

DOI:10.11830/ISSN.1000-5013.201610016



采用多叉树模型数据迁移 算法的设计与实现

宋春红¹, 王佳斌¹, 郑力新²

(1. 华侨大学 工学院, 福建 泉州 362021;

2. 华侨大学 工业智能化技术与系统福建省高校工程研究中心, 福建 泉州 362021)

摘要: 针对目前传统关系型数据库中的历史数据向非关系型数据库迁移的低效率问题, 提出利用多叉树模型对历史数据存储模式进行重构, 基于 4 种模式迁移规则对各表节点之间的关联关系进行分析, 推导算法完成传统关系型数据库中存储模式和历史数据的自动化迁移. 该算法不受源数据库存储模式的限制, 具有一定的通用性. 数据迁移实验表明: 在查询性能上, 基于多叉树的迁移算法比官方迁移工具 Sqoop 有较大的提高.

关键词: 关系型数据库; 非关系型数据库; 数据迁移; 多叉树模型; Sqoop

中图分类号: TP 31 **文献标志码:** A **文章编号:** 1000-5013(2018)06-0932-05

Design and Implementation of Data Migration Algorithm Using Multi Fork Tree

SONG Chunhong¹, WANG Jiabin¹, ZHENG Lixin²

(1. College of Engineering, Huaqiao University, Quanzhou 362021, China;

2. Industrial Intelligent Technology and System Engineering Research Center of Fujian Province Colleges
and Universities, Huaqiao University, Quanzhou 362021, China)

Abstract: In order to solve the issues on of low migration efficiency about the historical data from traditional relation database to nosql database, the multi-fork tree model is proposed to reconstruct the historical data storage model. Then analyzed the relationship between each table node based on four migration rules. Finally the migration algorithm is derived to complete the automatic migration of the storage model and historical data in the relational database. The algorithm has a certain versatility that is not restricted by the storage mode of source database. Data migration experiments verify that the migration algorithm based on the multi-fork tree has greatly improved the query performance compared with the apache official tool Sqoop.

Keywords: relational database; nosql database; data migration; multi-fork tree model; Sqoop

传统的 Web 应用系统通常基于 MySQL 和 Oracle 等传统关系型数据库. 然而, 大数据和云计算技术得到了快速的发展, 考虑到海量数据的高并发访问等特性, 关系型数据库不再适用^[1-2], 而非关系型数据库凭借其可拓展性、高可用性、对大量数据的快速查询能力, 以及对非结构数据的支持而呈现出更大的优势^[3-4]. 另外, 因为 MySQL 中的历史数据对于企业的业务研究和发展具有很大的现实和指导意义, 所以, 将传统数据模式和数据转换迁移到适合大数据平台的非关系型数据库具有重要的理论和应用价值. 随着 Hadoop 技术的不断发展和完善^[5-6], 相关的技术支持越来越丰富, 并且出现了很多数据迁移工

收稿日期: 2016-10-12

通信作者: 王佳斌(1974-), 男, 副教授, 主要从事大数据、物联网和嵌入式系统开发的研究. E-mail: fatwang@hqu.edu.cn.

基金项目: 国家自然科学基金青年科学基金资助项目(61505059); 福建省泉州市科技计划项目(2013Z12)

具和应用^[7],如 Apache 的官方应用 Sqoop 和 Kettle 等,不足之处是它们只能机械化地完成数据的迁移,而无法进行数据库中众多表模式的转换和迁移.但是,对关系型数据库到大数据的转换设计往往需要考虑到一些实际需求和平台特性的因素.所以,相关工作的完成存在很大的难度.NoSQL 数据库元素众多,也使 NoSQL 没能形成一个统一的标准.本文从应用广度和应用技术等多角度出发,选择 HBase 数据库^[8],针对文献^[9]提出的几种模式转换方法,提出一种基于多叉树模型的迁移算法.

1 数据库模式迁移思想

在进行实际操作时,基于文献^[9]和关系型数据库的关系范式^[10],将这 4 种转换模型进一步划分为两类,即单表模式的转换和多表模式的转换.

单表模式的转换较为简单,即完成单个表的转换.这种表与源数据库中的其他表之间不存在任何关联关系,当应用程序对这些表进行操作时也不会涉及到其他表.

多表模式即包含多个相互关联的表节点,各表之间通过主键和外键的方式进行关联.4 种转换模型

的分类,如图 1 所示.

将 4 种模式转换中基本表的转换归为单表模式,将内嵌转换、递归转换和分割转换划分为多表模式的转换.在对多表模式进行转换时,借助多叉树结构^[11-12].多叉树是节点的有限集合,当节点数为 0 时,该多叉树为空树;当节点数大于 0 时,该多叉树由一个根节点和它的子树构成.树中的每个节点都可以有任意个子节点,且子节点之间没有严格的排列顺序.

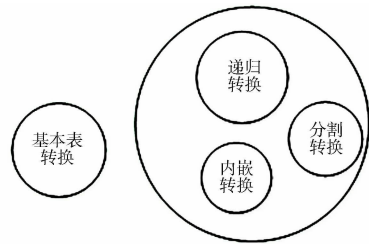


图 1 4 种转换模型的分类

Fig. 1 Classification on four kinds of transformation model

2 多叉树结构的设计与实现

进行数据模式的转换和迁移时,首先,从源数据库中获取描述数据库模式的信息并存储到多叉树结构中;然后,通过对树种各节点的遍历等操作完成模式的转换和迁移^[13].用多叉树的节点表示源数据库中的数据表.节点设计包含:用于存储表名的节点名、列属性名、主/外键属性名、父节点链表、孩子节点链表、标记信息等.

多叉树节点有 8 种成员函数.1) 无参构造函数 TreeNode(),实现表节点的初始化;2) setNodeName()为表节点设置节点名;3) addparentNode(TreeNode treenode)为表节点添加父节点(当根据各表之间的关系构造多叉树时);4) addChildNode(TreeNode treenode)为表节点添加孩子节点(当根据各表之间的关系构造多叉树时);5) getChildList()返回当前节点的孩子节点组成的 List;6) getParentList()返回当前节点的父节点组成的 List;7) getElders()返回当前节点的父辈节点;8) isLeaf()判断当前节点是否为叶子节点

通过构造表节点的属性和成员函数,存储表的各属性,通过调用各节点的成员函数灵活地实现表与表之间关系的描述.因此,可以通过构造多叉树,较为容易地实现源数据库中表模式的获取和转换.

3 数据库的迁移算法

3.1 表模式迁移算法实现

1) 单表模式的迁移.首先,寻找单表;然后,按照基本表的转换方法创建相应的 HBase 表.该算法流程图,如图 2 所示^[11].

2) 多表模式的迁移.多表模式的迁移相对单表模式的迁移较为复杂,并且要在单表模式的迁移完成之后进行.首先,对剩余的表节点构造多叉树;然后,对多叉树分情况进行内嵌变换、递归变

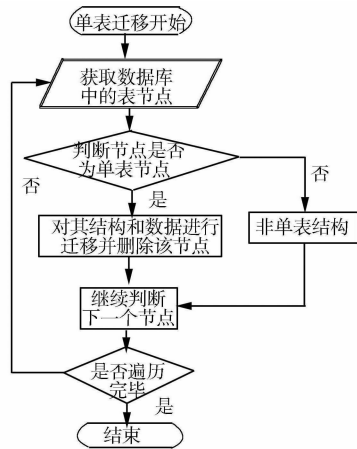


图 2 单表模式迁移图

Fig. 2 Single table schema migration

换和分割变换. 算法流程图, 如图 3 所示.

分割变换的模型, 如图 4 所示. 图 4 中: 1 为 website 节点创建新的副本节点 website'; 2 为依次删除 website 节点中首个父节点之后的所有父节点; 3 为依次将 website' 的父节点设置为 website 节点删除的父节点, 并将相应父节点的 childList 中的 website 节点更改为 website'.

多表模式的递归变换可以简单地看作是多个嵌套变换的组合, 因此, 在对多表模式进行转换时可以同时完成嵌套和递归变换. 具体方法为: 首先, 获取叶子节点; 然后, 利用节点的 getElders() 方法返回, 整个路径上的表节点作为一个列族迁移到 HBase 表中.

3.2 数据库的迁移过程

在数据库迁移的过程中, 需要在线连接 MySQL, HBase 数据库, 获取所有关系型数据表模式. 然后, 根据表与表之间的关系决定对表模式进行怎样的转换. 总的迁移过程的流程图, 如图 5 所示.

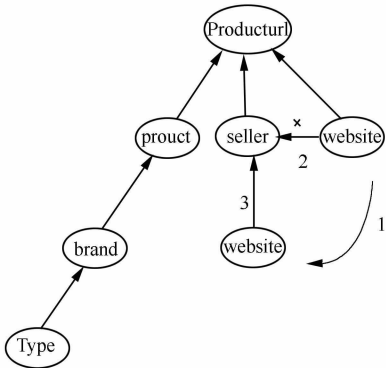


图 4 多表模式中的分割变换

Fig. 4 Segmentation in multi-table model

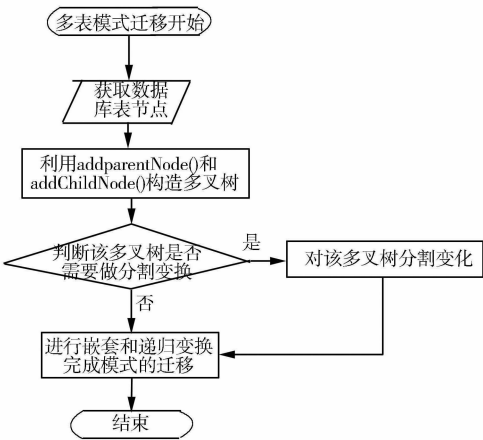


图 3 多表模式迁移

Fig. 3 Multi-table schema migration

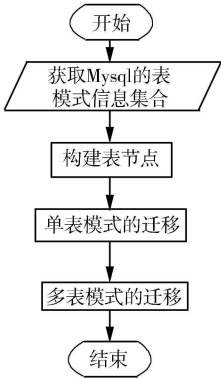


图 5 迁移过程流程图

Fig. 5 Migration process flow chart

4 迁移算法在 HBase 的具体实现

以在线商品比价系统为例, 对文中提到的迁移算法进行测试^[14]. 将现有历史数据和平台迁移到 HBase 分布式数据平台上, 并对算法的有效性进行验证. 在线商品比价系统在原有 MySQL 数据库中包含了 producturl, product, brand 和 website 等表, 表与表之间的关系模式图, 如图 6 所示.

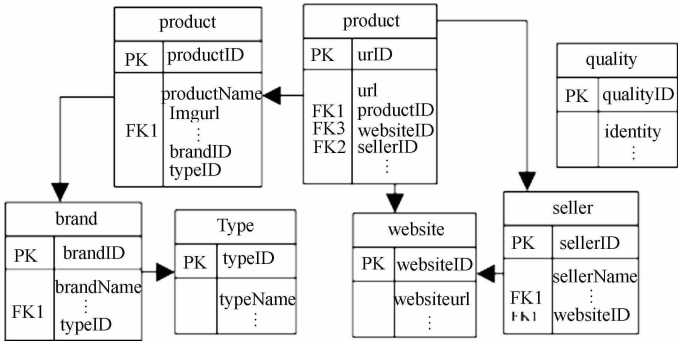


图 6 MySQL 数据库实例

Fig. 6 MySQL database instance

由上面的数据库实例模型可以对数据库中的表模式进行转换分析.

1) 单表模式转换. 由图 6 可知: 在线商品比价系统的数据库中 quality 表的属性与其他表之间不存在任何互联关系, 并且 quality 表与其他表之间不存在任何连接查询. 根据上面的算法将该表判定为单表, 直接将 quality 的表名作为迁移之后 HBase 中表的表名, 将 quality 的主键作为 HBase 中相应表的 RowKey, 且只存在一个名为 quality 的列族.

2) 分割变换. 按照上面的算法, 可以将实例中相互关联的表构建成一棵多叉树. 将会有两个父节点, 因此, 要对该节点进行分割变换, 即创建 website 的一个副本节点 website'. 将 website 的第 2 个父节点在 website 的 parentList 中删除, 并赋值给 website' 的 parentList. 修改相应的 parent 节点的孩子节点, 完成表模式的分割变换.

3) 嵌套变换和递归变换. HBase 数据库不能对多表进行 join 查询^[15-16], 因此, 不能像传统关系型数据库, 为了满足关系范式把相互关联的信息存储在多个表中来减少数据的冗余. 相反地, HBase 牺牲空间存储率, 将相互关联的多个表存储在一个大表中, 实现了巨量数据的高效查询^[17]. 因此, 要将图 6 中所有表构造成图 4 所示的多叉树; 然后, 依次从叶子节点开始通过嵌套和递归变换, 将每个分支作为 HBase 表的一个列族成员存储到 HBase 数据库中. 迁移后的 qualiyl 描述为 {Name=>'quality'}; producturl 描述为 {Name=>'product'}, {Name=>'product'}. 完成上述表模式的转换和迁移, 按照相应的列族和列的关系进行数据的批量迁移.

5 性能测试与分析

实验测试环境, 如表 1 所示. 表 1 中: f 为频率; c 为容量.

表 1 实验测试环境
Tab. 1 Test environment

主/从配置	IP	f (CPU)/GHz	c (RAM)/GB	c (ROM)/GB
Master	192.168.1.2	3.6	4	500
Slave1	192.168.1.3	3.6	4	500
Slave2	192.168.1.4	3.6	4	500
Slave3	192.168.1.5	3.6	4	500

分别用文中的迁移算法、Sqoop 对关系型数据库 Goods 进行迁移, 对比结果分析可知: 在自动化方面, Sqoop 迁移工具需要手动指定数据库名、数据表名、数据类型和迁移后在 HBase 端的列族名及列名等参数, 且这些参数都需要一一对应, 否则, 很难实现迁移后应用的正确运行. 所以, 在迁移的时候就会比较繁琐. 而文中提到的迁移算法, 在执行的时候能够自动获取 MySQL 端源数据库的表模式, 并转换为相应的 HBase 表模式, 确定 HBase 端的表名、列族名和列名等, 自动化程度很高.

在存储空间上, 因为文中工具通过内嵌、递归和分割变换规则对原始表中的数据进行了冗余复制, 因此, 同 Sqoop 相比, 迁移后的数据需要更大一些的存储空间. 数据存储容量对比, 如图 7 所示. 测试查询性能对比, 如图 8 所示.

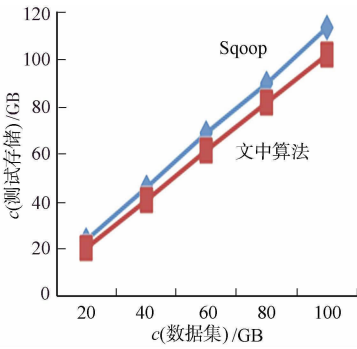


图 7 数据存储容量对比

Fig. 7 Data storage capacity contrast

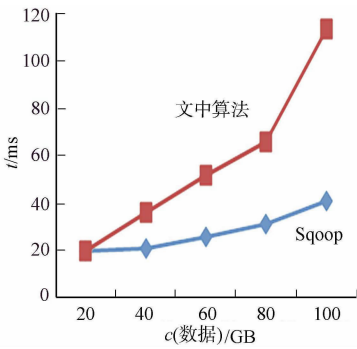


图 8 测试查询性能对比

Fig. 8 Test query performance contrast

在查询性能上, 由于 HBase 数据库不支持表间的 join 操作, Sqoop 迁移的数据结果在对“SELECT

productName,Imgurl,brandName where product. brandID=brand. brandID”这样的 SQL 语句的执行时效率会非常低^[18],而文中的算法,利用 4 种模式转换规则对源数据表的模型进行了转换,将表间的 join 操作转换为了同表内不同列族、不同列的查询操作. 依据 HBase 表内快速查询的特性,以及图 8 所示的测试结果可知:文中的算法更胜一筹.

6 结 论

对大数据迁移工具的研究背景和现状进行了阐述. 然后,基于文献[9]中的 4 种模式转换模型提出了一种基于多叉树模型的迁移算法,并进一步阐述了该算法实现的具体原理. 最后,以在线商品比价系统的迁移为例,实现了传统关系数据库 Goods 中表模式和历史数据向 HBase 数据库的自动迁移.

相对于大多数现有迁移工具而言,该算法一方面提高了自动化程度,另一方面,在尽量保证数据完整性的前提下,提高了迁移后的数据查询性能. 但是,在对 MySQL 的源数据表进行分割变换时,采用了完全备份的方法,这也在一定程度上加大了数据的存储空间. 所以,后续工作中还应加入对业务日志的分析模块,更细腻地完成表模式分割转换,达到优化存储空间占用率的目的.

参考文献:

[1] HE Jianke. 大并发数量中的 MYSQL 瓶颈与 NOSQL 介绍[EB/OL]. [2013-06-09]. <http://hejianke83.blog.163.com/blog/static/.05/>.

[2] 张华强. 关系型数据库与 NoSQL 数据库[J]. 电脑知识与技术,2011,20(7):4802-4804.

[3] 覃雄派,王会举,杜小勇,等. 大数据分析: RDBMS 与 MapReduce 的竞争与共生[J]. 软件学报,2012,23(1):32-45.

[4] 王珊,王会举,覃雄派,等. 架构大数据: 挑战、现状与展望[J]. 计算机学报,2011,34(10):1741-1752.

[5] VORA M N. Hadoop-HBase for large-scale data[C]//InternationalConference on Computer Science and Network-Technology. [S. l.]:IEEE Press,2011.

[6] NAHEMAN W,WEI J. Review of NoSQL databases and performance testing on HBase[C]//International Conference on MechatronicSciences, Electric Engineering and Computer. [S. l.]:IEEE Press,2013.

[7] Sqoop 官网. Apache software foundation[EB/OL]. [2016-11-23]. <http://sqoop.apache.org/>.

[8] RODEK L,POULSEN H F. A storage model of equipment data based on HBase[J]. Applied Mechanics and Materials,2015,713/714/715(2):2418-2422.

[9] 宋春红,王佳斌,郑力新. 一种 MySQL 到 HBase 的迁移策略的研究与实现[J]. 微型机与应用,2016,35(13):83-85.

[10] 百度百科. 数据库范式[EB/OL]. [2016-09-16]. <http://baike.so.com/doc/4367825-4573590.html>.

[11] 刘小晶,杜选. 数据结构: Java 语言描述[M]. 北京:清华大学出版社,2014:149-189.

[12] 刘增杰,张少君. MySQL5.5 从零开始学[M]. 北京:清华大学出版社,2012:340-402.

[13] 罗林球,孟琦,李晓,等. 异构数据库迁移的设计与实现[J]. 计算机应用研究,2006,23(12):233-235. DOI:10.3969/j. issn. 1001-3695. 2006. 12. 077.

[14] 方英兰,陈兵辉,唐苗. 基于 JDBC 的优购数据库迁移系统的设计与实现[J]. 北方工业大学学报,2013,25(1):5-10. DOI:10.3969/j. issn. 1001-5477. 2013. 01. 002

[15] XU Junwu,LIANG Junling. Research on a distributed storage application with HBase[J]. Advanced Materials Research,2013,631-632:1265-1269. DOI:10.4028/www.scientific.net/AMR. 631-632. 1265

[16] DIMIDUK N,KAURANA A,谢磊. HBase 实战[M]. 北京:人民邮电出版社,2013:3-4,29-31.

[17] 蒋燚峰. HBase 管理指南[M]. 北京:人民邮电出版社. 2013:82-85.

[18] 代志远,刘佳,蒋杰. HBase 权威指南[M]. 北京:人民邮电出版社,2013:339-364.

(责任编辑: 陈志贤 英文审校: 吴逢铁)