GraphQL Introduction

What is GraphQL : Why Use GraphQL : GraphQL vs Rest :
Who Uses GraphQL : GraphQL Development Tools : Working Demo

What is GraphQL?

GraphQL

What is GraphQL?

GraphQL

- Created by a Facebook internal development team in 2012

Created by:

- Nick Schrock
- Dan Schafer
- Lee Byron

Open-Sourced in 2015

In November 2018 the GraphQL Project was moved to the newly-established GraphQL foundation hosted by the non-profit Linux Foundation

What is GraphQL?

- Created by a Facebook internal development team in 2012
- GraphQL is a query language for APIs

Instead of multiple endpoints to retrieve data, a query is sent to one endpoint and the response only includes what was asked for

Query:

```
query {
    player(id: 1) {
        playerName
    }
}
```

Response:

```
{
    "data": {
        "player": {
            "playerName": "Phil Mickelson"
        }
    }
}
```

What is GraphQL?

GraphQL

- Created by a Facebook internal development team in 2012
- GraphQL is a query language for APIs
- Based on the idea of Schemas

The center of any GraphQL implementation

Defines the way the data is structured with object types and fields

Describes the functionality available to the clients that connect

Creates the relationships between types

Handles the data input validation for querying and mutating data

Large schemas can be broken down into smaller more manageable and scoped schemas

What is GraphQL?

- Created by a Facebook internal development team in 2012
- GraphQL is a query language for APIs
- Based on the idea of Schemas
- Gives clients the power to ask for exactly what they need and nothing more

Clients have the power to only ask for what they need

Only the fields and related data objects in the query are returned.

Allows to get multiple related data objects with one request saving on the amount of api calls required

Server only has to process the data asked for, so if a more costly calculation is part of the data object the server only needs to make that calculation if requested.

What is GraphQL?

- Created by a Facebook internal development team in 2012
- GraphQL is a query language for APIs
- Based on the idea of Schemas
- Gives clients the power to ask for exactly what they need and nothing more
- Makes it easier to evolve APIs over time

GraphQL Schema makes it easy to visualize your data structures which in turn minimize accidental breaking changes

Adding fields or relationships to types won't break the original functionality.

You can evolve your API without versioning

Continuous Evolution

API evolution is the concept of striving to maintain the "I" in API, the request/response body, query parameters, general functionality, etc., only breaking them when you absolutely, absolutely, have to.

Phil Sturgeon

Who is Using GraphQL?

GraphQL

intuit

ATLASSIAN

NBC NEWS

GitHub

twitter

PayPal

facebook

Pinterest

SERVERLESS

shopify

wayfair

Why Use GraphQL?

GraphQL

Why Use GraphQL?

- GraphQL **is fast**

You only request the data you need.

- Any data fields with more overhead are only requested when they are needed.
- Only the relationship data that has been requested is returned.
- Getting the related data you need in one call saving you the many API requests normally required to get the data needed.
- Instead of the server deciding what to send back the client makes the decision.

Why Use GraphQL?

- GraphQL is fast
- GraphQL **is flexible**

Probably it’s biggest advantage

- Ask only for what you need and only get back what you ask for
- Desktop or Web Apps with larger screens may want to load more data since there is more screen real estate to use.
- While the Mobile Apps have much smaller screens and can only display the basic data.
- The server only has to do the minimal amount of processing to satisfy the specific platform.

Why Use GraphQL?

- GraphQL is fast
- GraphQL is flexible
- GraphQL **is easy to use and easy to maintain**

New data fields and relational data can be added without introducing a breaking change.

- Simply add the field to the schema
- May not even have to modify the resolver
- For relational data simply add an object resolver

Everything will work just as before. The new fields and data can then be implemented in the client as needed

Why Use GraphQL?

GraphQL

- GraphQL is fast
- GraphQL is flexible
- GraphQL is easy to use and easy to maintain
- GraphQL **has native support for subscriptions (PubSub)**

With graphql subscriptions are easy and built-in

Use resolvers much as you would in your other query and mutation resolvers.

(the demo has an example of this in action)

GraphQL Operations?

GraphQL uses three types of operations (or query types):
Query, Mutation, and Subscription

GraphQL Operations?

- Query

The query type is used to retrieve information from the API.

This example gets the player with id of 1 and returns some player fields, a list of rounds, and course data.

```
query {
  player(id: 1) {
    id
    playerName
    rounds {
      roundDate
      score
      course {
        courseName
      }
    }
  }
}
```

GraphQL Operations?

GraphQL

- Query
- **Mutation**

Mutation queries are used to “mutate” the data. This replaces the create, update, and delete operations of a REST API.

This example creates a player and sets the playerName.

```
mutation {
 addPlayer (playerName: "Jimi Hendrix"){
  id
  playerName
 }
}
```

GraphQL Operations?

- Query
- Mutation
- **Subscription**

Subscriptions allow the client so subscribe to a channel. When there is an update that publishes to the subscription the client will be notified.

The client also has the ability to choose what will be returned

```
subscription {
  roundUpdated(roundId: 1) {
    id
    roundDate
    player {
      playerName
    }
  }
}
```

GraphQL Operations?

- Query
- Mutation
- Subscription

Resolvers (not an operation, but worth mentioning here)

Resolvers are not operations, but they are the part that makes the operations functional.

It doesn’t matter if it’s a Query or Mutation it is backed by a Resolver.

A Resolver is basically the function that is called when the operation is called.

The data needed for the Resolver to do its job is injected into the function.

```
function(parent, args, ctx, info) => {
     // Do work here
     // Examples shown in demo
}
```

GraphQL vs. Rest

GraphQL vs. Rest

	GraphQL	Rest
Architecture:	client-driven	server-driven

GraphQL vs. Rest

GraphQL

	GraphQL	Rest
Architecture:	client-driven	server-driven
Organized in Terms of:	schema and type system	endpoints

GraphQL vs. Rest

	GraphQL	Rest
Architecture:	client-driven	server-driven
Organized in Terms of:	schema and type system	endpoints
Operations:	query, mutation, subscription	create, read, update, delete

GraphQL vs. Rest

GraphQL

	GraphQL	Rest
Architecture:	client-driven	server-driven
Organized in Terms of:	schema and type system	endpoints
Operations:	query, mutation, subscription	create, read, update, delete
Data Fetching:	specific data with single call	fixed data with multi calls

GraphQL vs. Rest

	GraphQL	Rest
Architecture:	client-driven	server-driven
Organized in Terms of:	schema and type system	endpoints
Operations:	query, mutation, subscription	create, read, update, delete
Data Fetching:	specific data with single call	fixed data with multi calls
Community:	growing	large

GraphQL vs. Rest

	GraphQL	Rest
Architecture:	client-driven	server-driven
Organized in Terms of:	schema and type system	endpoints
Operations:	query, mutation, subscription	create, read, update, delete
Data Fetching:	specific data with single call	fixed data with multi calls
Community:	growing	large
Performance:	fast	multi network calls takes time

GraphQL vs. Rest

	GraphQL	Rest
Architecture:	client-driven	server-driven
Organized in Terms of:	schema and type system	endpoints
Operations:	query, mutation, subscription	create, read, update, delete
Data Fetching:	specific data with single call	fixed data with multi calls
Community:	growing	large
Performance:	fast	multi network calls takes time
Development Speed:	rapid	slower

GraphQL vs. Rest

GraphQL

	GraphQL	Rest
Architecture:	client-driven	server-driven
Organized in Terms of:	schema and type system	endpoints
Operations:	query, mutation, subscription	create, read, update, delete
Data Fetching:	specific data with single call	fixed data with multi calls
Community:	growing	large
Performance:	fast	multi network calls takes time
Development Speed:	rapid	slower
Self Documenting:	yes	no

GraphQL vs. Rest

	GraphQL	Rest
Architecture:	client-driven	server-driven
Organized in Terms of:	schema and type system	endpoints
Operations:	query, mutation, subscription	create, read, update, delete
Data Fetching:	specific data with single call	fixed data with multi calls
Community:	growing	large
Performance:	fast	multi network calls takes time
Development Speed:	rapid	slower
Self Documenting:	yes	no
File Uploads:	no	yes

What is a Graph?

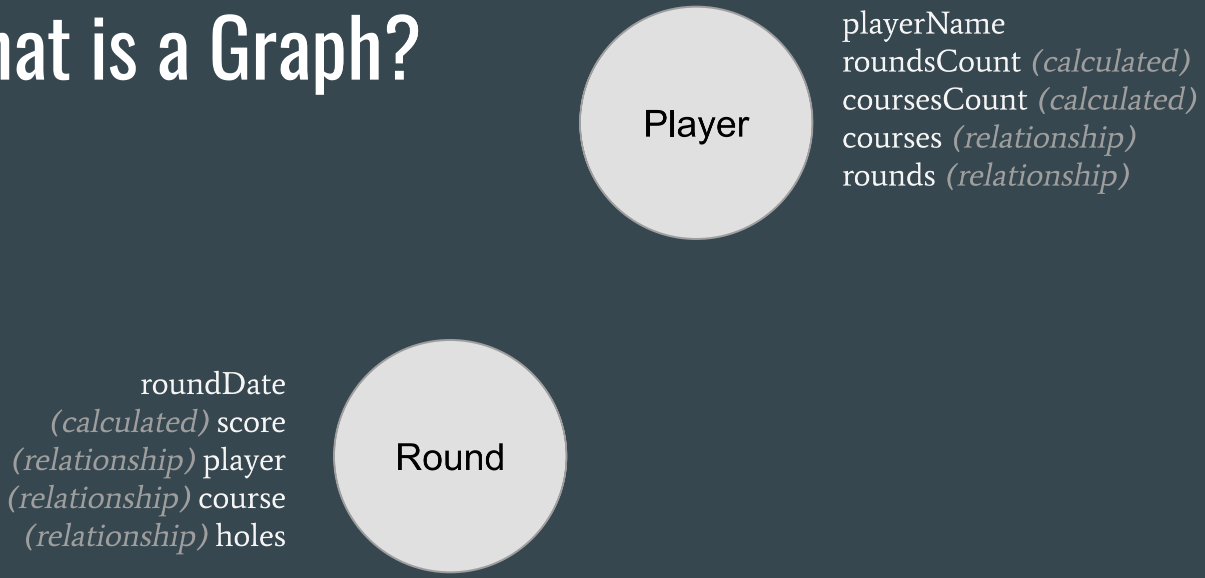
GraphQL

What is a Graph?

playerName
roundsCount *(calculated)*
coursesCount *(calculated)*
courses *(relationship)*
rounds *(relationship)*

GraphQL

What is a Graph?



What is a Graph?

GraphQL

playerName
roundsCount *(calculated)*
coursesCount *(calculated)*
courses *(relationship)*
rounds *(relationship)*

roundDate
(calculated) score
(relationship) player
(relationship) course
(relationship) holes

number
Score
(relationship) round
(relationship) course
(relationship) courseHole

What is a Graph?

Player

playerName
roundsCount *(calculated)*
coursesCount *(calculated)*
courses *(relationship)*
rounds *(relationship)*

Round

roundDate
(calculated) score
(relationship) player
(relationship) course
(relationship) holes

Course

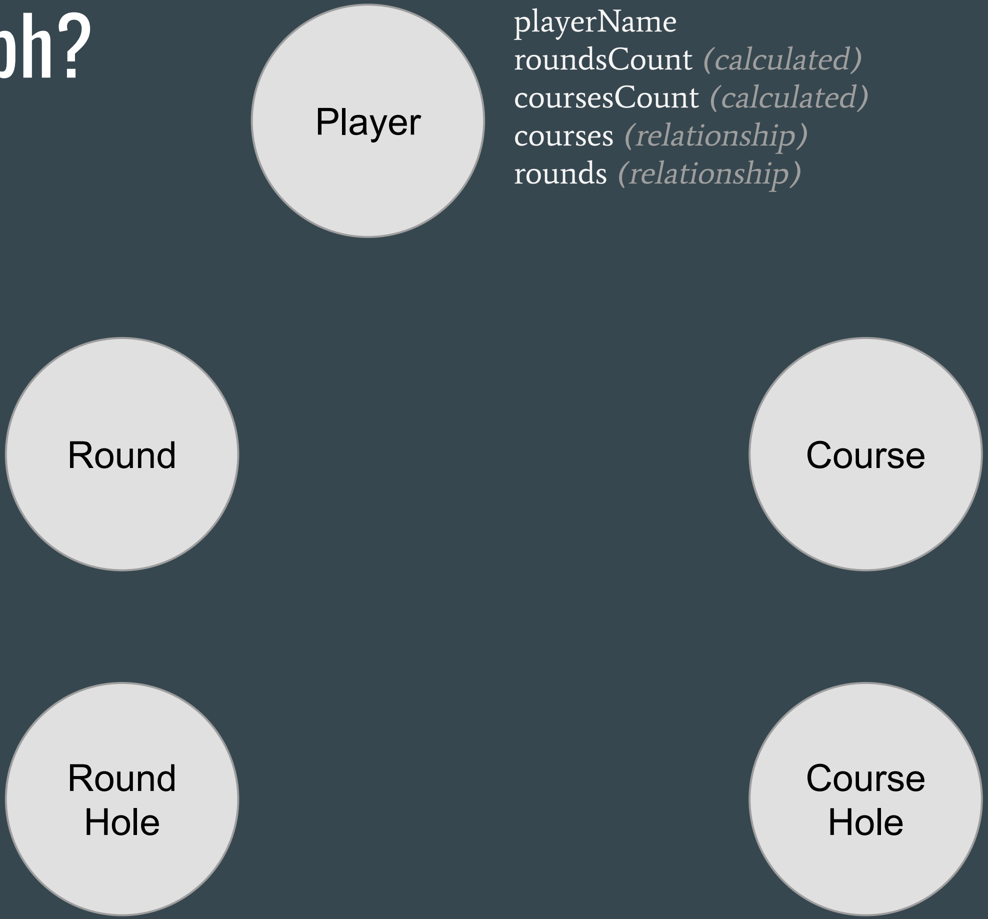
courseName
city
state
zip
roundsCount *(calculated)*
par *(calculated)*
rounds *(relationship)*
holes *(relationship)*
players *(relationship)*

Round Hole

number
Score
(relationship) round
(relationship) course
(relationship) courseHole

What is a Graph?

GraphQL



playerName
roundsCount *(calculated)*
coursesCount *(calculated)*
courses *(relationship)*
rounds *(relationship)*

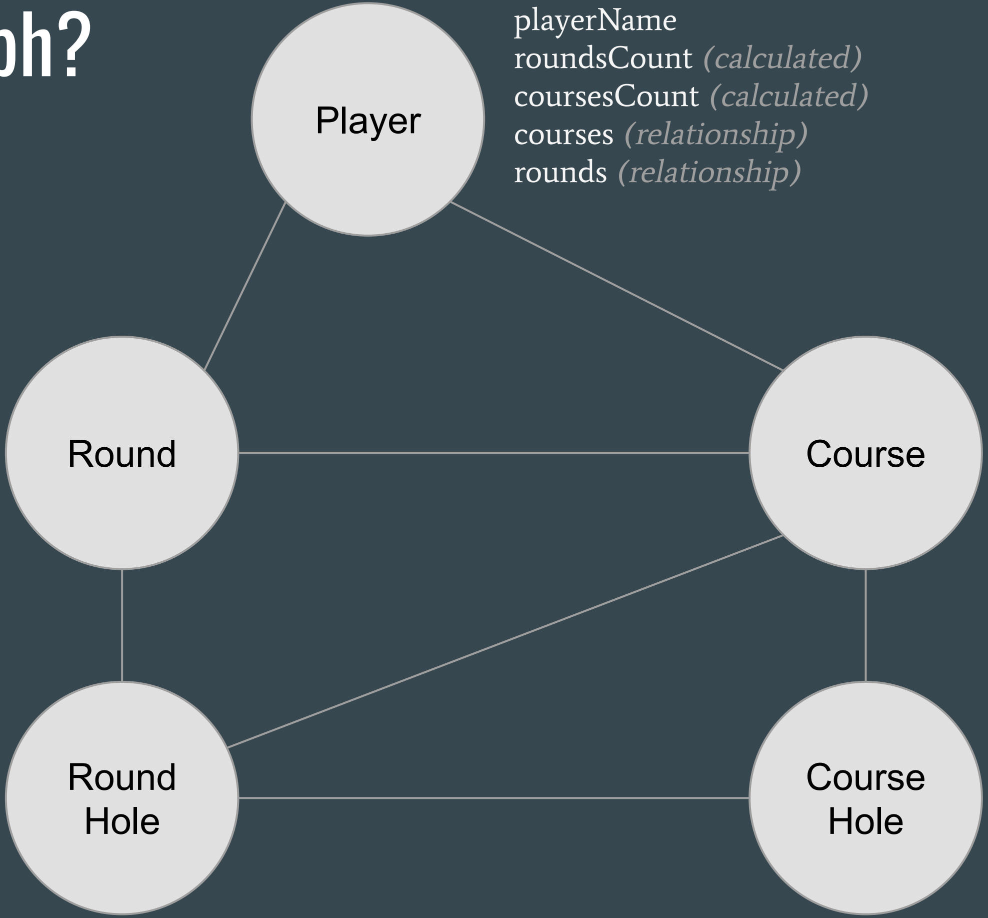
roundDate
(calculated) score
(relationship) player
(relationship) course
(relationship) holes

courseName
city
state
zip
roundsCount *(calculated)*
par *(calculated)*
rounds *(relationship)*
holes *(relationship)*
players *(relationship)*

number
Score
(relationship) round
(relationship) course
(relationship) courseHole

number
par
handicap

What is a Graph?



playerName
roundsCount *(calculated)*
coursesCount *(calculated)*
courses *(relationship)*
rounds *(relationship)*

roundDate
(calculated) score
(relationship) player
(relationship) course
(relationship) holes

courseName
city
state
zip
roundsCount *(calculated)*
par *(calculated)*
rounds *(relationship)*
holes *(relationship)*
players *(relationship)*

number
Score
(relationship) round
(relationship) course
(relationship) courseHole

number
par
handicap

What is a Graph?

- The Players Example

```
query {
  player(id: 2) {
    id
    playerName
  }
}
```

What is a Graph?

- The Players Example

```
query {
  player(id: 2) {
    id
    playerName
  }
}
```

```
{
  "data": {
    "player": {
      "id": 2,
      "playerName": "Phil Mickelson"
    }
  }
}
```

What is a Graph?

- The Players Example with Related Data

```
query {
  player(id: 2) {
    id
    playerName
    rounds {
      roundDate
      score
      course {
        courseName
      }
    }
  }
}
```

What is a Graph?

GraphQL

- The Players Example with Related Data

```
query {
  player(id: 2) {
    id
    playerName
    rounds {
      roundDate
      score
      course {
        courseName
      }
    }
  }
}
```

```
{
  "data": {
    "player": {
      "id": 2,
      "playerName": "Phil Mickelson",
      "rounds": [
        {
          "roundDate": "2019-09-01",
          "score": 92,
          "course": {
            "courseName": "SunRiver Golf Club"
          }
        },
         ...
      ]
    }
  }
}
```

Migrate REST to GraphQL

GraphQL

- It’s not too hard.
- Cover that another time...

GraphQL Development Tools

- **Relay** - A JavaScript framework for building data-driven React applications powered by GraphQL
- **Prisma** - Prisma is a database abstraction layer that turns your databases into GraphQL APIs with CRUD operations and real time capabilities
- **Vue-GraphQL** - A GraphQL client for the vue framework
- **GraphQL Editor** - Build, manage and collaborate on your schema
- **GraphQL Playground** - A GraphQL IDE for quick building and testing queries
- **GraphQL Voyager** - Create a data relational view of your data using your GraphQL Schema
- **GraphQL Docs** - A static documentation generator using your GraphQL Schema

Languages with GraphQL Implementations

GraphQL

PHP

- Graphql-PHP : https://github.com/webonyx/graphql-php
- Graphql-Laravel : https://github.com/rebing/graphql-laravel
- GraphQLBundle (symfony) : https://github.com/overblog/GraphQLBundle

Node

- GraphQL-Yoga : https://github.com/prisma/graphql-yoga
- Apollo-Server : https://github.com/apollographql/apollo-server/tree/master/packages/apollo-server-express

Python

- Graphene : https://github.com/graphql-python/graphene

Perl

- GraphQL-Perl : https://github.com/graphql-perl/graphql-perl

GoLang

- GraphQL-Go : https://github.com/graphql-go/graphql

Working Demo

GraphQL

The demo is a Golf players round tracker using GraphQL. You can add players, golf rounds, and courses.

Once the Demo is up and running you can use either the playground or postman to query and mutate the database.

GitHub Location: https://github.com/unitiweb/golf-players-graphql-demo

The instructions to get the demo up and running are in the README.md file.

Let's Look at the Demo ;-)

...

Any Questions

...

?