文章编号:1671-4598(2018)04-0225-06 DOI:10.16526/j.cnki.11—4762/tp.2018.04.059 中图分类号:TP274 文献标识码:A

基于 Hadoop 平台的数据迁移方法研究实现

王 铭,田 茂,赵 鑫,金山城
(湖北大学 计算机与信息工程学院,武汉 430062)

摘要:随着计算机科学的发展和大数据时代的到来,应用系统已经出现了数据海量化、用户访问高量化的局面,使得企业应用系统的原有关系型数据库(RDBMS)面临承担更大负荷的压力,系统的高性能要求得不到有效满足,对于关系型数据库所面临的问题,Hadoop 平台中的 HBase 数据库可有效解决;以关系型数据库中 MySQL 数据库及 Hadoop 平台中分布式数据库 HBase 数据库为研究基础,应对企业应用数据海量化增长,提出从关系型数据库(MySQL 数据库)向分布式数据库(HBase 数据库)进行数据迁移的方法,并通过研究 HBase 数据库存储原理提出从 MySQL 到 HBase 的表模式转换原则实现高效数据查询性能的数据迁移方法;最后,将该方法与同类数据迁移工具 Sqoop 进行比较,证明该方法进行数据迁移的便捷性和在迁移后数据库中进行连接查询的高效性。

关键词:Hadoop;关系型数据库;HBase;数据迁移

Research and Realization of Data Migration Tool Based on Hadoop Platform

Wang Ming, Tian Mao, Zhao Xin, Jin Shancheng
(College of Hubei University, Wuhan 430062, China)

Abstract: With the development of computer science and the advent of big data era, the application system has appeared the situation of mass data and user access. As a result, the original relational database (RDBMS) of enterprise application system is under the burden of greater load, The system's high-performance requirements cannot be met effectively. For the problems faced by the relational database, the HBase database in the Hadoop platform can be effectively solved. Based on the research on the MySQL database in the relational database and the HBase database in the distributed database on the Hadoop platform, this paper proposes a method of data migration from the relational database (MySQL database) to the distributed database (HBase database) based on the massive growth of enterprise application data. And through learning HBase database storage principle, proposing the table mode conversion principle from MySQL to HBase which could achieve efficient data query performance on data migration. Finally, the method is compared with Sqoop, a similar data migration tool, to demonstrate the ease with which the method can be migrated and the efficiency of the connection queries in the migrated database.

Keywords: Hadoop; relational database; HBase; data migration

0 引言

随着计算机与信息技术的迅速发展,企业应用系统的规模迅速扩大,所产生的数据呈海量型变化。大数据(Big Data)一词越来越多地被提及,人们就用它来描述和定义信息爆炸时代产生的海量数据,并命名与之相关的技术发展与创新[1]。在大数据时代,企业应用系统数据量快速增长已达到 PB(1PB=1000TB)甚至 EB(1EB=1000PB)级别,而大多数企业应用系统之前所使用的数据库都是传统关系型数据库[2],实际上关系型数据库已经无法满足当前海量数据处理的高扩展性、高可用性、高并发读写、高吞吐量、较低延迟及高效存储和查询等需求。与关系型数据库不同,NoSQL 数据库[3]由于其扩展性和可用性强以及灵活的数据模型,在大数据时代得到广泛应用。

Hadoop 生态系统中的 HBase 数据库是一个开源的基于列的非关系型分布式数据库,是 Apache 软件基金会的 Hadoop 项目的一部分,不仅拥有大部分 NoSQL 数据库所拥有的特点,而且因为能与 Hadoop 大数据平台进行集成,所以它能在数据存储方面提供更为强大的扩展性和在数据操作方面提供更为完善的操作性能[4]。基于 HBase 的巨大优势,如今企业的应用系统都开始将 HBase 数据库作为进行大数据的存储和处理的工具。但是,在实际需求的限制下,企业应用系统在更改存储系统时不应该丢失原有系统中的历史数据,而是需要将原有系统中的历史数据迁移到新构建的存储系统中,所以如何将原有应用系统中的历史数据尽量完整、有策略、自动迁移到新构建的 HBase 数据库中是一个值得研究的课题。

HBase 本身是一种 NoSQL 数据库,由于其数据存储结构与传统关系型数据库的数据结构有很大的差异,因此在设计数据存储表结构及表关系时也有很大的不同,所以要实现从关系型数据库向分布式数据库 HBase 进行数据迁移,需要重新设计新构建的存储系统 HBase 的表模式,这个过程是个比较复杂的过程,在完成数据迁移过程后还需要考虑数据迁移后数据查询及存储的性能问题,因此实现一个从关系型数据库向分布式数据库 HBase 进行数据迁移的方法是非常有必要的。

本文在上述背景及需求下,通过对 Hadoop 大数据平台、HBase 数据库存储原理及典型数据库迁移策略的研究,设计基于 Hadoop 平台数据迁移方法来实现将传统关系型数据库中的历史数据以及表模式向 HBase 数据库中进行迁移和转换,有效解决之前同类迁移工具无法对关系型数据库中表模式进行迁

收稿日期:2018-02-01; **修回日期**:2018-02-26。
基金项目:湖北省教育厅科学技术研究计划青年人才项目(Q20161012)。
作者简介:王 铭(1991-),男,湖北黄冈人,硕士生,主要从事计算机应用系统方向的研究。
田 茂(1970-),男,湖北武汉人,副教授,硕士生导师,主要从事嵌入式系统设计和物联网技术方向的研究。

移不足的问题。

本文组织结构如下：第 1 部分简述相关理论基础；第 2 部分介绍数据迁移方法的设计思想及具体实现；第 3 部分对数据迁移方法实现后的查询与存储等方面与同类迁移工具进行比较；第 4 部分总结全文。

1 相关理论

1.1 Hadoop 平台

Hadoop[5]是一个由 Apache 基金会根据 Google 公司发表的 Google 文件系统（GFS）和 MapReduce 的论文自行实现而成的分布式系统架构。使用者可以在不了解底层细节的情况下，开发分布式程序，充分利用集群的作用进行高速运算和存储。Hadoop 框架最核心的设计就是由 HDFS（Hadoop Distributed File System：分布式文件系统）和 MapReduce（分布式计算框架）组成。

HDFS[6]在 Hadoop 系统中为海量数据提供了存储，为了高数据吞吐量而优化的，适合那些有超大数据集的应用程序，有高容错性、低成本的特点。一般情况下，HDFS 集群主要由一个 NameNode 节点（master）和多个 DataNode 节点（slave）组成，在集群节点通信中，NameNode 发挥着管理、协调、操控的作用，主要负责管理文件系统的命名空间，协助客户端对文件的访问，并对 DataNode 发起的请求进行响应；而 DataNode 是 HDFS 中最终存储数据的节点，负责自身及其他物理节点的存储管理。客户端向 HDFS 发起访问文件请求时，首先需要从 NameNode 节点上获取文件在 HDFS 中所处的位置信息，根据文件位置信息找到存储数据块所在的 DataNode，最后读取 DataNode 上存储的数据。

MapReduce[7]是为海量数据提供了计算。它是面向大数据并行处理的计算模型、框架和平台。在一个 MapReduce 计算任务过程中，主要有两个阶段：Map（映射）阶段和 Reduce（归约）阶段，每个阶段都是以键值对作为输入和输出，Map 阶段负责对输入文件进行切分处理，然后汇总再分组给 Reduce 进行处理，达到高效的分布式计算效率。

1.2 HBase 存储原理

HBase[8]是 Apache Hadoop 中的一个子项目，是 Google BigTable 的开源实现，依托于 Hadoop 的 HDFS 作为最基本存储基础单元，通过使用 Hadoop 的 DFS 工具就可以看到这些数据存储文件夹的结构，还可以通过 Map/Reduce 的框架对 HBase 进行操作。

HBase 是一个分布式、面向列的数据存储系统，与关系型数据库的模式固定，面向行的数据库具有 ACID 性质不同，HBase 是一个适合于非结构化数据存储的数据库，是介于 Map Entry（key&value）和 DB Row 之间的一种数据存储方式。HBase 中的数据是依靠行健（RowKey）、列族（ColumnFamily）、列标识（ColumnQualifier）、时间戳（TimeStamp）来标识一个单元格中的数据。表 1 是 HBase 的简单逻辑视图。

表 1 HBase 逻辑视图

RowKey	TimeStamp	orderInfo		
		username	phoneNum	orderStatus
1001	T1	Jack	13879730231	1
	T2	Mack	13672939749	1
1002	T1	Joe	13678293749	2

表 1 表示的是在 HBase 数据库中保存的两条订单信息 1001 和 1002，其中 1001 和 1002 是表示行键，orderInfo 是列族，在 orderInfo 列族中包含有 username、phoneNum、orderStatus 列标识。如要获取某一单元格中的订单信息，则可使用{RowKey，column（=<family>+<qualifier>），version}，如若不指定时间戳，则默认获取的是最新的一条记录。为了更好的理解 HBase 的数据模型，可以使用 JSON 格式来表示。HBase 中数据模型的 JSON 格式的一般形式[9]如下：

```
RowKey{
  ColumnFamily1{
  Column1：
  T2：value2
  T1：value1
  Column2：
  T1：value3
}
ColumnFamily2{
  T3：value4
}
}
```

由表 1 和 HBase 数据模型的 JSON 格式的一般形式可以看出 HBase 与 RDBMS 之间有很大的不同，其实 HBase 是基于列族的、稀疏的、多维的数据库系统，所以在设计数据迁移方法时需要充分考虑 HBase 数据模型的特点。

HBase 的底层存储结构如图 1 所示，是按列族存储的。

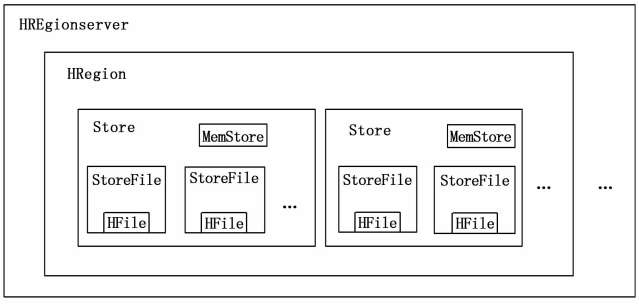

图 1 HBase 底层存储结构

某一 RowKey 的数据（如表 1）虽然在数据模型上看是在一行上，但是在 HBase 底层存储中是分开存储的，如图 1 所示，数据在存储时，HBase 会自动的将表水平划分成多个区域（Region），每个 Region 中会保存某段连续的数据，而一个 Region 包含多个 Store，这里每一个 Store 就是一个 ColumnFamily（列族），由此可以看出，HBase 在底层存储时，同一列族的数据存储在一个文件中，不同列族的数据存储在不同的文件之中，而且 HBase 中并不是所有的行和列中都存储数据，不存储数据的单元格不占存储空间，每一个 store 在初始化时设置了固定存储大小；另外，虽然 HBase 列标识可以有上百万列，由于 HBase 结构中的 HFile 有一定的大小限制，如果一行数据量太大，在存储时会导致 HFile 无法选择合适的分割点。

由于 HBase 底层存储有以上特点，在设计数据迁移方法的 HBase 表模式时应考虑无论数据量的大小，每个列族都占用固定的存储空间（初始化前确定）这一因素，所以应该尽可能少的创建列族，HBase 官方文档推荐是创建最好不多于三个

列族，故将相邻的数据、经常在一起使用的数据放在同一列族下，这样不仅可以减少数据存储空间而且可以有效减少检索时查询存储文件的次数。

综合考虑 HBase 数据模型及 HBase 数据存储的特点，在设计 HBase 的数据格式时需要充分利用行键（RowKey），适当的使用列族（ColumnFamily），将关系型数据库中需要索引的字段尽量放在 RowKey 或列族中来提高数据存储和查询的性能。

2　数据迁移方法的设计与实现

2.1　概述思想

由于 HBase 与关系型数据库的区别，所以在进行关系型数据库（MySQL）向非关系数据库（HBase）的数据迁移时，需要重新设计 HBase 的数据表模式。在设计 HBase 表模式时要充分考虑 HBase 数据的存储和查询特点，使得系统业务数据的存储性能和查询效率较高。

在数据存储上，由 1.2 节 HBase 存储原理可知，HBase 底层是按列族进行存储的，每个表中包含有多个区域，而一个区域中包含多个 store，这里的一个 store 就代表一个列族，每个 store 在初始化时给定固定值，由此可知，HBase 表中的列族越多就越占用空间，所以在设计 HBase 表时尽量按照官方文档建议列族最好不多于三个，故将相邻的、经常使用的数据放在一个列族中以减少存储空间和提高查询效率。

在数据查询上，由于 HBase 不支持表间关联查询[10]，所以在进行数据库迁移时需要充分考虑关系型数据库中的表间关系在 HBase 数据库中的体现，相对于关系型数据库，在 HBase 数据库中就需要对 HBase 表模式进行重新设计，提高 HBase 的查询效率，由此本文提出关于 MySQL 数据库到 HBase 数据库的几种表转换关系原则。

2.2　表模式设计

在对 HBase 的表模式进行设计时，以关系型数据库 MySQL 数据库的表间关系为基准，分为“基本变换”、“‘一对一’变换”、“‘一对多’变换”，“‘多对多’变换”四种变换，下面详细介绍其变换规则[11-14]。

2.2.1　基本变换

此类变换对应于 MySQL 数据库向 HBase 数据库进行单表迁移没有表间关系也即是基本表迁移的情况，表模式变换即用基本变换方法将 MySQL 数据库中表模式变换成符合 HBase 数据库的表模式。

具体变换方法说明：如图 2 所示，将 MySQL 数据库中 A 表的表名作为 HBase 数据库中对应表的表名 HA，在 HA 中创建表 A 的自定义列族 CF1，并将表 A 中的所有列作为 HA 表列族 CF1 中的列标识（ColumnQualifier）。最后以表 A 的主键作为 HA 的 RowKey 进行数据导入。

此类是数据库基本表的迁移，所以在海量数据的存储和查询的效率上与 HBase 的一般表的存储和查询效率一样。

2.2.2　“一对一”变换

此类变换对应于 MySQL 数据库中表间关系为“一对一”的关系类型的数据表迁移，表模式变换即将 MySQL 中两张表的记录转换成 HBase 中一张表的记录，存储两张相关联表的信息。

具体变换方法说明：如图 3 所示，在 MySQL 数据库中有

AID	AB	AC
A1	A1_AB	A1_AC
A2	A2_AB	A2_AC
A3	A3_AB	A3_AC

Transform

HBase：HA

RowKey	Timestamp	CF1	
		AB	AC
A1	TS1	A1_AB	A1_AC
A2	TS2	A2_AB	A2_AC
A3	TS3	A3_AB	A3_AC

图 2　基本变换

数据表 A 和 B，表 A 与表 B 的表间关系是“一对一”关系，现将表 A 的表名作为 HBase 数据表 HA 的表名，并创建表 A 和表 B 在 HA 中的列族 CF1 和 CF2，将表 A 中所有列添加到表 A 对应的列族 CF1 中，同时将与表 A 相关联的表 B 的所有列添加到相对应的列族 CF2 中，根据实际需求对表 B 进行保留或删除操作，即完成“一对一”关系变换。

MySQL：A

AID	AC	AD
A1	A1_AC	A1_AD
A2	A2_AC	A2_AD
A3	A3_AC	A3_AD

1　1

MySQL：B

BID	BC	BD	AID
B1	B1_BC	B1_BD	A1
B2	B2_BC	B2_BD	A2
B3	B3_BC	B3_BD	A3

Transform

HBase：HA

RowKey	Timestamp	CF1		CF2		
		AC	AD	BID	BC	BD
A1	TS1	A1_AC	A1_AD	B1	B1_BC	B1_BD
A2	TS2	A2_AC	A2_AD	B2	B2_BC	B2_BD
A3	TS3	A3_AC	A3_AD	B3	B3_BC	B3_BD

图 3　“一对一”变换

此类“一对一”变换，可以有效解决 HBase 中无法进行表间的连接查询问题，由“一对一”变换图可知，存储的数据看似冗余，消耗更多的存储空间，但是提高了表间数据的连接查询效率。

2.2.3　“一对多”变换

此类变换对应于 MySQL 数据库中表间关系为“一对多”的关系类型的数据表迁移，表模式变换即将 MySQL 中“一”的一端对应的多条关联表记录在 HBase 中的一条记录进行存储，而“多”的一端与“一对一”变换类似。

具体变换方法说明：如图 4 所示，在 MySQL 数据库中有表 A 和表 B，表 A 和表 B 的表间关系是“一对多”关系，现将表 A 的表名作为 HBase 中 HA 的表名，并创建表 A 的列族 CF1 以及关系表表 B 的列族 CF2，同时将表 A 的所有列添加到对应的 CF1 中，将与表 A 相关联的 N 条记录的表 B 的主键添加到列族 CF2 中；对于 MySQL 中表 B 类似于“一对一”变换，完成表 A 和表 B 两表的变换即完成“一对多”变换。

此类“一对多”变换，可以在 HBase 中通过行键查询可以得到表 A 的记录，并且可以由关联关系快速查询到表 A 对应表 B 的关联信息，由“一对多”变换图可知，存储的数据看似冗余，消耗了更多的存储空间，但是提高了表间数据的连接查询效率。

2.2.4　“多对多”变换

此类变换对应于 MySQL 数据库中表间关系为“多对多”的关系类型的数据表，而且在中间表中有本表自有属性列的情

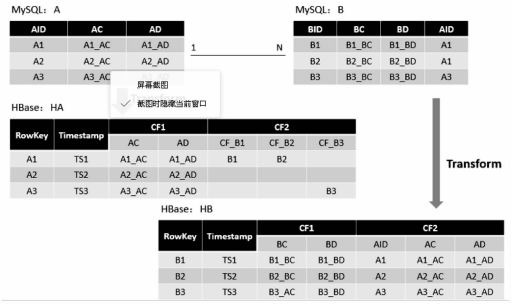

图 4 “一对多”变换

况，表模式转换即根据中间表信息，将 MySQL 中两张关联表对应的多条记录在 HBase 中一条记录中存储，同理将中间表中自有属性根据“一对多”或“多对多”表模式转换方式进行转换。

具体转换方法说明：如图 5 所示，在 MySQL 数据库中有表 A、表 B 以及表 C，其中表 A 和表 B 表间关系是“多对多”关系，表 A 和表 B 的中间表为表 C，表 C 中有本表自有属性。现将表 A 的表名作为 HBase 中 HA 的表名，并创建表 A 的列族 CF1、关联表表 B 的列族 CF2 以及中间表表 C 的列族 CF3，同时将表 A 中的所有列添加到列族 CF1 中，将与表 A 相关联的 N 条记录的表 B 的主键添加到 CF2 中，同理将中间表表 C 中与表 A 相关联的列添加到列族 CF3 中。对于表 B 的模式转换同表 A 一样，根据实际需求决定保存或删除表 C，即完成“多对多”变换。

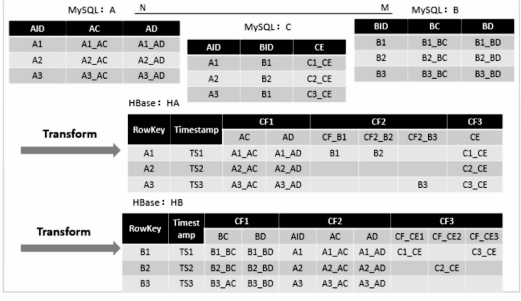

图 5 “多对多”变换

此类“多对多”变换，可以在 HBase 中根据行键查询快速得到表 A、表 B 中记录的关联信息，由以上变换图可知，存储的数据看似冗余，消耗了更多的存储空间，但是提高了表间数据的连接查询效率。

2.3 迁移方法及具体实现

2.3.1 迁移流程

在完成对 HBase 的表模式设计之后，需要了解数据迁移流程，本文在数据迁移过程中，利用了 XML 标记语言[15-16]对数据进行结构化处理的特点，采用 XML 文件作为存储源数据库表的元数据，由此对于基于 Hadoop 平台的数据迁移流程分为以下几个部分：

1）建立 HBase 数据库，并获得源数据库和目标 HBase 数据库的连接信息；

2）根据源数据库的连接信息，获取数据库表的元数据并保存在 XML 文件中；

3）根据 XML 文件信息，按照 2.2 节中表模式转换原则建立 HBase 数据库表模式；

4）将关系型数据库中的数据按照 2.2 节中表模式转换原则迁移至新建立的 HBase 数据表中。

整体数据迁移流程如图 6 所示。

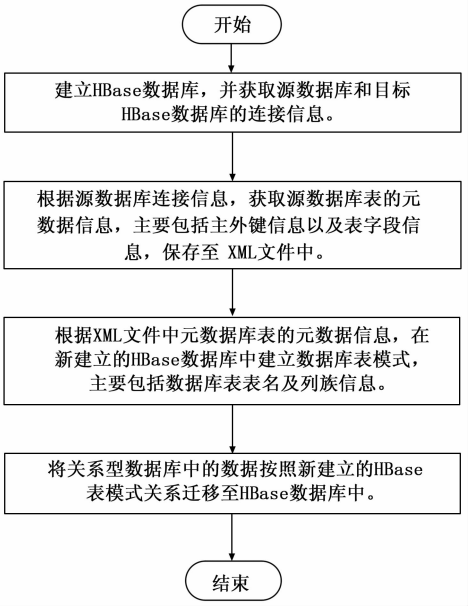

图 6 数据迁移流程

2.3.2 具体实现

在数据迁移的主要流程中，其中第二部分中需要创建源数据库表的元数据 XML 文件。在程序设计中通过以下步骤及方法获取数据库表元数据：

1）连接源数据库：public static Connection getConnectDB ();

2）获取源数据库的元数据：DatabaseMetaDatadbMetaData = ct. getMetaData ();

3）获取源数据库所有数据表：ResultSettablesResultSet = dbMetaData. getTables (catalog, null, null, new String [] {" TABLE"});

4）获取源数据库表的主键：ResultSetpkResultSet = dbMetaData. getPrimaryKeys (catalog, null, tableName);

5）获取源数据库表的外键：ResultSetfkResultSet = dbMetaData. getImportedKeys (catalog, null, tableName);

6）获取源数据库表的字段名：rsMetaData. getColumnName (i);

在获取源数据库所有表的元数据的基础上，创建保存源数据库表元数据的 XML 文件流程如图 7 所示。

根据创建 XML 文件流程，创建的 XML 文件格式如下：

<DataBase>
<table>
<tableName>
tableName
</tableName>
<primaryEntry>
<primaryKey>
<PKName>
primaryKeyName
</PKName>
</primaryKey>
</primaryEntry>
<foreignEntry>
<foreignKey>
<FKName>
foreignKeyName
</FKName>
</foreignKey>
</foreignEntry>
<columnEntry>
<column>
<columnName>
columnName
</columnName>
</column>
</columnEntry>
</table>
</DataBase>

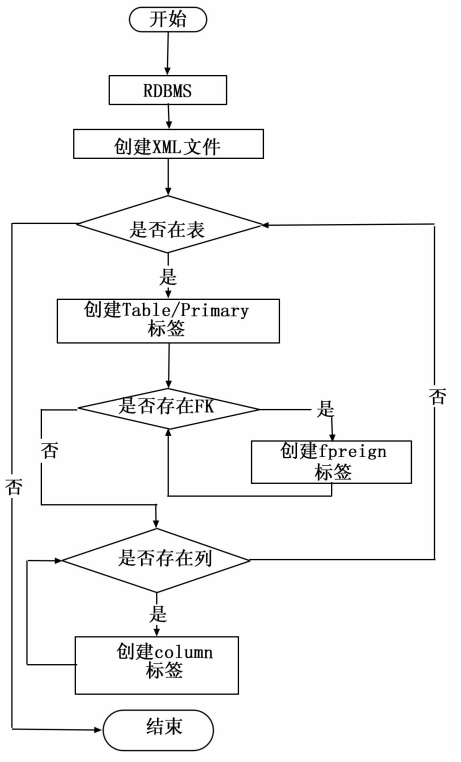

图 7　创建 XML 文件流程图

```
</PKName>
<PKNameType>
ColumnType
</PKNameType>
<pkColumnTypeLength>
ColumnTypeLength
</pkColumnTypeLength>
</primaryKey>
</primaryEntry>
<foreignKeyEntry>
<foreignKey>
<FKTableName>
ForeignKeyName
</FKTableName>
<FKColumn>
ForeignKeyColumnName
</FKColumn>
</foreignKey>
...
</foreignKeyEntry>
<columnFields>
<column>
<columnName>
columnName
</columnName>
<columnType>
columnType
</columnType>
<columnLength>
columnLength
</columnLength>
</column>
...
</columnFields>
</table>
<DataBase>
```

在此 XML 文件中记录着源数据库中所有表的元数据，主要包括表名、主键、外键所在表、字段名及字段类型，通过源数据库表元数据文件创建目标数据库 HBase 的表模式。

创建 HBase 表模式步骤如下：

1）通过高性能的 DOM4J 方式读取源数据库表元数据 XML 文件；

2）根据 XML 文件中＜Table＞标签，在 HBase 中创建 HBase 数据表及主表列族；

3）根据 XML 文件中＜foreignKey＞标签，根据 2.2 节表模式转换原则，创建外键列族。

步骤 2）中是通过 HTableDescriptor 对象和 HColumnDescriptor 对象来创建 HBase 数据表及主表列族，主要方法如下：

```
String tableName = table.get(0).getText();
  if(admin.tableExists(tableName)){
  admin.disableTable(tableName);
  admin.deleteTable(tableName);
  }else{
HTableDescriptor t =new HTableDescriptor
(tableName.getBytes());
HColumnDescriptorcf1 = new HColumnDescriptor(("CF_"+ta-
bleName)
.getBytes());
  t.addFamily(cf1);
  admin.createTable(t);
  }
```

如果在迁移过程中有符合 2.2 节所介绍的几种转换原则的数据表，则需要在完成主表的及主表列族的创建后，对相关外键表的列族进行创建，同样使用 HTableDescriptor 对象和 HColumnDescriptor 对象来进行创建操作，具体流程如图 8 所示。

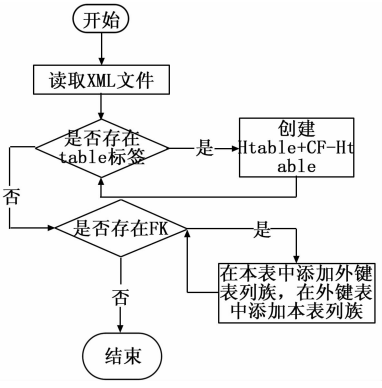

图 8　HBase 数据库表模式创建流程

完成创建 HBase 数据库的表模式后，接着进行源数据库数据向目标数据库的迁移操作，本文中自定义的 HBase 表模式特点及主要要体现数据迁移完成后的数据查询及存储性能，故在此选择使用 HBase 中的 Put 类进行数据迁移。首先通过

getConnectDB（）方法连接源数据库 MySQL 数据库，使用 SQL 查询语句获取 MySQL 数据库中数据，然后通过 HBaseConfiguration 配置连接 HBase 数据库，根据 2.2 节所定义的表模式转换原则，操作 HBase 中 Put 对象将查询出的 MySQL 数据库数据插入到 HBase 数据表中，即根据所设计的迁移方法完成数据迁移工作。

3 测试与比较

为了测试本文所设计的数据迁移方法在自动化、查询性能及存储方面的性能，选用了业界使用比较广泛的 Sqoop[17] 迁移工具对 MySQL 数据库数据进行迁移，并在不同数据集下在上述三个方面对两种迁移方式的性能进行比较，由于实验室条件限制，实验中选择了其中 500 万条数据进行测试，实验结果如下：

1）操作性能方面，使用 Sqoop 迁移工具进行数据迁移时，需要在命令行中选择数据迁移入得列族及指定 RowKey 进行迁移，而本文所设计的迁移方法在完成通用代码后均是自动完成上述工作，这一点比 Sqoop 迁移工具操作方便，自动化程度高。

2）查询性能方面，由于本文所设计的迁移方法在 2.2 节表模式设计阶段对数据的连接查询进行了深度优化，而 Sqoop 工具只是迁移数据表中的数据没有对表间关系进行迁移，所以本文所设计的迁移方法与 Sqoop 工具相比，表间的连接查询性能得到了很大的提高。在此本文以 MySQL 数据库中常见的表间连接查询“select od. userName，od. phoneNumfrom orderdetail od，orderinfooi where od. OrderID＝oi. OrderID”为例，实验结果图 9 所示。

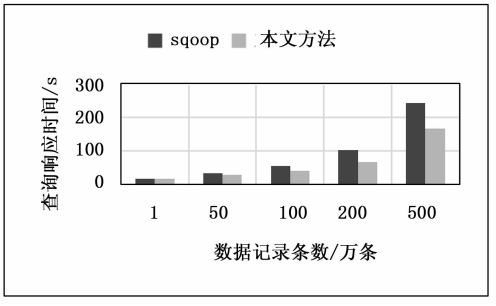

图 9 查询性能比较

3）在存储性能方面，因为本文所设计的迁移方法在进行表模式转换时，为优化表间连接查询效率进行了冗余处理，所以这方面与 Sqoop 相比，数据迁移后所占用的存储空间会较大，实验结果如图 10 所示。

通过以上的实验分析，本文所设计的数据迁移方法是以数据冗余牺牲少量存储空间换取表间连接查询性能的极大提高，在大数据时代，为了争取到更高的查询效率，牺牲少量存储空间是值得的。

4 结论

在利用本文基于 Hadoop 平台的数据迁移方法进行 MySQL 向 HBase 数据迁移，较同类迁移工具显得更加方便，而且该迁移方法不仅仅是进行系统历史数据的迁移，而且将 MySQL 数据库中的表间关系根据设定原则在 HBase 数据库中

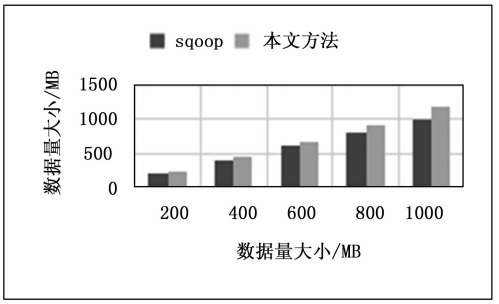

图 10 存储性能比较

进行重新设计表模式，虽然这样的表模式设计浪费了一定的数据存储空间，但是相对于其提高的查询效率带来的利益来说，损失一定的存储空间是值得的。本文所研究的数据迁移问题，提供的一种从关系型数据库向 HBase 进行数据迁移的方法，在当今大数据时代的那些既想保存系统历史数据又想发展大数据研究的互联网企业中，具有很大的应用价值。

参考文献：

[1] 何 莹. “大数据时代”的管理创新 [J]. 人力资源，2013（10）：62－63.
[2] 周亚翠. 关系型数据库 [J]. 现代情报，1998（6）：7－8.
[3] Stonebraker M. SQL databases v. No SQL databases [J]. Communications of the ACM，2010，53（4）：10－11.
[4] George L. HBase：the definitive guide [M]. Sebastopol，USA：O’Reilly Media，2011.
[5] Hadoop [EB/OL]. [2011－06]. http：//hadoop. apache. org/.
[6] Borthakur D. The Hadoop distributed file system：Architecture and design [DB/OL]. http：//hadoop. apache. org/common/docs/r0. 18. 0/hdfs _ design. pdf/.
[7] Lee K H，Lee Y J. Parallel data processing with Map Reduce：a survey [J]. ACMSIGMOD Record. 2011，40（4）：11－20.
[8] Apache HBase. Apache HBaseProject [EB/OL]. http：//hadoop. apache. org/hbase/，2014－10－28.
[9] LI C. Transforming relational database into HBase：A case study [A]. IEEE. Software Engineering and Service Sciences（ICSESS）[C]. ICSESS，2010：683－687.
[10] George L. HBase：the definitive guide [Z]. USA：O´Reilly Media，Inc.，2011.
[11] 郝树魁. Hadoop HDFS 和 MapReduce 架构浅析 [J]. 邮电设计技术，2012（7）：37－42.
[12] Kolaitis P. Schema mappings，data exchange and metadata management [A]. Proceedings of the 24th ACM Symposium on Principles of Database Systems（PODS）[C]. 2005：61－75.
[13] 李 刚. XML 讲义 [M]. 北京：电子工业出版社，2011.
[14] Arenas M，Libkin L. XML data exchange：consistency and query answering [A]. in Proceedings of the 28th ACM Symposium on Principles of Database Systems（PODS）[C]. 2005：13－24.
[15] Apache Sqoop. Apache SqoopProject [EB/OL]. http：//hadoop. apache. org/，2014－10－09.
[16] 付志成. 商品比价系统中大数据迁移及数据转换技术研究 [D]. 北京：北京邮电大学，2015.
[17] 杨寒冰，赵 龙，贾金原. HBase 数据库迁移工具的设计与实现 [J]. 计算机科学与探索，2013（3）：236－246.