摘 要

随着因特网应用和计算机技术的飞速发展，数据库逐渐成为信息系统的核心部分并广泛应用于企业、金融机构、政府及国防等各个领域。但是传统的关系数据库在面对日益增多的大量数据时暴露了很多问题，不能对其高效、实时的处理。对系统提出的新需求，传统的关系数据库并不能很好的解决。非关系型数据库是为弥补关系数据库的不足而产生的，OrientDB数据库就是非关系型数据库的一种。目前现有的非关系数据库对大数据量数据的处理能力很强但是在在很多关键领域，关系型数据库又是非关系型数据库无法代替的。如何结合关系型数据库和非关系型数据库的优点成为当前数据库领域的关键问题。

通过对分布式理论、关系数据库理论及非关系型数据库技术的学习和研究，本文基于Java语言实现了一个分布式的关系型数据库JSQL，JSQL是一个将关系数据库技术，非关系型数据库技术和分布式技术相结合的产物。JSQL采用常用的客服端-服务器架构，分为客服端和服务器端，服务器端又分为分布式管理节点和分布式数据库集群。分布式管理节点实现了数据库服务器的负载均衡和监控管理功能。分布式数据库集群节点主要包含五大模块，网络模块接收前端客服端的连接请求，对客服端进行认证和授权，并对连接进行管理，并实现基于Mysql的通信协议； Sql的解析和执行模块接收客户端的SQL请求，然后解析和执行数据库存储的调用，返回执行结果；审计模块是存储和分析所有对数据库的更改情况，向分布式管理节点提供审计数据；数据库引擎模块利用了非关系型Orientdb数据库引擎，实现可靠的数据存储；分布式模块利用hazlcast实现了数据库集群，本文提出了分布式多版本并发控制方法，用来解决分布式数据一致性问题。基于以上功能模块，ＪＳＱＬ具有高可用性，可扩展性，负载均衡等特性，同时从数据库底层考虑了数据库安全审计需求，加入了数据审计图形化界面显示审计结果。

论文对系统进行了功能和性能测试。功能测试结果表明，系统在功能上符合分布式数据库的基本要求，审计系统的功能也达到本论文的要求。论文通过对性能测试结果进行分析，认为系统的性能基本达到本论文的要求。但是本系统对复杂SQL语句的支持还不是很完善，最后提出了改进的方案。

关键词：分布式数据库，关系数据库，安全审计，Mysql，非关系型数据库

ABSTRACT

With the rapid development of Internet application and computer technology, the database has gradually become the core of information system Points and widely used in enterprises, financial institutions, government and national defense and other fields. However, the traditional relational database is facing increasing When exposed to a large number of data a lot of problems, can not be efficient and real-time processing. The new needs of the system Seeking, the traditional relational database can not be solved very well. Non-relational database is to make up for the lack of relational database, the OrientDB database Is a non-relational database. At present, the existing non-relational database has great ability to handle large amounts of data But in many key areas, relational databases are non-relational databases that can not be replaced. How to combine the advantages of relational databases and non-relational databases has become the key issue in the current database area.

Through the study of distributed theory, relational database theory and non-relational database technology, This article based on the Java language to achieve a distributed relational database JSQL, JSQL is a relational database technology, The combination of non-relational database technology and distributed technology. JSQL common client-server - server architecture, divided into customer service and server-side, Server-side is divided into distributed management nodes and distributed database nodes. Distributed management nodes to achieve a database server load balancing and monitoring and management functions. Distributed database node contains five major modules, The network module accepts the connection request of the front-end application, authenticates and authorizes the client-end, and manages the connection, And realize the communication protocol based on Mysql, so easy to migrate to Mysql users to the system; Sql parsing and execution module to accept the client's SQL request, and then parse and execute the database stored call, Return the execution result The audit module is to store and analyze all changes to the database, the distributed management node to provide audit data; The database engine module makes use of the non-relational OrientDB database engine for reliable data storage; Distributed modules use hazlcast to implement a database cluster. Based on the above functional modules, JSQL has high availability, scalability, load balancing and other characteristics, while taking into account the database security audit

needs from the bottom of the database, Joined the data audit graphical interface shows the audit results.

The thesis has carried on the function and the performance test to the system. The result of function test shows that the system conforms to the basic requirements of distributed database functionally, and the function of auditing system also meets the requirements of this dissertation. Papers through the performance test results analysis, that the performance of the system basically meet the requirements of this paper. However, the system of complex SQL statement support is not perfect, and finally proposed an improved program.

Keywords: Distributed database, relational database, security audit, Mysql, Non-relational database

目 录

第一章 相关理论和技术

本节介绍数据库和分布式数据库有关的理论知识，以及本系统所需要用到的关键算法和技术。

1.1 数据库系统

1.1.1 数据库系统概念

数据库管理系统是一种操纵和管理数据库的大型软件，用于建立、使用和维护数据库。它对数据库进行统一的管理和控制，以保证数据库的安全性和完整性。用户通过数据库系统访问数据库中的数据，数据库管理员也通过数据库系统进行数据库的维护工作。它可使多个应用程序和用户用不同的方法在同时或不同时刻去建立，修改和询问数据库。大部分数据库系统提供对数据的追加、删除等操作。数据库管理系统是数据库系统的核心，是管理数据库的软件。数据库管理系统就是实现把用户意义下抽象的逻辑数据处理，转换成为计算机中具体的物理数据处理的软件。有了数据库管理系统，用户就可以在抽象意义下处理数据，而不必顾及这些数据在计算机中的布局和物理位置。

1.1.2 关系数据库系统

1.1.2.1 关系模型

用于数据库管理的关系模型是基于谓词逻辑和集合论的一种数据模型，广泛被使用于关系型数据库之中。最早于1969年由埃德加·科德提出。关系模型的基本假定是所有数据都表示为数学上的关系，关系模型是采用二维表格结构表达实体类型及实体间联系的数据模型. 关系模型允许设计者通过数据库规范化的提炼，去建立一个信息的一致性的模型。

基本的关系建造块是域或者叫数据类型。元组是属性的有序多重集，属性是域和值的有序对。关系变量是域和名字的有序对（序偶）的集合，它充当关系的表头。关系是元组的集合。尽管这些关系概念是数学上的定义的，它们可以宽松的映射到传统数据库概念上。表是关系的公认的可视表示；元组类似于行的概念。关系模型的基本原理是信息原理：所有信息都表示为关系中的数据值。所以，关系变量在设计时刻是相互无关联的；反而，设计者在多个关系变量中使用相同的域，如果一个属性依赖于另一个属性，则通过参照完整性来强制这种依赖性。

关系数据库，是建立在关系数据库模型基础上的数据库，借助于集合代数等概念和方法来处理数据库中的数据，同时也是一个被组织成一组拥有正式描述性的表格，该形式的表格作用的实质是装载着数据项的特殊收集体，这些表格中的数据能以许多不同的方式被存取或重新召集而不需要重新组织数据库表格。关系数据库的定义造成元数据的一张表格或造成表格、列、范围和约束的正式描述。每个表格包含用列表示的一个或更多的数据种类。每行包含一个唯一的数据实体，这些数据是被列定义的种类。当创造一个关系数据库的时候，你能定义数据列的可能值的范围和可能应用于那个数据值的进一步约束。

1.1.3 事务与并发控制

1.1.3.1 事务

事务象征内执行工作单元的数据库管理系统中独立于其他事务的连贯和可靠的方式针。一个事务通常代表数据库中的任何变化。数据库环境中的事务有两个主要目的：为了提供可靠的工作单元，允许从故障中正确恢复，并保持数据库一致，即使在系统故障的情况下，执行停止，数据库上的许多操作仍未完成，状态不清楚。在同时访问数据库的程序之间提供隔离。如果没有提供这种隔离，程序的结果可能是错误的。根据定义，数据库事务必须是原子的，一致的，隔离的和持久的。数据库从业者通常使用缩写ACID来引用数据库事务的这些属性。交易提供了“全或无”的命题，指出在数据库中执行的每个工作单元必须完整或完全没有任何影响。此外，系统必须将每个事务与其他事务隔离开来，结果必须符合数据库中的现有约束，并且成功完成的事务必须写入持久存储。

将数据完整性视为最重要的数据库和其他数据存储通常包括处理事务以维护数据完整性的能力。单个事务由一个或多个独立的工作单元组成，每个读取和或写入数据库或其他数据存储的信息。当发生这种情况时，确保所有此类处理使数据库或数据存储处于一致状态时，通常很重要。

1.1.3.2 并发控制

在信息技术和计算机科学领域，特别是在计算机编程，操作系统，多处理器和数据库领域，并发控制可以确保并发操作的正确结果，同时尽快获得这些结果。计算机系统，软件和硬件都由模块或组件组成。每个组件被设计为正确地操作，即遵守或符合某些一致性规则。当通过消息传递或通过共享访问的数据同时进行操作的组件时，某个组件的一致性可能被另一个组件违反。并发控制的一般领域提供了规则，方法，设计方法和理论以保持组件在交互时并发运行的一致性，从

而保持整个系统的一致性和正确性。将并发控制引入系统意味着应用操作约束，这通常会导致某些性能下降。操作一致性和正确性应尽可能高效地实现，而不会将性能降低到合理的水平以下。与更简单的顺序算法相比，并发控制可能需要大量额外的复杂性和并发算法的开销。例如，并发控制失败可能导致数据损坏，从而导致读取或写入操作受损。

在并发控制数据库管理系统，其它事务对象和相关的分布式应用程序（例如，网格计算和云计算）确保数据库事务被执行同时在不违反各数据库的数据完整性。因此，并发控制是任何系统中的正确性的基本要素，其中两个数据库事务或更多的数据库事务或多个时间重叠执行，可以访问相同的数据，例如实际上在任何通用数据库系统中。因此，自20世纪70年代初出现数据库系统以来，已经积累了大量相关研究。上述参考文献中提出了一个完善的数据库系统并发控制理论：序列化理论，可以有效地设计和分析并发控制方法和机制。中提出了一种用于抽象数据类型的原子事务并发控制的替代理论。

为了确保正确性，一个DBMS通常可以保证只有序列化的交易时间表中产生，除非串行化是故意轻松提高性能，但只有在应用程序的正确性不受损害的情况。为了在失败（中止）事务（由于许多原因总是可能发生）的情况下维护正确性，计划也需要具有可恢复性属性。DB MS还保证不会导致提交的事务的影响，并且不会中止任何影响（回滚）交易保留在相关数据库中。整体交易表征通常由以下ACID规则汇总。随着数据库已经成为分布，或在分布式环境中进行合作需要，并发控制机制有效配置受到格外的关注。下面是并发控制的主要方法：

1.锁定

通过分配给数据的锁控制对数据的访问。对另一个事务锁定的数据项的事务的访问可能被阻止，直到锁定释放。

2.串行化

检查计划图中的周期并通过中止来破坏它们。

3.时间戳排序

为事务分配时间戳，并按时间戳顺序控制或检查对数据的访问。

4.承诺排序，控制或检查交易的提交事件的时间顺序与其各自的优先顺序兼容。

5.多分支并发控制

通过在每次写入对象时生成新版本的数据库对象，并根据调度方式允许事务对每个对象的最后相关版本的读取操作来增加并发性和性能。

6.索引并发控制

将访问操作同步到索引，而不是用户数据。专业方法提供了显着的性能提升。

每个事务都为其访问的数据维护一个私有工作空间，只有在事务提交之后，其更改的数据才会在事务外部变得可见。这种模式在许多情况下提供了不同的并发控制行为，并带来好处。

1.1.4 NoSQL数据库系统

NoSQL，泛指非关系型的数据库。随着互联网web2.0网站的兴起，传统的关系数据库在应付web2.0网站，特别是超大规模和高并发的SNS类型的web2.0纯动态网站已经显得力不从心，暴露了很多难以克服的问题，而非关系型的数据库则由于其本身的特点得到了非常迅速的发展。NoSQL数据库的产生就是为了解决大规模数据集合多重数据种类带来的挑战，尤其是大数据应用难题。虽然NoSQL的流行与火起来才短短一年的时间，但是不可否认，现在已经开始了第二代运动。尽管早期的堆栈代码只能算是一种实验，然而现在的系统已经更加的成熟、稳定。不过现在也面临着一个严酷的事实：技术越来越成熟——以至于原来很好的NoSQL数据存储不得不进行重写，也有少数人认为这就是所谓的2.0版本。该工具可以为大数据建立快速、可扩展的存储库。

对于NoSQL并没有一个明确的范围和定义，但是他们都普遍存在下面一些共同特征：

1.不需要预定义模式：不需要事先定义数据模式，预定义表结构。数据中的每条记录都可能有不同的属性和格式。当插入数据时，并不需要预先定义它们的模式。

2.无共享架构：相对于将所有数据存储的存储区域网络中的全共享架构。NoSQL往往将数据划分后存储在各个本地服务器上。因为从本地磁盘读取数据的性能往往好于通过网络传输读取数据的性能，从而提高了系统的性能。

3.弹性可扩展：可以在系统运行的时候，动态增加或者删除结点。不需要停机维护，数据可以自动迁移。

4.分区：相对于将数据存放于同一个节点，NoSQL数据库需要将数据进行分区，将记录分散在多个节点上面。并且通常分区的同时还要做复制。这样既提高了并行性能，又能保证没有单点失效的问题。

5.异步复制：和RAID存储系统不同的是，NoSQL中的复制，往往是基于日志的异步复制。这样，数据就可以尽快地写入一个节点，而不会被网络传输引起迟延。缺点是并不总是能保证一致性，这样的方式在出现故障的时候，可能会丢失少量的数据。

6.BASE：相对于事务严格的ACID特性，NoSQL数据库保证的是BASE特性。BASE是最终一致性和软事务。

1.1.4.1 OrientDB介绍

OrientDB是一种NoSQL数据库，虽然它不是关系数据库系统，但是它底层的存储引擎支持事务。所以本系统在后面就直接用的这个这个存储引擎减少系统的开发时间，同时也增加系统的稳定性。

1.2 分布式数据库

1.2.1 分布式数据库概述

分布式数据库是一个数据库，其中存储设备没有全部连接到一个共同的处理器。它可以存储在位于相同物理位置的多台计算机中; 或者可以分散在互连计算机的网络上。与其中处理器紧密耦合并构成单个数据库系统的并行系统不同，分布式数据库系统由松散耦合的站点组成，共享没有物理组件。

1.2.2 分布式数据库的特点

1.数据独立性

分布式数据库系统中，每个系统节点都具有数据独立性。数据独立性包括了数据的逻辑独立性和物理独立性，逻辑独立性是指用户的应用程序与数据库的逻辑结构是相互独立的，即当数据的逻辑结构改变时，用户程序也可以不变；物理独立性是指用户的应用程序与存储在磁盘上的数据库中数据是相互独立的。即，数据在磁盘上怎样存储由 DBMS 管理，用户程序不需要了解，这样当数据的物理存储改变了，应用程序不用改变。

2.分布透明性

分布式数据库还具有分布透明性，该特性使用户不必关心数据的逻辑分片，不必关心数据物理位置分布的细节及局部场地上数据库支持哪种数据模型。分布透明性的优点显而易见；有了分布透明性，用户的应用程序书写起来就如同数据没有分布一样，当数据从一个场地移到另一个场地时，不必改写应用程序，当增加某些数据的重复副本时，也不必改写应用程序。

3.节点自治性

构成分布式数据库的每一个物理数据库都是独立的，可以由数据库管理员分别进行管理，如同一个非网络本地数据库。

4.复制透明性

用户不用关心数据库在网络中各个节点的复制情况。被复制的数据，一般更新都由系统自动完成，在分布式数据库系统中，可以把一个场地的数据复制到其他场地存放，应用程序可以使用复制到本地的数据在本地完成分布式操作，避免通过网络传输数据，从而提高了系统的运行和查询效率。但是对于复制数据的更新操作，就要涉及到对所有复制数据的更新。

5.易于扩展性

在大多数网络环境中，单个数据库服务器最终会不满足使用。如果服务器软件支持透明的水平扩展，那么就可以增加多个服务器来进一步分布数据和分担处理任务。

1.2.3 数据分布和负载均衡

分布式系统如何拆解输入数据，将数据分发到不同的机器中。下面将介绍几种不同的数据分布方式。

1.2.3.1 哈希分布

哈希方式是最常见的数据分布方式,其方法是按照数据的某一特征计算哈希值, 并将哈希值与机器中的机器建立映射关系,从而将不同哈希值的数据分布到不同的机器上。只要哈希散列性比较好，数据就能均匀到分发到不同机器中。同时,需要管理的元信息很少,只需要知道哈希函数和模（一般是机器总数）。但是有个明显的缺点,扩展性很差。如果我想把集群规模扩大,可能所有的数据需要被重新迁移。针对哈希方式扩展性差的问题,一种思路是不再简单的将哈希值与机器做除法取模映射,而是将对应关系作为元数据由专门的元数据服务器管理。访问数据时,首先计算哈希值并查询元数据服务器,获得该哈希值对应的机器。同时,哈希值取模个数往往大于机器个数,这样同一台机器上需要负责多个哈希取模的余数。不过，需要管理的元数据就多了。

1.2.3.2 顺序分布

按数据范围分布是另一个常见的数据分布式, 将数据按特征值的值域范围划分为不同的区间,使得集群中每台(组)服务器处理不同区间的数据。对于上面哈希方式某个用户数据特别多我们就可以通过采用数据范围分布解决,动态划分范围空间,实现负载均衡。按数据范围分布数据需要记录所有的数据分布情况。一般的,往往需要使用专门的服务器在内存中维护数据分布信息, 称这种数据的分布信息为一种元数据。实际工程中,一般也不按照某一维度划分数据范围,而是使用全部数据划分范围, 从而避免数据倾斜的问题。使用范围分布数据的方式的最大优点就是可以灵活的根据数据量的具体情况拆分原有数据区间, 拆分后的数据区间可以迁移到其他机器,一旦需要集群完成负载均衡时, 与哈希方式相比非常灵活。另外,当集群需要扩容时,可以随意添加机器,而不限为倍增的方式, 只需将原机器上的部分数据分区迁移到新加入的机器上就可以完成集群扩容。而缺点就是元数据可能会成为瓶颈。

1.2.3.3 一致性哈希

一致性哈希的基本方式是使用一个哈希函数计算数据或数据特征的哈希值, 令该哈希函数的输出值域为一个封闭的环,即哈希函数输出的最大值是最小值的前序。将节点随机分布到这个环上,每节点负责处理从自己开始顺时针至下一个节点的全部哈希值域上的数据。

一致性哈希的优点在于可以任意动态添加、删除节点,每次添加、删除一个节点仅影响一致性哈希环上相邻的节点。但也有很明显的缺点,随机分布节点的方式使得很难均匀的分布哈希值域, 尤其在动态增加节点后,即使原先的分布均匀也很难保证继续均匀,由此带来的另一个较为严重的缺点是 ,当一个节点异常时,该节点的压力全部转移到相邻的一个节点,当加入一个新节点时只能为一个相邻节点分摊力。为此，一种改进是引入“虚节点”的概念。系统初始时就创建许多虚节点, 虚节点的个数一般远大于未来集群中机器的个数, 将虚节点均匀分布到一致性哈希值域环上,其功能与基本一致性哈希算法中的节点相同。为每个节点分配若干虚节点。操作数据时,首先通过数据的哈希值在环上找到对应的虚节点,进而查找元数据找到对应的真实节点。这样,一旦某个节点不可用,该节点将使得多个虚节点不可用,从而使得多个相邻的真实节点负载失效节点的压里。同理,一旦加入一个新节点,可以分配多个虚节点,从而使得新节点可以负载多个原有节点的压力,从全局看,较容易实现扩容时的负载均衡。

1.2.3.4 负载均衡

在计算中，负载平衡可以改善多个计算资源（如计算机，计算机集群，网络链接，中央处理单元或磁盘驱动器）之间的工作负载分配。负载平衡旨在优化资源使用，最大化吞吐量，最小化响应时间，并避免任何单一资源的过载。使用负载均衡而不是单个组件的多个组件可以通过冗余来增加可靠性和可用性。负载平衡通常涉及专用软件或硬件，如多层交换机或域名系统服务器进程。有很多种负载均衡的算法可用，下面是常用的一些算法：

1.轮询法

轮询法是负载均衡中最常用的算法，它容易理解也容易实现。轮询法是指负载均衡服务器将客户端请求按顺序轮流分配到后端服务器上，以达到负载均衡的目的。

2.加权轮询法

简单的轮询法并不考虑后端机器的性能和负载差异。给性能高、负载低的机器配置较高的权重，让其处理较多的请求；而性能低、负载高的机器，配置较低的权重，让其处理较少的请求。加权轮询法可以很好地处理这一问题，它将请求顺序且按照权重分派到后端服务器。

3.最小连接数法

即使后端机器的性能和负载一样，不同客户端请求复杂度不一样导致处理时间也不一样。最小连接数法根据后端服务器当前的连接数情况，动态地选取其中积压连接数最小的一台服务器来处理当前的请求，尽可能提高后端服务器的利用效率，合理地将请求分流到每一台服务器。为什么根据连接数可以合理地利用服务器处理请求呢？考虑一个客户端请求的处理逻辑较复杂，需要服务器的处理时间较长，由于客户端需要等待服务器的响应，故需要保持与服务器的连接，这样一来，客户端就需要与服务器保持较长时间的连接。

4.随机法

随机法也很简单，就是随机选择一台后端服务器进行请求的处理。由于每次服务器被挑中的概率都一样，客户端的请求可以被均匀地分派到所有的后端服务器上。

5.源地址哈希法

源地址哈希的思想是根据获取客户端的IP地址，通过哈希函数计算得到的一个数值，用该数值对服务器列表的大小进行取模运算，得到的结果便是客服端要访问服务器的序号。采用源地址哈希法进行负载均衡，同一IP地

址的客户端，当后端服务器列表不变时，它每次都会映射到同一台后端服务器进行访问。如果后端服务器是一缓存系统，当后端服务器增加或者减少时，采用简单的哈希取模的方法，会使得命中率大大降低，这个问题可以采用一致性哈希的方法来解决。

1.2.4 数据复制和一致性

1.2.4.1 复制的概述

复制几乎是构成分布式系统，尤其是分布式存储和分布式数据库的关键所在，那么本文就来综合谈论下复制技术。

简单说复制本身可以分为同步复制和异步复制，两者的区别在于前者需要等待所有副本返回写入确认，而后者只需要一个返回确认即可。从用途上，复制可以分为两类，一类用于确保不同副本的表现行为一致，另一类则用于允许不同副本之间的数据差异，先来看看前者。有若干种手段用于确保不同副本之间的状态一致。

第一种叫主从复制。主从之间可以是异步复制，也可以是同步复制。例如MySQL，在默认情况下采用异步复制，异步复制容易引起数据丢失，比如主从结构中，主节点的写入请求还没有复制到从节点就挂了，当从节点被选为新的主节点之后，在这之前写入没有同步的数据就会被丢失。即便采用了同步复制，也只能提供相对较弱的基本保障，考虑如下情形：主接收写入请求然后发到从节点，从节点写入成功后并发送确认给主，如果此时主节点正准备发送确认信息给客户端时挂了，那么客户端就会认为提交失败，可是从节点已经提交成功了，如果这是从节点被提升为主，那么就出现问题了。在主从复制结构里，异步复制相比同步复制具备更高的吞吐量和更低延迟，因此，结合同步和异步复制是一个常见选项。第三种则引入分区一致性算法Paxos，又叫复制状态机。复制状态机在数据库开发的很多领域都可以遇到，比如Google Megastore，针对不同分区的每次提交采用复制状态机来确保每个分区的全局事务提交时序； Google Spanner在单分区内也采用了类似的设计。复制状态机主要用于满足两点需求：客户端在面对任何一个副本时都具备完全一致的访问行为；每个副本在执行请求时都需要按照完全一致的顺序来进行。

1.2.4.2 一致性和可用性

在理论计算机科学中，CAP定理，又被称作布鲁尔定理，它指出对于一个分布式计算系统来说，不可能同时满足以下三点：

1.一致性

2.可用性

3.容忍网络分区

根据定理，分布式系统只能满足三项中的两项而不可能满足全部三项。理解CAP理论的最简单方式是想象两个节点分处分区两侧。允许至少一个节点更新状态会导致数据不一致，即丧失了C性质。如果为了保证数据一致性，将分区一侧的节点设置为不可用，那么又丧失了A性质。除非两个节点可以互相通信，才能既保证C又保证A，这又会导致丧失P性质。

CAP理论在互联网界有着广泛的知名度，知识稍微宽泛一点的工程师都会把其作为衡量系统设计的准则。大家都非常清楚地理解了CAP：任何分布式系统在可用性、一致性、分区容错性方面，不能兼得，最多只能得其二，因此，任何分布式系统的设计只是在三者中的不同取舍而已。

CAP还有一个方面很多人认识不清，那就是放弃一致性其实有隐藏负担，即需要明确了解系统中存在的不变性约束。满足一致性的系统有一种保持其不变性约束的自然倾向，即便设计师不清楚系统中所有的不变性约束，相当一部分合理的不变性约束会自动地维持下去。相反，当设计师选择可用性的时候，因为需要在分区结束后恢复被破坏的不变性约束，显然必须将各种不变性约束一一列举出来，可想而知这件工作很有挑战又很容易犯错。放弃一致性为什么难，其核心还是“并发更新问题”，跟多线程编程比顺序编程难的原因是一样的。

1.3 算法和技术

1.3.1 LBLCR算法

负载均衡在多节点之间按照一定的策略分发网络或计算处理负载, 提供了一种廉价而有效的方法来扩展服务器带宽, 增加吞吐量, 提高数据处理能力, 同时又可以避免单点故障。负载均衡包括静态负载平衡和动态负载平衡。只是利用系统负载的平均信息, 而忽视系统当前的负载状况的方法被称为静态负载均衡; 根据系统当前的负载状况来调整任务划分的方法被称为动态负载均衡。目前, 静态负载均衡已越来越不能满足需要, 动态负载均衡技术有取而代之的趋势。分布式局部性最少连接算法是本系统所采用的动态负载均衡算法，该算法(简称 LBLCR) 是针对目

标 IP 地址的负载均衡, 主要用于 Cache 集群系统。与 LBLC 算法的不同之处在于, 它要维护从一个目标 IP 地址到一组服务器的映射, 而不是从一个目标 IP 地址到一台服务器的映射。 LBLCR 算法首先找出目标 IP 地址对应的服务器组, 按“最小连接” 原则从该服务器组中选出一台服务器, 若服务器没有超载, 将请求发送到该服务器; 若服务器超载, 则按“最小连接”原则从整个集群中选出一台服务器, 将该服务器加入到服务器组中, 将请求发送到该服务器。另外, 若该服务器组有一段时间未被修改, 则将最忙的服务器从服务器组中删除, 以降低复制的程度。

1.3.2 副本和分布式MVCC技术

1.3.2.1 副本的概念

副本指在分布式系统中为数据或服务提供的冗余。对于数据副本指在不同的节点上持久化同一份数据，当出现某一个节点的存储的数据丢失时，可以从副本上读到数据。数据副本是分布式系统解决数据丢失异常的唯一手段。另一类副本是服务副本，指数个节点提供某种相同的服务，这种服务一般并不依赖于节点的本地存储，其所需数据一般来自其他节点。

1.3.2.2 副本一致性

分布式系统通过副本控制协议，使得从系统外部读取系统内部各个副本的数据在一定的约束条件下相同，称之为副本一致性。副本一致性是针对分布式系统而言的，不是针对某一个副本而言。分布式系统为了提高可用性，总是不可避免的使用副本的机制，从而引发副本一致性的问题。

1.3.2.3 分布式 MVCC

分布式 MVCC技术不但能解决分布式副本一致性问题，还能实现分布式事务。假设在一个分布式系统中，更新操作以事务进行，每个事务包括若干个对不同节点的不同更新操作。更新事务必须具有原子性，即事务中的所有更新操作要么同时在各个节点生效，要么都不生效。假设不存在并发的事务，即上一个事务成功提交后才进行下一个事务。

基于 MVCC 的分布式事务的方法为：为每个事务分配一个递增的事务编号，这个编号也代表了数据的版本号。当事务在各个节点上执行时，各个节点只需记录更新操作及事务编号，当事务在各个节点都完成后，在全局元信息中记录本次事务的编号。在读取数据时，先读取元信息中已成功的最大事务编号，再于各个节点上读取数据，只读取更新操作编号小于等于最后最大已成功提交事务编号的操作，并将这些操作应用到基础数据形成读取结果。

1.4 本章小结

本节介绍数据库有关的理论知识。包括硬件知识和数据库事务相关理论，以及分布式系统系统有关的算法和协议。另外，本章最后讨论了目前几种成熟的分布式的系统，作为实例学习和学习研究。

致 谢

时间过的真快，转眼间，2年多的研究生生活就快要结束了。回顾自己的研究生学习生涯，感慨万千。论文的完成，除了自己的努力以外，更离不开老师和学弟们的帮助，在论文成稿之际，衷心感谢给予自己悉心指导和热情帮助的各位老师和学弟们。

论文的完成，首先要感谢我的校内导师曹晟教授，在整个论文的的写作过程中，曹老师都给予了我很大的关心和帮助。从作者毕业论文的选题、写作一直到最终完成的过程中，曹老师都是在百忙的工作中以一贯认真负责的态度认真仔细阅读作者的论文，给予作者耐心的指导，使得论文能够顺利的完成。他严肃的科学态度，严谨的治学精神，以及精益求精的工作作风，深深地感染和激励着我。

在这一年多的时间里，我还要感谢我的学弟们。感谢你们陪我一起开发这个分布式数据库系统，我们一起学习，一个努力，才能让本系统顺利完成。任何一个系统都要靠团队合作，更何况分布式系统这个既具有挑战性的工程项目，没有你们的帮助，就不可能按时完成这个系统。对学弟们的帮助，在此表示非常的感谢。

最后，作者非常感谢负责评审论文的教师、专家和教授，感谢你们认真负责的阅读论文，感谢你们为论文提出的宝贵意见和建议。