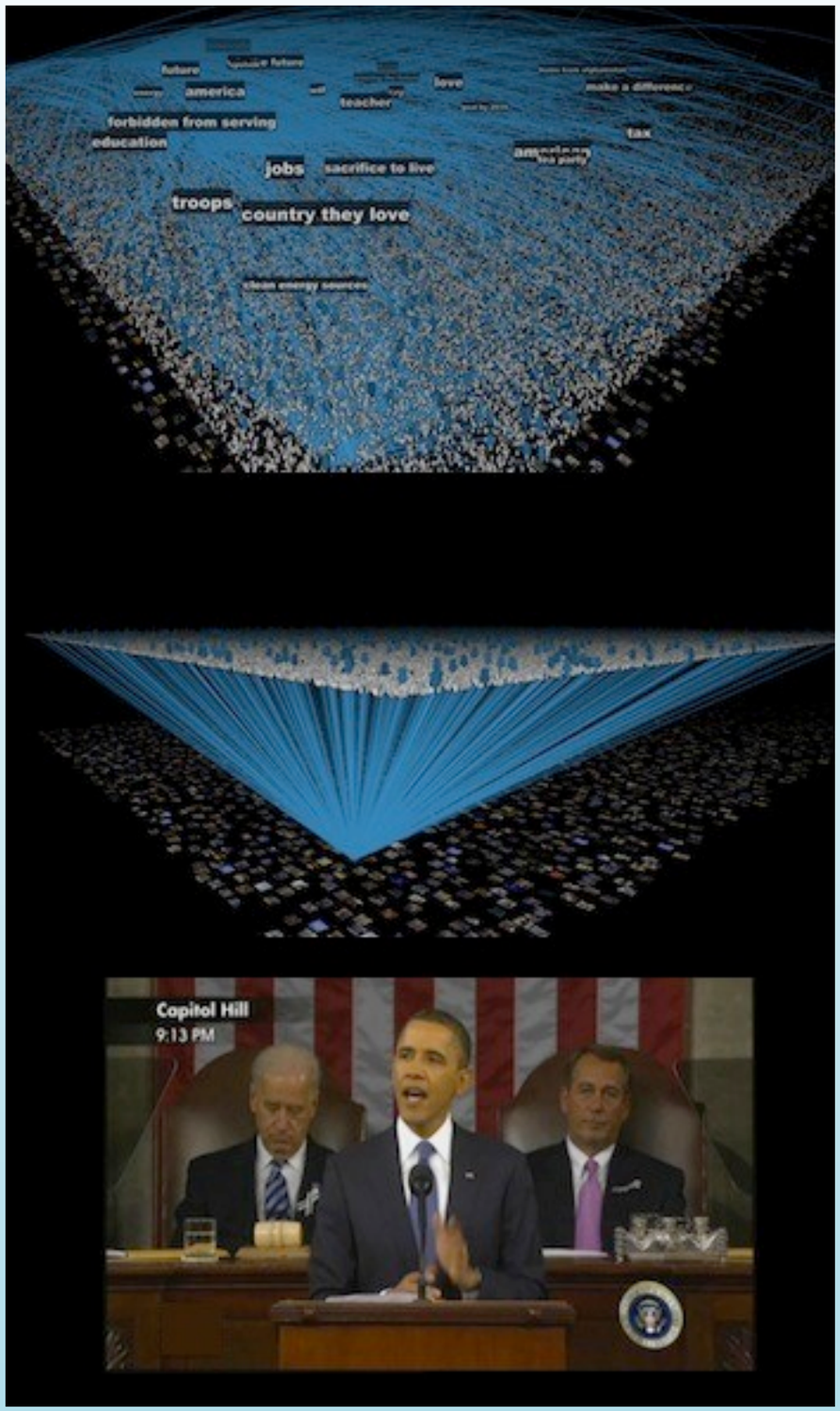
Fowler & Christakis „Connected“



New York Times R&D „Cascade“



Deb Roy - MIT & Bluefin Labs

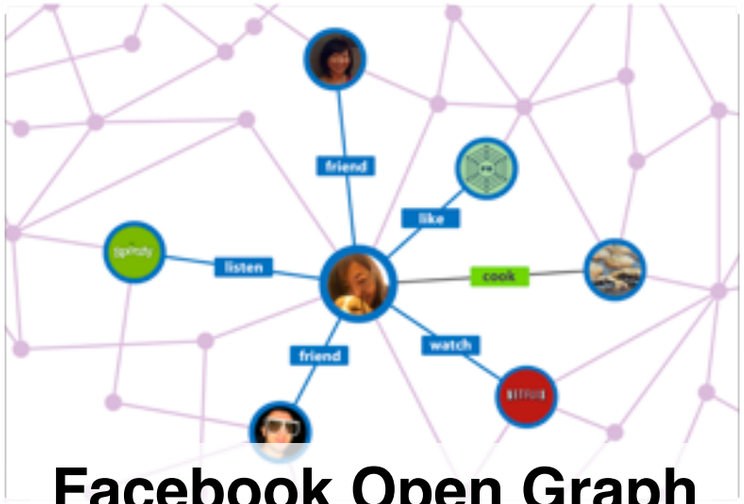


„Birth of a Word“ TED Talk

Researches Social Reactions
to (Media) Events

Everyone is talking about graphs...

Everyone is talking about graphs...



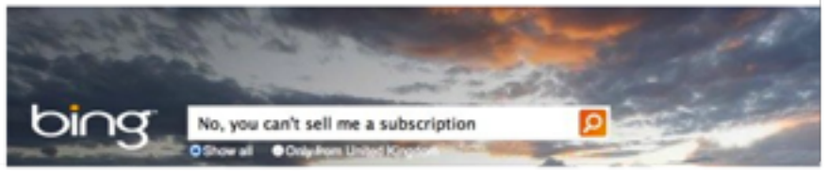
Facebook Open Graph

Google Just Got A Whole Lot Smarter, Launches Its Knowledge Graph

FREDERIC LARDINOIS

Bing one-ups knowledge graph, hires Encyclopaedia Britannica to supply results

By Daniel Cooper posted Jun 8th 2012 3:02PM



Why the Interest Graph Is a Marketer's Best Friend

1 day ago by **Nadim Hossain**

5

Everyone is talking about graphs...

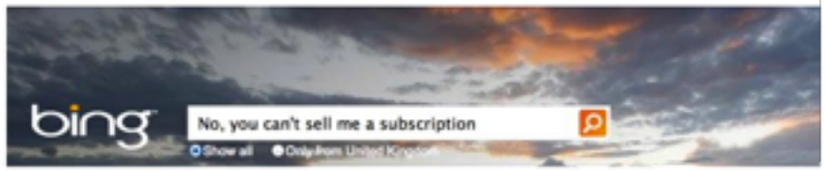
Facebook Open Graph

Google Just Got A Whole Lot Smarter, Launches Its Knowledge Graph

FREDERIC LARDINOIS

Bing one-ups knowledge graph, hires Encyclopaedia Britannica to supply results

By Daniel Cooper posted Jun 8th 2012 3:02PM



Why the Interest Graph Is a Marketer's Best Friend

1 day ago by **Nadim Hossain**

5

Neo4j
The World's Leading Graph Database

Each of us has not only one graph, but many!

Graph DB 101

A graph database...

A graph database...

NO: not for charts & diagrams, or vector artwork

A graph database...

NO: not for charts & diagrams, or vector artwork

YES: for storing data that is structured as a graph

A graph database...

NO: not for charts & diagrams, or vector artwork

YES: for storing data that is structured as a graph

remember linked lists, trees?

A graph database...

NO: not for charts & diagrams, or vector artwork

YES: for storing data that is structured as a graph

remember linked lists, trees?

graphs are the general-purpose data structure

A graph database...

NO: not for charts & diagrams, or vector artwork

YES: for storing data that is structured as a graph

remember linked lists, trees?

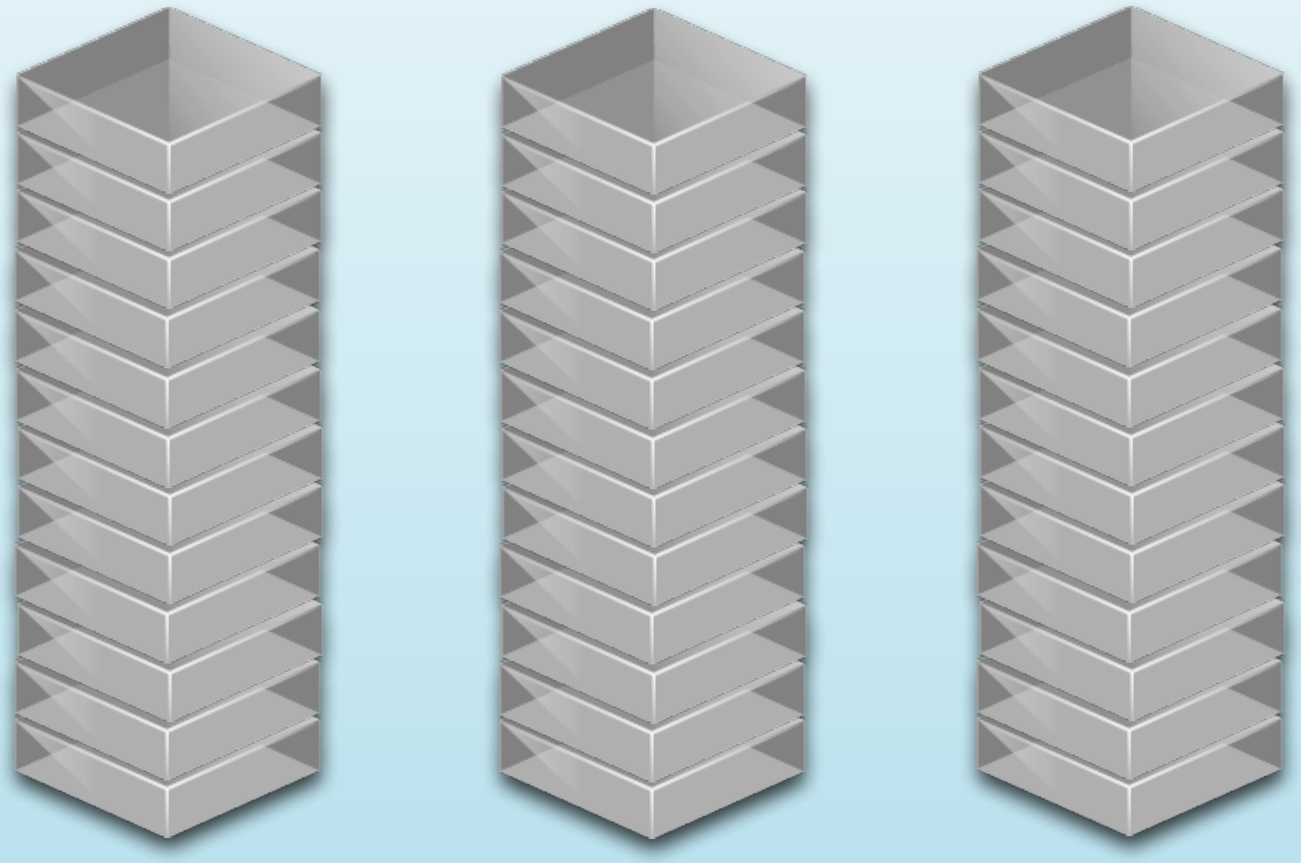
graphs are the general-purpose data structure

"A relational database may tell you the average age of everyone in this session,

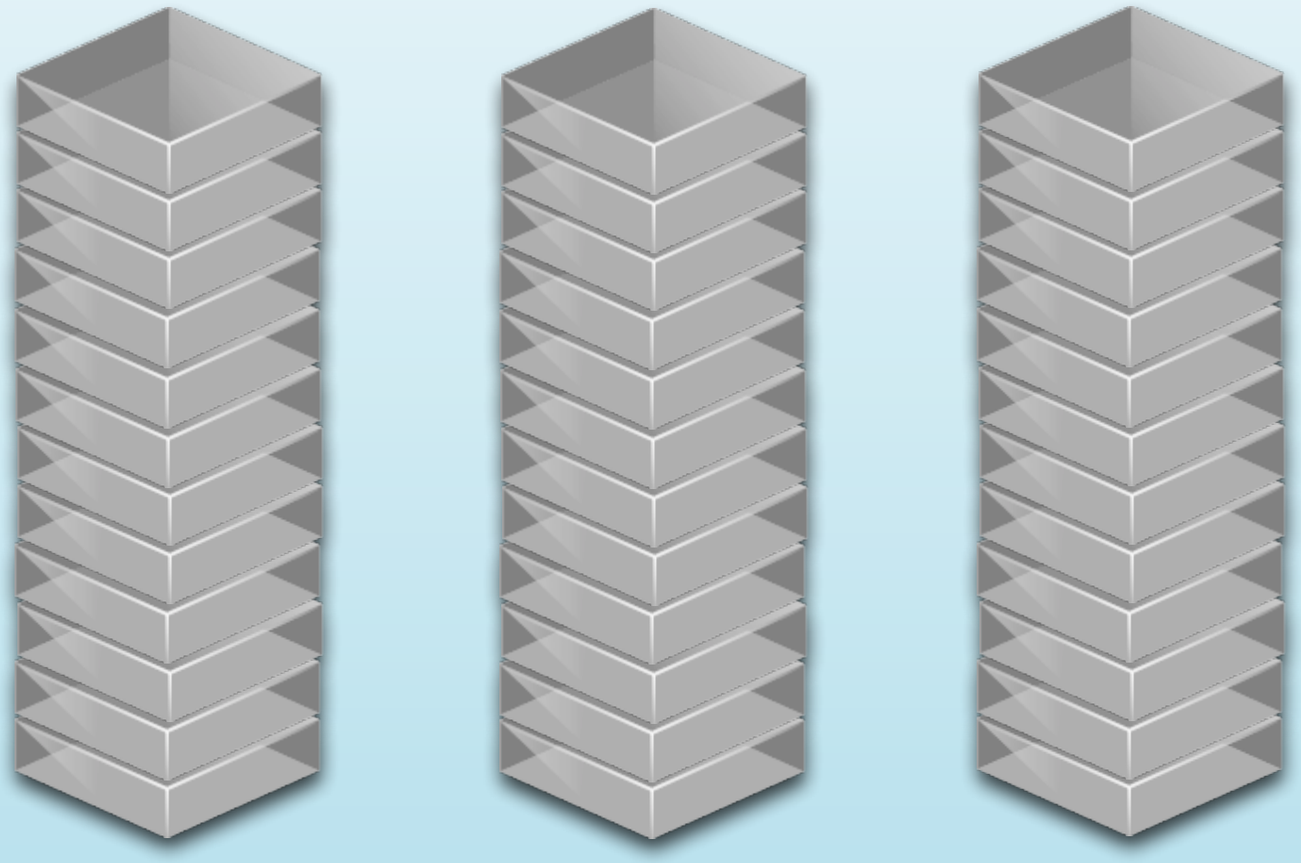
but a graph database will tell you who is most likely to buy you a beer."

You know relational

You know relational

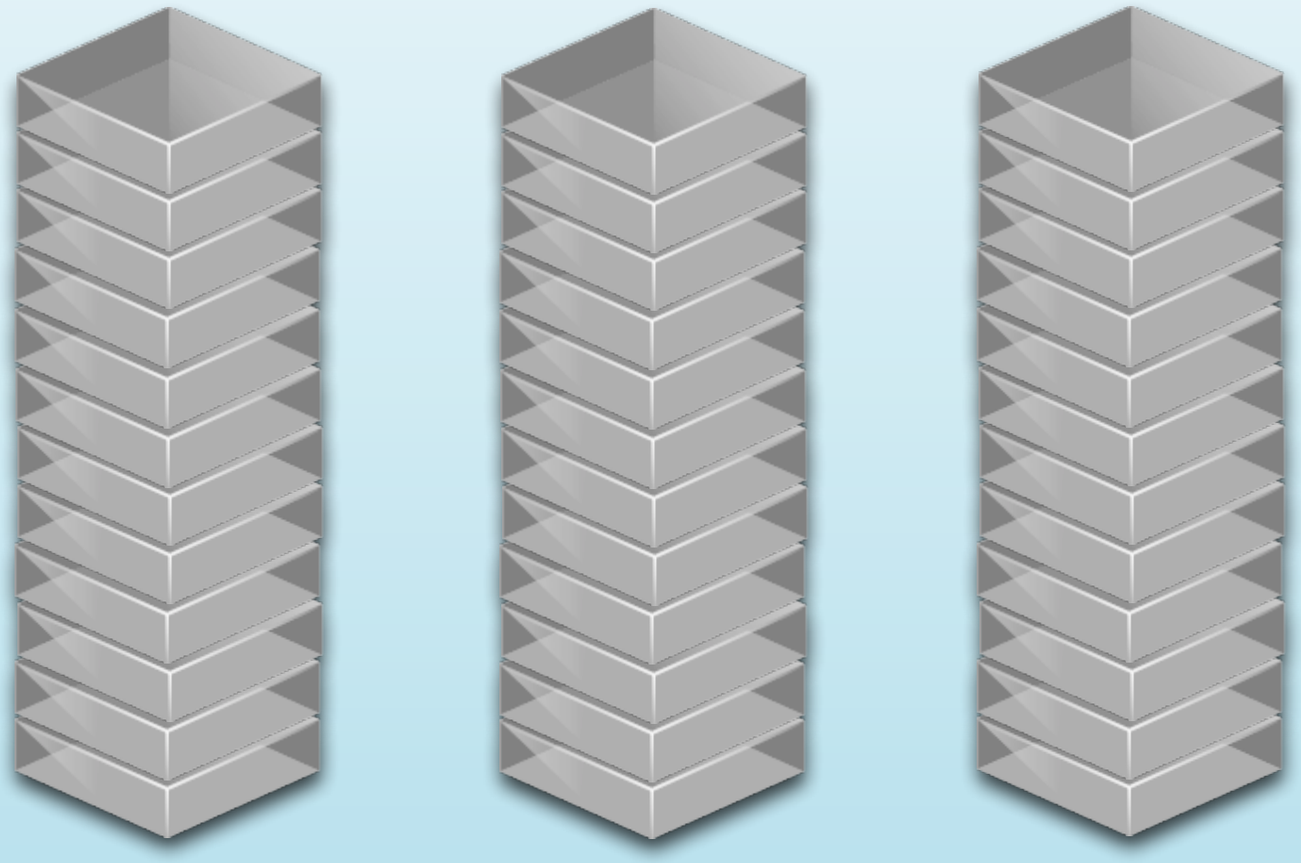


You know relational



foo

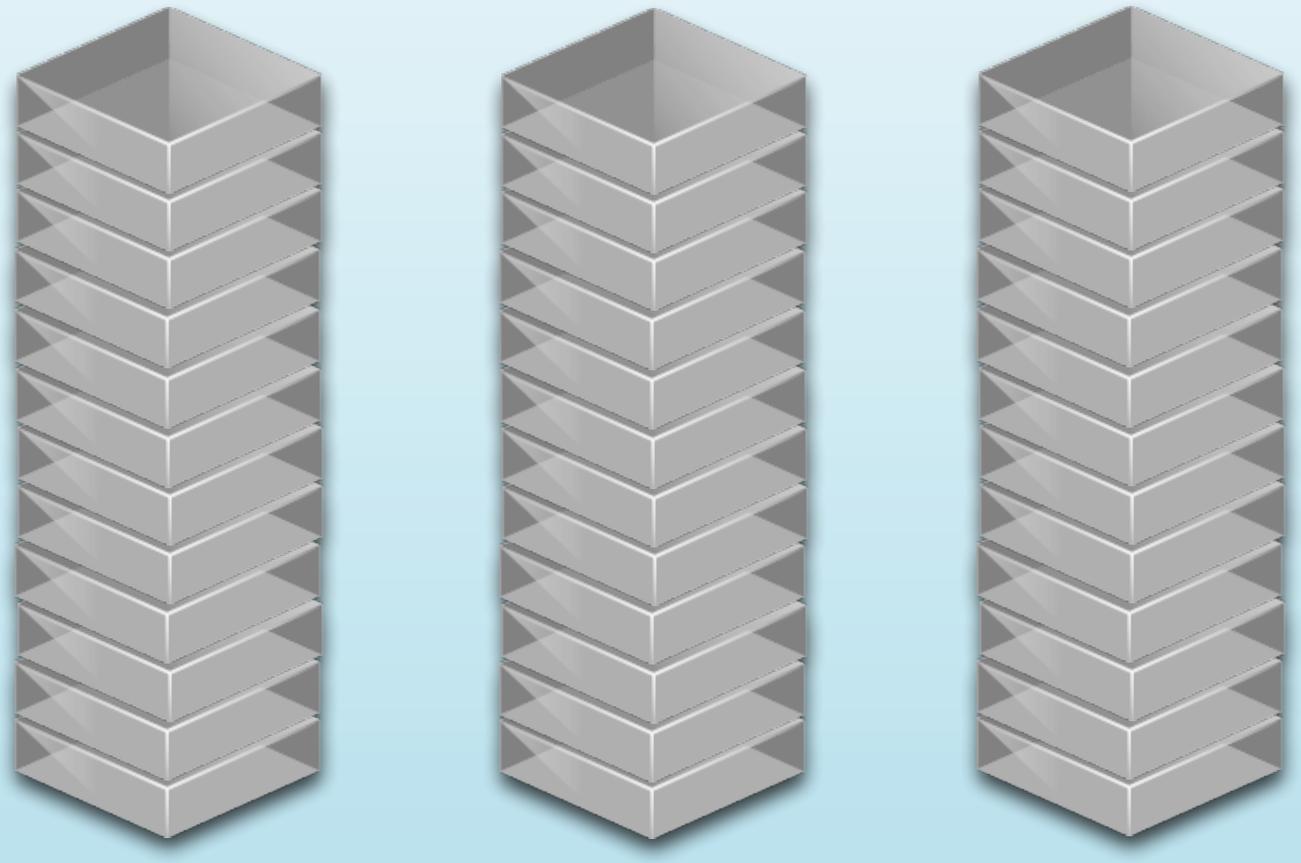
You know relational



foo

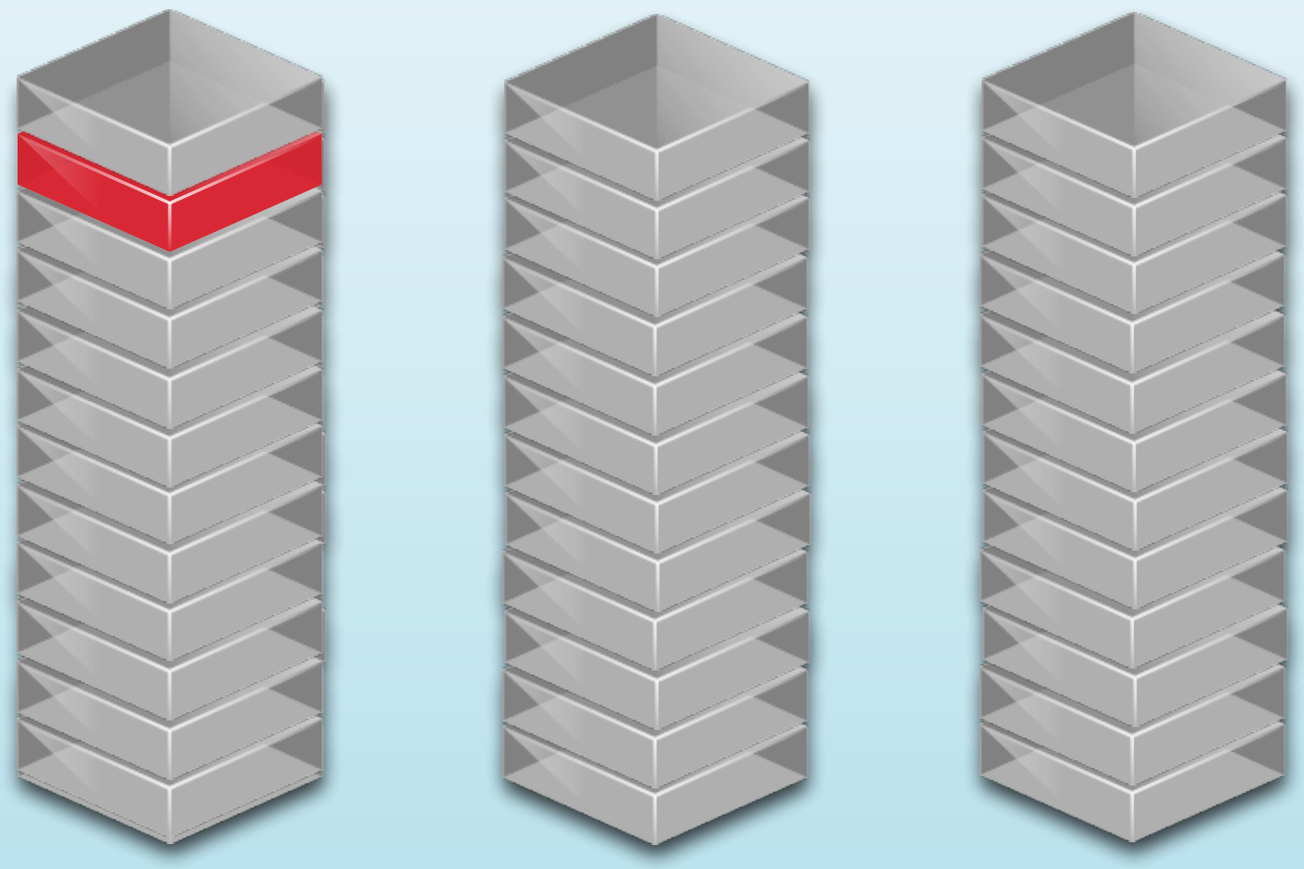
bar

You know relational



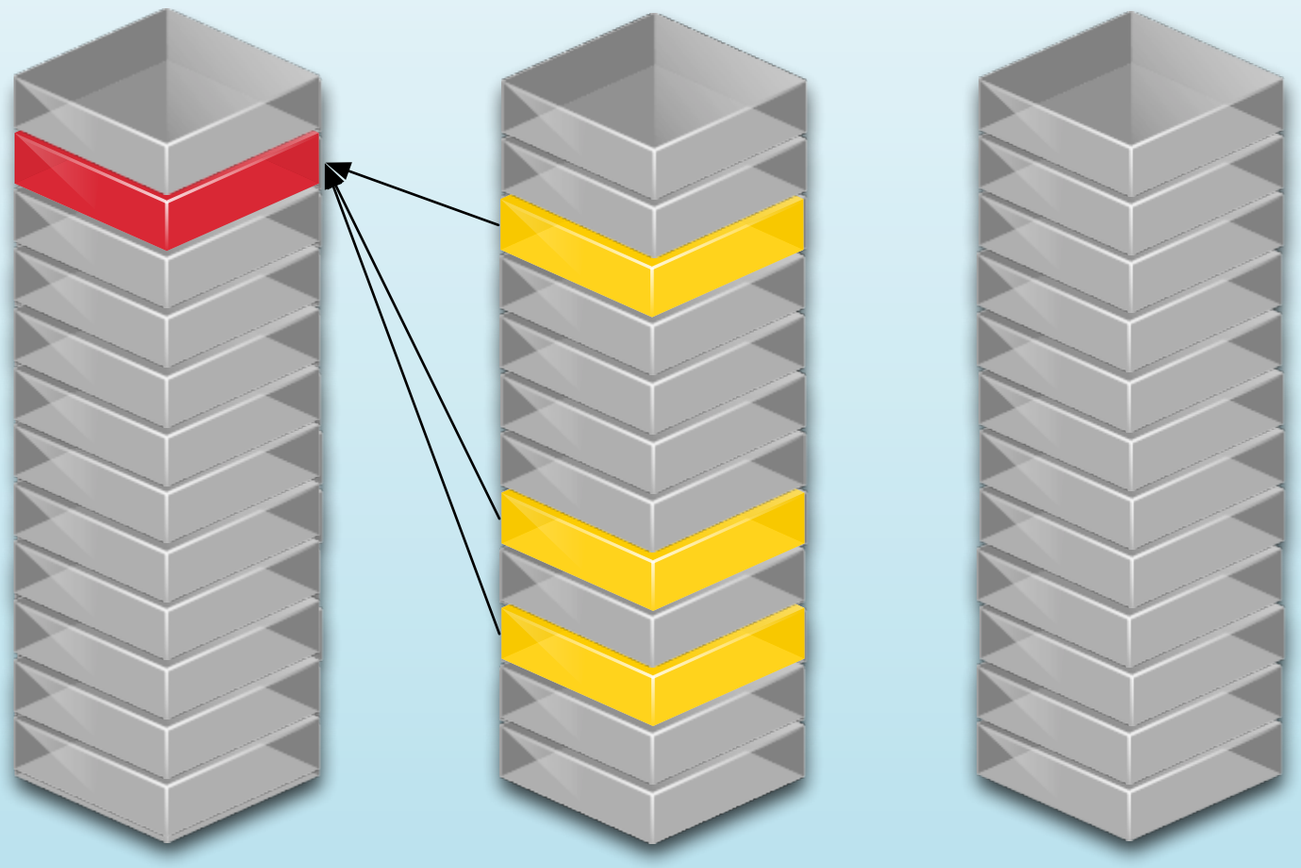
`foo` `foo_bar` `bar`

You know relational



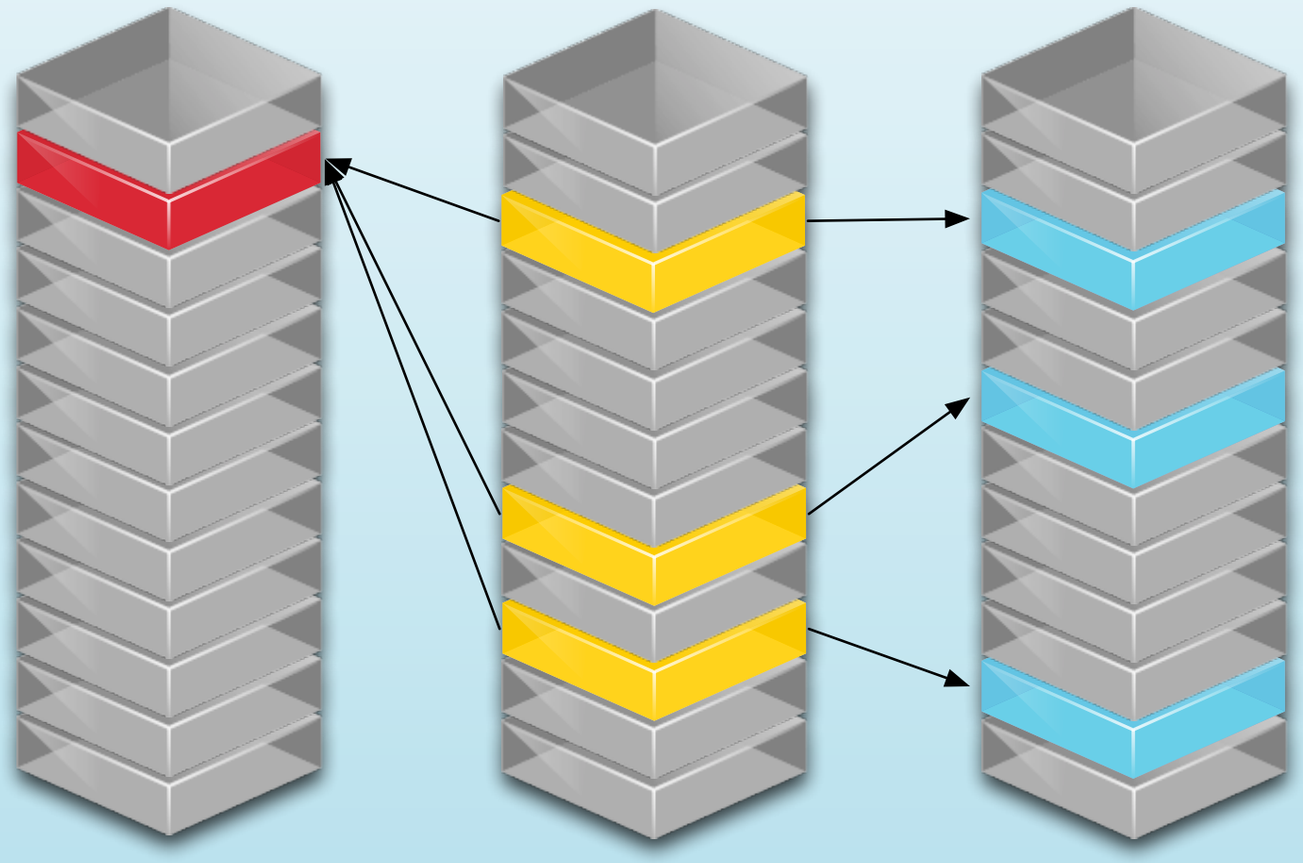
`foo` `foo_bar` `bar`

You know relational



foo foo_bar bar

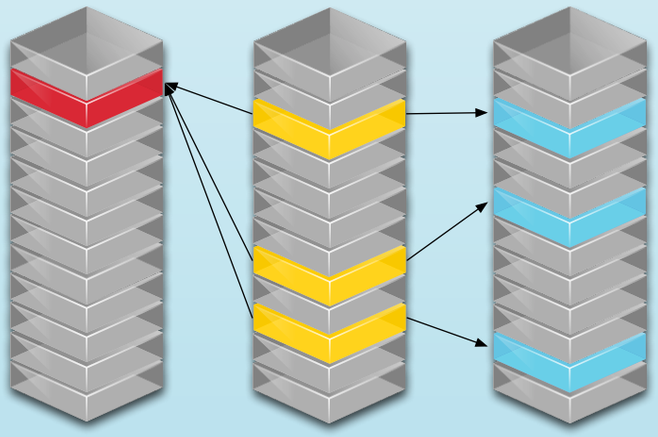
You know relational



foo foo_bar bar

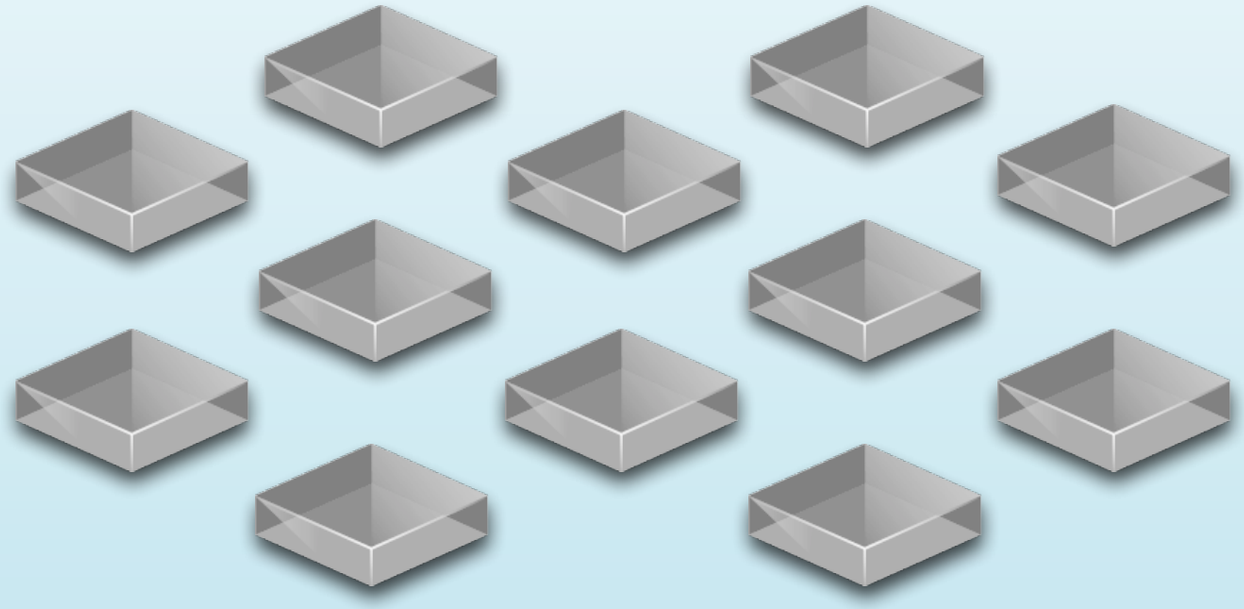
You know relational

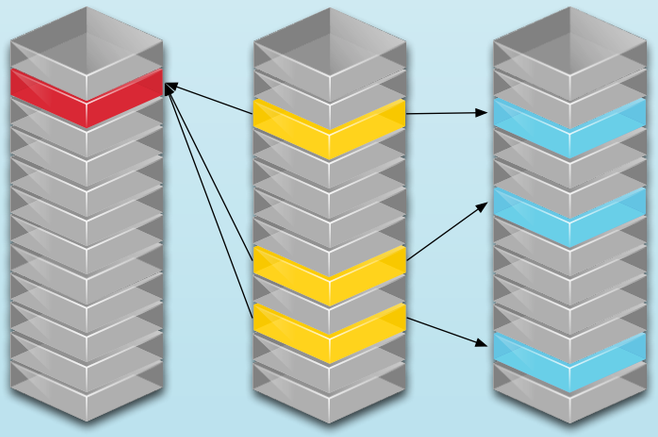
now consider relationships...



You know relational

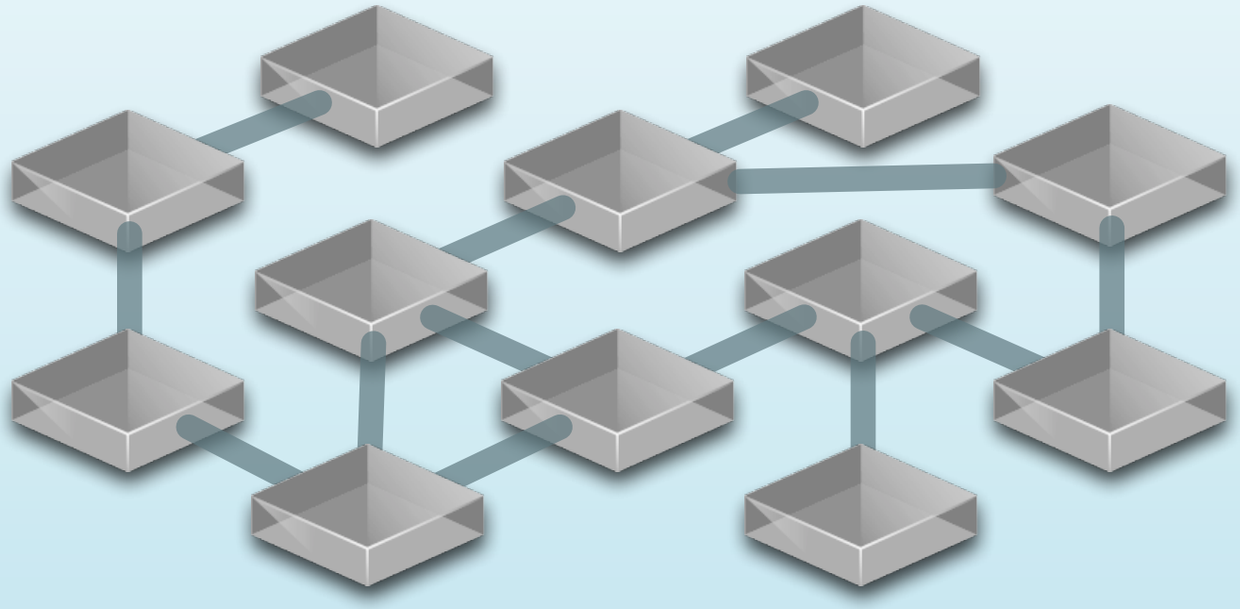
now consider relationships...

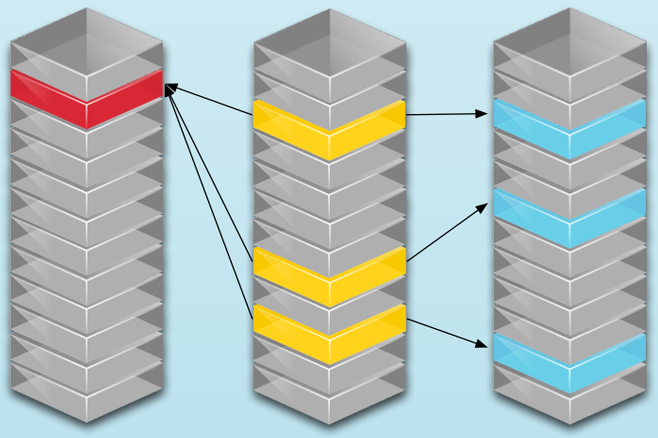




You know relational

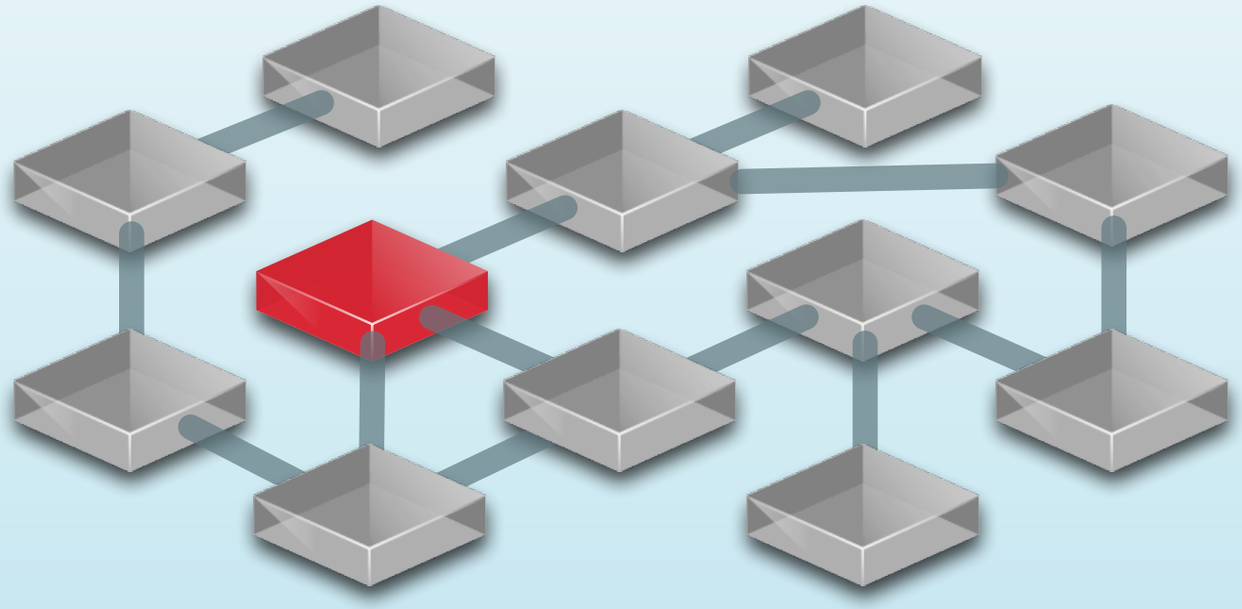
now consider relationships...

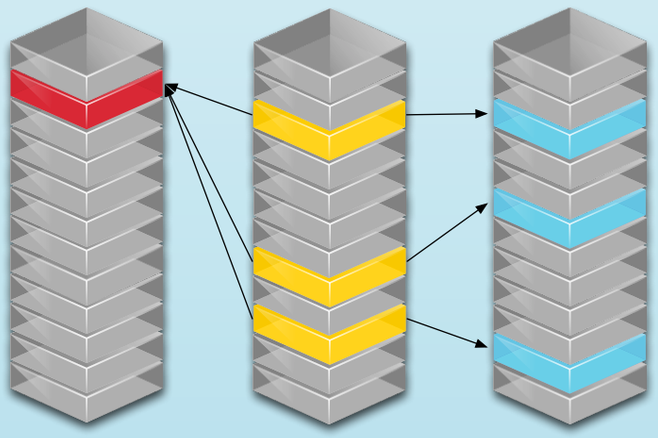




You know relational

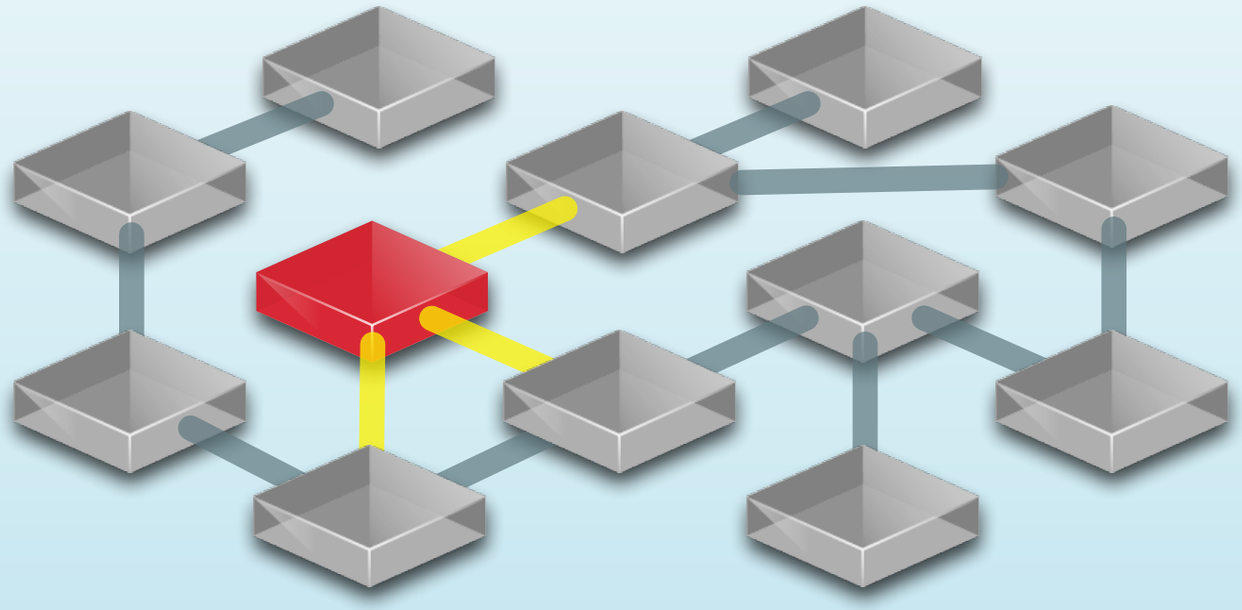
now consider relationships...

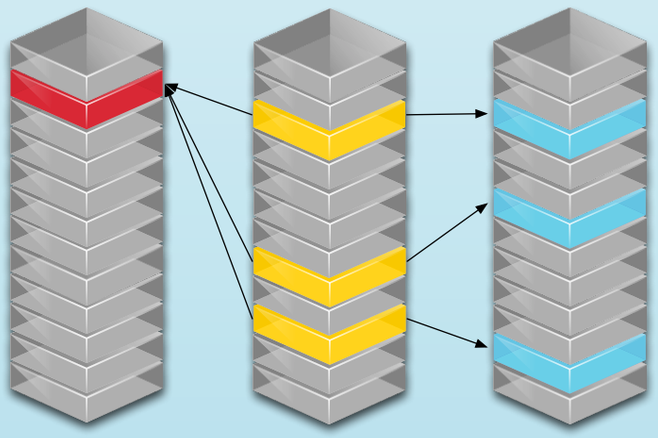




You know relational

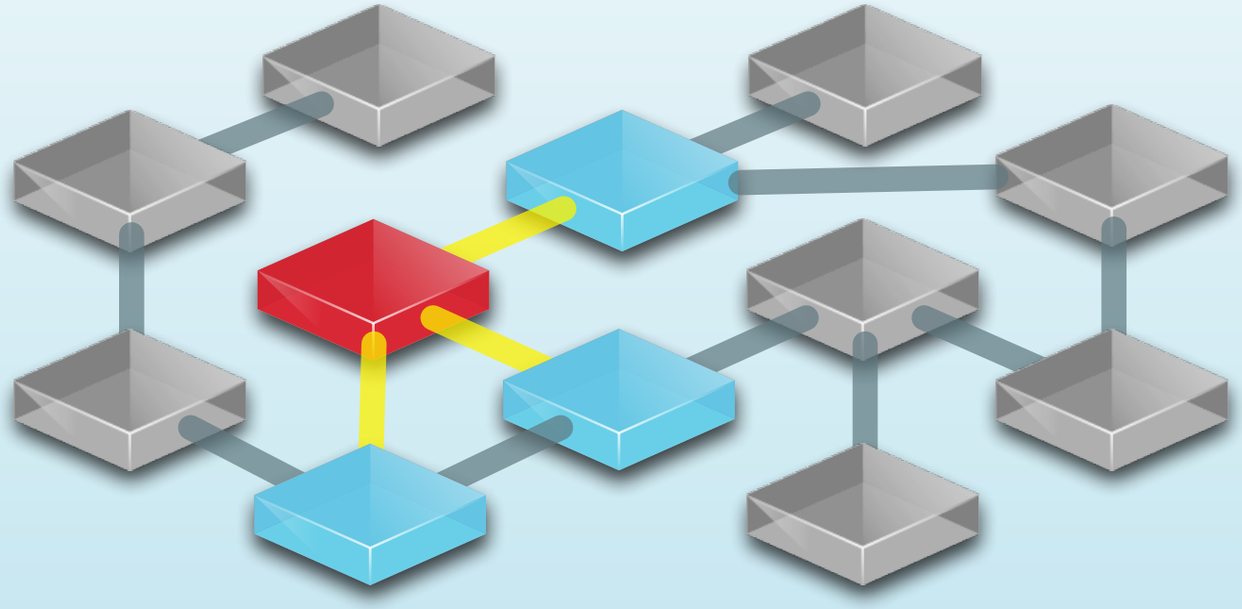
now consider relationships...

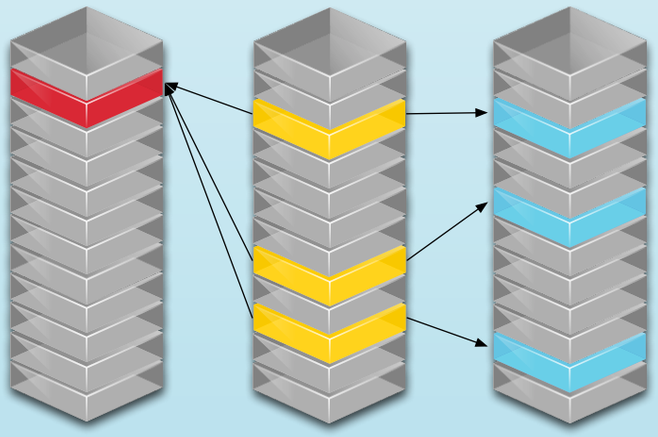


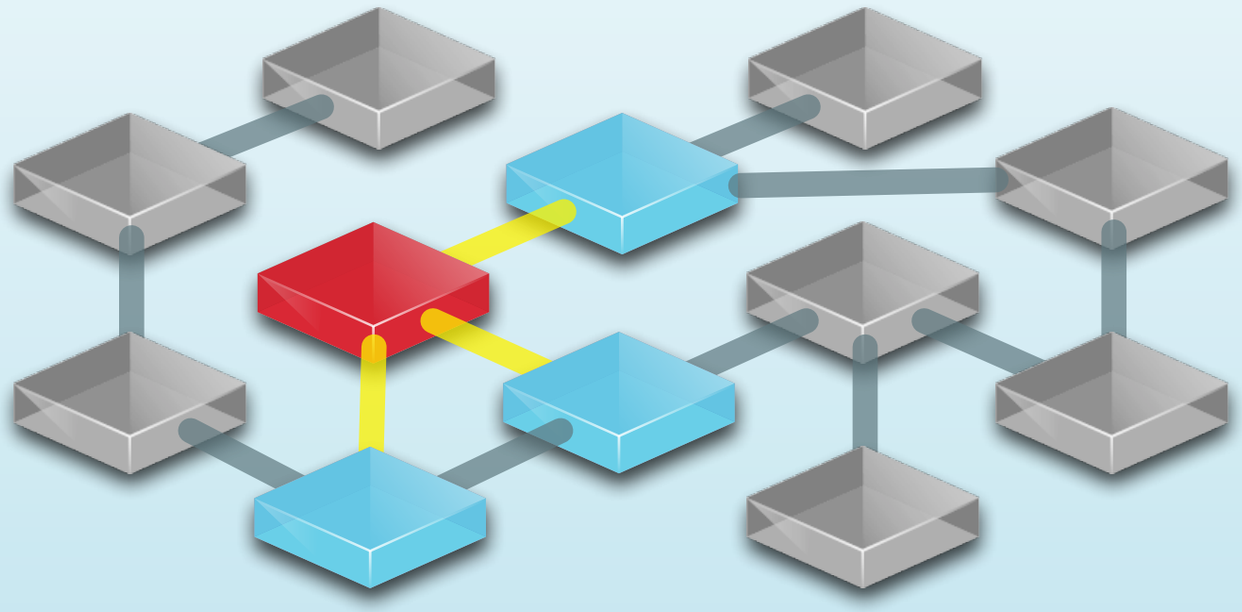


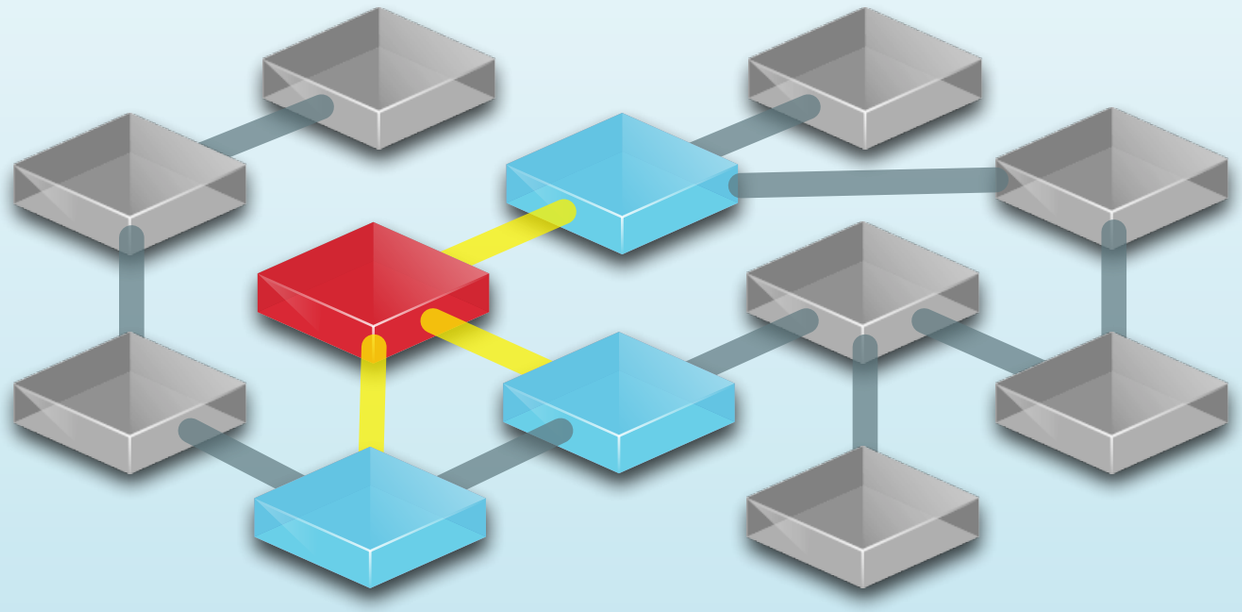
You know relational

now consider relationships...

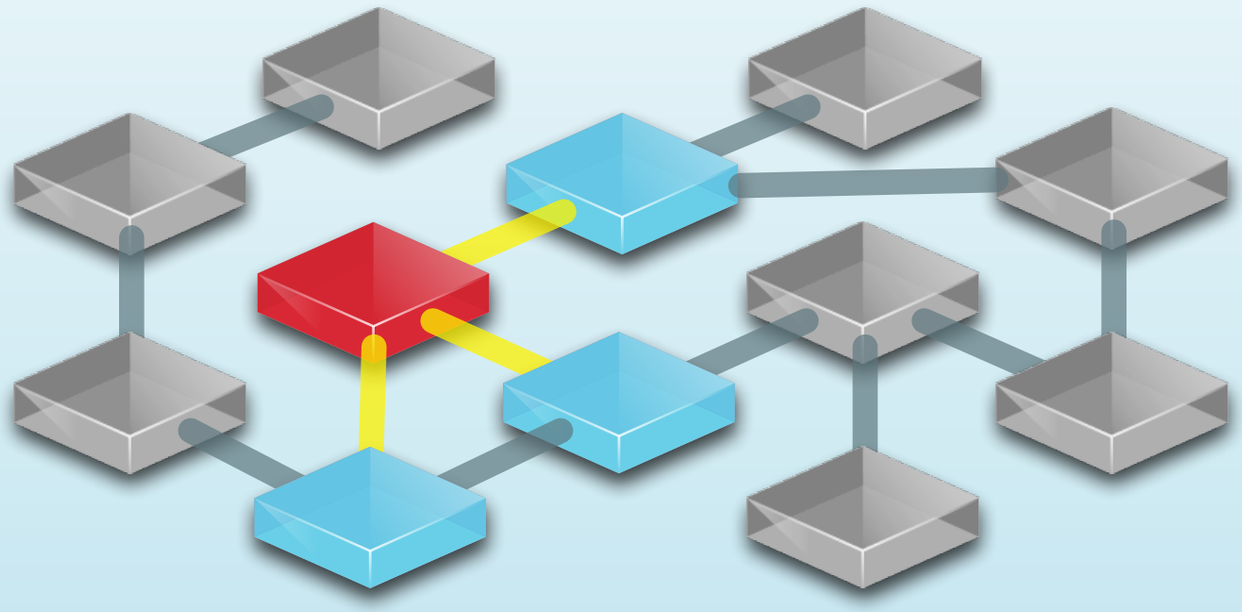




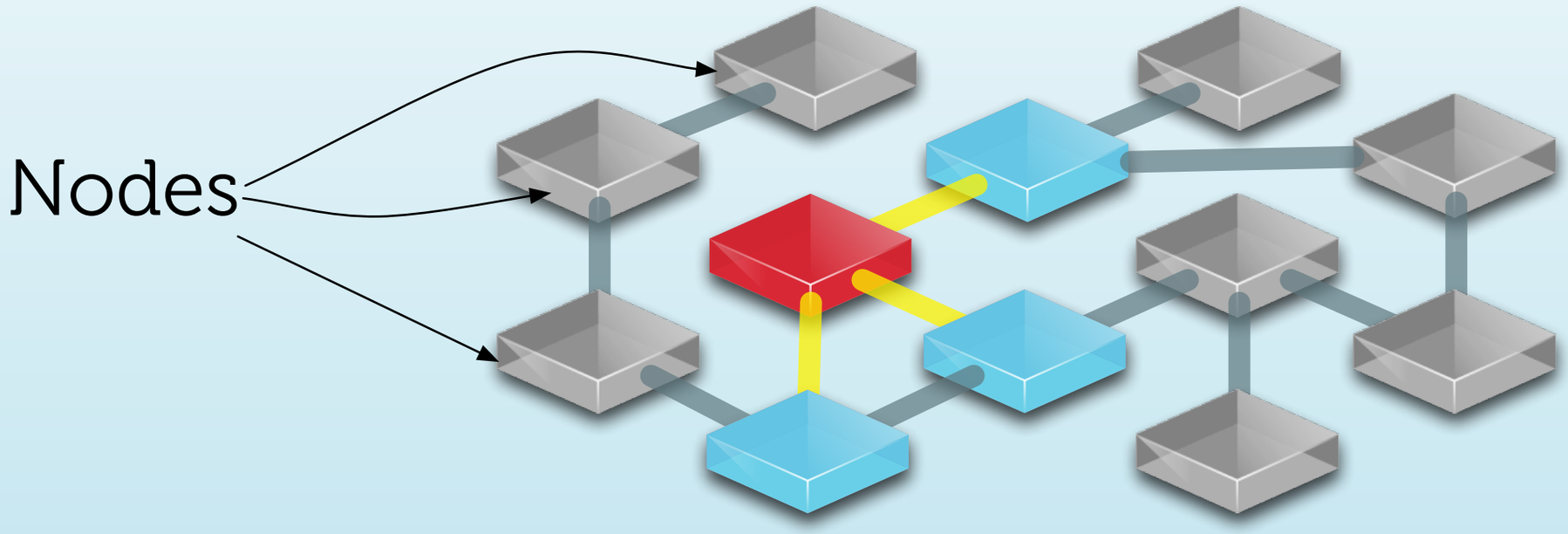




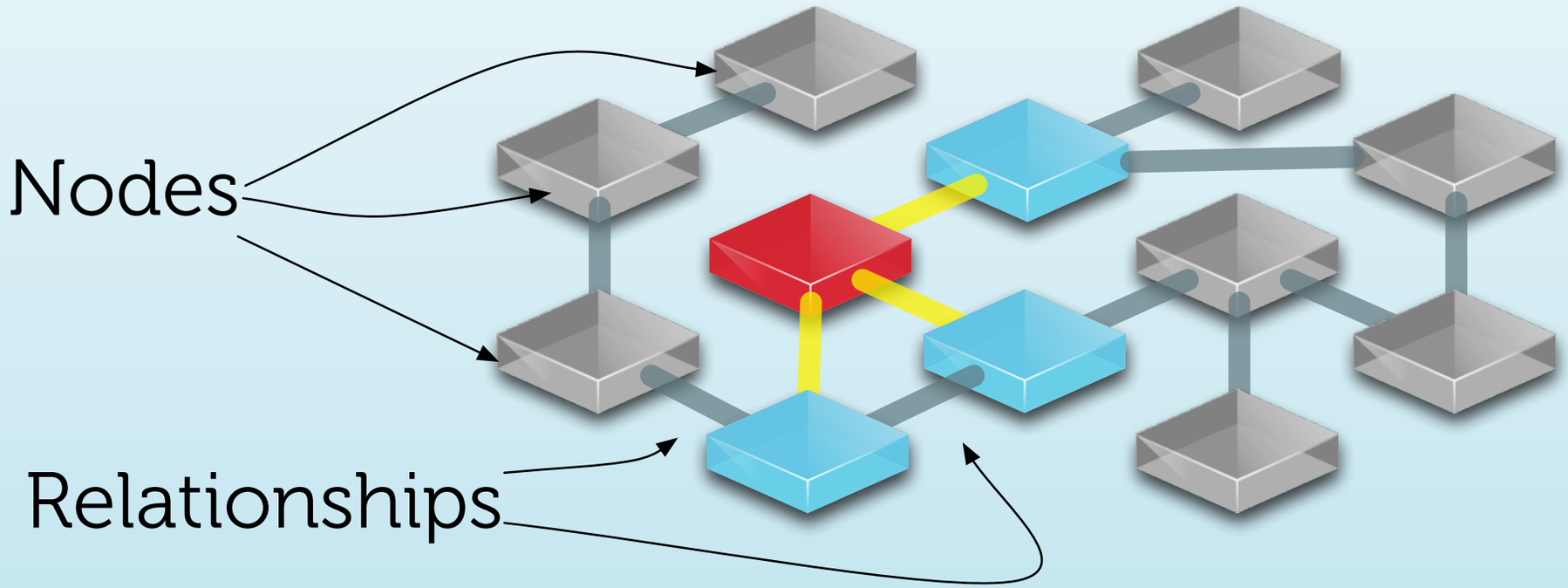
We're talking about a Property Graph



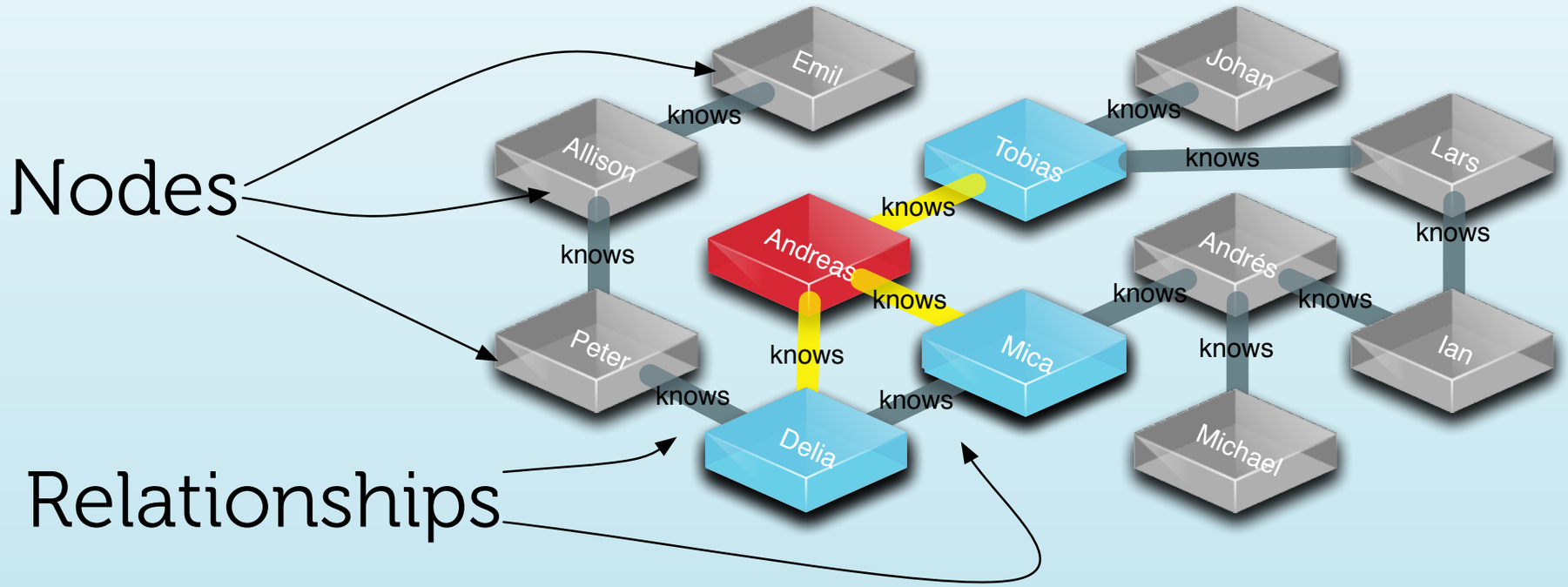
We're talking about a Property Graph



We're talking about a Property Graph

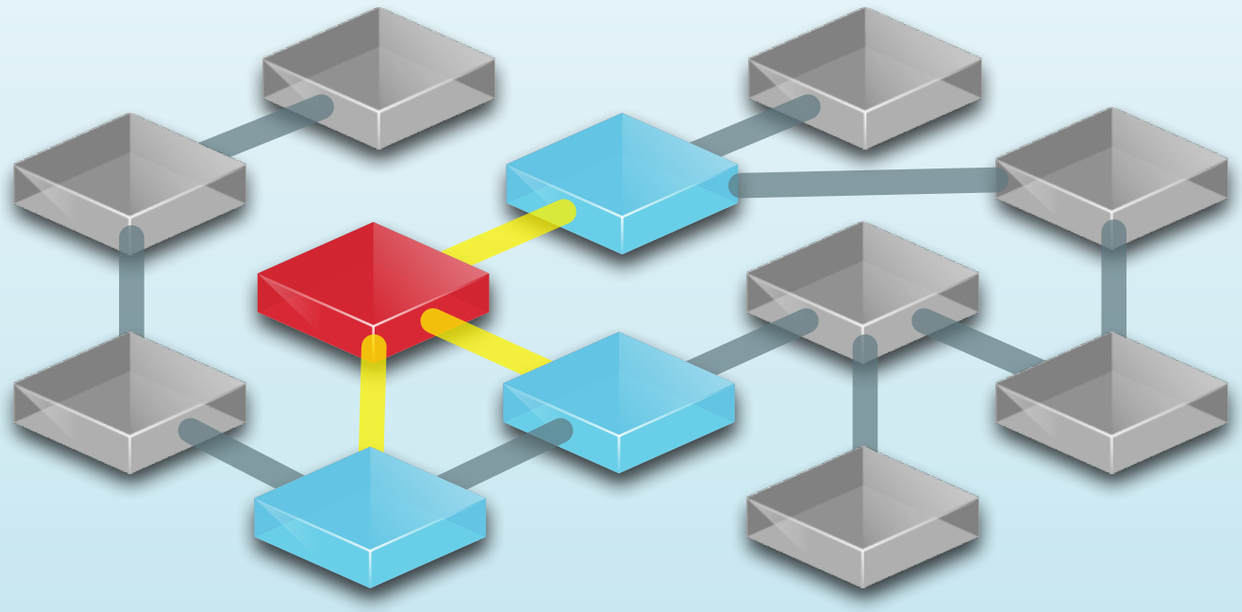


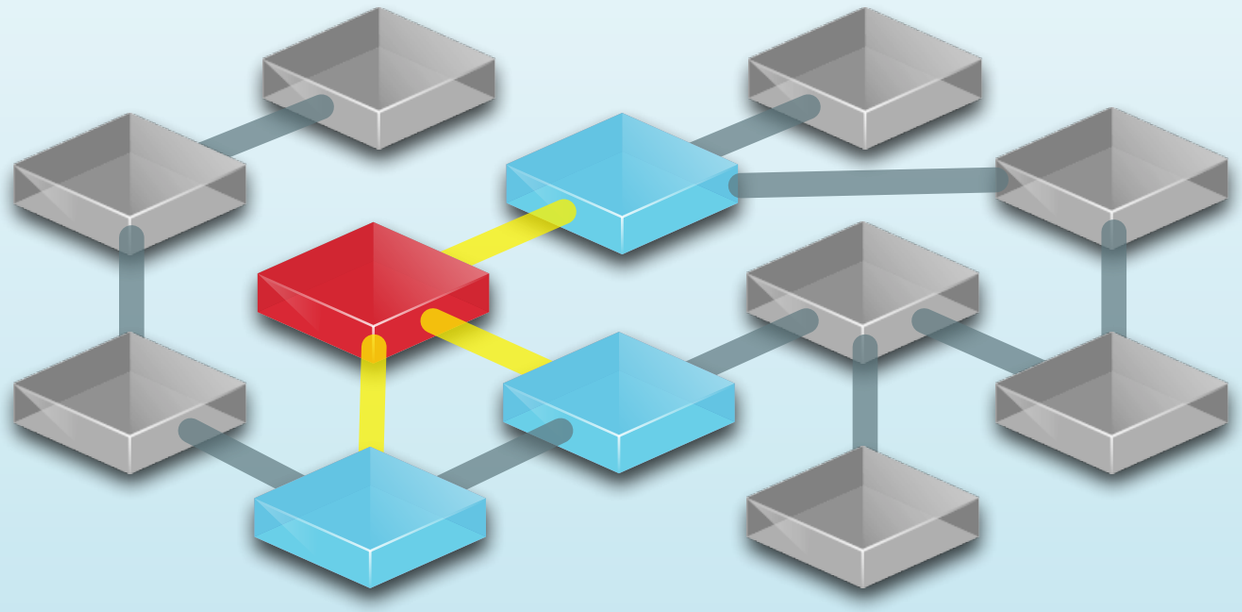
We're talking about a Property Graph



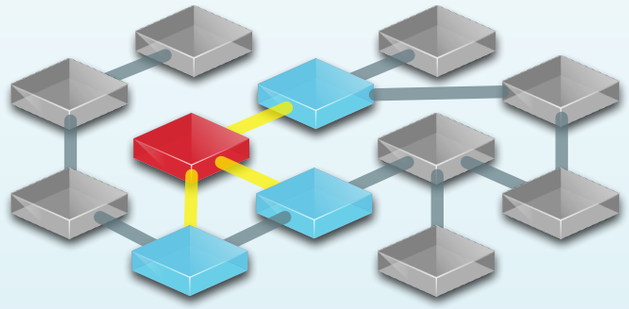
Properties (each a key+value)

\+ Indexes (for easy look-ups)



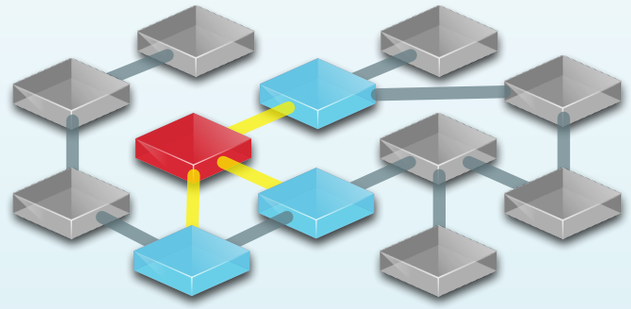


Looks different, fine. Who cares?



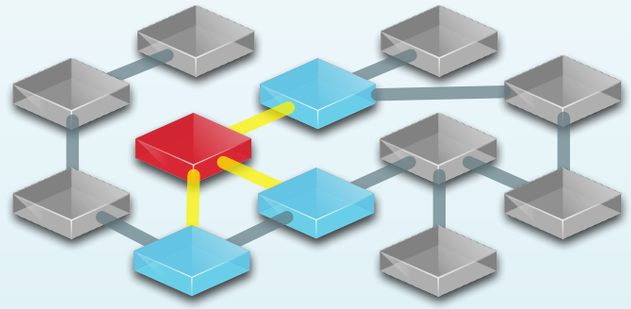
Looks different, fine. Who cares?

- a sample social graph



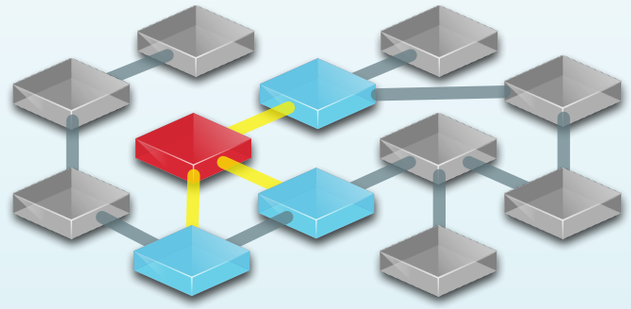
Looks different, fine. Who cares?

- a sample social graph
  - with ~1,000 persons



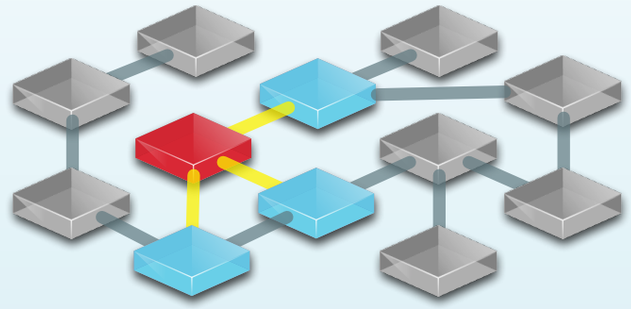
Looks different, fine. Who cares?

- a sample social graph
  - with ~1,000 persons
- average 50 friends per person



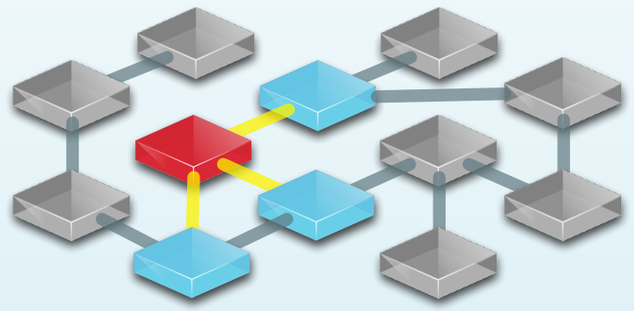
Looks different, fine. Who cares?

- a sample social graph
  - with ~1,000 persons
- average 50 friends per person
- pathExists(a,b) limited to depth 4



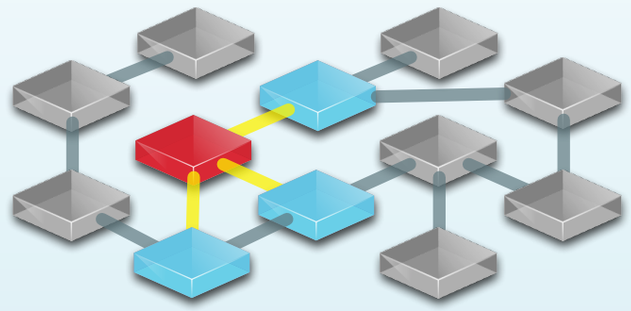
Looks different, fine. Who cares?

- a sample social graph
  - with ~1,000 persons
- average 50 friends per person
- pathExists(a,b) limited to depth 4
- caches warmed up to eliminate disk I/O



Looks different, fine. Who cares?

- a sample social graph
  - with ~1,000 persons
- average 50 friends per person
- pathExists(a,b) limited to depth 4

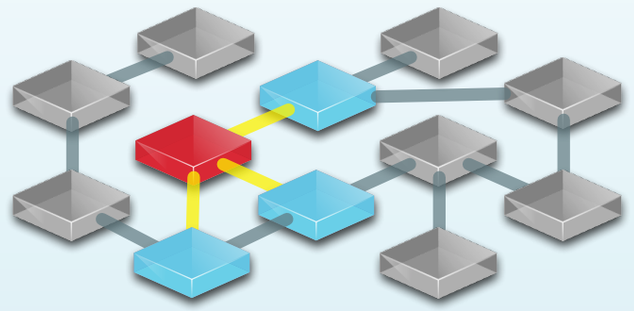


- caches warmed up to eliminate disk I/O

	# persons	query time
Relational database	1,000	2000ms

Looks different, fine. Who cares?

- a sample social graph
  - with ~1,000 persons
- average 50 friends per person
- pathExists(a,b) limited to depth 4

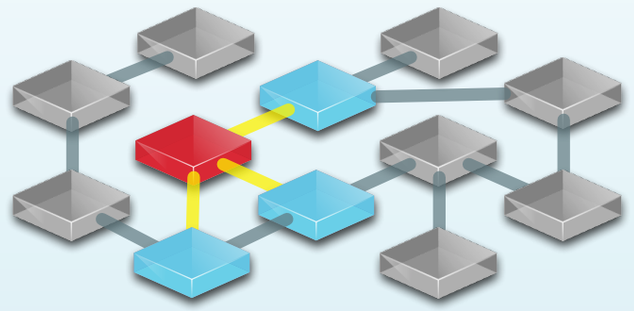


- caches warmed up to eliminate disk I/O

	# persons	query time
Relational database	1,000	2000ms
Neo4j	1,000	2ms

Looks different, fine. Who cares?

- a sample social graph
  - with ~1,000 persons
- average 50 friends per person
- pathExists(a,b) limited to depth 4



- caches warmed up to eliminate disk I/O

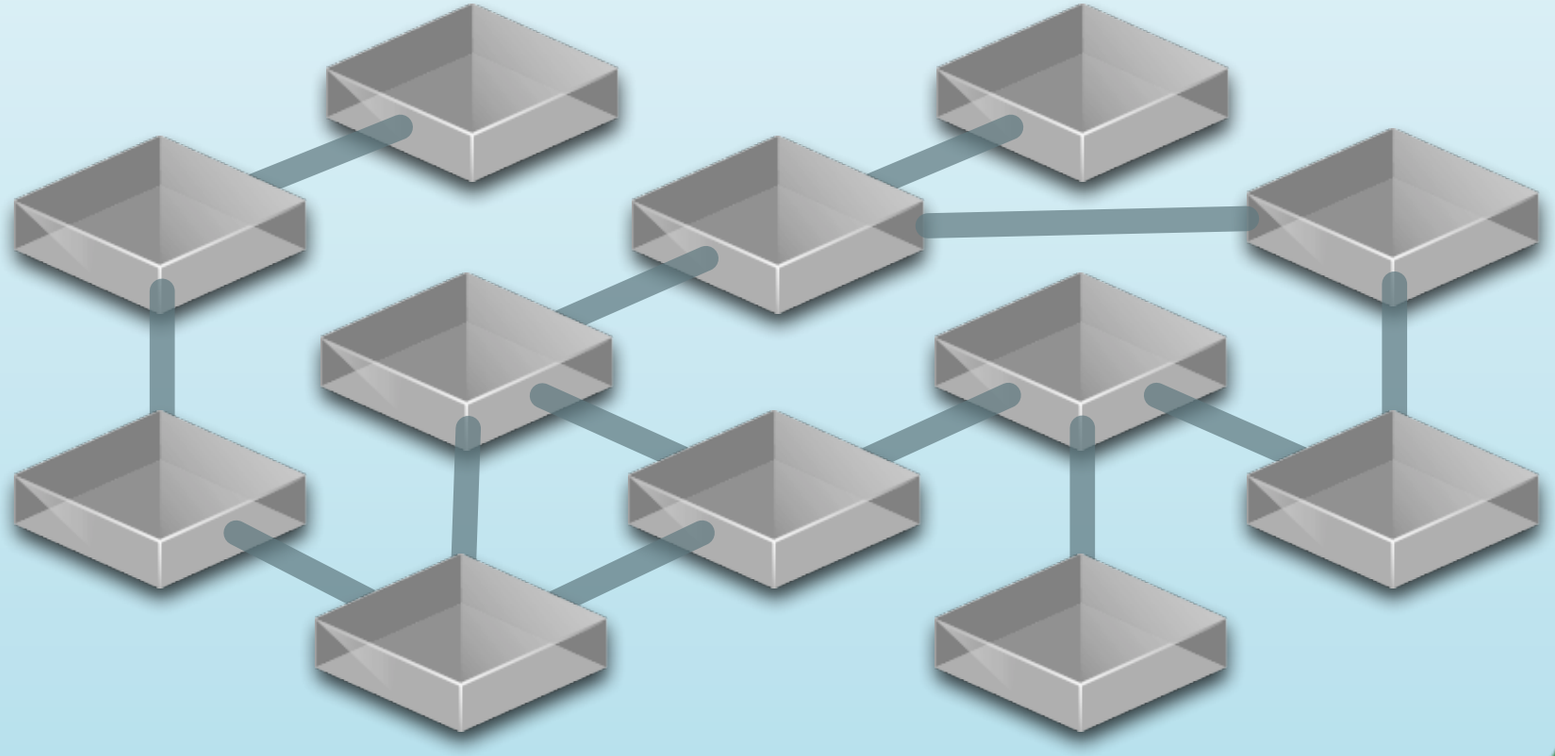
	# persons	query time
Relational database	1,000	2000ms
Neo4j	1,000	2ms
Neo4j	1,000,000	2ms

Graph Database: Pros & Cons

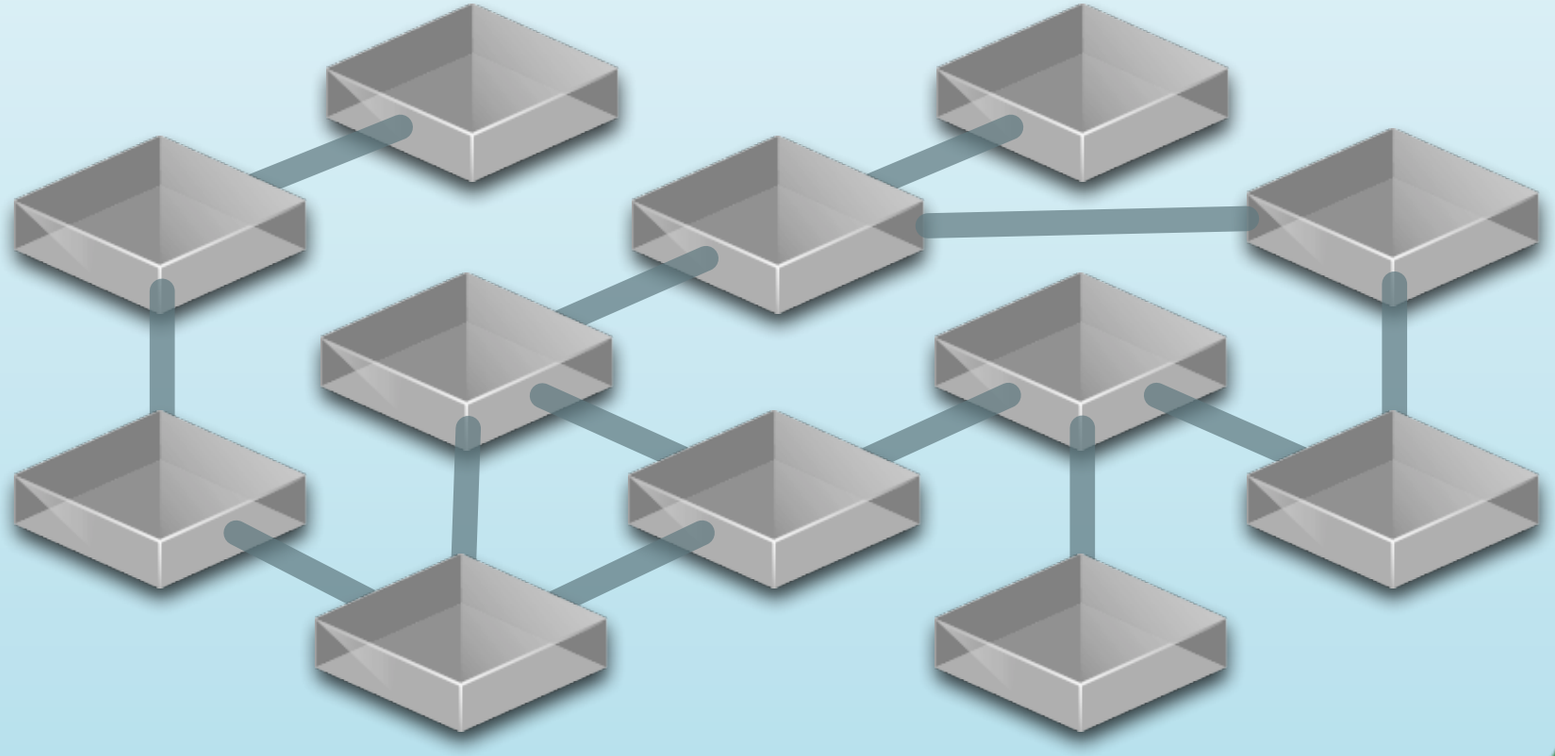
- Strengths
  - Powerful data model, as general as RDBMS
  - Fast, for connected data
  - Easy to query
- Weaknesses:
  - Sharding (though they can scale reasonably well)
    - also, stay tuned for developments here
  - Requires conceptual shift
    - though graph-like thinking becomes addictive

And, but, so how do you query this "graph" database?

Query a graph with a traversal

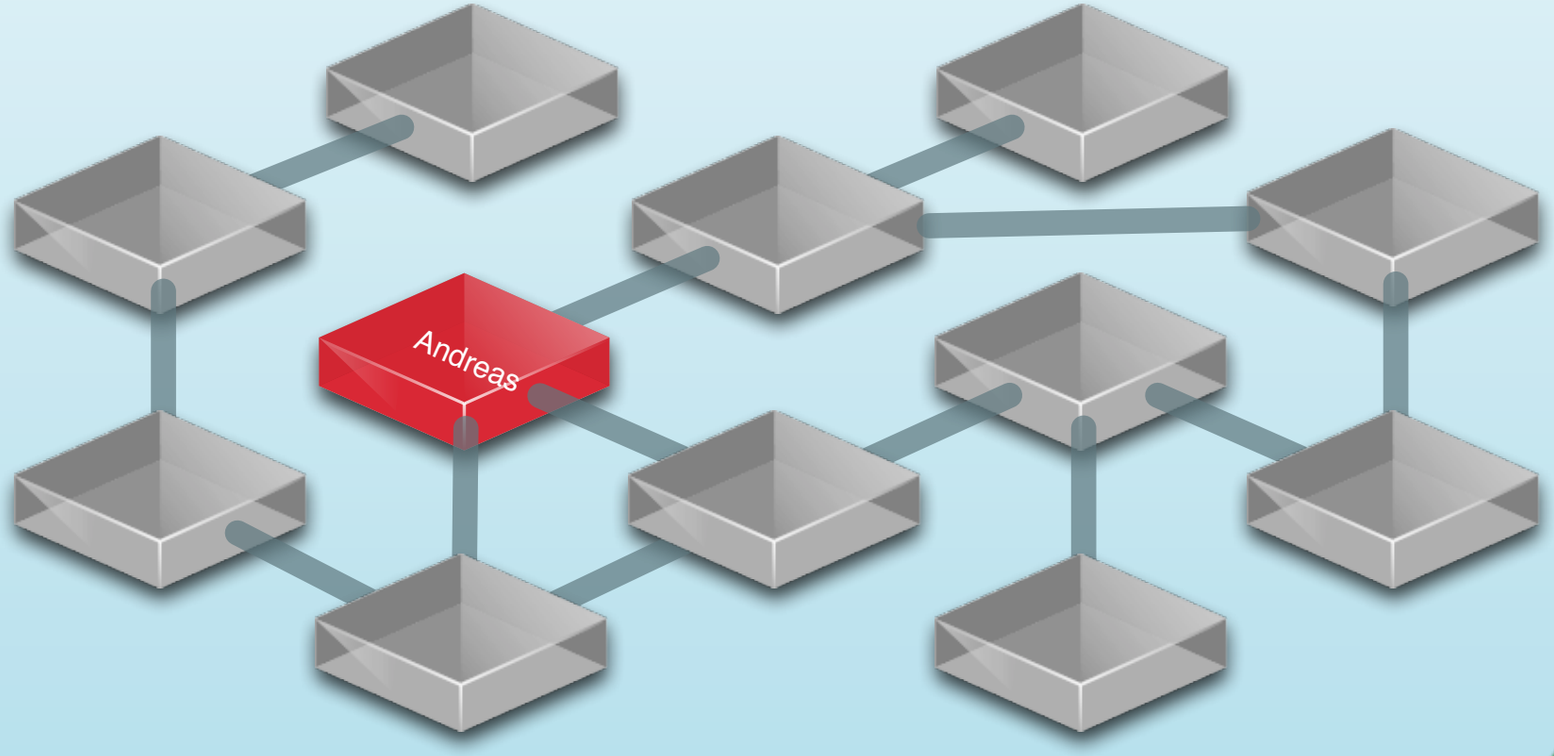


Query a graph with a traversal



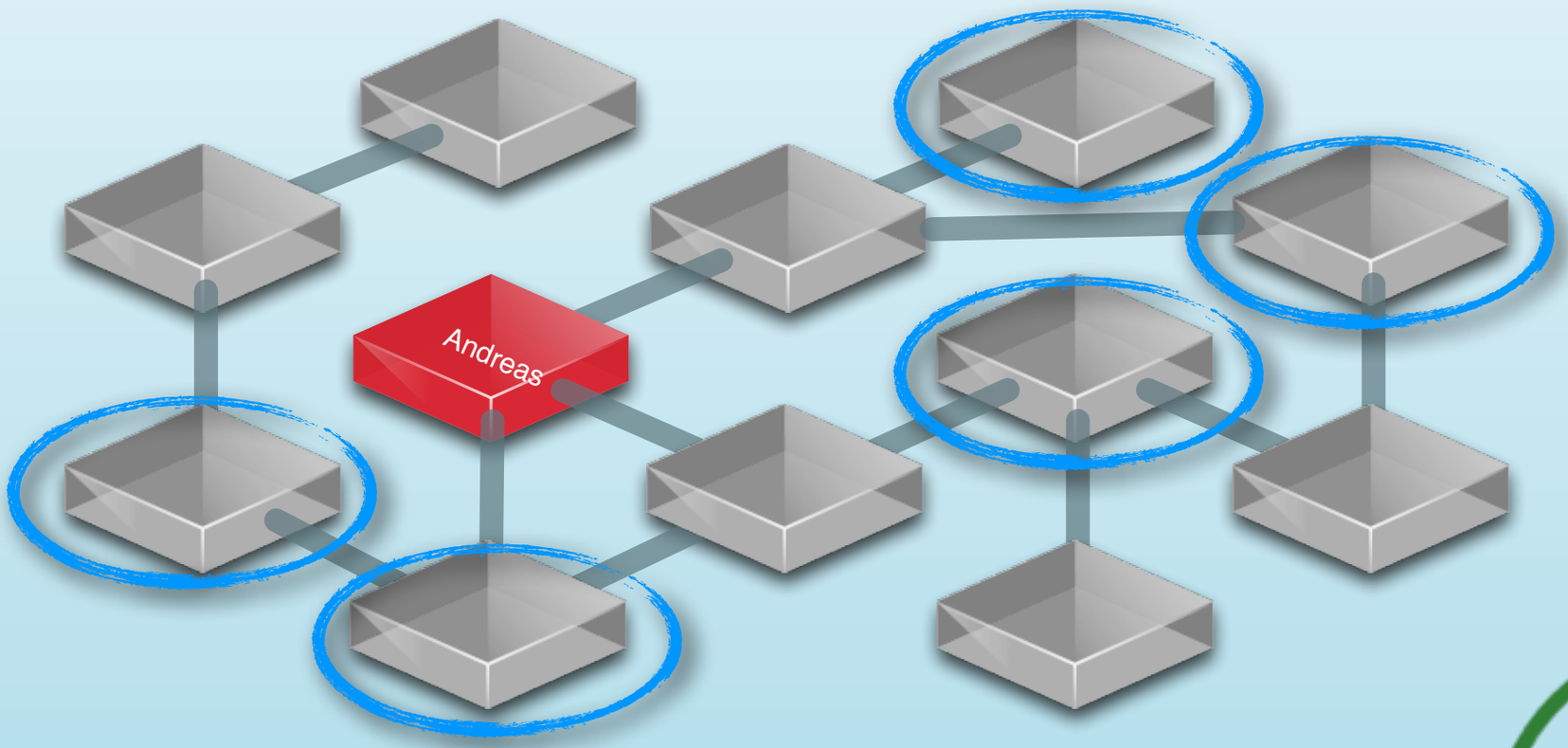
Query a graph with a traversal

```
// lookup starting point in an index
start n=node:People(name = 'Andreas')
```

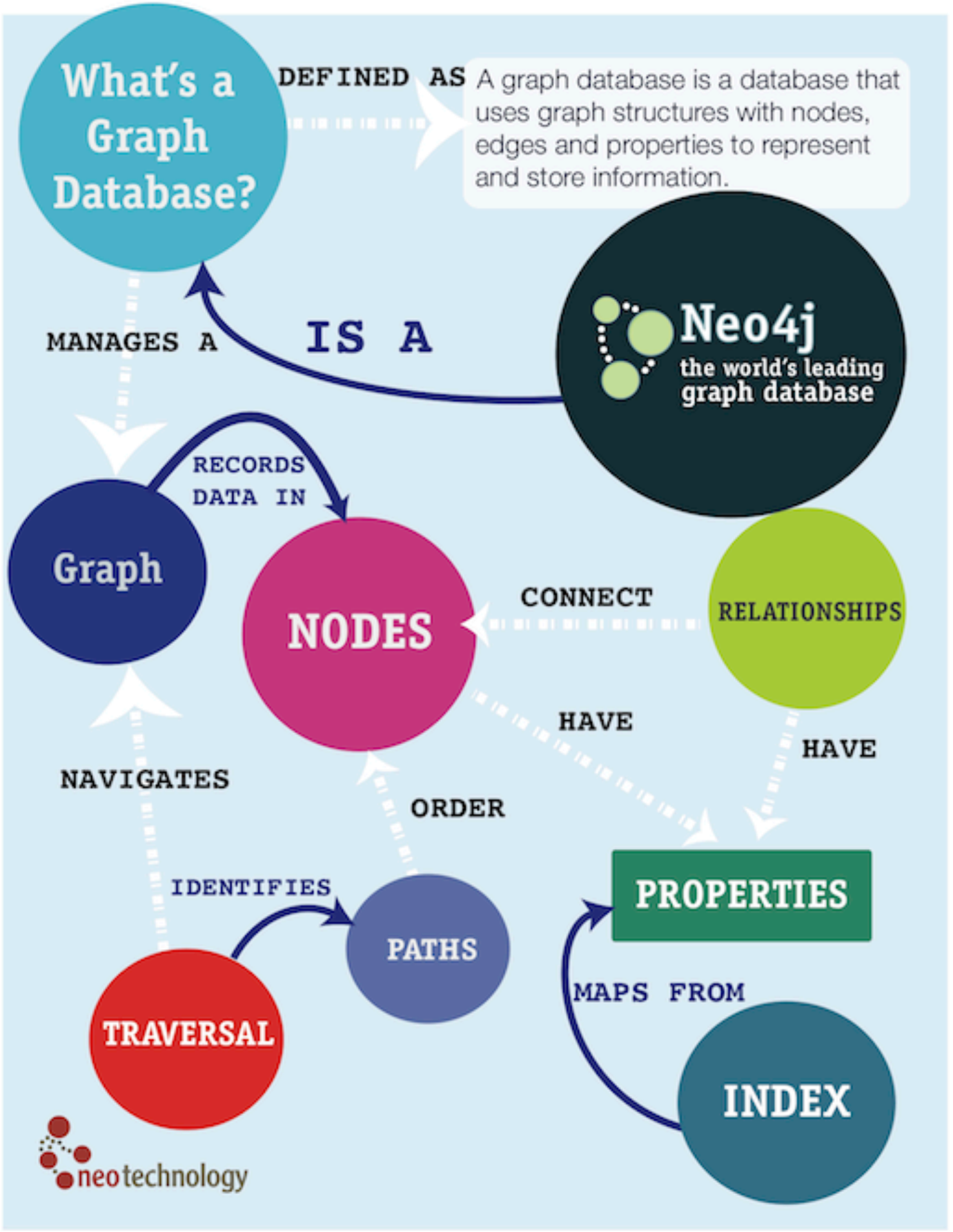


Query a graph with a traversal

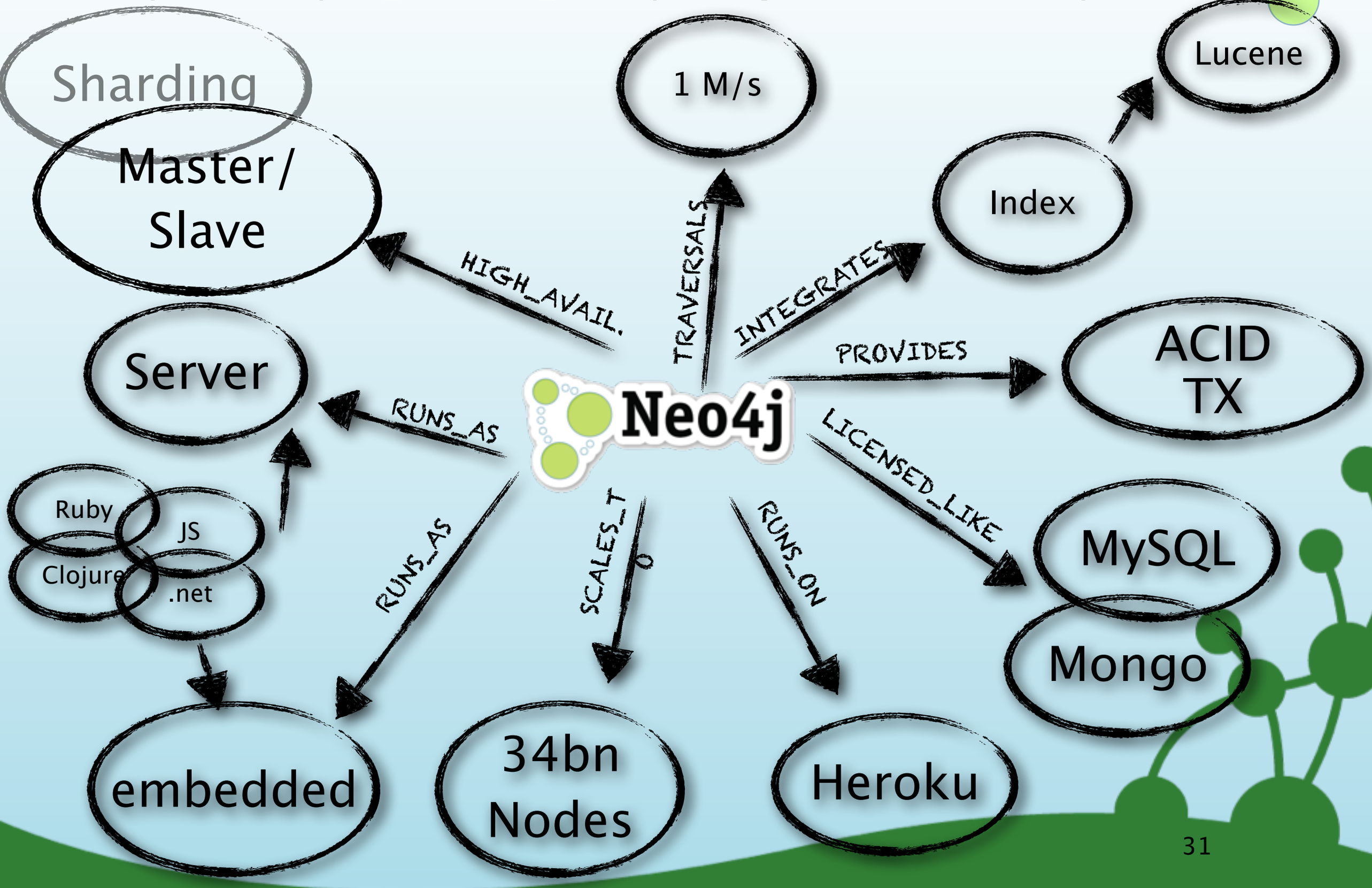
```
// then traverse to find results
start n=node:People(name = 'Andreas')
  match (n)--()--(foaf) return foaf
```



Neo4j – the Graph Database


What's a Graph Database?
DEFINED AS
A graph database is a database that uses graph structures with nodes, edges and properties to represent and store information.
Neo4j
the world's leading graph database
IS A
MANAGES A
RECORDS DATA IN
Graph
NODES
CONNECT
RELATIONSHIPS
HAVE
HAVE
NAVIGATES
ORDER
IDENTIFIES
PATHS
PROPERTIES
MAPS FROM
TRAVERSAL
INDEX
neotechnology

(Neo4j) -[:IS_A]-> (Graph Database)



Neo4j is a Graph Database

Neo4j is a Graph Database

- A **Graph** Database:

Neo4j is a Graph Database

- A **Graph** Database:
  - a schema-free Property Graph

Neo4j is a Graph Database

- A **Graph** Database:
  - a schema-free Property Graph
  - perfect for complex, highly connected data

Neo4j is a Graph Database

- A **Graph** Database:
  - a schema-free Property Graph
  - perfect for complex, highly connected data
- A Graph **Database**:

Neo4j is a Graph Database

- A **Graph** Database:
  - a schema-free Property Graph
  - perfect for complex, highly connected data
- A Graph **Database**:
  - reliable with real ACID Transactions

Neo4j is a Graph Database

- A **Graph** Database:
  - a schema-free Property Graph
  - perfect for complex, highly connected data
- A Graph **Database**:
  - reliable with real ACID Transactions
  - scalable: 32 Billion Nodes, 32 Billion Relationships, 64 Billion Properties

Neo4j is a Graph Database

- A **Graph** Database:
  - a schema-free Property Graph
  - perfect for complex, highly connected data
- A Graph **Database**:
  - reliable with real ACID Transactions
  - scalable: 32 Billion Nodes, 32 Billion Relationships, 64 Billion Properties
  - fast with more than 1M traversals / second

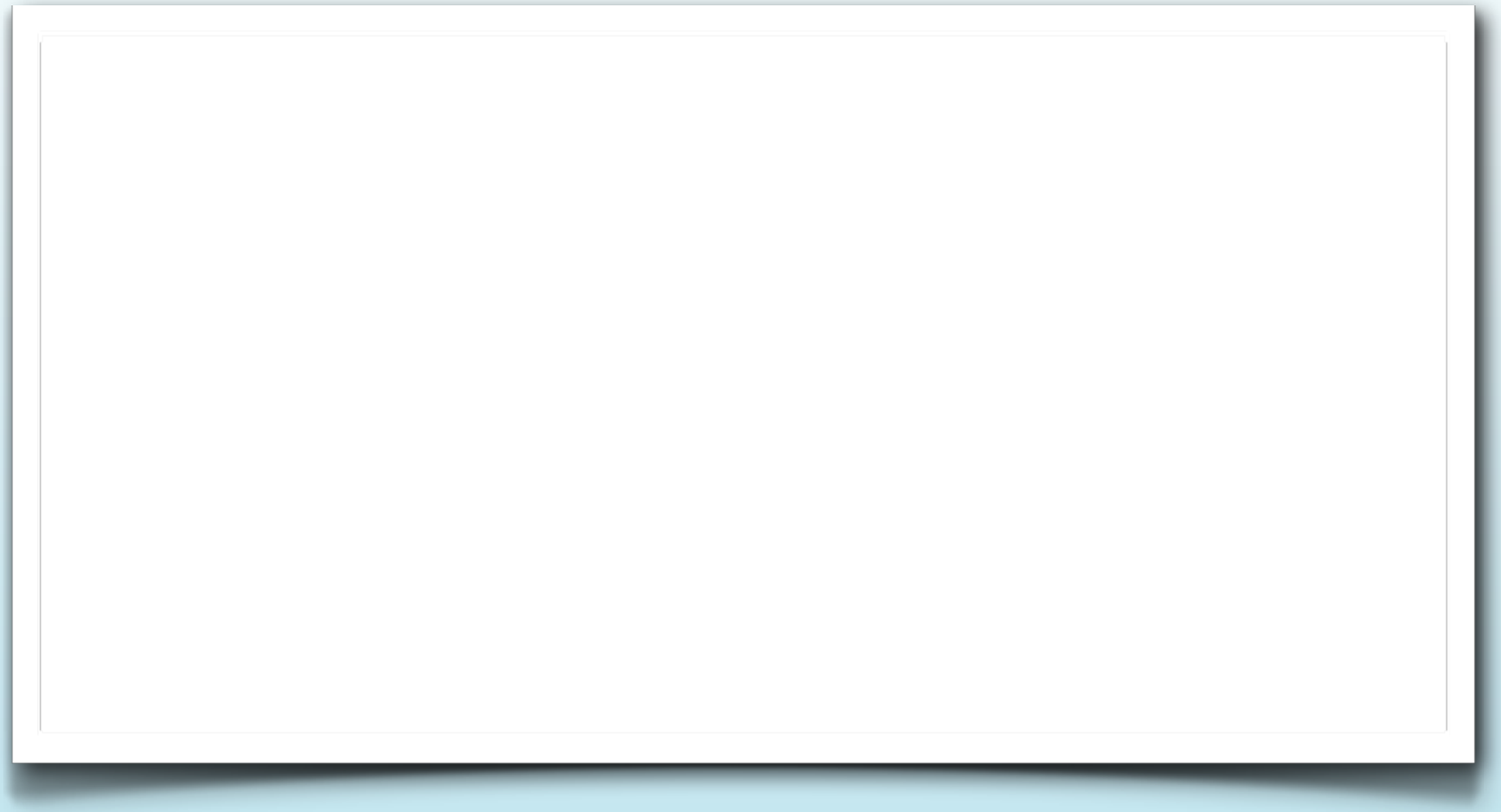
Neo4j is a Graph Database

- A **Graph** Database:
  - a schema-free Property Graph
  - perfect for complex, highly connected data
- A Graph **Database**:
  - reliable with real ACID Transactions
  - scalable: 32 Billion Nodes, 32 Billion Relationships, 64 Billion Properties
  - fast with more than 1M traversals / second
  - Server with REST API, or Embeddable on the JVM

Neo4j is a Graph Database

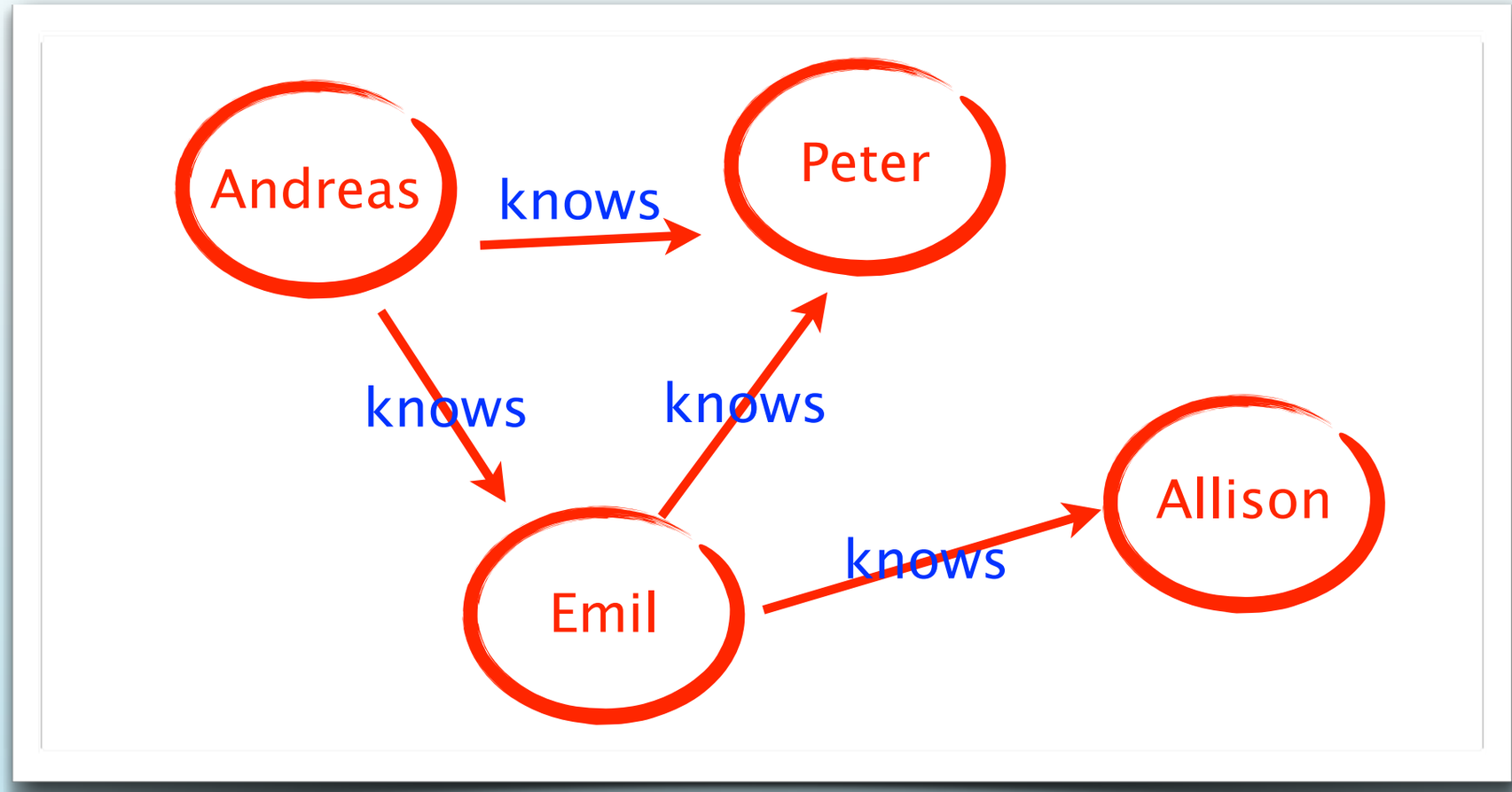
- A **Graph** Database:
  - a schema-free Property Graph
  - perfect for complex, highly connected data
- A Graph **Database**:
  - reliable with real ACID Transactions
  - scalable: 32 Billion Nodes, 32 Billion Relationships, 64 Billion Properties
  - fast with more than 1M traversals / second
  - Server with REST API, or Embeddable on the JVM
  - higher-performance with High-Availability (read scaling)

Whiteboard --> Data

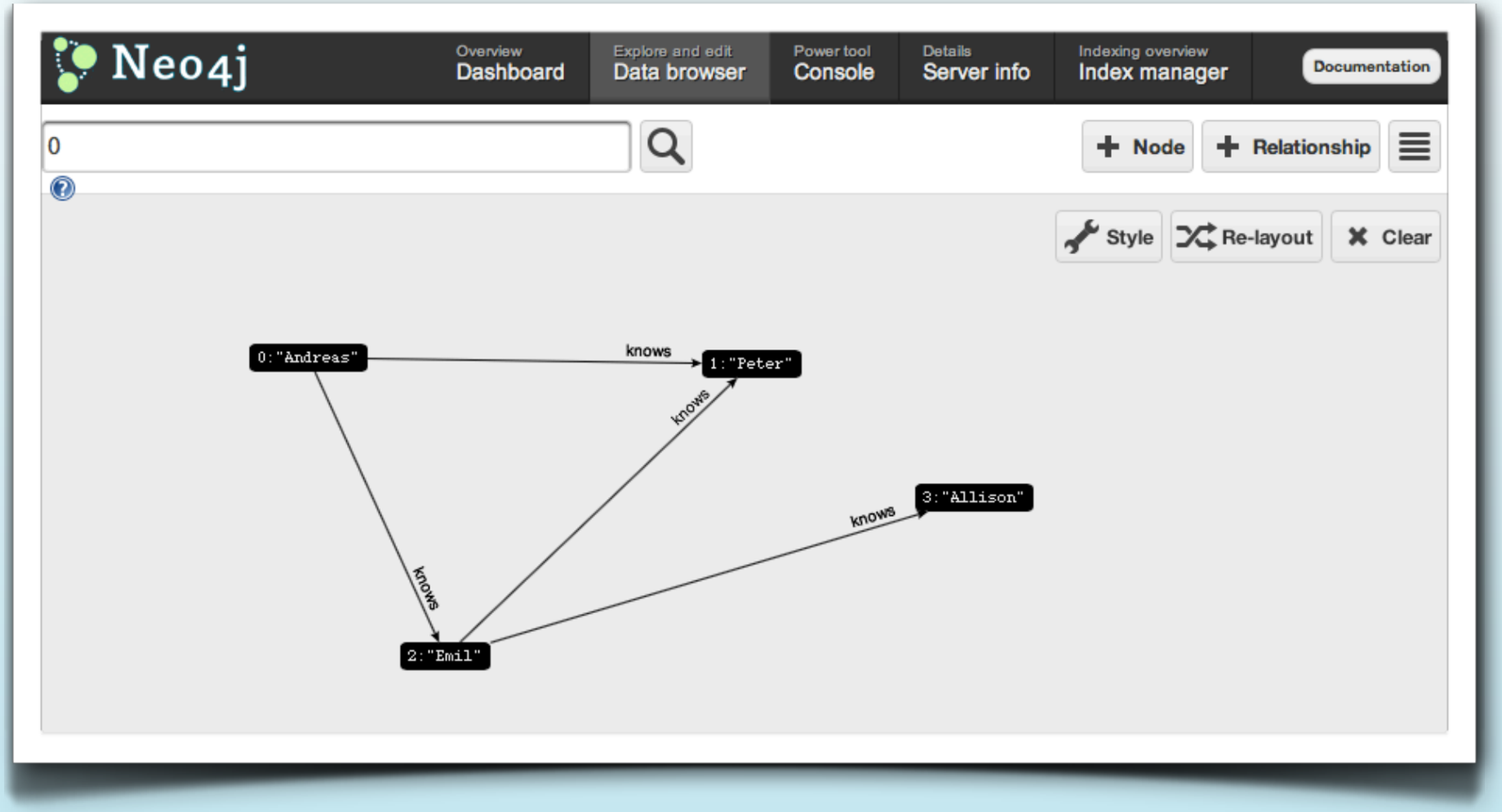


Whiteboard --> Data

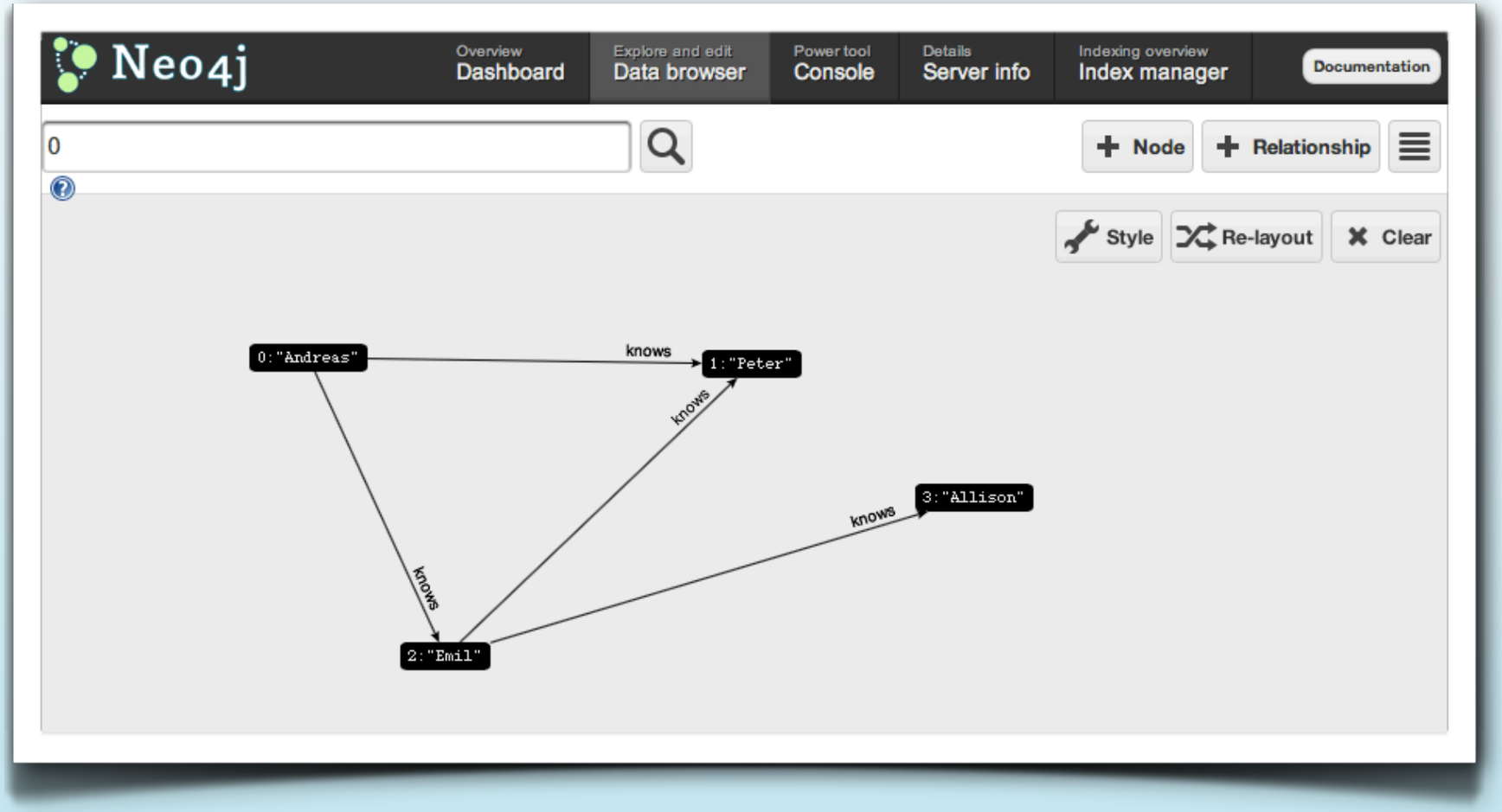
Whiteboard --> Data



Whiteboard --> Data



Whiteboard --> Data




```
// Cypher query - friend of a friend
start n=node(0)
match (n)--()--(foaf)
return foaf
```

Two Ways to Work with Neo4j

Two Ways to Work with Neo4j

- 1. Embeddable on JVM

Two Ways to Work with Neo4j

- 1. Embeddable on JVM
  - Java, JRuby, Scala...

Two Ways to Work with Neo4j

- 1. Embeddable on JVM
  - Java, JRuby, Scala...
  - Tomcat, Rails, Akka, etc.

Two Ways to Work with Neo4j

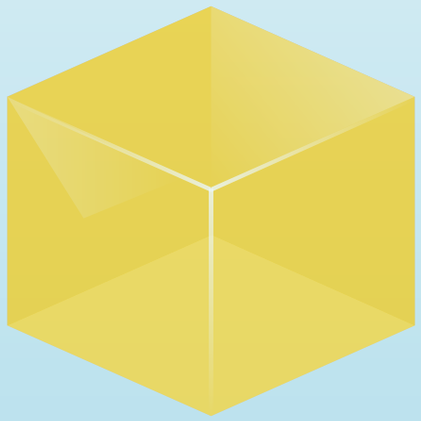
- 1. Embeddable on JVM
  - Java, JRuby, Scala...
  - Tomcat, Rails, Akka, etc.
  - great for testing

Two Ways to Work with Neo4j

- 1. Embeddable on JVM
  - Java, JRuby, Scala...
  - Tomcat, Rails, Akka, etc.
  - great for testing

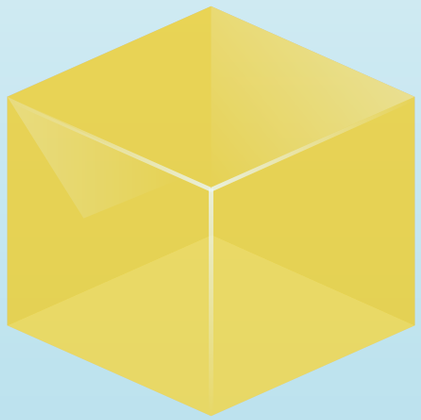
Two Ways to Work with Neo4j

- 1. Embeddable on JVM
  - Java, JRuby, Scala...
  - Tomcat, Rails, Akka, etc.
  - great for testing



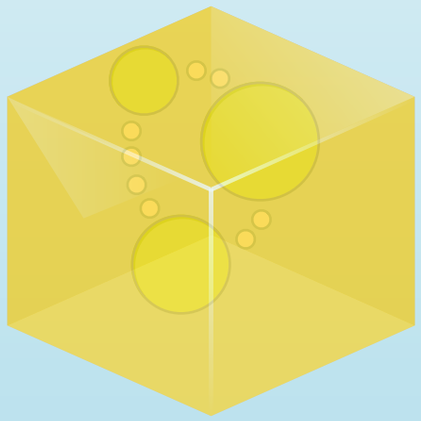
Two Ways to Work with Neo4j

- 1. Embeddable on JVM
  - Java, JRuby, Scala...
  - Tomcat, Rails, Akka, etc.
  - great for testing



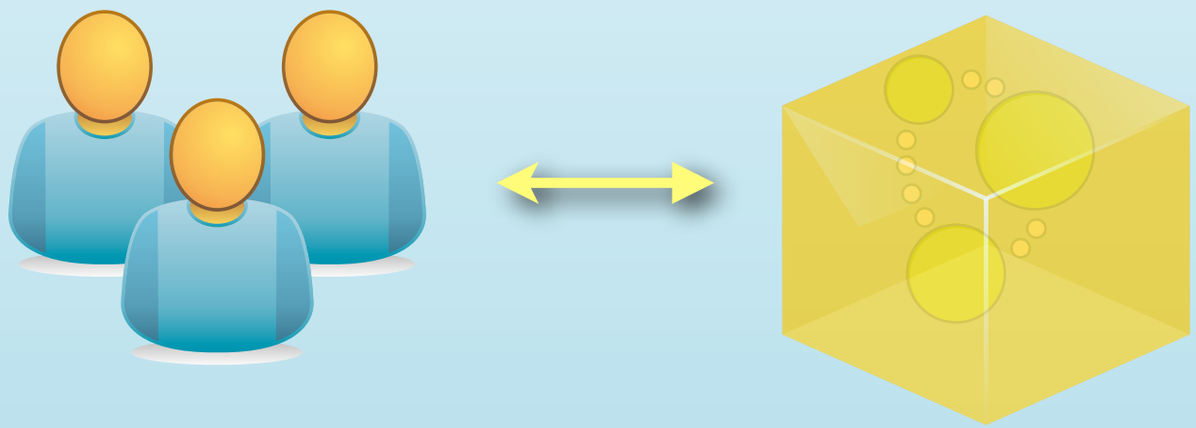
Two Ways to Work with Neo4j

- 1. Embeddable on JVM
  - Java, JRuby, Scala...
  - Tomcat, Rails, Akka, etc.
  - great for testing



Two Ways to Work with Neo4j

- 1. Embeddable on JVM
  - Java, JRuby, Scala...
  - Tomcat, Rails, Akka, etc.
  - great for testing



Two Ways to Work with Neo4j

Show me some code, please

```
GraphDatabaseService graphDb =
    new EmbeddedGraphDatabase(“var/neo4j”);
Transaction tx = graphDb.beginTx();
try {
  Node steve = graphDb.createNode();
  Node michael = graphDb.createNode();

  steve.setProperty(“name”, “Steve Vinoski”);
  michael.setProperty(“name”, “Michael Hunger”);

  Relationship presentedWith = steve.createRelationshipTo(
  michael, PresentationTypes.PRESENTED_WITH);
  presentedWith.setProperty(“date”, today);
  tx.success();
} finally {
  tx.finish();
}
```

Spring Data Neo4j

```
@NodeEntity
public class Movie {
  @Indexed private String title;
  @RelatedToVia(type = “ACTS_IN”, direction=INCOMING)
  private Set<Role> cast;
  private Director director;
}

@NodeEntity
public class Actor {
  @RelatedTo(type = “ACTS_IN”)
  private Set<Movies> movies;
}

@RelationshipEntity
public class Role {
  @StartNode private Actor actor;
  @EndNode private Movie movie;
  private String roleName;
}
```

neo4j.rb

```
   gem install neo4j

require 'rubygems'
require 'neo4j'

class Person
  include Neo4j::NodeMixin
  property :name, :age, :rank
  index :name
  has_n :friends
end

Neo4j::Transaction.run do
  neo = Person.new :name=>'Neo', :age=>29
  morpheus =
Person.new :name=>'Morpheus', :rank=>'Captain'
  neo.friends << morpheus
end

neo.friends.each {|p|...}
```

Cypher Query Language

- Declarative query language
  - Describe *what* you want, not *how*
  - Based on pattern matching
- Examples:

```
START david=node:people(name="David")   # index lookup
MATCH david-[:knows]-friends-[:knows]-new_friends
WHERE new_friends.age > 18
RETURN new_friends

START user=node(5, 15, 26, 28)   # node IDs
MATCH user--friend
RETURN user, COUNT(friend), SUM(friend.money)
```

Create Graph with Cypher

```
CREATE
  (steve {name: “Steve Vinoski”})
    -[:PRESENTED_WITH {date:{day}}]->
  (michael {name: “Michael Hunger”})
```

Two Ways to Work with Neo4j

Two Ways to Work with Neo4j

- 2. Server with REST API

Two Ways to Work with Neo4j

- 2. Server with REST API
  - every language on the planet

Two Ways to Work with Neo4j

- 2. Server with REST API
  - every language on the planet
  - flexible deployment scenarios

Two Ways to Work with Neo4j

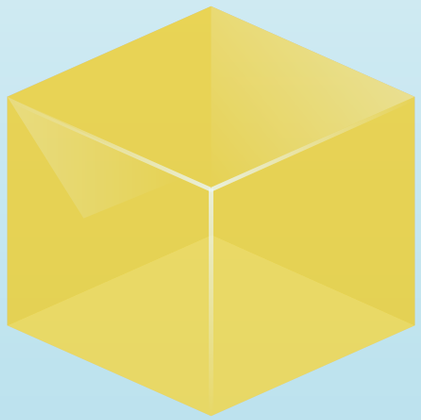
- 2. Server with REST API
  - every language on the planet
  - flexible deployment scenarios
  - DIY server, or cloud managed

Two Ways to Work with Neo4j

- 2. Server with REST API
  - every language on the planet
  - flexible deployment scenarios
  - DIY server, or cloud managed

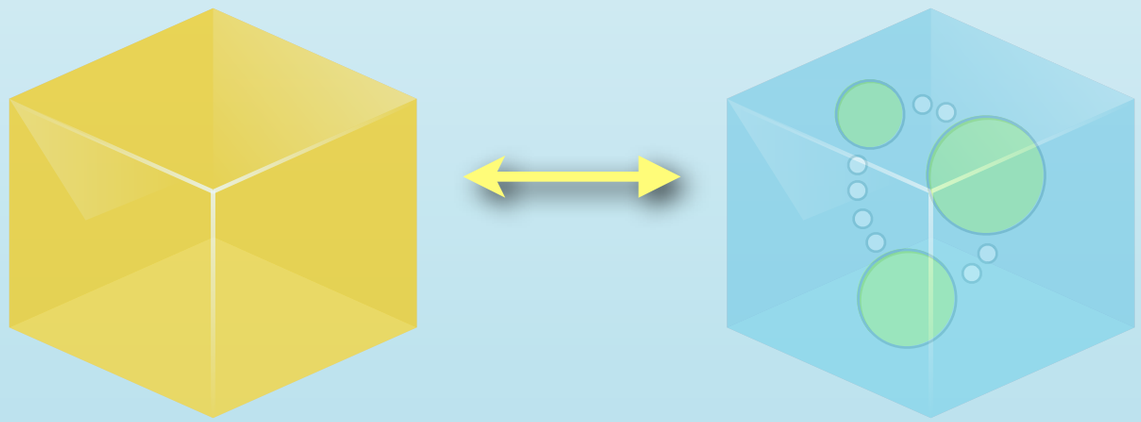
Two Ways to Work with Neo4j

- 2. Server with REST API
  - every language on the planet
  - flexible deployment scenarios
  - DIY server, or cloud managed



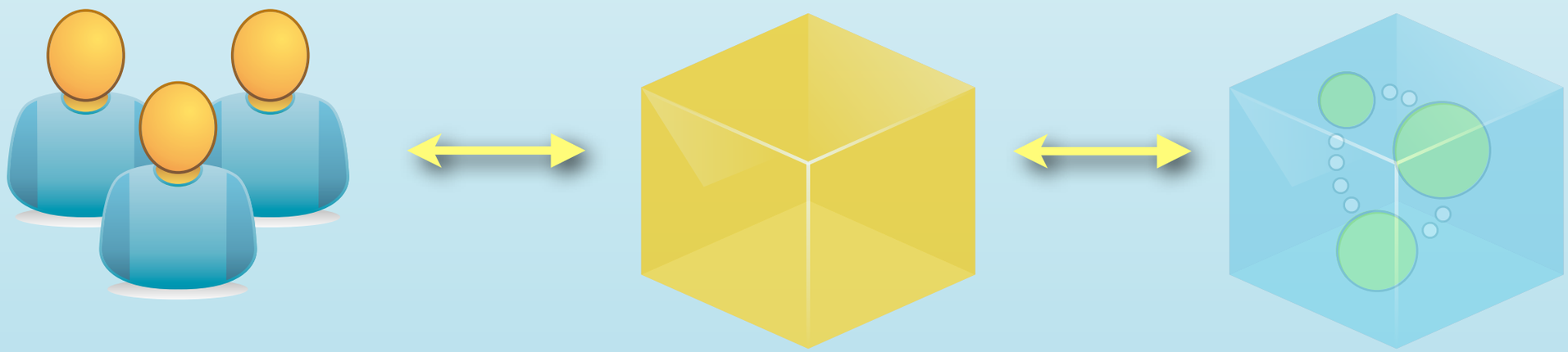
Two Ways to Work with Neo4j

- 2. Server with REST API
  - every language on the planet
  - flexible deployment scenarios
  - DIY server, or cloud managed



Two Ways to Work with Neo4j

- 2. Server with REST API
  - every language on the planet
  - flexible deployment scenarios
  - DIY server, or cloud managed



Two Ways to Work with Neo4j

Bindings

REST://

Two Ways to Work with Neo4j

Two Ways to Work with Neo4j

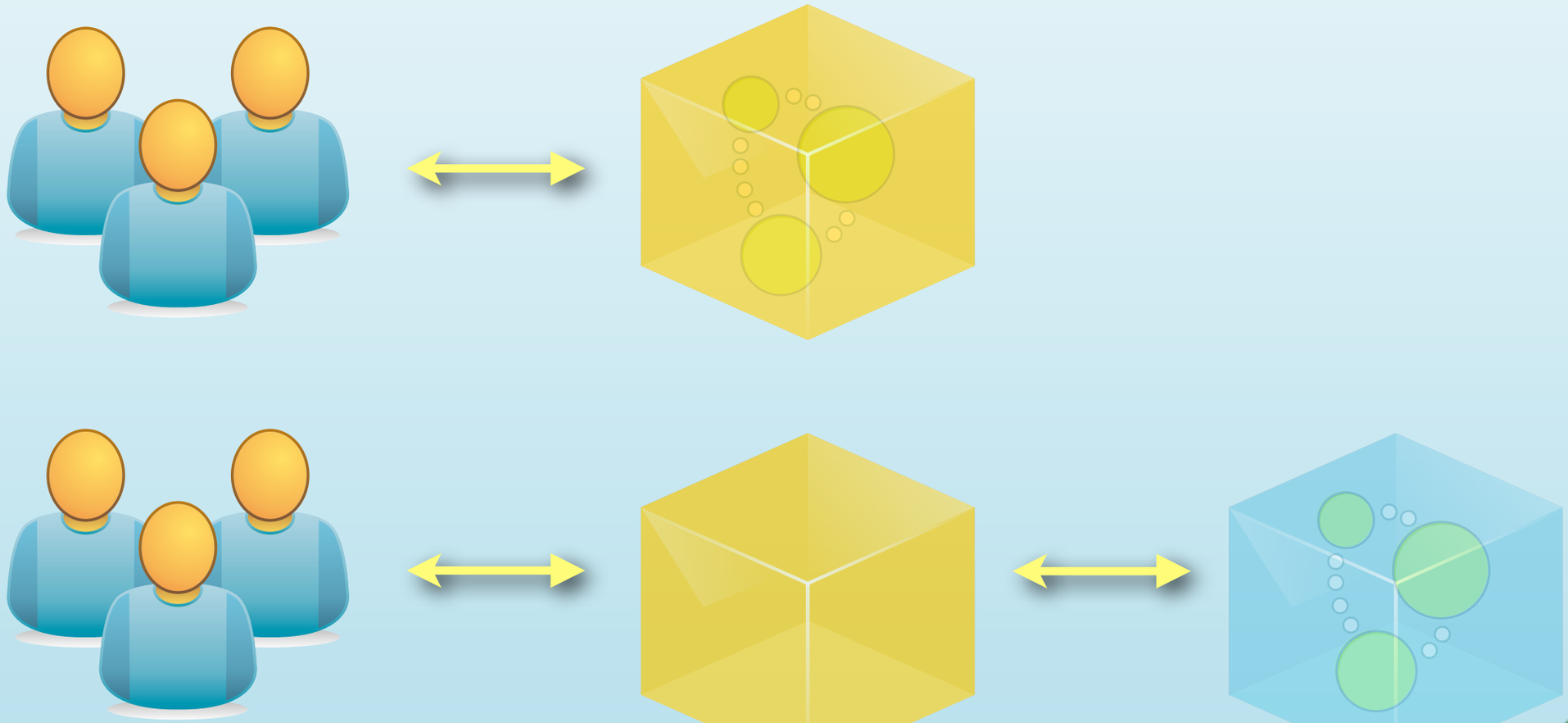
- Server capability == Embedded capability

Two Ways to Work with Neo4j

- Server capability == Embedded capability
  - same scalability, transactionality, and availability

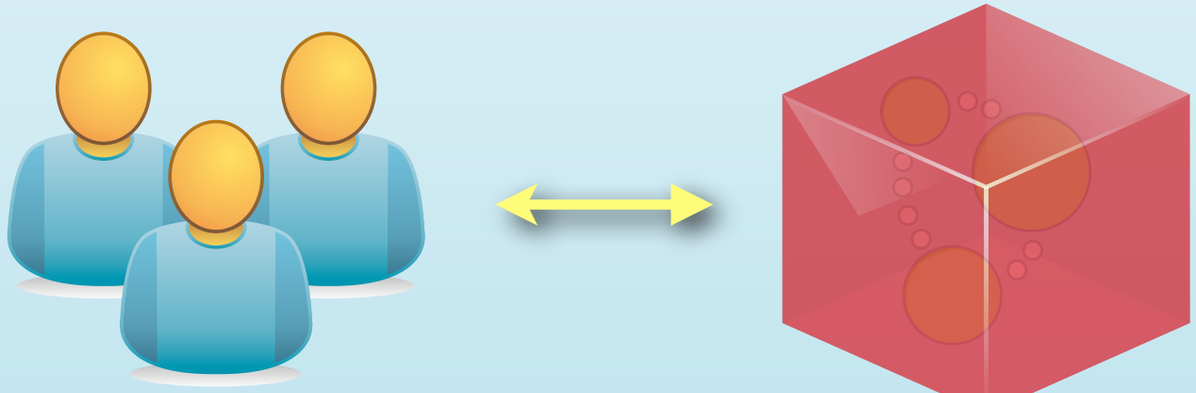
Two Ways to Work with Neo4j

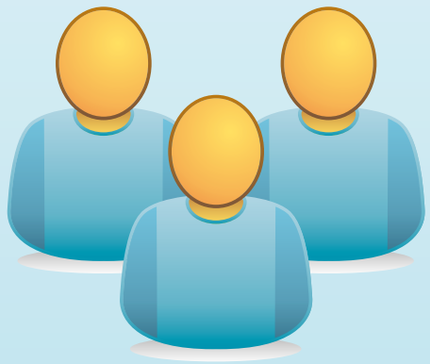
- Server capability == Embedded capability
  - same scalability, transactionality, and availability

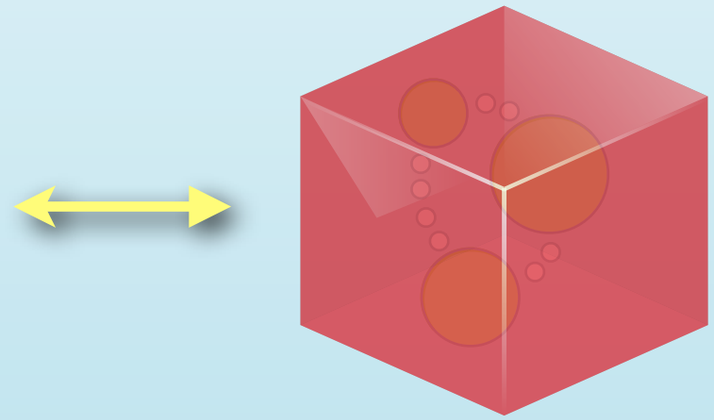


Two Ways to Work with Neo4j

- Server capability == Embedded capability
  - same scalability, transactionality, and availability







How to get started?

How to get started?

- Documentation

How to get started?

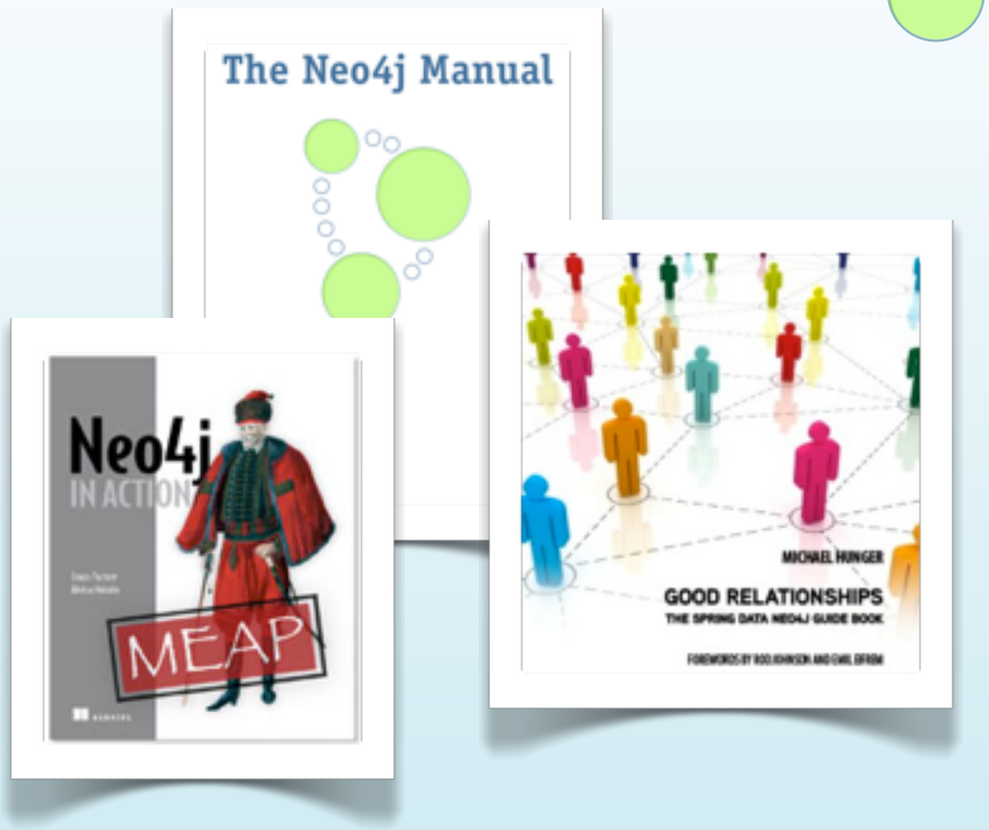
- Documentation
  - docs.neo4j.org - tutorials+reference

How to get started?

- Documentation
  - docs.neo4j.org - tutorials+reference
  - Neo4j in Action

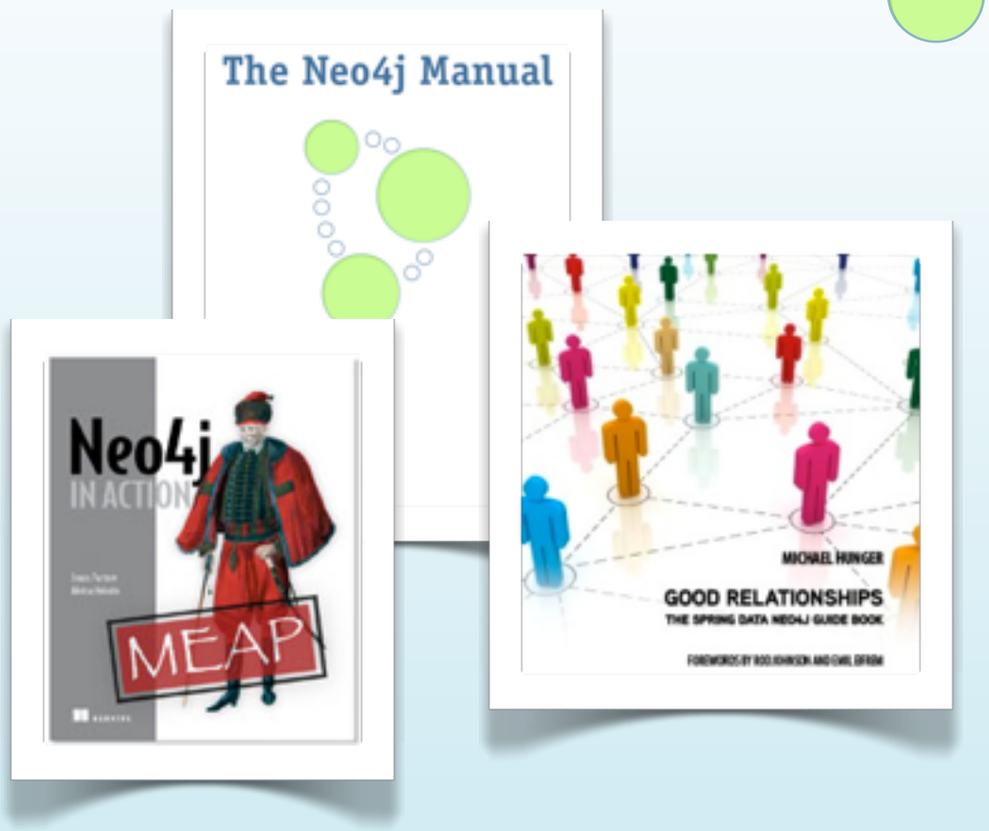
How to get started?

- Documentation
  - docs.neo4j.org - tutorials+reference
  - Neo4j in Action
  - Good Relationships



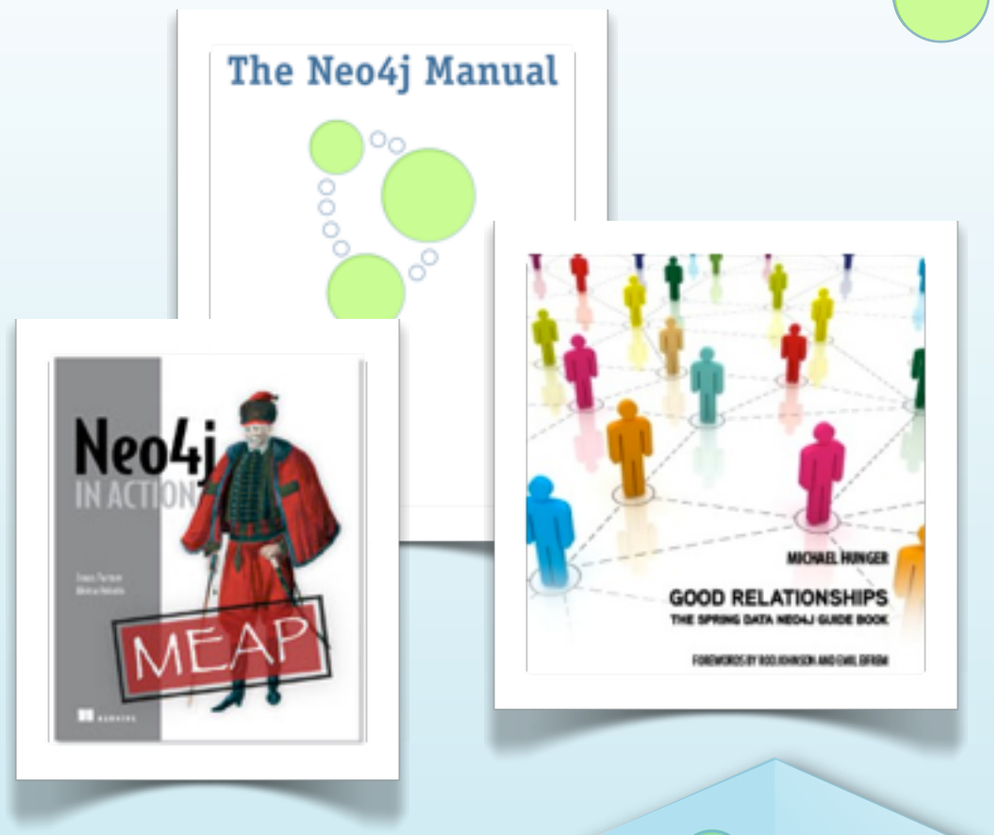
How to get started?

- Documentation
  - docs.neo4j.org - tutorials+reference
  - Neo4j in Action
  - Good Relationships
- Get Neo4j



How to get started?

- Documentation
  - docs.neo4j.org - tutorials+reference
  - Neo4j in Action
  - Good Relationships
- Get Neo4j
  - http://neo4j.org/download



How to get started?

- Documentation
  - docs.neo4j.org - tutorials+reference
  - Neo4j in Action
  - Good Relationships
- Get Neo4j
  - http://neo4j.org/download
  - http://addons.heroku.com/neo4j/

How to get started?

- Documentation
  - docs.neo4j.org - tutorials+reference
  - Neo4j in Action
  - Good Relationships
- Get Neo4j
  - http://neo4j.org/download
  - http://addons.heroku.com/neo4j/
- Participate
  - http://groups.google.com/group/neo4j
  - http://neo4j.meetup.com
  - a session like this one ;)

Thank you!