WRITING QUERIES IN GraphQL

GraphQL in Action

2021-11-17 wangwei17(@baidu.com)

如果有一种技术能够让我无比激动，
让我忍不住要与大家分享，
那么，这就是，
GraphQL！

目录

1. 什么是 GraphQL

- GraphQL 和 GraphDB 的关系就像 JavaScript 和 Java 的关系一样。
- 仅仅从消费者角度看，GraphQL 是一种 Data APIs 的 *Query Language*。
- GraphQL 是把数据看作图结构的一种 API 技术。
- GraphQL 不仅和 GraphDB 没有什么关联，而且和 *QL* 也没有太大关联。

2. 为什么要创造 GraphQL

- 2011 年，Facebook 注意到移动化浪潮的到来，而当时，Facebook 却还没有移动 App。
- 随着时间的推移，Facebook 发现，浏览器能够提供的能力无法满足他们的功能需求，例如现在看起来很普通的信息流的下拉刷新功能
- 更重要的是，Facebook 慢慢变成了一个不仅限于图片和评论的平台，而是一个面向数十亿用户的、新型的、社交平台。
- 随着这种平台战略地位的转变，对技术提出了很大的挑战。
- 浏览器应用中使用的 REST API 对于移动 App 而言，欠缺灵活性（**想想我们自己的产品**）。
- 为解决这个巨大的挑战，他们想开发可以在本地处理数据并为用户提供更好的移动体验的 API。
- 2012年，Facebook 的一名工程师提出了一个想法：更加面向端而非服务端的 API。
- 这个想法和另外两名员工擦出了火花，于是他们一起开始了一些革命性的事情——GraphQL。
- 一周后，他们将这套思想用在了 Facebook App 上，然后在2015年，他们开源了 GraphQL 标准。

From https://smartbear.com/blog/graphql-what-it-does

3.1 GraphQL 的变革——改变了对数据的基本认识

对数据的认识从资源转变为图。

```
GET /tracks/ID
```

```
GET /playlists/ID
```

```
GET /playlists/ID/tracks
```

REST

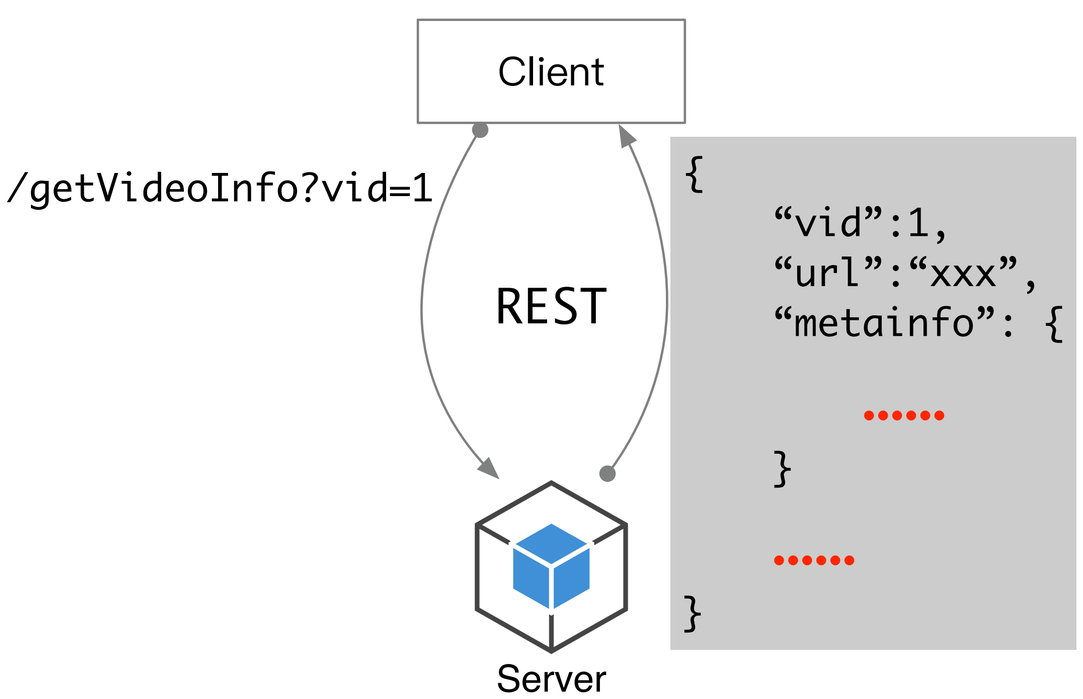

```
user --- OWNS --- playlist
  |                  |
LIKES              CONTAINS
  |                  |
  v                  |
track <--------------┘
```

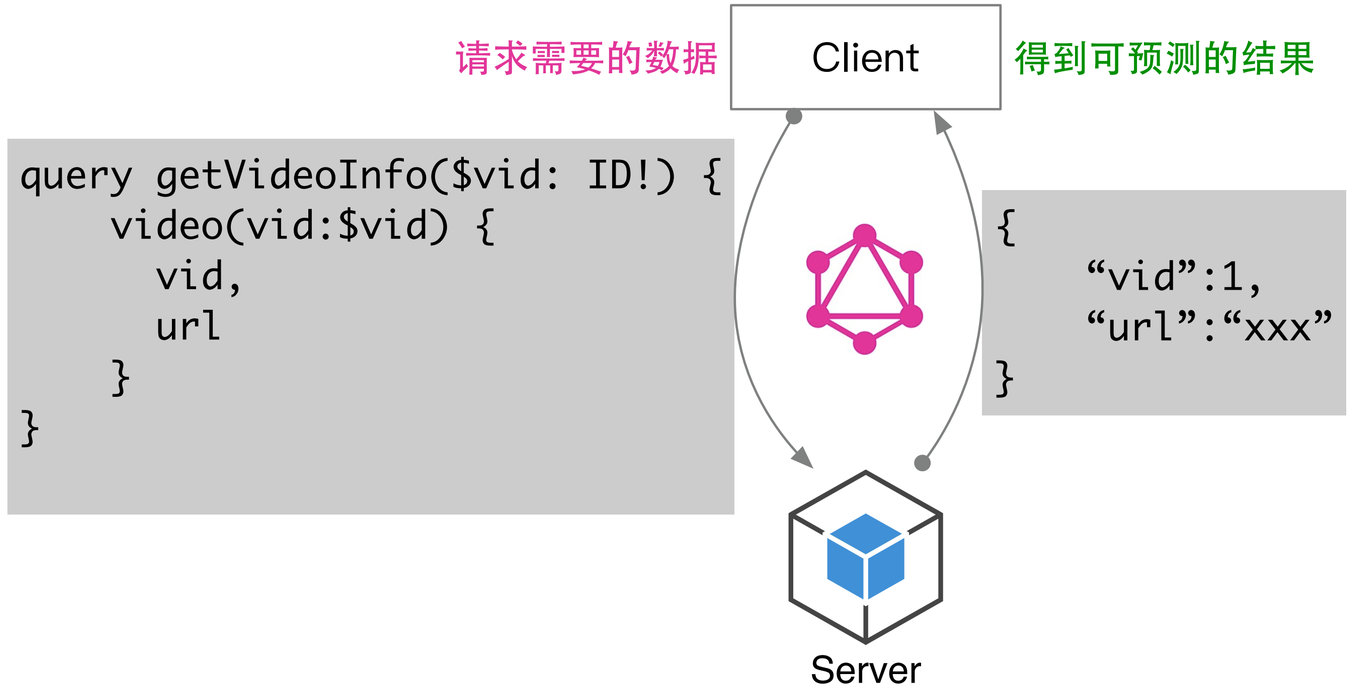
```
user
 └-OWNS-> playlist
          └-CONTAINS-> track
                       └-LIKED_BY-> users
```

GraphQL

3.2 GraphQL 的变革——改变了通信双方的话语权

- REST 模式下，返回数据的结构由 Server 来控制，Client 被动接受，没有选择权

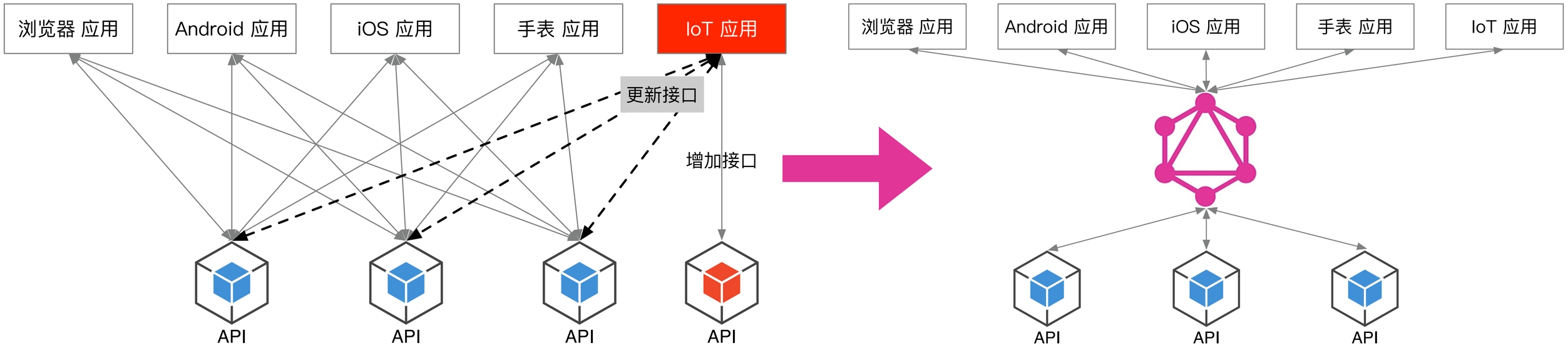




- GraphQL 改变了通信双方的地位，Client 可以根据自己的需要要求 Server 返回指定结构的数据。
- 没有哪种技术（gRPC除外）可以像 GraphQL 一样，在不改变 Server 代码的前提现，实现这个功能。
- 无论如何，这难道不是一件令人激动的事情吗?

3.3 GraphQL 的变革——改变了系统的通信拓扑

- 不同的应用类型需要提供不同的用户体验，因此，对应的后端 API 要么升级要么新增 API 来支持。



- 随着应用类型的丰富，GraphQL 改变了系统的通信拓扑，这对后端 API 的维护而言简直是福音。
- 没有什么是抽象一层解决不了的事情，如果有，就再抽象一层。

3.4 GraphQL 的变革——根植于框架的强类型系统

- 强类型系统好不好？强类型系统太严苛了，这会降低开发效率？弱类型系统就好吗？真的如此吗？

弱类型语言

强类型语言

以 JSON 为基础的 REST API 本质上就是一种弱类型的信息交换语言。

我一次又一次的见到：上下游接口字段类型不一致导致的线上问题。

- 越是大型的项目，越有必要使用强类型系统，因此此时沟通是影响效能的因素之一。我们以为的不一定就是我们以为的。
- 在 YAPI，JSON Schema等系统，类型系统和我们的框架并不是强相关的，并且一般都会落后代码。
- 在 GraphQL 中，类型系统是支撑整个框架的基本元素，类型系统和框架是强相关，和代码同步。

3.4 GraphQL 的变革——根植于框架的强类型系统

```
type Video {
    vid: ID!
    url: String!
    isVertical: Boolean
    videoMetaInfo: VideoMetainfo!
    tags: [VideoTagGroup!]
}
```

```
query getVideoInfo($vid: ID!) {
    video(vid:$vid) {
      vid,
      url
    }
}
```

```
query getVideoInfo($vid: ID!) {
    video(vid:$vid) {
      vid,
      url,
      uploadUser
    }
}
```

```
{
    “vid”:“1”,
    “url”:“xxx”
    “isVertical”: 1
}
```

3.5 GraphQL 的变革——基于强类型系统的文档

- 说到接口文档，我只能用两个字来形容：“呵呵”！！！

- 作为 Client，好不容易要做这个需求了，Server 告诉我说还没有文档，这是让我盲码吗?
- 作为 QA，测试的时候 RD 告诉我们没有接口文档，这是要那样？猜着测吗?

没有文档对于我们而言是最好的情况了

- Server 偷偷改了字段的类型，你倒是通知一声呀?
- 翻箱倒柜找了一个接口说明的 WIKI，按照 WIKI 调了半天了，你告诉我 WIKI 有点落后了……

3.5 GraphQL 的变革——基于强类型系统的文档

- 这种接口文档，眼熟不?

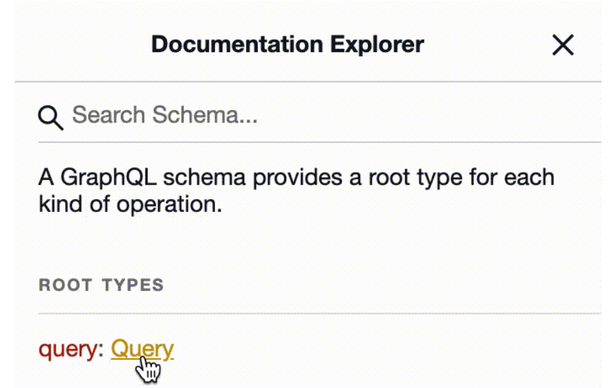

```
⊟{
    "info":⊟{
        "access_state":⊞Array[0],
        "need_vcode":0,
        "vcode_md5":"8751/zAYbDN8oZ4w12vXM4Iyg74gjJWi8mrejLCe2MbrdX++EsZ+b6XIrvFrsfN8",
        "vcode_prev_type":0,
        "vcode_type":0
    },
    "data":⊟{
        "strategy":⊟{
            "start_live":⊟{
                "switch":1,
                "text":"正常"
            },
            "user_verify":⊟{
                "switch":1,
                "type":1,
                "text":"正常"
            }
        },
        "user_info":⊞Object{...},
        "live_info":⊞Object{...},
        "user_status":0,
        "sdk":⊞Object{...},
        "user_watermark":"百度直播",
        "is_show_redpacket":1
    },
    "error":⊞Object{...},
    "server_time":1439023,
    "time":1552639172,
    "ctime":0,
    "logid":2372092114,
    "error_code":"0"
}
```


我们总想着根据左边的接口响应来自动生成接口定义。

谁能按照libx264的实现把H264协议给还原出来?

GraphQL 的文档和基于 注解 的文档还不一样。

利用 GraphQL，代码写完之后，接口文档自动就有了。

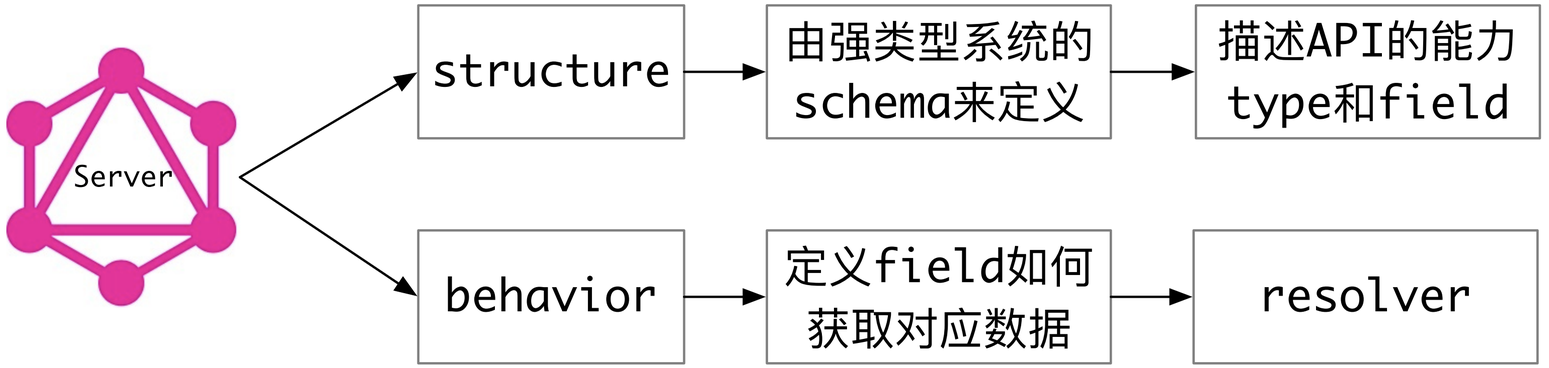


GraphQL 强大的文档功能，解决了大型系统各团队间的沟通成本，提升了团队协作的效率。

3.6 GraphQL —— 规则的改变者

- GraphQL 改变了前后端团队的交互方式、颠覆了前后端团队的通信方式，使得他们可以更顺畅而高效地协作，被视为一种革命性的新思路、新技术。
- 互联网改变了人类世界的交流方式一样，GraphQL 正在彻底改变 API 的交流方式。
- Communication 是一件非常难的事，无论是组织间的沟通还是软件间的通信，康威定律也说，组织设计的系统会受限于组织的通信结构。能够变革沟通/通信方式的技术，必然会开启一个新的世界，搅起一场风云。
- 根据 https://graphql.cn/users 的信息，目前已经有近100家企业在使用 GraphQL，并且据我了解，有使用了 GraphQL 技术，但是还未出现在该列表中的公司也很多，例如 Netflix, 爱奇艺，携程……

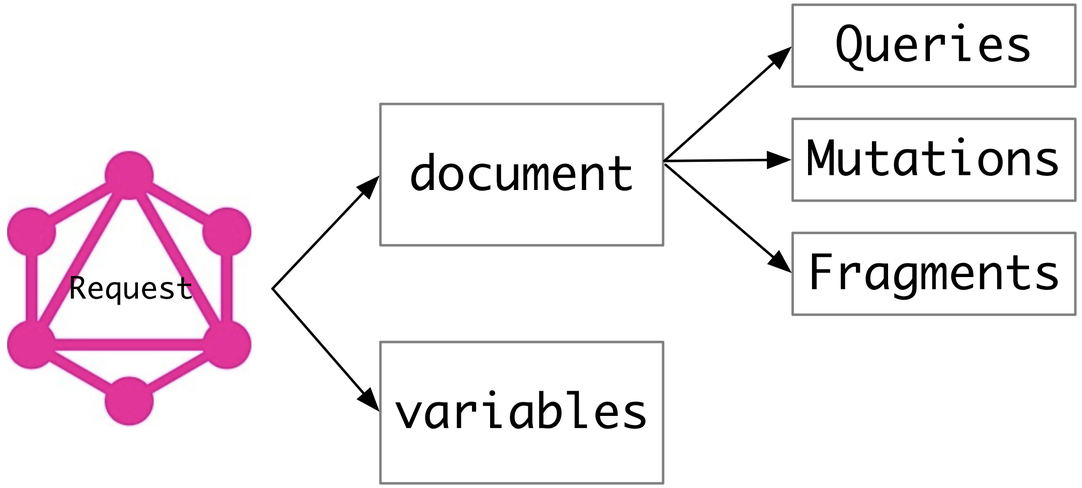
4 GraphQL中的基本概念——GraphQL Server




```python
class Mutation(ObjectType):
    """
    Mutation Type
    """
    uploadVideo = CreateVideo.CreateVideo.Field()


schema = Schema(query=Query, mutation=Mutation)
```

```python
class CreateVideo(Mutation):
    """
    上传Video
    """
    class Arguments:
        """
        Arguments
        """
        video_data = VideoInput(required=True)

    ok    = Boolean()
    video = Field(Video.Video)
```

4 GraphQL中的基本概念——GraphQL Request



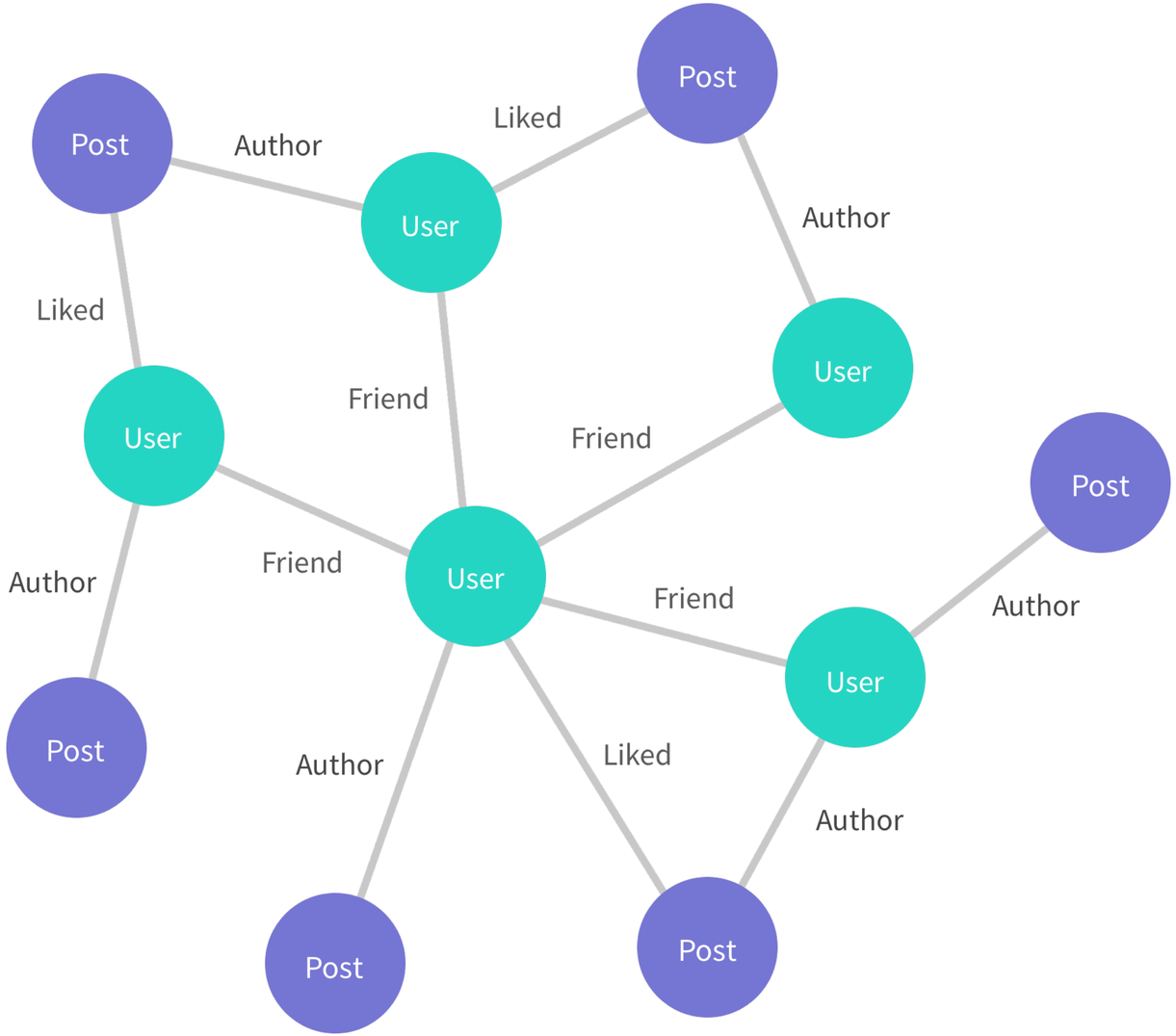

```
query getVideoInfo($vid: ID!) {
    video(vid:$vid) {
      ...video
    }
}

fragment video on Video {
  vid
  url
  videoMetainfo {
    codec
    fps
    duration
  }
}
```

QUERY VARIABLES REQUEST HEADERS

```
{"vid": 5}
```

读接口和写接口严格分离，避免读接口中的隐含写操作。

4 GraphQL中的基本概念——Node & Edge



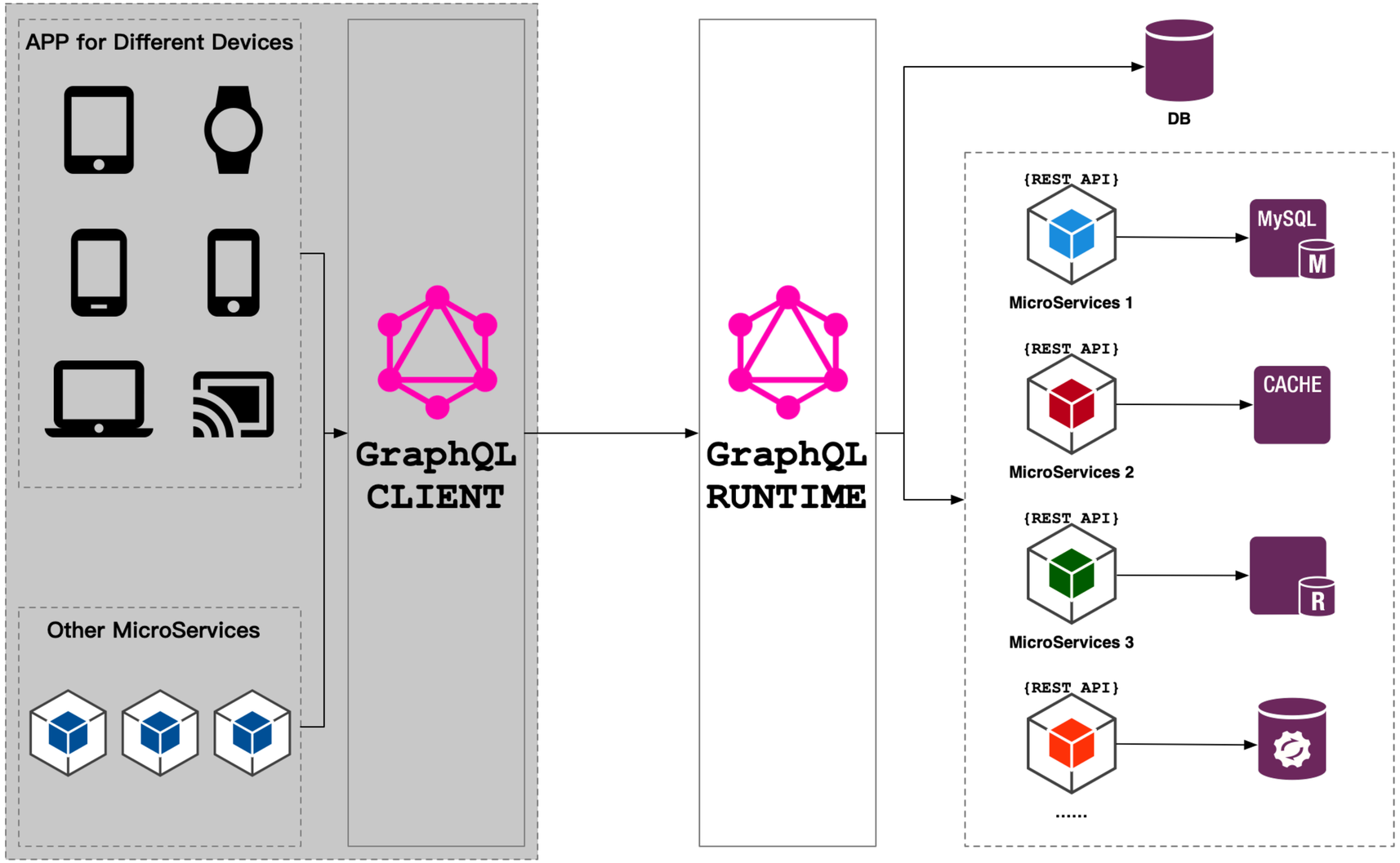
把数据看作图结构，以图理论来理解数据，而不是把数据存储于 GraphQL DB。

```
{
  user(id: "ZW5jaG9kZSBIZWxsb1dvcmxk") {
    id
    name
    friendsConnection(first: 3) {
      edges {
        cursor
        node {
          id
          name
        }
      }
    }
  }
}
```

From https://www.apollographql.com/blog/graphql/explaining-graphql-connections/

5 GraphQL中的整体框架

- 一种标准和规范
- 一种把数据视为图结构的查询语言
- 一种服务
- 一种框架
- 一种运行时



6 GraphQL生态发展

- 2019年，成立了 GraphQL Foundation, 后被 Linux Foundation 托管，以促进 GraphQL 的发展。
- GraphQL Foundation 的成员目前有：AWS，Facebook，IBM，Twitter，APOLLO，……
- 支持 20+ 主流语言，包括 C/C++，Golang，PHP，JAVA/Kotlin，Swift/Objective-C，JavaScript……
- 2021年7月，Spring 与 GraphQL Java 联手推出 Spring GraphQL，随着 Spring GraphQL 的推出，会有越来越多的开发者参与进来，让这个先进的理念得到落地的可能。
- 目前使用 GraphQL 的公司有：Facebook，Netflix，GitHub，PayPal，Twitter，……
- Apollo 为 GraphQL 的发展和落地应用提供了丰富的框架和工具支持。

如今，几年之后，情况已经完全变了~

7 是否要选择 GraphQL 呢

“Microservices Buy Your Options”

From ***Monolith to Microservices***

GraphQL Buy Your Options.

- 业务场景
- 团队文化
- 成本/收益
- 配套设施
- ……

Thanks

Why graphQL is The Future

bright coding