GraphQL

Need →

GraphQL was invented at Facebook because the company needed to solve many technical issues with its mobile application.

1. excels at network latency in low-power devices.
2. decouples the FE and BE more

Why →

If you **analyze any data model**, big or small, **you'll always find it to be a graph of objects** with many relations between them

Developers of data API consumers(FE) can now think and implement their way of what they want would be the best way to extracting the data.

What →

- GraphQL is a **query language** for data APIs (**perspective of the frontend**)
- GraphQL is also a **runtime layer** that needs to be implemented **on the backend**
- BE developers get to decide the **schema** of the data.
- **declarative, flexible, and efficient**

How →

- The client sends that text request to the GraphQL API service
- The GraphQL runtime layer accepts the text request

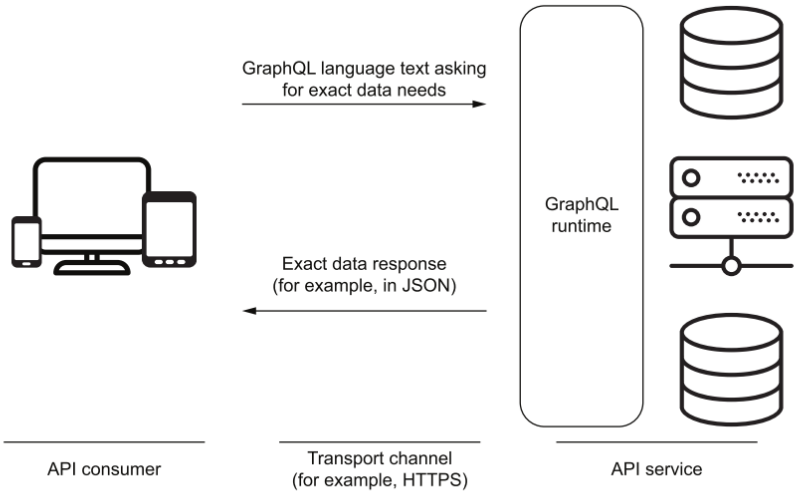


Figure 1.1 GraphQL is a language and a runtime.

communication → GraphQL clients communicate with GraphQL agents through JSON

GraphQL Query →

```
{
  employee: (id: 42) => {
      name,
      email,
      birthDate,
      hireDate,
  };
}
```

How Query works ? → The query is sent via some TCP based protocol (ex: HTTP) by GraphQL client to a GraphQL server that can understand and translate it into something the data-storage-engine(ex: SQL) can understand.
Then, the GraphQL server can take what the storage engine responds with, translate it into something like JSON or XML, and send it back to the client application.

Type System → At the core of GraphQL is a strong type system that is used to describe data and organize APIs.

Types ensure that clients ask for only what is possible and provide clear and helpful errors.

Specification → By facebook
https://spec.graphql.org/

Query vs Mutation → you use queries; and when you need to modify data, you use mutations.

Query vs Mutation

Queries represent **READ** operations. **Mutations** represent **WRITE-then-READ** operations.

Mutations usually **trigger events for subscriptions.**

Subscription

used for **real-time data monitoring** requests**.**
That's usually done with **WebSockets for web applications**

GraphQL service

split into two major parts

- **structure**
- **behaviour**

Structure

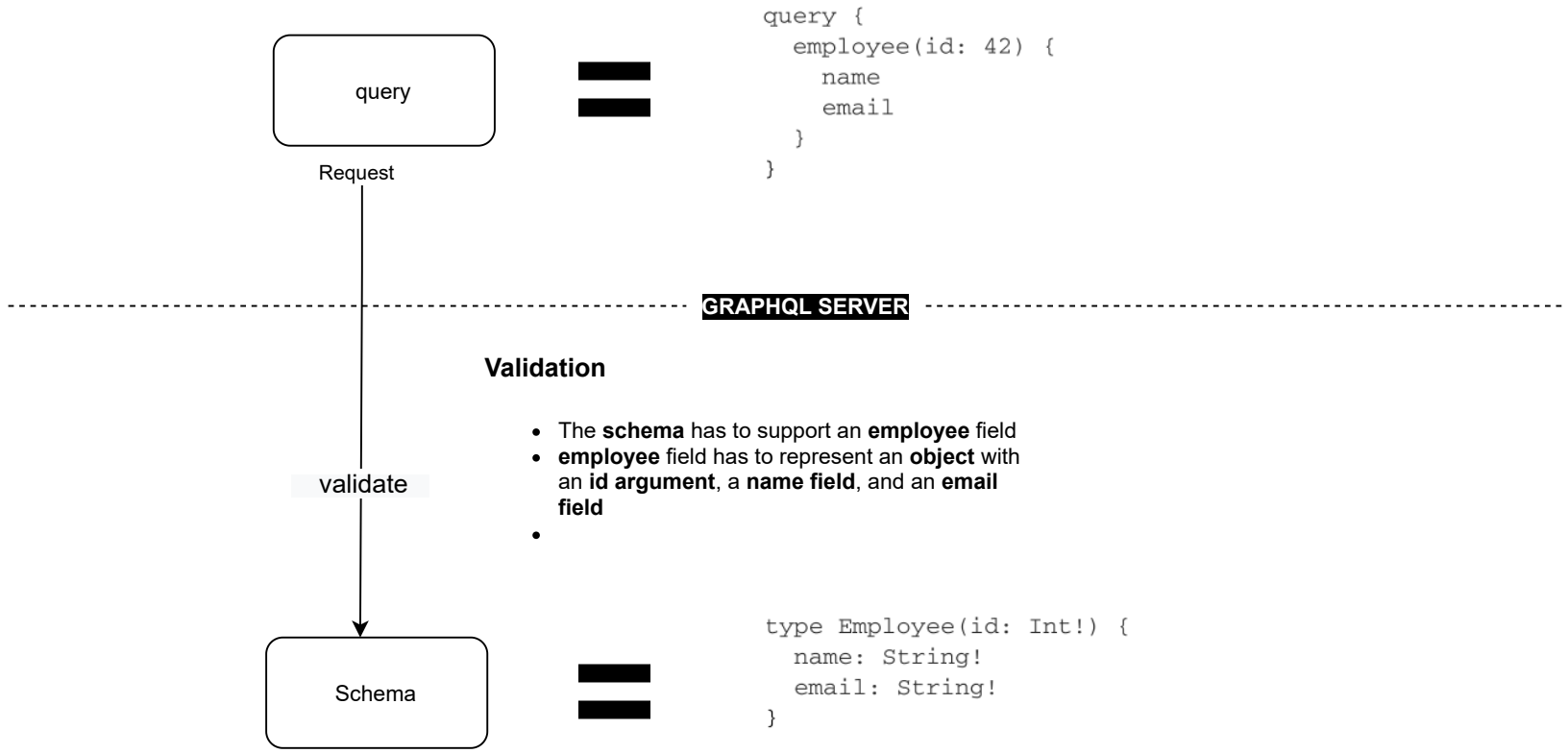
- The **structure** is defined with **a strongly typed schema**
- The **schema** is basically **a graph of fields**
- These **fields have Types**
- this **schema** is like a **catalog** of all the **operations** a **GraphQL API** can handle

Behaviour

- The **behavior** is naturally implemented with **resolver functions**
- **schema fields** are simply **resolver functions**
- **A resolver function** represents **the instructions** on how and where **to access raw data**.

RPC

- GraphQL is essentially a way for clients to invoke **remote —resolver—functions**.
- So it is often compared with **remote-procedure-calls** in **distributed** computing concept



Validation

- The **schema** has to support an **employee** field
- **employee** field has to represent an **object** with an **id argument**, a **name field**, and an **email field**
-