关系数据库中无模式数据存取实现方法*

索 剑，罗中良
（惠州学院计算机科学系，广东 惠州 516007）

摘 要：关系型数据库较好的满足了企业级管理面向事务处理的数据一致性要求，但目前越来越多的 Web2.0 应用使得大量强调高并发下高效存取的非结构化数据出现。对于处理关系型数据库与 NoSQL 模式的融合和扩展问题，该文提出了在传统关系型数据库数据存取过程中，将数据及存取命令重新解构，使数据存取分为模式和无模式两种方式分别进行处理的思想，从而解决了数据一致性和高效存取的矛盾。最后，在同一 MySQL 数据库中对相同数据分别进行模式和无模式环境下并发海量数据插入、查询等操作，测试结果证明了此方法的确有比关系型数据更高的存取效率。

关键词：非结构化数据；关系型数据库；RDBMS；NoSQL；模式；ACID；CAP
中图分类号：TP311.13 **文献标志码：**A **文章编号：**0529－6579（2013）04－0083－06

The Implementation of Schema-Less Data Access in Relational Database

SUO Jian, *LUO Zhongliang*
(Department of Compute Science, Huizhou University, Huizhou 516007, China)

Abstract: Relational database is a very satisfiable transaction processing in enterprise management, however, with the advancing of Web2.0 era, there are more and more people increasingly focus on the exceptional performance of high-concurrency accessing of unstructured data. As to the question of the combination and extension of relational database and NoSQL schema, the paradox of data consistency and high-performance data access can be solved by the re-deconstruct of data and access instruction in the data accessing process of traditional database, which implement the idea of dividing the data access into schema and schema-less process. At last, the test in mass-concurrent insertion, query, and other database operations of the same data have been made in the same MySQL database. The results under schema and schema-less environment prove this solution is more efficient than relational database.

Key words: unstructured data; relational database; RDBMS; NoSQL; schema; ACID; CAP

基于 Web2.0 的应用推动了大数据时代到来，传统关系型数据库面临着高并发读写、高可扩展性、高可用性要求的挑战，大量的数据处理不仅涉及结构化数据，也面向准结构化数据和非结构化数据。随着系统的不断增大，关系型数据库逐渐力不从心，NoSQL 数据库逐渐进入应用领域。

为了承载大规模数据通常采用的是分布式数据库系统，目前比较一致的观点是在分布式系统中一致性（Consistent）、可用性（Available）和分区容错性（Partition-tolerant），很难同时满足[1-4]。RDBMS 的基石是保证服务器端和客户端的一致性，同时保证良好的响应性能和用户体验，但保障跨区域以及良好的扩展性较为困难；NoSQL 运动试图在跨区域系统不断扩展的现实驱动下，满足良好的

* **收稿日期：**2012－12－05
基金项目：广东省科技计划资助项目（2012B010100038）；惠州市科技计划资助项目（2011C020005005）；惠州学院校级特色专业资助项目（TS2011001）；惠州学院校级教学成果培育项目（CG2011008）；惠州学院质量工程资助项目（SZ2012001）
作者简介：索剑（1971 年生），男；E-mail：Rogen@hzu.edu.cn

用户体验，这样只能牺牲即时一致性，通过最终一致性（Eventually Consistancy）满足数据存取需求[5]。将不同特性和机制视为矛盾并不可取，有效地将二者结合起来有更广阔的应用场景。

正是由于两类数据库不同的特点和具体应用场景的约束，关于关系型数据库和 NoSQL 的结合与扩展逐渐成为业界讨论的热点。唐李洋等以分布式多节点架构的索引构建为背景，提出了建立在分布可扩展的数据存储 Cassandra（一种 NoSQL）之上的分布式反向索引，进行扩展性的讨论[6]；姚林等提出将 NoSQL 作为镜像引入关系数据库架构系统以实现分布式存储和改进[7]；张国荣将关系数据库的基础数据通过 MongoDB 非关系型的实现了电子表单系统[8]；基于此方向的研究和应用还有很多[9-14]，事实上，MySQL、Oracle 等已开始在传统关系型数据库的基础上融合 NoSQL 思想的实践。

总体来看，两种方案可以实现数据在此场景下的存储：一是将数据分别存储在关系型数据库和 NoSQL 数据库中，以关系数据库和 NoSQL 数据库相应完整的机制分别驱动支撑；二是将数据存储在关系数据库中，关系型的数据遵守关系数据库的规则，非关系存储的数据通过对关型数据库的无模式改造而实现。本文正是基于后者思想而实现的。

1 关系型数据库存取机制和基本原理

从外部访问到内部磁盘存储，RDBMS 为了保证 ACID（原子性 Atomicity、一致性 Consistency、隔离性 Isolation、持久性 Durability）特性，设计了复杂的多层结构。从数据存取的架构上来看，可以分为三层：访问接口层、逻辑处理层、物理存储层。访问接口层负责为不同类型的数据访问进程提供接口，面向外部链接；逻辑处理层是 RDBMS 的核心，负责查询处理、事务管理、恢复管理、存储管理，由统一的资源管理支撑，实现 ACID 的保障；物理存储层面向磁盘等物理存储设备，处理中间数据和结果数据的外部存储。

在客户端对关系型数据库进行存取数据时，首先建立客户端和服务器的联系，在接收客户端的数据访问指令后，RDBMS 对访问指令进行词法、语法、完整性验证、语义分析，并根据数据字典等资源实施合适的的优化处理、事务管理（包括锁的实现和执行）策略，分析后的操作指令在统一的保障环境下执行，执行后的中间数据、结果等以指定方式存储或返回。

从结构上来看，每一次数据存取，都必须经过

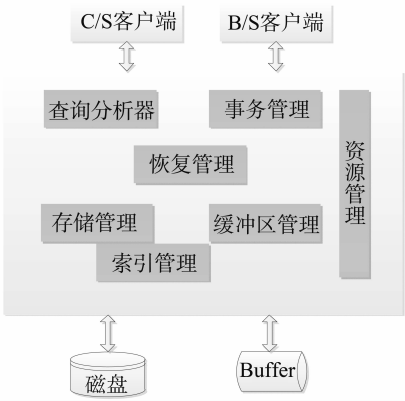

图 1 关系数据库数据存取机制

Fig. 1 The data access mechanism in relational database

上述三层结构，执行从查询处理子系统、事务管理子系统、恢复管理子系统到存储管理子系统的每一步操作。所涉及的索引、静态数据结构、存储引擎、并发控制、连接等基础功能性操作需考虑大量的一致性、准确性问题，从而导致存取效率较低、扩展不易以及分布式实现代价高。

2 NoSQL 的存取机制

NoSQL 不仅仅可以使用 SQL 的方式进行数据管理，还可结合多种数据模型和操作模型。由于产品繁多，同时不像关系数据库那样成熟标准，因此提供了多种的架构方式，通常并没有强制的数据结构约束。但总体思想是类同的，最常见的是基于 Key 值的存储模型，主要有：Key-Values、Key－结构化数据、Key－文档类型、列式存储、BigTable、行式存储等，除此之外还有图结构存储模型。NoSQL 系统通常注重性能和扩展性，而非强事务机制，通常 NoSQL 系统仅提供行级别的原子性保证，也就是说同时对同一个 Key 下的数据进行的两个操作，在实际执行时是会串行的，从而保证每一个 Key-Value 对不会被破坏。NoSQL 适宜使用分布式架构，通过使用 HASH 环算法实现分布存储与拓展[1,7]，保证数据的最终一致性。

3 关系数据库中无模式数据的存取方法

3.1 基本存取思想

通过对关系型数据库分析可以看出，RDBMS 为了保证 ACID 会实施很多保障性措施，若借助 NoSQL 的思想，在 RDBMS 中数据服务器对客户端发送数据与指令执行解释执行的时候，重新改写并实现查询处理功能，使之在处理 SQL 解释、锁管

理、查询优化、解释操作等时选择性执行，如图 2 所示，使数据可以根据选择分两种方式进入执行和操作阶段。一是按原有 RDBMS 中的规范执行，保持一致性但效率较低；二是最小度执行这些保障性内容，效率较高但一致性较低。同时根据需要，以同一思想重建事务管理和数据存储。这样可以加快所有的数据处理。

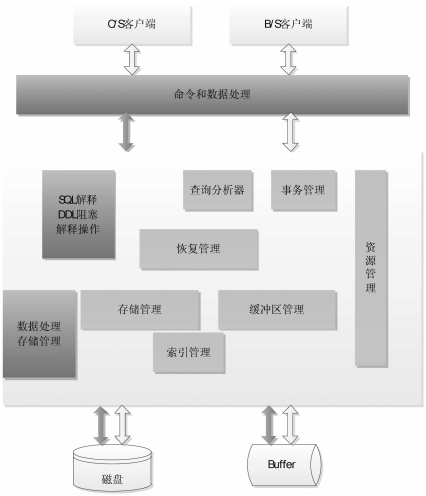

图 2　无模式数据存取思想

Fig. 2　The design of schema-Less data access

这样的存取思想需要解决好两个问题：存取的数据间可以实现分组存取，解决好数据与数据间的关系问题；连接关系型数据库并兼容原有功能与 SQL 语句的操作。

3.2　无模式存取的设计架构

如图 3，按照上述思想，可以将处理划分为三个层次，分别是接口处理层、核心处理层、数据处理层。

3.2.1　*接口处理层*　从外部应用程序获取数据访问的指令和参数，并对指令作解释，对数据模式初始化，做出相应转换；同时作为通道返回最终处理的数据结果集。该层主要由参数收集模块、模式判断模块、初始化处理模块、上层 Helper 管理文件模块、Error Message 等几个模块构成。

参数收集模块对外部发送过来的数据与指令进行收集。主要收集的内容有连接关系型数据库配置参数，包括适配器、host、数据库用户名与密码、数据库名称、进程数量等；数据库模式类型；上层 Helper 方法的调用参数；外部接口需要提供的最基本表结构类型；底层数据结构需要的参数；外部接口传输过来需要操作的数据参数等。

模式判断模块由参数收集模块根据传送的数据

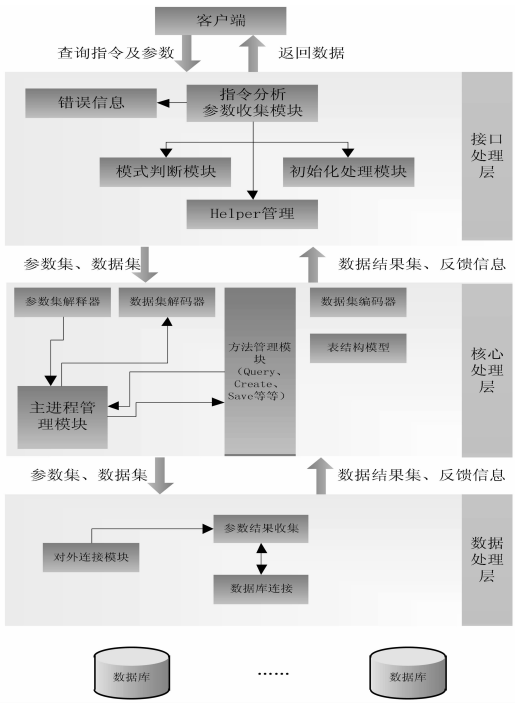

图 3　结构示意图

Fig. 3　Structural diagram

类型来判断触发，若传输 DB type 参数为无模式扩展的类型时，将返回参数收集模块，并继续正常运行；若为普通模式，把此 M－数据模型定为普通 Table 表，按照关系模式进行处理，并返回信息提示给参数收集停止其他运行操作。

初始化处理模块根据参数收集模块传过来的数据参数、类型、索引等几个主要参数在模块内初始化，将直接被数据处理层用于链接关系型数据库与对数据库进行操作。

上层 Helper 管理文件模块针对无模式模型，由参数收集模块发送数据模型参数，Helper 管理模块根据这些参数动态生成 Model 拥有的相应字段方法，而后这些方法将会直接在核心处理层的方法管理模块 other helper 中注册，以便在外部调用。

Error Message 模块用于返回以上操作过程中出现的错误。

3.2.2　*核心处理层*　使这些数据直接保存为 Storage 格式，规避 ACID 特性相应的附加操作；提供其他应用与框架的主要接口层功能，提供多进程处理操作。包括参数集解释处理器、主进程管理模块、方法管理模块、表结构模型、数据集编码和数据集解码等模块。

参数解释处理主要负责解释由上层命令接收发来参数。主要参数包括字段、字段默认值配置、索引表名、进程处理数、调用方法名、集群数据库参

数、M－模型名等。

主进程管理模块对分发的进程进行管理。在原有的关系型数据库模式中，没有采用分布式的数据操作，因为其中的数据都是有关系的，采用分布式操作数据反而会加大数据操作困难与关系掉链[17]。但由于减弱了关系型数据库中数据的关联，所以这些操作数据间都是各自独立的，包括操作过程中也是互不影响的，所以添加了扩展的关系型数据库是能比较简便的采用分发多进程处理操作。Web 应用程序每次调用运行时，默认提供一个主进程运行，通过进程处理数（Process Number）来分发出对应的进程数量。当子进程运行时，主进程等待，待标志的所有子进程运行完毕后，主进程继续运行，并收集各子进程运行结果，所以对外返回数据与信息都是只有通过主进程，子进程只是参与内部运算操作。未被标志的子进程自行运行到任务结束，无需返回任何信息给主进程。

方法管理模块中的方法分为两类：一类是实体数据对象方法，这类方法主要存放在 OtherHelper 模块，属于关系型数据库操作方法。第二类方法是无模式的类方法，这类方法主要存放在 ClassMethod 这个方法管理块中，接口处理层中赋予 M－数据模型的 Helpers 方法主要就是类方法。类方法与对象方法都保存在 Method 块中，便于代码管理与提高代码的可读性。这里提供了 6 种常用的类方法：Create、Update、Save、Query、Delete、Drop。

表结构模型将会根据传来参数 Index、Attribute 与 Model 所构建的两种类型无模式数据存储结构，实体数据存储结构与弱关联索引表结构[15]，同时索引表也建立弱关联关系。用 Primary_ key 表示关系型数据库表主键，id 表示无模式数据实体唯一实体标志，采用 uuid 时间生成格式，contents 表示无模式数据实体存储的字段，created_ at、updated_ at 默认提供时间显示字段。

数据集编码主要的任务是根据数据实体不同需要进行 Json Key-Values 编码，存储到 contents 字段中。数据集编码后，在 Web 应用程序对数据集进行搜索时，需要对返回结果进行数据集解码，以便获取适合外部调用的数据。

3.2.3 *数据处理层* 提供不同关系型数据库连接适配器，与数据库连接并支持将不同数据根据规则保存到分布式的数据仓库集群中。主要包括参数结果收集模块、对外连接模块和数据库连接模块等。

参数结果收集模块分为参数子模块与结果子模块，前者针对上层参数收集与等待传送，后者进行底层分布式数据库集群数据获取的收集，并且提供共享缓存点。

对外连接模块提供了端口的监听方式，当有请求（例如 HTTP 请求）来对端口进行数据操作处理时，这时候会根据操作类型提供不同回调方法，若操作时存储数据，这时候就会对在参数结果收集模块中的参数数据进行分组，然后分发到对外连接模块，然后再通过原有的请求方法把数据发送过去；倘若操作时获取数据，就通过对外连接模块把请求回来的数据暂存在参数结果收集模块中的结果子模块中，等待所有数据都获取完成后，主进程再把最后的数据结果集发送到上层分层结构中。

数据库连接模块分为适配器子模块、SQL 方法扩展子模块、插件子模块三层。适配器子模块针对诸如 Sqlite、Mysql、Oracle、Postgres、Sql server ado 等不同关系型数据库提供不同的适配器用于连接；SQL 方法扩展子模块用于对应不同关系型数据库对标准的 SQL 语言进行的扩展[16]；插件子模块用于在底层扩展新功能提供的协调。

3.3 无模式存取方法的实现

在关系数据库中实现无模式数据的存取流程可用图 4 描述。

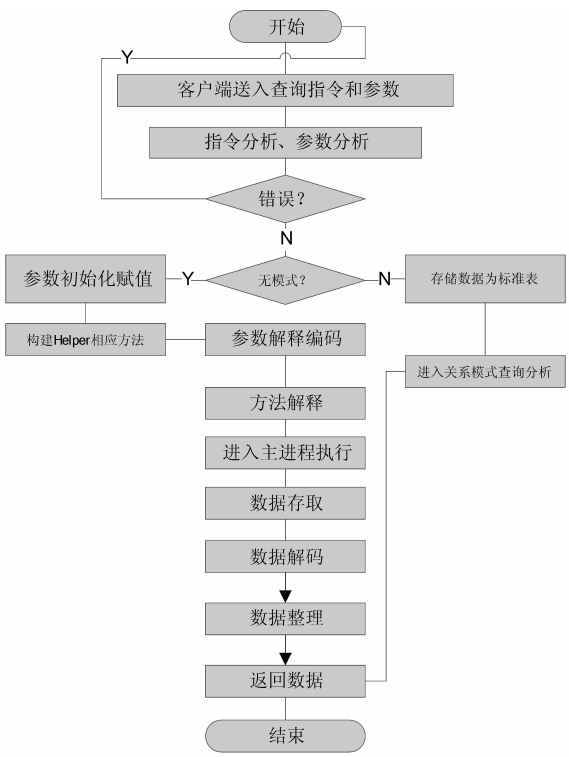

图 4 存取流程图

Fig. 4 Access flow diagram

在 Linux Ubuntu11.10 操作系统环境下，使用关系型数据库 Mysql，通过 Ruby1.9.2 语言实现了上述三层结构，进行了数据库的模拟运行，功能正常。

4　性能测试与对比

在关系数据库中实现了无模式数据的存取后，与关系存取性能做了对比测试。更新测试、查询测试、DDL 阻塞测试描述如下。

测试环境是一台三星笔记本计算机，处理器为 Intel 酷睿 i3，主频 2.2 GHz，物理内存 2 G，SATA 硬盘 500 G。

4.1　数据插入测试

测试模拟微博网站短时间内并发数据分别以传统关系存储方式和无模式扩展方式（进程数为 1）写入，模拟数据量将分为 100 条、1 000 条、100 000 条三类，测试两种方式插入所需的时间，单位为秒（s）。插入的数据库表结构为：Create table materiel（mid int primary key，name vachar（200），models vachar（200），making vachar（200），customer vachar（233），quantity integer）。并发插入 100、1 000、10 000 条记录的测试结果如表 1。

表 1　并发数据插入测试

Table 1　Concurrent insertion test

数据/条	关系存储方式/s	无模式扩展存储/s
100	3.972 4	0.298 1
1 000	40.018 2	1.988 4
10 000	434.119 4	20.540 9

测试结果表明无模式扩展存储的并发插入速度明显优于关系型数据库操作，随着并发数据量逐渐增多，关系型存储方式的数据插入效率明显越来越低。在增加进程数后，无模式扩展以原有效率可支持更多的并发数。

4.2　查询测试

查询是数据库存取的关键指标，短时间响应查询设定关系型模式存储数据和进程数分别为 1、2、5 的无模式扩展针对 50 000、100 000、1 000 000 条记录的查询，其中进程数 2 和 5 针对进程数取了平均值，测试结果如表 2。

从上测试结果可以看出，无模式存储方式的检索速度明显优于关系型存储模式，在针对分布式数据库集群中以及多进程数据搜索中，无模式存储方式可开启多进程处理查询请求，搜索速度将进一步提高。

表 2　查询测试

Table 2　Query test

数据量/条	关系存储/s	无模式扩展进程数 1/s	无模式扩展进程数 2/s	无模式扩展进程数 5/s
50 000	0.029 5	0.008 6	0.005 4	0.002 0
100 000	0.053 7	0.019 4	0.009 9	0.003 4
1 000 000	0.706 9	0.146 3	0.075 3	0.017 4

4.3　DDL 阻塞测试

前已叙及，为了保证 ACID 特性，关系数据库在数据存取前后必须进行加锁解锁处理，这样一旦出现 DDL 相关修改，势必影响存取速度。图 5 是模拟删除索引表的 DDL 阻塞，可以看出在 50 000 条数据集中完成插入的 1 条数据操作所耗费时间为 0.1 s，在删除索引表时 DDL 阻塞情况下完成插入 1 条数据的时间 2.4 s，阻塞等待时间约 2.3 s；相同环境下也做了删除 1 条数据的测试，时间基本等同。若数据集很大，则堵塞时间将会进一步增加。

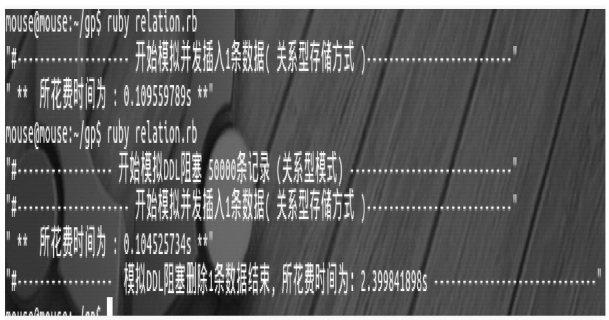

图 5　DDL 阻塞测试 1

Fig. 5　DDL blocking test 1

无模式扩展存取方式的设计方案规避了锁处理的过程，插入和删除相当于对于单表操作，可有效避免关系型数据库的 DDL 阻塞。图 6 是无模式数据测试插入和删除 1 条数据的时间，可以看出没有引起 DDL 阻塞。

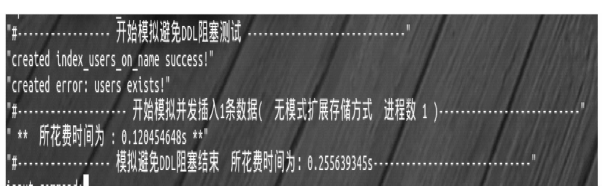

图 6　DDL 阻塞测试 2

Fig. 6　DDL blocking test 2

事实上在关系模型下引发 DDL 阻塞的情况还有很多，例如若表结构发生了改变，不仅会引起 DDL 阻塞，而且还需对数据库表结构进行重新的编码过程，这样既引发效率问题也增加了出现错误的风险。在无模式环境下基于上述思想，只要在 M－模型中添加需要的字段，就可以直接改变属性集，较好的解决了问题。

除了以上基本的存取测试，还做了扩展性的测试。模拟分布式集群下分别并发插入 10 000 条数据，检索 10 000 条记录通过 HTTP 请求发送到 Memcached 的共享缓存点的过程。测试结果表明无模式存储方式的扩展，能支持分布式数据集群，把数据按照定好的规则保存到数据库中，也能在所有的数据库集群中获取响应检索的数据，返回数据结果集。

5 结 语

相比传统关系数据库支持的 ACID 语义约束了大数据量高效访问，NoSQL 让数据库具备了非关系、可水平扩展、可分布和开源等特点，将关系数据库和 NoSQL 模式融合在一起可以提供不同应用场景的需求，可在满足数据一致性的同时提供高效存取和分布式应用和拓展，这种融合可以应用在 Web 应用服务器、内容管理系统、结构化事件记录、移动应用专用的服务器端存储、文件存储等多个方面。本文提出了一种在关系型数据库中构建无模式数据存取的方法，将主要的数据与这些数据的索引分离开来，定义一种新的数据存储模式存储数据，减弱关系型模式数据库中数据间的联系，形成自定义的一种“无模式”持久方案，从而实现关系型模式数据库与无模式结合，进行并行数据存取。提供了在强调一致性、高关系性的场景中采用关系型模式的 ACID 模型，在强调高可用性、高扩展性与多变数据模式的场景采用 BASE（Basis Availability，Soft Status，Eventually Consistancy）模型的一种方案。

参考文献：

[1] WETH C，DATTA A. GutenTag：A multi-term caching optimilized tag query processor for key-value based noSQL storage systems [EB/OL]. http://arxiv. org/pdf/1105. 4452，2011－06，2012－11.

[2] GILBERT S，LYNCH N. Brewer's conjecture and the feasibility of consistent，available，partition-toleram Web services[J]. SI－GACT News，2002，33(2)：51－59.

[3] ERIC Brewer. CAP twelve years later：How the “rules” have changed [EB/OL]. http://www. infoq. com/articles/cap-twelve-years-later-how-the-rules-have-changed；jsessionid＝25A629918DB53CE5EB24C6A7EEF3D01D，2012－05，2012－10.

[4] BANQ. CAP 原理和 BASE 思想[EB/OL]. http://www. jdon. com/37625. 2009－11，2012－11.

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

[6] 唐李洋，倪志伟，李应. 基于 Cassandra 的可扩展分布式反向索引的构建[J]. 计算机科学，2011，6：187－190.

[7] 姚林，张永库. NoSQL 的分布式存储与扩展解决方法[J]. 计算机工程，2012，3：40－42.

[8] 张国荣. 基于关系型与非关系型数据库的电子表单系统设计与实现[D]. 广州：中山大学，2012.

[9] 吕美英，郭显娥. NOSQL 和可扩展的 SQL[J]. 大同大学学报：自然科学版，2012，5；15－18.

[10] 曾凯，曾斌，杨英，等. 扩展 SQL 跟踪数据技术在数据库性能诊断上的应用[J]. 计算机应用与软件，2006，1：128－130.

[11] 王锐. 基于 MongoDB 的关系网络分析技术研究与应用[D]. 长沙：国防科学技术大学，2011.

[12] 刘一梦. 基于 MongoDB 的云数据管理技术的研究与应用[D]. 北京：北京交通大学，2012.

[13] 蔡柳青. 基于 MongoDB 的云监控设计与应用[D]. 北京：北京交通大学，2011.

[14] 王光磊. MongoDB 数据库的应用研究和方案优化[J]. 中国科技信息，2011，20：93－94，96.

[15] BRET Taylor. How friend feed uses MySQL to store schema-less data [EB/OL]. http://www. zhmy. com/jinghuarizhi_2010_08_663. html，2009－2，2012－10.

[16] 周渊，王力生. Mysql 中 InnoDB 存储引擎在 NUMA 系统上的优化[J]. 科技传播，2011. 1：155－156.

[17] 冯祖洪. MySQL 的分布式数据库访问法[J]. 计算机应用，2002，8：4－6.