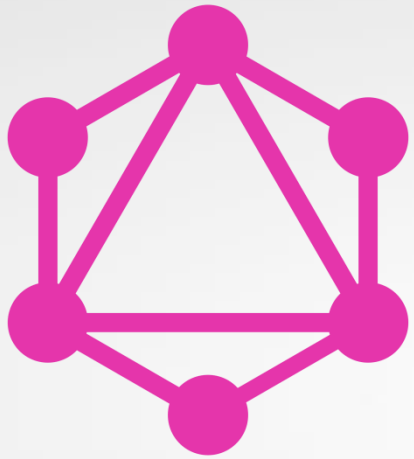
esprit
Se former autrement
HONORIS UNITED UNIVERSITIES



GraphQL

Module SOA

A.U 2024-2025

United Nations Educational, Scientific and Cultural Organization
UNESCO Chair "Project-based learning" ESPRIT School of engineering, Tunisia
uniTwin
EUR-ACE® Délivrée par la Commission des Titres d'Ingénieur
CONFÉRENCE DES GRANDES ÉCOLES
cdio CONCEIVE DESIGN IMPLEMENT OPERATE

Objectifs

- Comprendre le besoin ainsi que le principe du GraphQL.
- Lire et récupérer des données en utilisant des requêtes GraphQL.
- Ecrire et publier des valeurs avec le principe de mutation.
- Créer des fonctions chargées de renseigner les données avec le principe du Resolver.

Plan

- Introduction
- Présentation de GraphQL
- GraphQL vs REST
- Principes de GraphQL
- Requetes, mutations et Resolver en GraphQL

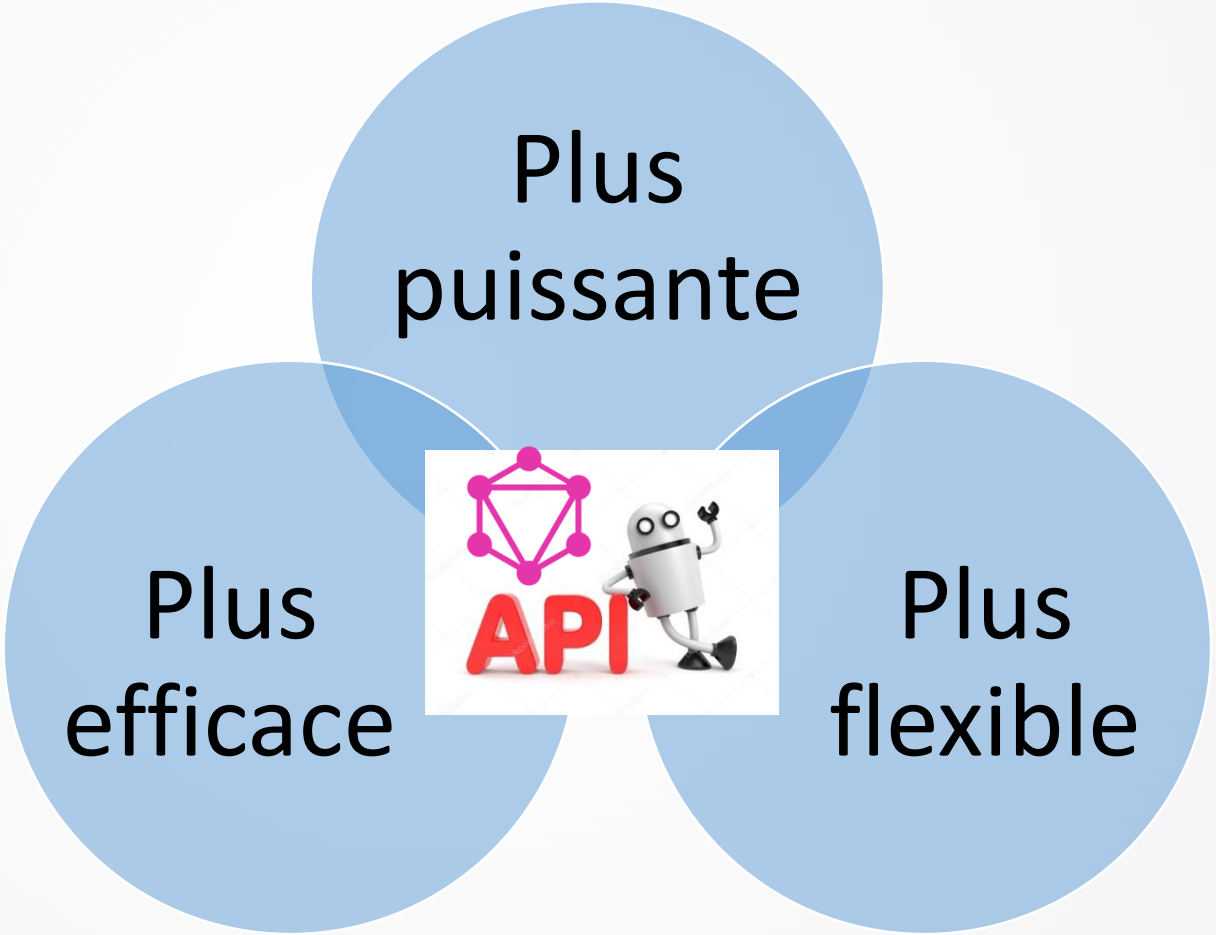
Introduction

- Utilisation accrue des mobiles:
  - Besoin de chargement de données efficace en minimisant la quantité de données à transférer sur le réseau.
- Données exposées par le serveur doivent souvent être modifiée pour tenir compte des exigences côté client:
  - La structure de la réponse du serveur doit être fixée par le client.

Introduction

- Variété de différents frameworks et plateformes front-end :
  - Nécessite la création d'une nouvelle API qui répondrait aux exigences de tous.

Solution



Présentation de GraphQL

- **GraphQL** est l'acronyme de **Graph Q**uery **L**anguage

- **Un langage de requêtes** flexible et environnement d'exécution pour développer des APIs

- Démarré chez Facebook en 2012, est devenu OpenSource en 2015

Présentation de GraphQL

- Représente la nouvelle façon de concevoir des API modernes:
  - fonctionnant sur un seul point de terminaison via HTTP,
  - optimisant les performances et la flexibilité.

- Les types de données renvoyées par l'API sont spécifiés via un *schéma* définissant **un graphe de types de données**.

Motivation pour GraphQL

- GraphQL est une alternative à REST

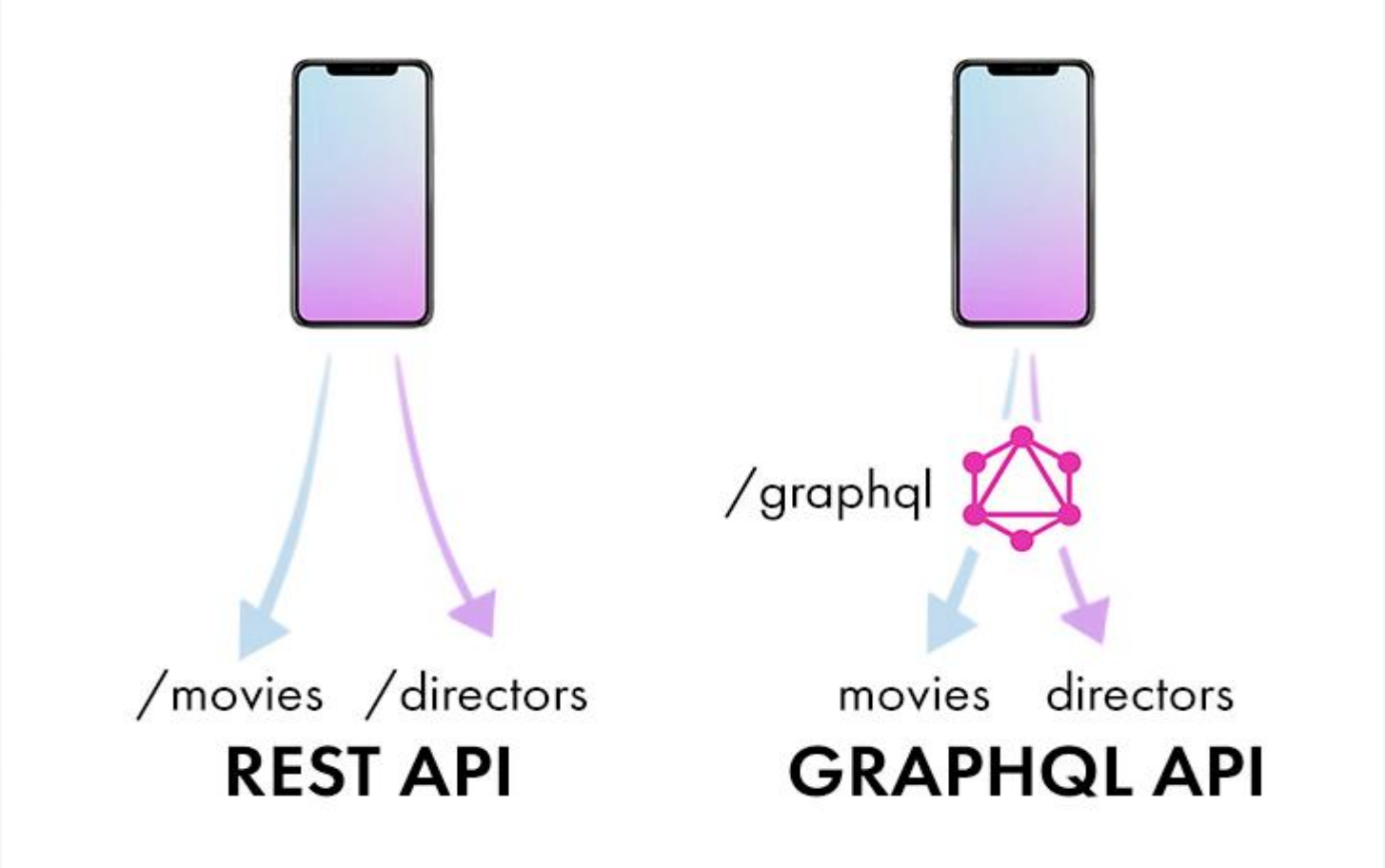
Source: http://www.google.com/trends/explore?hl=fr#q=rest%20api%2Csoap%20api&cmpt=q

Pourquoi GraphQL ?

- Palier aux problèmes qu'on peut rencontrer avec REST:
  - La sur-extraction (téléchargement de données inutiles):
    - Le développeur peut **demander exactement ce qu'il attend, ni plus ni moins;**
  - La sous-extraction (l'envoi de plusieurs requêtes):
    - On peut obtenir toutes les informations nécessaires **en une seule requête** contrairement à REST qui nécessite plusieurs appels (plusieurs URIs).
- Permet au client d'être sélectif sur ce qui est renvoyé, et de décider quelles données sont réellement nécessaires.

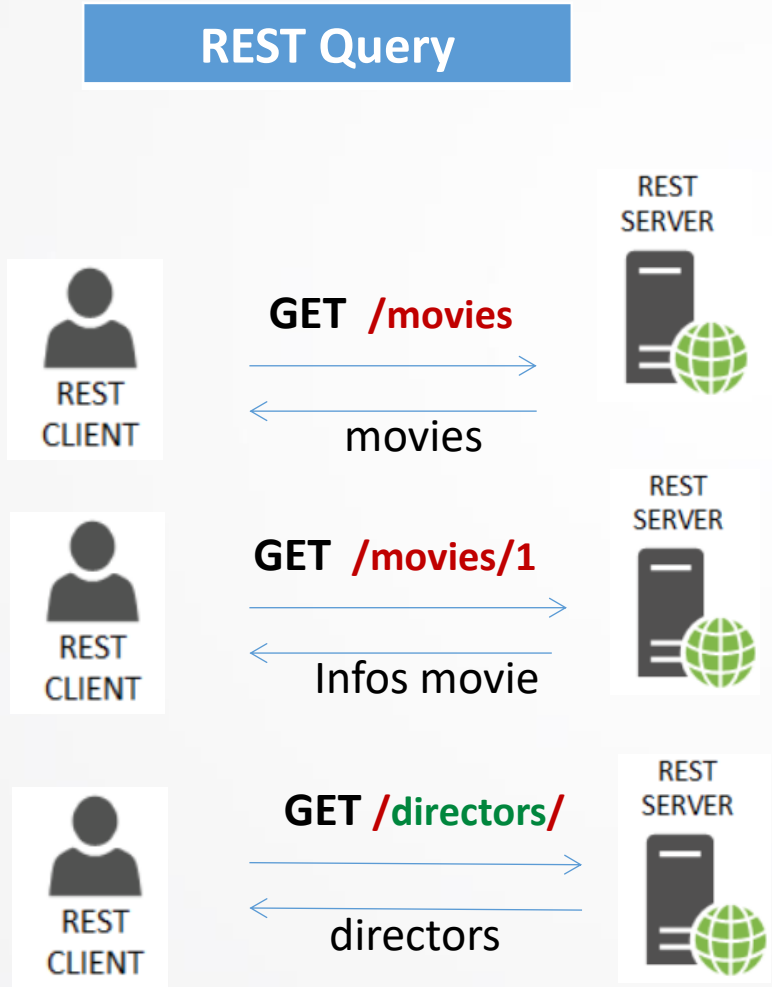
REST vs GraphQL

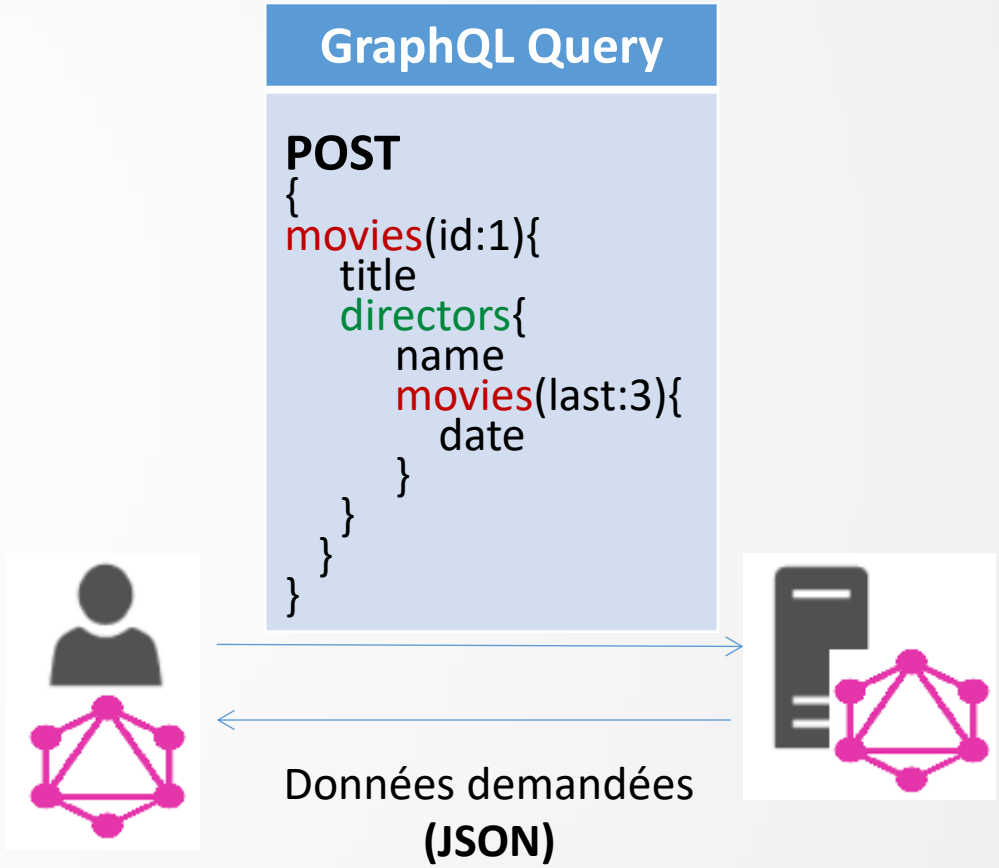
- **API endpoints** (points de terminaison)



REST vs GraphQL

- **API endpoints** (points de terminaison)





GraphQL

REST vs GraphQL

- **Méthodes**

REST	GraphQL
✓**POST**: Create	✓**Query**: Read
✓**PUT**: Update	✓**Mutation**: Create Update and Delete
✓**GET**: Read	✓**Subscription**: pour des fonctionnalités en temps réel
✓**Delete**: Delete	⇨ Toutes les opérations passent dans le corps de la requête HTTP via la méthode ***POST***
✓etc	

REST vs GraphQL

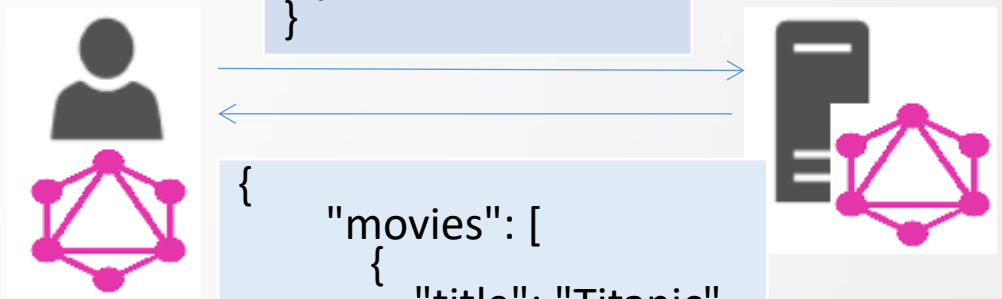
- **La sur-extraction** (téléchargement de données inutiles)

REST Query

(-) Tous les films avec tous les champs, même les données inutiles dans notre traitement, seront retournés.

GraphQL Query

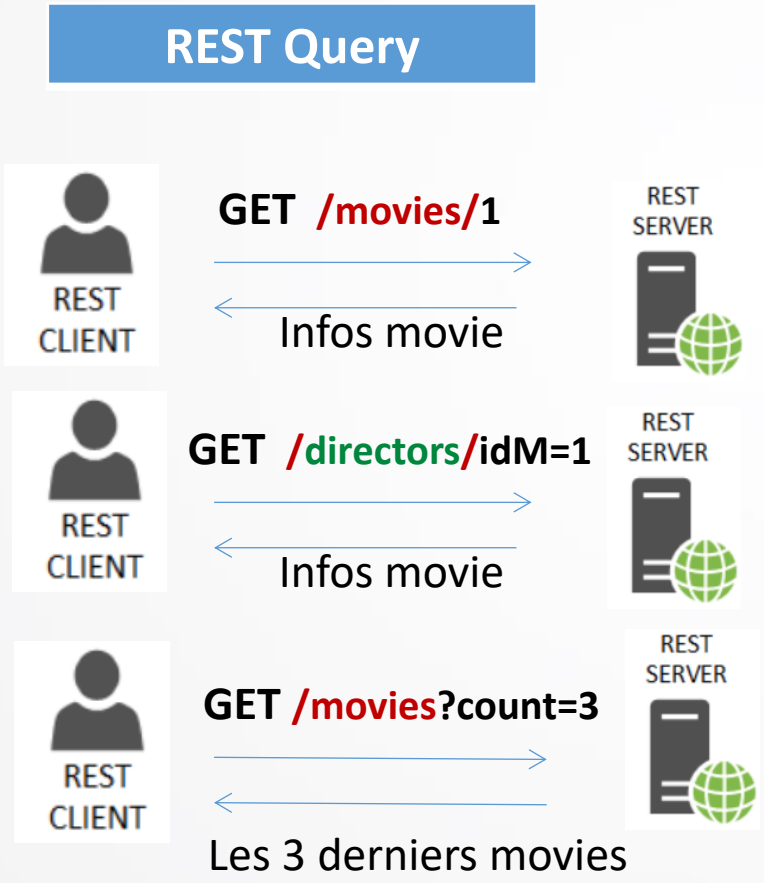
```
POST
{

movies(last:2){
    title
  }
}
```

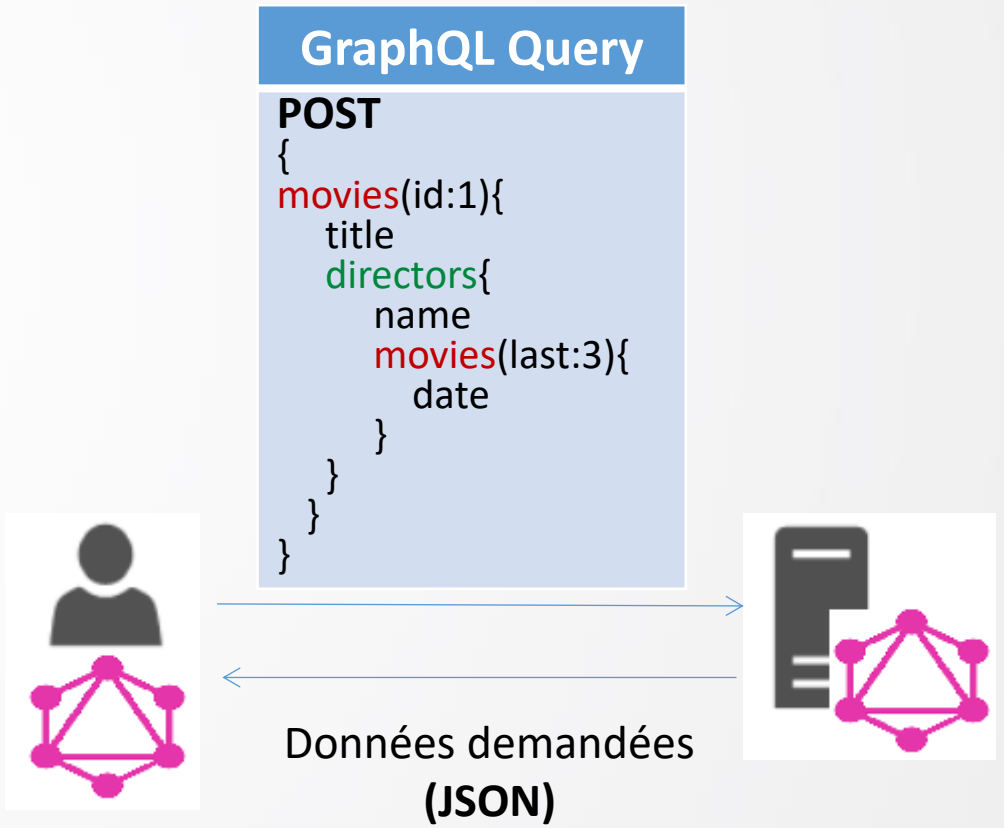


```
{
      "movies": [
        {
          "title": "Titanic"
        },
        {
          "title": "X-Man"
        }
      ]
}
```

REST vs GraphQL

- **La sous-extraction** (l'envoi de plusieurs requêtes)





REST vs GraphQL

- **Autres Critères:**

	REST	GraphQL
Structure de réponse	Fixé par le serveur	Fixé par le client
Format de données	JSON, XML, TXT, HTML	JSON
P. Transport	HTTP	HTTP
P. Communication	HTTP	HTTP
Méthodes HTTP	GET, POST, PUT, DELETE	POST
Temps de réponse	Lent	Rapide

Définition de données en GraphQL

SDL (Schema Definition Language)

- GraphQL a son propre langage de définition:
  - des données, leurs types ainsi que les opérations à exécuter (Requêtes, mutations ou Subscription).
- La syntaxe pour écrire des schémas est appelée Schema Definition Language (SDL).

Schema en GraphQL

- Elément central dans la conception d'une API GraphQL.
- On y définit:
  - Les objets GraphQL: leurs champs et types
  - Toutes les requêtes disponibles: **Query**
  - Toutes les actions disponibles: **Mutations**
  - Les mises à jour de données en temps-réel: **Subscriptions**
  - Règles métiers (exp: ! si champs obligatoire)

Schema en GraphQL

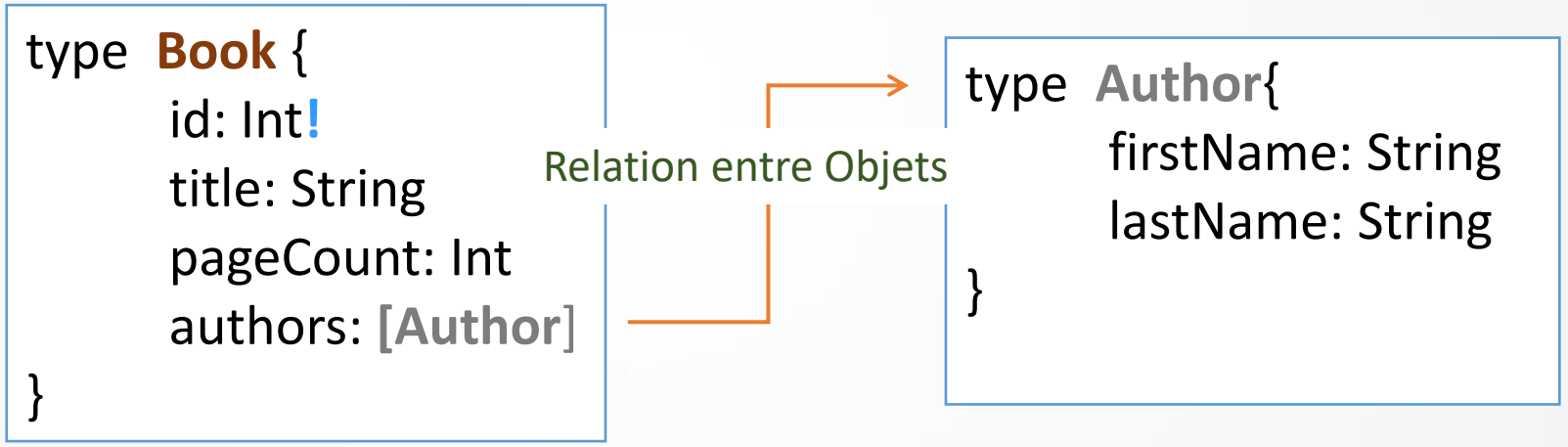
Type

- L'élément le plus basique de chaque schéma

- Permet de:
  - Etablir des relations entre différents éléments de schéma,
  - Définir les opérations GraphQL autorisées à effectuer sur le serveur.

Schema en GraphQL

Objet



! : exprime un champs optionnel

Schema en GraphQL

- Methodes

- **requête**: correspond à l'action read, c'est-à-dire « GET » en REST.

- **mutations**. Toutes les autres actions (create, update et delete), c'est-à-dire « POST », « PUT » et « DELETE » en REST.

Query (Read)

- Definition

```
type  Query{
        book(id: Int!): Book
        books : [Book]
        allAuthors(count: Int): [Author]
}
```

Requête/Query

- Test

```
{
  books{
        title
   }
}
```

```
{
  books{
        title
        author{
            firstName
        }
   }
}
```

```
{
   "data": {
      "books": [
         { "title": "l'oeil et le mot"  },
         { "title": "En route pour les étoiles" }
      ]
   }
}
```

```
{   "data": {
      "books": [
         {   "title": "l'oeil et le mot",
            "author": [
               {   "firstName":   "Jean-François"  }
            ]
         },
         {"title": "En route pour les étoiles" ,
           "author": [
               {   "firstName":   "Marlène"  }

            ]
         }
      ]   } }
```

Mutation

- Definition

```
type  Mutation{
 createBook (id: Int, title: String, pageCount:  int): Book
 deleteBook (id: Int!): [Book]
 addAuthor (firstName: String!, lastName!): [Author]
}
```

Mutation

- Test

```
mutation{
   createBook( id: "123",
                  title: "l'oeil et le mot ",
                   pageCount: 240)   {
                      id
                       title
                    }
   }
```

```
{
   "data": {
      "createBook":  {
          "id": "123" ,
          "title": " l'oeil et le mot "
       }
    }
}
```

Mutation

- Test

```
mutation{
  deletBook( id: "123")
                {
                   id
                    title
                }
  }
```

```
{
   "data": {
     "deleteBook":  []
    }
}
```

Exemple Schema en GraphQL

Objets

```
type  Book {
        id: Int!
        title: String
        pageCount: Int
        author: Author
}
```

```
type  Author{
        firstName: String
        lastName: String
}
```

Requête

```
type Query {
        book(id: Int!): Book
        books : [Book]
}
```

Mutation

```
type Mutation {
        createBook(title: String, pageCount:  Int,
                        authors: [Author]): [Book]
}
```

Resolvers

- Liaison avec le métier

- Un résolveur est assigné à chaque champs.
  - Fonction qui renvoie des données pour les champs du schéma.

- A le rôle de chercher et récupérer ta data de n'importe quelle source de données: Base de données, fichiers, etc.

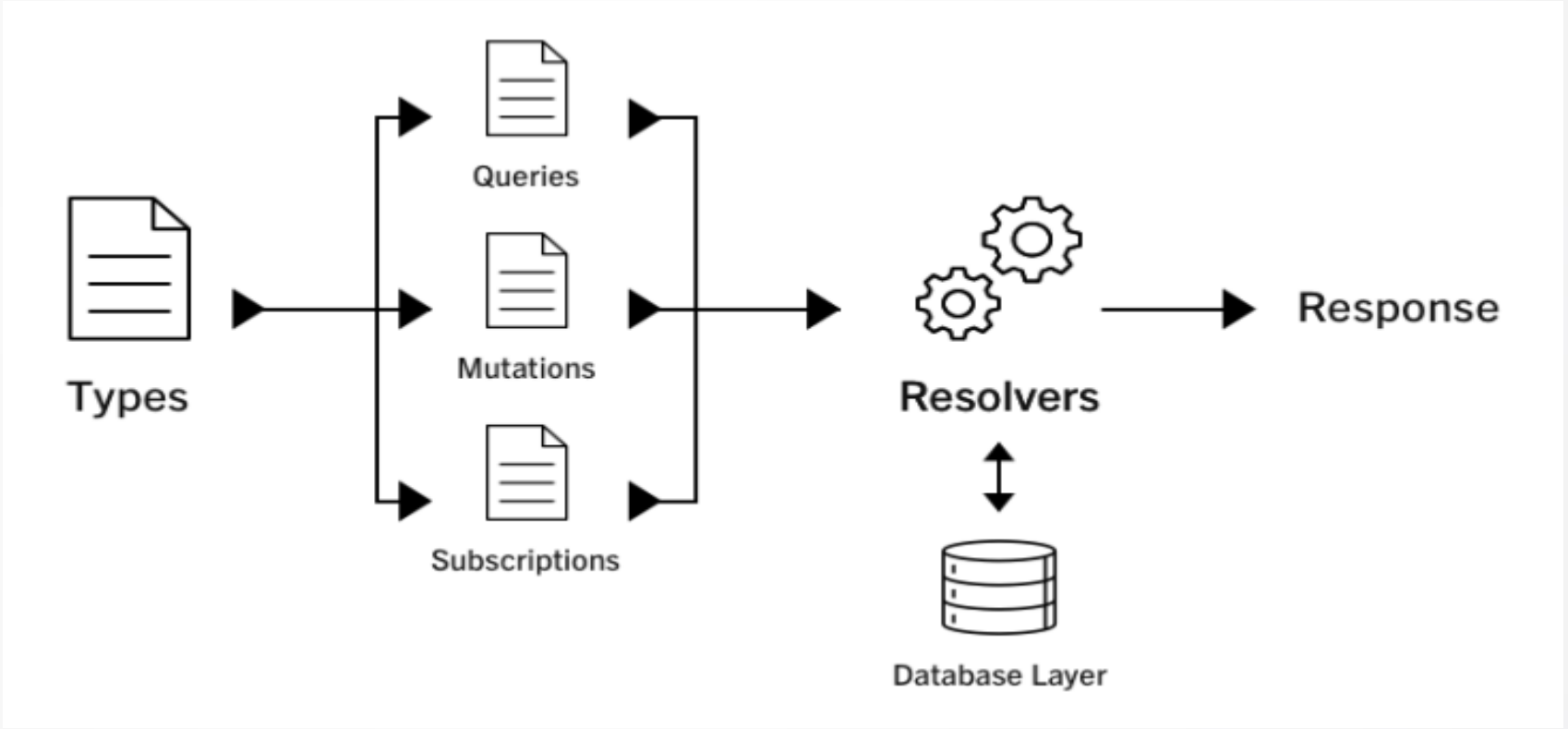
Subscription

- Données en temps réel

- Mécanisme pour les mises à jour de données en temps réel,

- Apporter des fonctionnalités en temps réel dans votre application GraphQL.

En résumé


Queries
Mutations
Subscriptions
Types
Resolvers
Response
Database Layer