An Introduction to GraphQL
Tutorial at ISWC 2019, October 27, 2019

2. Schemas and the Query Language

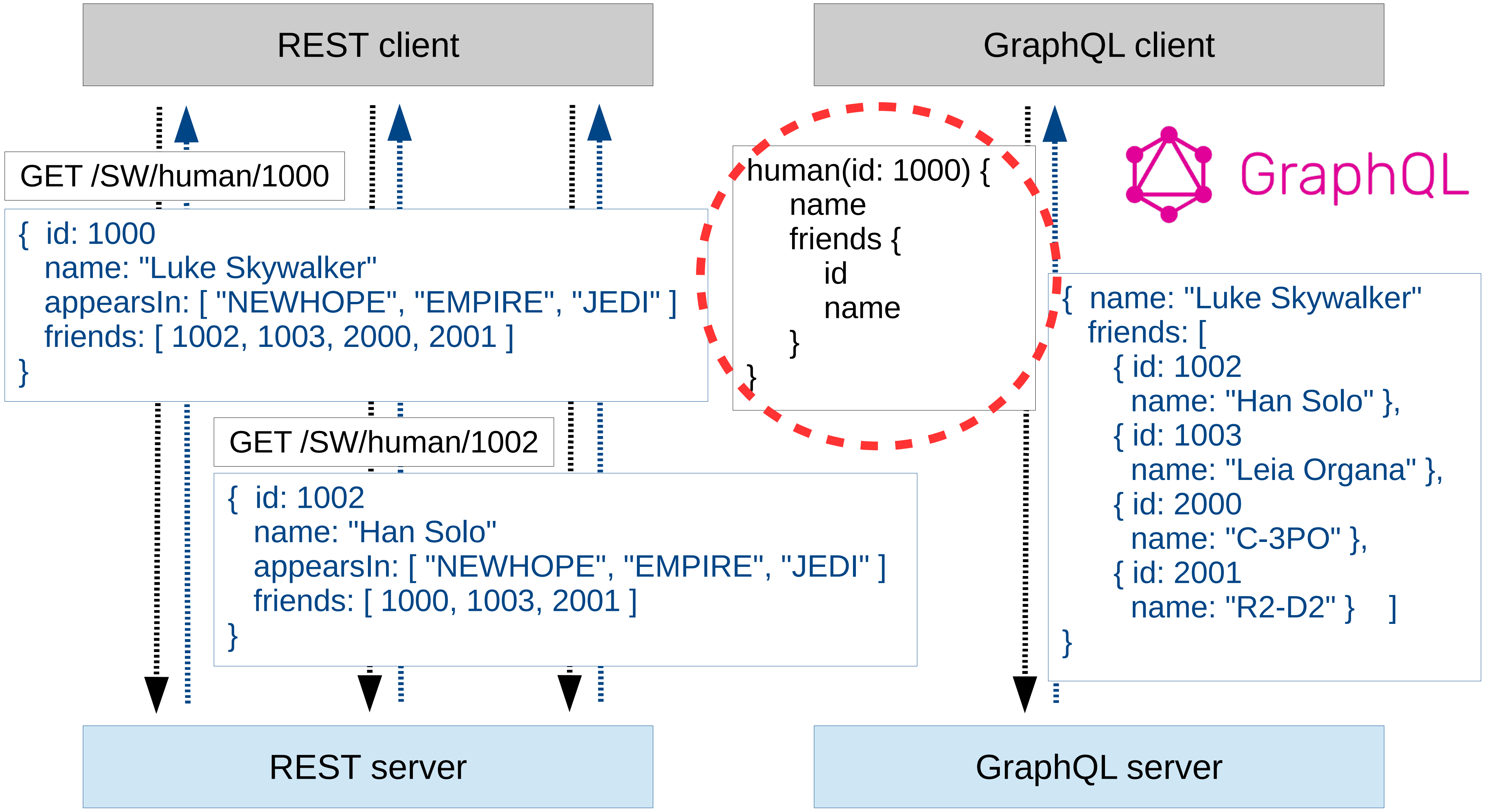
Olaf Hartig[a], Ruben Taelman[b]

(a) Dept. of Computer and Information Science, Linköping University, Sweden

(b) Ghent University – imec – IDLab, Belgium

LINKÖPING UNIVERSITY

GraphQL Example (in Comparison to REST)



```
GET /SW/human/1000
```

```
{  id: 1000
   name: "Luke Skywalker"
   appearsIn: [ "NEWHOPE", "EMPIRE", "JEDI" ]
   friends: [ 1002, 1003, 2000, 2001 ]
}
```

```
GET /SW/human/1002
```

```
{  id: 1002
   name: "Han Solo"
   appearsIn: [ "NEWHOPE", "EMPIRE", "JEDI" ]
   friends: [ 1000, 1003, 2001 ]
}
```

```
human(id: 1000) {
    name
    friends {
        id
        name
    }
}
```

```
{  name: "Luke Skywalker"
   friends: [
     { id: 1002
       name: "Han Solo" },
     { id: 1003
       name: "Leia Organa" },
     { id: 2000
       name: "C-3PO" },
     { id: 2001
       name: "R2-D2" }    ]
}
```

Example Query

```
{
  hero(episode: EMPIRE) {
    name
    friends {
      name
      appearsIn
    }
  }
}
```

“argument”

“fields”

GraphQL Schema

declaration of an object type with its fields and their types

(built-in) scalar type

declaration of an interface type and an implementation

```
type Starship {
  id: ID!
  name: String!
  length(unit: String): Float
}

interface Character {
  id: ID!
  name: String!
  friends: [Character]
  appearsIn: [Episode]!
}

type Droid implements Character {
  id: ID!
  name: String!
  friends: [Character]
  appearsIn: [Episode]!
  primaryFunction: String
}
```

argument

```
type Human implements Character {
  id: ID!
  name: String!
  friends: [Character]
  appearsIn: [Episode]!
  starships: [Starship]
  totalCredits: Int
}

union SearchResult = Human | Droid | Starship

enum Episode { NEWHOPE, EMPIRE, JEDI }

type Query {
  hero(episode: Episode!): Character
  droid(id: ID!): Droid
  node(id: ID!): SearchResult
}
```

declaration of a union type

declaration of the query type (possible root fields of queries)

Implementation Type-Specific Fields

```
{
  hero(episode: EMPIRE) {
    name
    friends {
      name
      appearsIn
    }
    ~~totalCredits~~
  }
}
```

```
type Human implements Character {
  id: ID!
  name: String!
  friends: [Character]
  appearsIn: [Episode]!
  starships: [Starship]
  totalCredits: Int
}

union SearchResult = Human | Droid | Starship

enum Episode { NEWHOPE, EMPIRE, JEDI }

type Query {
  hero(episode: Episode!): Character
  droid(id: ID!): Droid
  node(id: ID!): SearchResult
}
```

Implementation Type-Specific Fields

```
{
  hero(episode: EMPIRE) {
    name
    friends {
      name
      appearsIn
    }
    ... on Human {
      totalCredits
    }
  }
}
```

"inline fragment"

```
type Human implements Character {
  id: ID!
  name: String!
  friends: [Character]
  appearsIn: [Episode]!
  starships: [Starship]
  totalCredits: Int
}

union SearchResult = Human | Droid | Starship

enum Episode { NEWHOPE, EMPIRE, JEDI }

type Query {
  hero(episode: Episode!): Character
  droid(id: ID!): Droid
  node(id: ID!): SearchResult
}
```

Example Query with *Inline Fragments*

```
{
 hero(episode: EMPIRE) {
   name
   friends {
     name
     ... on Droid {
       primaryFunction
     }
   }
 }
}
```

Result:

```
{
 hero {
  name: "Luke Skywalker"
  friends: [
    {name: "Han Solo"}
    {name: "Leia Organa"}
    {name: "C-3PO"
     primaryFunction:"Protocol"}
    {name: "R2-D2"
     primaryFunction:"Astromech"}
  ]
 }
}
```

Example Query with *Aliases*

```
{
 hero(episode: EMPIRE) {
  heroname: name
  ... on Human {
   starships {
    name
    feet: length(unit: FOOT)
    meters: length(unit: METER)
   }
  }
 }
}
```

“alias”

Result:

```
{
 hero {
  heroname: "Luke Skywalker"
  starships: [
   { name: "X-wing"
     feet: 41.0
     meters: 12.5 }
   { name: "Imperial shuttle"
     feet: 65.6
     meters: 20.0 }
  ]
 }
}
```

Query with Name

```
query myExampleQuery {
  hero(episode: EMPIRE) {
    name
    friends {
      name
      appearsIn
    }
  }
}
```

Variables

```
query myExampleQuery($ep: Episode) {
  hero(episode: $ep) {
    name
    friends {
      name
      appearsIn
    }
  }
}
```

passed in a separate, transport-specific variables dictionary (usually JSON)

```
{
  "ep": "EMPIRE"
}
```

Default Values

```
query myExampleQuery($ep: Episode = EMPIRE) {
  hero(episode: $ep) {
    name
    friends {
      name
      appearsIn
    }
  }
}
```

Meta Fields

```
{
 hero(episode: EMPIRE) {
   name
   friends {
     name
     __typename
   }
 }
}
```

Result:

```
{
 hero {
  name: "Luke Skywalker"
  friends: [
   { name: "Han Solo"
     __typename: "Human" }
   { name: "Leia Organa"
     __typename: "Human" }
   { name: "C-3PO"
     __typename: "Droid" }
   { name: "R2-D2"
     __typename: "Droid" }
  ]
 }
}
```

Introspection

```
{
  __schema {
    types {
      name
    }
  }

  __type(episode: "Droid") {
    name
    fields { name }
  }
}
```

Definition of Mutation Operations

```
{
  type Mutation {
    createRatingForEpisode(ep: Episode!,
                           rating: Int!): Rat
  }

  type Rat {
    ep: Episode
    rating: Int
  }
}
```

Using Mutations

```
{
  type Mutation {
    createRatingForEpisode(ep: Episode!,
                           rating: Int!): Rat
  }

  type Rat {
    ep: Episode
    rating: Int
  }
}
```

```
mutation {
  createRatingForEpisode(
     ep: HERO,
     rating: 2
  ){
    rating
  }
}
```

Input Types

```
{
 type Mutation {
   addReviewForEpisode(ep: Episode!,
                       review: ReviewInput!):
                                      Review
 }
 input ReviewInput {
   stars: Int!
   comment: String
 }
 type Review {
   stars: Int!
   comment: String
 }
}
```

Input Types

```
{
 type Mutation {
   addReviewForEpisode(ep: Episode!,
                       review: ReviewInput!):
                                  Review
 }
 input ReviewInput {
   stars: Int!
   comment: String
 }
 type Review {
   stars: Int!
   comment: String
 }
}
```

```
mutation {
  addReviewForEpisode(
     ep: HERO,
     review: { stars: 2 }
  ){
    stars
    comment
  }
}
```

www.liu.se

LINKÖPING
UNIVERSITY