# 引言

## 选题背景

## 文献综述

## 研究目的及意义

## 论文主要工作

## 论文章节安排

# 相关技术背景

# 需求分析

## 功能需求分析

### 存储层需求

存储层主要解决数据采集、数据存储、数据流转换等问题。为上层计算层的相应模块提供数据支撑。

#### 研报数据采集模块

**研报数据基本情况**

卖方研究报告由券商研究所发布为主，可以从两个维度进行分类。一个维度是从获取角度，主要包括资讯平台上的公开研报、金融终端订阅的付费研报、分析师团队微信公众号内容。另一个维度是从研究报告的类别，研究报告类别角度又分为两个层次。一个是按照研究领域进行分类，新财富2020榜单将研究领域分为了30个领域；另一个是按照研报的内容，大致可分为行业研究报告、公司研究报告、宏观策略、券商晨报。

从采集器的设计角度，需要对不同数据源和采集频率差别较大的情况分别设计研报数据采集器。

**数据源**

采集模块需要涵盖的数据源包括：1. 东方财富研报中心 2. 东方财富公告大全。

东方财富研报中心，涵盖部分券商的公开研究报告，包括行业研究报告、公司研究报告、宏观研报、新股研报、策略报告、券商晨报。其中宏观研究、策略研报、券商晨报可以归为行业研究报告类型，新股研报可以归为公司研究报告。两类报告的内容在结构上会有所不同，数据处理的流程会有差异，从灵活性出发需要针对性的需要设计两套研报采集器来定期获取两类研究报告数据。

上市公司公告分为定期公告和临时公告，公告发出后，卖方研究会及时发出相应点评。因此对于需要及时获取的事件点评类研究报告，可以根据将公告作为线索来获取点评报告。因此需要设计一个独特的采集器，首先获取公告大全中相应的公告信息，然后根据公告对应的上市公司去及时获取对应公司的事件点评报告。

#### 研报数据存储模块

**数据结构多样性**

一篇研报的数据由4部分组成：研报摘要、标题及预测等相关信息、研报内容。研报摘要一般是文字形式的非结构化数据；研报内容一般为PDF格式的文件，少数为PPT格式的文件；标题及预测等相关信息为结构化数据。因此需要存储模块能够对文字内容、文件这类非结构化数据以及结构化数据都具有较好的支撑。

**数据存储量**

研报数据的原始形态一般为PDF格式的文件，其中包含大量的图片。平均每篇研报的大小在10MB左右，根据每日500篇研报测算，每日新增存储在5GB。系统应能够存储近前后10年内的所有研报，存储需求大约为40TB。

**一致性需求**

一个研究领域和一家上市公司相关的研究报告应能够尽量存储在同一个表结构中。由于数据源的多样性，和采集器并发的不确定性，需要存储模块能够在性能要求范围内解决一致性问题。

### 计算层需求

计算层基于存储层的数据进行转换，最终为应用层的操作型应用提供数据。

#### 研报内容结构化模块

研报内容一般以PDF文件的形式进行发布，对数据分析应用并不友好。因此需要将研报内容结构化。

研报内容以PDF形式进入数据存储层后，结构化模块对存储进来的研究报告进行处理，将一篇研究报告变成半结构化的数据。

研究报告一般逻辑性较强，描述内容的目的性明确，篇章结构化的转换能够提高研报的精简性同时方便热点挖掘模块的计算。

用途主要有两个：

1. 篇章结构化，为结构化的研报阅读功能提供数据基础
2. 为热点挖掘模块的计算提供数据基础

#### 研报热点挖掘模块

研报热点挖掘模块通过分析研报内容结构化模块处理后的数据，为热点挖掘模块提供数据支撑。

### 应用层需求

**结构化的研报阅读**

**市场热点发现**

## 性能需求分析

### 研报数据采集模块

不同研究报告类型的发布周期与频次有着很大的不同。从买方投研需求角度，一般报告以日频更新即可满足需求，但是对于突发事件的点评类报告却需要足够的实时性，采集的及时性应在分钟级别。

### 研报数据存储模块

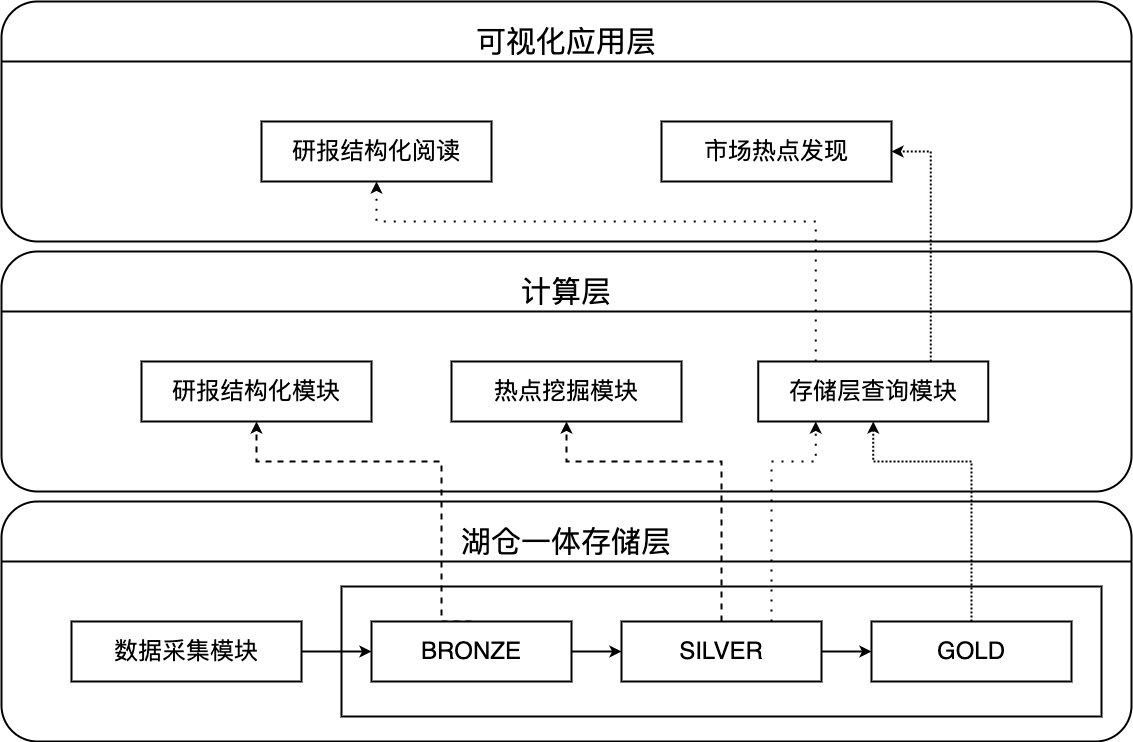
### 研报内容结构化模块

### 研报热点挖掘模块

## 可行性分析

# 系统设计

### 整体架构设计



系统整体分为三层，从数据进入系统到应用分为湖仓一体存储层、计算层、可视化应用层。

湖仓一体存储层负责整个大数据架构的存储，为计算层提供分析模型所需要的数据准备。湖仓一体存储层包括两大部分，意识数据采集模块，从各种研报数据源采集研报数据进入系统。数据采集模块采集的研报数据会被写入使用湖仓一体存储引擎DELTALAKE控制的对象存储中。存储采用湖仓一体的三层经典架构：

Bronze表：存储进入系统的原生数据，获取到研报数据不做修改的存储。

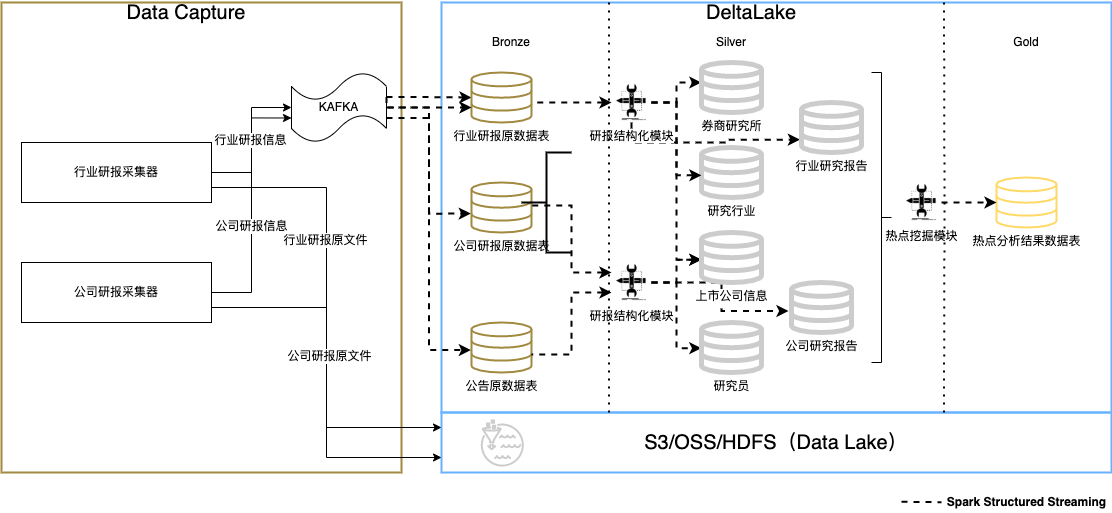
Silver表：该表是在 Bronze表的基础上进行加工的到的中间数据，一般对Bronze中的原始数据进行转换清洗，成为数据科学训练的数据源。

Gold表：该表是基于业务需求的数据表，表数据是高度集成的，用于最上层的可视化应用。

计算层提供了系统中所有计算相关的功能，包括了湖仓一体层数据转换的功能模块，研报进入原生数据表即Bronze表后会使用研报结构化模块将文件形式的研报转换成结构形式的研报送入Silver层的表中。进入Silver层的表的数据会被热点挖掘模块作为数据源来进行分析，的到热点趋势分析结果数据，分析的结果数据会被送入Gold层的表中。存储层的查询模块因为涉及到多个存储层数据的查询和查询策略算法的设计所以也放在计算层中。存储查询模块从Silver层的表中查询结构化处理后的研报数据提供给可视化应用层中的研报结构化模块，从Gold层的表中查询热点分析后的结果数据提供给可视化应用层中的市场热点发现模块，从Bronze层中查询研报原始数据供用户预览下载。

最上层是面向用户的可视化应用层，这一层的内容不是固定不变的，随着用户应用场景的扩充会不断有新的应用场景诞生，就需要在应用层实现更多的模块。当需要实现新的功能时计算层需要增加对应的模块提供数据，向湖仓一体存储层传导就会新增一些对应的表内容。

### 存储层设计



**这里还要改一改，Silver层的数据是否直接被应用层来查，数据结构是否合适**

存储层在整体上分为数据采集模块和湖仓一体存储模块两大部分。

如图，左边为数据采集模块DC（Data Capture）部分。根据对源数据的分析可以用两个研报采集器进行覆盖，分别为行业研报采集器和公司研报采集器。采集器从数据源获取研究报告原文，和研报相关信息（作者、发布时间、机构、所属行业等）。将研究报告的原文存储到数据湖中，考虑到研究报告原文数据量很大，架构中选用经济性突出的公有云对象存储；将研报相关信息和原文存储路径存储到DeltaLake的Bronze层中。

如图，右边为湖仓一体的存储模块。DeltaLake同样使用研报原文存储的对象存储作为存储单元，使得整体架构非常精简，维护成本大幅降低。虚线表示使用Spark Structured Streaming产生的数据流。研报相关信息被送入KAFKA后，就成为第一个数据流的写入并且输出到湖仓一体存储模块的Bronze层中，该层不对数据做任何处理，存储原始文件和原始信息。随后进入Bronze层中的数据成为下一个数据流的写入，通过研报数据化模块的转换，将研报原文和相关信息转换成结构化的数据存储到Silver层的表中供热点挖掘模块使用，热点挖掘模块使用Silver层中的结构化数据得出热点分析结果数据写入到Gold层中的热点分析结果表。

#### 研报采集模块设计

获取研报的数据源有两种形式，一个是从API形式的订阅接口获取，一个是从券商研究所的公众号中获取。API形式的订阅接口中包括了行业研究报告和公司研究报告，券商研究所的公众号中获取的一般为行业研究报告。

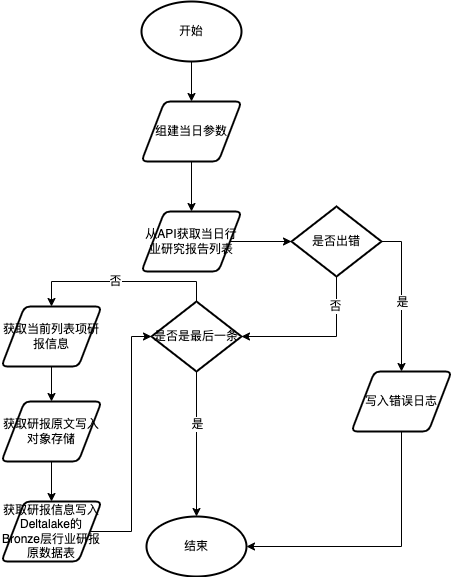
行业研究报告的发布往往没有特定时间点，一般聚焦在某一段时间的行业热点问题，但是为了实现行业热点挖掘功能，必须至少满足日频的采集频率。

公司研究报告的发布，一类是和行业研究报告发布具备同样特性的不定期覆盖报告，另一类是事件驱动的点评类报告，其中以上市公司公告事件为主要。

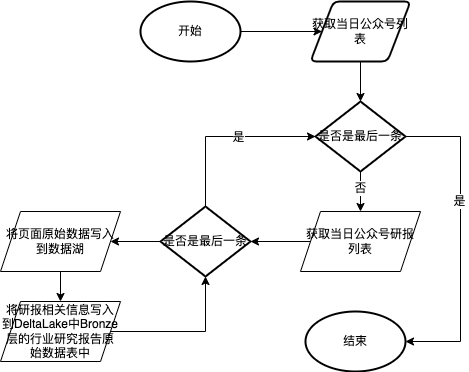
根据上述对数据源的分析，需要实现三类数据采集器。

1. API接口行业研究报告采集器（日频采集，覆盖所有研究领域）
2. 公众号行业研究报告采集器（日频采集，覆盖部分券商研究所）
3. API接口公司研究报告采集器（日频采集，覆盖所有上市公司）

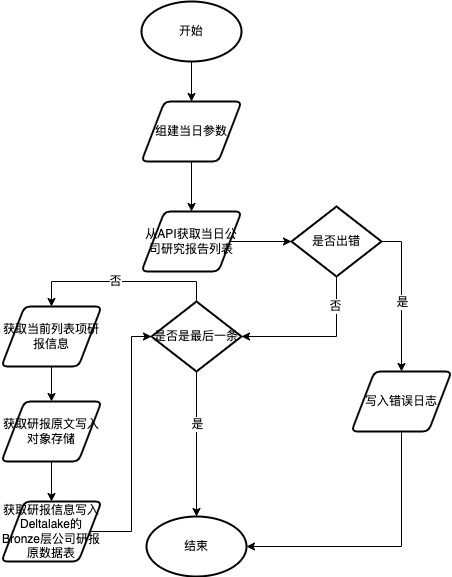
API接口行业研究报告采集器的采集过程：



公众号行业研究报告采集器：



API接口公司研究报告采集器的采集过程：



两个API接口的采集器逻辑上基本相同，都是先组建相应的API参数获取到研究报告的列表信息，然后遍历这个列表去获取详细的内容。获取详细内容后，首先将研报原文写入到数据湖中也就是云上的对象存储中去，然后将研报相关信息写入到KAFKA中，这会触发湖仓一体存储模块的数据流处理。

公众号行业研究报告采集器的逻辑略有不同，通过遍历需要观察的公众号获取行业研究报告，先获取当日所有的报告列表，获取到研报对应的URL后，将网页内容原始数据保存到数据湖也就是云上的对象存储中去，然后获取研报相关信息，写入到KAFKA中，触发湖仓一体存储模块的数据流处理。

#### 研报数据存储模块设计

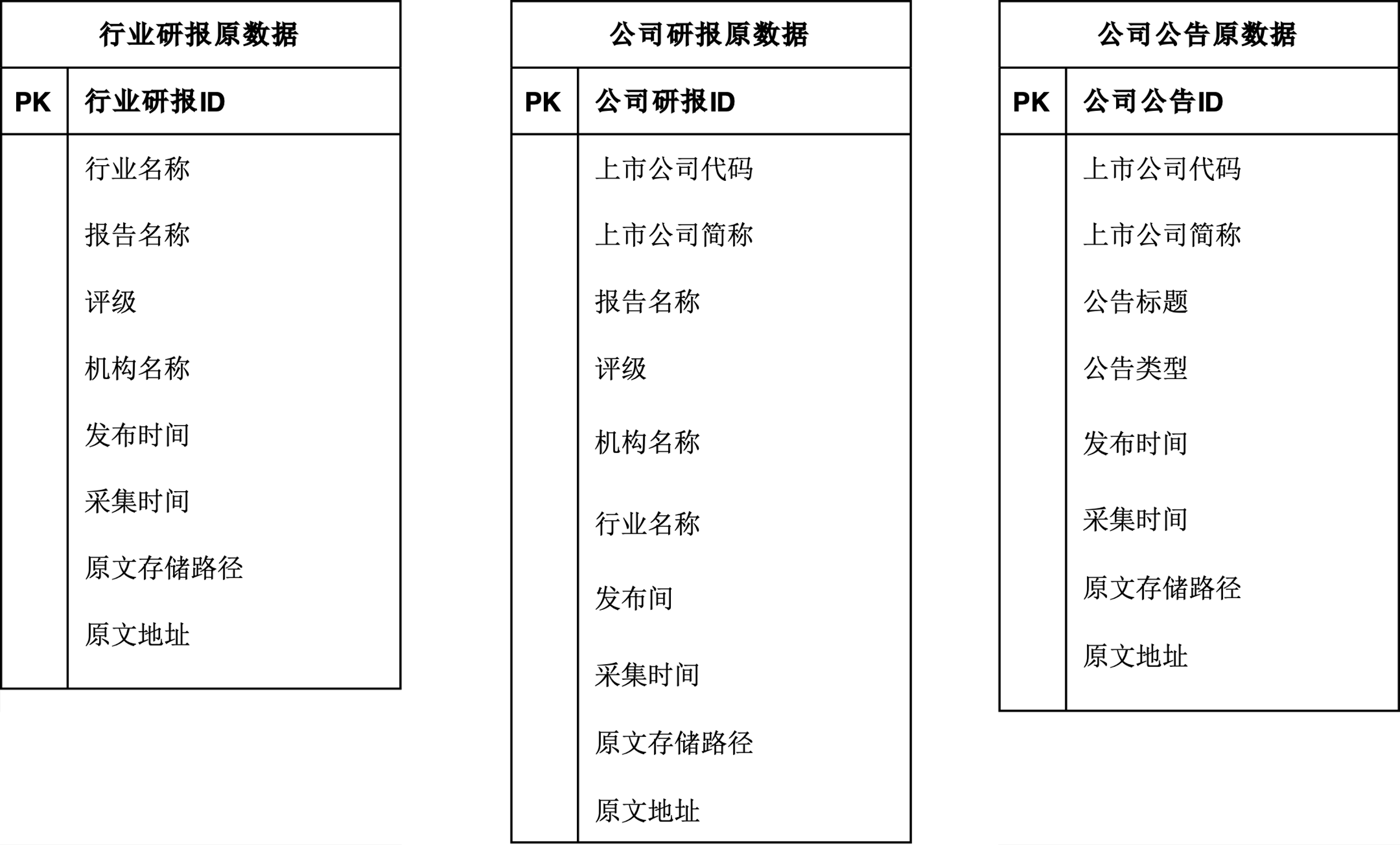
**存储**

存储模块采用DeltaLake为存储引擎，底层存储采用对象存储，可以选用HDFS、AWS S3、阿里云OSS等支持DeltaLake接口的对象存储。

使用DeltaLake的最直接好处是数据流转换的三层中的表都以PARQUET格式存储在对象存储上。研报原文件也同样存储在对象存储之上。架构较过往采用的LAMDA和KAPPA架构有了极大的精简。

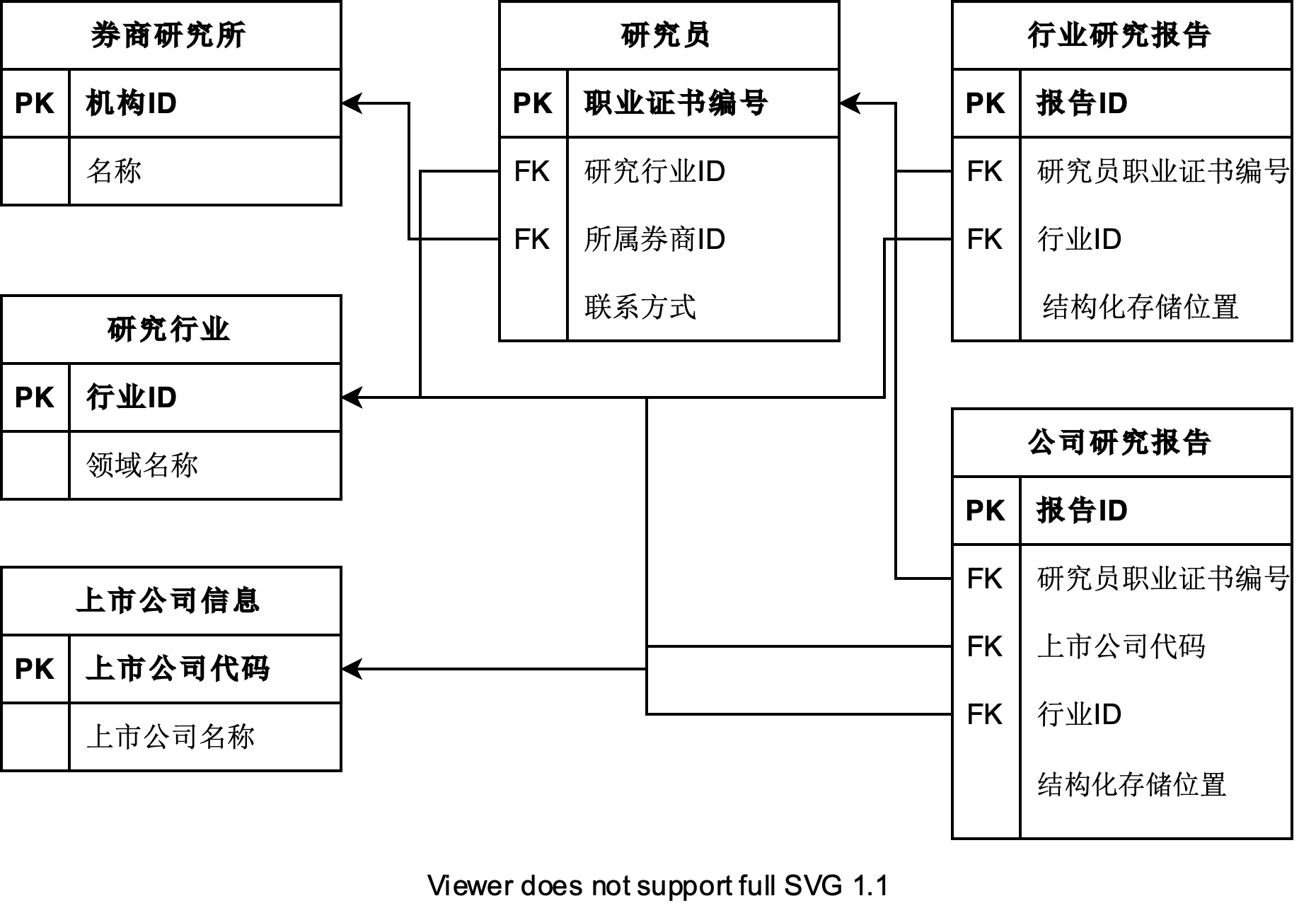
**三层数据架构**

DeltaLake中的表分为三层，Bronze表包括三张表，一是行业研报原数据表，二是公司研报原数据表，三是公告原数据表。这三张表在Delta Lake中的表结构设计如下：



这三张表的设计基本按照源数据的原始信息来进行存储，因此表的设计并不需要满足任何范式要求。

Silver层的表的内容有两个来源，一个是Bronze层的原数据表，另一个是来自研报结构化模块处理存储在对象存储中的研报文件的到的信息，Silver层的表的用途一是提供给热点挖掘使用另一个是提供给研报阅读模块使用，因此表结构在设计中尽量去满足第三范式，同时为了上层应用的便利性会做出部分让步。



在Silver层中的几张表中，研究员表中的全部信息、行业研究报告表中的结构化存储位置、公司研究报告表总的结构化存储位置都来自于研报结构化模块对研报原始文件的分析结果。

Gold层的表直接面向应用场景，给市场热点发现模块提供数据。

######这一部分的数据表设计待设计，暂时不确定下来。#######

**使用Structured Streaming实现持续计算**

Structured Streaming是Spark中实现连续流处理的机制，能够实现数据流的自动流转。外部采集的数据从KAFKA中进入DeltaLake中的Bronze层，以及DeltaLake中各层次表之间的数据流转都是用Structured Streaming来进行处理。

一共需要实现10个Structured Streaming处理单元，列表如下：

|  |  |  |
| --- | --- | --- |
| **序号** | **ReadStream** | **WriteStream** |
| **KAFKA-Bronze层** | | |
| 1 | KAFKA行业研报采集 | DeltaLake 行业研报原数据表 |
| 2 | KAFKA公司研报采集 | DeltaLake 公司研报原数据表 |
| 3 | KAFKA公司公告采集 | DeltaLake 公告原数据表 |
| **Bronze层-Silver层** | | |
| 4 | DeltaLake行业研报原数据表 | DeltaLake券商研究所 |
| 5 | DeltaLake研究行业 |
| 6 | DeltaLake行业研究报告 |
| 7 | DeltaLake研究员 |
| 8 | DeltaLake公司研报原数据表 | DeltaLake券商研究所 |
| 9 | DeltaLake研究行业 |
| 10 | DeltaLake研究员 |
| 11 | DeltaLak公司研究报告 |
| **Silver层- Gold层** | | |
| 12 | DeltaLake行业研究报告 | DeltaLake热点分析结果数据表 |
| 13 | DeltaLake公司研究报告 | DeltaLake热点分析结果数据表 |

### 计算层设计

计算层提供两方面的功能，一是为湖仓一体存储层中的数据流转提供算法支撑，另一个是为上层的应用层提供DeltaLake Gold层数据的查询服务，集中进行查询算法的优化。一共包含三个模块，研报结构化模块处理从Bronze层到Silver层的数据流转的算法实现，其核心是实现研报文章的结构化，提供篇章结构化相应的算法；研报热点挖掘模块处理Silver中的表，得到热点趋势数据存储到Gold层中。

#### 研报结构化模块设计

研究报告一般是以PDF格式的文件进行发布，公众号中以文章形式进行发布。内容角度，研究报告一般逻辑性非常强，段落之间关系，段内关系都比较明确，一般都是总分的形式。并且文章开头会写摘要来表达整篇研报的核心内容。同时研报标题内容都为核心观点。

研报结构化模块的功能是将PDF格式的文件内容转换成JSON形式的结构化研报内容，一是为研报热点挖掘模块提供数据，二是配合应用层的研报阅读模块给用户提供更简洁高效的研报阅读体验。

同一家券商研究所一般使用相同的格式来发布研究报告，因此研报结构化模块需要具备足够的兼容性，对不同券商研究过发布的研报都能提供结构化算法的支撑。

#### 研报热点挖掘模块设计

#### 存储层查询模块设计

### 应用层设计

#### 市场热点发现模块设计

#### 结构化研报阅读模块设计

# 系统实现

### 研报采集模块实现

### 研报数据存储模块实现

### 研报内容结构化模块实现

### 研报热点挖掘模块实现

# 系统测试与验证

## 系统测试概述

## 系统测试环境

## 系统功能测试

# 总结与展望

## 论文工作总结

## 工作展望

# 参考文献

# 致谢