# 项目报告

- 对每种存储方式结合本项目说明各自适用于处理什么查询，针对本项目在存储优化中做了什么工作，优化前后的比较结果是怎样的?
- 如何保证数据质量？哪些情况会影响数据质量?
- 数据血缘的使用场景有哪些?

## 1. ETL概述

在本次练习中, 我们使用python爬取了5.5万亚马逊商品页面, 使用Python脚本和kettle进行数据清洗和处理, 筛选出约2.5万电影数据。

在爬虫设计中, 我们使用 ip代理, 使用 fake-useragent 随机切换浏览器UA等方式来突破Amazon的反爬虫机制。使用re、xpath和bs4来解析爬取到的html、提取商品信息, 使用 jsonlines 格式存储网页解析后的数据。

在数据清洗流程中, 我们通过解析商品信息的几个特定属性进行判断, 尽可能地将非电影的商品信息去除而将电影的商品信息保留。

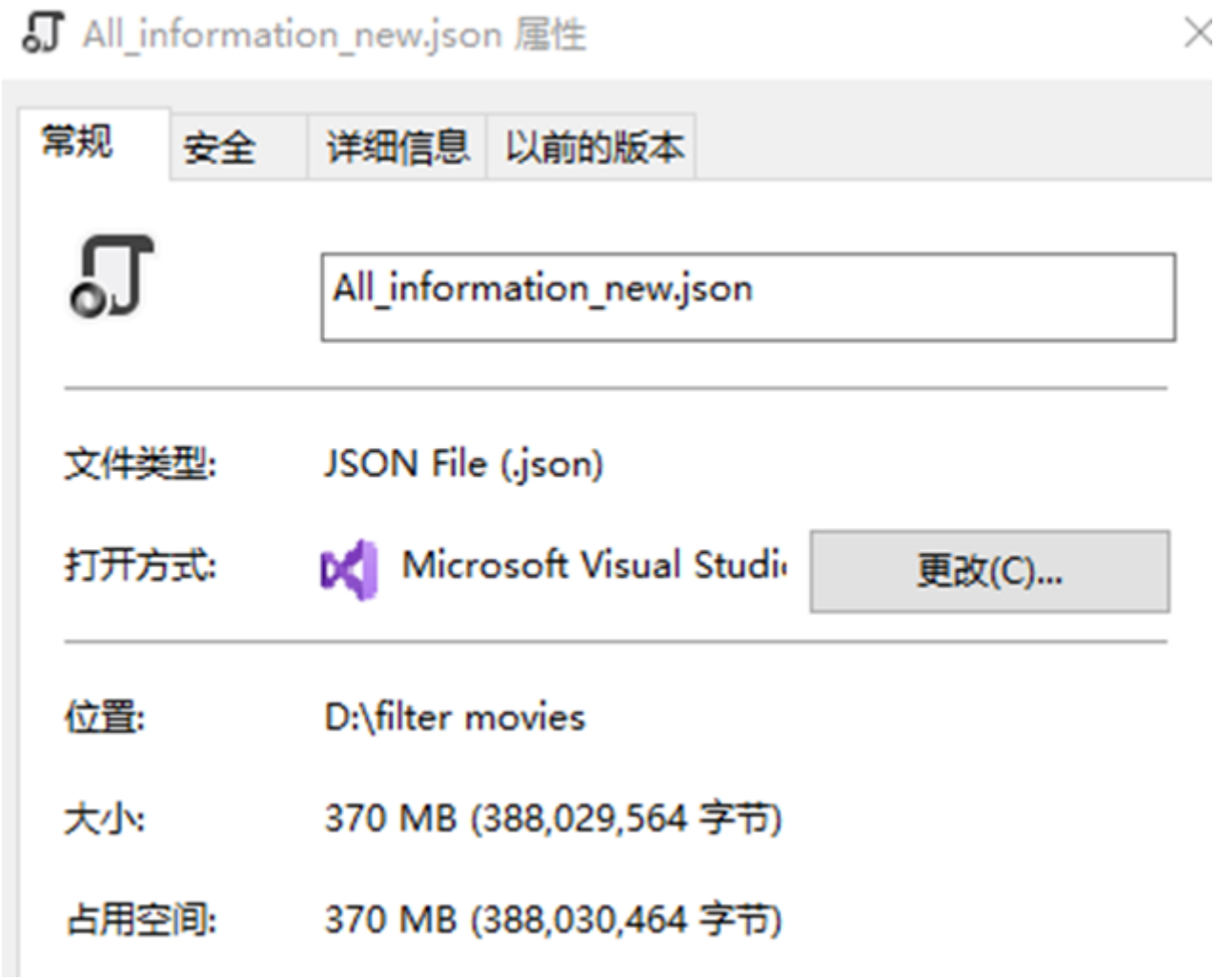
最终, 我们将剩余信息导出为CSV文件, 导入数据库。

### 1.1 数据获取

原始数据集来自亚马逊电影评论数据集https://nijianmo.github.io/amazon/index.html的"Small" subsets for experimentation 中的 Movies and TV 5-core，用python从中提取出6万asin。使用python爬虫爬取以这些asin构造url的Amazon网页信息，记录下在爬取过程中发现的失效页面

使用re模块、xpath模块和bs4模块解析这些页面，以jsonlines格式存储解析出的信息。由于Amazon的不同页面格式可能存在区别，使用大量try语句块避免异常，提高程序的健壮性。

最终结果为：约1000个页面失效或因缺失过多信息被丢弃，成功获得约55000条信息。


All_information_new.json 属性
常规
安全
详细信息
以前的版本
All_information_new.json
文件类型:
JSON File (.json)
打开方式:
Microsoft Visual Studi
更改(C)...
位置:
D:\filter movies
大小:
370 MB (388,029,564 字节)
占用空间:
370 MB (388,030,464 字节)


{"asin": "B000HKDEL0", "title": "Billy Wilder Speaks", "star_rating": "4.2 out of 5 stars", "style": "Documentary", "production_format": "DVD", "production_detail": "{'Aspect
'Ratio': '1.33:1', 'Is Discontinued By Manufacturer': 'No', 'MPAA rating': 'Unrated (Not Rated)', 'Product Dimensions': '0.7 x 7.5 x 5.4 inches; 3.92 Ounces', 'Item model number':
'2254939', 'Director': 'Gisela Grischow, Volker Schl\u00f6ndorff', 'Media Format': 'Multiple Formats, Closed-captioned, Color, Full Screen, NTSC, Subtitled, Black & White', 'Run
time': '1 hour and 11 minutes', 'Release date': 'October 17, 2006', 'Actors': 'Jean Arthur, Humphrey Bogart, James Cagney, Gary Cooper, Tony Curtis', 'Subtitles': 'English',
'Producers': 'Bret Wood, Eberhard Junkersdorf, Tom Brown, Volker Schl\u00f6ndorff', 'Language': 'Unqualified', 'Studio': 'Kino Lorber films', 'ASIN': 'B000HKDEL0', 'Writers':
'Volker Schl\u00f6ndorff', 'Number of discs': '1'}", "additional_option": "['B01M9IBTIQ', 'B00UGPNWBI', 'B01M08FUAH']", "other_format": "[]", "all_user_review": "[{'user_id':
'AH4GLZ6R27YASGUIX43CO42UGJHA', 'user_name': 'ignacio f.', 'rating': '4.0 out of 5 stars', 'review_summary': 'Fascinating', 'review_date': 'June 20, 2018', 'review_text': \"It
probably helped that Wilder's interlocutor spoke German. Fascinating material.\", 'helpfulness': ''}, {'user_id': 'AH67INGSDCFNE7FDPLMNL4CPBYRA', 'user_name': 'Kindle Customer',
'rating': '5.0 out of 5 stars', 'review_summary': 'Great gift for the TCM fan in your life.', 'review_date': 'January 20, 2014', 'review_text': 'Billy is the master, everyone
quotes him but this video lets you hear it from the horses mouth. The documentary filmmaker did a great job of letting Billy speak his mind, but also prompted him with questions
his fans want answers to most. Great buy only on Amazon Video.', 'helpfulness': 'One person found this helpful'}, {'user_id': 'AFTFMH5GJGN6ZKPB6FST4Y4MJ5WA', 'user_name': 'NG -
Real Comment', 'rating': '4.0 out of 5 stars', 'review_summary': 'A must have for film aficionados', 'review_date': 'April 3, 2013', 'review_text': 'I have a dvd collection of
the great moguls, directors (Warner Brothers, Goldwyn, John Ford, Louis B Mayer, MGM, Harry Cohn, etc), and this was an important one to add to my collection.  I love the movies
and the studios, directors, great actors who entertain us.', 'helpfulness': 'One person found this helpful'}, {'user_id': 'AE5LHZJD76NBKZJV2V26MMMCRMBQ', 'user_name': 'Amazon
Customer', 'rating': '5.0 out of 5 stars', 'review_summary': 'Billy, we miss you.', 'review_date': 'September 7, 2008', 'review_text': 'There was/is no greater film director than
Billy Wilder. This DVD appears to be a cut down version of a three part TV documentary interview.  Still, it is a \"must have\" for any film fan or anyone interested in cinema.
Billy Wilder was simply the best.\\n\\nBloody Ham', 'helpfulness': 'One person found this helpful'}, {'user_id': 'AE5JGXO3VUKB4LUOFANGU5UV6FCA', 'user_name': 'sarah hundt',
'rating': '5.0 out of 5 stars', 'review_summary': 'Billy Wilder is a shining director', 'review_date': 'October 21, 2015', 'review_text': 'What a genius.  Up there with Mel
Brooks', 'helpfulness': ''}, {'user_id': 'AHKJIWVLF7UZI2FQRZTEX3UHA6FA', 'user_name': 'Andy Harmon', 'rating': '5.0 out of 5 stars', 'review_summary': 'Loved It', 'review_date':
'March 11, 2015', 'review_text': 'I loved this film. A great look at one of the worlds greatest directors.', 'helpfulness': ''}, {'user_id': 'AHT22SYXKJXIGD5TISJCWVZVRYSA',
'user_name': 'jla', 'rating': '5.0 out of 5 stars', 'review_summary': 'Five Stars', 'review_date': 'September 12, 2015', 'review_text': 'great director', 'helpfulness': ''},
{'user_id': 'AEVDFZXATVIG24BYNY3JNF6JPIJQ', 'user_name': 'A. Pacheco Jr.', 'rating': '3.0 out of 5 stars', 'review_summary': 'Rare interview with raw filmaking', 'review_date':
'May 19, 2008', 'review_text': \"It's a priviledge to watch a master speak. The option to keep the filming simple as closeups with movie clips is a little too simple. Also the
decision to provide Enlgish subtitles only to the german spoken segments is confusing.\", 'helpfulness': '4 people found this helpful'}, {'user_id':
'AG2AUO7WP6QSKE67LQ2RIEWQNN7A', 'user_name': 'Marilyn A. Rock', 'rating': '5.0 out of 5 stars', 'review_summary': 'My husband just loved it.', 'review_date': 'August 6, 2008',
'review_text': 'My husband is studying German and Billy Wilder goes from English to German during the interview.I also found it to be very entertaining.', 'helpfulness': ''},
{'user_id': 'AGLMH7GYYOVSH3IFQDXJZQLWZ3HQ', 'user_name': 'calvinnme', 'rating': '5.0 out of 5 stars', 'review_summary': 'An informal conversation turns into a great interview',
'review_date': 'July 10, 2009', 'review_text': 'The maker of this documentary and director Billy Wilder originally intended what was filmed here to just be a dry run of what
would be the later actual interview portion of a documentary on Wilder\\'s career. The two sit down in Wilder\\'s crowded office and Wilder just starts talking about the various
aspects of his long career. The one stipulation that Wilder made was that this footage not be released in his lifetime. Wilder and the interviewer go back and forth between
German and English - depending upon what language best expresses the points they wish to make - with helpful subtitles for those of us who speak English when either speaks
German.Wilder says some things that don\\'t surprise me - for example that Jack Lemmon was the definition of a professional. Wilder would not have used him so much and Lemmon
would not have been such a great performer had that not been the case.  However, Wilder\\'s insights into Marilyn Monroe were new to me. He said while making \"Some Like it Hot\"
that sometimes they would spend all day trying to get one take in which Marilyn had just one line, to the point where he wanted to pin the line to the wall so she could just
read it. Other days she would come in and have pages of dialogue memorized. He also had some interesting things to say about making films on the Holocaust immediately after the
war and the impact they had on German audiences at the time.At any rate, this dry run turned out to be so good that it became the actual interview. I highly recommend it to
people who are interested in Billy Wilder\\'s career, since it is almost entirely Billy Wilder talking about the projects he worked on, his philosophies of filmmaking, and the
people with which he worked. It\\'s a fascinating documentary.', 'helpfulness': '9 people found this helpful'}]"}
{"asin": "B00MP2FVIC", "title": "One Piece: Season 6: Voyage Two", "star_rating": "4.8 out of 5 stars", "style": "Anime & Manga", "production_format": "DVD", "production_detail":

## 1.2 数据清洗与处理

判断标准：

- 我们仔细分析了亚马逊平台上的电影商品信息, 找到了一些它们不同于非电影商品的共同点, 以此为判断依据来分析一个商品是否是电影

1. 电影商品以及类电影商品(如纪录片)通常含有Director属性, 而多集电视剧一类的商品通常采用多导演拍摄, 因此信息中通常没有Director属性. 因此可以观察商品信息是否包含有Director属性, 如果没有则认为不是电影
2. 商品信息中是否出现视频长度, 如果出现而且时长过长或过短则认为不是电影
3. 商品信息中是否出现MPAA分级, 若有此属性且属性为NotRated一类的无意义分级, 则认为该商品不是电影
4. 部分非电影商品具有明显特征, 如带有Boxed Sets 和TV
5. 商品格式为Audio CD的认为不是电影

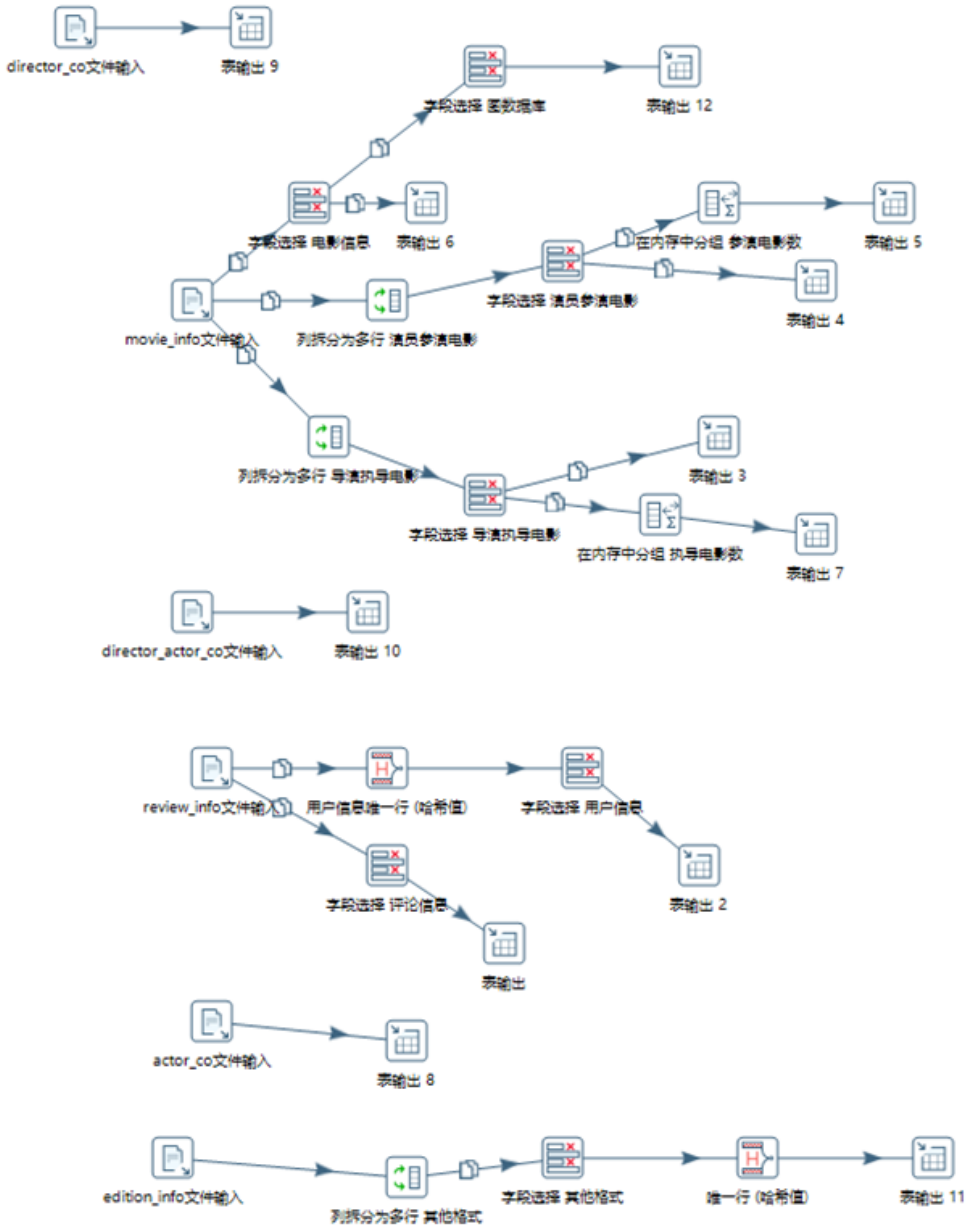
筛选之后剩下约2万3千条我们认为较大概率是电影的信息。

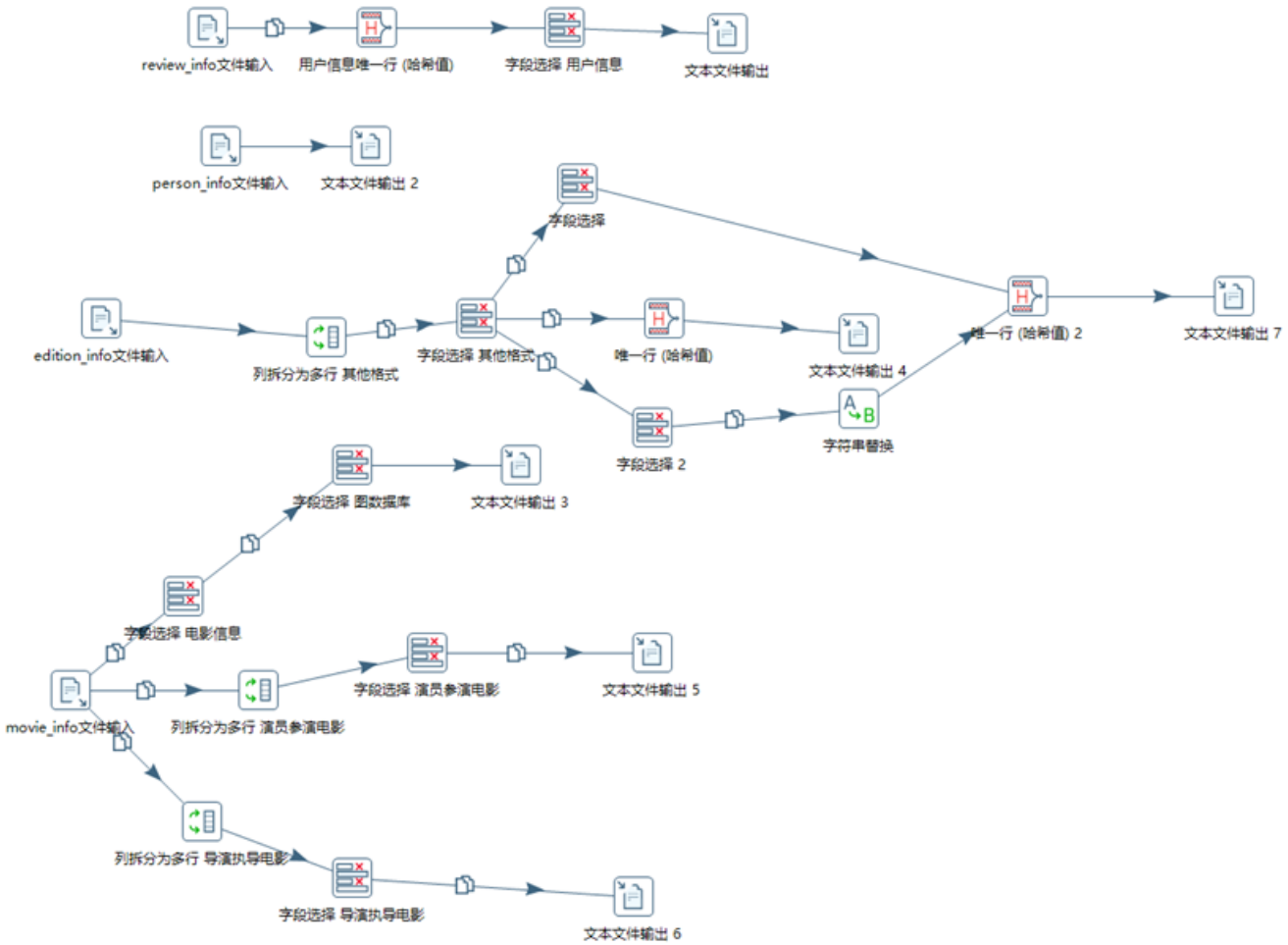
数据处理：

部分商品的Director和Actor可能为*或Various，将这些处理成空值

由于页面格式不同，电影style的数据可能在Amazon电影风格列表之外，将这些数据处理成空值

最后使用python脚本与kettle将jsonlines格式的信息转换成能导入数据库对应表的csv文件。


director_co文件输入
表输出 9
字段选择 医数据库
表输出 12
字段选择 电影信息
表输出 6
在内存中分组 参演电影数
表输出 5
字段选择 演员参演电影
表输出 4
movie_info文件输入
列拆分为多行 演员参演电影
列拆分为多行 导演执导电影
表输出 3
字段选择 导演执导电影
在内存中分组 执导电影数
表输出 7
director_actor_co文件输入
表输出 10
review_info文件输入
用户信息唯一行 (哈希值)
字段选择 用户信息
表输出 2
字段选择 评论信息
表输出
actor_co文件输入
表输出 8
edition_info文件输入
列拆分为多行 其他格式
字段选择 其他格式
唯一行 (哈希值)
表输出 11




## 1.3 溯源查询

### 溯源查询

从约55000条数据中筛选出31952条非电影数据以及667个失效页面。

共发现3部哈利波特系列电影

```
f = False
for i in title_list:
    if f is True:
        title_list.remove(i)
    serial_movie = []
    flag = False
    for j in title_list:
        if i != j:
            if difflib.SequenceMatcher(None, i[0], j[0]).quick_ratio() > 0.6 and difflib.SequenceMatcher(None, i[2], j[2]).quick_ratio() > 0.8 and difflib.SequenceMatcher(None, i[3], j[3]).quick_ratio() > 0.8:
                serial_movie.append(j)
                flag = True
    if flag is True:
        sub_list = []
        for info in serial_movie:
            sub_list.append((info[0], info[1]))
        csvwriter2.writerow([(i[0], i[1]), sub_list])
    for title in serial_movie:
        title_list.remove(title)
```

通过电影名称、导演与演员的相似性判断两部电影是否是同一部电影或是系列电影。

# 2. 存储方式及优化

根据实验，得出如下结论（详情请见数据存储涉及说明文档）

- Mysql适用于小数据量的单表查询；
- Neo4j适用于处理大量复杂的、相互连接的数据，可以规避关系型数据库中的大规模的表连接，比如查询两个演员合作最多的次数等；
- 而Hive更适用于实时性要求不高的场合。如果数据量足够大，那么相对于分析统计海量数据的执行时间，Hive的延迟已经不能成为它的缺点，但是在本项目中，由于数据量并没有那么大，如果从效率的角度来考虑并不适用。当然，由于Hive搭建在分布式文件系统的基础上，所以具有很好的扩展性和容错性。

# 2.1 Mysql 优化

## 2.1.1 表结构优化

### 2.1.1.1 增加冗余的优化

首先通过将部分冗余的信息存储在不同的表中，提高了查询的效率。比如我们在电影表中存储了电影的版本信息：电影的版本和其他版本的数量，而并没有将其作为表的形式抽象出来；在电影和用户的关系表中存储了评论的内容，评分等信息，而并没有将其作为表的形式抽象出来。这是为了尽量减少联表查询的需求，以应对更加复杂的查询条件，提高查询效率。

### 2.1.1.2 增加统计属性

对于一些高频的、高复杂度的查询任务，我们选择提前将其计算出来，作为属性冗余的存储在数据库中，虽然增加了冗余，但是大大加快了查询效率。比如演员和演员，演员和导演，导演和导演之间的合作次数，电影版本的数量，每个演员或导演参加的电影总数，这些信息都被提前计算并存储在数据库中，提升了查询效率。

## 2.1.2索引优化

为某些关键的字段添加索引，可以很有效的提高数据查询和导入数据库的时间效率，尤其是数据仓库的项目中，数据量非常庞大，添加索引对效率的增益就非常突出。

举例测试结果如下：

1. 在movie电影实体表中，添加发布日期：release_year,release_month,release_day作为联合索引

| 查询类型 | 添加索引前时间(ms) | 添加索引后时间(ms) |
| --- | --- | --- |
| 查询某一年某一月某一日有多少电影 | 226 | 2 |

2. 在actor_actor演员关系表中，添加演员姓名actor_name1，actor_name2作为索引

   在director_actor演员导演关系表中，添加演员姓名actor_name，导演姓名director_name作为索引

| 查询类型 | 添加索引前时间(ms) | 添加索引后时间(ms) |
| --- | --- | --- |
| 按人名名称查询合作最多的演员和合作次数 | 44 | 5 |

可见，索引的使用对于该数据仓库的效率的提高作用很大。

## 2.2 Hive 优化

### 2.2.1 存储空间优化

经查阅文献资料发现，Hive提供了多种文件存储的格式，常见的有TextFile，Parquet，ORC，Sequence，RC，AVRO等。其中Textfile文件存储格式未经压缩，会占用大量存储空间，并且查询速度较慢，而ORC文件由于是列示存储，对于不需要查询整张表的所有数据的sql，存储格式的压缩比与查询速度表现都比较良好，所以我们选择以ORC文件格式存储数据，因此将上述以TextFile文件格式存储的数据库转换为ORC存储格式。

新建一个数据库 create database orcdata;然后把原数据库的每张表导入到新的数据库

insert overwrite table orc_database_name.table_name select * from text_database_name.table_name; 命令把Textfile文件格式的表放进ORC文件格式的数据表中。

| | A | B |
| --- | --- | --- |
| 1 | **TEXTFILE** | **ORC** |
| 2 | 39.0M | 15.9M |

可以看到存储空间压缩了将近百分之40，效果非常好

下面分别执行两条sql来对比性能

**查询某电影的所有评论用户数量**

SQL

```
select count(*) from review where asin='6302782082'
```

| | A | B |
| --- | --- | --- |
| 1 | **TEXTFILE** | **ORC** |
| 2 | 5s 209ms | 2s 303ms |

**查询某电影评分大于3的评论平均分**

```sql
select avg(rating) from review where rating>3 and asin='6302782082'
```

| | A | B |
|---|---|---|
| 1 | **TEXTFILE** | **ORC** |
| 2 | 4s 560ms | 4s 767ms |

### 2.2.2索引优化

ORC文件格式内置多种轻量级索引，已经能提供很好的查询性能优化，为验证建立密集索引是否会提升查询性能，

**执行语句：查询评分等于3的评论信息**

```sql
Select count(*) from review where rating=3
```

| | A | B |
|---|---|---|
| 1 | **无索引** | **有索引** |
| 2 | 7s 670ms | 7s 950ms |

发现建立索引反而会降低查询速度，推测可能与ORC文件的压缩方式与存储方式有关

### 2.2.3分区表优化

Hive在使用条件查询的时候会扫描整张表，消耗很多时间做没必要的工作。若大量的查询是基于某个字段的，那么我们可以对数据进行分区处理。分区表是hive提供的一个特性，利用partition关键字可以把现成的数据表分成多个分区（文件夹）存储分区将表的数据在物理上分成不同的文件夹，以便于在查询时可以精准指定所要读取的分区目录，从而降低读取的数据量。

比如crew和movie的联系集role这张表，一个crew要么是actor，要么是director。而我们的查询中有很多基于角色来分类的，所以重建一张表partition_role，以role作为区分属性，执行命令insert overwrite table orc_database.role partition(role) select * from role，在物理上把数据库拆分成了两个文件存储

下面来比较性能

**查询人员中作为演员参演过电影的数量**

```sql
select count(*) from role where role='actor'
```

| | A | B |
|---|---|---|
| 1 | **original** | **Partition** |
| 2 | 2s 279ms | 1s 316ms |

**查询人员中成为演员的最多次数**

SQL

```sql
select count(*) times from role
where role='actor'
group by name
order by times desc
limit 1;
```

| | A | B |
|---|---|---|
| 1 | **original** | **Partition** |
| 2 | 7s 688ms | 4s 764ms |

可以看到分区之后，由不进行全表扫描，而是针对分区查询，所以效率有很大的提升。

在优化的过程中，由于有大量的按照时间查询电影信息，我尝试把movie按照年份分区，得到了大概40-50份的分区，执行语句**查询12月份发行的电影数量**

SQL

```sql
Select count(*) from movie where release_month=12
```

比较运行时间

| original | Partition |
|---|---|
| 4s 360ms | 4s 350ms |

发现并没有很大的提升，考虑如下两个原因：

- 数据量太小，hive适用于百万级别的数据，在本次实验中，电影的数量只有较少的两万多条，提升效果不显著
- 虽然分区在物理上把文件划分成了多个部分，但逻辑上却产生了很多小文件。也就是整个MapReduce启动的任务数要增加，这样并不利于整个查询速度的提升

## 2.2.4MapReduce优化

我们都知道hive底层默认使用的是MR引擎，无论是map还是reduce的数量都会极大地影响整个查询的速度，我们可以通过设置一些参数来调整

- 注意到hadoop默认分块大小是128M，对于比较大的数据来说，进行分块无疑可以使得map任务算力均衡，达到比较好的效果；但若一个任务中有很多小文件，如10M，20M等等，这样一个小文件也会被当作一个map任务来执行，而一个map任务启动和初始化的时间远远大于逻辑处理的时间，就会造成很大的资源浪费。所以我们应该合理设置分块的大小来使得资源合理分配。
- Reduce的个数对整个作业的运行性能有很大影响。如果Reduce设置的过大，那么将会产生很多小文件，对NameNode会产生一定的影响，而且整个作业的运行时间未必会减少；如果Reduce设置的过小，那么单个Reduce处理的数据将会加大，很可能会引起OOM异常。实验过程中发现**部分任务**中不管数据量多大，不管有没有调整reduce个数的参数，任务中一直都只有一个reduce任务。经过探究发现，分为以下三种情况
  - 没有group by的汇总，比如下面两种sql语句执行起来是不一样的

SQL

```sql
select asin,count(1) from movie where asin = '2012-07-04' group by asin;'

select count(1) from movie where asin = '2012-07-04';
```

  - 使用了Order by进行排序
  - sql中包含笛卡尔积。

在设置reduce个数的时候需要考虑这两个原则：使大数据量利用合适的reduce数；是单个reduce任务处理合适的数据量；

以下是我为了优化尝试修改的部分参数

| | A | B |
|---|---|---|
| 1 | **描述** | **参数设置** |
| 2 | 设置分块大小 | set dfs.block.size=16M |
| 3 | 设置每个Map最大输入大小 | set mapred.max.split.size=16777216 |
| 4 | 开启并发执行 | set hive.exec.parallel=true |
| 5 | 设置并发线程数 | set hive.exec.parallel.thread.number=16 |
| 6 | 设置map的任务数 | set mapred.map.tasks=4 |
| 7 | 设置reduce的任务数 | set mapred.reduce.tasks=4 |
| 8 | 设置每个reduce任务处理的数据量 | set hive.exec.reducers.bytes.per.reducer=128M |
| 9 | 设置每个任务最大的reduce数 | set hive.exec.reducers.max=999 |

优化效果并不显著，推测是本次任务的数量太小

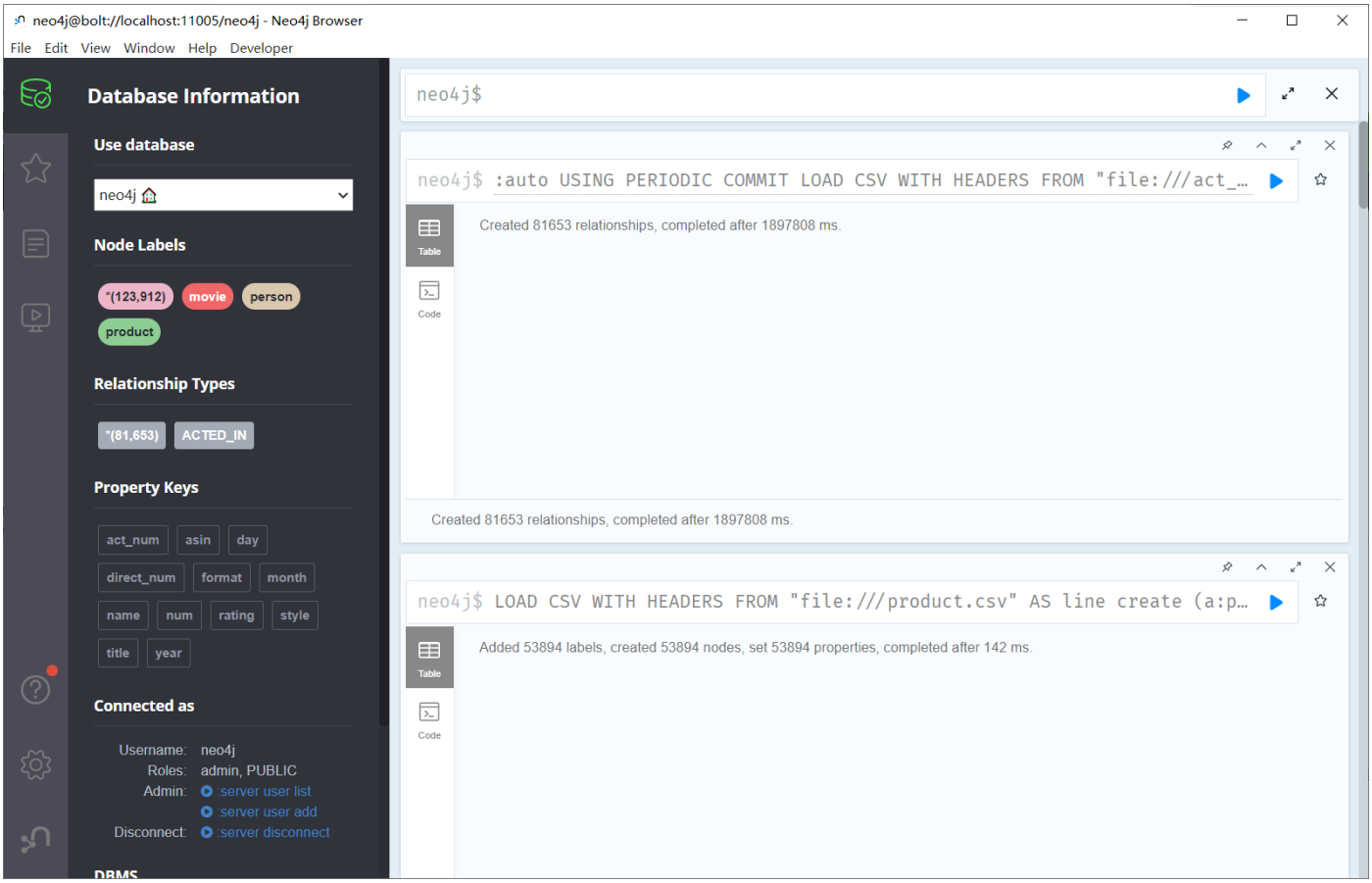
## 2.3 Neo4j优化

对于图数据库，较为重要和有效的查询优化方法是建立索引， 对于movie的结点，以asin建立索引，对于person结点，以name建立索引：

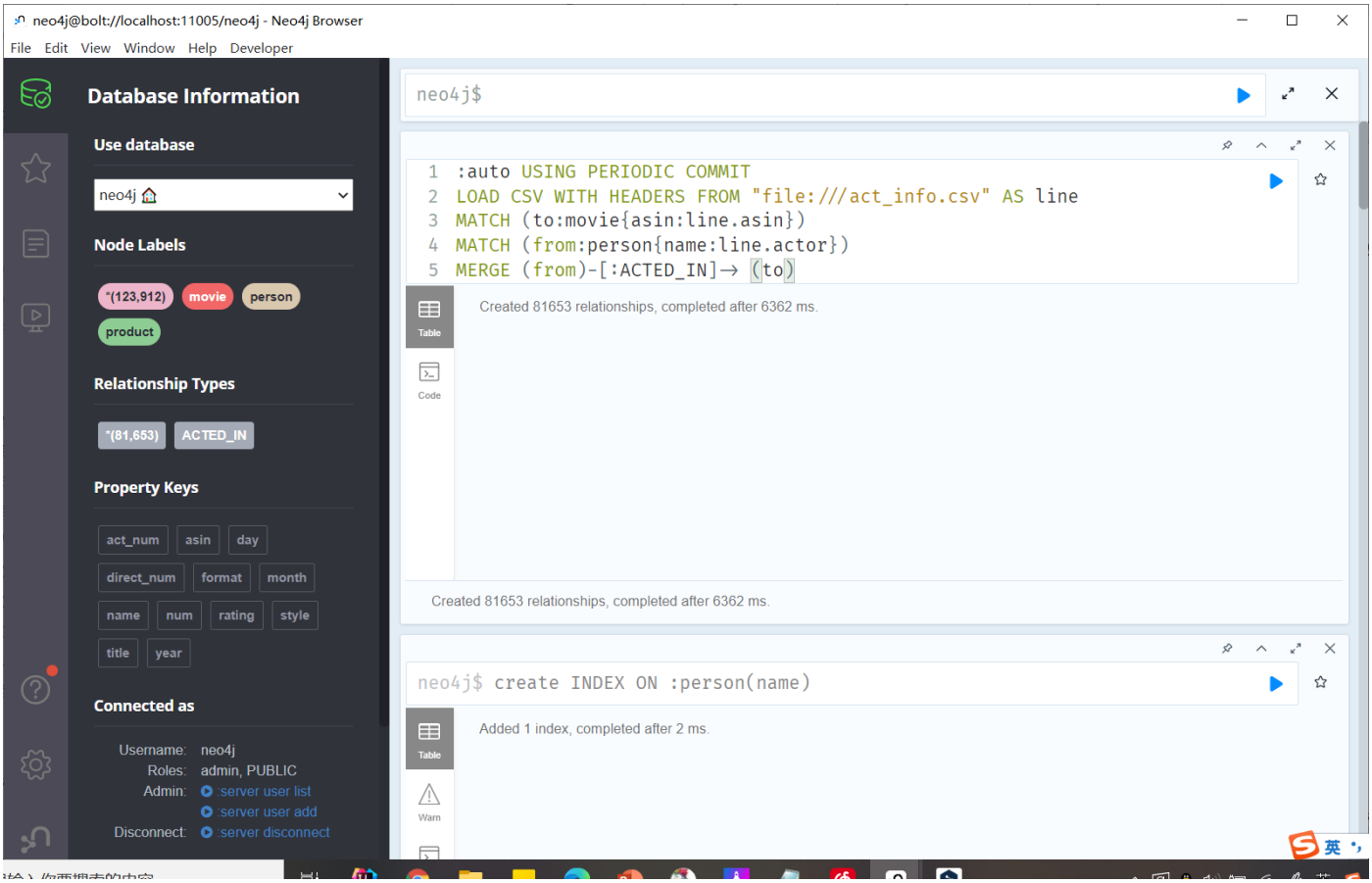
CREATE INDEX ON :movie(asin)

CREATE INDEX ON :person(name)

在做person和movie的联系ACTED_IN的建立时，需要匹配符合要求的演员和电影，如果不添加索引，即相当于需要对人员和电影整张表进行笛卡尔积，效率显然会很慢，需要花费1897808ms，即接近20分钟才能完成这个工作：




建立索引优化后，查询并建立联系的效率显著提升，只需要6362ms就可以完成这项工作：




对于这种两个万级数据的匹配工作，使用索引进行查询优化使效率提高了近300倍，可以看到使用索引对较大数据量查询工作的优化效果是十分明显的。对于不同的查询任务，建立所需要的的对应索引即可。

# 3. 数据质量

在实际的项目中，有些环节可能导致数据质量的降低：

（1）数据源方面，亚马逊有些页面上有部分信息是混乱的，有可能缺失部分字段，电影名有可能不够精确，演员名字间的分隔符在不同页面可能有不同

（2）数据处理方面，可能存在导演和演员同名的情况，对于电影和TV的筛选条件不同也可能导致误筛和漏筛

对于这些因素，我们采取了一些措施来保证数据质量：

（1）部分商品的Director和Actor可能为*或Various，将这些处理成空值

（2）对于导演和演员同名的情况，将该人员既设为导演也设为演员

（3）使用较为合理的TV筛选条件

...

# 4. 数据血缘

数据流：

网页—>jsonlines文件—>csv文件

Movie.csv < movie_information.json

User.csv < movie_information.json

Review.csv < movie_information.json

Director.csv < movie_information.json

Actor.csv < movie_information.json

## Mysql

**movie.csv < Movie.csv**

movie_ID < Movie.asin

format<Movie.format

title<Movie.title

otherformat_num<Movie. otherformat_count

star_rating<Movie. star_rating

release_year<Movie.release_time

release_month<Movie.release_time

release_day<Movie.release_time

style<Movie.style

**director.csv<Director.csv Movie.csv**

director_name < Director.director_name

movie_num< Director.csv Movie.csv

**actor.csv<Actor.csv**

actor_name < Actor.actor_name

movie_num< Actor.csv Movie.csv

**user.csv<User.csv**

user_ID<User.userID

user_name<User.user_name

**actor_actor.csv< Actor.csv**

actor_name1 < Actor.actor_name

actor_name2 < Actor.actor_name

number< Actor.csv Movie.csv

**director_actor.csv< Actor.csv Director.csv Movie.csv**

actor_name < Actor.actor_name

director_name < Director.director_name

number< Actor.csv Movie.csv

**director_director.csv< Director.csv Movie.csv**

director_name1 < Director.director_name

director_name2 < Director.director_name

number< Actor.csv Movie.csv

**user_movie.csv< User.csv Movie.csv Review.csv**

user_ID<User.userID

movie_ID < Movie.asin

content<Review.review_summary

rating<Review.rating

status<Review.status

commentTime<Review.review_date

**actor_movie.csv< Actor.csv Movie.csv**

actor_name < Actor.actor_name

number< Actor.csv Movie.csv

**director_movie.csv< Director.csv Movie.csv**

director_name < Director.director_name

number< Actor.csv Movie.csv

## Hive:

**movie_info.csv < Movie.csv**

asin < Movie.asin

title < Movie.title

rating < Movie.star_rating

other_format < Movie.other_format

additional_format<Movie.additional_format

year < Movie.release_time

month < Movie.release_time

day < Movie.release_time

season<Movie.release_time

style < Movie.style

**user_info.csv < User.csv**

id<User.id

name<User.name

**all_all.csv < Movie.csv Director.csv Actor.csv**

asin<Movie.asin

crew1<Director.name Actor.name

crew2<Director.name Actor.name

**crew_info.csv<Crew.csv**

name<Crew.name

**review.csv<User.csv Movie.csv Review.csv**

Name<User.name

asin<Movie.asin

rating<Review.rating

summary<Review.summary

text<Review.text

helpness<Review.helpness

time<Review.time

## Neo4j：

**movie_info.csv < Movie.csv**

asin < Movie.asin

title < Movie.title

rating < Movie.star_rating

format < Movie.format

year < Movie.release_time

month < Movie.release_time

day < Movie.release_time

style < Movie.style

:Label =movie

**product_info.csv < Movie.csv**

asin < Movie.other_format

:Label =product

**person_info.csv < Movies.csv Actor.csv Director.csv**

name < Director.name Actor.name

act_num < Movie.actor （统计数量）

direct_num < Movie.director （统计数量）

:Label =person

**Include.csv < Movies.csv**

asin < Movie.asin

:Type =INCLUDE

**act_info.csv < Movies.csv Person.csv**

asin < Movie.asin

name < Actor.name Movie.actor

:Type =ACTED_IN

**direct_info.csv < Movies.csv Person.csv**

asin < Movie.asin

name < Director.name Movie.director

:Type =DIRECT_TO

**person_co.csv < Movies.csv Person.csv**

person1 < Director.name Actor.name

person2 < Director.name Actor.name

count < Director.name Actor.name Movie.actor Movie.director

:Type =COOPERATE