文章编号：1001-4632（2014）01-0135-07

基于NoSQL数据库的大数据查询技术的研究与应用

朱建生，汪健雄，张军锋

（中国铁道科学研究院 电子计算技术研究所，北京 100081）

摘 要：基于NoSQL数据库理论，根据应用场景的不同，将NoSQL数据库分为面向高性能读写、面向文档和面向分布式计算的3种类型。对比分析这3种类型数据库的6种代表产品的优缺点，结合铁路客票实名制售票信息综合分析系统中的大数据操作的需求，选用NoSQL数据库中的面向分布式计算的Cassandra数据库。基于Cassandra数据库，提出铁路客票实名制信息综合分析系统的技术架构，并设计反向索引以构建客票实名制乘车信息的查询策略和查询流程。通过性能测试，验证了NoSQL数据库技术在处理大数据查询和分析中的高可用性，可突破传统关系型数据库和数据仓库在应用中所遇到的查询性能、扩展性以及投资成本的瓶颈。

关键词：NoSQL数据库；Cassandra数据库；大数据处理；反向索引；数据查询

中图分类号：U293.221；TP391 文献标识码：A doi：10.3969/j.issn.1001-4632.2014.01.21

截至2011年底，中国铁路客票预订与发售系统（China Railway Ticketing and Reservation System，TRS）[1]除少量普通列车的中间站外，对于全路旅客列车均实现了实名制售票。TRS在售票时记录每个乘车人的乘车信息和实名身份信息，随着实名制售票数据的长期积累和不断完善，急需研究铁路客票实名制信息查询技术，并建立铁路客票实名制信息查询分析系统，用于统计、分析和查询售检票过程中产生的实名制售票信息，以丰富铁路客运业务分析数据类型，全面挖掘旅客购票和乘车的规律，为铁路客运客户关系管理奠定基础。

由此对铁路客票实名制查询分析系统提出如下需求：具有较高横向扩展能力的数据存储机制；针对大数据进行查询策略专项优化；具有较高级的数据挖掘分析和研判应用。而建立该系统仅依靠关系型数据库已经无法满足需求，必须借助数据仓库存储策略和数据挖掘技术。近年来，随着高性能计算技术的高速发展，带动了分布式计算、并行计算和虚拟化技术的不断进步，为寻求低成本、高性能的数据挖掘计算带来了机遇。因此，本文基于NoSQL（Not Only Structured Query Language，不仅限于结构化查询语言）数据库技术，提出铁路客票实名制信息综合分析系统技术架构，设计反向索引以构建高性能的数据查询策略及处理流程，以满足系统的功能和性能需求。

1 NoSQL数据库理论基础

NoSQL数据库是由许多理论支撑作为前提的，其中与建立铁路客票实名制查询分析系统相关的理论包括CAP理论、扩展Bigtable存储模型和一致性哈希算法。

1.1 CAP理论

CAP定理：对于分布式系统的要求体现在一致性、可用性和分区容错性；对于任一事实存在的分布式系统，只可同时满足上述3个方面中的任意2点，而无法三者兼顾[2]。

根据CAP理论，关系型数据库管理系统（Relational Database Management System，RDBMS）可满足一致性和可用性，但无法很好地支持分布式应用，因此在小规模数据量时可达到很好的效应，但随着数据量和应用范围的增长，性能大幅度下降。对于许多大数据应用而言，侧重于系统的可用性，而对于一致性的要求可以降低，从而产生了弱一致性理论BASE（Basically，Available，Soft-state，Eventual consistency），即反ACID（Atomicity，Consistency，Isolation，Durability）模型。BASE理论的思想是，对于分布式系统，只

收稿日期：2012-07-10；修订日期：2013-08-06

基金项目：中国铁道科学研究院行业服务技术创新项目（1151DZ1003）

作者简介：朱建生（1972—），男，山西太原人，研究员。

?1994-2014 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

需要满足最终一致性（Eventual Consistency）即可，而且可以是异步的，即柔性状态（Soft－state）。而 NoSQL 正是利用最终一致性[3]满足了可用性和分区容错性，近年来得到了广泛的关注，因此 NoSQL 是 CAP 理论的产物。

1.2 扩展 Bigtable 存储模型

Bigtable 是一个分布式的结构化数据存储系统，底层是一个稀疏的、分布式的、持久化存储的多维度排序 Map 结构，通常用来处理分布在数千台普通服务器上的 PB 级的大数据[4]。这种 Bigtable 存储模型可以按如下定义进行形式化描述：设 Bigtable 存储模型中关键字为 K，列族集合 $F=\{F_i \mid i=1, 2, \cdots, I\}$ 中有 I 个列族 F_i；超列集合 $S=\{S_j \mid j=1, 2, \cdots, J\}$ 中有 J 个超列 S_j；列集合 $C=\{C_m \mid m=1, 2, \cdots, M\}$ 中有 M 个列 C_m。则各存储内容之间的关系如下[5]

$$K \rightarrow \{F_i \mid i=1,\cdots,I\} \tag{1}$$

$$F \rightarrow \{S_j, C_j \mid, j=1,2,\cdots,J\} \tag{2}$$

$$S \rightarrow \{C_m \mid m=1,2,\cdots,M\} \tag{3}$$

其中符号→表示映射关系，式（2）表示 S 和 C 可属于同一层次，F 也可以映射至 C。

主流的 NoSQL 数据库如 HBase 和 Cassandra 数据库等都是采用了扩展的 Bigtable 存储模型。

1.3 一致性哈希算法

为实现在集群中对服务器节点的数据访问，NoSQL 数据库通常使用哈希取模的方式将数据存储在服务节点中。如集群中可用服务节点数为 N，那么 key 值为 K 的数据请求可以使用简单的哈希函数 hash（K）mod N 找到对应的服务节点，该方法具有简单易用的特点。但随着服务节点的扩充，可能会使得缓存无法命中，导致服务节点需要重新建立缓存并出现大量的缓存数据迁移，从而引起系统负荷剧增而宕机的可能。为了解决该问题，1997 年 David 等学者提出了一致性哈希算法（consistent hashing）[6]。NoSQL 数据库中的一致性哈希算法步骤如下。

（1）将多个服务节点看作圆环上的多个节点（理论上最多支持 2^{32} 个服务节点，顺时针分布），计算出集群中每个服务节点的哈希值，并将其分配到圆环中的节点上，如图 1 所示（以 5 个服务节点为例）；然后使用同样的方法求出所需存储的 key 的哈希值，也将其分配到该环形区间的服务节点上。

（2）从数据映射到的位置开始顺时针查找服务节点，并将数据保存到找到的第 1 个服务节点上。如果查找了 2^{32} 个节点仍然没有找到服务节点，则将该数据保存到第 1 个服务节点上。

（3）假设在原有集群的基础上新增加 1 个服务节点 6（见图 1），且访问策略不变，则将该服务节点逆时针方向相邻节点的 key 对应的数据迁移到新增服务节点上，使得仅在新增服务节点 6 与服务节点 2 之间的区间上存在数据找不到服务节点的可能，从而提高了缓存的命中率。

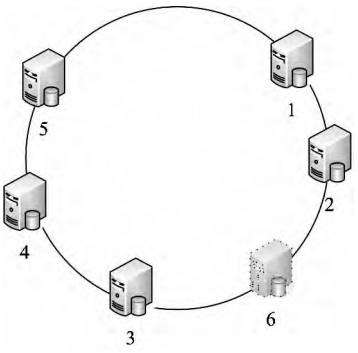

图 1 一致性哈希算法示意图

对于分布式大数据系统而言，多数情况下只需要基于 BASE 理论寻求 CAP 平衡中的可用性和分区容错性，并达到满足高并发的需求，对一致性的要求只需要满足 BASE 最终一致性即可。NoSQL 由于具有最终一致性特性，其作为传统关系型数据库管理系统的进一步发展和补充[7-8]，为大数据查询、分析和挖掘提供了有效的途径。

2 NoSQL 数据库分类及选型

NoSQL 是多种非关系型数据库的集合，根据应用场景的不同可将这些非关系数据库概括为以下 3 种类型。

1）面向高性能读写的 NoSQL 数据库

面向高性能读写的 NoSQL 数据库具有较为出色的读写性能。在一般大型网站平台的构建中，通常使用面向高性能读写的 NoSQL 数据库，代表产品主要有 Memcached 和 Redis 数据库。

2）面向文档的 NoSQL 数据库

面向文档的 NoSQL 数据库，在保证大数据存储的基础上具有良好的查询性能。其数据一般采用 json 格式的文档存储。灵活的 json 格式使得可以对特定字段建立索引，为实现关系型数据库的部分功能提供了可能。代表产品主要有 MongoDB 和 CouchDB 数据库。

?1994-2014 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

3）面向分布式计算的 NoSQL 数据库

面向分布式计算的 NoSQL 数据库具有良好的横向扩展能力，要求在不停止服务的情形下，增加更多的节点；对任意 1 个节点写操作的数据会同步复制到其他节点上。代表产品主要有 Cassandra 和 Voldemort 数据库。

表 1　NoSQL 数据库的分类与对比

分类	代表产品	原理	优点	缺点
面向高性能读写的 NoSQL 数据库	Memcache	通过维护内存哈希表存储各种格式的数据，如图像、视频、文件和数据库检索结果等	是 key-value 缓存服务技术领域的领头羊。减少了数据库的负载，提升了系统访问速度	缺乏认证和安全管理机制
	Redis	数据库全部加载进内存中操作，定期通过异步操作将数据库中的数据冲刷到硬盘上进行保存。	纯内存操作，性能出色。支持 Python，Ruby，Erlang 和 PHP 多种客户端	不具有可扩展能力，存储量受物理内存限制，只能依赖客户端实现分布式读写
面向文档的 NoSQL 数据库	MongoDB	介于关系数据库和非关系数据库之间的产物。自带 GridFS 分布式文件系统，以支持大数据的存储	数据结构松散，采用 bjson 格式存储数据，支持复杂数据类型。查询语言功能强大且易于掌握，数据的访问效率较高。	并发读写效率不突出
	CouchDB	基于 Erlang/OTP 构建的高性能分布式容错非关系型数据库系统。采用简单文档数据类型	保证大数据存储的同时，具有良好的查询性能	没有内置水平扩展的解决方案，但有外部的解决方案
面向分布式计算的 NoSQL 数据库	Cassandra	采用环形集群结构和 Dynamo 哈希一致性算法，按列存储，适用于结构化和半结构化数据的存储	满足高横向扩展性，支持动态列结构，数据库模式灵活使得增加或者删除字段非常方便	查询语言功能稍弱于 MongoDB
	Voldemort	基于 Berkley DB 的持久化技术以及哈希一致性算法策略	在 CAP 中选择了 AP，对读取性能做了优化；适用于高并发应用场景	未对写入性能做优化

将 NoSQL 数据库的这 6 个代表产品的原理、优点和缺点列于表 1。由表 1 可以看出，每个产品都是适用于特定的应用场景。因此，单纯地判断某一数据库明显优于另一数据库是没有意义的，必须根据特定的应用场景和性能需要选择合适的 NoSQL 数据库产品。其中，面向分布式计算的数据库 Cassandra[9-10] 结合了 Bigtable 的列族和 Dynamo 的分布式系统技术，具有良好的横向扩展性、动态的列结构、灵活的模式定义，支持对 Key 进行范围查询，支持丰富的数据结构，具有强大的查询语言功能，近年来成为分布式大数据分析系统存储架构的重要解决方案。考虑到对铁路客票实名制信息综合分析系统的需求、程序编制的灵活性以及系统建立的前期投入成本，决定选用 Cassandra 数据库作为铁路客票实名制信息综合分析系统的 NoSQL 数据库。

3　NoSQL 数据库在铁路客票实名制信息综合分析系统中的应用

根据实名制乘车信息系统的业务场景与数据规格，提出基于 Cassandra 数据库的铁路客票实名制信息综合分析系统的技术架构，利用反向索引技术构建高性能的数据查询策略和处理流程。

3.1　系统技术架构

使用分层的 Java 设计模式和基于 Cassandra 数据库的数据层组织模式，设计铁路客票实名制信息综合分析系统的技术架构，如图 2 所示。该技术架构分为 4 层，各层的主要功能如下。

1）数据层

TRS 系统售票信息、实名制身份信息等数据通过系统工作流程序的调度，使用开源 ETL 工具 Kettle，定期进行数据抽取、转换并加载到 Cassandra 数据库中，供服务层调用。

2）服务层

服务层主要基于数据层的数据，利用 Java 技术，结合工作流机制，为应用层提供关联查询、批量比对、统计分析、挖掘分析和数据接口等服务。服务层支持服务扩展，是实现业务应用的基础。

3）应用层

应用层是基于服务层的接口功能，提供实名制信息管理、研判和推送发布等应用。

4）展现层

展现层基于浏览器应用建立人性化的用户界

?1994-2014 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

面，为应用系统、管理系统、相关网站等应用提供导航功能，并设立统计分析结果、研判分析结果、信息资源情况等信息栏目，展现各应用中用户关注的信息。

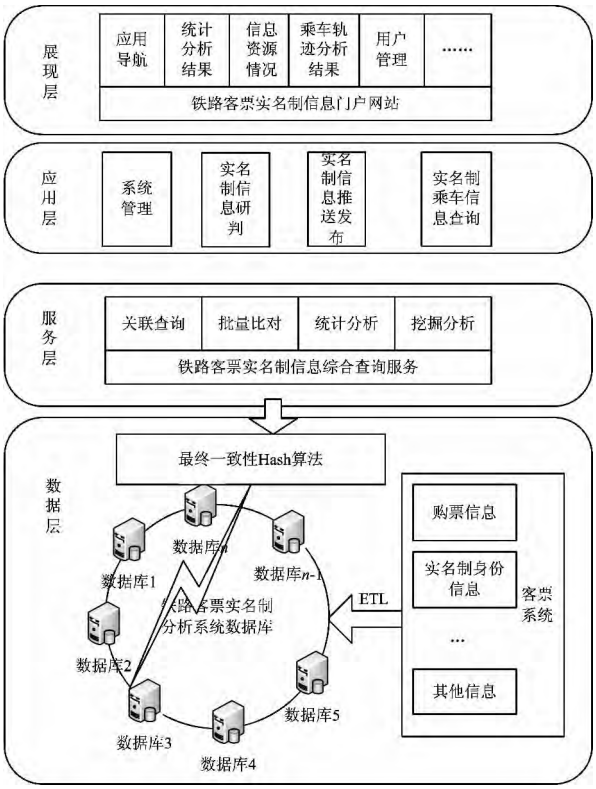

图 2 铁路客票实名制信息综合分析系统技术架构

3.2 业务场景及其涉及的信息

铁路客票实名制信息综合分析系统的 1 个典型业务场景是：根据单条或批量旅客身份信息查询旅客购票信息和乘车轨迹；或者根据席位信息、车次信息查询旅客的身份信息；或者是以上组合条件的查询。该典型业务场景中涉及的售票信息和旅客身份信息分别如下。

1）实名制售票信息

实名制售票信息包括由客票系统产生的售票存根、废票存根、改签存根和退票存根，各存根包含的信息分别如下。

售票存根包含的信息：售票车站、售票窗口、票号、车次、乘车日期、席别、席位号、发站和到站等。

废票存根包含的信息：作废售票车站、作废窗口、作废时间、售票车站、售票窗口和票号等。

退票存根包含的信息：退票车站、退票窗口、退票日期、售票车站、售票窗口和票号等。

改签存根包含的信息：改签车站、改签窗口、改签新票票号、售票车站、售票窗口、票号等。

2）实名制旅客身份信息

实名制旅客身份信息记录了与旅客所购买车票相关联的信息，包括旅客姓名、证件类型和证件号码、售票车站、售票窗口和票号等。

3.3 基于 Cassandra 数据库的反向索引设计

Cassandra 数据库的数据存储模型是一种 5 维的分布式 key-value 结构，其包括键值空间（key space）、列（column）、列族（column family）、超列（super column）和行（row），以 4 级嵌套的形式存在，如图 3 所示。

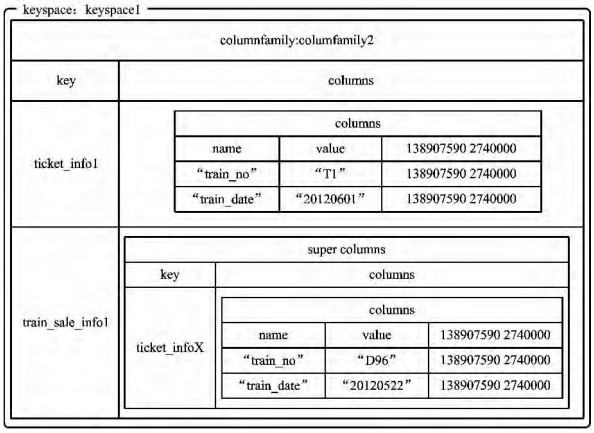

图 3 Cassandra 数据库的数据存储模型

Cassandra 数据库存储数据的流程为：在 Cassandra 数据库写入数据之前，先将写入动作记录到提交日志（commit log）中；然后将数据写入到 column family 对应的 Memtable 中（Memtable 是一种将数据按照 key 排序的内存结构）；达到系统预先设定的条件时，再将 Memtable 的数据批量持久化到磁盘上，存储为 SSTable[8] 结构。

若查询条件为使用存储模型中的部分列进行检索，根据式（1）—式（3）可以建立如下反向索引 K'。

$$K' \rightarrow \{F_i \mid i = 1,2,\cdots,I\}, \tag{4}$$

其中，$K' = \{C_l \mid C_l \subset C\}$

对不同用户的查询请求，根据式（4）在对应的不同索引结构中进行查询。例如：若需要在图 3 的数据存储模型中找出车次为 D96 的售票信息，则需先建立如图 4 所示的反向索引，将车次作为 key 与售票信息 ID 建立联系，即通过 train_no 找到对应的 ticket_info_index，然后根据 ticketinfo_index 判断数据分布在 Cassandra 集群的哪些节点上，最后在这些节点再通过索引反向检

?1994-2014 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

索相应的 ticketinfo _ID，从而快速读取到该车次的其他信息。而在关系型数据库中，是使用 where train _no＝D96 的 SQL 查询条件实现查询的。由此可见，在 Cassandra 数据库中，通过建立反向索引，虽然占用了一定的存储空间，但正是通过这种空间换时间的方式，才获得了相对较高的查询性能。

ticketinfo_index	
key	columns
train_no	ticketinfo_ID

图 4　Cassandra 数据库的反向索引

3.4　查询策略

由反向索引关系式（式（4））可知，从实名制售票业务数据 ticketinfo 和实名制旅客身份信息 passengerinfo 中找出可以对应的键值信息，然后做关联操作就可查询到符合条件的实名制乘车信息。因此，根据 Cassandra 数据库的反向索引技术，设计如 5 所示的查询策略。

该查询策略为首先根据查询条件，分别针对实名制售票业务数据 ticketinfo 和实名制旅客身份信息 passengerinfo 设计反向索引 ticketinfo _index 和 passengerinfo _index。ticketinfo _index 的 columns 包括了这 2 张表关联时需要的唯一键值，根据车票信息进行查询的条件（train _date，train _no）包括在 ticketinfo _index 的 key 列表中；而 passengertinfo _index 的 columns 也包括了这 2 张表关联时需要的唯一键值，根据实名制身份信息查询所需要的查询条件（id _kind，id _no）包含在 passengertinfo _index 的 key 列表中。这样当任意 1 个查询条件或其查询条件的组合查询请求到来时，都将转换为 2 组满足条件的唯一索引键值列表，分别通过唯一索引键值列表去检索相应的信息，则可以很快地检索并获取到关联信息。

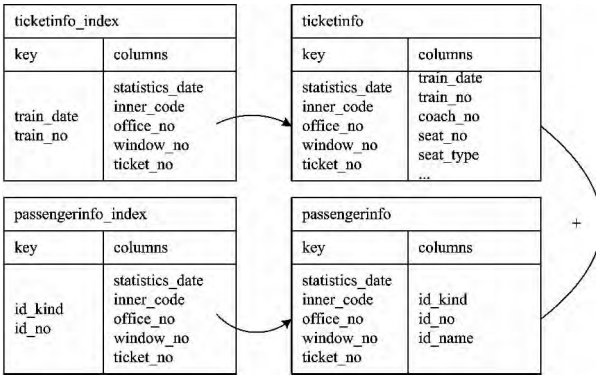

图 5　查询策略

3.5　数据更新与查询流程

根据建立的系统技术架构、反向索引和查询策略，针对 TRS 系统中的售票存根、废票存根、改签存根和退票存根的特点，设计如图 6 所示的数据更新与查询流程。

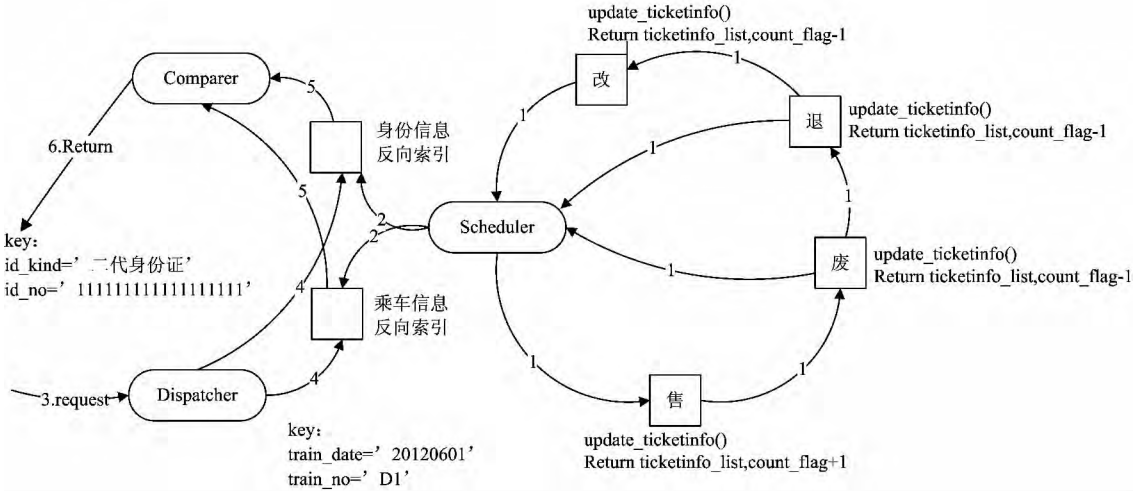

图 6　数据更新与查询流程

由图 6 所示的数据更新步骤如下。

Step 1　对加载进来的信息，根据唯一索引值判断是否为售票存根。若是，则将该信息写入乘车信息库，转 Step 2；否则，根据唯一索引值，依次判断是废票存根、退票存根、改签存根中的哪一种，相应地执行 update _ticketinfo（）过程，删

?1994-2014 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

除这条记录，转 Step 2。

Step 2 更新乘车信息和身份信息的反向索引，供查询流程使用。

Step 3 提交 1 个实名制乘车组合查询请求。

Step 4 分发器（Dispatcher）将请求拆分成针对 2 个事实表的子请求，分别检索乘车信息的反向索引和身份信息的反向索引。

Step 5 比较器（Comparer）将子请求的检索结果进行关联，查询记录。

Step 6 返回查询结果，查询结束。

4 性能测试

为了验证基于 NoSQL 的铁路客票实名制信息综合分析系统的查询性能，针对 Cassandra 数据库的查询性能进行测试，并与数据库仓库 SybaseIQ、关系型数据库 SybaseASE 的查询性能进行对比。3 种数据库的测试环境配置见表 3，其中 SybaseIQ 和 SybaseASE 在同一台小型机上。采用 NoSQL 查询程序和 SQL 查询语句对这 3 种数据库的查询性能分别进行测试。

表 2 测试环境配置

数据库	主机类型	处理器	内存	操作系统
Cassandra	3 节点 PC 服务器	每节点双核	每节点 8G	RedHat5
SybaseIQ	单节点小型机	CPU 数浮动	最大 70G	HP-UX
SybaseASE	单节点小型机	10C（max online engines）	最大 78G	HP-UX

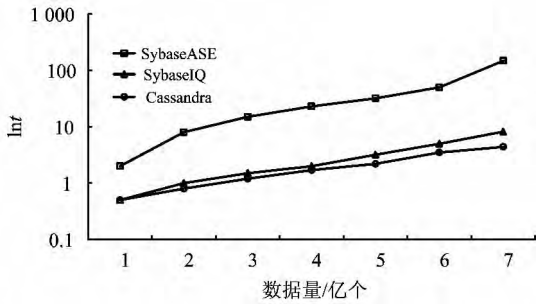

图 7 关联查询性能曲线

以测试所需访问的最大数据量为横坐标，以查询耗时 t 的自然对数为纵坐标，绘制查询性能曲线，如图 7 所示。由图 7 可看出：随着查询数据量的增大，Cassandra 与 SybaseIQ 的查询耗时变化不大，几乎呈线性分布，而关系型数据库 SybaseASE 的耗时呈指数级增长；使用 SQL 语句查询的 SybaseIQ，尽管使用了列存储技术并运行在配置更高的小型机上，但查询效率的变化与 Cassandra 数据库几乎相当，这是因为 Cassandra 数据库以其独特的设计架构、索引机制获得了高效的查询性能，同时由于其灵活的横向扩展性，使得不断加入的廉价的 PC 服务器可以获得更大的存储空间和数据分布均衡性，这意味着使用较低廉的成本，可以获得与采用昂贵的硬件配置和商业数据仓库软件相当的查询性能，甚至可以获得高于商业数据仓库的查询性能。

5 结 语

本文分析了 NoSQL 数据库的理论基础，根据应用场景的不同，将 NoSQL 数据库分为面向高性能读写、面向文档和面向分布式计算的 3 种类型，根据 3 种类型 NoSQL 数据库的 6 种代表产品的优缺点以及铁路客票实名制售票信息综合分析系统的需求，选用面向分布式计算的 Cassandra 数据库。基于 Cassandra 数据库，提出了铁路客票实名制信息查询分析系统的技术架构，并设计反向索引实现了实名制乘车信息查询的查询策略和查询流程。通过性能测试，验证了采用 NoSQL 数据库，极大地缩短了数据搜索时间，提高了数据读写效率和横向扩展能力，说明 NoSQL 数据库可为大数据查询分析提供有效的途径。

参 考 文 献

[1] 王军，刘春煌. 铁路客票发售和预订系统 [J]. 中国铁道科学，2001，22（3）：137-139.
（WANG Jun，LIU Chunhuang. Railway Ticket Selling and Booking System [J]. China Railway Science，2001，22（3）：137-139. in Chinese）

[2] GILBERT S，LYNCH N. Brewer's Conjecture and the Feasibility of Consistent，Available，Partition-Tolerant Web Services [J]. Sigact Newsletter，2002，33（2）：51-59.

[3] VOGELS W. Eventually Consistent [J]. Queue，2008，6（6）：14-19.

?1994-2014 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

[4] FAY Chang, JEFFREY Dean, SANJAY Ghemawat, et al. A Distributed Storage System for Structured Data [C] //In Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation (OSDI'06). USA: USENIX Association, 2006: 205-218.

[5] 姜龙翔，王鑫，李旭，等。一种大规模 RDF 语义数据的分布式存储方案 [J]. 计算机应用与软件，2011，28 (11)，30-33.
(JIANG Longxiang, WANG Xin, LI Xu, et al. A Distributed Storage Scheme for Large-Scale RDF Semantic Data [J]. Computer Applications and Software, 2011, 28 (11), 30-33. in Chinese)

[6] DAVID Karger, ERIC Lehman, TOM Leighton, et al. Consistent Hashing and Random Trees: Distributed Caching Protocols for Relieving Hot Spots on the World Wide Web [C] //Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing. New York: ACM, 1997: 654-663.

[7] Wikipedia: NoSQL [EB/OL]. [2010-06-20]. http://en.Wikipedia.Org/Wiki/NoSQL.

[8] NoSQL Database [EB/OL]. [2010-06-20]. http://NoSQLNoSQL.Database.Org/.

[9] Apache Cassandra [EB/OL]. [2010-06-20]. http://Cassandra.Apache.Org/.

[10] LAKSHMAN A, MALIK P. Cassandra: Structured Storage System on a P2P Network [C] //Proceedings of the 28th ACM Symposium on Principles of Distributed Computing. Canada: ACM, 2009: 5-5.

Research and Application of Large Data Query Technology Based on NoSQL Database

ZHU Jiansheng, WANG Jianxiong, ZHANG Junfeng

(Institute of Computing Technologies, China Academy of Railway Sciences, Beijing 100081, China)

Abstract: Based NoSQL database theory and different application scenarios, NoSQL database can be divided into three types for high-performance read and write, for documents and for distributed computing. According to the comparative analyses of the advantages and disadvantages of six representative products for these three types of databases, and combining with the demands for large data manipulation in the integrated railway real-name ticketing information analysis system, Cassandra database is chosen as NoSQL database for distributed computing. The technical architecture of integrated railway real-name ticketing information analysis system is proposed based on Cassandra database, and inverted indices are designed to build the query strategies and query processes of travel information for ticket real-name system. The high availability of NoSQL database technology in handling and analyzing large data queries has been verified through performance tests. The bottlenecks of query performance, scalability and investment cost of traditional relational database and data warehouse in applications can be broken through.

Key words: NoSQL database; Cassandra database; Large data processing; Inverted index; Data query

（责任编辑　刘卫华）

?1994-2014 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net