JD-chain穿透式数据检索系统（Argus）

1 产生背景

1.1 快速数据检索

区块链中区块高度的增长代表着区块链中的数据的增长，对数据的分析需求也相应产生，将所需要的数据精确和快速检索出来是数据分析的基础。

1.2 业务增强

区块链数据是严谨的业务数据，对业务数据的分析有利于业务的增强。穿透式检索能够为区块中包含的业务数据建立索引，因此可以快速准确的检索出所需要的业务数据，为业务分析提供支持

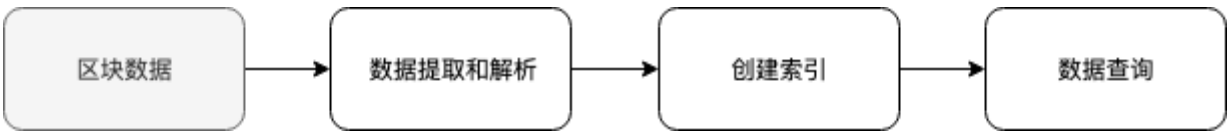
2 技术方案

2.1 数据流程

Argus按照以下思路实现穿透式检索：

1. 通过jd-chain提供的API提取出可以使用的数据，并根据jd-chain定义的数据结构，解析得到区块元信息数据
2. 通过定义Schema并根据定义解析业务数据，得到业务信息数据
2. 为解析得到的数据在图数据库中创建图关系索引，并提供对数据及关系进行查询的语言接口
3. 提供一个查询引擎，通过查询语言进行查询

数据流程如图所示：



2.2 数据的提取和解析

jd-chain提供了基础数据的查询接口，基于该接口可以获得需要索引的数据，之后按照得到的数据结构解析即可。

2.3 数据索引

图数据库能够直接的展现数据内在的关联，因此采用基于图数据库理论的Dgraph作为索引数据库。

通过将需要索引的数据转化为RDF格式数据，提交到Dgraph中进行索引，实现了数据的保存和索引。

2.4 数据查询

支持通过标准GraphQL来进行数据的查询。

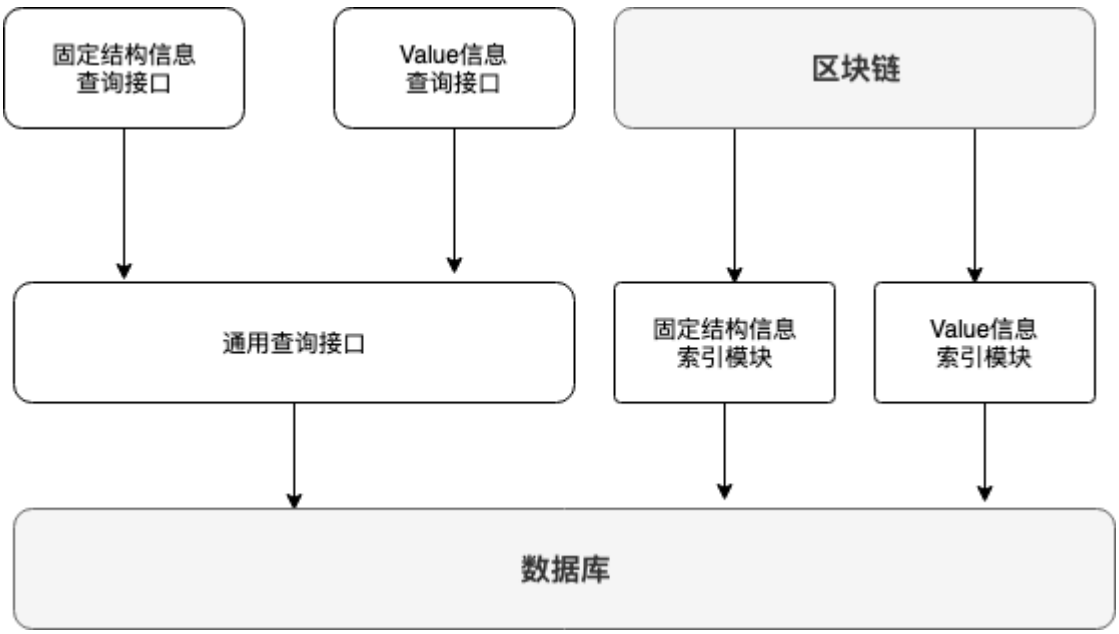
此外，为了降低熟悉SQL的用户的学习成本，Argus查询模块支持通过SQL进行简单地查询。

3 系统整体架构设计

从功能上，整个系统分为三大部分：

- 区块元数据索引模块，为区块元数据创建索引
- 区块业务数据索引模块，为区块中的业务数据创建索引
- 数据查询，提供对区块元数据和业务数据的检索API，以及扩展查询支持，包括对SQL的支持

如图所示：



4 Schema机制

4.1 什么是Schema

对数据的索引需要了解数据的模式，这里称为Schema。无论是固定结构信息还是Value信息，都遵循一定的Schema，只有在这个前提下，才能知道索引什么和如何索引。

4.2 固定结构信息Schema

区块固定数据可以认为是一种固化在jd-chain中的Schema，除非jdchain数据结构发生变化，否则不会发生变化，因此在本系统实现中直接固化在代码中。

4.3 Value信息Schema

业务数据是多变的，因此Value信息也是多变的，也因此Value信息的Schema需要通过外部输入。

根据外部输入的Schema的对Value进行解析和索引。

举例，如果区块中存在以下数据

```
{"id": 1463, "name": "culture clash"}
```

如果没有外部结构录入，系统无法自动识别出其中有意义的内容。

如果外部定义Schema如下：

```
type Company{
    id(isPrimaryKey: Boolean = true):                    Int
    name:                                                String
}
```

根据Schema的定义，即可以识别出其中的 **id** 和 **name** 是有意义的，即可以将其进行索引。

4.4 Schema的生命周期

Schema的生命周期可分为两个过程：

1. 创建索引
2. 移除索引

下面详细说明。

4.4.1 创建Schema

创建Schema是根据数据结构，创建一个与数据模式匹配的Schema。

Schema创建完成后，并不会立即开始索引数据。

4.4.2 开始索引

开始索引是指定Schema，并根据Schema读取数据创建可供查询的数据的操作。

启动索引后，系统会根据Schema定义开始创建索引数据，这是一个持续的过程；如果监测到数据有更新，会实时为新数据创建索引。

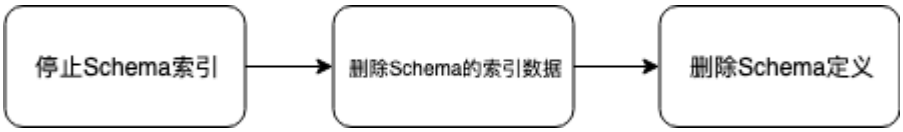
4.4.3 停止索引

停止索引是指定Schema停止创建索引数据的操作，停止索引后，之前创建的索引并不会被移除。

4.4.4 移除Schema

移除Schema是移除之前创建的Schema的操作。移除Schema会同时移除根据该Schema创建的索引。

移除索引的过程如下图所示：



删除Schema之前需要确认已经停止其索引；删除Schema后，在系统中该Schema并不会立即删除，而是首先需要清理该Schema之前索引的数据，索引数据清理完成后，该Schema才会被删除。

5 区块固定结构信息索引

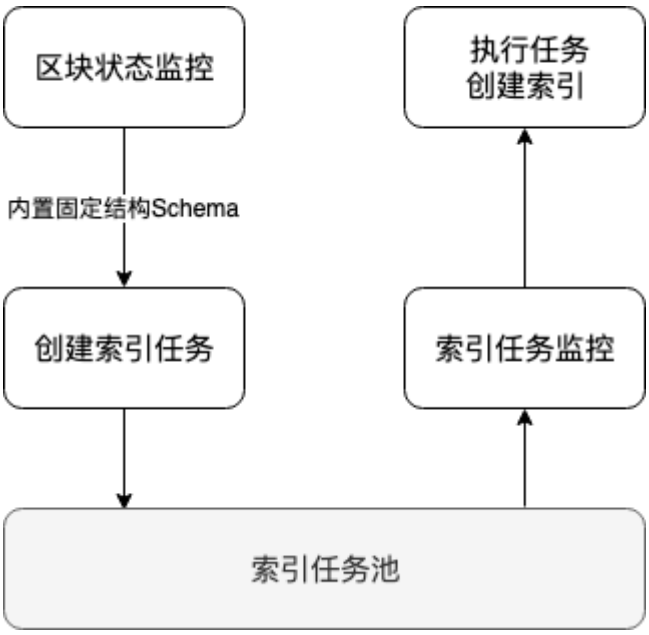
5.1 意义

区块固定结构信息包括区块链上的账本，以及账本上区块、合约、账户、交易的基础信息。

通过对固定结构信息的索引，可以根据关键字查找到相应的结构信息。

5.2 架构设计和原理

如下图所示，整个元数据索引过程以索引任务为中心完成：



索引的基本原理是，以区块为单位，为每个需要索引的区块创建一个索引任务，并将创建的任务添加到任务池中。对于任务池中的任务，由任务执行模块负责根据任务描述执行相应地索引。

因为区块会不断增加，因此有专门的区块状态监控模块负责监控区块的变化，当监测到有新区块产生时，创建针对新增区块进行索引的任务。

对索引任务池的变化有专门的组件进行监控，当有需要执行的任务时，会提取该任务，并根据任务的定义执行相应地索引。

5.3 索引内容

针对jd-chain的元信息索引，目前涵盖了以下信息的索引：

- 账本
- 区块
- 交易
- 用户
- 数据账户
- 合约
- Key和Version

6 Value数据索引

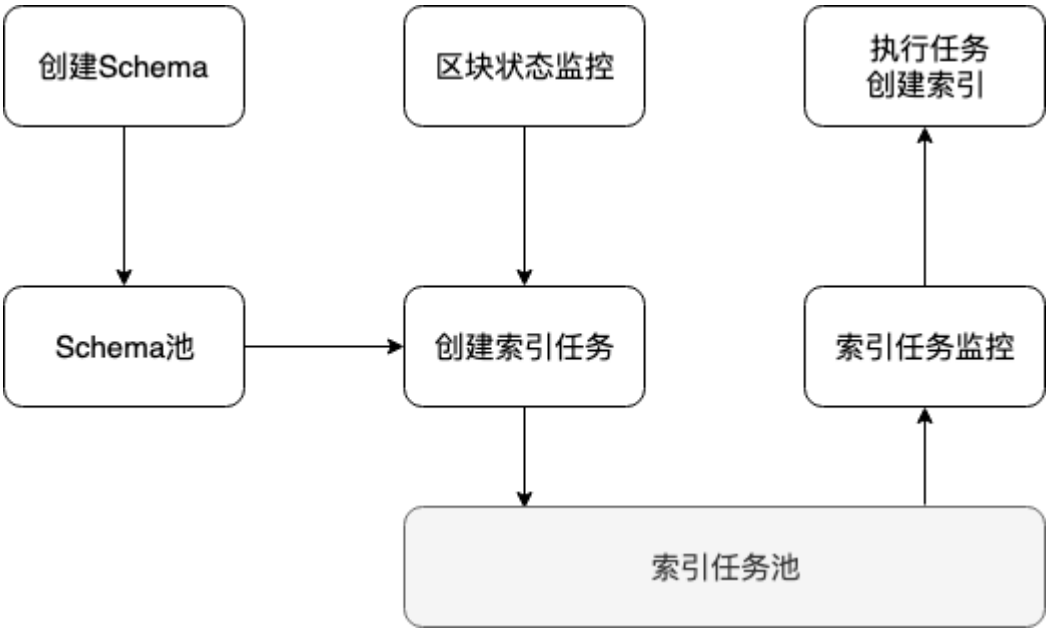
6.1 意义

对业务数据的索引，可以通过索引快速查找需要的数据，也可以从索引中发现业务本身蕴含的意义。

6.2 架构设计和原理

与固定结构的索引相比，业务数据需要首先定义业务Schema，之后在此基础上解析区块中的业务数据，并进行索引。

如下图所示，整个业务数据索引过程以业务Schema和索引任务为中心完成：



6.3 Schema

对于同样的数据，根据不同Schema解析索引的数据是不同的，每个Schema都有自身对应的索引结果。

在本系统中，Schema创建的时间可能在Value数据产生之前，也可能在Value数据产生后。

7 数据查询

7.1 查询内容

需要查询的内容包含两个主要部分：

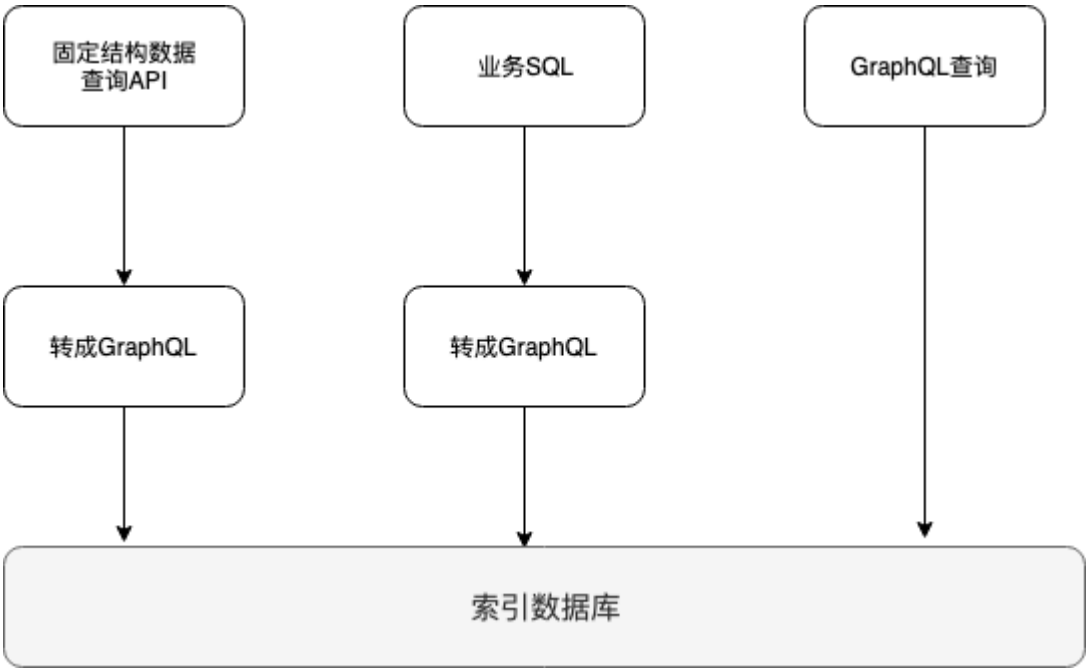
- 区块元数据查询
- 业务数据查询

针对需要查询的内容，提供了两套API进行支持：

- 针对区块元数据查询的一组API，其中的每个API针对不同的具体查询内容，用户只需要输入不同的参数即可
- 针对业务数据查询的一组API，其中的API接收用户的查询语句，查询语句在系统中执行后，将查询结果返回给用户

7.2 查询模块架构

查询模块架构如下图所示：



如架构图中所示：

1. 对于元数据的查询，通过将请求转成对应的GraphQL查询语句，实现元数据的查询
2. 对业务数据的查询，通过将查询的SQL语句转成对应的GraphQL语句，实现业务数据的查询；同时，对于比较复杂的

8 关键组件

基于之前介绍的元数据索引模块和业务数据索引模块，在数据索引方面已索引任务为核心，通过创建任务-执行任务两个步骤完成索引的工作。下面介绍一下主要的功能组件（不与具体代码挂钩）：

8.1 区块数据拉取

区块数据拉取负责拉取指定账本的数据，并完成解析，以供其它组件使用，包括状态监控和数据索引

8.2 区块状态监控

因为区块链上的区块是不断增长的，那就意味着无论是元数据还是业务都在不断变化，那么监测到这个变化，就是必须得。区块状态监控组件通过jd-chain接口监测每个账本的区块状态，并将变化信息传递给任务创建的组件。

8.3 索引任务创建

索引任务创建组件接收区块状态监控组件的事件通知，在接到事件通知后，根据目的的不同，将这些变化转化为具体的任务，例如元数据索引任务，基础业务数据索引任务，复杂业务数据索引任务等等。

创建完的任务被保存在数据库中完成持久化，并且每个任务都包含一个状态属性，标识其执行状态，任务执行组件可以根据状态筛选任务。

8.4 索引任务监控

索引任务监控组件监控数据库中需要执行的任务，发现尚未执行的任务后，根据任务种类，发起执行流程。

8.5 任务执行

索引的目的多样，因此索引任务也分多个种类，索引任务的种类通过name属性区分。

根据不同的任务种类，启动不同的执行组件来完成该任务。

任务有专门的状态字段标识任务的状态，任务执行组件针对具体种类任务的具体状态发起执行，并在执行成功后改变其状态。如果某类任务的状态没有执行组件负责，那说明这个任务已经完成（系统默认状态为0时表明任务执行完成，该任务不会再被监控到，因此不会再被执行）。

9 总结

综上所述，本文档描述了以下内容：

1. 为什么发起这个项目
2. 项目的思路
3. 技术方案的选择和具体架构，以及一些关键的细节

希望因此对项目的实际运用提供一些必要的帮助。