数据仓库项目报告

1 数据存储及优化对比

1.1 关系型数据库

1.1.1 适用查询范围

关系型数据库在处理需要高度结构化数据和支持复杂查询的场景时表现良好。适用于需要数据一致性、完整性和事务处理的应用。

- 关系型数据库基于表格模型，数据存储在由多个表格组成的关系中，每个表格包含多行数据和多个列（即属性），每个属性只有一种数据类型。采用关系代数等数学概念和方法来处理数据，使用结构化查询语言(SQL)进行查询和操作，所以适用于存储和检索结构化数据，但是需要定义好表和字段并保证数据结构固定且不经常发生变化。在本项目中，由于需要进行存储优化，在数据库结构进行了一些调整，比如进行添加冗余字段等，导致需要对数据进行更新或重新录入，在一些数据库表的操作上遇到了相当多的问题。
- 关系型数据库擅长应对复杂的关系和连接，允许在不同表之间建立关系，以执行联接操作；易于对数据进行分组和计算；还支持嵌套查询，对于在查询中使用子查询来获取特定条件下的数据非常有用；同时支持索引，通过在表上创建索引，可以提高查询性能，尤其是对于大型数据集
- 关系型数据库提供事务支持用于保证ACID属性，强调数据的完整性和一致性，数据结构清晰、易于理解、容易维护，适用于需要复杂查询、事务处理和保障保证数据一致性的场景，广泛应用于企业信息管理、金融系统、人力资源管理、学术研究等领域。

1.1.2 存储优化工作

在本项目中，我们对物理模型进行反规范化，并通过建立索引等操作对存储进行优化

1. 星型模型使用

本项目采用星型模型结构，对数据存储有一定的冗余。首先建立一个movie表，然后将movie的属性都保存在事实表中，然后将演员，上映时间等信息保存到其他维度表中。

1. 字段设置
    1. 在review表中，time字段记录了该评论的时间戳，我们并没有采用`datetime`等和时间有关的数据类型来存储时间戳，而是采用了`bigint`类型的数据来存储。使用bigint类型存储时间戳可以确保跨不同数据库和编程语言的一致性；相对于 `datetime` 类型，`bigint` 类型通常需要较少的存储空间，这对于像review表这样的行数很多的表来说，可以节省相当的存储空间。在某些情况下，使用 `bigint` 类型进行日期和时间的比较可能比使用 `datetime` 类型更高效。这是因为整数的比较操作通常比日期时间类型的比较要快。
    2. 在movie表中，我们使用`float`类型的数据来存储movie_score等和评分相关的数据。`float` 类型的存储空间通常小于 `double` 类型，而电影评分并不需要较高的精度，因此使用 `float` 可以减少存储空间的消耗。
    3. 在release_date表中，我们使用varchar类型的数据来存储year, month, day, weekday等字段。这允许存储非标准日期格式或混合格式数据直接以字符串形式存储日期时间等字段，也可以避免不同应用程序间日期格式解析的兼容性问题。对于这些字段，我们也建立了索引，加速了查询。
2. 冗余存储
    1. 在电影表中存储相关评论数，在查询最受欢迎的演员组合时，可以避免与电影评论表进行join操作，由于电影评论表数据量巨大，所以会节省大量的时间
    2. 将时间单独抽象成一张表，将时间戳存储为不同数据格式，便于查询时直接进行匹配而不用处理数据，节省查询时间
    3. 我们对于评论的情感进行了两次优化，第一次将评论的情感单独抽象为一张表，避免根据情感进行查询时与原来的评论表进行join操作，但是由于是百万数量级，直接在电影表中增加一个字段，表示该电影中对应正面评价的比例。
3. 建立索引

除了主键和外键等数据库自动建立的索引外，对查询操作中需要用到的字段建立单独的单列索引和组合索引，便于快速进行查询操作

- 演员表中为演员姓名建立单列索引
- 导演表中为导演姓名建立单列索引
- 风格表中为电影风格建立单列索引
- 时间表中为年份建立单列索引，为年和月建立组合索引，为年和季度建立组合索引，为工作日建立单列索引
- 版本表中为电影语言建立单列索引，为电影格式建立单列索引

1. 建立视图
    1. 建立演员和演员之间合作的视图，字段包括演员1的id、演员2的id以及合作的电影id
    2. 建立演员和导演之间合作的视图，字段包括演员id、导演id以及合作的电影id

1.1.3 优化前后对比

在对评论的情感得分进行优化时，我们第一次选择将评论的情感得分单独抽象为一张表，在使用关系型数据库进行查询时，统计对应 `movie_id` 的所有评论情感得分中大于0.2（我们设定的正面情感阈值）的比例，由于评论数量是百万数量级，最后的查询耗时较长（选择所有标题包含Harry的电影查看其信息，共需要56.938s）：


```
        },
        {
            "movieId": "6303874606",
            "movieTitle": "Harry & Son ",
            "movieScore": 0.0,
            "runtime": "1 hour and 57 minutes",
            "date": null,
            "directors": null,
            "actors": null,
            "format": null,
            "edition": null,
            "genre": "Drama",
            "positive": "100.00%"
        }
    ],
    "consuming_time": 56.938
}
```


经过我们的优化，使用冗余存储，直接在电影表中增加一个字段 `review_sentiment_rate`，查询同样的电影信息，只需要19.546s即可。

```
            "positive": "40.00%"
        },
        {
            "movieId": "6303874606",
            "movieTitle": "Harry & Son ",
            "movieScore": 0.0,
            "runtime": "1 hour and 57 minutes",
            "date": null,
            "directors": null,
            "actors": null,
            "format": null,
            "edition": null,
            "genre": "Drama",
            "positive": "100.00%"
        }
    ],
    "consuming_time": 19.546
}
```


1.2 分布式数据库

1.2.1 适用查询范围

分布式数据库擅长于处理大规模数据和高并发访问，适用于需要横向扩展和高可用性的场景，包括大型互联网应用、大数据处理、分布式日志存储等

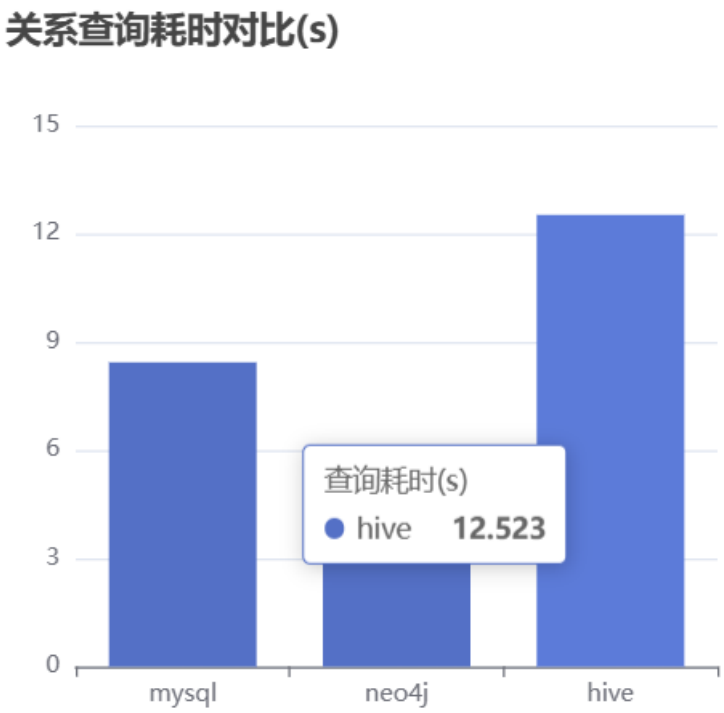
- 具有扩容需求。分布式数据库支持水平扩展，允许数据分布在多个节点上。查询可以同时在多个节点上执行，从而提高系统的吞吐量和性能。特别是当数据库分库后，数据通常位于不同的节点，使用关系型数据库可以高效进行数据查询。并且分布式数据库可以处理分布式事务，允许在多个节点上执行的操作保持一致性，确保数据的一致性和完整性。
- 考虑负载均衡。分布式数据库通过负载均衡来分配查询负载，确保每个节点都在其容量范围内运行，同时可以优化查询执行计划，以充分利用分布式环境中多个节点的计算资源
- 性能需求严格。分布式数据库需要支持在不同节点上创建和维护索引，这有助于减少数据的传输并提高查询效率。为了加速查询性能，分布式数据库需要支持在不同节点上创建和维护索引。这有助于减少数据的传输并提高查询效率。
- 容错要求较高。分布式数据库需要具备容错和高可用性的特性，确保在节点故障或网络问题的情况下系统能够继续运行。这通常涉及数据备份、故障转移等机制
- 具有扩容需求。分布式的数据存储在不同的物理设备中，需要增加数据时，可以通过添加设备来横向拓展，解决容量问题

1.2.2 存储优化工作

1. 将关系型数据库中的视图存储为实际的表，这样不仅可以复用之前的代码，还可以节省演员和演员以及演员和导演之间的join操作，加快查询速度
2. 建立与关系型数据库相同的数据库表，通过冗余字段避免join操作，并建立了相同的索引以提高查询效率
3. 建立外部表用于管理存储在数据库外部的数据，数据库中只存储元数据，而将实际数据存储到文件系统中，这种方式允许直接在数据所在位置进行查询处理，减少了数据移动的需要，避免其数据移动成为查询瓶颈，从而提升查询效率。

1.2.3 优化前后对比

若不将视图建立为实际的表，则在分布式数据库中会出现更多的join操作，拖慢了查询的速度，在将视图建立为实际表后，速度加快了约5秒，下图是加快后的结果，查询时间为12.523s.



1.3 图数据库

1.3.1 适用查询范围

图数据库专注于处理图形结构数据，擅长处理节点和边缘之间的关系，提供灵活的设计模式和高效的关联查询，适用于复杂的关系型数据，善于处理大量的、复杂的、互联的、多变的网状数据。支持数据的实时增删查改，保证ACID事务性，同时提供可视化、高可用、备份恢复等功能。

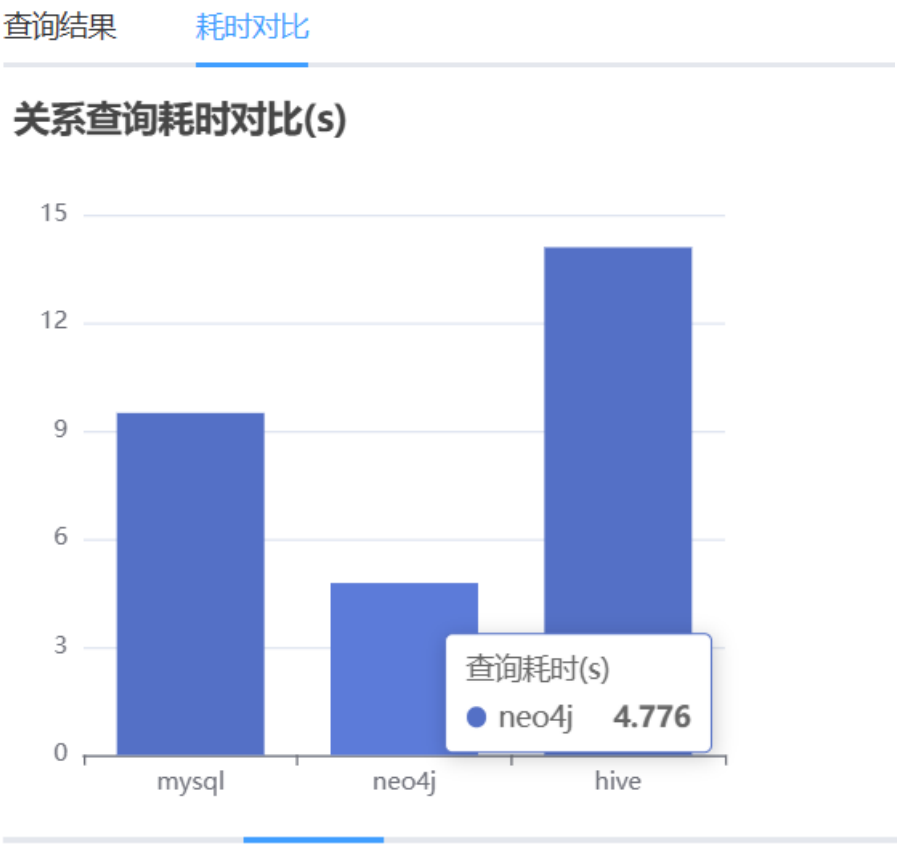
- 复杂关系查询。图数据库擅长处理复杂的关系，可以轻松查找节点之间的关系，例如查找节点的邻居或查找共同的连接点。在本项目中，对演员和导演之间的关系进行查询时，使用关系型数据库不仅查询语句逻辑较为复杂（特别是查询两个人以上的组合时），并且运行效率较低，而使用图数据库进行查询即可简单且高效的获得不同节点之间的关系。
- 路径发现。图数据库支持查询两个节点之间的最短路径或所有可能的路径，并且内置了各种图算法，可以用于分析图中的重要节点、发现社区结构，对于路由规划、网络分析和依赖关系分析非常有用。
- 递归查询。图数据库支持递归查询，可以轻松地查找节点到其自身的路径，适用于存储和查询具有复杂关系模式的数据，这些关系模式难以用传统的关系型数据库模型表示，这对于查找包含自我引用关系的数据非常有用。

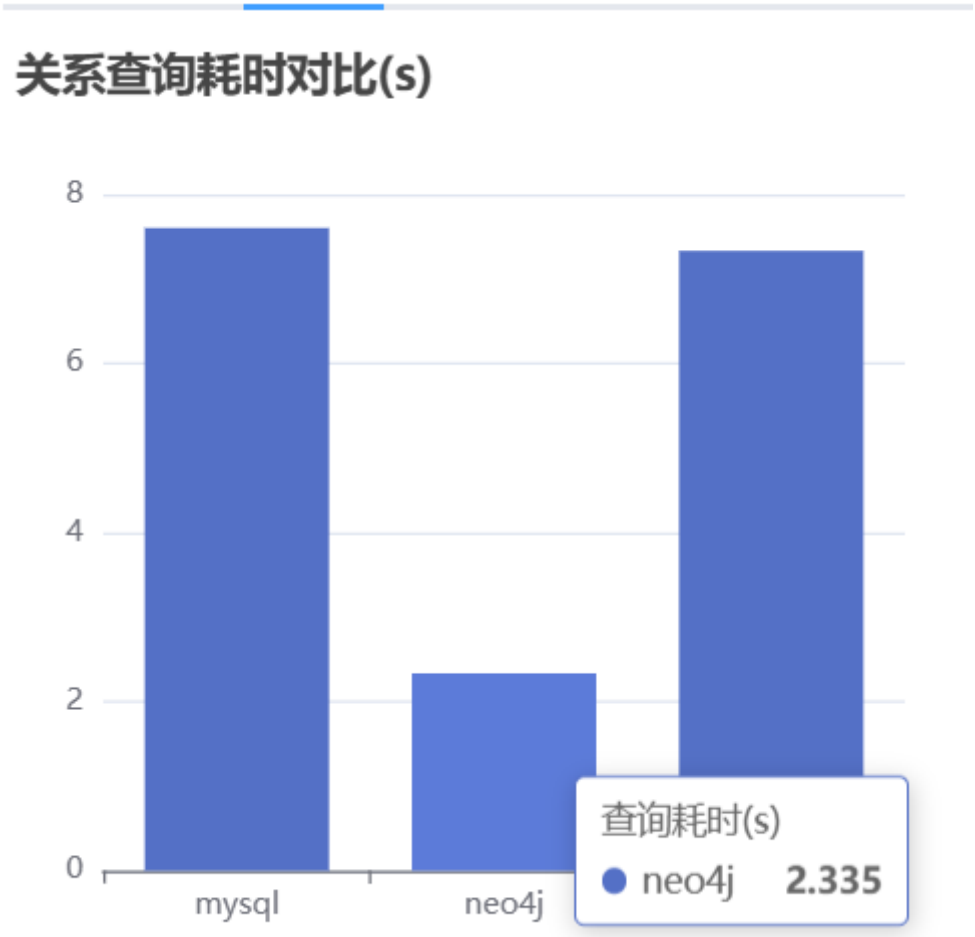
1.3.2 存储优化工作

1. 存储字段选择
    1. 在数据库中只存储三类节点(演员、导演、电影)以及两种关系(导演与电影间的执导关系和演员与电影之间的参演关系)，便于查询时快速进行计算
    2. 在数据库中电影节点中存储电影相关的评论数，便于统计
2. 建立索引
    1. 对查询时三类节点中将会用到的字段建立索引，分别是演员名、演员名、电影名、电影风格

1.3.3 优化前后对比

建立图数据库索引可以加速图数据库关系的查询，左图是添加索引之前，查询演员之间的关系耗时为4.776秒，右图是添加索引之后，查询演员之间的关系耗时2.335秒。





2 数据质量保证

2.1 保证数据质量方法

1. 数据爬取
    1. 使用爬虫获取数据时，使用selenium模拟浏览器操作，并使用amazoncaptcha库处理弹出的验证码进行反爬操作，爬取一定数量后更换ip地址(尝试建立代理池但是网上爬取的免费代理质量较差，所以手动切换代理节点)，并尝试更换cookie。组员之间使用各自不同的方式爬取数据，相互补充，并保留一份完整的网页数据，避免信息遗漏。
    2. 爬取时只爬取movies&TV的数据，后续再筛掉TV数据
    3. 对爬取到的404网页进行标记
    4. 从网页中抽取数据时采用xpath，并且避免由于网页结构变化导致抽取数据失效
2. 数据清洗
    1. 将组员之间的数据按ANSI进行merge操作，整理一份完整数据，避免数据遗漏
    2. 去除明显错误数据，包括除了电影id以外皆为空的数据等。
    3. 去除人名中的错误字符(比如双引号)
    4. 去除数据中不是movie的数据。首先对电影评价集进行处理，保留包含关键词film/movie的电影，然后去除电影名称中包含season/episode的数据
    5. 使用评论数据集中最早的评论时间填充缺失的部分电影上映时间
    6. 使用模糊匹配统一人名的不同表示形式，比如同一个姓名的缩写与全名，大小写等，统一为全名，优先呈现大写形式
    7. 对电影进行去重(电影名和导演相同即认为是同一部电影)时，保留重复组中的第一项，并用将要被去除的重复数据填充第一项中的空白数据

2.2 数据质量影响因素

1. 由于网页结构不同或者关键信息命名不同导致爬取信息不完整或者错误
2. 由于是从单一数据源爬取数据，该电影的相关信息本就是不完整的，然后我们并没有使用第三方数据源进行补充(尝试使用豆瓣和TMDB的接口，但是对请求数有限制，接口也没有完全符合要求的，如果重新爬取过于麻烦)，也会影响数据质量
3. 由于数据库本身设计可能存在缺陷，导致数据录入时无法校验或者校验不准确，导致数据重复或者不完整
4. 由于Amazon采取的反爬虫措施阻碍了爬虫正常工作，影响数据的完整性和准确性。我们已经解决了验证码，但是在爬取的过程中发现验证码解决后的网页，detail部分的数据会被隐藏，导致重新爬取了近10条数据。
5. 由于网络连接不稳定或中断可能导致数据爬取中断或部分数据丢失
6. 由于不同数据库有不同的数据模型和约束，所以需要进行转换和验证，在这个过程中也会可能影响数据质量
7. 由于数据清洗、转换、装载过程中，逻辑混乱导致数据出现丢失、重复、失真等质量问题。本项目由于进行了一次较大的数据库更新，重新处理了数据，过程中还涉及团队的相互合作，难以避免地出现一些对于数据的处理偏差，可能导致数据丢失或者重复

3 数据血缘使用场景

1. 数据血缘是一种记录和追踪数据流动路径的技术，用于了解数据在整个信息系统中的传递、转换和处理过程。
    1. 数据质量管理： 数据血缘可以用于监测和评估数据的质量。通过追踪数据流，可以识别数据质量问题的根本原因，快速定位数据质量问题的发生地点，以便进行及时修复和改进。

2. 合规性和审计： 在需要遵循法规和标准的行业，数据血缘可以用于跟踪敏感数据的处理路径，确保数据的合规性。此外，审计人员可以使用数据血缘信息验证数据的准确性和完整性，以确保数据的可信度。
3. 数据治理： 数据血缘是数据治理的关键组成部分。通过了解数据如何被创建、修改和使用，组织可以更好地管理其数据资产。数据血缘信息有助于制定数据治理政策、标准和规程，并确保其执行。
4. 故障排查和性能优化： 在数据处理管道中发生故障或性能下降时，数据血缘可以帮助诊断问题。通过查看数据流的路径，可以迅速确定导致故障或性能问题的环节，并进行有效的排查和优化。
5. 变更管理： 当数据架构或处理逻辑发生变更时，数据血缘可以追踪这些变更的影响。这有助于团队了解变更对数据流程的影响，减少潜在的风险和错误。
6. 数据资产管理： 组织通常拥有大量的数据资产，包括数据集、报表和分析模型等。数据血缘可以用于跟踪这些数据资产的使用情况，了解哪些资产是关键的，哪些是过时的，以便进行有效的管理和优化。
7. 数据安全： 在安全方面，数据血缘可以追踪敏感数据的流动路径，帮助组织了解数据的暴露风险。通过监控数据流动，可以及时发现潜在的安全漏洞和威胁。
8. 业务影响分析： 当发生数据变更或系统升级时，了解数据血缘可以帮助业务部门了解这些变更对业务流程的影响。这对于规划和实施变更管理非常重要。

2. 在本项目中，我们使用一张与movie表相同的表格记录数据血缘，但是除了movie_id列外，每个单元格存储movie表中该位置的数据来自哪个movie_id。

movie_id	movie_title	Language	Release date	Rated	Run time	movie_actor	movie_director	Media Format	Genres	Customer Revie	IMDb	main_actor	edition	review_num
B004ZWHZUC	transparent	English	22-Sep-11	NR	1h		Jules Rosskam		LGBTQ,Docume	4	7.4			2
B004ZWHT3U	transparent	English	22-Sep-11	NR	1h	Josh Moltane	Jules Rosskam		LGBTQ,Docume	4	7.4	Josh Moltane		2
B0095D5454	Red Dawn	English Dialogu	7-Jan-99	PG-13	1 h 53 min	Patrick Swayze,	John Milius		Adventure,Action,Malicious,Ten		6.3	Patrick Swayze, C Thomas How		158
B005QG2DGC	Red Dawn	English Dialogu	7-Jan-99	PG-13	1 h 53 min	Patrick Swayze,	John Milius		Adventure,Actic	4.7	6.3	Patrick Swayze, C Thomas How		158
B0034ZDUK0	Paradise Expres	English	22-May-10	ALL	53min	Grant Withers [	#		Comedy,Drama,Adventure,Rom		5.7	Grant Withers Dorothy Appleby		1
B0034ZJM2A	Paradise Expres	English	22-May-10	ALL	53min	Grant Withers [	#		Comedy,Drama	4.1	5.7	Grant Withers Dorothy Appleby		1
B001CMZJUG	The Hollywood	English	16-Jun-11	ALL	60min		Gene Feldman		Documentary,S	4.4	7.6			1
B001CMTJQQ	The Hollywood	English	16-Jun-11	ALL	60min	Rupert Allan, R	Gene Feldman		Special Interest	4.4	7.6			1
B003DEQQTO	Turned Out: Sexual Assault Beh		4-Mar-09	18+	56min		MVD		Documentary,S	3.5	6.8			12
B003DEMO0Y	Turned Out: Sexual Assault Beh		4-Mar-09	18+	56min		MVD		Documentary,S	3.5	6.8	David Mendenhall Jr., Danny Tr		12
B003BUZG9Q	Zombie Atrocity SPECIAL EDITIC		16-Mar-10	18+	1 h 33 min		#		Comedy,Horror	4.5	7.4			2
B003BUVGYK	Zombie Atrocity SPECIAL EDITIC		16-Mar-10	18+	1 h 33 min	Jeffrey S. Broml	#		Horror,Comedy	4.5	7.4	Jeffrey S. Bromley, Laurie Becke		2
B002VD1NAA	The Bros		20-Feb-07	R	1 h 29 min		Jonathan Figg		Comedy,Drama	5	2.8			5
B002VD1NB4	The Bros		20-Feb-07	R	1 h 29 min	John Tindall, Jo	Jonathan Figg		Music Videos a	5	2.8	John Tindall, Joachim Wiese, Jo		5
B000NW3IEK	24	English	17-Feb-09	TV-14			Jon Caesar		Suspense,Dram	4.7	8.4			3
B000HW9V5Q	24	English	21-Jan-12	TV-14		Keifer Sutherlar	Jon Caesar		Suspense,Dram	4.8	8.4	Keifer Sutherland, Kim Raver, M		3

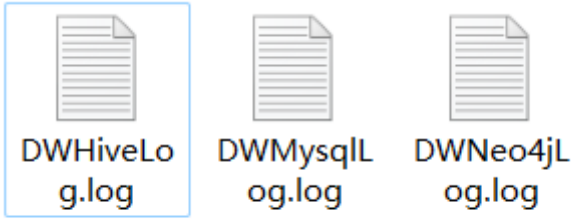
上表去重的同时会得到下面这样一张映射表，movie_id、movie_title、movie_director列的数据与去重后的数据保持不变，但是其余列记录了对应位置的数据来自于原来的哪一行，如果该位置的数据不是通过填充得到的，则无需记录对应关系。（将重复数据都挪到了一起便于观察比较）

movie_id	movie_title	Language	Release date	Rated	Run time	main_actor	movie_director	Media Format	Genres	Customer Revie	IMDb	movie_actor	edition	review_num
B004ZWHZUC	transparent					B004ZWHT3U	Jules Rosskam					B004ZWHT3U		
B0095D5454	Red Dawn						John Milius			B005QG2DGC				
B0034ZDUK0	Paradise Express						#			B0034ZJM2A				
B003DEQQTO	Turned Out: Sexual Assault Behind Bars					B003DEMO0Y	MVD							
B003BUZG9Q	Zombie Atrocity SPECIAL EDITION - The Italian Zombie Movie Part 2					B003BUVGYK	#					B003BUVGYK		
B002VD1NAA	The Bros					B002VD1NB4	Jonathan Figg					B002VD1NB4		
B000NW3IEK	24					B000HW9V5Q	Jon Caesar					B000HW9V5Q		

4 日志记录

记录日志是计算机系统中的关键实践，主要用于故障排查、性能监测和安全审计。通过捕获关键事件和异常情况，日志有助于追踪问题、提高系统性能，并支持安全审计和合规性需求。此外，日志也在业务分析、版本跟踪和用户支持等方面发挥作用，为系统维护和优化提供重要信息。

本项目中，我们将对于三个数据库的操作以日志的形式记录到三个不同的日志文件中，便于进行问题定位等操作。3个日志文件及部分日志内容如下：



```
wrapping com.mysql.cj.jdbc.ConnectionImpl@1fb310c2] will not be managed by Spring
2023-12-30 23:50:17.716 [http-nio-5000-exec-9] DEBUG c.m.d.m.V.selectFormatByMovieId - ==>  Preparing: select DISTINCT(format) from version
  where movie_id=?
2023-12-30 23:50:17.716 [http-nio-5000-exec-9] DEBUG c.m.d.m.V.selectFormatByMovieId - ==> Parameters: B000HKY9W8(String)
2023-12-30 23:50:17.736 [http-nio-5000-exec-9] DEBUG c.m.d.m.V.selectFormatByMovieId - <==      Total: 1
2023-12-30 23:50:17.736 [http-nio-5000-exec-9] DEBUG org.mybatis.spring.SqlSessionUtils - Closing non transactional SqlSession [org.apache
 .ibatis.session.defaults.DefaultSqlSession@746eee2e]
2023-12-30 23:50:17.737 [http-nio-5000-exec-9] DEBUG org.mybatis.spring.SqlSessionUtils - Creating a new SqlSession
2023-12-30 23:50:17.737 [http-nio-5000-exec-9] DEBUG org.mybatis.spring.SqlSessionUtils - SqlSession [org.apache.ibatis.session.defaults
 .DefaultSqlSession@70d0b9b7] was not registered for synchronization because synchronization is not active
2023-12-30 23:50:17.737 [http-nio-5000-exec-9] DEBUG o.s.jdbc.datasource.DataSourceUtils - Fetching JDBC Connection from DataSource
2023-12-30 23:50:17.737 [http-nio-5000-exec-9] DEBUG o.m.s.t.SpringManagedTransaction - JDBC Connection [HikariProxyConnection@1934370333
 wrapping com.mysql.cj.jdbc.ConnectionImpl@1fb310c2] will not be managed by Spring
2023-12-30 23:50:17.737 [http-nio-5000-exec-9] DEBUG c.m.d.m.V.selectFormatByMovieId - ==>  Preparing: select DISTINCT(format) from version
  where movie_id=?
2023-12-30 23:50:17.737 [http-nio-5000-exec-9] DEBUG c.m.d.m.V.selectFormatByMovieId - ==> Parameters: B000MTEFUA(String)
```

5 小组贡献度

学号	姓名	百分比
2153174	陈华机	25%
2152614	崔宸睿	25%
2152056	王宇轩	25%
2153273	陈嘉瑞	25%