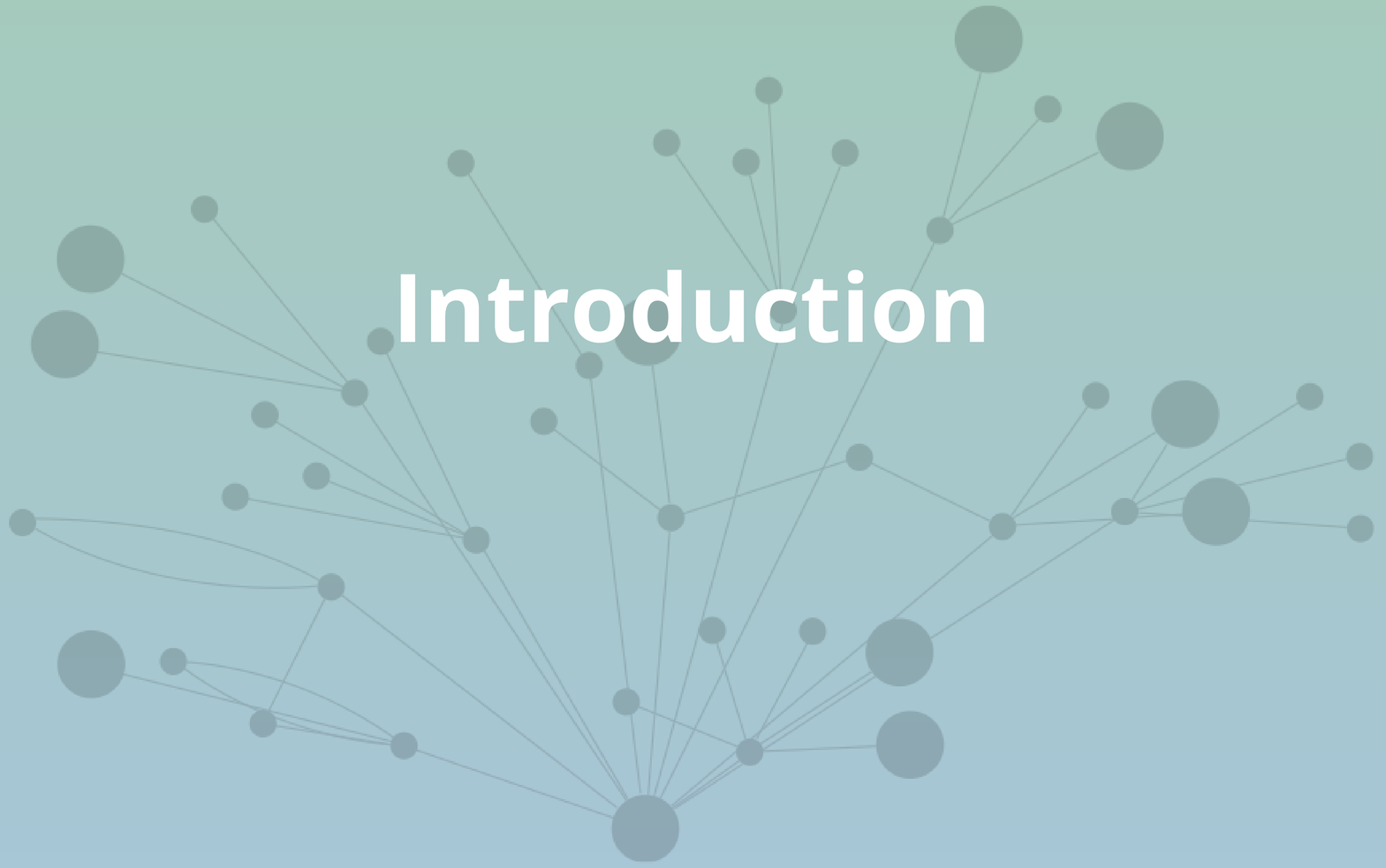
GRAND Stack

Benoit Simard (@logisima)

neo4j



Introduction

Benoit Simard

- **Graph addict**
- Data liberator
- Web developer
- Works at **Neo4j**
- **benoit@neo4j.com**
- **@logisima**

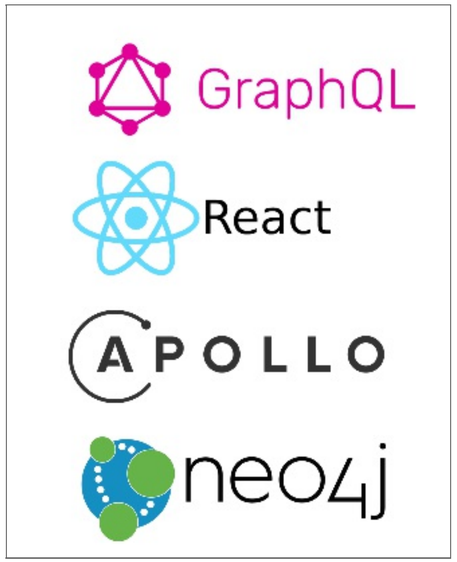
Agenda

- What is this Stack ?
- Why a new stack?
- How to use it ?

La stack

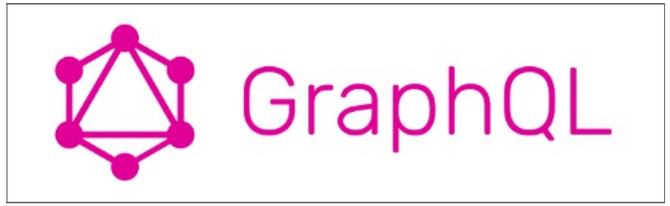
http://grandstack.io

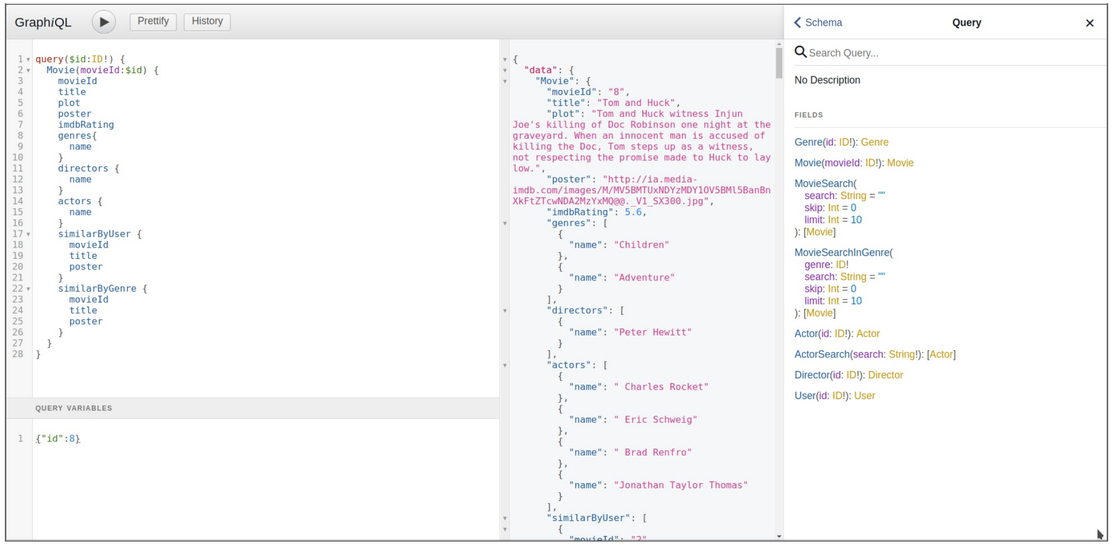
GRAND



GATAND

GraphQL





- A new paradigm for **building APIs**
- **Schema** definition
- Query language for APIs
- Community of tools

Angular

- JavaScript library for building user interfaces
- Component based
  - Reusable
  - Composable
- Written in Typescript (type matters !)

```
import { Component, Input, OnInit } from '@angular/core';

@Component({
  selector: 'display-object',
  templateUrl: './display-object.component.html',
  styleUrls: ['./display-object.component.scss']
})
export class DisplayObjectComponent  implements OnInit{

  @Input() object:any;
  @Input() blacklist:Array<String> = [];
  @Input() inlineSeparator:any = null;
  keys:any;

  constructor() {}

  ngOnInit() {
    // find keys that are not blacklisted and wherethe key has a value
    this.keys = Object.keys(this.object).filter( (key) =>{
      let result = true;

      if(key == '__typename'){
        result = false;
      }

      if(this.blacklist.includes(key)) {
        result = false;
      }

      if(this.object[key] == null || this.object[key] == '') {
        result = false;
      }

      return result;
    })
  }

}
```

Apollo

"A set of tools designed to leverage GraphQL and work together to create a great workflow"

Client-side tooling

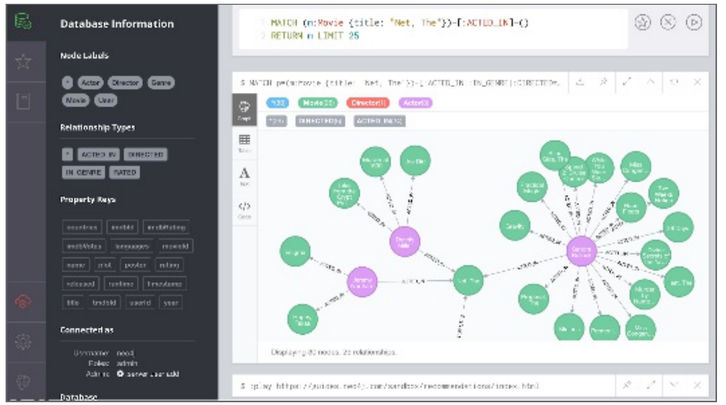
- Frontend framework integrations
- Caching
- Code generation

Server-side tooling

- Schema creation
- Mocking
- Schema stitching
- Performance monitoring

Neo4j

- Is a graph **database** (ACID compliant)
- Is a **graph** database (Native)
- Schema less
- Exists since 2010



Example : a movie application

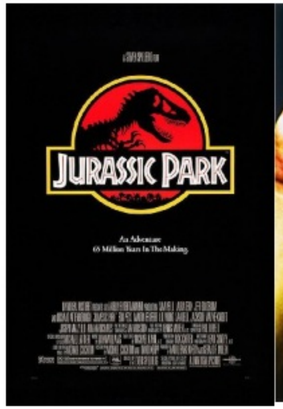
https://github.com/sim51/grand-movies-example

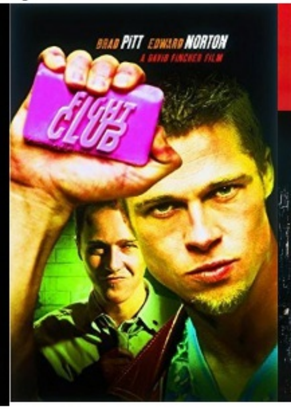
Matrix, The

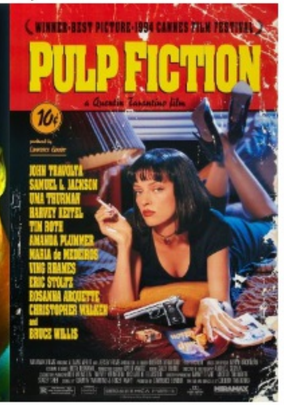
Title	Matrix, The
Description	A computer hacker learns from mysterious rebels about the true nature of his reality and his role in the war against its controllers.
Date	Mar 31, 1999
Directors	Lana Wachowski Andy Wachowski
Actors	Hugo Weaving Carrie-Anne Moss Laurence Fishburne Keanu Reeves
Genres	Thriller Sci-Fi Action

Recommendations

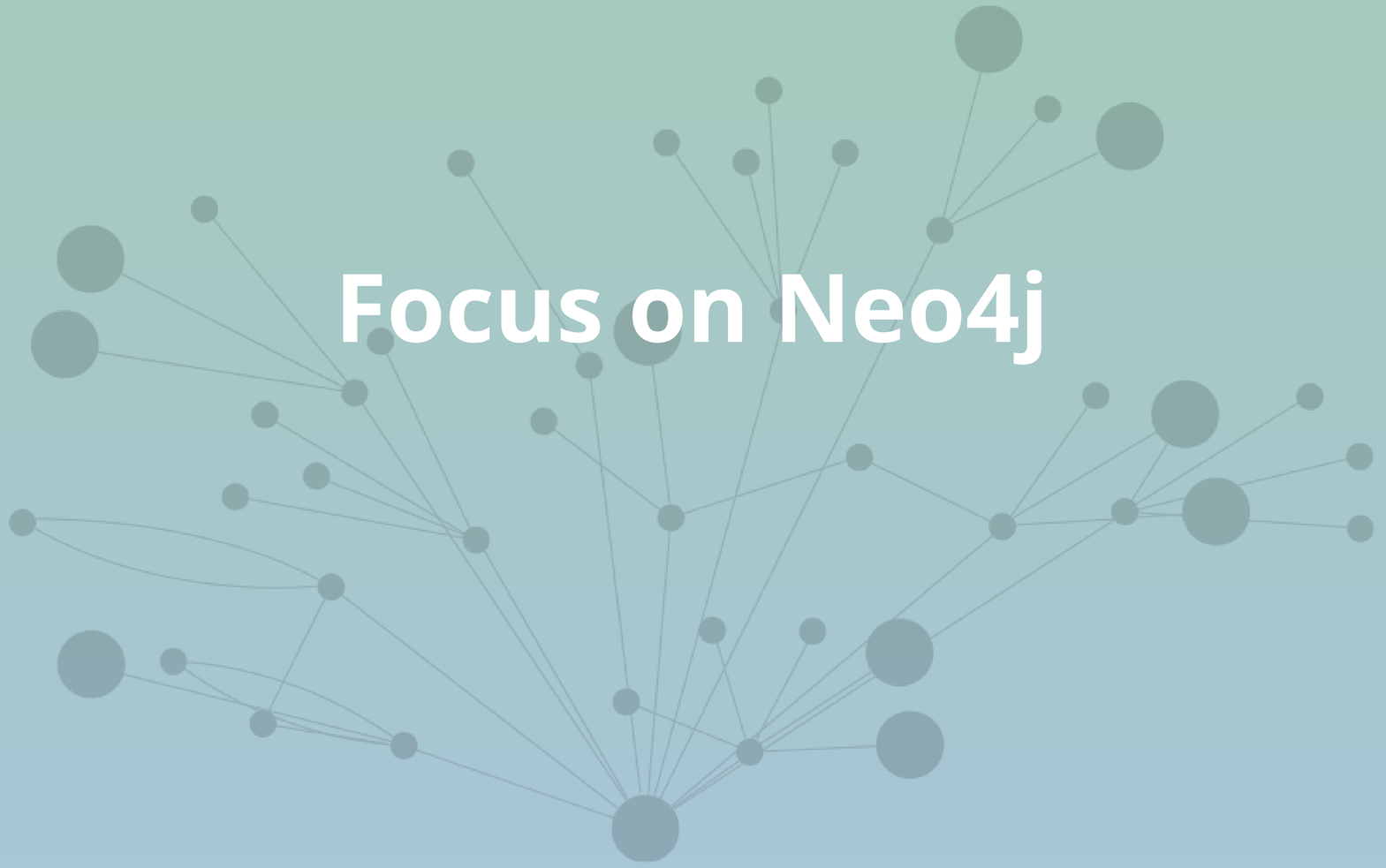
Star Wars: Episode IV - A New Hope - 8.7 Star Wars: Episode V - The Empire Strikes Back - 8.8Jurassic Park - 8.1 Fight Club - 8.9 Pulp Fiction - 8.9



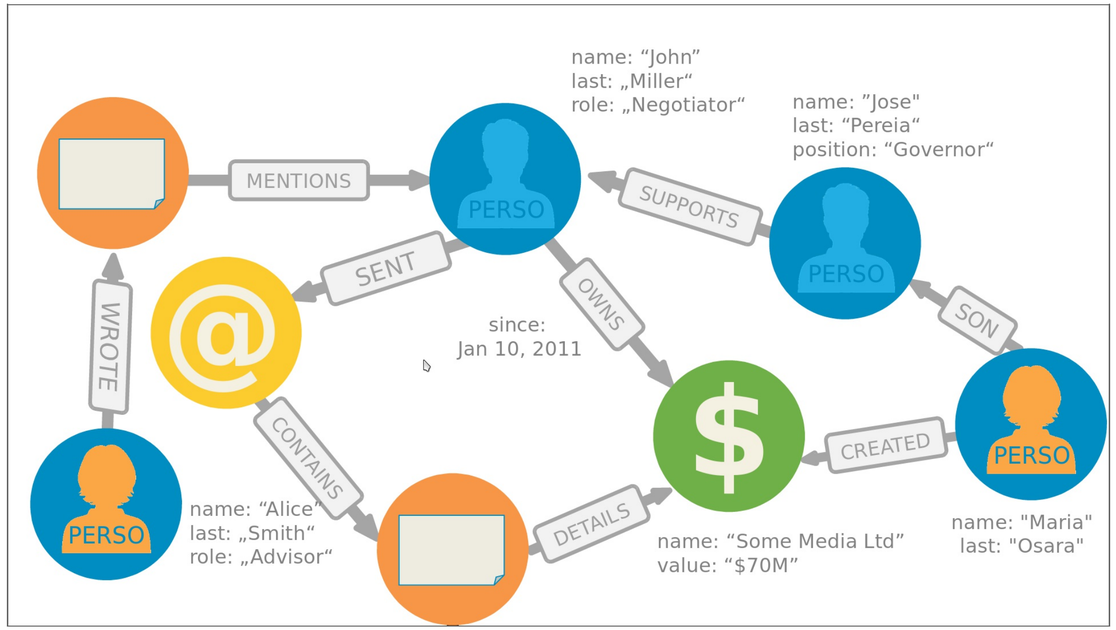




Focus on Neo4j



A graph of properties



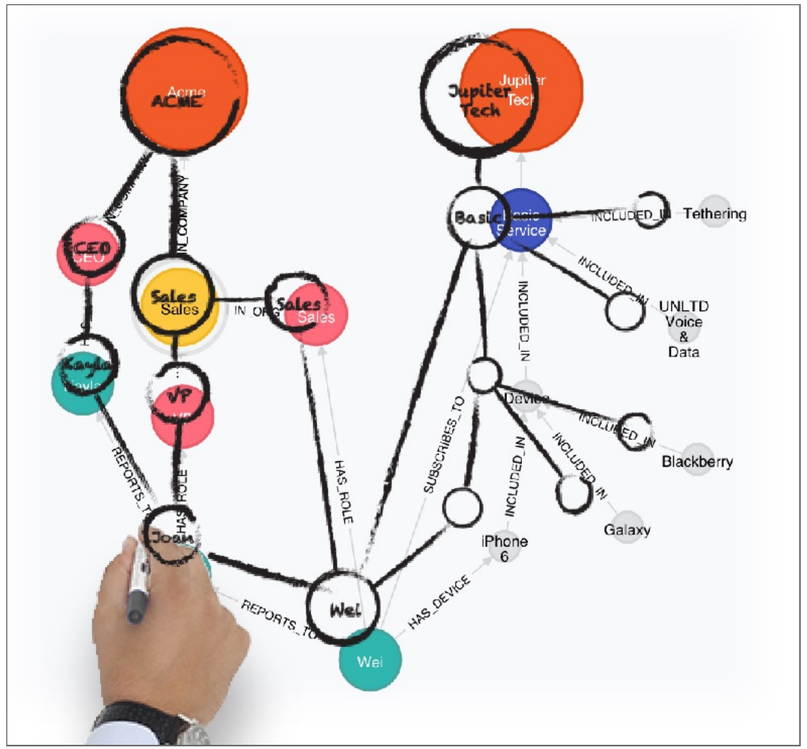
Nodes

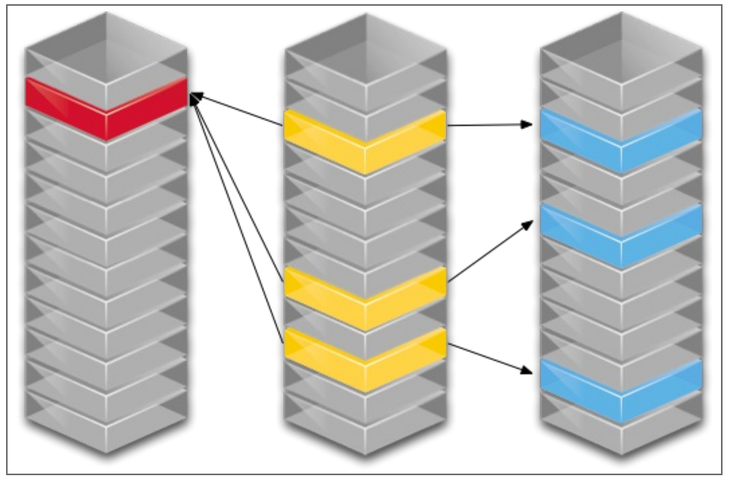
- The entity of your model
- Can have labels
- Can have properties

Relationships

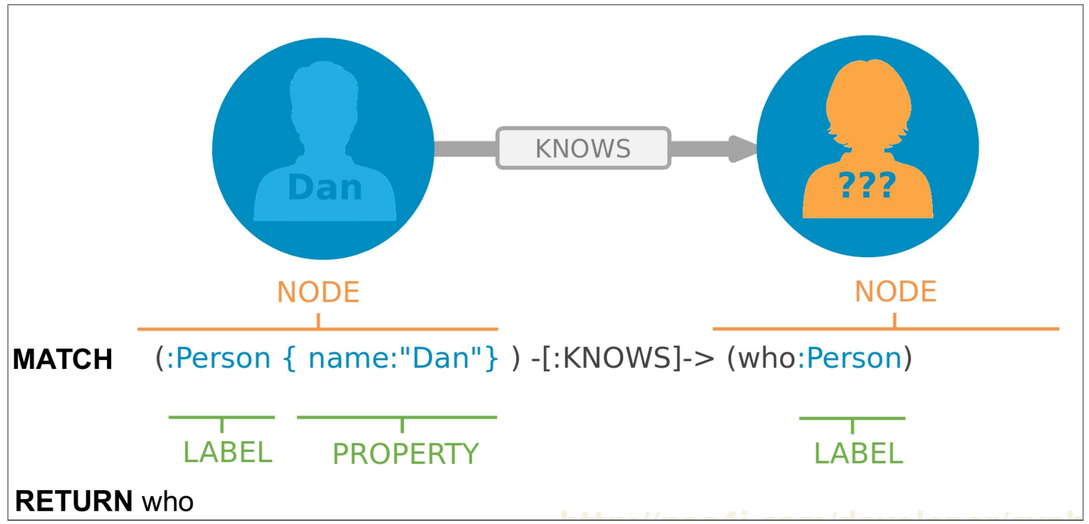
- Links two nodes, with a **direction** and a **type**
- Can have properties

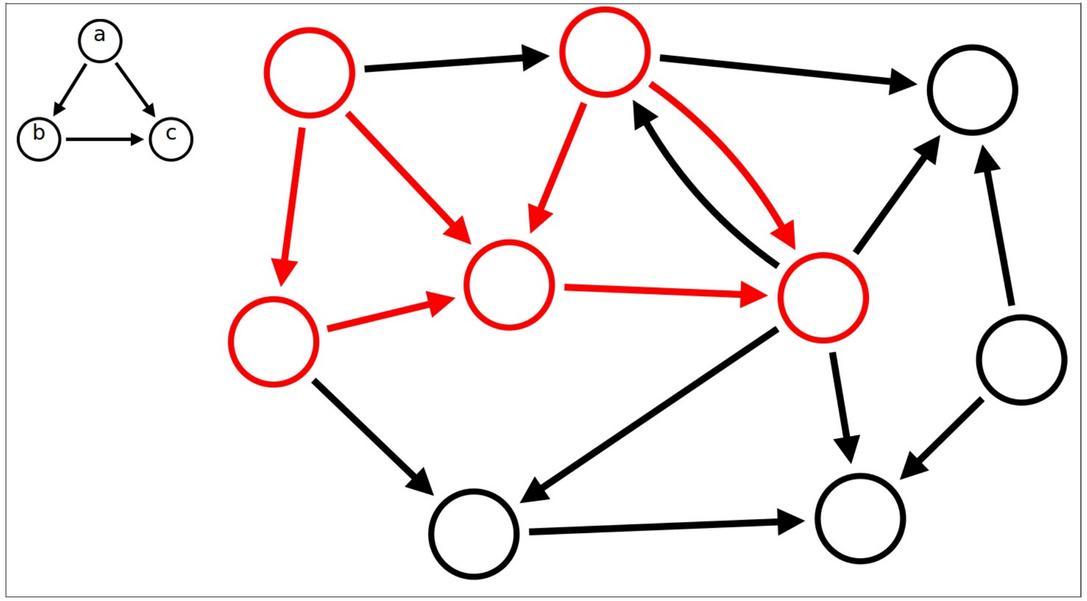
A local approach





Cypher


Dan
KNOWS
???
NODE
NODE
MATCH (:Person { name:"Dan"}) -[:KNOWS]-> (who:Person)
LABEL
PROPERTY
LABEL
RETURN who

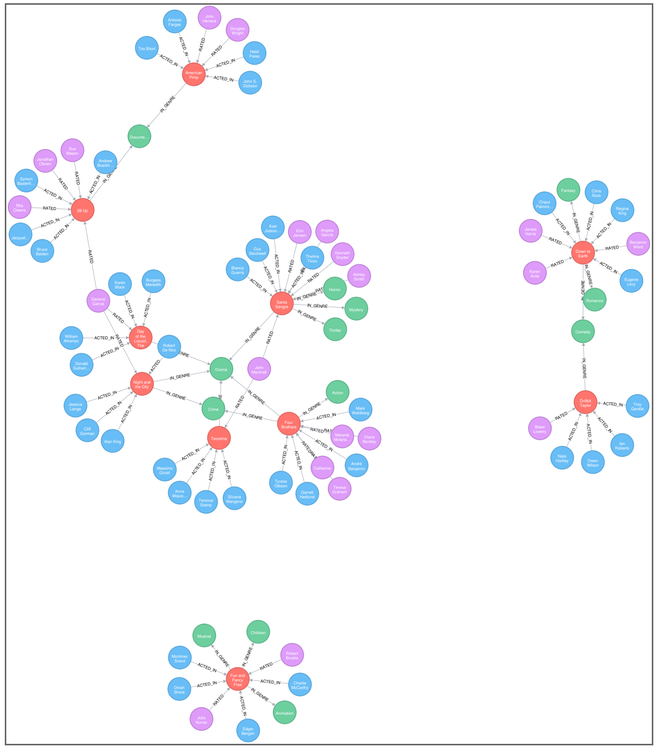

a
b
c

The movie dataset

https://neo4j.com/sandbox-v2/

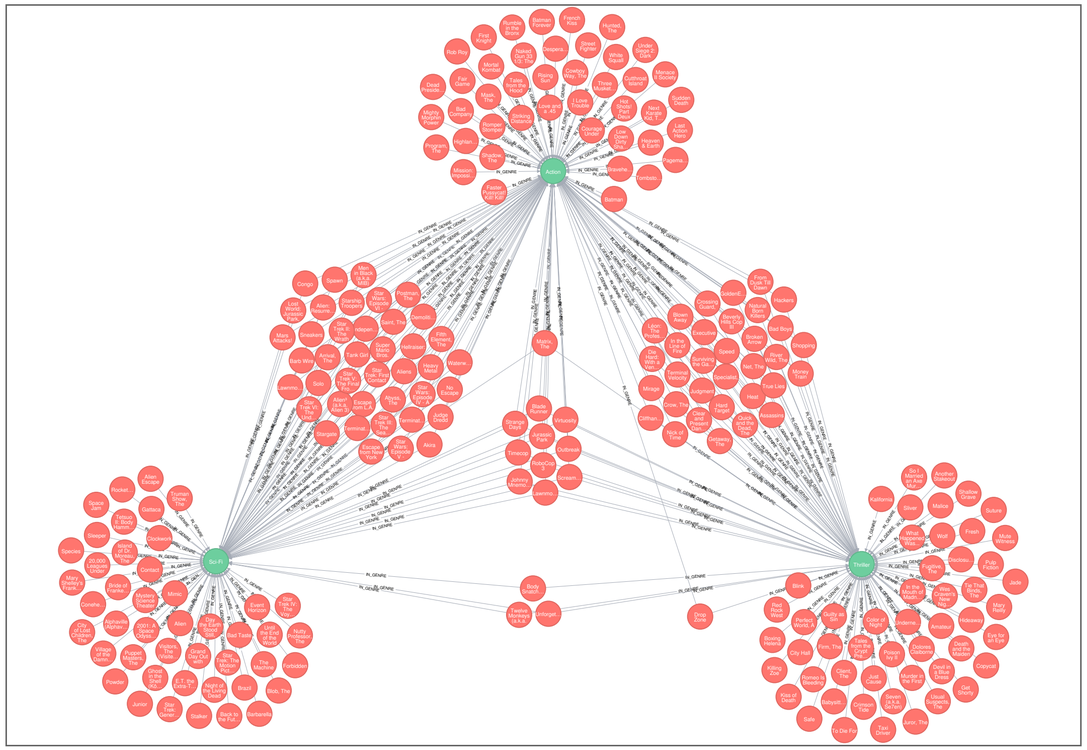
Recommendations

Generate personalized real-time recommendations using a dataset of movie reviews.

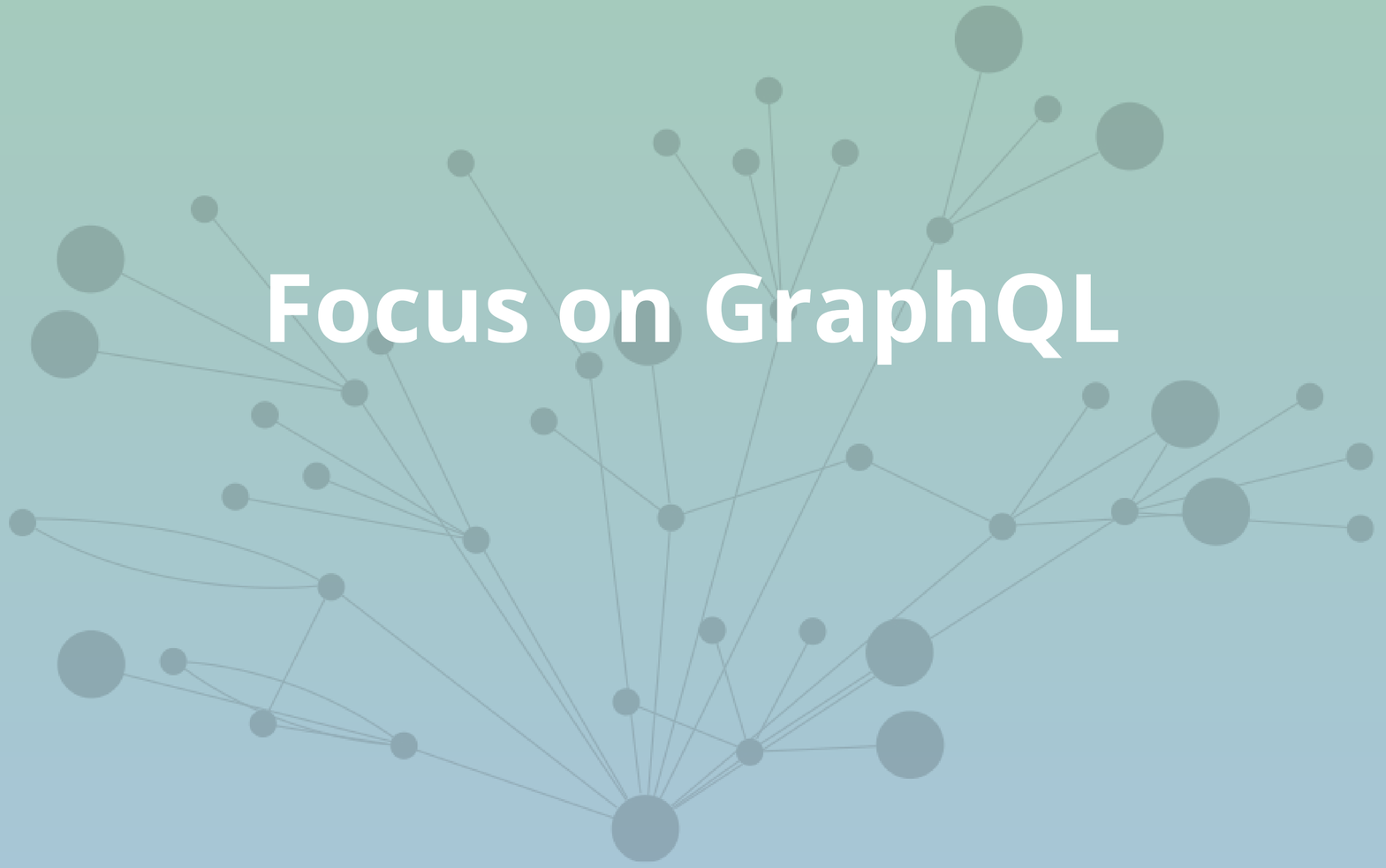


Recommendation

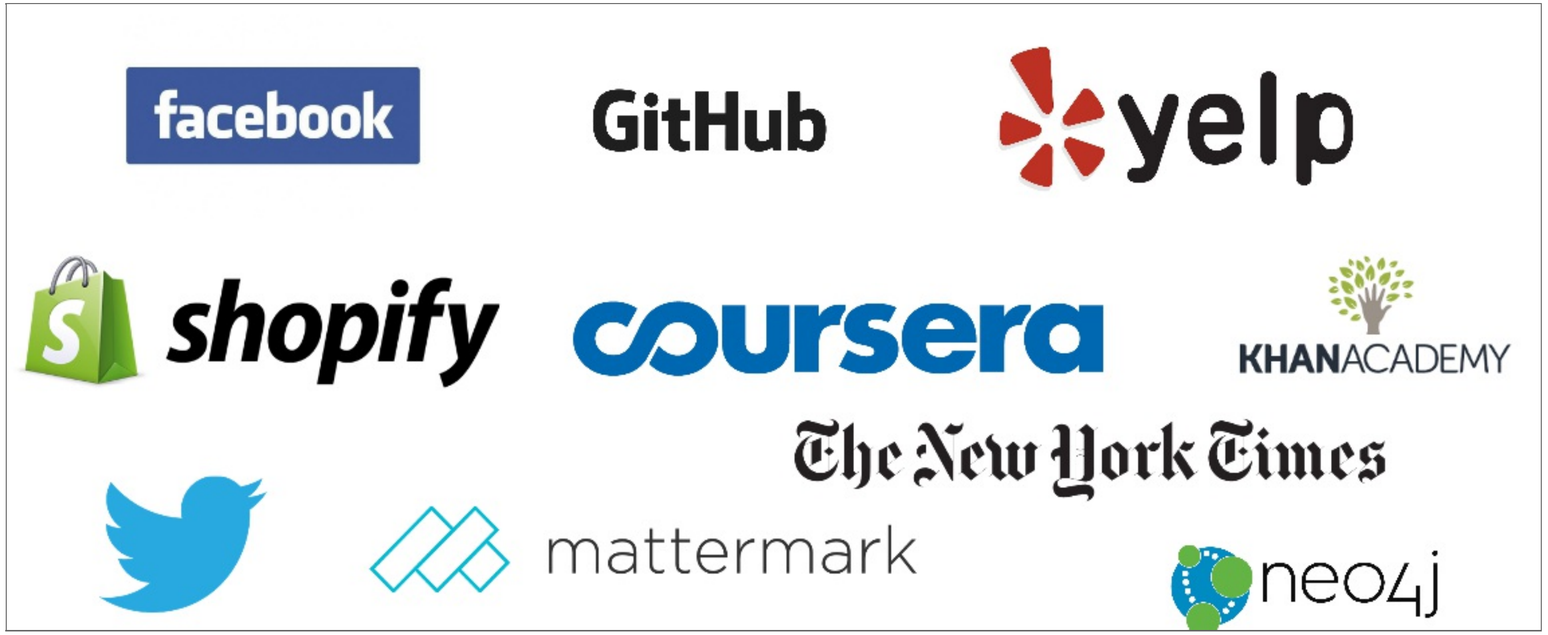
```
MATCH
  (m:Movie
```



Focus on GraphQL



A large adoption

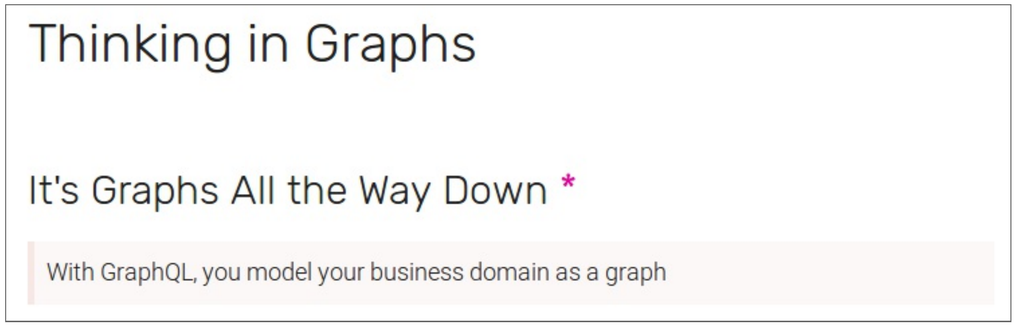


GraphQL

“A query language for your API, and a server-side runtime for executing queries by using a type system you define for your data”

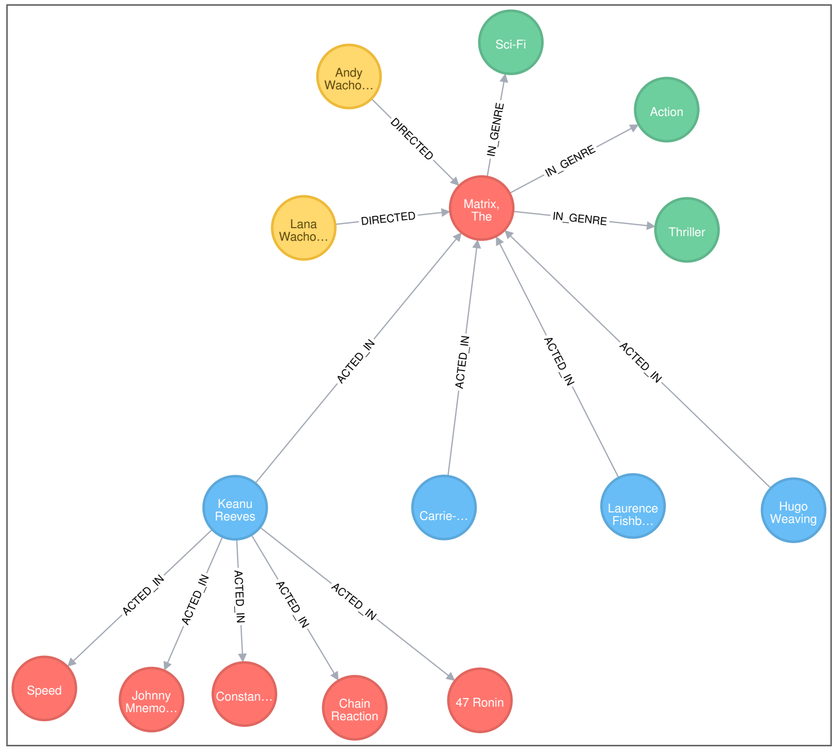
“GraphQL isn’t tied to any specific database or storage engine”

“A GraphQL service is created by defining types and fields on those types, then providing resolver functions for each field on each type”



Your application model is a Graph !

```
query($id:ID = 2571)
{
```



Your application model is a Graph !

```
query($id:ID = 2571)
{
```

```
{
  "Movie": {
```

Schema Definition Language

GraphQL Cheat Sheet

- **Type :** The graph model definition
- **Resolvers :** How to fetch data from datasource
- **Query :** What queries you can do
- **Mutations :** What changes you can do

Type

Movie

```
type Movie {
  movieId: ID!
```

Actor

```
interface Person {
  name: ID!
```

Queries / Mutation

Movie

```
type Query {
  Movie(movieId:
```

Actor

```
type Query {
  Actor(id: ID!):
```

Resolvers

```
Movie {
```

Use The Schema, Luke

neo4j-graphql-js

Translate your schema directly in Cypher, thanks to annotation.

```
{
  Movie(title: "River Runs Through It, A") {
    title
    year
    imdbRating
    actors {
      name
    }
  }
}
```

```
MATCH (movie:Movie {title:"River Runs Through It, A"})
RETURN movie { .title , .year , .imdbRating,
  actors: [(movie)<-[ACTED_IN]-(movie_actors:Actor) | movie_actors { .name }] }
AS movie
SKIP 0
```

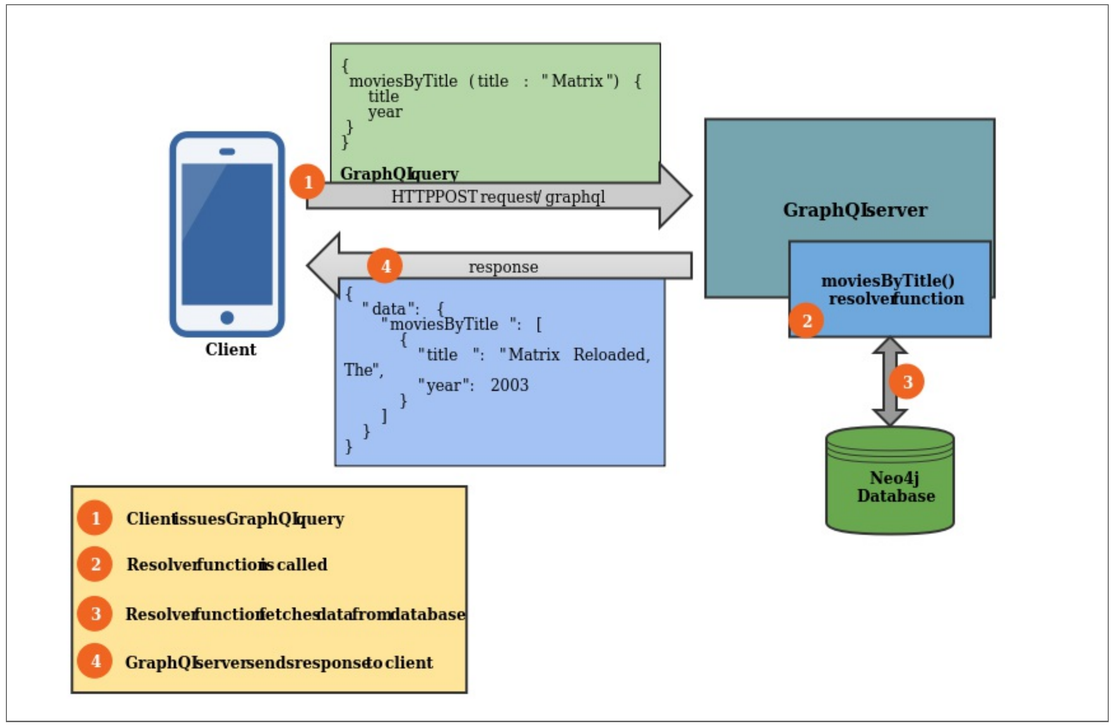
openCypher

```
type Movie {
    movieId: ID!
    title: String
    year: Int
    plot: String
    poster: String
    imdbRating: Float
    genres: [String]
    similar(first: Int = 3, offset: Int = 0): [Movie] @cypher(statement: "WITH {this} AS this MATCH (this)--(:Genre)--(o:Movie) RETURN o")
    mostSimilar: Movie @cypher(statement: "WITH {this} AS this RETURN this")
    degree: Int @cypher(statement: "WITH {this} AS this RETURN SIZE((this)--())")
    actors(first: Int = 3, offset: Int = 0): [Actor] @relation(name: "ACTED_IN", direction:"IN")
    avgStars: Float
    filmedIn: State @relation(name: "FILMED_IN", direction: "OUT")
}
```

Focus on Apollo

GraphQL tool box

- **Server :** Apollo server helps you to create GraphQL server
- **Client:** Apollo client helps you to communicate with a GraphQL server



Server

Just an express.js server with some custom endpoints.

```
import express from
'express';
```

Server - II

```
/**
 * This part is
```

Client - Angular - I

In your application module :

```
// Apollo & GraphQl
import { HttpModule
```

Client - Angular - II

In your components :

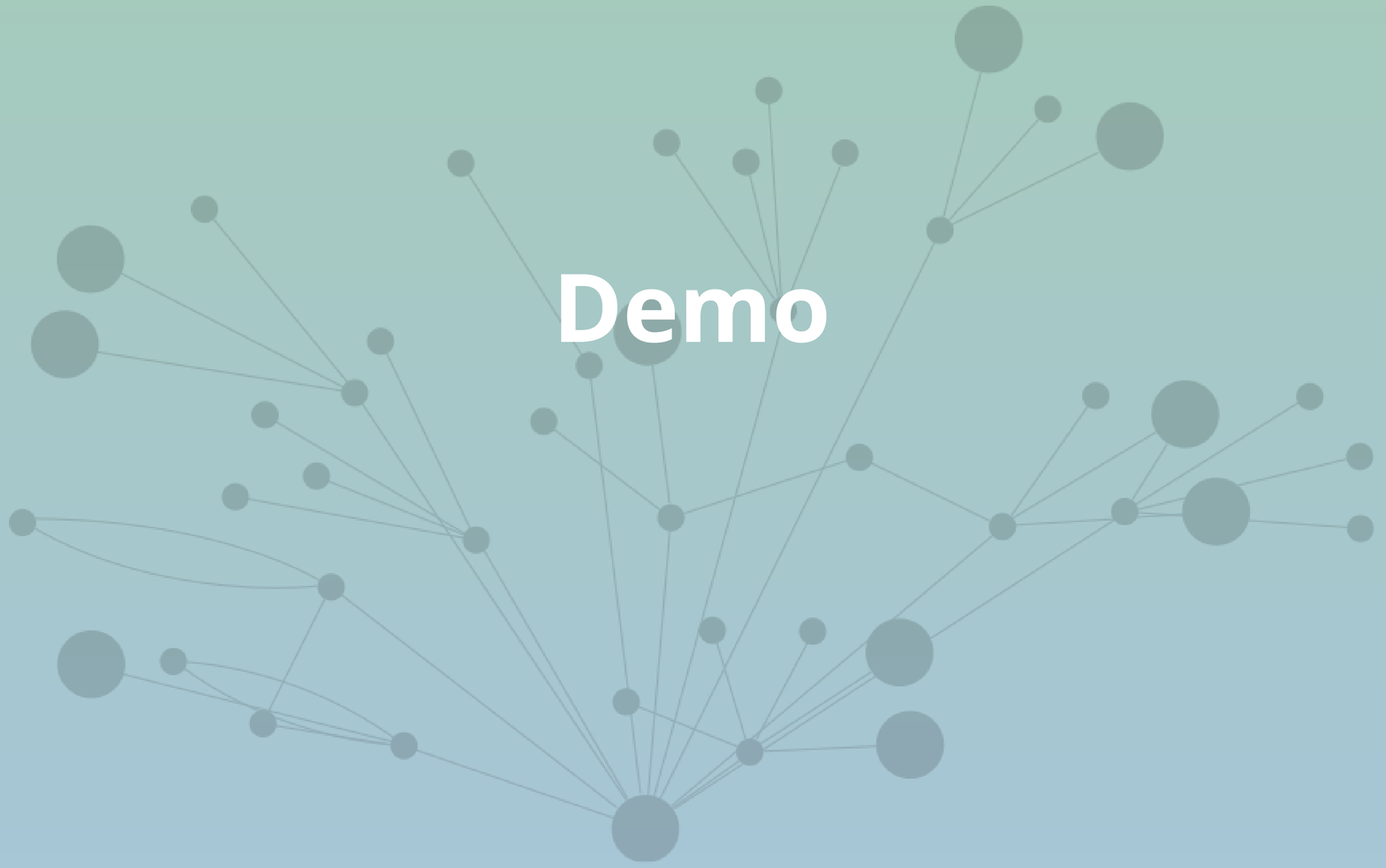
```
import { Apollo }
from 'apollo-
```

Typescript generation

https://github.com/apollographql/apollo-codegen

```
cd
~/worspaces/grandsta
```

Demo



Any questions ?

Thanks

Resource

Thanks to Will Lyon from Neo4j for the help