GraphQL overview

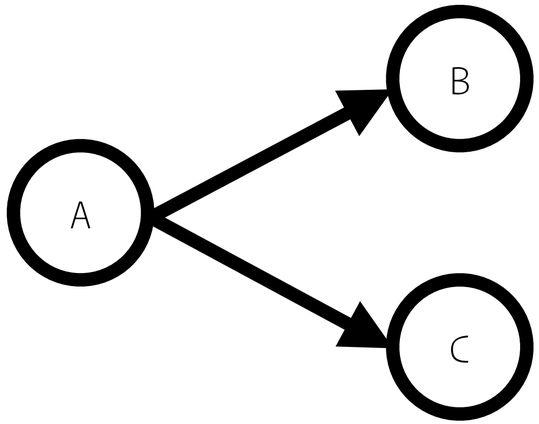
Learn what GraphQL is in under 3 minutes

You can download this presentation as PDF

GraphQL

What is it?

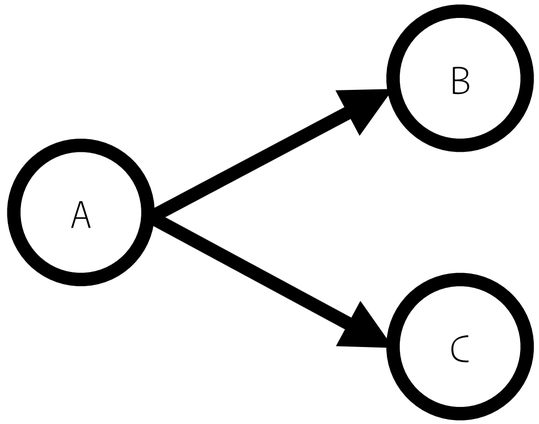
- Graph is a set of objects, where some pairs of that objects can be related (discrete math)
- For now let’s assume object is something abstract
- QL stands for Query Language
- Another example of a Query Language is the well known SQL

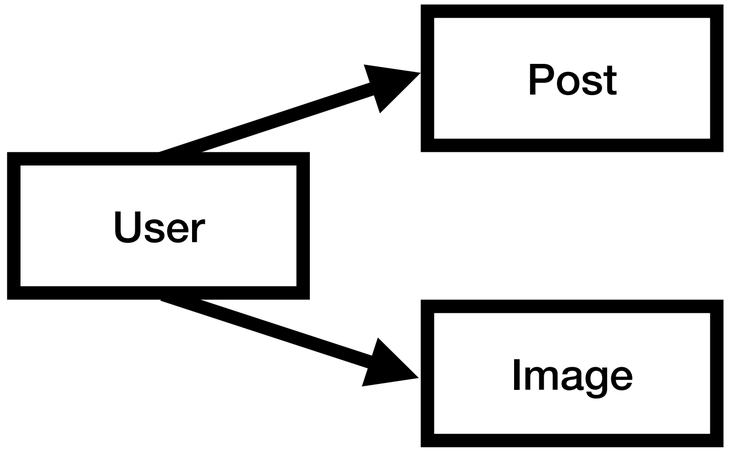


Why Graphs?

Representing related objects

- The graph with `A`, `B` and `C` can represent objects
- An object might represent a database row
- Database rows have relations with other tables

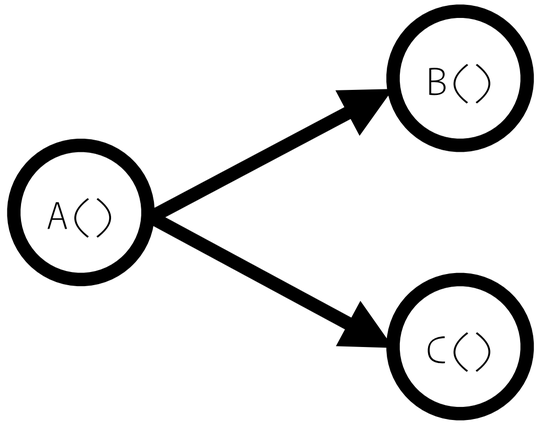


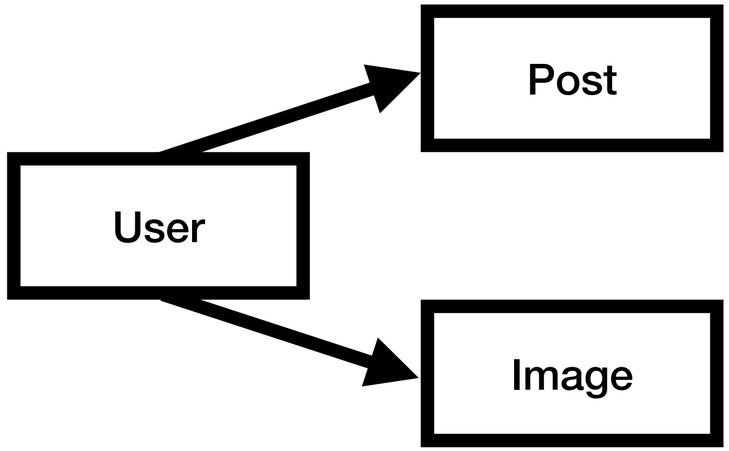


Querying data

How does GQL query work

- A, B and C are functions that return data
- These are called **Resolvers**
- Resolver functions are called to produce next value
- Resolver functions are called until all objects returned contain only scalars (plain value like integer, string)





GraphQL query

A look at GraphQL query

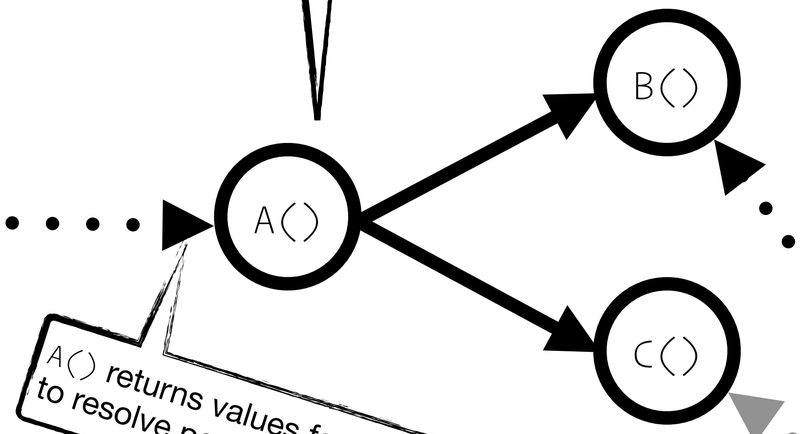
Query name

```
query UserWithPostAndImage {
  user {
    id
    name
    post {
      id
      title
    }
    image {
      url
    }
  }
}
```

Scalars

Fields query asks for

Resolver function job is to return scalar field values (integer, string) and call next resolver for complex field types (object types)



Resolver A() resolves scalar values of user

A() returns values for id, name and calls B() to resolve post {...} and C() to resolve image

Resolver B() resolves scalar values of post

Resolver C() resolves scalar values of image

SQL Query

Compare GQL with SQL

Query name

Fields query asks for

```
query UserWithPostAndImage {
  user {
    id
    name
    post {
      id
      title
    }
    image {
      url
    }
  }
}
```

GraphQL query is like SQL SELECT operation - used only to read the data

```
SELECT
    user.id,
    user.name,
    post.id,
    post.title,
    image.url
FROM users
JOIN post ON post.user_id = user.id
JOIN image ON image_user_id = user.id
```

JSON

Query returns JSON data

Query name

Fields query asks for

```
query UserWithPostAndImage {
  user {
    id
    name
    post {
      id
      title
    }
    image {
      url
    }
  }
}
```

Response contains only the data query asked for

```
{
  "data": {
    "user": {
      "id": 1,
      "name": "Piotr Jura",
      "post": {
        "id": 1,
        "title": "Welcome to GQL"
      },
      "image": {
        "url": "http://laravello.xyz/image.png"
      }
    }
  }
}
```

Mutations

For modifying data

Every variable starts with a $ dollar, then comes the name, finally the type. ! means the it's non nullable

Variables allow arguments to contain dynamic values. It means, you don't have to put the value of argument directly into query/mutation string

Variables

„mutation" keyword

Just like with queries, you might want to specify what fields should be returned, if the return type is an Object. Return type can be scalar too (eg. Boolean or Int)

Expected fields

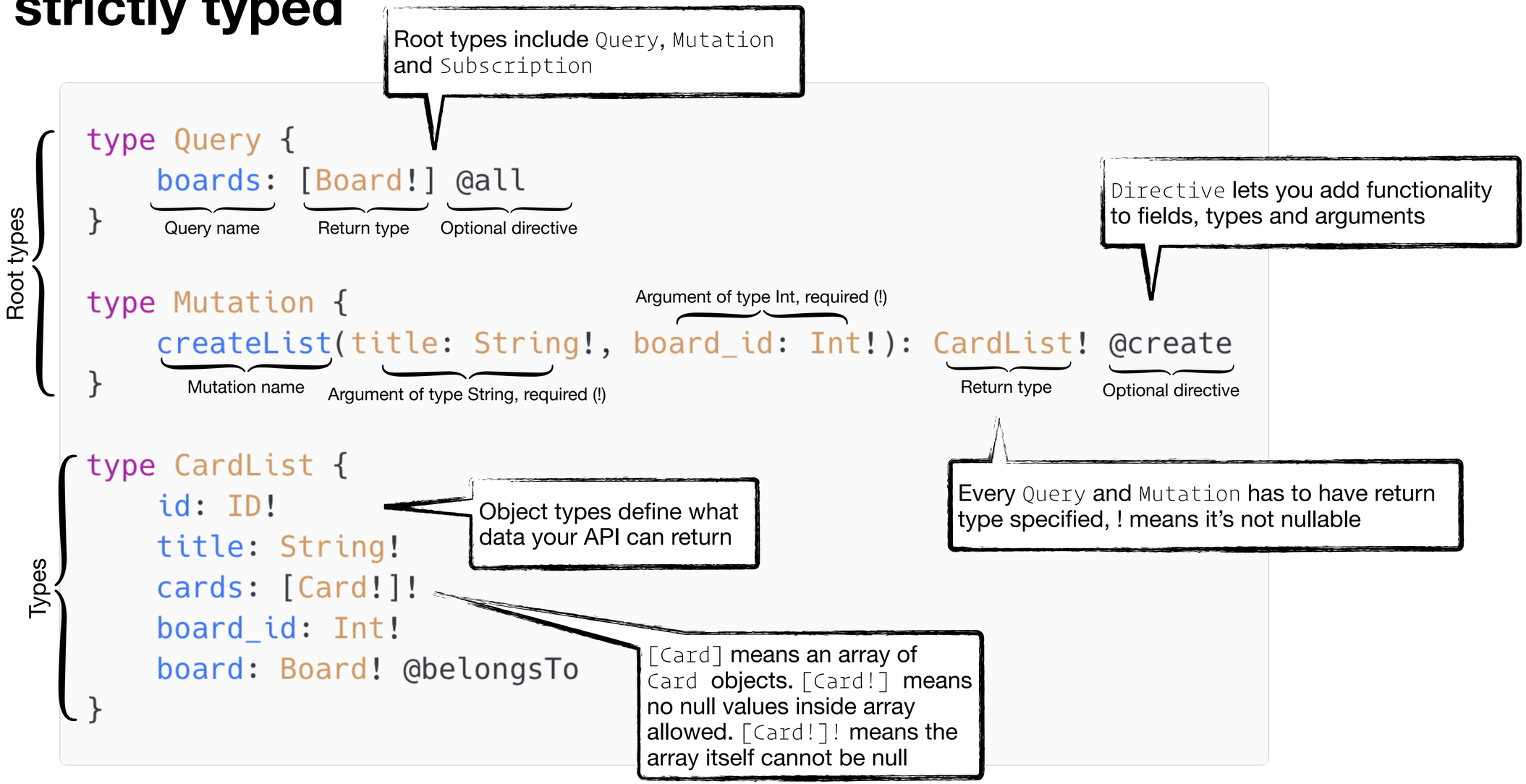
```
mutation($title: String!, $board: Int!) {
  createList(title: $title, board_id: $board) {
    id
    title
  }
}
```

Arguments

Mutations usually take some arguments (queries do too!)

GraphQL Schema

GQL is strictly typed



GraphQL types

Recap

- Query can (only) be used to fetch data
- Mutation can (only) be used to add/modify data
- Subscription to get real time updates from server (not covered in the course)
- Object types define the structure and type of your data

GraphQL Server and Client

Recap

- GraphQL is a query language
- GraphQL is only a specification
- It’s not tied to any programming language on the server (Node.js, PHP, Python)
- It’s not tied to any programming language on the client (JavaScript, Swift)
- It’s strongly typed (every argument, return value, field on object has to have a type specified)