十年一日，深入成就深度
业精于专，专注成就专业

数字有机体工作库及大规模存储与管理系统

用户手册（V3.4.7）

成都天心悦高科技发展有限公司

2015 年 10 月

版权声明

成都天心悦高科技发展有限公司

地址：成都市武侯区棕南小区

电话：028-83318559

邮编：610054

目录

1 引言

1.1 编写约定

非常感谢您使用成都天心悦高科技发展有限公司的产品，本公司将竭诚为您提供最好的服务。

本手册假定用户能够理解并使用 Linux 的基本 shell 命令。

文中出现的‘#’号表示数字有机体系统的命令行提示符。

文中没有特别说明是批量同步数据库时，默认表示实时同步数据库。

命令格式描述中的斜体字表示应由用户填充的部分，”[ ]”表示命令中可选的参数。

为了阅读方便，文档以灰底黑框的形式呈现某些重要的配置操作。不过，由于数字有机体系统和 Windows 采用不同的字符集和文本规范，请不要直接拷贝文档中的命令行或者配置行到数字有机体系统中，请重新输入。

因软件更新，本手册可能包含技术上不准确的地方或文字错误。

本手册的内容将做定期的更新，恕不另行通知；更新的内容将会在本手册的新版本中加入。

本公司随时会改进或更新本手册中描述的产品或程序。

1.2 如何使用本手册

本手册的阅读对象为数字有机体工作库的安装、管理、维护和使用人员。安装人员需阅读第三章，管理维护人员需要阅读第四、第五、第六和第十三章，使用人员需要仔细阅读第三章以后的各个章节。第十五章提供常见问题解决方法。

数字有机体工作库（DosSQL）是参照 MySQL 开发而来。为了兼容 MySQL 的各种应用，它充分继承了 MySQL 的各种语法。本手册没有提及 MySQL 语法的使用方法，详细情况，请参阅《MySQL 参考手册》。如果有 DosSQL 不支持，但 MySQL 又支持的特性，请参阅第十四章的“限制与约束”。

1.3 相关文档说明

数字有机体系统包括数字有机体工作平台和数字有机体工作库，本文档是数字有机体工作库的用户手册。相关的其它文档还有：

- 有关数字有机体工作平台的使用请参阅《数字有机体工作平台及抗毁容灾系统用户手册》；
- 有关如何在数字有机体工作平台上开发应用程序，请参考《数字有机体工作平台及抗毁容灾系统开发手册》；

- 有关如何在数字有机体工作库开发应用程序，请参考《数字有机体工作库及大规模存储与管理系统开发手册》。

1.4 术语

数据库系统: 数据库系统是指在计算机系统中引入数据库后的系统，一般由数据库、数据库管理系统（及其开发工具）、应用系统、数据库管理员构成。其中数据库是长期存储在计算机内，有组织、可共享的大量数据的集合。数据库中的数据按一定的数据模型组织、描述和存储，具有较小的冗余度、较高的数据独立性和易扩展性，并可为各种用户共享。数据库管理系统是为数据库的建立、使用和维护而配置的软件。它建立在操作系统的基础上，对数据库进行统一的管理和控制。用户使用的各种数据库命令以及应用程序的执行，都要通过数据库管理系统。数据库管理系统还承担着数据库的维护工作，按照 DBA 所规定的要求，保证数据库的安全性和完整性。应用系统是指利用数据库完成特定功能的用户程序。数据库管理员是指那些创建、使用和维护数据库的专门人员，又称 DBA（DataBase Administrator）。

数字有机体工作库: 它的全称是数字有机体工作库及大规模存储系统，简称为数字有机体工作库，有时也被称作数字有机体数据库系统。它是融合数据库技术、计算机网络技术和并行处理技术开发的数据库系统，简称为 DosSQL，是由分布在高速广域网内的大量高性能服务器构成的能够提供高度并行数据处理、大存储容量、高可靠性及可用性的数据库系统。它能满足当前高度复杂的数据处理需求和高可靠性要求，同时能够根据用户的需求变化进行动态配置，实现系统的动态伸缩和升级。数字有机体数据库系统不仅具有集中式数据库系统的所有功能，而且比集中式数据库系统具有更高的效率、可靠性及可用性，能满足多媒体等复杂数据的处理。该系统对外透明，即用户以使用单机数据库系统一样的方法使用这个系统，而不用了解这个系统的具体构成和规模。这样的系统具有极高的性价比，能够应用于大量复杂数据处理的环境，其高效性、高性价比和动态伸缩性已成为宽带网络运营商以及电子政务等应用的首选平台，能够为电子政务、视频点播等诸多应用提供数据管理，具有广阔的应用前景。

1.5 如何获得技术支持

在您遇到问题时，请首先联系您的产品提供商。大多数问题都可以在产品提供商的技术支持人员的帮助下得以解决。

您也可以通过产品提供商致电本公司的技术服务热线：028-83318559，获得电话技术支持。您还可以发送邮件，邮件地址是：tianxinyue@126.com。如果您确实需要本公司提供上门服务，本公司将竭诚为您服务。

2 简介

2.1 数字有机体系统简介

数字有机体系统（英文名称为 Digital Organism System，缩写为 DOS）是在刘心松教授带领下，由成都天心悦高科技发展有限公司的研发人员前后千余人次，经过三十多年的技术积累，研发成功的基础系统。

研发这种系统的原始宗旨是向生物特别是人类个体和群体的结构、机理和特性逼近，是一种人能化的新的系统模式。这种系统集操作系统、数据库系统、大规模存储、抗毁容灾、高伸缩、高智能、高灵活、自搜索、自传播、自复制、自修复、自重构、自适应、系统间的兼容性、群体间的协作性、对资源的动态管理调度合理配置、大小新旧机器混合使用等特性为一体，是一个整体解决方案，是面向所有应用的统一的（应用）系统平台。

数字有机体系统主要由数字有机体工作平台、数字有机体抗毁容灾系统、数字有机体工作库、数字有机体大规模存储与管理系统、数字有机体安全系统组成。这是从底层作起的一个一体化平台，可以在此平台上开发任何应用，形成任何应用系统。例如现在已有的应用系统就有数字有机体流媒体系统、数字有机体监控系统、数字有机体会议系统、数字有机体网关、数字有机体管理系统、数字有机体控申系统、数字有机体侦查指挥系统等。

本文有时将数字有机体工作平台及抗毁容灾系统，数字有机体工作库及大规模存储与管理系统和数字有机体安全系统统称为数字有机体系统。数字有机体工作平台及抗毁容灾系统含盖常规操作系统但远高于常规操作系统，是一个在 Linux 之上的、面向很多应用的、统一的、人能化的应用系统平台。数字有机体工作库及大规模存储与管理系统含盖常规数据库系统但远高于常规数据库系统，是一个在 Mysql 之上的、面向很多应用的、统一的、人能化的应用数据平台。

有时将数字有机体工作平台及抗毁容灾系统简称为数字有机体工作平台甚至工作平台。

有时将数字有机体工作库及大规模存储与管理系统简称为数字有机体工作库甚至工作库。

2.2 数字有机体工作库简介

数字有机体工作库由接口子系统、通信子系统、执行子系统和管理子系统等构成，各子系统相互协作实现相关功能。它可以作为一个单机数据库系统，也可以由大量服务器通过高速网络连接，构成一个超大型数字有机体工作库。由于系统具有高存储容量、高动态伸缩性、高可用性等特点，它可以为电子政务、工业、交通、教育、旅游、视频点播、校园网等对数据可靠性要求高、存储容量大的多种应用场合提供数据管理平台。它具有如下主要的功能和特性：

1) 海量存储

系统支持 P/EB 级数据存储。系统存储容量可根据用户数据的存储需求动态扩展，能够支持大量视频、音频等多媒体数据，同时能够存储多种文本数据以及二进制格式的软件，能够满足多种应用需求。

2) 分布性

系统的分布性主要体现在数据分布性、执行分布性以及管理分布性。数据分布性指系统的所有用户数据根据可靠性以及执行效率的要求分布在系统中的各个节点，它们之间没有主次之分，是完全对等的关系。执行分布性指用户的每一条数据库操作指令能够被系统自动解析，选择最佳的服务器节点完成，缩短响应时间，比单机数据库系统的操作效率有显著的提高，特别是复杂操作，效率可以成倍的提高。管理分布性指整个系统的维护和管理分布在所有的节点上，不存在核心节点或中心节点，也没有主次之分，这样既保证了高可靠性，又避免了系统瓶颈。

3) 并行性

系统的并行性主要体现在操作并行性和管理并行性。操作并行性指用户的不同操作在不同节点并行处理，同时根据需要用户的同一操作也可在不同的节点上并行处理，这样既提高了操作的效率，又加大了系统的吞吐量。管理并行性指对整个系统的维护在不同的节点上并行进行，而通过特定的协议协调和维护它们之间的一致性，这样提高了系统管理的效率，同时保证了高可靠性。这是集中式系统或单机系统以及 Cluster 系统无法做到的，是数字有机体工作库区别于其它数据库系统的显著特征。

4) 动态伸缩性

动态伸缩性指系统的规模、服务器能力能够根据需要进行动态配置，而不像单机系统服务能力从一开始就受限于服务器的性能指标。系统能够动态地增加/减少数据库服务器节点的数量，根据需要动态地改变服务能力，可以由大量服务器组成的系统提供服务，同时也可动态地更换或升级系统中的服务器，而不影响整个系统的正常使用。因此可以根据业务量的增长需求分期分批投资、建设。

5) 高可靠性及高可用性

系统根据用户对数据库的使用频率、可靠性要求以及配置情况，对数据库的分布以及冗余度动态决策，从而保证在任何时候用户数据的可用性。同时在一定环境下系统能够自动进行重构，这样在发生节点故障时，系统通过自动重构用户的数据分布，分离故障节点从而保证用户数据的高可靠性及可用性。

6) 高安全性

系统除具有一般数据库系统的安全特性外还提供远程数据同步功能，能够通过广域网络将系统中的数据从本地导出到远地存储，也可以方便地将数据从远地导入，因此即使发生灾难性事故也能保证数据的安全。数据通信支持 SSL 安全通信，并采用证书的机制防止主机的伪冒，具备主机识别的功能。

7) 使用方便性

系统提供多种接口，方便用户的二次开发。系统支持多种开发工具，如 ODBC、JDBC 等，支持 C/C＋＋、Perl、JAVA、PHP 等高级程序语言，完全兼容其它系统，能够开发动

态网页等各种数据库应用程序。

8) 硬件通用性

系统使用的各种硬件设备均为通用设备，构建系统方便、灵活，维护简单。

3 安装

在使用天心悦公司发布的数字有机体系统发行版时，DosSQL 将伴随系统自行安装。但是 DosSQL 系统也可以单独安装到其它的 Linux 发行版中。如果需要其它 Linux 发行版的数字有机体工作库安装包，请联系本公司。

如果使用的是天心悦公司发布的数字有机体系统发行版，DosSQL 已经安装，安装人员可以省去安装的步骤，直接对其进行配置。如果是对原有的 DosSQL 进行升级，应该先卸载原有的 DosSQL 文件，然后再进行重新安装。特别地，如果 DosSQL 的数据目录中有数据，卸载之前还需要进行数据备份，备份的方式请参阅第 13 章。

在 Linux 系统上查看已安装的数字有机体工作库的版本信息，请执行指令“rpm -q --info DosSQL”。

3.1 运行环境

数字有机体工作库可以运行在各种体系结构的服务器上，目前只发布 i386 版本和 x86_64 版本。其它体系结构的版本需要定制。

安装 DosSQL 需要的硬件最低配置如下：

CPU	Intel 至强 1.6G 4core 或以上处理器
内存	2G 或以上内存
硬盘	40G 或以上硬盘
网卡	100/1000M 网卡

结合当前的服务器性能水平，考虑业务运行的需求，作为工作库服务器，当前建议的配置如下：

整体	中高性能的服务器
CPU	Intel 1.6G 4 核或以上处理器
内存	128G 或以上的服务器专用内存
硬盘	4 块以上的 15000 转硬盘或者固态硬盘构建的 RAID5 系统，或者外接 RAID5 的磁盘阵列。
网络接口	千兆或者万兆网络接口

该系统推荐运行在数字有机体工作平台上，即数字有机体系统发行版。在必要时也可运行于普通 GUN Linux （内核版本 2.6 或以上）平台。

3.2 DosSQL 的部署模式

数字有机体工作库是由多台数据库服务器通过高速网络互联，协同工作而形成的。它的最小基本单元是数据库服务器，多台服务器组成一个有机站，多个有机站再组成数字有机体工作库系统。

数字有机体工作库依托广域网按照数字有机体站的结构进行部署。同一子网允许组建多个站，但是同一个站必须保证各节点处于同一子网内。系统的部署依托于系统安装时的配置文件“/etc/dossql.cnf”。

在系统配置参数（“/etc/my.cnf”和“/etc/dossql.cnf”）没有作变动的情况下，节点故障死亡后可以直接重启。如果有所修改，应该先修改故障死亡节点的参数使其和运行的节点参数一致，然后再重新启动，以避免出现异常。

在安装 DosSQL 之前，安装者首先需要了解待装计算机的数量、服务器的配置及每台服务器的 IP 地址。然后再由安装者进行规划，最后才开始安装。规划的内容是：

- 整个 DosSQL 系统的通信端口
- 将要部署的站数量
- 每个站的 ID 和站内通信端口
- 每个站的计算机数量及 IP 地址

在规划系统的部署方案是，需要充分考虑系统的应用需求和运行网络的情况。常见的部署方式有单站集中部署、多站集中部署和多站多地部署几种情况。下面分别予以说明。

3.2.1 单站集中部署

如果部署环境和应用需求是如下的，则建议单站集中部署：

1）要访问工作库的应用系统集中部署在一起，无多地部署需求；

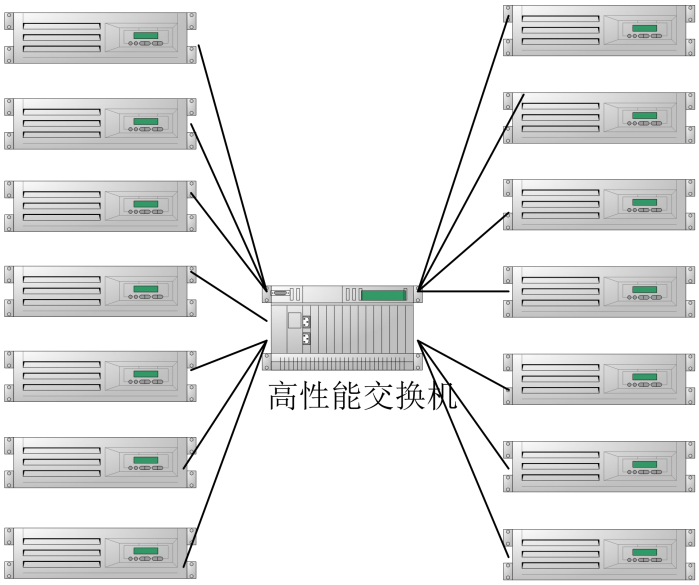
2）要部署的工作库服务器不超过 40 台；

3）要部署工作库的服务器处于同一个高速交换机下，或者在同一个高速交换网中，不跨 IP 子网部署；

4）要部署工作库的服务器处于同一个广播域中，即服务器间可以通过子网广播相互通信；

5）无异地抗毁容灾需求，因此无需异地部署。

满足上述条件时，系统建议采用单站集中部署方式，其部署结构如下图所示。通常，40 台以下的服务器可以用一台高性能交换机组网，使得任意两台服务器都可以线速通信。建议交换机端口的速度至少为千兆。采用万兆端口的交换机将提升系统的整体性能。交换机再连入用户网络。



所有的服务器在一台交换机上，或者由多台交换机堆叠而成的交换网中。这个交换网不应当配置有 VLAN 或者防火墙之类的，即必须确保任意两台服务器都可以相互通信，而且每台服务器的广播都可以到达所有的服务器。

在这种部署结构下，常见的应用模式有两种：第一种是大量应用数据库集中部署。通过将大量应用的数据库集中部署在这个系统中，可实现资源共享，提高应用数据库的可靠性、性能和安全性，同时也简化管理和维护。第二种是大数据分析。如果某个数据库的数据量非常大，而且要求的查询性能很高，则采用这种部署结构时，可将数据库均衡拆分到这些服务器上，通过数字有机体工作库的分布式并行查询功能，提升查询性能，满足大数据分析的需求。采用这种部署结构时，大多数单表查询和表连接的性能都可以近似线性的提升。原来在单机上需要 1 小时的查询，采用 40 台服务器后，可能 1 分多钟就可完成。

3.2.2 多站集中部署

如果工作库服务器的数量超过了 40 台，或者业务系统虽然是集中部署的，但是业务系统分别部署在不同的 IP 子网中，则建议采用多站集中部署方式。因此，多站集中部署的环境通常是：

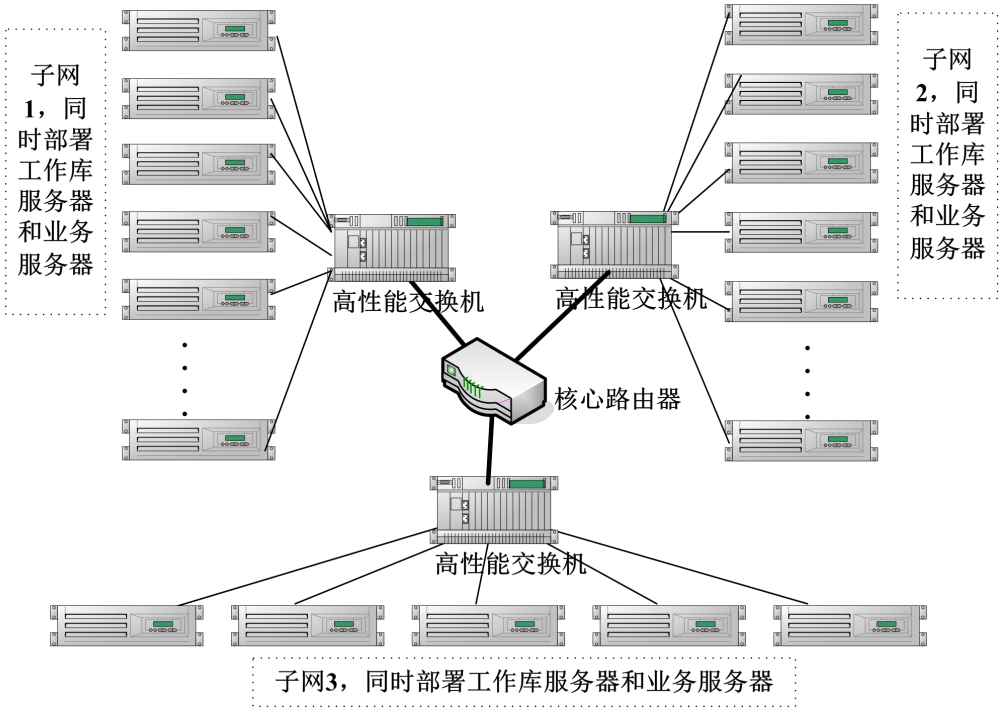
1）要访问工作库的应用系统集中部署在一起，无多地部署需求，但分别部署在不同的子网中；

2）或者要部署的工作库服务器超过 40 台；

3）要部署工作库的服务器可以处于几个 IP 子网中，以便应用系统就近访问；

4）无异地抗毁容灾需求，因此无需异地部署。

这时，我们将同一个 IP 子网的工作库服务器组织为一个数字有机体站，不同 IP 子网的服务器则在不同的站内。IP 子网自然需要路由器来互联，以便所有服务器仍然可以相互通信。但是一台服务器的广播则仅限于本子网中。因此，多站集中部署的结构如下图所示。



在单个子网中部署的工作库服务器最好不超过 40 台。在一个子网内建议同时部署工作库服务器和业务服务器。当然也可以将服务器同时作为工作库服务器和业务服务器。同一子网内的服务器配置为一个数字有机体站，即使用一个相同的组通信端口。所有服务器使用相同的管理通信端口，以便他们组织为一个系统。业务的数据库最好就近存储，当然也可以配置为允许存储在其他站中，但是因为是集中部署的，意义不大。不过，当某个站的资源不足，或者有服务器故障时，其他站的服务器可以作为在线的后备资源。系统自动使用它们，以便业务系统仍然可以访问数据库。

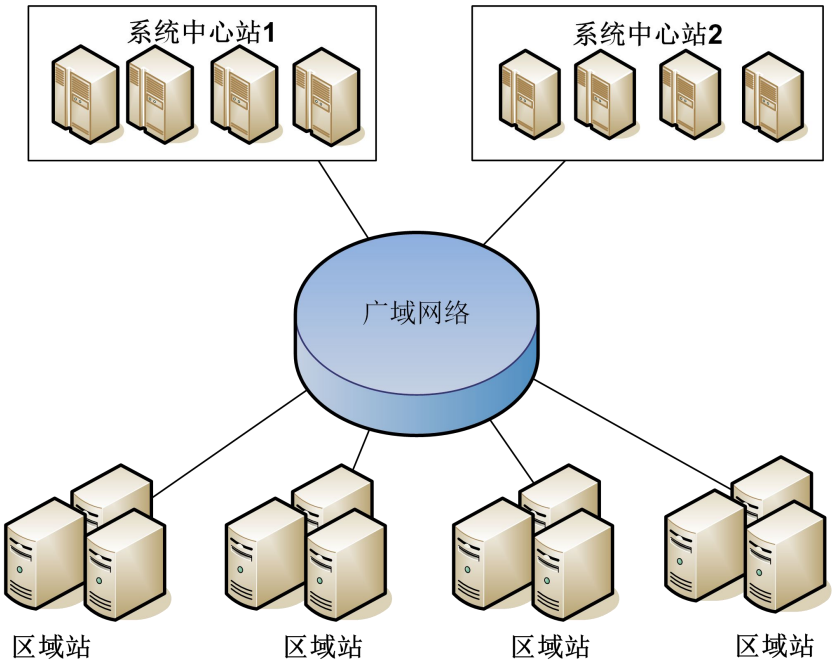
一个系统中站的数量可以很多，理论上是没有限制。

采用这种部署结构的主要目的有两个：一是解决大量服务器集中部署的需要，二是满足各个业务系统分子网部署的需求。前者主要用于有上百台服务器的大数据分析系统的部署。后者则主要满足大量业务系统分别独立部署的需求。

3.2.3 多站多地部署

如果系统存在异地容灾需求，则必须采用多站多地部署或者大规模部署模式。又或应用系统本身就分别部署在多个地点，则集中部署的数据库将无法满足应用系统的访问需求。多站多地部署的模式大致可以分为两类，一类是层级结构，另一类是对等结构。

层级结构的部署模式往往应用于需要数据汇聚的场景，例如银行系统的分层数据汇集。它的部署示意图如下。



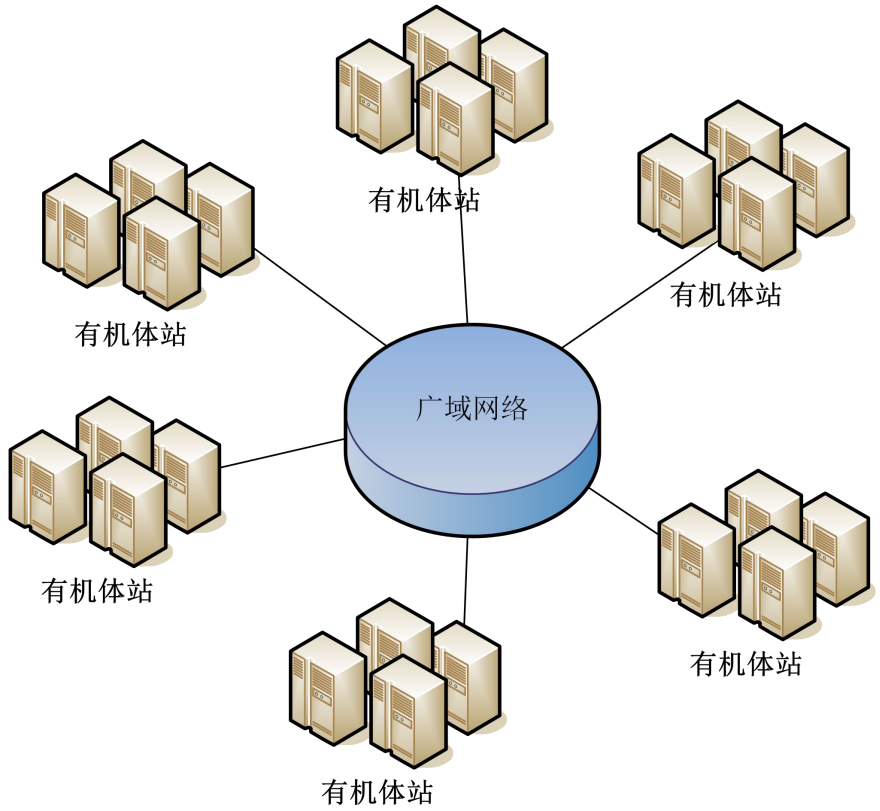
以银行系统为例，系统中心站可能是部署在省级或者总行的数字有机体站，区域站则是部署在各个地区或者分支结构的数字有机体站。数据库的部署方式有两种。一种是每个区域有自己独立的数据库，各个区域的数据库异地备份到中心站，从而使得数据自动汇聚到中心站。

数据库副本间同步的方式可以是实时同步，也可以是批量同步（即异步日志同步）。实时同步适用于区域站到中心站带宽充足，延迟不大的环境。批量同步方式则对带宽的要求不那么严格，而且容忍网络的短暂中断。

数据库的第二种部署方式是表分片。对大数据表按照区域进行分割，每个区域存储属于本区域的数据，然后在中心站建立表分片存储库的副本。副本间的同步仍然可以采用实时同步或者批量同步。这种方式的优点是对应于来说，仍然以单一库表的方式访问系统，而无需考虑对多个独立数据库的集成访问，简化了应用编程。

各个区域站的数据都已经异地复制到中心站，因此具备了异地备份的功能。而单一的中心站无法实现异地复制，因此在需要抗毁容灾时，应建立一个相当规模的异地的中心站，以便实现中心站的抗毁容灾。

如果系统部署并没有层级关系，在应用逻辑上也不存在层级关系，则系统可以采用对等部署模式。其部署结构如下图所示。



在对等部署模式下，所有的数字有机体站都是完全平等的。数据库可以在任意站间复制，从而实现异地的备份。当然，管理员也可以手动的控制数据库副本的位置，以满足某些特定需求。如果对大数据表进行分片，则分片存储库可以分布在任意站上，从而充分利用各地服务器的能力。这种部署方式的优点是各个站的资源可以相互共享，从而更好的利用系统资源。这种部署方式常应用于大型内容提供商的多数据中心。

3.3 DosSQL 安装与配置

DosSQL 的安装文件为 rpm 安装包。可使用 RPM 指令进行安装，下面是安装的具体步骤：

1) 获取安装包 DosSQL-3.3.0-1.x86_64.rpm 到/home/目录下；

```
# /home/DosSQL-3.3.0-1.x86_64.rpm
```

2) 进入数据库安装包目录，并执行安装指令；

```
#cd /home/
#rpm    -ivh --nodeps    DosSQL-3.3.0-1.x86_64.rpm
```

3) 安装完成后，首先杀死数据库进程，修改配置文件/etc/my.cnf 和/etc/dossql.cnf。

```
#service dossql stop
```

DosSQL 有三个配置文件，分别是“/etc/my.cnf”、“/etc/loginiips.log”和“/etc/dossql.cnf”。其中“/etc/my.cnf”配置文件是 DosSQL 单机的系统变量，DosSQL 是基于 MySQL 开发而来，因此详情请参考 MySQL 的用户手册；“/etc/loginips.log”是一个系统自动维护的 DosSQL

服务器清单，用于启动时检测其它服务器是否已经存在；“/etc/dossql.cnf”是 DosSQL 分布式服务器管理的系统参数配置文件，通过配置它可以构建大规模的数字有机体工作库服务器群。manager_port 不配置时的缺省参数是 18188；station_id 不配置时的缺省参数是 1；station_cast_port 不配置时的缺省参数是 28192。“/etc/dossql.cnf”的参数配置详细说明如下：

```
[DosSQL]
# This is an important tcp port, all nodes of system communicat with it.
# The ports: manager_port, manager_port + 1, manager_port + 2
# and manager_port + 3 are in use.
manager_port=8888（系统内的通信端口，同一系统内的节点配置须一致）

# The host ip that localhost startup from it. It of other station.
neighbor_ip=（邻居节点，配置为其它站的 IP，一般配置为系统内最大的节点 IP，最大 IP 节点则不配置本参数）

# The station identifier, a station contain max 254 servers, all nodes
# of station should to configure as the same.
station_id=3（站 ID，有机体数据库系统可以由多个站组成，同一个站的 IP 须在同一网段）

# This udp port is used for communicating in station, all nodes of station
# should to configure as the same.
station_cast_port=8777（站内的通信端口，同一站的节点须配置一致，不同站须配置不同的端口）

# The dosdb engine use system user@password, it will not be used if you set '0'
dosdb_use_system_user=1（dosdb 是否使用系统用户，1 表示使用，0 表示不使用）

# The follow is SSL related arguments, default is not use ssl.
use_ssl=0（是否使用主机间 SSL 安全通信，使用 SSL 通信将同时具备主机识别功能）
ssl_ca=/etc/do/cacert.pem（SSL 的根证书）
ssl_cert=/etc/do/server-cert.pem（SSL 的公钥证书）
ssl_key=/etc/do/server-key.pem（SSL 的私钥证书）
ssl_password=123456（SSL 证书密码）

# The tcp ports that it maybe will used, default is 7000-7500
min_tcp_port=7000（系统使用的 TCP 通信端口的最小值）
max_tcp_port=7500（系统使用的 TCP 通信端口的最大值）

# The follow is used to set the default number of database replica.
```

```
min_dup_num=3（数据库副本的默认最小数量）
max_dup_num=5（数据库副本的默认最大数量）

# The local_ip if it is not configured, used the 'eth0', else use the
# configured.
local_ip=（本机主机地址，不配置时自动采用 eth0 网口地址，多网口并需要采用其它端口时可以配置具体的想要的 IP 地址）

# Wait time, when connection coming too many and fast, will sleep, unit is 'ms'.
schedule_delay_time=400（调度延迟时间，多个连接同时到达时，前后两个连接之间的时间间隔，本参数值越大，调度的速度就越慢，但是负载就越均衡；本参数值越小，调度的速度就越快，但是负载就越不均衡）

# The follow is arguments related SQL extcute, advise not to modify them.
wait_time_of_proxythd=60（分布式事务执行等待时间）
timeout_for_create_exe_proxy=10（新建分布式事务执行单元超时时间）

# The bandwidth defined band width, unit is 'MB'.
bandwidth=10（主机的带宽）

# The min_disk_capacity defined min reserve disk capacity, unit is 'MB'.
min_disk_capacity=50（最小磁盘报警容量）
```

DosSQL 工作库系统的服务监听端口缺省值是 3306，站内服务器和系统内服务器的通信端口是工具用户的需求自定义的。

想要 DosSQL 系统运行性能较好，需要配置“/etc/my.cnf”。配置之前，请先查看内存总容量和操作系统是否 32 位机，参数“innodb_buffer_pool_size”一般配置为内存总容量的 50%~80%（单位为 M），如果操作系统是 32 位机，参数“innodb_buffer_pool_size”配置应该不超过 3GB。更多的参数配置可参照《MySQL 5.7 参考手册》。

“/etc/my.cnf”中的二进制日志选项“lower-case-table-names”缺省关闭，值为 0，表示数据库和表名字母大小写敏感；当需要表示数据库和表名字母大小写不敏感时，请将选项“lower-case-table-names”设置为 1；有 InnoDB 时不能设置为 2，值 2 和 0 均表示数据库和表名字母大小写敏感。

如果使用 SSL 安全连接，配置文件“/etc/my.cnf”中的“[dossqld]”下的 SSL 参数需要配置，否则程序无法正确加载 SSL。配置示例如下：

```
[dossqld]
ssl-ca=/etc/my/cacert.pem
ssl-cert=/etc/my/server-cert.pem
```

```
ssl-key=/etc/my/server-key.pem
```

注意，如果 DosSQL 服务器成功运行，它将记录一些关键的主机地址到配置文件“/etc/loginips.log”，用于下次启动时使用。如果是重新安装，或者是重新配置现有的服务器时，请务必把配置文件“/etc/loginips.log”的内容清空。

3.4 DosSQL 启动与停止

1) 授权使用

未经授权使用的主机，即使正确安装了 DosSQL 也无法启动。启动前，需要获取由成都天心悦高科技发展公司颁发的授权证书。授权方式请参阅《数字有机体系统(DOS-4.0)安装指南》。

2) 启动和停止

首次启动 DosSQL 之前，请务必初始化数据库的数据目录，初始操作如下所示：

```
#cd /usr/local/dossql/
# ./scripts/dossql_install_db
```

用以下方法启动数字有机体工作库：

```
#/etc/init.d/dossql start
或者：
#service dossql start
```

启动成功的标志如下：

```
Starting DosSQL
[ ok .....
```

初次启动，成功后的默认管理员用户名是“root”，密码为空。DosSQL 提供字符界面的监视器 DosSQL 工具程序，应该在 Linux 系统的超级终端使用 DosSQL 登陆，方法如下，-u 后面给出登陆的用户名，-p 后面给出登陆用的密码：

```
root@server54:/home/dos# DosSQL   –uroot   -p""
Enter password:
Welcome to the DosSQL monitor.   Commands end with ; or \g.
Copyright (c) 2000, 2018, TianXinYue.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
DosSQL[(none)]$
```

用以下方法停止数字有机体工作库：

```
#/etc/init.d/dossql stop
或者：
#service dossql stop
```

停止 DosSQL 服务后显示如下：

```
Shutting down DosSQL
[ ok ......
```

您可以使用“service dossql status”指令查看 DosSQL 服务进程，查看进程显示如下：

```
[ ok ] DosSQL running (12554).
```

您还可以使用“service DosSQL restart”指令重新启动 DosSQL 服务，重启显示如下：

```
Shutting down DosSQL
[ ok ......
Starting DosSQL
[ ok.
```

在 4.2 版以前的 DOS 发行版中，DosSQL 工作库系统默认是自动启动的。自动启动是由自启动脚本“/usr/sbin/dpserver_start”来实现的，在该脚本中的“service dossql start”启动命令完成 DosSQL 的。如果，您不需要 DosSQL 开机自动启动，您可以在“/usr/sbin/dpserver_start”文件中注释掉该启动命令。

在 4.2 版的 DOS 发行版中，DosSQL 工作库系统虽然也是自动启动的，但是是作为系统服务独立启动的。可以使用系统命令“update-rc.d dossql disable”来取消工作库系统的自动启动，也可以使用系统命令“update-rc.d dossql enable”来恢复工作库系统的自动启动。

3.5 卸载 DosSQL

如果数据库系统已经启动，那么先关闭该系统，然后执行卸载指令（例如 x86 机型的卸载命令）：

```
#rpm    -e --nodeps    DosSQL-3.3.0-1.x86_64
```

4 权限管理

在数字有机体工作库中，系统的管理权限主要分为两类：第一类是不针对具体数据库的授权，属于系统管理员权限。第二类是针对具体数据库的授权，属于数据库用户权限。数字有机体工作库采用统一权限管理机制，每个数据库节点上的权限都保持一致。

对系统管理员权限的解释：该类用户主要用 CREATE USER、DROP USER、RENAME USER、GRANT 和 REVOKE 来授权。不支持 SET PASSWORD 语句和直接对数据库权限表的修改。特别标志是并非针对具体某个数据库授权（*.*）。该类用户具有创建数据库并对本站创建的数据库进行授权的权限（并不具有对其它站创建的数据库的管理权限），具备设置数据库属性及查看数据库属性的权限。

对数据库用户权限的解释：该类用户是由系统管理员授权的用户。该类用户不具备创建数据库的权力，只具备访问操作具体数据库的能力。该类用户也不可以设置或者查询数据库属性。

在数字有机体工作库中，系统管理员权限会存储在数字有机体站内所有节点上，而数据库用户权限会存储在所对应的数据库的所有副本机器上。

在对数据库进行访问时，应该使用专门的数据库用户权限（用户类型）进行数据库连接，而不应该使用系统管理员权限登录。

这里不支持 SET PASSWORD 语句，如果确实需要修改密码，可以使用 GRANT 语句，如果还是不行，可以先删除用户，然后重新授权。

4.1 CREATE USER 语法

CREATE USER user [IDENTIFIED BY [PASSWORD] 'password']

[, user [IDENTIFIED BY [PASSWORD] 'password']] ...

CREATE USER 用于创建新的 DosSQL 账户。要使用 CREATE USER，您必须拥有 DosSQL 数据库的全局 CREATE USER 权限，或拥有 INSERT 权限。对于每个账户，CREATE USER 会在没有权限的 mysql.user 表中创建一个新记录。如果 账户已经存在，则出现错误。

使用自选的 IDENTIFIED BY 子句，可以为账户给定一个密码。user 值和 密码的给定方法和 GRANT 语句一样。特别是，要在纯文本中指定密码，需忽略 PASSWORD 关键词。要把密码指定为由 PASSWORD()函数返回的混编值，需包含关键字 PASSWORD。

4.2 DROP USER 语法

DROP USER user [, user] ...

DROP USER 语句用于删除一个或多个 DosSQL 账户。要使用 DROP USER，您必须拥有 mysql 数据库的全局 CREATE USER 权限或 DELETE 权限。使用与 GRANT 或 REVOKE 相同的格式为每个 账户命名；例如，'jeffrey'@'localhost'。 账户名称的用户和主机部分与

用户表记录的 User 和 Host 列值相对应。

使用 DROP USER，您可以取消一个账户和其权限，操作如下：

DROP USER user;

该语句可以删除来自所有授权表的帐户权限记录。

要点：DROP USER 不能自动关闭任何打开的用户对话。而且，如果用户有打开的对话，此时取消用户，则命令不会生效，直到用户对话被关闭后才生效。一旦对话被关闭，用户也被取消，此用户再次试图登录时将会失败。这是有意设计的。

4.3 GRANT 和 REVOKE 语法

```
GRANT priv_type [(column_list)] [, priv_type [(column_list)]] ...
    ON [object_type] {tbl_name | * | *.* | db_name.*}
    TO user [IDENTIFIED BY [PASSWORD] 'password']
        [, user [IDENTIFIED BY [PASSWORD] 'password']] ...
    [REQUIRE
        NONE |
        [{SSL| X509}]
        [CIPHER 'cipher' [AND]]
        [ISSUER 'issuer' [AND]]
        [SUBJECT 'subject']]
    [WITH with_option [with_option] ...]

object_type =
        TABLE
    | FUNCTION
    | PROCEDURE

with_option =
        GRANT OPTION
    | MAX_QUERIES_PER_HOUR count
    | MAX_UPDATES_PER_HOUR count
    | MAX_CONNECTIONS_PER_HOUR count
    | MAX_USER_CONNECTIONS count
REVOKE priv_type [(column_list)] [, priv_type [(column_list)]] ...
    ON [object_type] {tbl_name | * | *.* | db_name.*}
    FROM user [, user] ...

REVOKE ALL PRIVILEGES, GRANT OPTION FROM user [, user] ...
```

GRANT 和 REVOKE 语句允许系统管理员创建 DosSQL 用户账户，授予权限和撤销权限。

DosSQL 账户信息存储在 mysql 数据库的表中。

如果授权表拥有含有 mixed-case 数据库或表名称的权限记录，并且 lower_case_table_names 系统变量已设置，则不能使用 REVOKE 撤销权限，必须直接操纵授权表。（当 lower_case_table_names 已设置时，GRANT 将不会创建此类记录，但是此类记录可能已经在设置变量之前被创建了。）

授予的权限可以分为多个层级：

全局层级

全局权限适用于一个给定服务器中的所有数据库。这些权限存储在 mysql.user 表中。GRANT ALL ON *.*和 REVOKE ALL ON *.*只授予和撤销全局权限。

数据库层级

数据库权限适用于一个给定数据库中的所有目标。这些权限存储在 mysql.db 和 mysql.host 表中。GRANT ALL ON db_name.*和 REVOKE ALL ON db_name.*只授予和撤销数据库权限。

表层级

表权限适用于一个给定表中的所有列。这些权限存储在 mysql.talbes_priv 表中。GRANT ALL ON db_name.tbl_name 和 REVOKE ALL ON db_name.tbl_name 只授予和撤销表权限。

列层级

列权限适用于一个给定表中的单一列。这些权限存储在 mysql.columns_priv 表中。当使用 REVOKE 时，您必须指定与被授权列相同的列。

子程序层级

CREATE ROUTINE, ALTER ROUTINE, EXECUTE 和 GRANT 权限适用于已存储的子程序。这些权限可以被授予为全局层级和数据库层级。而且，除了 CREATE ROUTINE 外，这些权限可以被授予为子程序层级，并存储在 mysql.procs_priv 表中。

当后续目标是一个表、一个已存储的函数或一个已存储的过程时，object_type 子句应被指定为 TABLE、FUNCTION 或 PROCEDURE。当从旧版本的 DosSQL 升级时，要使用本子句，您必须升级您的授权表。

要使用 GRANT 或 REVOKE，您必须拥有 GRANT OPTION 权限，并且您必须用于您正在授予或撤销的权限。

要撤销所有权限，需使用以下语法。此语法用于取消对于已命名的用户的所有全局层级、数据库层级、表层级和列层级的权限。

REVOKE ALL PRIVILEGES, GRANT OPTION FROM user [, user] ...

要使用本 REVOKE 语法，您必须拥有 DosSQL 数据库的全局 CREATE USER 权限或 UPDATE 权限。

对于 GRANT 和 REVOKE 语句，priv_type 可以被指定为以下任何一种：

权限	意义
ALL [PRIVILEGES]	设置除 GRANT OPTION 之外的所有简单权限
ALTER	允许使用 ALTER TABLE
ALTER ROUTINE	更改或取消已存储的子程序
CREATE	允许使用 CREATE TABLE
CREATE ROUTINE	创建已存储的子程序
CREATE TEMPORARY TABLES	允许使用 CREATE TEMPORARY TABLE
CREATE USER	允许使用 CREATE USER, DROP USER, RENAME USER 和 REVOKE ALL PRIVILEGES。
CREATE VIEW	允许使用 CREATE VIEW
DELETE	允许使用 DELETE
DROP	允许使用 DROP TABLE
EXECUTE	允许用户运行已存储的子程序
FILE	允许使用 SELECT...INTO OUTFILE 和 LOAD DATA INFILE
INDEX	允许使用 CREATE INDEX 和 DROP INDEX
INSERT	允许使用 INSERT
LOCK TABLES	允许对您拥有 SELECT 权限的表使用 LOCK TABLES
PROCESS	允许使用 SHOW FULL PROCESSLIST
REFERENCES	未被实施
RELOAD	允许使用 FLUSH
REPLICATION CLIENT	允许用户询问从属服务器或主服务器的地址
REPLICATION SLAVE	用于复制型从属服务器（从主服务器中读取二进制日志事件）
SELECT	允许使用 SELECT
SHOW DATABASES	SHOW DATABASES 显示所有数据库
SHOW VIEW	允许使用 SHOW CREATE VIEW
SHUTDOWN	允许使用 DosSQLadmin shutdown
SUPER	允许使用 CHANGE MASTER, KILL, PURGE MASTER LOGS 和 SET GLOBAL 语句，DosSQLadmin debug 命令；允许您连接（一次），即使已达到 max_connections。
UPDATE	允许使用 UPDATE
USAGE	“无权限”的同义词
GRANT OPTION	允许授予权限

当从旧版本的 DosSQL 升级时，要使用 EXECUTE, CREATE VIEW, SHOW VIEW, CREATE USER, CREATE ROUTINE 和 ALTER ROUTINE 权限，您必须首先升级您的授权表。

REFERENCES 权限目前未被使用。

当您想要创建一个没有权限的用户时，可以指定 USAGE。

使用 SHOW GRANTS 来确定帐户拥有什么权限。

您可以通过使用 ON *.*语法赋予全局权限，或通过使用 ON db_name.*语法赋予数据库层级权限。如果您指定了 ON *并且您已经选择了一个默认数据库，则权限被赋予到这个数据库中。(警告：如果您指定了 ON *同时您没有选择一个默认数据库，则权限是全局的。)

FILE, PROCESS, RELOAD, REPLICATION CLIENT, REPLICATION SLAVE, SHOW DATABASES, SHUTDOWN 和 SUPER 权限是管理性权限，只能进行全局授权(使用 ON *.*语法)。

其它权限可以被全局授权，或被赋予为其它层级。

对于一个表，您可以指定的 priv_type 值只能是 SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, GRANT OPTION, INDEX 和 ALTER。

对于一个列（也就是，当您使用一个 column_list 子句时），您可以指定的 priv_type 值只能是 SELECT, INSERT 和 UPDATE。

在子程序层级，您可以指定的 priv_type 值只能是 ALTER ROUTINE, EXECUTE 和 GRANT OPTION。CREATE ROUTINE 不是一个子程序层级的权限，因为您必须拥有此权限，才能创建一个子程序。

对于全局、数据库、表和子程序层级，GRANT ALL 只能赋予在您正在授权的层级中存在的权限。例如，如果您使用 GRANT ALL ON db_name.*，这是一个数据库层级语句，因此不会授予全局权限，如 FILE 等。

DosSQL 允许您对不存在的数据库目标授予权限。在此情况下，将被授予的权限必须包括 CREATE 权限。这个性质是有意设计的，目的是允许数据库管理员为将在此后被创建的数据库目标预备用户 账户和权限。

要点：当您取消一个表或数据库时，DosSQL 不会自动撤销任何权限。但是，如果您取消一个子程序，则被赋予该子程序的所有子程序层级的权限都被撤销。

注意：GRANT 语句用于在全局层级或数据库层级赋予权限。当在 GRANT 语句中指定数据库名称时，允许使用‘_’和‘%’通配符。这意味着，如果您想要使用‘_’字符作为一个数据库名称的一部分，您应该在 GRANT 语句中指定它为‘_’，以防止用户可以访问其它符合此通配符格式的数据库；例如，GRANT ... ON `foo\_bar`.* TO ...。

为了接纳对来自任意主机的用户授权的权利，DosSQL 支持以 user_name@host_name 的形式指定 user 值。如果一个 user_name 或 host_name 与一个不加引号的标识符一样是合法的，那么您不需要对它加引号。不过，要指定一个包含特殊字符（如‘-’）的 user_name 字符串，或一个包含特殊字符或通配字符（如‘%’），则引号是必要的；例如，'test-user'@'test-hostname'。分别对 username 和 hostname 加引号。

您可以在 hostname 中指定通配符。例如 user_name@'%.loc.gov'适用于在 loc.gov 域中的任何主机的 user_name。同时 user_name@'144.155.166.%'适用于 144.155.166 C 级子网中的任何主机的 user_name。

简单形式 user_name 是 user_name@'%'的同义词。

DosSQL 不支持 usernames 中的通配符。通过把带有 User=''的登录项插入到 mysql.user

表中，或通过使用 GRANT 语句创建一个带有空名称的用户，可以定义匿名用户：

DosSQL> GRANT ALL ON test.* TO "@'localhost' ...

当把带引号的值是，需使用反勾号(‘’)为数据库、表、列和子程序名称加引号。使用单引号(‘’)为 hostnames、usernames 和 密码加引号。

警告：如果您允许匿名用户连接到 DosSQL 服务器，则您应该同时向所有本地用户授予 user_name@localhost 权限。否则，当有名称的用户试图从本地机器登录 DosSQL 服务器时，mysql.user 表中的用于 localhost 的匿名用户帐户会被使用。

您可以通过执行以下查询来确定这是否适合于您。以下查询列举了所有匿名用户：

DosSQL$ SELECT Host, User FROM mysql.user WHERE User='';

如果您想要删除本地匿名用户账户，以避免出现刚才谈到的问题，则需使用以下语句：

DosSQL$ DELETE FROM mysql.user WHERE Host='localhost' AND User='';

DosSQL$ FLUSH PRIVILEGES;

GRANT 支持最长为 60 个字符的 hostnames。数据库、表、列和子程序名称最长可为 64 个字符。Usernames 最长可为 16 个字符。 注释：不能通过更改 mysql.user 表来改变 usernames 的允许长度。如果试图这么做，会导致出现不可预见的问题，可能会造成用户无法登录 DosSQL 服务器。

对于表或列的权限是作为各个权限层级的逻辑 OR 权限被附加形成的。例如，如果 mysql.user 表指定一个用户拥有全局 SELECT 权限，则该权限不能被数据库、表或列层级的登录项定义。

可以按下列方法计算列权限：

global privileges

OR (database privileges AND host privileges)

OR table privileges

OR column privileges

在多数情况下，您只在一个权限层级下向用户授予权利，所以寿命通常不是那么复杂。

如果您对一个在 mysql.user 表中不存在的 username/hostname 组合授予权限，则增加一个登录项并保持在此处，直到使用 DELETE 语句删除为止。换句话说，GRANT 可以创建用户表登录项，但是 REVOKE 不会取消它们；您必须使用 DROP USER 或 DELETE 明确地操作。

如果创建了一个新的用户，或者如果您拥有全局授权权限，则用户密码被设置为由 IDENTIFIED BY 子句指定的密码（如果给定了一个）。如果用户已拥有了一个密码，则此密码被新密码替代。

警告：如果您创建了一个新用户，但是不指定 IDENTIFIED BY 子句，则用户没有 密码。这是很不安全的。不过，您可以启用 NO_AUTO_CREATE_USER SQL 模式，来防止 GRANT 创建一个新用户（否则 GRANT 会这么做），除非给定了 IDENTIFIED BY 来为新用户提供一个密码。

在 IDENTIFIED BY 子句中，密码应被作为文字密码被给定。没有必要使用 PASSWORD()函数，因为该函数用于 SET PASSWORD 语句。例如：

GRANT ... IDENTIFIED BY 'mypass';

如果您不想以明白的文字发送密码，并且您知道 PASSWORD()返回给密码的混编值，则您可以指定混编值，前面加入关键词 PASSWORD：

GRANT ...

IDENTIFIED BY PASSWORD '*6C8989366EAF75BB670AD8EA7A7FC1176A95CEF4';

在一个 C 程序中，您可以通过使用 make_scrambled_password() C API 函数得到混编值。

如果您为一个数据库授予权限，则在 mysql.db 表中，会根据需要创建登录项。如果使用 REVOKE 删除了所有的数据库权限，则本登录项被删除。

如果一个用户不拥有表权限，则当用户申请表清单时（例如，使用 SHOW TABLES 语句），表名称不显示。

SHOW DATABASES 权限允许账户通过发布 SHOW DATABASE 语句来观看数据库名称。不拥有此权限的账户只能看到他们拥有部分权限的数据库，并且如果使用 --skip-show-database 选项启动服务器，则根本不能使用本语句。

WITH GRANT OPTION 子句给予用户能力，可以在指定的权限层级，向其它用户给定其拥有的任何权限。您应该留心您给予了谁 GRANT OPTION 权限，因为拥有不同权限的两个用户可以联合使用权限！

您不能向其它用户授予您自己没有的权限；GRANT OPTION 权限只允许您赋予您自己拥有的权限。

要注意，当您在某个特定权限层级向一个用户授予 GRANT OPTION 权限时，用户拥有的该层级的任何权限（或未来将被给定的权限）也可以由该用户授予。假设您向一个用户赋予了数据库 INSERT 权限。如果您然后赋予数据库 SELECT 权限，并指定了 WITH GRANT OPTION，则该用户不仅可以向其它用户给予 SELECT 权限，还可以给予 INSERT。如果您然后向用户授予数据库 UPDATE 权限，则用户可以授予 INSERT, SELECT 和 UPDATE。

您不应该向一个常规用户授予 ALTER 权限。如果您这么做，则该用户可以尝试通过对表重新命名来破坏授权系统！

MAX_UPDATES_PER_HOUR count 和 MAX_CONNECTIONS_PER_HOUR count 选项限制了在任何给定的一小时期间，用户可以执行的查询、更新和登录的数目。如果 count 是 0（默认值），这意味着，对该用户没有限制。

MAX_USER_CONNECTIONS count 选项限制了账户可以同时进行的连接的最大数目。如果 count 是 0（默认值），则 max_user_connections 系统可以决定该 账户同时连接的数目。

注释：要对一个原有的用户指定任何这类资源限制型选项，同时又不影响原有的权限，需使用 GRANT USAGE ON *.* ... WITH MAX_...。

除了根据 username 和密码进行常规鉴定外，DosSQL 还可以检查 X509 证明属性。要为 DosSQL 账户指定与 SSL 有关的选项，需使用 GRANT 语句的 REQUIRE 子句。

对于一个给定的账户，有多种可能性可以限制连接类型：

如果账户没有 SSL 或 X509 要求，并且如果 username 和 密码是有效的，则允许不加

密连接。但是，如果客户端有正确的证明和关键文件，则根据客户端的选择，也可以使用加密连接。

REQUIRE SSL 选项用于告知服务器，对于该账户只允许 SSL 加密连接。注意，如果有允许任何非 SSL 连接的访问控制记录，则本选项可以被忽略。

DosSQL$ GRANT ALL PRIVILEGES ON test.* TO 'root'@'localhost'

-> IDENTIFIED BY 'goodsecret' REQUIRE SSL;

REQUIRE X509 意味着客户端必须拥有一个有效证明，除非不需要确切的证明、发布者和主题。唯一的要求是，应可以使用 CA 证明其中之一来验证签名。

DosSQL$ GRANT ALL PRIVILEGES ON test.* TO 'root'@'localhost'

-> IDENTIFIED BY 'goodsecret' REQUIRE X509;

REQUIRE ISSUER 'issuer'用于对连接尝试进行限定，客户端必须出示一个由 CA’issuer’发布的有效的 X509 证明。如果客户端出示的证明是有效的，但是有一个不同的发布者，则服务器会拒绝连接。使用 X509 证明就意味着要加密，所以在这种情况下，SSL 选项是不必要的。

DosSQL$ GRANT ALL PRIVILEGES ON test.* TO 'root'@'localhost'

-> IDENTIFIED BY 'goodsecret'

-> REQUIRE ISSUER '/C=FI/ST=Some-State/L=Helsinki/

O=DosSQL Finland AB/CN=Tonu Samuel/Email=tonu@example.com';

注意，ISSUER 值应被作为一个单一字符串输入。

REQUIRE SUBJECT 'subject'用于对连接尝试进行限定，客户端必须出示一个包含主题 subject 的有效的 X509 证明。如果客户端出示的证明是有效的，但是有一个不同的主题，则服务器会拒绝连接。

DosSQL$ GRANT ALL PRIVILEGES ON test.* TO 'root'@'localhost'

-> IDENTIFIED BY 'goodsecret'

-> REQUIRE SUBJECT '/C=EE/ST=Some-State/L=Tallinn/

O=DosSQL demo client certificate/

CN=Tonu Samuel/Email=tonu@example.com';

注意，SUBJECT 值应被作为一个单一字符串输入。

需要 REQUIRE CIPHER 'cipher'来确认使用了密码和足够长度的关键字。如果使用了采用短型加密关键字的旧算法，SSL 本身会比较脆弱。使用本选项，您可以要求使用特定的密码方法来许可一个连接。

DosSQL$ GRANT ALL PRIVILEGES ON test.* TO 'root'@'localhost'

-> IDENTIFIED BY 'goodsecret'

-> REQUIRE CIPHER 'EDH-RSA-DES-CBC3-SHA';

SUBJECT, ISSUER 和 CIPHER 选项可以在 REQUIRE 子句中结合，例如：

DosSQL$ GRANT ALL PRIVILEGES ON test.* TO 'root'@'localhost'

-> IDENTIFIED BY 'goodsecret'

-> REQUIRE SUBJECT '/C=EE/ST=Some-State/L=Tallinn/

O=DosSQL demo client certificate/

CN=Tonu Samuel/Email=tonu@example.com'

-> AND ISSUER '/C=FI/ST=Some-State/L=Helsinki/

O=DosSQL Finland AB/CN=Tonu Samuel/Email=tonu@example.com'

-> AND CIPHER 'EDH-RSA-DES-CBC3-SHA';

注意，SUBJECT 和 ISSUER 值各自应被作为一个单一字符串输入。

在 REQUIRE 各选项之间，AND 关键词是自选的。

选项的顺序无所谓，但是选项不能被指定两次。

当 dossqld 启动后，所有的权限被读入存储器中。

注意，如果您正在使用表权限或列权限，即使只对一个用户使用，服务器也会对所有用户检查表权限和列权限，这会略微降低 DosSQL 的速度。与此类似，如果您对某些用户限制查询、更新或连接的数目，则服务器必须监测这些值。

标准 SQL 版本和 DosSQL 版本的 GRANT 之间的最大区别是：

在 DosSQL 中，权限与 hostname 和 username 的组合有关，与单一的 username 无关。

标准 SQL 不拥有全局层级或数据库层级权限，也不支持 DosSQL 支持的所有权限类型。

DosSQL 不支持标准 SQL TRIGGER 或 UNDER 权限。

标准 SQL 权限以一种分等级的方式进行组织。如果您取消一个用户，则用户被授予的所有权限都被撤销。在 DosSQL 中，如果您使用 DROP USER，也会如此。

在标准 SQL 中，当您取消一个表时，对一个表的所有权限会被撤销。在标准 SQL 中，当您撤销一个权限时，根据该权限被授予的所有权限也会被撤销。在 DosSQL 中，只有使用明确的 REVOKE 语句，或通过操作存储在 DosSQL 授权表中的值，才能取消权限。

在 DosSQL 中，可以只对一个表中的部分列拥有 INSERT 权限。在此情况下，如果您忽略您不拥有 INSERT 权限的那些列，您仍然可以对表执行 INSERT 语句。如果没有启用严格的 SQL 模式，则被忽略的列被设置为各自隐含的默认值。在严格模式下，如果某个被忽略的列没有默认值，则该语句被拒绝。

您不拥有 INSERT 权限的列被设置为各自的默认值。标准 SQL 要求您拥有所有列的 INSERT 权限。

在 DosSQL 中，如果您只拥有一个表中的部分列的 INSERT 权限，同时如果您从 INSERT 语句中忽略您不拥有权限的列，则您仍然可以对表执行 INSERT 语句；那些列将被设置为各自的默认值。在严格模式下，即当 sql_mode='traditional'时，如果某些被忽略的列没有默认值，则 INSERT 语句将被拒绝。

4.4 RENAME USER 语法

RENAME USER old_user TO new_user

[, old_user TO new_user] ...

RENAME USER 语句用于对原有 DosSQL 账户进行重命名。要使用 RENAME USER，您必须拥有全局 CREATE USER 权限或 DosSQL 数据库 UPDATE 权限。如果旧账户不存在

或者新账户已存在，则会出现错误。old_user 和 new_user 值的给定方法与 GRANT 语句一样。

5 系统维护指令

数字有机体系统是一个人能化系统，DosSQL 系统是数字有机体系统的重要组成部分，它能将大量的服务器整合起来为应用提供数据服务。为了保障系统的正常运行，系统提供了几个维护指令，方便用户准确地检测系统，判断系统运行是否正常。在系统异常时，能及时解决。

在带宽足够宽的一个有限地域范围内的服务器组成一个或者多个站，若干站形成一个系统。因此，由所有的服务器组成的系统，范围从大到小依次是“系统”、“站”、“单独主机”，在指令中采用关键字“ALL”表示“系统”范围，用关键字“STATION”表示“站”范围。

每一列用于描述一个主机节点状态，状态包括：

1) 主机所属于的“站”的编号（Station_id），它是一个整数型的数字；
2) 主机的 IP 网络地址（Host），它是一个点分十进制的 IP 字符串；
3) 是否站首（Station_admin），它是一个布尔量，用 Yes 表示属于站首，否则用 No 表示；
4) 是否链首（Section_admin），它是一个布尔量，用 Yes 表示属于链首，否则用 No 表示；
5) 是否管理者（Section_manager），它是一个布尔量，用 Yes 表示属于管理者，否则用 No 表示；
6) 是否可用状态（Host_availability），它是一个布尔量，用 Yes 表示该主机可用，否则用 No 表示。

以下给出了执行系统维护命令的显示列。

站编号	主机 IP 地址	是否站首（Y/N）	是否链首（Y/N）	是否管理者（Y/N）	可用状态（Y/N）
Station_id	Host	Station_admin	Section_admin	Section_manager	Host_availability

范围从小到大，系统维护指令可以查询某些节点主机的状态，也可以查询本站所有的节点主机状态，还可以查询系统内所有的节点主机状态。

使用本小节的指令需要超级用户的权限。

5.1 SHOW HOST STATUS

SHOW HOST STATUS 指令用于查询本地主机的状态。

如输入：

```
show host status \G
```

显示为：

```
*************************** 1. row ***************************
          Station_id: 12
                Host: 192.168.2.55
       Station_admin: No
       Section_admin: No
    Section_manager: No
Host_availability: Yes
1 row in set (0.00 sec)
```

5.2 SHOW HOST STATUS ON <(‘ip_1’ [,’ ip_2’ ,…])>

SHOW HOST STATUS ON <(‘ip_1’[,’ip_2’,…])>指令用于查询用户给定的主机的状态，“<(‘ip_1’[,’ip_2’,…])>”表示主机列表，当主机列表包括多个 IP 地址时，需要使用“()”，如果只包括一个 IP 地址时，可以省略“()”。

主机列表包括多个 IP 地址时，如输入：

```
show host status on ('192.168.2.54','192.168.2.55')\G
```

显示为：

```
*************************** 1. row ***************************
          Station_id: 12
                Host: 192.168.2.55
       Station_admin: No
       Section_admin: No
    Section_manager: No
Host_availability: Yes
*************************** 2. row ***************************
          Station_id: 12
                Host: 192.168.2.54
       Station_admin: Yes
       Section_admin: Yes
    Section_manager: Yes
Host_availability: Yes
2 rows in set (0.00 sec)
```

主机列表只包括一个 IP 地址时，如输入：

```
show host status on '192.168.2.55'\G
```

或者

```
show host status on ('192.168.2.55')\G
```

显示为：

```
*************************** 1. row ***************************
```

```
          Station_id: 12
                Host: 192.168.2.55
       Station_admin: No
       Section_admin: No
    Section_manager: No
Host_availability: Yes
1 rows in set (0.00 sec)
```

5.3 SHOW STATION HOST STATUS

SHOW STATION HOST STATUS 指令用于查询本站的所有主机节点的状态。包括站内活动的主机地址和本地主机所知道的死亡节点地址。

如输入：

```
show station host status \G
```

显示为：

```
*************************** 1. row ***************************
          Station_id: 12
                Host: 192.168.2.55
       Station_admin: No
       Section_admin: No
    Section_manager: No
Host_availability: Yes
*************************** 2. row ***************************
          Station_id: 12
                Host: 192.168.2.54
       Station_admin: Yes
       Section_admin: Yes
    Section_manager: Yes
Host_availability: Yes
2 rows in set (0.00 sec)
```

如果关闭其中的一台主机，再次查询，第二行则显示一个主机不可用，并且不能得知它的其他相关信息。

```
show station host status \G
*************************** 1. row ***************************
          Station_id: 12
                Host: 192.168.2.55
       Station_admin: Yes
       Section_admin: Yes
    Section_manager: Yes
Host_availability: Yes
*************************** 2. row ***************************
```

```
        Station_id: NULL
              Host: 192.168.2.54
     Station_admin: No
     Section_admin: No
   Section_manager: No
Host_availability: No
2 rows in set (0.00 sec)
```

注意：本指令查询结果为不可用的主机节点时，“Station_id”为空值。不可用的情况可能是该主机已经故障，如断网、关机、dossqld 程序未启动，也可能是已经不再属于本系统，以上任意可能均存在。如果出现意外的主机地址，请查看配置文件“/etc/loginips.log”，如果想要意外的主机地址不再显示，您需要停止 dossqld 服务，并清空配置文件“/etc/loginips.log”。一般地，我们可以忽略故障的主机地址。

5.4 SHOW ALL HOST STATUS

SHOW STATION HOST STATUS 指令用于查询系统内的所有主机节点的状态。包括站内活动的主机地址和本地主机所知道的死亡节点地址。

如输入：

```
show all host status \G
```

显示为：

```
*************************** 1. row ***************************
        Station_id: 12
              Host: 192.168.2.55
     Station_admin: No
     Section_admin: No
   Section_manager: No
Host_availability: Yes
*************************** 2. row ***************************
        Station_id: 12
              Host: 192.168.2.54
     Station_admin: Yes
     Section_admin: Yes
   Section_manager: Yes
Host_availability: Yes
2 rows in set (0.00 sec)
```

如果关闭其中的一台主机，再次查询，第二行则显示一个主机不可用，并且不能得知它的其他相关信息。

```
show all host status \G
*************************** 1. row ***************************
        Station_id: 12
```

```
                Host: 192.168.2.55
      Station_admin: Yes
      Section_admin: Yes
    Section_manager: Yes
Host_availability: Yes
*************************** 2. row ***************************
         Station_id: NULL
                Host: 192.168.2.54
      Station_admin: No
      Section_admin: No
    Section_manager: No
Host_availability: No
2 rows in set (0.00 sec)
```

注意：本指令查询结果为不可用的主机节点时，“Station_id”为空值。不可用的情况可能是该主机已经故障，如断网、关机、dossqld 程序未启动，也可能是已经不再属于本系统，以上任意可能均存在。如果出现意外的主机地址，请查看配置文件“/etc/loginips.log”，如果想要意外的主机地址不再显示，您需要停止 dossqld 服务，并清空配置文件“/etc/loginips.log”。一般地，我们可以忽略故障的主机地址。

6 副本管理指令

DosSQL 支持多副本复制。每个数据库的副本数可以不同。副本的放置位置可由系统自动调整。同时，为了有效地管理这些数据库的副本，系统提供了以下的副本管理指令。通过这些指令，你可以指定副本的数量，改变副本同步的方式，指定副本放置的位置和查询数据库副本位置等。

副本管理包括以下这些内容：

1) 查询某节点上、站内、系统内的所有数据库；
2) 定义（新建、删除）数据库，新增数据库副本到指定节点，减少指定节点的数据库副本；
3) 查询及修改数据库的属性；
4) 查询某节点上、站内、系统内的数据库的副本分布状况，以及它们的状态，是否为可用状态，如果某个没有数据库的所有副本均为不可用状态，需要人工干预并设定其中一个副本为可用状态。

6.1 数据库查询

本小节意在向用户阐述数据库的查询指令。根据“节点—站—系统”这种从小到大的范围，数据库的查询指令包括：

- 查询节点上的数据库指令；
- 查询本站的数据库的指令；
- 查询系统内的数据库指令。

以下是查询数据库指令的显示列：

数据库名称
Database

6.1.1 SHOW DATABASES

SHOW DATABASES 指令用于查询本地主机包括的所有的数据库。显示的数据库列表按照字母顺序排序排列。其中每个 DosSQL 主机都包含“information_schema”、“mysql”、“performance_schema”三个系统数据库，它们都是各个主机私有的，并不与其他主机上的该库互为副本关系。

使用本小节的指令需要具有查询数据库用户权限。

如输入：

show databases;

显示为：

```
+--------------------+
| Database           |
+--------------------+
| information_schema |
| aabb               |
| abc                |
| abcd               |
| mysql              |
| performance_schema |
+--------------------+
6 rows in set (0.00 sec)
```

6.1.2 SHOW DATABASES ON <'ip'>

SHOW DATABASES ON <'ip'>指令用于查询用户给定的 IP 地址上的数据库。它是 SHOW DATABASES 指令的一种延伸，使得管理员可以单点查询除本机之外的其他主机上的数据库。

使用本小节的指令需要超级用户的权限。

如输入：

```
show databases on '192.168.2.55';
```

显示为：

```
+--------------------+
| Database           |
+--------------------+
| information_schema |
| aabb               |
| abc                |
| abcd               |
| mysql              |
| performance_schema |
+--------------------+
6 rows in set (0.00 sec)
```

6.1.3 SHOW STATION DATABASES

SHOW STATION DATABASES 指令用于查询本站内的所有数据库。

使用本小节的指令需要超级用户的权限。

如输入：

```
SHOW STATION DATABASES;
```

显示为：

```
+----------+
| Database |
+----------+
| information_schema |
| aabb       |
| abc        |
| abcd       |
| kla        |
| kll        |
| lv         |
| mmnn       |
| mmxx       |
| mysql                  |
| performance_schema |
| zzss       |
| zzxx       |
| zzyy       |
| zzzaaa     |
| zzzbbb     |
+----------+
16 rows in set (0.00 sec)
```

6.1.4 SHOW ALL DATABASES

SHOW ALL DATABASES 指令用于查询系统内的所有的数据库。
使用本小节的指令需要超级用户的权限。

如输入：

```
show all databases;
```

显示为：

```
+----------+
| Database |
+----------+
| information_schema |
| aabb       |
| abc        |
| abcd       |
| kla        |
| kll        |
| lv         |
| mmnn       |
| mmxx       |
```

```
| mysql                |
| performance_schema |
| zzss         |
| zzxx         |
| zzyy         |
| zzzaaa     |
| zzzbbb     |
+----------+
16 rows in set (0.01 sec)
```

6.2 数据库定义及属性管理

本小节意在向用户阐述数据库的定义指令和属性管理指令，这些指令包括：

- 新建和删除数据库指令；
- 增加和减少数据库副本的指令；
- 查询和修改数据库属性的指令。

6.2.1 SHOW CREATE DATABASE

SHOW CREATE DATABASE 指令用于查询新建表的 SQL 语句，可用于数据库备份，也可以用于查询数据库的属性。但是本指令不方便获取任意属性的值，如果需要，请使用“show database options”指令查询数据库的属性，此指令及诸多属性将在接下来的小节详细介绍。

使用本小节的指令需要具有查看数据库的用户权限。

如输入：

```
show create database test \G
```

显示为：

```
*************************** 1. row ***************************
       Database: test
Create Database: CREATE DATABASE `test` /*!40100 DEFAULT CHARACTER SET utf8 */ /*!50700 slow_log_exe No min_sum_dup 1 max_sum_dup 4 out_section Yes out_station Yes admin_section 1 admin_station 12 admin_node '192.168.2.55' master_addr '192.168.2.55' random_master Yes wait_time 90 relay_time_threshold 120 */
1 row in set (0.00 sec)
```

使用“show database options”指令无法查询到数据库的创建 SQL，但是属性的值显示一致，如下所示：

```
DosSQL[(none)]$show database options test\G
*************************** 1. row ***************************
    Slow_log_exe: No
         Charset: utf8
```

```
                Collate: utf8_general_ci
            Min_sum_dup: 1
            Max_sum_dup: 4
            Out_section: Yes
            Out_station: Yes
          Admin_section: 1
          Admin_station: 12
             Admin_node: 192.168.2.55
            Master_addr: 192.168.2.55
          Random_master: Yes
              Wait_time: 90
Relay_time_threshold: 120
1 row in set (0.00 sec)
```

6.2.2 SHOW DATABASE OPTIONS <db>

SHOW DATABASE OPTIONS <db>指令用于查询指定数据库的属性。
使用本小节的指令需要具有修改数据库属性的用户权限。

数据库目前包括以下这些属性，其中缺省值是由配置文件“/etc/dossql.cnf”确定的，下的缺省值为系统出厂缺省值。

数据库属性	描述	缺省值
Slow_log_exe	数据库的模式属性，值为 No（0）时表示实时同步数据库，值为 Yes（1）时表示批量同步数据库	No
Charset	数据库的字符集	latin1
Collate	数据库的字符对比集	latin1_swedish_ci
Admin_section	数据库新建时所在的区域（目前区域未使用）	1
Admin_station	数据库新建时所在的站点，副本放置时优先满足此站	新建数据库所在的站
Admin_node	数据库新建时所在的节点，副本放置时优先满足此节点	新建数据库所在的主机 IP 地址
Out_section	是否允许副本放置在本区域外（目前区域未使用）	Yes
Out_station	是否允许副本放置在本站外	Yes
Min_sum_dup	数据库副本数量限制的最小数量（下限）	2
Max_sum_dup	数据库副本数量限制的最大数量（上限）	3
Master_addr	此属性只在 Slow_log_exe 为 Yes 时有效。网络 IP 地址，记录身份为 master 的节点	新建数据库所在的主机 IP 地址
Random_master	此属性只在 Slow_log_exe 为 Yes 时有效。当 master 节点故障后，是否允许自动竞选出一个新的 master 节点，值为 Yes（1）为表示允许，值为 No（0）表示不允许	Yes
Wait_time	此属性只在 Slow_log_exe 为 Yes 时有效。这里的	90

	值为正整数，单位为秒。如果 random_master 属性值允许自动竞选出一个新的 master 节点，则等待 wait_time 秒后开始竞选	
Relay_time_threshold	此属性只在 Slow_log_exe 为 Yes 时有效。master 和 slave 之间的数据更新总是 master 优先，当更新事务快速执行时，master 和 slave 之间的数据可能差距越来越大，为了避免这种情况继续恶化，这里限定了同一个事务在 master 和 slave 上执行的时间最大差值，当差值大于本属性后，master 就主动放慢事务的执行速度，这样就给了 slave 足够的追赶时间，从而使两者之间差距在可控的范围，而不是太大。这里的值为非负数，单位为秒，值为 0 时表示没有此项属性限制。 造成这种现象的原因可能是 slave 节点接收更新事务的日志太慢，也可能是 salve 节点性能差，不能快速执行接收到的更新日志。	120

如输入：

```
show database options test\G
```

显示为：

```
*************************** 1. row ***************************
          Slow_log_exe: No
               Charset: utf8
               Collate: utf8_general_ci
         Min_sum_dup: 1
         Max_sum_dup: 4
         Out_section: Yes
         Out_station: Yes
       Admin_section: 1
       Admin_station: 12
          Admin_node: 192.168.2.55
         Master_addr: 192.168.2.55
       Random_master: Yes
           Wait_time: 90
Relay_time_threshold: 120
1 row in set (0.00 sec)
```

6.2.3 CREATE DATABASE

CREATE DATABASE 指令用于新建一个数据库。该指令在使用时可以附带数据库的属性，也可以不附带，不附带是采用缺省配置。新建成功后，依据属性的值可能同时新建多个数据库的副本。

使用本小节的指令需要具有新建数据库的用户权限。

如输入：

create database test charset=utf8 min_sum_dup=1 max_sum_dup=4;

或者

create database test charset utf8 min_sum_dup 1 max_sum_dup 4;

显示为：

Query OK, 0 rows affected (0.02 sec)

采用以下两种方式查询：

DosSQL[(none)]$show create database test \G

*************************** 1. row ***************************

Database: test

Create Database: CREATE DATABASE `test` /*!40100 DEFAULT CHARACTER SET utf8 */ /*!50700 slow_log_exe No min_sum_dup 1 max_sum_dup 4 out_section Yes out_station Yes admin_section 1 admin_station 12 admin_node '192.168.2.55' master_addr '192.168.2.55' random_master Yes wait_time 90 relay_time_threshold 120 */

1 row in set (0.00 sec)

DosSQL[(none)]$show database options test\G

*************************** 1. row ***************************

Slow_log_exe: No

Charset: utf8

Collate: utf8_general_ci

Min_sum_dup: 1

Max_sum_dup: 4

Out_section: Yes

Out_station: Yes

Admin_section: 1

Admin_station: 12

Admin_node: 192.168.2.55

Master_addr: 192.168.2.55

Random_master: Yes

Wait_time: 90

Relay_time_threshold: 120

1 row in set (0.00 sec)

根据查询结果可知，新建数据库是附带的属性起效。

采用缺省属性新建的数据库是实时模式数据库，如果需要，新建批量模式的数据库的指令为：

create database test slow_log_exe=Yes;

或者

create database test slow_log_exe Yes;

6.2.4 CREATE DATABASE ON <('ip_1'[,' ip_2' ,…])>

CREATE DATABASE ON 指令是用于新建数据库的。它适用于数字有机体数据库的分布式模式，属于 CREATE DATABASE 指令的扩展。

该指令的用途主要体现在以下两个方面：

1) 在指定的多台服务器上新建数据库副本，并根据给定的主机列表，自动修改数据库的最小、最大副本数量限制属性。
2) 新增加数据库的副本到给定的主机地址，不会自动修改数据库的最小、最大副本数量限制属性，与命令 ADD DATABASE ON 的功能是一致的。

该指令的语法是在 CREATE DATABASE 的基础上新增加了主机列表来实现的。完全的语法如下：

```
CREATE DATABASE [IF NOT EXISTS] db_name
      [ON <('ip_1'[,'ip_2',…])>]
      [create_specification [, create_specification] ...]

create_specification:
      [DEFAULT] CHARACTER SET charset_name
    | [DEFAULT] COLLATE collation_name
    | [MIN_SUM_DUP 2]
    | [MAX_SUM_DUP 3]
    …
    | [SLOW_LOG_EXE 1]
```

例如：CREATE DATABASE xxx ON (“192.168.2.89”,”192.168.2.90”) CHARACTER SET utf8 SLOW_LOG_EXE 1;

6.2.5 ALTER DATABASE

ALTER DATABASE 指令用于修改数据库的属性。修改属性的方式和新建时的方式一致，直接附带在修改命令的后边，各个属性的标签和值之间既可以使用等号（“=”）也可以使用空格符号（“ ”）。

修改属性“min_sum_dup”和“max_sum_dup”可以改变数据库副本的数量。修改属性“slow_log_exe”可以改变数据库的模式，由实时模式转换为批量模式后，执行修改命令的主机被设置为 Master 副本，其他副本均为可用状态；由批量模式转换为实时模式后，仅仅原 Maser 副本变为可用副本，其他副本将从可用副本得到恢复。

使用本小节的指令需要具有修改数据库的用户权限。

如输入：

```
alter database test min_sum_dup=2;
```

显示为：

Query OK, 1 row affected (0.06 sec)

查询修改后的属性，显示为：

```
DosSQL[(none)]$show database options test\G
*************************** 1. row ***************************
            Slow_log_exe: No
                 Charset: utf8
                 Collate: utf8_general_ci
              Min_sum_dup: 2
              Max_sum_dup: 4
              Out_section: Yes
              Out_station: Yes
           Admin_section: 1
           Admin_station: 12
              Admin_node: 192.168.2.55
             Master_addr: 192.168.2.55
           Random_master: Yes
               Wait_time: 90
Relay_time_threshold: 120
1 row in set (0.00 sec)
```

6.2.6 DROP DATABASE

DROP DATABASE 指令用于删除数据库及其所有的副本。

使用本小节的指令需要具有删除数据库的用户权限。

如输入：

```
DosSQL[(none)]$drop database test;
```

显示为：

```
Query OK, 0 rows affected (0.03 sec)
```

6.2.7 DROP DATABASE <db> ON <'ip'>

DROP DATABASE <db> ON <'ip'>用于删除给定 IP 地址上的具体数据库，它与 ADD DATABASE <db> on <'ip'>命令结合可以实现数据库副本的转移。

注意：如果待删除的数据库副本的数量已经处于到副本属性中的“MIN_SUM_DUP”的限制，使用本指令将会失败，如果用户确实需要删除副本，应该先调用 ALTER 指令修改属性 “MIN_SUM_DUP”，使它变小。

使用本小节的指令需要超级用户的权限。

如输入：

```
drop database abc on "192.168.2.54";
```

显示为未成功，原因是删除副本后将不符合数据库的属性规范。

```
ERROR 1047 (08S01): Can not drop dup any more, the min_sum_dup is '2'
```

首先查看该数据库的的最小副本数量属性 Min_sum_dup 为 2。

```
DosSQL[(none)]$show database options abc \G
*************************** 1. row ***************************
          Slow_log_exe: No
               Charset: utf8
               Collate: utf8_general_ci
         Min_sum_dup: 2
         Max_sum_dup: 3
         Out_section: Yes
         Out_station: Yes
       Admin_section: 1
       Admin_station: 12
          Admin_node: 192.168.2.54
         Master_addr: 192.168.2.54
       Random_master: Yes
           Wait_time: 90
Relay_time_threshold: 120
1 row in set (0.00 sec)
```

然后查看该数据库的副本数量为 2。

```
DosSQL[(none)]$show database status abc;
+-------+----------+-------------+-------------+-----------------+
| Db_id | Database | Slow_log_exe | Host              | Db_availability |
+-------+----------+-------------+-------------+-----------------+
| 0     | abc      | No           | 192.168.2.54 | Yes             |
| 0     | abc      | No           | 192.168.2.55 | Yes             |
+-------+----------+-------------+-------------+-----------------+
2 rows in set (0.00 sec)
```

分析：数据库的属性限定了该库的副本数最少应该为 2，并且该数据库的副本数量正也为 2，如果这种情况下删除一个副本，系统将会依据属性规则自动增加一个副本，最终还是 2 个副本。本指令拒绝这种无谓的操作，如果确实有需要，应该先修改数据库的属性 Min_sum_dup，使其变小，然后再调用本指令删除，即可成功。如以下的操作示例：

先修改 Min_sum_dup 属性，使其变小。

```
DosSQL[(none)]$alter database abc min_sum_dup=1;
Query OK, 1 row affected (0.04 sec)
```

调用本指令删除某节点上的数据库副本。

```
DosSQL[(none)]$drop database abc on "192.168.2.54";
Query OK, 1 row affected (0.00 sec)
```

6.2.8 ADD DATABASE ON <('ip_1'[,' ip_2',…])>

ADD DATABASE <db> ON <('ip_1'[,'ip_2',…])>指令用于添加给定数据库的副本到给定的 IP 地址。

注意：如果待添加的数据库副本的数量已经达到副本属性中的“MAX_SUM_DUP”的限制，使用本指令将会失败，如果用户确实需要新增副本，应该先调用 ALTER 指令修改属性 “MAX_SUM_DUP”，使它变大。具体方法请参考“DROP DATABASE ON”指令的介绍。

使用本小节的指令需要超级用户的权限。

同时添加到多个主机时，输入：

add database abc on (‘192.168.2.54’,’192.168.2.55’);

显示为：

Query OK, 2 rows affected (0.00 sec)

只添加到一个主机时，输入：

add database abc on ‘192.168.2.54’;

显示为：

Query OK, 1 rows affected (0.00 sec)

6.3 数据库分布及状态管理

本小节意在向用户阐述数据库的状态查询指令，状态信息主要说明数据库副本的分布状况和是否是可用状态。根据“节点—站—系统”这种从小到大的范围，查询指令包括：

- 查询节点上的数据库的状态指令；
- 查询本站的数据库的的状态指令；
- 查询系统内的数据库的状态指令。

以下是查询数据库指令的显示列：

数据库组编号	数据库名称	数据库模式	主机 IP 地址	状态
Db_id	Database	Slow_log_exe(Y/N/?)	Host	Db_availability (Y/N)

使用本小节的指令需要超级用户的权限。当系统由多台服务器组成时，数据库副本的状态为不可用时（即 Db_availability 等于 No），该副本将不能被访问。当系统只由一台服务器组成时，无论数据库副本的状态是否为可用，它作为一个单机版数据库，都能被访问。

6.3.1 SET DATABASE STATUS <db>

SET DATABASE STATUS <db>用于设置当前副本为“可用”状态，它只有当所有副本

全为“非可用”时，才能执行成功。

如输入：
SET DATABASE STATUS test;

显示为：
Query OK, 1 row affected (0.00 sec)

说明：造成所有副本状态均为“非可用状态”的情况非常复杂，可能是工作库在恢复的过程断电，也可能是人为（如黑客）修改数据等因素。因此，管理员需要到副本所在的服务器上仔细考察并确定将要把哪个副本被设置为可用状态。如果被确定为可用状态的副本不具备成为可用状态的能力，由于系统本身无法判断而强制按照用户的需求设置为了可用状态，可能导致其它的“非可用状态”状态副本无法成功地恢复。

6.3.2 SHOW DATABASE STATUS <db>

SHOW DATABASE STATUS <db>指令是 SHOW DATABASE STATUS 指令的一个延伸，用于查询本地的具体的一个数据库的副本状态。

如输入：
show database status vv;

显示为：
```
+-------+----------+-------------+-------------+-----------------+
| Db_id | Database | Slow_log_exe | Host              | Db_availability |
+-------+----------+-------------+-------------+-----------------+
| 1     | vv       | No           | 192.168.2.54 | Yes             |
| 1     | vv       | No           | 192.168.2.55 | Yes             |
+-------+----------+-------------+-------------+-----------------+
2 rows in set (0.00 sec)
```

6.3.3 SHOW DATABASE STATUS

SHOW DATABASE STATUS 指令用于显示本地主机上的除系统库数据库之外的数据库的状态。

如输入：
show database status;

显示为：
```
+-------+----------+-------------+-------------+-----------------+
| Db_id | Database | Slow_log_exe | Host              | Db_availability |
+-------+----------+-------------+-------------+-----------------+
| 1     | aabb     | No           | 192.168.2.55 | Yes             |
```

```
| 2       | abc         | No              | 192.168.2.55 | Yes                |
| 3       | abcd        | No              | 192.168.2.54 | Yes                |
| 3       | abcd        | No              | 192.168.2.55 | Yes                |
| 4       | vv          | No              | 192.168.2.54 | Yes                |
| 4       | vv          | No              | 192.168.2.55 | Yes                |
+-------+----------+-------------+-------------+-----------------+
6 rows in set (0.00 sec)
```

其中，相同数据库副本具有一致的“Db_id”，“Database”显示副本的名称，“Slow_log_exe”显示数据库的模式（Yes 表示批量数据库，No 表示实时数据库），“Host”则显示副本所在的主机，“Db_availability”则显示该副本是否为可用状态（Yes 表示可用，是正常的，否则用 No 来表示）。

6.3.4 SHOW STATION DATABASE STATUS

SHOW STATION DATABASE STATUS 指令用于查询本站的数据库的状态。

如输入：

```
show station database status;
```

显示为：

```
+-------+----------+-------------+-------------+-----------------+
| Db_id | Database | Slow_log_exe | Host              | Db_availability |
+-------+----------+-------------+-------------+-----------------+
| 1       | aabb        | No              | 192.168.2.55 | Yes                |
| 2       | abc         | No              | 192.168.2.55 | Yes                |
| 3       | abcd        | No              | 192.168.2.54 | Yes                |
| 3       | abcd        | No              | 192.168.2.55 | Yes                |
| 4       | kla         | No              | 192.168.2.54 | Yes                |
| 5       | kll         | No              | 192.168.2.54 | Yes                |
| 6       | lv          | No              | 192.168.2.54 | Yes                |
| 7       | mmnn        | No              | 192.168.2.54 | Yes                |
| 8       | mmxx        | No              | 192.168.2.54 | Yes                |
| 9       | vv          | No              | 192.168.2.54 | Yes                |
| 9       | vv          | No              | 192.168.2.55 | Yes                |
| 10      | zzss        | No              | 192.168.2.54 | Yes                |
| 11      | zzxx        | No              | 192.168.2.54 | Yes                |
| 12      | zzyy        | No              | 192.168.2.54 | Yes                |
| 13      | zzzaaa      | No              | 192.168.2.54 | Yes                |
| 14      | zzzbbb      | No              | 192.168.2.54 | Yes                |
+-------+----------+-------------+-------------+-----------------+
16 rows in set (0.01 sec)
```

6.3.5 SHOW ALL DATABASE STATUS

SHOW ALL DATABASE STATUS 指令用于查询系统内的数据库的状态。

如输入：
show all database status;

显示为：

```
+-------+----------+-------------+-------------+-----------------+
| Db_id | Database | Slow_log_exe | Host              | Db_availability |
+-------+----------+-------------+-------------+-----------------+
| 1       | aabb       | No                | 192.168.2.55 | Yes                  |
| 2       | abc        | No                | 192.168.2.55 | Yes                  |
| 3       | abcd       | No                | 192.168.2.54 | Yes                  |
| 3       | abcd       | No                | 192.168.2.55 | Yes                  |
| 4       | kla        | No                | 192.168.2.54 | Yes                  |
| 5       | kll        | No                | 192.168.2.54 | Yes                  |
| 6       | lv         | No                | 192.168.2.54 | Yes                  |
| 7       | mmnn       | No                | 192.168.2.54 | Yes                  |
| 8       | mmxx       | No                | 192.168.2.54 | Yes                  |
| 9       | vv         | No                | 192.168.2.54 | Yes                  |
| 9       | vv         | No                | 192.168.2.55 | Yes                  |
| 10      | zzss       | No                | 192.168.2.54 | Yes                  |
| 11      | zzxx       | No                | 192.168.2.54 | Yes                  |
| 12      | zzyy       | No                | 192.168.2.54 | Yes                  |
| 13      | zzzaaa     | No                | 192.168.2.54 | Yes                  |
| 14      | zzzbbb     | No                | 192.168.2.54 | Yes                  |
+-------+----------+-------------+-------------+-----------------+
16 rows in set (0.00 sec)
```

7 分布式事务

DosSQL 支持标准的 SQL 语法、预处理语句、存储过程、事件、函数、触发器等。除部分 MySQL 语言不支持外，其余均支持，具体的 SQL 语法详细说明，请查阅《MySQL 5.7 参考手册》。

在 DosSQL 系统中，所有的更新语句通常情况下需要同时在多个副本上执行，执行的结果是多个副本之间要么都执行成功，要么都执行失败。这种原子性的操作被称作事务，由于这种事务具有分布式的特点，因此被称作分布式事务。

在 DosSQL 系统中，同一个库的数据同时存储在多台服务器上。在进行数据查询和更新时，通常需要保证在各台服务器上并发的操作能够满足事务要求，即需要系统支持分布式事务。数字有机体工作库支持数据库查询（即读）和写的分布式事务。其中，数据的写操作的分布式事务根据时间的及时性被分为实时的分布式事务和批量的分布式事务。采用写的实时的分布式事务的数据库被称为实时数据库，反之为批量数据库。数据的读操作一般都是发生在某一数据库副本的，但是从 DosSQL-3.0.0 版本开始，DosSQL 支持分片表，使得物理上属于多个表的数据归属于一个逻辑表。在对分片的逻辑表进行查询时，DosSQL 对其进行分布式并行查询优化，并行快速地处理，能极大地加快查询速度。

不过，分布式事务处理实现复杂，需要在多台服务器间同步操作，因此其性能将无法达到单机事务操作的水平。某些应用场景中，应用允许查询获得的不是最新的数据，即并不要求并发的查询满足事务属性要求。这时，可以采用异步数据同步方式。简单的说，数据库的多个副本中只有一个副本（称为主本）总是最新的，其它副本（称为从本）的数据可能不包含最近的更新。每个更新总是先在主本上完成，然后再批量的、异步的更新到从本上。我们将这种方式称为“批量同步方式”，采用这种方式更新副本的数据库称为“批量同步数据库”。

相比实时数据库，批量同步数据库具有更好的数据更新性能；反之，相对于批量同步数据库，实时数据库具有更好的数据一致性、完整性和可靠性。因此，不同的应用应当根据应用需求选择数据库类型。

在数字有机体工作库中，采用数据库属性“slow_log_exe”表明数据库的类型。当该属性的值为“0”时数据库是实时数据库。当该属性的值为“1”时数据库为批量同步数据库。在必要时，可以通过 “ALTER DATABASE”指令修改数据库的“slow_log_exe”属性，使数据库的更新方式在实时同步和批量同步间转化。

7.1 实时同步

在没有特别说明是批量同步数据库的情况下，默认都属于实时数据库。这里的“实时”可以用数据库的多个副本之间的数据更新的时间差来衡量。实时数据库各个副本间的更新延迟在一个通信延迟范围内，通常是几毫秒甚至微秒级的。因此大多数应用场景都可忽略这种差异。这种数据库模式被大多数的应用所接受。

新建实时数据库的指令如下：

```
DosSQL$ CREATE DATABASE test_1;
```

在使用分布式事务时，以下几点需要注意：

1） 如果连接不是自动提交每条语句的，则每个事务都必须显式的提交或者回滚，否则事务将一直延续下去。前一个事务结束后，新的事务即开始；
2） 如果一个事务中当前语句执行失败，则整个事务将隐含的被回滚，新的语句将开始一个新事务；
3） 如果一个事务中嵌入了一条非事务性语句（主要是数据库结构操作语句，如创建和修改表结构等），则事务在非事务性语句的前后分开，不会作为一个事务处理；
4） 如果在连接提交事务时服务器故障，则连接提交的事务的结果是不确定的。即可能是提交成功，也可能失败；
5） 如果连接执行事务的中途服务器故障，则连接当前的事务将被放弃。连接自动重连后，原来的事务无法继续执行，需要重新开启；
6） 不要假定事务的提交总是成功的。由于需要在多台服务器间同步，提交事务也可能因为通信故障等失败，因此应该检查事务提交的结果；
7） 尽量不要使用持续时间很长的事务（例如需要持续数秒的事务），因为事务内的操作可能对记录或者表加锁，从而造成冲突事务被阻塞。在没有需要时，避免使用事务，例如读取数据；
8） 在使用事务的情况下，尽量不要使用表锁，因为这将显著的降低事务的并发度，从而降低系统性能。

7.2 批量同步

相对于实时数据库，批量同步数据库的副本间存在明显的同步延迟。因此，这种数据库适用于对数据一致性要求不严格的地方，比如论坛、微博、图片网站等。这时，建议采用读写分离的原则。对数据库的写操作由专门的一些连接完成，而读操作则分散到各个副本上并行执行，从而提升系统的整体性能。这种模式的数据库不保证每个副本时时刻刻都一致，但是保证数据库各个副本的数据最终达到一致。它和 MySQL 的复制有点类似。不同点在于，MySQL 的复制是固定配置的，数字有机体工作库中的批量同步是自动的选择副本位置的，不是固定节点地；MySQL 的复制是以服务器为单位的，数字有机体工作库是以数据库为单位的。

批量同步数据库的基本原理和 MySQL 复制是一致的，以一个副本为主本（master），其余副本为从本（slave）。数据更新时优先发生在 master 上，slave 在随后的时间中逐渐从 master 上获取更新。

批量同步数据库有以下这些独特的属性：

批量同步数据库属性	描述
Master_addr	网络 IP 地址，记录身份为 master 的节点
Random_master	当 master 节点故障后，是否允许自动竞选出一个新的 master 节点，

	值为 1 表示允许，值为 0 表示不允许
Wait_time	这里的值为正整数，单位为秒。如果 random_master 属性值允许自动竞选出一个新的 master 节点，则等待 wait_time 秒后开始竞选
Relay_time_threshold	master 和 slave 之间的数据更新总是 master 优先。当更新事务快速执行时，master 和 slave 之间的数据可能差距越来越大，为了避免这种情况继续恶化，这里限定了同一个事务在 master 和 slave 上执行的时间最大差值，当差值大于本属性后，master 就主动放慢事务的执行速度，这样就给了 slave 足够的追赶时间，从而使两者之间差距在可控的范围，而不是太大。这里的值为非负数，单位为秒，值为 0 时表示没有此项属性限制。 造成这种现象的原因可能是 slave 节点接收更新事务的日志太慢，也可能是 salve 节点性能差，不能快速执行接收到的更新日志。

新建批量同步数据库的指令如下：

```
DosSQL$ CREATE DATABASE test_2 slow_log_exe=1;
```

更改批量数据库的属性的指令如下：

```
DosSQL$ ALTER DATABASE test_2 random_master=1 relay_time_threshold=200;
```

8 分片引擎

从 DosSQL-3.0 开始，数字有机体工作库系统开始支持分片的功能。特别注意的是，它和分区的区别：分区和分片的相同点是把一个大的表空间分成多个小的表空间，从而减少在访问表空间时的互斥，达到提高查询和写表的性能；分区和分片的不同点是分区之后的表空间和原表空间在同一服务器上，而分片之后的表空间和原表空间的位置没有必然关系，使得分片表空间可能位于同一服务器，也可能位于多台服务器。

表的分区和分片的出现都是为了解决大数据的各种应用。分区适应于单机数据库系统的应用，可以为本地并行查询优化提供数据基础；分片同时具备分区的优点，并更加适用于分布式多服务器的并行查询优化，它对大量数据的存储、分析的应用具有具备非常高性能，但是对于那些数据量较少的应用，使用分片反而会降低性能，这里推荐小数据量的应用使用传统的工作库。

为了尽可能体现分片的读写性能，分片表对应的副本数量的配置缺省设置为 1，如果用户对数据的安全性要求特别的高，这里可以适当地设置分片表对应的副本数量。每个分片表对应的副本缺省采用实时同步。

从前面的分析可知，分片后的表的数据会被存储在多个不同的数据库中。这样，它能够提供以下的一些重要功能：

1） 与单个磁盘或单个节点相比，可以存储更多的数据，它把数据分开存储到不同的服务器磁盘；
2） 对于那些已经失去保存意义的数据，通常可以通过删除与那些数据有关的分片，很容易地删除那些数据。相反地，在某些情况下，添加新数据的过程又可以通过为那些新数据专门增加一个新的分片，来很方便地实现；
3） 通过跨多个子分片来分散数据查询，来获得更大的查询吞吐量；
4） 通过直接访问多个子分片来写数据，充分利用服务器的系统资源（如 CPU、内存、磁盘），并实现同一逻辑表的在多台服务器上并行 IO，从而突破单机服务器的最大 IO 限制；
5） 为分布式并行查询优化提供坚实的基础，它把逻辑上属于一个表的内容分别存储到不同的服务器的表空间，各个表空间互不干扰，使得从逻辑表的行查询时可以充分利用各台服务器的资源并且并行地查询成为可能。理论上讲，大数据查询时，并行度越大（即分片越多，分布得越分散），性能越好。

分片和分区对于大数据的处理具有很大的优势，但是也存在一些限制。比如它们不支持触发器、存储过程、函数等数据库程序语言。

对分片的支持，当前版本只支持 RANGE、LIST 分片。

HASH 分片、KEY 分片，以及它们的混合分片暂时还不支持，修改分片表的修改操作也还没能支持，这些操作将在后续的版本中陆续支持。

分片表新建是否需要指定用于访问分片数据库的用户名、口令等取决于配置参数“dosdb_use_system_user”。参数“dosdb_use_system_user”被设定为 1 时，新建分片表不

需要额外的用户名，它采用系统用户名，但是存在一定的安全隐患。因此，如果对安全的要求很高，可以在启动 DosSQL 前，修改参数“dosdb_use_system_user”为 0，即不使用系统用户，那么，此时，分片表的新建就必须要指定用于访问分片数据库的用户名、口令和主机地址。

8.1 分片类型

RANGE 分片：基于属于一个给定连续区间的列值，把多行分配给分片。参见 8.1.1 节，“RANGE 分片”。

LIST 分片：类似于按 RANGE 分片，区别在于 LIST 分片是基于列值匹配一个离散值集合中的某个值来进行选择。参见 8.1.2 节，“LIST 分片”。

不支持 HASH 分片。

8.1.1 RANGE 分片

RANGE 分片是从属于一个连续区间值的集合，这些区间要连续且不能相互重叠。RANGE 分片通过使用“PARTITION BY RANGE (expr)”来实现，其中“expr” 是某列值或一个基于某个列值、并返回一个整数值的表达式，然后通过“VALUES LESS THAN (value)”的方式来定义每个分片，其中“value” 是某列值或一个基于某个列值、并返回一个整数值的表达式。

通过添加一个 PARTITION BY RANGE 子句把这个表分割成 4 个区间，如下所示:

```
CREATE TABLE tbl_name (
    id INT NOT NULL,
    name VARCHAR(30),
    time_t DATE NOT NULL DEFAULT '1970-01-01',
    store_id INT
)
ENGINE=DOSDB
PARTITION BY RANGE (store_id) (
    PARTITION p0 VALUES LESS THAN (1000) COMMENT = 'database "db_name", table "tb0"' ENGINE = DOSDB,
    PARTITION p1 VALUES LESS THAN (2000) COMMENT = 'database " db_name ", table "tbl"' ENGINE = DOSDB,
    PARTITION p2 VALUES LESS THAN (3000) COMMENT = 'database " db_name ", table "tb2"' ENGINE = DOSDB,
    PARTITION p3 VALUES LESS THAN MAXVALUE COMMENT = 'database " db_name ", table "tb3"' ENGINE = DOSDB
);
```

注意，每个分片都是按顺序进行定义，从最低到最高。这是 PARTITION BY RANGE 语

法的要求；在这点上，它类似于 C 或 Java 中的“switch ... case”语句。

MAXVALUE 表示最大的可能的整数值。现在，store_id 列值大于或等于 3000（定义了的最高值）的所有行都将保存在分片 p3 中，如果不定义该分片则大于或等于 3000 的值插入将报错“Table has no partition for value 3000”。

在 VALUES LESS THAN 子句中使用一个表达式也是可能的。这里最值得注意的限制是 DosSQL 必须能够计算表达式的返回值作为 LESS THAN (<)比较的一部分；因此，表达式的值不能为 NULL。由于这个原因，store_id 列已经被定义为非空（NOT NULL）。

对于 COMMENT 子句，它用于指定分片的链接库名和表名；参数“database”后面跟分片链接库名；参数“table”后面跟分片链接表名。如果没有指定此参数，由系统生成默认库名和表名。

说明：在现版本中创建分片表时，需要注意 COMMENT 子句的“database”参数和“table”参数。在创建表时，首先执行“SHOW ALL DATABASE STATUS;”语句，查看系统内所有库。然后分三种情况处理：

- 当系统内没有与要创建的分片链接库名相同时，可以直接执行创建分片表语句；
- 当系统内有与要创建的分片链接库名相同时，首先执行创建分片表语句，然后在分片链接库中再执行一遍创建表语句（这里不需要“PARTITION BY RANGE”子句）；
- 当系统内有与要创建的分片链接库名相同时，并且链接库内也有相同表名时，此时你需要换个表名，再执行。

8.1.2 LIST 分片

LIST 分片在很多方面类似于 RANGE 分片。和按照 RANGE 分片一样，每个分片必须明确定义。它们的主要区别在于，LIST 分片中每个分片的定义和选择是基于某列的值从属于一个值列表集中的一个值，而 RANGE 分片是从属于一个连续区间值的集合。LIST 分片通过使用“PARTITION BY LIST(expr)”来实现，其中“expr” 是某列值或一个基于某个列值、并返回一个整数值的表达式，然后通过“VALUES IN (value_list)”的方式来定义每个分片，其中“value_list”是一个通过逗号分隔的整数列表。

下面通过添加一个 PARTITION BY LIST 子句把这个表分割成 4 个区间，如下所示:

```
CREATE TABLE tbl_name (
    id INT NOT NULL,
    name VARCHAR(30),
    time_t DATE NOT NULL DEFAULT '1970-01-01',
    store_id INT
)
ENGINE=DOSDB
PARTITION BY LIST(store_id)(
    PARTITION p0 VALUES IN (3,5,6,9,17) COMMENT = 'database "db_name", table "tb0"'
ENGINE = DOSDB,
    PARTITION p1 VALUES IN (1,2,10,11,19,20) COMMENT = 'database "db_name", table
"tb1"' ENGINE = DOSDB,
    PARTITION p2 VALUES IN (4,12,13,14,18) COMMENT = 'database "db_name", table
```

```
"tb2"' ENGINE = DOSDB,
    PARTITION p3 VALUES IN (7,8,15,16) COMMENT = 'database "db_name", table "tb3"'
ENGINE = DOSDB
);
```

注意，在当前版本中，当使用 LIST 分片时，只能匹配整数列表。

如果试图插入列值（或分片表达式的返回值）不在分片值列表中的一行时，那么“INSERT”查入将失败并报错。例如，假定 LIST 分片的采用上面的方案，下面的查入将失败：

INSERT INTO tbl_name VALUES (224, 'Linus', '2014-10-12', 21);

这是因为“store_id”列值 21 不能在用于定义分片 p0,p1,p2 或 p3 的值列表中找到。要重点注意的是，LIST 分片没有类似如“VALUES LESS THAN MAXVALUE”这样的包含其它值在内的定义。将要匹配的任何值都必须在值列表中找到。

说明：在现版本中创建分片表时，需要注意 COMMENT 子句的“database”参数和“table”参数。在创建表时，首先执行“SHOW ALL DATABASE STATUS;”语句，查看系统内所有库。然后分三种情况处理：

- 当系统内没有与要创建的分片链接库名相同时，可以直接执行创建分片表语句；
- 当系统内有与要创建的分片链接库名相同时，首先执行创建分片表语句，然后在分片链接库中再执行一遍创建表语句（这里不需要“PARTITION BY LIST”子句）；
- 当系统内有与要创建的分片链接库名相同时，并且链接库内也有相同表名时，此时你需要换个表名，再执行。

8.2 分片管理

8.2.1 RANGE 和 LIST 分片的管理

删除分片

从一个按照 RANGE 或 LIST 分片的表中删除一个分片，可以使用带一个 DROP PARTITION 子句的 ALTER TABLE 命令来实现。

下面语句从 tbl_name 表删除 p0 分片：

```
ALTER TABLE tbl_name DROP PARTITION p0;
```

注意，当删除了一个分片，也同时删除了该分片中所有的数据。如果希望从所有分片删除所有的数据，但是又保留表的定义和表的分片模式，使用 TRUNCATE TABLE 命令。如果希望改变表的分片而又不丢失数据，使用“ALTER TABLE ... REORGANIZE PARTITION”语句。

添加分片

要增加一个新的 RANGE 或 LIST 分片到一个前面已经分片了的表，使用“ALTER TABLE ... ADD PARTITION”语句。对于使用 RANGE 分片的表，可以用这个语句添加新的区间到已有分片的序列的前面或后面。

下面语句将 RANGE 分片 p3 添加到 tbl_name 表：

```
ALTER TABLE tbl_name ADD PARTITION (PARTITION p3 VALUES LESS THAN (2000));
```

注意，对于通过 RANGE 分片的表，只可以使用 ADD PARTITION 添加新的分片到分片列表的高端。

采用一个类似的方式，可以增加新的分片到已经通过 LIST 分片的表。

下面语句将 LIST 分片 p2 添加到 tbl_name 表：

```
ALTER TABLE tbl_name ADD PARTITION (PARTITION p2 VALUES IN (7, 14, 21));
```

注意，不能添加这样一个新的 LIST 分片，即该分片包含有已经包含在现有分片值列表中的任意值。

合并或拆分

使用“REORGANIZE PARTITION”拆分或合并分片，没有数据丢失。

不支持把分片合并的结果保存到为已经存在的分片（数据库和表）。

“REORGANIZE PARTITION”的基本语法是：

```
ALTER TABLE tbl_name REORGANIZE
PARTITION partition_list INTO (partition_definitions);
```

其中，tbl_name 是分片表的名称，partition_list 是通过逗号分开的、一个或多个将要被改变的现有分片的列表。partition_definitions 是一个是通过逗号分开的、新分片定义的列表，它遵循与用在“CREATE TABLE”中的 partition_definitions 相同的规则。应当注意到，在把多个分片合并到一个分片或把一个分片拆分成多个分片方面，没有限制。

当使用“ALTER TABLE ... REORGANIZE PARTITION”来对已经按照 RANGE 和 LIST 分片表进行重新分片时，下面是一些要记住的关键点：

- 用来确定新分片模式的 PARTITION 子句使用与用在 CREATE TABLE 中确定分片模式的 PARTITION 子句相同的规则。
- 新分片模式不能有任何重叠的区间（适用于按照 RANGE 分片的表）或值集合（适用于重新组织按照 LIST 分片的表）。
- partition_definitions 列表中分片的合集应该与在 partition_list 中命名分片的合集占有相同的区间或值集合。
- 对于按照 RANGE 分片的表，只能重新组织相邻的分片；不能跳过 RANGE 分片。
- 不能使用 REORGANIZE PARTITION 来改变表的分片类型；例如，不能把 RANGE 分片变为 HASH 分片，反之亦然。也不能使用该命令来改变分片表达式或列。
- 新的分片所在的位置（目标数据库和目标表）不能和原有的分片所在位置相同。
- 为了保护数据，防止数据丢失，重新分片后，原有的分片数据库将保留，如果用户不再需要，可自行手动删除。

8.2.2 分片维护

重建分片

这和先删除保存在分片中的所有记录，然后重新插入它们，具有同样的效果。它可用于整理分片碎片，目前暂不支持。

示例：

ALTER TABLE tbl_name REBUILD PARTITION p0, p1;

优化分片

如果从分片中删除了大量的行，或者对一个带有可变长度的行（也就是说，有VARCHAR，BLOB，或 TEXT 类型的列）作了许多修改，可以使用“ALTER TABLE ... OPTIMIZE PARTITION”来收回没有使用的空间，并整理分片数据文件的碎片。

示例：

ALTER TABLE tbl_name OPTIMIZE PARTITION p0, p1;

在一个给定的分片表上使用“OPTIMIZE PARTITION”等同于在那个分片上运行CHECK PARTITION，ANALYZE PARTITION，和 REPAIR PARTITION。

分析分片

读取并保存分片的键分布。

示例：

ALTER TABLE tbl_name ANALYZE PARTITION p3;

修补分片

修补被破坏的分片。

示例：

ALTER TABLE tbl_name REPAIR PARTITION p0,p1;

检查分片

可以使用几乎与对非分片表使用 CHECK TABLE 相同的方式检查分片。

示例：

ALTER TABLE tbl_name CHECK PARTITION p1;

这个命令可以告诉你表 tbl_name 的分片 p1 中的数据或索引是否已经被破坏。如果发生了这种情况，使用“ALTER TABLE ... REPAIR PARTITION”来修补该分片。

8.3 分片表的分布式优化查询

从 DosSQL-3.0.0 版本开始，DosSQL 支持分片表，使得物理上属于多个表的数据归属于一个逻辑表。在对分片的逻辑表进行查询时，DosSQL 对其进行分布式并行查询优化，并行快速地获取并处理数据，能极大地加快查询速度。

分片表的查询针对大数据的大多数查询性能具有明显的提升，对少部分的查询反而有所下降，但是结合一些相应的处理可以解决，在以下的介绍中将予以详细说明。

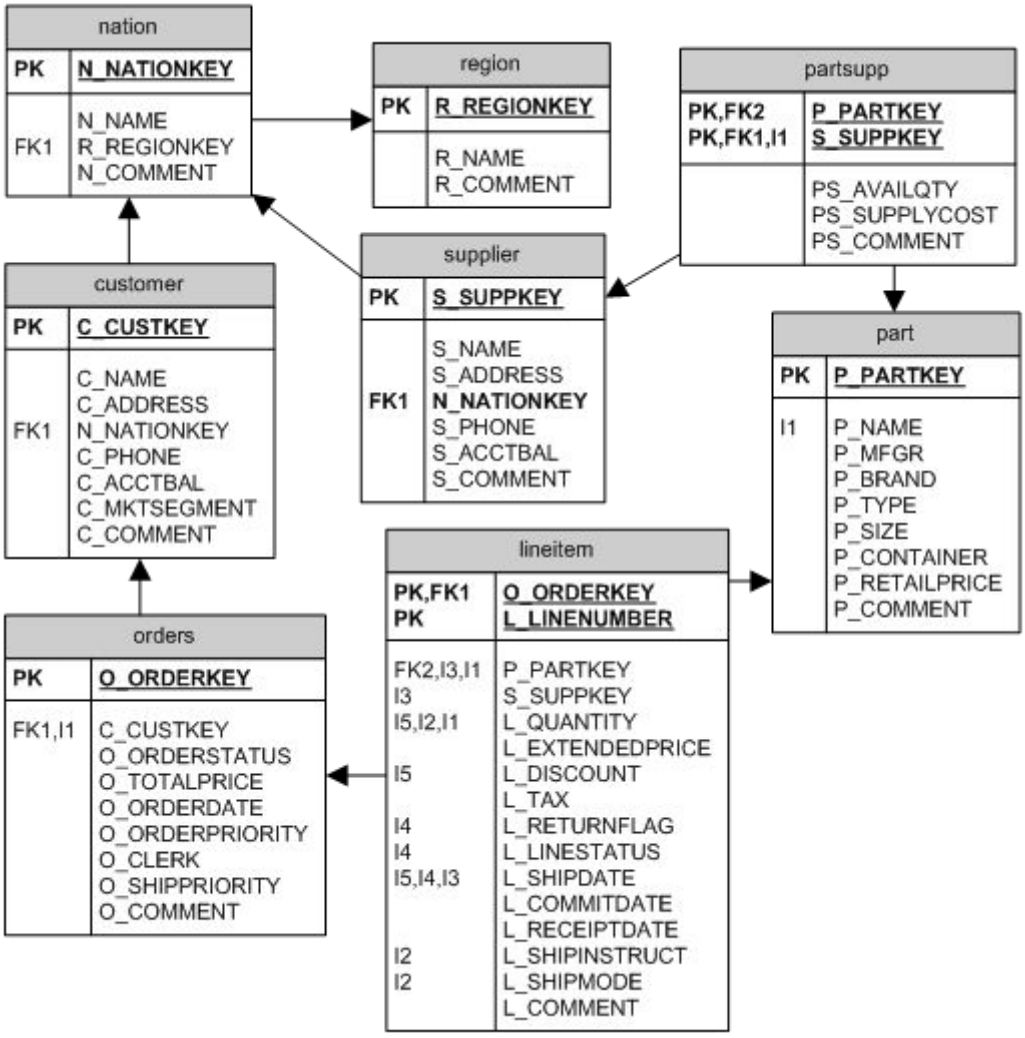
需要特别说明的是，分片表的查询需要通信开销，因此，对小表进行分片没有价值，反而降低查询速度。

以下以 TPCH-H 基准测试的数据为例，对 DosSQL 的分布式分片查询优化进行说明。

8.3.1 实例数据库和分片策略

为了便于展示优化特性，使用数据库标准测试 tpch 的数据库结构。tpch 数据库的结构

如图所示。（具体表的定义参见附录）。



在 tpch 测试脚本中，程序 dbgen 用于生成数据库数据。参数为数据库的大小，以 GB 为单位。在总数据量为 1GB 大小时，这个库中，nation 和 region 表都很小(几百 KB 左右)，分别为 25 条和 5 条记录。小的表为 supplier（1 万，11MB）。中等的表为 customer（15 万，43M）、part（20 万，58M）、partsupp（80 万，214M）。大表为 orders（150 万，247M）和 lineitem（600 万，2416M）。按照逻辑，orders 是订单表，lineitem 是订单物品的详细信息表。

当总数据量增加时，这个库中的 nation 和 region 表的大小保持不变，分别有记录 25 和 5 条。其他表的数据则成倍增长。因此当总数据量为 5GB 时，小的表为 supplier（5 万条，21MB）。中等的表为 customer（75 万条，128MB）、part（100 万，247MB）。大的表为 partsupp（400 万，1162MB）、orders（750 万，1162MB）和 lineitem（3000 万，11GB）。当表的数量增加到 10GB 数，则 lineitem 表达到 6000 万条记录，22GB 大小）。

数字有机体工作库的当前版本只支持水平分片，即表按照一定的条件将记录划分到多个分片中，且条件是预先定义好的。这和分区概念相同。同样的，分区上的限制在 DosSQL 上仍然存在。

本系统有一个特殊的功能，即支持诱导分片。表 R 和表 S 有同值域属性 R.a 和 S.b。

表 R 的分片规则为 p(R.a)，同样表 S 的分片规则也为 p(S.b)，即只是将分片规则中的 R.a 替换为 S.b。我们称表 R 和表 S 间存在水平诱导分片关系。显然，表 S 和表 R 的分片数相同。而且，对 R.a=S.b 的两条记录，必然放在相同下标的分片中。因此，可以认为表 R 的分片和表 S 的分片将一一对应。

管理员为每个表分别指定分片条件，并标识多个表间存在诱导分片关系。DosSQL 将存在诱导分片关系的多个表的相同序号的分片组织在一起，存储在相同的节点上，且组织在同一个存储库中（存储分片数据的库）。这使得可以利用诱导分片的特性优化查询。不存在诱导分片关系的分片可以独立分库存储，以便充分利用各台服务器的性能。

每个分片可以根据需要进行独立的复制，以保证数据的可靠性。

一个逻辑数据库的表可以部分是分片的，其他则是未分片的。对分片的表来说，数据不直接存储在该数据库中（实际上是存储在独立的分片数据存储库中），但是表的基本信息仍然存储在该数据库中，如表的结构、表的索引信息等。

对于数据库表进行分片需要考虑表查询的方式和表的结构。如 tpch 中的 lineitem 表和 order 表。这两个表都是大表，查询时主要以 orderkey 为条件，因此分片时应当考虑按照 orderkey 进行分片。而且 orderkey 也是 lineitem 表的外键，因此两者可以按照 orderkey 诱导分片。

选择分片模式时，首先选择范围模式。这种模式有利于实现分片剪除。所谓分片剪除即处理查询时，如果查询条件和分片的条件组合会导致结果总是为假，则该分片无需进行查询。这大大的减少了查询开销。反之，用 mod 这样的方式虽然也可能出现分片剪除，但是很少有这样的查询条件，不利于分片表查询的优化。

因此，对 tpch 的 order 和 lineitem 表，按照 orderkey 的范围进行分片，并按照 orderkey 做诱导分片是最合适的。

表分片的数量受几个方面因素的影响。首先是系统中服务器的数量。表分片的数量必须大于等于服务器数量。如果处理器都是多核的，且内存充足，甚至可以到达 2 倍以上。其次，理想地，每台服务器上存储的分片的总大小应该小于每台服务器上 innobase 缓存空间的大小。即 my.cnf 文件中 innodb_buffer_pool_size 的大小。这样，分片的数据可以完全载入到内存中，从而获得最理想的性能。最后，分片的分布应当按照服务器的性能尽量均衡分布。如果自动分布的不够好，可以通过添加和删除副本的方式调整位置。

最后一个问题是，多大的表应该分片？这是一个难以准确回答的问题。理论上，如果表的查询操作在单台服务器上都能顺利的运行（即查询速度都在几秒以下），则完全不需要分片，因为单台服务器就满足性能需求。如果要增加并发度，可以考虑用多副本的方式，而不是分片。只有当单台服务器的查询性能无法满足要求时，分片才有价值。应该认识到，分片也带来了通信和传输数据的开销，如果并行执行带来的好处比通信开销都小，则分片是不适合的。

8.3.2 查询优化的普遍规则

在书写查询语句时，以下规则是普遍有效的：

1）尽量多的增加选择条件。这样有助于减少查询时处理的记录数，从而提升速度。

2）尽量少的选择结果字段。对不需要的字段就不要出现在选择结果中。尽量避免使用星号通配符作为结果字段列表。

3）尽量避免使用 left、right 和没有连接条件的 join 操作。这样的操作可能产生非常庞大的结果集，因此效率非常低下。

4）根据常用的查询设计好表的主键和索引等，查询时尽量利用主键和索引。

8.3.3 单表查询

8.3.3.1 条件选择

对没有分片的表来说，从表中按照条件查询数据的处理方式有以下几种：

1）索引上的条件检索：如果表有索引，且查询条件只使用索引的字段，则处理方式为根据条件读取索引数据，只对满足条件的数据再到表中直接获取记录。如果选择结果也只在索引字段中，就连到表中读取数据都不需要了。如果查询条件涉及到主键，甚至可以利用唯一索引的特性快速读取到记录。

下面是利用索引查询的例子，它将使用 lineitem_k5 或者 lineitem_k6 进行检索。

```
select L from lineitem where L_SHIPDATE=’1997-02-01’;
```

注意，如果索引由多个字段构成，而条件只引用后面的某个字段，则无法使用索引。例如下面的例子中，虽然 L_DISCOUNT 出现在 lineitem_k6 索引中，但是条件没有引用第一个索引字段 L_SHIPDATE，因此无法使用索引。

```
select * from lineitem where L_DISCOUNT < 0.02;
```

改为如下的查询将使用索引：

```
select * from lineitem where L_DISCOUNT < 0.02 and L_SHIPDATE=’1997-02-01’;
```

下面的语句将只获取索引 lineitem_k6，因此速度更快。

```
select L_SHIPDATE, L_DISCOUNT from lineitem where L_SHIPDATE=’1997-02-01’;
```

下面的查询使用表的主键，查询速度更快：

```
select * from lineitem where L_ORDERKEY < 10 and    L_LINENUMBER < 100;
```

而下面这条语句只读取主索引，速度就再更快了。

```
select l_orderkey from lineitem where L_ORDERKEY < 10 and    L_LINENUMBER < 100;
```

2) 无法利用索引，则查询将遍历整个表，速度自然要低很多。例如：

```
select * from lineitem where L_RETURNFLAG = “F”;
```

因此，要尽量避免只使用非索引字段做查询条件。

总结起来，单表条件查询的优化要点是：

1）尽量减少获取的字段数，不需要的字段就不要获取；

2）如果有主键索引尽量利用主键索引，否则尽量利用唯一索引，以及其他索引。

3）尽量给出充足的条件，而不是放宽查询范围。

对分片表来说，上述的所有查询都将分散到各个分片上并行执行，因此查询速度将提升。

8.3.3.2 非排序和统计的 limit

如果单表的查询语句即没有使用统计函数，也没有使用排序，则这样的语句中若有limit(offset, num)限制，系统将尝试分别从各个分片节点上获取 2*(offset + num)条满足条件的数据。其中 offset 是开始获取位置，num 是要获取的记录数。如果 offset 不是很大，则并行查询的速度仍然是很快的。随着 offset 增大，传输的数据量将增加，查询的效率将降低。

即使是未分片的情况，随着 offset 增大，查询的速度也要降低。为了更好的解决这个问题，可能要使用下节描述的优化方式。

8.3.3.3 排序时的 limit

对没有分片的表来说，单表排序的处理速度受以下因素影响：

1）排序能否由索引完成

如果排序字段是某个索引的全部或者从开始起的字段（中间没有未使用的字段），则排序可以由该索引完成。例如：

select l_orderkey from lineitem where L_ORDERKEY > 10 order by L_ORDERKEY;

如果排序字段不是所有从第一个字段开始的连续字段也是不能使用索引的，例如：

select l_orderkey,l_linenumber from lineitem where L_ORDERKEY > 10 order by L_LINENUMBER;

2) 排序字段的个数和长度

如果排序无法用索引完成，则排序的字段数越多，长度越长，排序的速度就越慢。

3）参与排序的数量

要排序的数据量越大排序越慢。

因此，建议为经常需要排序的字段建立索引，否则排序速度难以提高。

在排序的结果上再做 limit 限制是常见的用法，例如下面的查询语句：

select * from lineitem order by L_ORDERKEY limit 10000, 10;

和分片表相比，这条语句在未分片的表上查询速度更快，原因是排序字段是表的主键，不分片时，查询可以直接读取表主键索引快速完成，因此速度很快。

在分片情况下，索引字段上的排序加 limit 会变慢。其原因是：由于数据分片存储了，从单个分片上取出的数据难以判断其位置，因此执行方式只能是从各个分片将数据取回，然后排序，再定位置。显然这比直接利用单表的索引慢很多。

对于这样的需求，建议的解决办法是：在库中增加一个只包含排序字段和表主键字段的表，用排序字段做该表的索引，且改变不做分片处理。查询时，先在这个表上定位出要获取的记录，然后再在分片表中按照条件查询需要记录。因此，针对上面的查询语句，先在本地创建如下的表：

```
CREATE TABLE `local_lineitem` (
  `L_ORDERKEY` int(11) NOT NULL,
  `L_LINENUMBER` int(11) NOT NULL,
```

PRIMARY KEY (`L_ORDERKEY`,`L_LINENUMBER`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

通常，排序字段的数据量相对整个表的数据量来说是很小的，例如这里原始表有 24GB 的数据，而 local_lineitem 表只有 1.74GB 的数据，显然可以在单个表中存储和使用，而且查询都在主键上，因此速度是很快的。

现在将 select * from lineitem order by L_ORDERKEY limit 10000, 10;语句转换为如下的语句：

select * from local_lineitem order by L_ORDERKEY limit 10000, 10; 这将获得 10 条 lineitem 的以主键为标志的记录，然后再用如下语句查询需要的数据：

select * from lineitem where (`L_ORDERKEY`, `L_LINENUMBER`) in ((10052,2),(10052,3), (10052,4) ….);这条语句以刚查出的 10 条记录的主键作为条件来查询，则速度将很快。

两条语句的总耗时仍然在 0.1 秒以下，完全可以满足要求。当然这可能比完全的本地表要慢些，而且还需要维护一个附加的表，但是当表非常庞大时，这仍然是一种有效的解决办法。

8.3.3.4 单表的统计处理

如果表未分片，单表统计处理的速度等于：检索数据的时间 + 排序时间（有分组时）+ 统计处理时间。

检索数据的时间开销和前面介绍的单表条件选择是相同的，排序时间开销和前面讲的单表排序类似。当然，如果不分组则不需要排序。统计处理时间和查询结果的记录数相关。查询结果记录越多的统计开销越大。

在分片时，max、min、sum 和带条件的 count 都将推送到各个分片上处理，从而提升处理速度。但是，如果查询结果中包含了除此之外的其他统计函数，则无法推送出去。查询速度提升仅来自并行获取数据。

这时，可将求平均值的函数转换为 sum 和 count 的组合，从而使得语句可以并行执行。例如：

select sum(`L_EXTENDEDPRICE`), count(*) from lineitem where `L_SHIPDATE` like “1998-02%”;

对于无法推送出去的函数，系统将并行地从各个节点上获取数据，然后在发起节点上完成分组统计处理。例如：

select std(`L_EXTENDEDPRICE`) from lineitem where `L_SHIPDATE` like “1998-02%”;

8.3.4 多表联合查询

8.3.4.1 表连接概述

DosSQL 支持 inner join，left join 和 right join。现在只将 inner join 推送到分片上进行，其他 join 操作不能并行执行，只能从各个分片上获取数据后再连接。因此，建议尽量避免使用 left join 或者 right join。

最常见的是两个表的连接操作。如果两个表中只有一个是分片表(即另一个是本地表)，则系统将把本地表的数据传送到各个分片节点上，分别连接后再返回结果。例如如下的查询：

select c_custkey, c_name, o_orderkey, o_orderdate from customer, orders where C_CUSTKEY = O_CUSTKEY and o_orderdate like “1992-02%”;

这时，建议使用 STRAIGHT_JOIN 来指定连接的顺序为“本地表，分片表”。这样，如果本地表上有过滤条件，则这些条件将先应用到本地表上，从而减少分布式并行执行的通信开销。而且，如前所述，建议尽量多的给出查询条件。

如果两个表都是分片表，且两个表间无法形成诱导分片连接，则系统将先获取一个表的数据，然后再把获取的数据推送到另一个分片表的分片节点上并行执行，例如：

select o_orderkey, o_orderdate, l_shipdate, l_linenumber from orders, lineitem where l_shipdate = o_orderdate and l_shipdate like “1992-02%”;

这个查询中的 orders 和 lineitem 虽然是诱导分片的，但查询的连接条件中没有分片字段相等的条件，即没有 l_orderkey = o_orderkey 的条件，因此查询不是诱导分片连接，无法一次分布式执行。这时，建议用 STRAIGHT_JOIN 将过滤后记录量更少的表放在前面。

如果参与连接的两个表是诱导分片的，且有两个表的分片字段相等的必要条件（即如果两个表的分片字段不相等则整个查询条件为假)，则连接是诱导分片连接，从而可以一次在各个分片上执行。例如：

select o_orderkey, o_orderdate, l_shipdate, l_linenumber from orders, lineitem where l_orderkey = o_orderkey and l_shipdate like “1992-02%”;

这个查询中，l_orderkey = o_orderkey 就是构成诱导连接的连接条件。显然，这减少了数据获取和传输的开销，因此性能更好。

如果查询涉及到的不仅是两个表，而是多个表，则系统首先要进行连接顺序优化，然后再考虑对其中的分片表进行分布式连接。例如：

select n_name, c_name, o_orderkey, l_linenumber from nation, customer, orders, lineitem where N_NATIONKEY = C_NATIONKEY and C_CUSTKEY = O_CUSTKEY and O_ORDERKEY = L_ORDERKEY and N_REGIONKEY = 1;

系统在优化连接顺序时，选择的执行策略是 orders、customer、nation、lineitem。即先读取 orders 的数据，然后依次和 customer、nation、lineitem 连接。

要注意的是：某些时候优化结果的可能并不是最优的，这时可以用 STRAIGHT_JOIN

选项指定连接顺序。例如：

select STRAIGHT_JOIN n_name, c_name, o_orderkey, l_linenumber from nation, customer, orders, lineitem where N_NATIONKEY = C_NATIONKEY and C_CUSTKEY = O_CUSTKEY and O_ORDERKEY = L_ORDERKEY and N_REGIONKEY = 1;

这样，系统不再尝试优化连接顺序，而是直接按照 from 子句中指定的先后顺序依次连接。在查询中存在诱导分片连接时，注意连接顺序尤其是重要的。要尽量形成诱导分片连接，这样更易于分布式处理。因此，上面的例子中，虽然两者的查询结果是相同的，但是后一条使用 STRAIGHT_JOIN 的语句在分片环境上查询性能更优。

总结起来，在普遍规则的基础上，多表联合查询还需要注意的是：可以先用 desc 或者 explain 获得多表连接的处理顺序，然后根据各个表的数据量用 STRAIGHT_JOIN 指定连接顺序。其基本思路是：

（1）将条件过滤后数据量小的表放在前面，大的表放在后面；

（2）尽量将诱导分片的表连续放置。

8.3.4.2 诱导分片和诱导分片连接

如前所述，本系统支持诱导分片。表 R 和表 S 有同值域属性 R.a 和 S.b。表 R 的分片规则为 p(R.a)，同样表 S 的分片规则也为 p(S.b)，即只是将分片规则中的 R.a 替换为 S.b。我们称表 R 和表 S 间存在水平诱导分片关系。显然，表 S 和表 R 的分片数相同。而且，对 R.a=S.b 的两条记录，必然放在相同下标的分片中。因此，可以认为表 R 的分片和表 S 的分片将一一对应。

诱导分片的定义方式是：partition by range (L_ORDERKEY) REFERENCES orders。具体参见附录中的表定义。

诱导分片的最大意义是诱导分片连接的处理。如实例数据库中 orders 和 lineitem 是诱导分片，o_orderkey 和 l_orderkey 分别是两个表的分片字段。这样，对下面的连接来说，由于存在两个表的分片字段对应相等的条件，则连接操作可以在各个分片节点上并行执行，从而提升连接速度。如果不是诱导分片的，则必须先获取一个表的数据，然后推送到另一个表的分片节点上并行连接，其执行开销更大。

select o_orderkey, o_orderdate, l_shipdate, l_linenumber from orders, lineitem where l_orderkey = o_orderkey and l_shipdate like “1992-02%”;

在关系数据库中，根据外键关系联合查询两个表的数据是很常见的，因此按照表间的外键依赖关系进行诱导分片是合理的。例如 lineitem 表通过 L_ORDERKEY 字段外键依赖为 ORDER 表的 O_ORDERKEY 字段，这就是诱导分片的基础。

8.3.4.3 多表连接时的统计处理

如果多表连接的选择结果中有统计函数，则统计处理能否推送到各个节点上分布式执行依赖于以下条件（必须满足所有条件）：

1）多表连接操作的最后一个表是分片表，且分布式并行执行更优。

2）统计函数只包括 max、min、sum、count，且没有 distinct 要求。

如果满足上述两个条件，则统计处理将随连接的分布式并行处理一并完成。例如下面的查询：

select c_custkey, c_name, o_orderkey, max(O_TOTALPRICE) from customer, orders where C_CUSTKEY = O_CUSTKEY group by C_CUSTKEY order by C_CUSTKEY;

select c_name, o_orderkey, max(L_TAX) from customer, orders, lineitem where C_CUSTKEY = O_CUSTKEY and L_ORDERKEY = O_ORDERKEY group by C_CUSTKEY order by C_CUSTKEY;

即使统计函数无法分布式并行执行，其中的分片表连接仍然可以分布式并行执行。这也有助于提高查询速度。

8.3.4.4 多表连接时 limit 的处理

在满足下面条件时，limit 将推送到各个分片节点上处理：

1）查询没有使用半连接优化的嵌套子查询；

2）查询没有聚合函数也无排序字段；

3）分布式并行连接是查询的最后一个连接。

如果满足上述条件，则 limit 将在各个分片节点上应用。否则，仍然需要获得完整的连接结果后再继续处理。

相对于未分片的表来说，主要是嵌套子查询和多个表的联合查询时可能有优势。其原因是它可以尽早的根据 limit 限制结束处理。

8.3.5 直接删除和直接修改

当修改或者删除分片表的数据时，通常的内部流程是先查询出所有满足条件的记录，然后在对每条记录进行删除或者修改。对单机数据库来说，这并不是太大的问题。但对分布式的分片数据库来说，则将产生巨大的开销。

一种简化的内部流程是直接到需要的分区上删除或者修改数据，不再查询那些数据满足条件。这种实现方式显然比前一种快很多。

本数据库系统支持直接修改和删除分区表的记录。但要开启这项功能，则必须将配置文件(my.cnf)中，binlog_format 参数设置为 mixed，而且会话的该参数也为 mixed 类型。当采用 ROW 类型时，不能采用直接修改或者删除记录的方式。

要注意的是：当 binlog_format 参数为 mixed 时，如果表有触发器或者修改数据的存储过程，或者其他的某些隐式的修改实现时，可能导致表的多个副本间数据不一致。因为当 binlog_format 参数为 mixed 时，某些语句记录的 binlog 日志并不能完全反应语句的影响，而副本间的同步依赖于 binlog 日志的正确性。因此，这时应当将 binlog_format 参数设置为 ROW 类型。

如果您确定你的语句不会有上述情况，而且要操作的是分区表，你可以将 binlog_format 参数设置为 mixed，从而启用该项功能。

8.3.6 糟糕的查询语句

在某些情况下，分片后查询的性能反而会降低。下面分别描述这些情况。

8.3.6.1 关联的嵌套子查询

当使用嵌套的子查询时，如果子查询和父查询没有使用相同的表，则两者是没有关联的。这时通常对子查询进行物化处理，然后再用子查询的结果表处理父查询。例如下面的查询语句：

select L_ORDERKEY, L_SHIPDATE from lineitem where L_SHIPDATE in (select O_ORDERDATE from orders where O_ORDERKEY < 10);

对这条语句，系统将先执行子查询并将结果写入一个临时表中，然后用临时表和 lineitem 表连接。对这样的嵌套查询，系统可以利用分布式并行获取和分布式并行连接加快执行速度。

如果嵌套子查询无法进行物化，则子查询可能转换为连接或者半连接来处理。例如下面的语句将转换为连接来处理：

select L_ORDERKEY, L_SHIPDATE from lineitem where L_SUPPKEY in (select S_SUPPKEY from supplier where S_NATIONKEY = 1);

这条查询中，L_SUPPKEY 字段是 lineitem 表外键依赖于 supplier 表的字段，而且是 in 子查询，因此转换为连接来处理。这个转换时系统的查询优化器自动完成的。要注意的是，对 exist 子查询则不会做这样的转换。因此，如何可能，建议将 exist 子查询手动改写为 in 子查询。

如果 IN 子查询的前面字段列表部署外键依赖字段，或者是反过来的情况，则会使用到特殊的优化策略，则是分片表的查询性能将急剧降低。例如如下的语句：

select s_name from supplier where S_SUPPKEY in (select L_SUPPKEY from lineitem where L_ORDERKEY < 10);

SET SESSION optimizer_switch='semijoin=off';

最糟糕的情况是：对父查询的每条满足条件的记录（放开子查询条件后），都需要执行子查询的情况。这时，无论表分不分片，查询的性能都将非常低。因此，建议尽量避免使用这样的查询。建议通过编程的方式将其转换为更多的独立查询来实现。

8.3.6.2 left(right)join

对大表来说，由于 left(right) join 将产生很大的结果集，一般都是不建议采用的。

8.4 常见问题和注意事项

1）表字符集的设置

通常，每个数据库有一个默认的字符集，当创建表未指定字符集时，将使用数据库默认的字符集。本系统支持修改数据库的默认字符集。

每一个表也有字符集。对分片表来说，这个字符集不仅保存在逻辑表的信息中。最重要的是要作用到分片数据库的相应表上。但是，本系统不支持修改表的字符集。因此，在创建表时一定要谨慎的选择表的字符集，并在创建时明确指定。如果实在要修改表的字符集，则需要同时修改逻辑表和所有分区数据库中对应表的字符集，并且要一致。

2）执行语句报 28 号错误

错误“Get error 28 from storage engine”。出现这个错误的大多数情况是语句涉及大数据集的传递和保存，其数据量超过了入口节点内存限制，导致操作被终止。例如，一个表有 6000 万条数据，执行下面的语句时就可能出现这样的错误。

SELECT * FROM `lineitem_copy` LIMIT 59998000,1000。

对于这样的情况，只能将查询分解后分别执行。

3）分区表复制及使用相同的

9 抗毁容灾

DosSQL 的特殊结构使其天生就具备容灾抗毁的特性。它主要表现在系统抗毁和数据容灾这个方面。单机版的 MySQL 和 Oracle 等常用的数据库系统不具备抗毁的特性。数据库服务器故障（如磁盘损坏、断电、停机等）会导致数据无法访问或者丢失。而 DosSQL 系统中不存在固定的中心节点，由大量服务器构成，它们之间通过相互协作组织系统，数据以数据库为单位进行多副本存储。由于无中心节点，因此系统不会因某些服务器故障而无法运行。每个数据库都有多个副本，即使某些服务器出现故障，系统中仍然有可用的副本，数据不会丢失。

除了以上提到的系统组织结构和多副本存储机制外。DosSQL 还实现了故障自行诊断和系统自动康复。当发现系统内的某台服务器故障后，系统将把故障的服务器排除到系统之外，防止用户访问出错；当诊断发现原来故障的服务器恢复正常后，系统会自动把正常的服务器融合到现有的系统之中；当诊断发现数据库的副本由于某种原因（包括服务器故障与恢复、命令修改副本数量属性）需要变动时，系统会自动调整数据库副本的位置和数量。

基于 DosSQL 的抗毁容灾特性，开发人员能够快速地开发出各种具备“业务容灾”的应用程序。

9.1 系统故障后自动重构

传统的数据库系统可能出现一些难以解决的故障，如服务器死亡和网络分隔（分片）之后，系统就再难以正常工作。故障的情况大概有以下几种：

1) 系统内的数据库服务器死亡（如服务器被断电、服务器被关机）；
2) 系统内的数据库服务器网络相互之间分离（如服务器网线故障、连接各个服务器的交换机或路由器故障）；
3) 系统内的数据库服务器既有死亡又有分离的情况。

数字有机体系统号称“打不烂系统”。在出现以上的故障后，DosSQL 系统会自动进行重构，使破碎的系统恢复正常，继续为用户提供服务。基于这个功能，DosSQL 系统在安装时能很好地支持在线扩展数据库服务器，而不影响正在进行的工作和业务。

需要注意的是：网络分割发生后，原有的数字有机体系统被分成了多个独立的子系统，每个子系统的数据都有可能是用户需要的。但是，系统自动的合并过程却可能和用户的需求是不一致的。为了消除这种不一致，用户可以保留需要的分片，先将不需要的分片关机，联通网络后，再开启关闭了的服务器，以此来加入系统。

9.2 本地副本和异地副本

一个数据库的副本可以分布在不同站内，也可以只放置在一个站内。数字有机体工作库系统中的站可以是地理上相隔很远的数据中心。这样，如果一个数据库的副本放置在不同的站内，例如同时放置在北京、上海和成都，则它们之间就形成了异地容灾。这种部署结构有利于提升系统的可靠性，甚至可以抵御严重的自然灾害和恶意打击。数据库的访问性能主要受到网络延迟、传输带宽和服务器本身性能限制。因此，即使是相隔万里的站点，如果有足够好的网络，仍然可以获得良好的性能。

当然，许多时候并不需要数据库放置在不同的站点。这时，可以限制数据库只能在本站放置，即不允许站外放置。在一个有限的地域内要获得高网络速度和低传输延迟并不困难，一台性能优良的交换机即可解决问题。因此，在一个站内放置数据库副本是最易获得良好性能的情况。

应用可以根据自己的需要指定每个数据库的副本放置位置，从而在可靠性和性能间获得平衡。

9.3 副本故障后自动恢复

DosSQL 系统同时支持多副本和单副本两种情况。为了保证数据的可靠性、可用性，以及安全性，本公司强烈建议用户采用多副本机制。

DosSQL 系统能够自动重构。重新加入系统的服务器上的数据库副本将以恢复的形式来更新数据内容。这个过程由 DosSQL 自行完成，无需人为干涉。它将智能地检测数据库的数据组成，根据数据的新旧情况，有选择地进行日志恢复和完全备份恢复。

如果系统中已经有数据库的足够副本，则故障节点恢复后，其上的数据库副本可能丢弃，即不再恢复起来。

9.4 副本自动管理

DosSQL 系统自行完成各个数据库的副本管理。管理的内容主要包括副本的数量和副本的位置。

数据库副本的数量是由该数据库的多项属性和系统的当前状况共同确定的，以下是判断的优先级：

1) 是否有正常状态的数据库副本；
2) 是否允许将数据库副本放置到本站之外的站；
3) 数据库副本允许的最小副本数和最大数；
4) 系统当前的站及服务器情况。

如果没有正常状态的数据库副本，副本的数量将维持不变。如果已有的副本同时在多

个站内，但数据库的属性不允许放置副本到外站，副本管理将先删除站外的副本，然后再根据副本允许的数量范畴在本站内自动新建副本。副本的数量将满足最小副本数和最大副本数限制。如果副本数量小于最小副本数，系统将根据各台服务器的空余存储容量和负载选择服务器创建新副本。如果副本的数量大于了最大副本数限制，副本管理就将删除多余的副本。删除的原则是尽量删除性能较差服务器上的副本。此外，系统还会统计数据库连接的请求状况，并根据请求热度（即访问数量）来智能迁移副本，使访问者就近访问，从而提升访问数据的速度。

影响数据库副本数量和位置的属性如下表所示：

数据库副本管理属性	描述
Admin_section	数据库新建时所在的区域（目前区域未使用）
Admin_station	数据库新建时所在的站点，副本放置时优先满足此站
Admin_node	数据库新建时所在的节点，副本放置时优先满足此节点
Out_section	是否允许副本放置在本区域外（目前区域未使用）
Out_station	是否允许副本放置在本站外
Min_sum_dup	数据库副本数量限制的最小数量（下限）
Max_sum_dup	数据库副本数量限制的最大数量（上限）

数据库的属性（通常是 out_station、min_sum_dup、max_sum_dup）是可以由“ALTER DATABASE”指令来修改的（有些属性是不能被修改的）。用户可以通过修改数据库属性来控制数据库的副本。此外，用户也可以通过使用“ADD DATABASE”和“DROP DATABASE”来人为干预副本位置的分布。

修改数据库副本位置相关的属性如下：

```
DosSQL$ ALTER DATABASE test out_station=0 min_sum_dup=3 max_sum_dup=5;
```

注意：同时修改多个属性时，各个属性的修改需用空格分割，不是逗号。等号前后不能有空格或者其它符号。

修改数据库属性后，系统需用一定的时间来响应属性改变。例如需要重新计算副本的数量，调整副本的位置等。因此，需要等待系统完成处理。你可以用副本查询指令查看当前的副本放置状况。

要提醒的是，改变某些属性可能带来额外的系统开销，从而影响系统的当前服务能力。比如，增大数据库的最小副本数可能要求系统创建新的副本。如果数据库内已经有大量数据，则创建新副本必然带来较大的数据复制开销，从而影响系统当前的服务。因此，建议修改属性这样的操作尽量在系统空闲的时候进行。

10 数据库访问

DosSQL 中的数据库一般有多个副本存在。每个副本位于不同的节点上。有的节点可能很忙碌，有的节点可能很空闲。为了最大地发挥各个节点的作用，每次连接数据库时都需要连接到最空闲的节点，使各个节点都忙碌起来，从而做到并行计算，并充分利用资源。

数据库连接将消耗内存等资源，因此数据库服务器支持的最大连接数量是有限制的。假设一台数据库服务器的最大连接限制数量为 COUNT_MAX，那么 MySQL 等单机数据库系统的最大连接限制就是 COUNT_MAX。在 DosSQL 系统中，假设有 N 个数据库副本，那么 DosSQL 系统的最大连接限制就是 COUNT_MAX 的 N 倍。

DosSQL 系统中的数据库存在多个副本，对于读操作，任意一个副本就能满足用户的需求；但是对于写操作，为了保持数据的实时一致性，各个副本需要协同工作，不同的副本处理能力有所差异。因此，快速的写操作需要连接一个适当的副本。

由前面的介绍可知，数据库的连接类型可分为以读为主和以写为主两类。由于分片表的后台是大量的数据库，因此，分片表的操作也可以分为以读为主和以写为主两类。在开发应用程序时，建立任意类型的数据库连接，都能满足用户的需求，实现各种读写操作，但是为了更好地体现 DosSQL 的性能，本手册推荐开发人员同时使用“以读为主”和“以写为主”这两类连接，尽可能地使用“以读为主”的连接进行查询操作，而尽可能地使用“以写为主”的连接进行更新和事务操作。

10.1建立以写为主的连接

默认情况下，建立的连接是以读为主的连接。建立以写为主的连接，可以大幅度提高数据的更新。要建立这种连接，需设置连接选项“MYSQL_SERVER_FOR_UPDATE”和“MYSQL_SERVER_REDIRECT”。其中，“MYSQL_SERVER_REDIRECT”是默认被设置的，以下为编程使用示例。

```
int for_update = 1;
MYSQL *conn = mysql_init(NULL);
mysql_options(conn, MYSQL_SERVER_FOR_UPDATE, (char *) &for_update);
mysql_real_connect(conn, host, user, pswd, db, 0, NULL, 0);
```

10.2调度延迟和负载均衡

调度策略的目标是尽量把空闲的节点利用起来，使得忙的节点不是太忙，闲的节点不是太闲，最终达到负载均衡。因此这里需要考虑各台服务器的负载情况。服务器的负载处于一种波动的状态，瞬时值忽大忽小。某时刻的负载难以代表一台服务器在未来一段时间内的负载。DosSQL 累计统计一定时间段的负载，并计算该时间段内的平均负载值。负载波动较小，也相对准确，能代表服务器的负载状况。但是不同节点之间的负载信息的相互

传达需要充足的传输时间，因此配置文件“/etc/dossql.cnf”中的数据库连接调度延迟时间参数“schedule_delay_time”对调度的负载均衡影响至关重要，参数在配置文件中的情况如下所示。

```
# Wait time, when connection coming too many and fast, will sleep, unit is 'ms'.
schedule_delay_time=400
#调度延迟时间: 多个连接同时到达时，前后两个连接之间的时间间隔。本参数值越大，调度的速度就越慢，但是负载就越均衡；本参数值越小，调度的速度就越快，但是负载就可能不均衡。
```

参数“schedule_delay_time”单位为毫秒，配置结束后系统需要重新启动才能生效。该参数的值越小，数据库的连接速度就越快，负载均衡效果就越差；该参数的值越大，数据库的连接速度就越慢，负载均衡效果就越好。用户应该根据不同的需求来确定该值。

10.3智能连接

DosSQL 和 MySQL 等单机数据库是不同的。单是从副本的角度出发，MySQL 是单服务器单副本的。用户在连接某个数据库的时候必须指明访问的数据库所在的服务器地址。如果地址错误就无法访问。因此，MySQL 等单机数据库系统是单一访问入口的。DosSQL 是由多台服务器构成的，各个服务器上要么有某个数据库的副本，要么没有副本。无论有无，用户在连接数据库的时候只要指明了 DosSQL 系统中的任何一台服务器的地址，数据库都能正常地连接。因此，DosSQL 系统是多访问入口的。在数据库服务端的应用中，使用 DosSQL 客户端连接数据库时，如果用户指明的数据库访问地址是错误的，连接库将智能地查找一个正确的地址进行连接而不报错。

结合前面提到的“多访问入口”优势，DosSQL 能够避免单机数据库系统连接时的单点故障，即使故障了一台数据库服务器，其它数据库服务器还能够继续被使用。

其中，智能连接是采用连接选项“MYSQL_SERVER_REDIRECT”来实现，新建的数据库连接已经被默认使用。采用 MySQL 的原生开发库不具备这种能力。

使用 DosSQL 监视器进行智能连接的时候，需要使用数据库的名称，智能连接使用数据库的名称寻找访问资源，调度并连接一个负载相对较轻的 DosSQL 服务器。没有使用数据库时，DosSQL 监视器连接的地址是 localhost，已经建立好的连接不会再转移。使用 DosSQL 监视器进行智能连接的方式如下：

```
root@server54:/home/dos# DosSQL -uroot -p"" test
Enter password:
Welcome to the DosSQL monitor.  Commands end with ; or \g.
Copyright (c) 2000, 2018, TianXinYue.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
DosSQL[test]$
```

10.4透明访问

DosSQL 支持标准的 SQL 语法，以及 MySQL 的扩展语法。但是，DosSQL 系统由分布在网络中的多台数据库服务器组成，相互之间协同工作，数据的组织采用多副本机制。因此，它是和单机或者集群 MySQL 数据库不同的。不过，DosSQL 仍然采用 MySQL 的开发接口库和 SQL 语句，隐藏底层的分布式和多副本等细节，让用户在使用 DosSQL 时就像使用 MySQL 单机数据库系统一样。这种对用户完全透明的访问机制，极大的方便了用户和开发人员。他们无需重头学习一种新的数据操作语言，也无需从头学习一个新系统的使用。同时，这种透明性也使得原有的各种 MySQL 应用可以直接运行在数字有机体工作库上，从而避免了移植需要的大量开销。

10.5分片表数据的读与写

对于普通的数据表而言，它仅仅隶属于一个数据库，因此在连接的时候没有任何的争议。但是，对于分片的数据表而言，一个分片表的逻辑表虽然只属于某一个数据库，但它的分片数据库却存在多个，连接不同的分片数据库会对表的读与写产生影响。针对不同的应用，影响的大小有所不同：如果是小数据的查询，或者是少量数据的写入，这种影响几乎可以忽略；如果是针对大量数据的查询或者大量数据的写入，这种影响就会很大。

为了解决这个问题，DosSQL 设计了一个系统变量，使用户可以通过设置系统变量来控制，如果用户是以写性能为主，读性能次之，则设置变量 dosdb_read_and_write_weight_level 为 1；如果用户是以读性能为主，写性能次之，则设置变量 dosdb_read_and_write_weight_level 为 0。

变量 db_read_and_write_weight_level 被设置为 0 后，分片数据库的连接将会尽量被分散，充分利用每台服务器的计算能力；变量 db_read_and_write_weight_level 被设置为 1 后，分片数据库的连接将会尽量采用有利于写数据的方式。

特别提示：变量 db_read_and_write_weight_level 被修改后，分片数据库的原有连接可能会重新调整，因此，在事务中，严禁修改该变量。

11 安全

数字有机体工作库的安全特征主要体现在统一权限管理、安全通信、主机身份鉴别和数据库副本机制四个方面。其中安全通信、主机身份鉴别都需要 SSL 安全证书。证书需要两套，一套用于 DosSQL 服务器之间的通信；一套用于客户端到 DosSQL 服务器之间的通信。证书的制作方式请参阅 11.5 章节。

11.1统一权限管理

DosSQL 的权限机制采用访问控制列表（ACL）。能满足对安全度要求较高的用户。数字有机体工作库中有多台服务器，每台服务器都是它的一个访问入口，因此管理好每个数据库服务器的权限至关重要。

数字有机体工作库采用统一权限管理机制，使每台数据库服务器上的权限都保持一致。数据库管理员只需在系统中的任意一台服务器上修改数据库的访问权限即可在整个系统中产生效果。无需登录每台服务器分别设置权限。这极大的减轻了权限管理的复杂度，并且保证各台服务器上的权限策略是一致的，不会因为人为疏漏而在某台服务器上留下访问后门。

11.2安全通信

DosSQL 的通信由两个方面组成：一方面是 DosSQL 系统之间的各个数据库服务器之间的通信，另一方面是 DosSQL 的客户端和服务端之间的通信。

数据库服务器之间的通信采用的是 SSL 的改进版本。这时，不仅每次通信的密钥是随机产生的，而且每次通信的加密算法也是随机选取的，从而避免了单一加密算法容易破解的问题。服务器间的通信加密对系统的最终用户是透明的。当然，采用加密通信必然产生额外的开销。不过，在今天这个网络监听无处不在的时代，这些额外开销是值得的。

而 DosSQL 的客户端和服务端之间的通信则继续沿用 MySQL 的方案，这样可以兼容 MySQL 的客户端开发库，从而兼容 MySQL 系统的应用程序。

11.3主机身份鉴别

数字有机体系统由大量服务器组成，相互之间通过高速的网络互联互通。各台服务器相互协调，通过资源共享，并行计算能带来优于单机的巨大好处。但是，组成数字有机体系统的各台服务器之间如果缺乏有效的“身份证”，就会给非法人员（黑客）提供入侵的机会。采用安全通信，系统阻断了攻击者通过网络截取数据的能力。但是，可能面临攻击者采用一台非法的主机冒充系统中的合法主机，从而从其它主机上获得信息，或者非法使用其它主机的资源。为了有效应对这种攻击，DosSQL 系统给每台合法的数据库服务器颁发

了一个“身份证”证书。没有“身份证”的非法服务器将无法和系统中的任何服务器通信。实际上，系统中的服务器将拒绝任何没有合法身份证书的节点的通信请求。

服务器的合法“身份证”是由用户颁发的经过数字签名后的可以有效防止篡改的数字证书。每台服务器的证书和服务器的 IP 地址绑定，即使非法人员获取到有效的证书也无法入侵 DosSQL 系统。

证书的配置在配置文件“/etc/dossql.cnf”中的“ssl”处，配置参数部分如下：

```
# The follow is SSL related arguments, default is not use ssl.
use_ssl=1（是否使用主机间 SSL 安全通信，使用 SSL 通信将同时具备主机识别功能）
ssl_ca=/etc/do/cacert.pem（SSL 的根证书）
ssl_cert=/etc/do/server-cert.pem（SSL 的公钥证书）
ssl_key=/etc/do/server-key.pem（SSL 的私钥证书）
ssl_password=123456（SSL 证书密码）
```

11.4数据库副本机制

前面提到的安全机制可以有效地阻止非法人员的“软”攻击，但是这还不够。常见的数据库方案中，数据库服务器磁盘损坏将可能造成数据的丢失，即使采用了每天备份的机制，也只能挽留大部分的数据。对于这种“硬”伤，DosSQL 能有效地解决。它采用多副本机制，某个副本所在的服务器故障之后，还有其它副本可用，从而保证了磁盘损坏的情况下仍然有正确数据可以使用。

11.5SSL 证书制作

数字有机体工作库使用的数字证书是使用 OpenSSL 制作的证书。该证书内不仅封装了颁发证书机构的信息，还包含了证书持有者的 IP。在鉴别证书时，不仅鉴别对方的证书真伪，还鉴别对方的 IP 和证书持有者 IP 是否相同，从而防止攻击者窃取证书后假冒系统主机。因此，用户需要妥善保管每台主机的证书。如果没有使用数字有机体发行版，找不到该证书制作工具，请与本公司联系。

为方便 SSL 证书的制作，本公司给出了证书制作的脚本，该脚本被安装在数字有机体发行版中的“/usr/bin/create_ca/”目录。制作方式是先进入“/usr/bin/create_ca/”目录，然后执行“./create_ca”脚本开始证书制作。制作步骤如下（没有截图说明的部分可以采用缺省配置）：

第一步、输入证书的 IP 地址；

第二步、输入证书的有效天数；

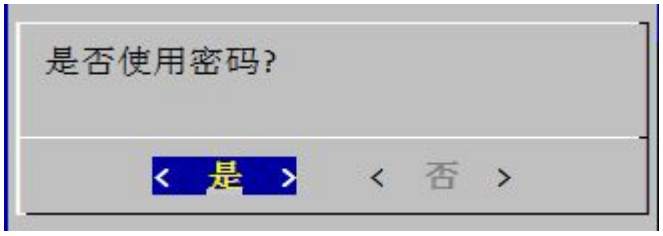
第三步、确定是否制作机构证书。只有第一次制作时才选择“是”，只有具有相同机构证书的服务器才能正确连接。如果选择“是”，您应该记住输入的密码 passwod_a；

第四步、开始制作主机证书，首先生成主机的私钥，并提示输入封装私钥的密码，如下所示。在输入并确认密码 passwod_a 后，即开始输入主机的信息，这些信息条目和机构证书的相同，只是有一个可选的挑战密码和公司名称，建议直接回车略过。

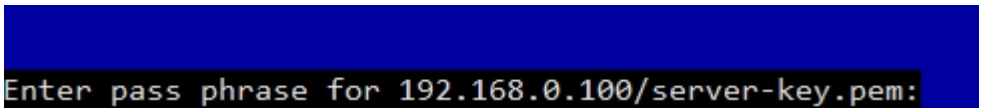
```
开始制作key.pem
Generating a 1024 bit RSA private key
.....................................................................
..............++++++
writing new private key to '192.168.0.100/server-key.pem'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
```

第五步、选择是否需要 SSL 证书的密码，制作 DosSQL 服务端到客户端的证书是不使用密码的，需要选择“否”；制作 DosSQL 服务器之间通信的证书是需要密码的，需要选择

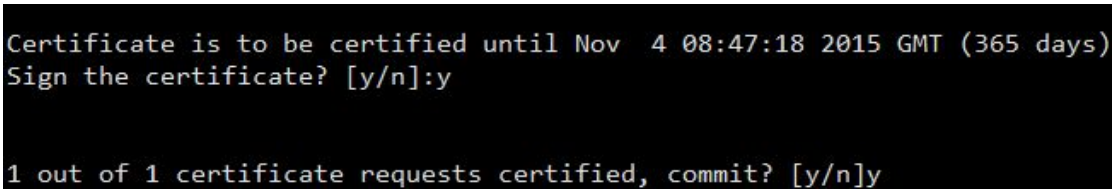
“是”；



第六步、输入 SSL 证书的密码 password_b，这个密码将用于配置文件“/etc/dossql.cnf”中的“ssl_password”参数；



第七步、选择签名和提交，完成证书的制作。



上述证书制作过程中，要注意两点：

- 如果重新为一台主机制作证书，则需要先删除其原来的证书目录，并且从文件 CA/index.txt 中删除原来的证书记录，否则将无法重新生成证书。
- 机构私钥证书的密码不能忘记，否则将无法制作证书。同样，主机私钥证书的密码也不能忘记，否则无法配置下面的参数。

12 DosSQL 高级特性及应用

DosSQL 工作库系统实现了数据库的多种高级特性（如多副本机制、数据的强一致性、数据的弱一致性、表分区、表分片、安全通信、主机身份鉴别）。它支持各种高级特性的组合，也支持单一副本的形式（即普通的数据库模式）。因此，它属于一款通用的数据库系统。

通常情况下的数据库应用能够很好地在 DosSQL 上得到支持。此外，它的高级特性能有效地帮助企业及个人解决一系列的问题，从而满足目前市场上的各种应用。

DosSQL 能在多种情况之下使用，用来解决各种不同的问题。本章节用于向用户提供一些常用的注意事项，并推荐说明一些应用的解决方案。

12.1普通数据库模式

对于一般的数据库应用，用户没有什么特殊的需求，只希望像使用单机的 Oracle、SQL-server 或者 DB2 那样使用 DosSQL，用于存取数据。DosSQL 能够很好地满足用户的需求，用户只需要在新建数据的时候指明副本的最大、最小数量限制为 1，如以下的 SQL：

CREATE DATABASE test MAX_SUM_DUP=1 MIN_SUM_DUP=1

按照以上的方式新建的数据库采用的单副本的机制，使用的时候和普通的单机数据库系统相同。即使这样，使用 DosSQL 系统能够有效地将多台服务器整合成为一套系统，这种整合服务器资源的功能是其它数据库系统所不具备的功能。

12.2DosSQL 的高级特性

DosSQL 工作库具有多种高级特性，适用于各种特殊需求的性用。组合使用各种高级特性能够有效地解决企业及其个人的应用需求。以下介绍它的多种特性和优缺点。

高级特性	优点	缺点
多副本机制	1) 提升数据的安全性； 2) 提升数据的可用性； 3) 提升数据的并行查询性能； 4) 使系统负载均衡成为可能； 5) 保证数据访问的不间断性。	降低数据的写和更新性能。
强一致性	1) 提升多副本机制下的数据可用性；	多副本的情况下降低数据的写和更新性能。

	2) 提供多副本之间的实时备份机制。	
弱一致性	1) 提升多副本之间数据的写和更新用性； 2) 提供多副本之间的批量备份机制。	多副本之间数据可能在短时间内不一致，降低了 slave 副本数据的可用性。
表分区	1) 能够轻易删除不再使用的表分区数据； 2) 能够方便地新增加一个数据表分区； 3) 对于带有 WHERE 子语句的查询，如果数据只存储在一个或几个分区，查询优化能够只查询那些有数据的分区，避免全表扫描，从而提高查询性能； 4) 并行执行 SUM()、MAX()等统计（聚集）函数； 5) 把分区的子表分别存储到不同磁盘，可以有效提高磁盘的读写性能。	1) 不支持触发器； 2) 不支持存储过程； 3) 不支持 SQL 函数； 4) 不支持用户自定义函数 UDF； 5) 不支持插件 plugin； 6) 不支持用户变量。
表分片	1) 能够轻易删除不再使用的表分区数据； 2) 能够方便地新增加一个数据表分区； 3) 对于带有 WHERE 子语句的查询，如果数据只存储在一个或几个分区，查询优化能够只查询那些有数据的分区，避免全表扫描，从而提高查询性能； 4) 并行执行 SUM()、MAX()等统计（聚集）函数； 5) 并行执行单表的 SQL 查询； 6) 把分区的子表分别存储	1) 不支持触发器； 2) 不支持存储过程； 3) 不支持 SQL 函数； 4) 不支持用户自定义函数 UDF； 5) 不支持插件 plugin； 6) 不支持用户变量。

	到不同磁盘和服务器，可以有效提高磁盘的读写性能； 7) 支持直接对分片子表的操作，能提高数据访问的性能和并行度。	
安全通信、主机身份鉴别	1) 提升数据在网络传输中安全； 2) 提升服务的安全，防止黑客的非法入侵。	降低通信的性能，从而降低事务的性能

12.2.1 用于实时备份

推荐使用多副本机制，并且使用强一致性，能够实现实时备份的功能。副本的数量为2时，实现传统的双机实时热备份功能；副本的数量大于2时能实现多机实时热备份。

这种机制保证副本之间的数据无时无刻都保持一致，无论从哪个副本获取数据都是一致的。

12.2.2 用于批量备份

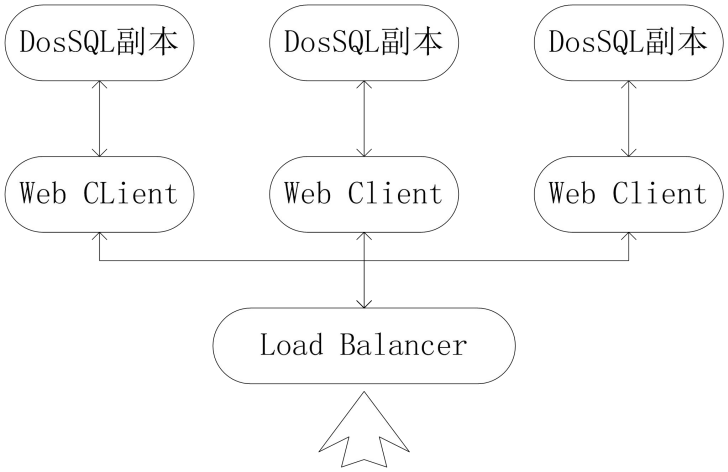
推荐使用多副本机制，并且使用弱一致性，能够实现批量备份的功能。副本的数量为2时，实现传统的双机批量热备份功能；副本的数量大于2时能实现多机批量热备份。

这种备份机制几乎不影响数据库副本的性能，能让副本的写速度接近单副本的性能。是备份系统的最佳选择。另外，对于一些应用，实时查询的要求没有多高，比如“论坛网站”、“视频网站”等，使用这种方法，使系统整体对外的可供查询的副本数量增多，能大弧度地提高系统的整体查询性能。

12.2.3 以查询数据为主的应用

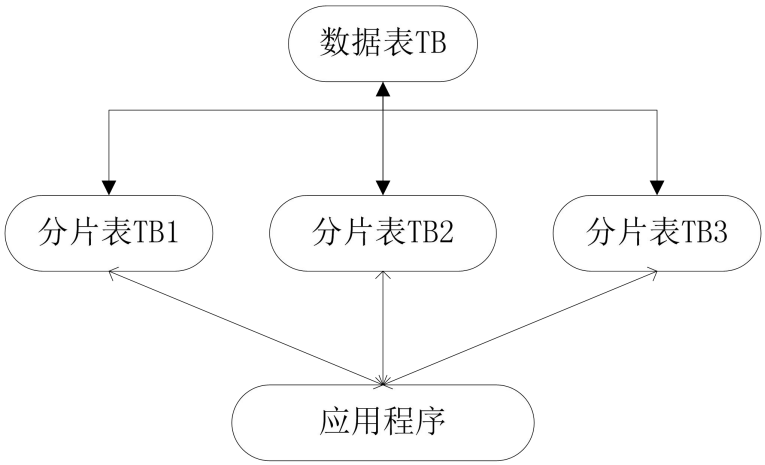
以查询为主的应用时目前的主流应用。例如网站，日常生活中，人们经常上网获取各种有效信息。上网过程中，用户主要的活动是浏览、搜索各种信息，少量地需要提交和修改数据。

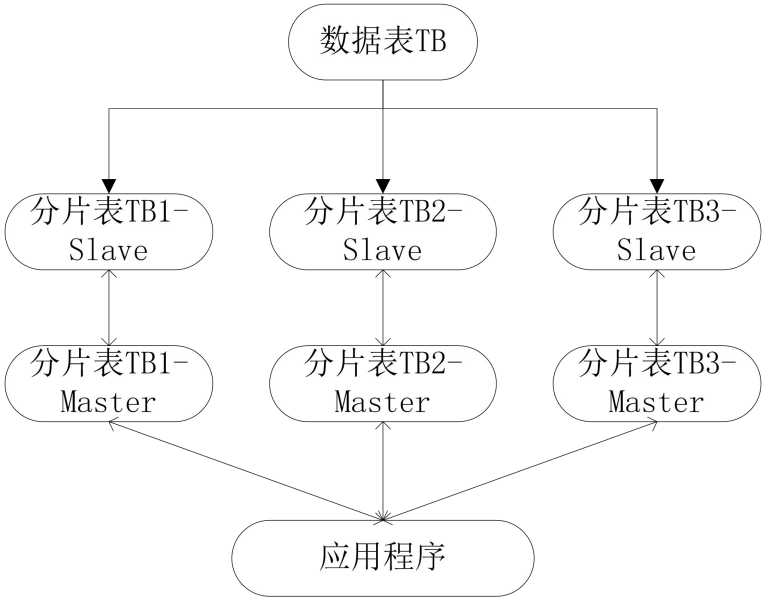
对于这种应用，推荐使用多副本机制，并且使用强一致性。如果查询的数据量很大，并且需要高强度的分析，这里还可以使用表分片的功能，使得大数据的查询能够分布式并行地进行。另外，多副本机制还能实现负载均衡。

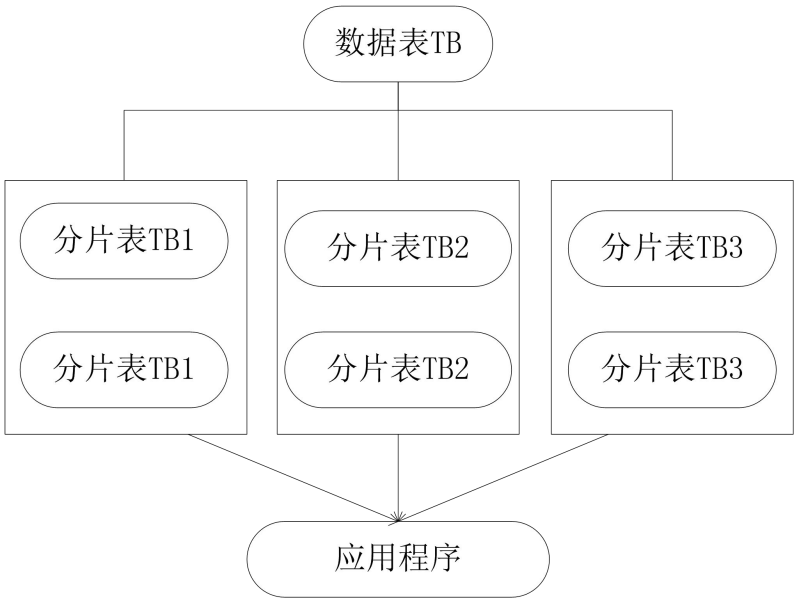


12.2.4 以写数据为主的应用

和前一种应用需求相反，有一些应用需要尽量快的写速度，查询速度要求并不是很高。对于这种应用，我们需要分析一下影响写数据的性能的各种因素以及用户的需求：

需求	方案	优点
写速度快	采用表分片，每个分片表相关的数据库副本数量均为“1”时，并且采用直接访问各个分片表，直接写数据到各个分片。  如上图所示，物理上表 TB、TB1、TB2 和 TB3 属于四个表，逻辑上 TB1、TB2 和 TB3 属于 TB，是 TB 表的三个分片。这样，应用就可以直接访问 TB1、TB2 和 TB3 这三个分片表进行写数据。由于三个分片表物理上独立，拥有独立的表空间，因此同时向三个表写数据时互不干扰，具有较好的性能。三个子分片分别位于不同的服务器时，当写的数据量足够大时可以充分利用各个服务器的系统资源，从而达到最佳的写性能。理论上讲，这里分片越多，写性能将呈线性提高。	写数据的性能最优

<table>
<tr>
<td>写速度快
批量备份</td>
<td>采用表分片，每个分片表相关的数据库副本数量均为“2”时，并且采用直接访问各个分片表，直接写数据到各个分片，每个分片相关数据库属性为批量模式。
 

和前面的情况相比，这里的分片表多出了批量库副本，批量库副本不会刻意降低写表的速度，但是日志线程将稍微地增加系统的开销。因此这里的写性能和前面的情况相比将有稍微的降低。当服务器的性能足够好时，和前者的性能基本一致。</td>
<td>写数据的性能次之</td>
</tr>
<tr>
<td>写速度快
实时备份</td>
<td>采用表分片，每个分片表相关的数据库副本数量均为“2”时，并且采用直接访问各个分片表，直接写数据到各个分片。
 

这里的分片表采用了实时的多副本机制。批量副本机制几乎不降低写的性能。但是实时副本机制为了维护多副本之间的强一致性将降低写数据的性能。</td>
<td>写数据的性能较上者次之，但是查询性能最好</td>
</tr>
</table>

	但是优点却是：1）数据的实时备份，保证了数据的高可用性；2）数据的实时备份，意味着从任意副本上、任意时刻查询的数据具有一致性，因此查询的并行度提高，并行查询性能更好。	

12.2.5 处理大数据

随着互联网科技的快速发展和信息化建设的进一步加强，各种数据和信息越来越多。用户对大数据处理的需求越来越大。

首先，海量数据的处理是存储问题。DosSQL 支持海量数据存储，一台服务器上的一个普通副本理论上可以支持到 64TB 的数据。而 DosSQL 支持把一个逻辑上的数据库以表分片的方式分别存储到多台服务器上，因此，从理论上将，DosSQL 分片数据库的容量远超 64TB，可以无限扩展。

其次，数据的产生速度太快，传统的系统无法满足信息数据的快速写入和更新速度。传统的数据库系统产品一般采用的单机单副本，即使是采用分区的方式，最终也会因为受限于磁盘的 I/O 瓶颈，使数据的写入和更新速度受阻。而 DosSQL 系统支持表分片机制，它能够把逻辑上属于一个表的数据分别部署到不同的多台服务器上，并允许用户同时从逻辑表和各个具体的分片表上进行访问。如果用户从逻辑表写入或者更新数据，不会加快写入数据的速度；但如果用户直接从分片表写入或者更新数据，则会大大地加快写入速度。这里是显而易见的，比如存在 5 台相同配置的 DosSQL 服务器，由于受磁盘 I/O 性能的限制，采用传统的数据库系统的最大写入速度为 Sm，那么，分片表依次部署在 5 台服务器上 DosSQL 系统，在每台服务器上的最大写入速度也应该为 Sm，同时对 5 台 DosSQL 服务器进行写操作时的速度应该为 Sm×5。

最后，从海量数据中检索出有用的信息也是一个重要的问题。在大数据的检索中，磁盘 I/O 的限制仍然是关键所在，一些大数据的离线分析，在指令发出数天甚至一月之后才可能得出查询结果。这样的结果很难让用户满意，但却又无可奈何。DosSQL 支持的表分片机制，可以将大数据尽可能地分解，分存到多台服务器，然后利用其先进的分布式并行查询机制获取结果，从而避免磁盘 I/O 的瓶颈。并且，DosSQL 系统保证：当分片数据库服务器故障后，只要还有可以使用的资源，分布式并行查询还会继续执行，避免节点故障导致整条查询失效。

13 日常维护

日常维护的目的是保证系统正常运行，发现故障后立即解决，并保证数据不丢失。以下将介绍 DosSQL 系统日常维护中经常要用到的方法。

13.1监控系统运行状况

判定 DosSQL 系统运行是否正常，首先要确定组成系统的各台服务器是否在运行，然后确定各台服务器上的服务程序是否运行正常。

DosSQL 系统会自行维护各台服务器的工作状态。用户可以使用系统维护指令“SHOW ALL HOST STATUS”来查询目前整个系统中在工作的服务器；用户还可以使用系统维护指令“SHOW STATION HOST STATUS”来查询当前站中在工作的服务器。

要确定各台服务器上的服务是否运行正常，需要登录各台服务器，然后通过“ps –ax|grep dossqld”命令查询服务进程运行的情况。如果没有服务进程在运行，则可尝试启动服务。如果无法正常启动服务，则查看“/usr/local/dossql/data/”目录下的数据库运行日志，从而了解程序异常退出的原因。

如果各台服务器上的服务都是运行正常的，则可能存在因网络连通等问题数字有机体工作库系统的组织结构出现了问题。这时，可以在每台服务器上查看“Host”和“Host_availability”，以确定各台服务器上看到相同的系统组织结构。如果各台服务器上看到的组织结构不同，则需要启动部分服务器来解决问题。

如果各台服务器上的系统组织结构是相同的，则再确认各台服务器上看到的副本位置信息是否相同。如果系统过于庞大，可以在各个站点中随机地抽取部分服务器来检验。可使用指令“SHOW DATABASES”来查看每台服务器上有的数据库副本，然后使用副本管理指令“SHOW ALL DATABASE STATUS”来查询目前整个系统中的数据库副本位置信息，还可以使用指令“SHOW STATION DATABASE STATUS”来查询当前站点的数据库副本位置信息。如果查询得到的各台服务器上的状态正常（Db_availability 状态为“Yes”）的副本一致，那么说明系统运行正常；否则，可能是副本之间有恢复或者迁移，如果都不是，系统可能出现了故障。

如果检测到有数据库服务器没有运行，通常可能有以下的这些情况：

1) 服务器没有启动。造成这种结果的原因有很多，通常的原因是断电后管理员没有及时开启服务器。处理方式是启动该服务器；
2) 服务器的网络出现故障，导致该服务器和其它的服务器失去联系，处理方式是修复网络，DosSQL 系统会自行修复。
3) 如果检测到服务器后台程序没有运行，或者在反复重新启动。那么此服务器的授权使用日期可能已经“过期”。处理方式是向成都天心悦高科技发展有限公司获取使用授权。

如果检测到服务器运行异常或者副本信息不一致，解决方法是：选定一台服务器为标

准，关闭所有其它不一致的服务器，然后重新启动这些服务器，通过 DosSQL 自动恢复到一致状态。

13.2查看服务器运行负载

服务器上的连接数量可以表示服务器的负载情况。因此，可以登录要查看的服务器，使用“show processlist”指令来查看当前的连接情况。查询是在 DosSQL 的命令行界面下进行的。进入命令行界面方式是在 Linux 终端中键入“DosSQL”，如下图所示。

```
root@server54:/home/dos# dossql
Welcome to the DosSQL monitor.  Commands end with ; or \g.

Copyright (c) 2000, 2014, TianXinYue.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

DosSQL[(none)]$show processlist;
+----+------+-----------+------+---------+------+----------+------------------+
| Id | User | Host      | db   | Command | Time | State    | Info             |
+----+------+-----------+------+---------+------+----------+------------------+
| 21 | root | localhost | NULL | Query   |    0 | starting | show processlist |
+----+------+-----------+------+---------+------+----------+------------------+
1 row in set (0.10 sec)

DosSQL[(none)]$
```

图 11-1 查看服务器连接状况

该指令将逐一列出当前服务器上的在进行的任务(即连接)。如果发现有程序不断连接，导致 DosSQL 的连接数剧增，应该检查该程序的配置是否正确，并及时杀死该程序的进程。

13.3数据库的热备份与导入

支持副本机制的 DosSQL 在服务器之间有备份，但是从防止数据被恶意破坏出发，仍然需要定期离线备份。这里介绍 DosSQL 如何实现数据库的离线备份与导入。

13.3.1 数据备份

dossqldump 客户端工具位于“/usr/local/dossql/bin/”目录，可用来转储数据库或备份数据库，进行备份或将数据转移到另一个 SQL 服务器(不一定是一个 DosSQL 服务器)。转储包含创建表和/或装载表的 SQL 语句。

有 3 种方式来调用 **dossqldump** ：

shell> dossqldump [options] db_name [tables]

shell> dossqldump [options] ---database DB1 [DB2 DB3...]

shell> dossqldump [options] --all--database

如果没有指定任何表或使用了---database 或--all--database 选项，则转储整个数据库。

要想获得你的版本的 **dossqldump** 支持的选项，执行 **dossqldump ---help**。

如果运行 **dossqldump** 没有--quick 或--opt 选项，**dossqldump** 在转储结果前将整个结果集装入内存。如果转储大数据库可能会出现问题。该选项默认启用，但可以用--skip-opt 禁用。

dossqldump 支持下面的选项：

---help，-?

显示帮助消息并退出。

--add-drop--database

在每个 CREATE DATABASE 语句前添加 DROP DATABASE 语句。

--add-drop-tables

在每个 CREATE TABLE 语句前添加 DROP TABLE 语句。

--add-locking

用 LOCK TABLES 和 UNLOCK TABLES 语句引用每个表转储。重载转储文件时插入得更快。

--all--database，-A

转储所有数据库中的所有表。与使用---database 选项相同，在命令行中命名所有数据库。

--allow-keywords

允许创建关键字列名。应在每个列名前面加上表名前缀。

---comments[={0|1}]

如果设置为 0，禁止转储文件中的其它信息，例如程序版本、服务器版本和主机。

--skip—comments 与---comments=0 的结果相同。 默认值为 1，即包括额外信息。

--compact

产生少量输出。该选项禁用注释并启用--skip-add-drop-tables、--no-set-names、

--skip-disable-keys 和--skip-add-locking 选项。

--compatible=name

产生与其它数据库系统或旧的 DosSQL 服务器兼容的输出。值可以为 ansi、mysql323、mysql40、postgresql、oracle、mssql、db2、maxdb、no_key_options、no_tables_options 或者 no_field_options。要使用几个值时，用逗号将它们隔开。这些值与设置服务器 SQL 模式的相应选项有相同的含义。

该选项不能保证同其它服务器之间的兼容性。它只启用那些目前能够使转储输出更兼容的 SQL 模式值。例如，--compatible=oracle 不映射 Oracle 类型或使用 Oracle 注释语法的数据类型。

--complete-insert，-c

使用包括列名的完整的 INSERT 语句。

--compress，-C

压缩在客户端和服务器之间发送的所有信息（如果二者均支持压缩）。

--create-option

在 CREATE TABLE 语句中包括所有 DosSQL 表选项。

---database，-B

转储几个数据库。通常情况，**dossqldump** 将命令行中的第 1 个名字参量看作数据库名，后面的名看作表名。使用该选项，它将所有名字参量看作数据库名。CREATE DATABASE IF NOT EXISTS db_name 和 USE db_name 语句包含在每个新数据库前的输出中。

---debug[=debug_options]，-# [debug_options]

写调试日志。debug_options 字符串通常为'd:t:o,file_name'。

--default-character-set=charset

使用 charsetas 默认字符集。如果没有指定，**dossqldump** 使用 utf8。

--delayed-insert

使用 INSERT DELAYED 语句插入行。

--delete-master-logs

在主复制服务器上，完成转储操作后删除二进制日志。该选项自动启用--master-data。

--disable-keys，-K

对于每个表，用/*!40000 ALTER TABLE tbl_name DISABLE KEYS */;和/*!40000 ALTER TABLE tbl_name ENABLE KEYS */; 语句引用 INSERT 语句。这样可以更快地装载转储文件，因为在插入所有行后创建索引。该选项只适合 MyISAM 表。

--extended-insert，-e

使用包括几个 VALUES 列表的多行 INSERT 语法。这样使转储文件更小，重载文件时可以加速插入。

--fields-terminated-by=...，--fields-enclosed-by=...，--fields-optionally-enclosed-by=...，

--fields-escaped-by=...，--行-terminated-by=...

这些选项结合-T 选项使用，与 LOAD DATA INFILE 的相应子句有相同的含义。

--first-slave，-x

不赞成使用，现在重新命名为--lock-all-tables。

--flush-logs，-F

开始转储前刷新 DosSQL 服务器日志文件。该选项要求 RELOAD 权限。请注意如果结合--all--database(或-A)选项使用该选项，根据每个转储的数据库刷新日志。例外情况是当使用--lock-all-tables 或--master-data 的时候：在这种情况下，日志只刷新一次，在所有 表被锁定后刷新。如果你想要同时转储和刷新日志，应使用--flush-logs 连同--lock-all-tables 或--master-data。

--force，-f

在表转储过程中，即使出现 SQL 错误也继续。

--host=host_name，-h host_name

从给定主机的 DosSQL 服务器转储数据。默认主机是 localhost。

--hex-blob

使用十六进制符号转储二进制字符串列(例如，'abc' 变为 0x616263)。影响到的列有 BINARY、VARBINARY、BLOB。

--lock-all-tables，-x

所有数据库中的所有表加锁。在整体转储过程中通过全局读锁定来实现。该选项自动关闭--single-transaction 和--lock-tables。

--lock-tables，-l

开始转储前锁定所有表。用 READ LOCAL 锁定表以允许并行插入 MyISAM 表。对于事务表例如 InnoDB 和 BDB，--single-transaction 是一个更好的选项，因为它根本不需要锁定表。请注意当转储多个数据库时，--lock-tables 分别为每个数据库锁定表。因此，该选项不能保证转储文件中的表在数据库之间的逻辑一致性。不同数据库表的转储状态可以完全不同。

--master-data[=value]

该选项将二进制日志的位置和文件名写入到输出中。该选项要求有 RELOAD 权限，并且必须启用二进制日志。如果该选项值等于 1，位置和文件名被写入 CHANGE MASTER 语句形式的转储输出，如果你使用该 SQL 转储主服务器以设置从服务器，从服务器从主服务器二进制日志的正确位置开始。如果选项值等于 2，CHANGE MASTER 语句被写成 SQL 注释。如果 value 被省略，这是默认动作。

--master-data 选项启用--lock-all-tables，除非还指定--single-transaction(在这种情况下，只在刚开始转储时短时间获得全局读锁定。又见--single-transaction。在任何一种情况下，日志相关动作发生在转储时。该选项自动关闭--lock-tables。

--no-create-db，-n

该选项禁用 CREATE DATABASE /*!32312 IF NOT EXISTS*/ db_name 语句，如果给出---database 或--all--database 选项，则包含到输出中。

--no-create-info，-t

不写重新创建每个转储表的 CREATE TABLE 语句。

--no-data，-d

不写表的任何行信息。如果你只想转储表的结构这很有用。

--opt

该选项是速记；等同于指定 --add-drop-tables--add-locking --create-option --disable-keys--extended-insert --lock-tables --quick --set-charset。它可以给出很快的转储操作并产生一个可以很快装入 DosSQL 服务器的转储文件。该选项默认开启，但可以用--skip-opt 禁用。要想只禁用确信用-opt 启用的选项，使用--skip 形式；例如，--skip-add-drop-tables 或--skip-quick。

--password[=password]，-p[password]

连接服务器时使用的密码。如果你使用短选项形式(-p)，*不能*在选项和密码之间有一个空格。如果在命令行中，忽略了--password 或-p 选项后面的 密码值，将提示你输入一个。

--port=port_num，-P port_num

用于连接的 TCP/IP 端口号。

--protocol={TCP | SOCKET | PIPE | MEMORY}

使用的连接协议。

--quick，-q

该选项用于转储大的表。它强制 **dossqldump** 从服务器一次一行地检索表中的行而不是检

索所有行并在输出前将它缓存到内存中。

--quote-names，-Q

用‘’字符引用数据库、表和列名。如果服务器 SQL 模式包括 ANSI_QUOTES 选项，用‘"’字符引用名。默认启用该选项。可以用--skip-quote-names 禁用，但该选项应跟在其它选项后面，例如可以启用--quote-names 的--compatible。

--result-file=file，-r file

将输出转向给定的文件。该选项应用在 Windows 中，因为它禁止将新行‘\n’字符转换为‘\r\n’回车、返回/新行序列。

--routines，-R

在转储的数据库中转储存储程序(函数和程序)。使用---routines 产生的输出包含 CREATE PROCEDURE 和 CREATE FUNCTION 语句以重新创建子程序。但是，这些语句不包括属性，例如子程序定义者或创建和修改时间戳。这说明当重载子程序时，对它们进行创建时定义者应设置为重载用户，时间戳等于重载时间。

如果你需要创建的子程序使用原来的定义者和时间戳属性，不使用--routines。相反，使用一个具有 DosSQL 数据库相应权限的账户直接转储和重载 mysql.proc 表的内容。

--set-charset

将 SET NAMES default_character_set 加到输出中。该选项默认启用。要想禁用 SET NAMES 语句，使用--skip-set-charset。

--single-transaction

该选项从服务器转储数据之前发出一个 BEGIN SQL 语句。它只适用于事务表，例如 InnoDB 和 BDB，因为然后它将在发出 BEGIN 而没有阻塞任何应用程序时转储一致的数据库状态。当使用该选项时，应记住只有 InnoDB 表能以一致的状态被转储。例如，使用该选项时任何转储的 MyISAM 或 HEAP 表仍然可以更改状态。

--single-transaction 选项和--lock-tables 选项是互斥的，因为 LOCK TABLES 会使任何挂起的事务隐含提交。

要想转储大的表，应结合--quick 使用该选项。

--socket=path，-S path

当连接 localhost(为默认主机)时使用的套接字文件。

--skip--comments

参见---comments 选项的描述。

--tab=path，-T path

产生 tab 分割的数据文件。对于每个转储的表，**dossqldump** 创建一个包含创建表的 CREATE TABLE 语句的 tbl_name.sql 文件，和一个包含其数据的 tbl_name.txt 文件。选项值为写入文件的目录。

默认情况，.txt 数据文件的格式是在列值和每行后面的新行之间使用 tab 字符。可以使用--fields-xxx 和--行--xxx 选项明显指定格式。

注释：该选项只适用于 **dossqldump** 与 **dossqld** 服务器在同一台机器上运行时。你必须具有 FILE 权限，并且服务器必须有在你指定的目录中有写文件的许可。

--tables

覆盖---database 或-B 选项。选项后面的所有参量被看作表名。

--triggers

为每个转储的表转储触发器。该选项默认启用；用--skip-triggers 禁用它。

--tz-utc

在转储文件中加入 SET TIME_ZONE='+00:00'以便 TIMESTAMP 列可以在具有不同时区的服务器之间转储和重载。(不使用该选项，TIMESTAMP 列在具有本地时区的源服务器和目的服务器之间转储和重载)。--tz-utc 也可以保护由于夏令时带来的更改。--tz-utc 默认启用。要想禁用它，使用--skip-tz-utc。

--user=user_name，-u user_name

连接服务器时使用的 DosSQL 用户名。

--verbose，-v

冗长模式。打印出程序操作的详细信息。

--version，-V

显示版本信息并退出。

--where='where-condition', -w 'where-condition'

只转储给定的 WHERE 条件选择的记录。请注意如果条件包含命令解释符专用空格或字符，一定要将条件引用起来。

例如：

"--where=user='jimf'"

"-wuserid>1"

"-wuserid<1"

--xml，-X

将转储输出写成 XML。

还可以使用--var_name=value 选项设置下面的变量：

max_allowed_packet

客户端/服务器之间通信的缓存区的最大大小。最大为 1GB。

net_buffer_length

客户端/服务器之间通信的缓存区的初始大小。当创建多行插入语句时(如同使用选项--extended-insert 或--opt)，**dossqldump** 创建长度达 net_buffer_length 的行。如果增加该变量，还应确保在 DosSQL 服务器中的 net_buffer_length 变量至少这么大。

还可以使用--set-variable=var_name=value 或-O var_name=value 语法设置变量。然而，现在不赞成使用该语法。

dossqldump 最常用于备份一个整个的数据库：

shell> dossqldump --opt db_name > backup-file.sql

你可以这样将转储文件读回到服务器：

shell> DosSQL db_name < backup-file.sql

或者为：

shell> DosSQL -e "source /path-to--backup/backup-file.sql" db_name

dossqldump 也可用于从一个 DosSQL 服务器向另一个服务器复制数据时装载数据库：

shell> dossqldump --opt db_name | DosSQL --host=remote_host -C db_name

可以用一个命令转储几个数据库：

shell> dossqldump ---database db_name1 [db_name2 ...] > my_databases.sql

如果你想要转储所有数据库，使用--all--database 选项：

shell> dossqldump --all-databases > all_databases.sql

如果表保存在 InnoDB 存储引擎中，dossqldump 提供了一种联机备份的途径(参见下面的命令)。该备份只需要在开始转储时对所有表进行全局读锁定(使用 FLUSH TABLES WITH READ LOCK)。获得锁定后，读取二进制日志的相应内容并将锁释放。因此如果并且只有当发出 FLUSH...时正执行一个长的更新语句，DosSQL 服务器才停止直到长语句结束，然后转储则释放锁。因此如果 DosSQL 服务器只接收到短("短执行时间")的更新语句，即使有大量的语句，也不会注意到锁期间。

shell> dossqldump --all-databases --single-transaction > all_databases.sql

对于点对点恢复(也称为“前滚”，当你需要恢复旧的备份并重放该备份以后的更改时)，循环二进制日志或至少知道转储对应的二进制日志内容很有用：

shell> dossqldump --all-databases --master-data=2 > all_databases.sql

或

shell> dossqldump --all-databases --flush-logs --master-data=2 > all_databases.sql

如果表保存在 InnoDB 存储引擎中，同时使用--master-data 和--single-transaction 提供了一个很方便的方式来进行适合点对点恢复的联机备份。

13.3.2 数据的导入

数据库备份数据时要执行备份 SQL 文本文件，只需要执行备份 SQL 文本文件就能够实现数据库的导入。在远离数据库服务器的客户端，用户先要连接到数据库，然后再执行备份 SQL 文件即可。在数据库服务器上，可以执行以下的指令来完成数据的导入：

```
#cat sql_file_path | DosSQL –uuser –ppassword db
```

注释：以上命令中的“sql_file_path”是 SQL 备份文件的路径，“user”是访问数据库“db”的用户名，“password”是访问数据库“db”的密码。

13.4数据库的冷备份与导入

冷数据的备份是指 DosSQL 服务没有启动的情况下，进行数据备份。发生这样的情况可能有：

- 服务器故障，不能启动，这时需要把服务器的磁盘取下来装在其它的服务器上，然后备份磁盘；
- DosSQL 授权过期，无法启动服务进程。

13.4.1 备份数据

冷备份不能以数据库为单位进行，只能备份整个 DosSQL 的数据目录。备份可以采用拷贝（cp）、远程拷贝（scp），或者压缩文件夹的方式进行。例如，执行以下指令进行压缩备：

1) 执行“tar -zcf /usr/local/dossql_bak.tgz /usr/local/dossql/data/”压缩 DosSQL 数据目录；
2) 拷贝“/usr/local/dossql_bak.tgz”压缩文件到安全的目录。

13.4.2 数据的导入

冷备份的数据是把整个 DosSQL 数据目录都备份起来的，因此还原回去的时候就需要事先删除原有的数据目录，然后解压备份的数据目录，使用解压后目录作为数据目录，具体的操作步骤如下：

1) 删除原有的 DosSQL 数据目录，执行“rm -fr /usr/local/dossql/data/”；
2) 从安全目录拷贝备份数据“/usr/local/dossql_bak.tgz”到“/usr/local/”目录；
3) 到“/usr/local/”目录，执行解压缩命令还原本分数据目录“tar -zxf DosSQL_bak.tgz”。

13.5日志管理

随着系统长时间的运行，日志可能变得越来越大，当系统长久运行时，可能占用较多的磁盘空间。因此管理员应该定期清除日志。清除日志时不得删除日志文件，删除日志文件后，后续的日志将无法得到保存，所以，管理员可按照以下示例的方式清空日志文件。

```
root@server55:/usr/local/dossql/data# l server55.err
-rw-r--r--. 1 root root 63815  1 月  16 09:30 server55.err
root@server55:/usr/local/dossql/data# > server55.err
root@server55:/usr/local/dossql/data# l server55.err
-rw-r--r--. 1 root root 0  1 月  16 10:16 server55.err
root@server55:/usr/local/dossql/data#
```

以上先查看了“server55.err”文件的大小为“63815”；然后调用“>”命令清空了“server55.err”文件；然后再次查看它的大小，发现为“0”。

13.5.1 调试日志

调试日志文件路径在 DosSQL 安装目录的 data 子目录下（如“/usr/local/dossql/data/”），此日志通常名为 hostname.err。当系统出现异常时，可以使用“tracelog”命令来打开日志功能，以方面调试。使用方法如下：

```
tracelog [err|ok|cata|rela|sys|renew|sync|dup|syntax|trans|recovery|orien|batch|dead|privs]
<open|close|flush>
```

默认情况下只有“err”是开启的；如果要全部打开请执行“tracelog open”，全部关闭执行“tracelog close”；日志数据是定量往磁盘写，如果需要及时刷新日志时执行“tracelog flush”；如果需要打开某个模块的日志，执行“tracelog OPTION open”；如果需要关闭某个模块的日志，执行“tracelog OPTION close”。

可选参数 OPTION 有以下选项：

日志参数选项	描述
Err	错误日志，所有错误的日志全部打印，默认开启
Ok	正常调试相关的日志，默认关闭
Cata	副本目录相关的日志，默认关闭
Rela	可靠消息相关的日志，默认关闭
Sys	系统管理相关的日志，默认关闭
Renew	系统重构相关的日志，默认关闭
Sync	定时器相关的日志，默认关闭
Dup	副本管理相关的日志，默认关闭
Syntax	副本管理指令相关的日志，默认关闭
Trans	事务执行相关的日志，默认关闭
Recovery	副本恢复相关的日志，默认关闭
Orien	系统定位相关的日志，默认关闭
Batch	批量事务相关的日志，默认关闭
Dead	节点检测相关的日志，默认关闭
Privs	用户权限相关的日志，默认关闭

13.5.2 查询日志

它记录所有 DosSQL 活动，在诊断问题时非常有用。此日志文件可能会很快地变得非常大，因此不应该长期使用它。此日志通过名为 hostname.log，位于 data 子目录下（如“/usr/local/dossql/data/”）。可用命令“set global general_log”控制。

13.6错误汇报

如果您发现 DosSQL 或 dossqld 程序功能异常。这可能是由于您的使用方式出错，也可能是工作库程序本身存在的 BUG，如果出错，请您务必按照用户手册进行操作，如果仍然有错，请您汇报给我们；如果 DosSQL 或 dossqld 程序发生断错误，也请您务必把错误汇报给我们。

DosSQL 系统的错误可能是由单台服务器引起，也可能发生在多台服务器之间，还有可能产生段错文件“core”。因此，汇报错误时，请您尽量将以下信息汇报给我们。

出现错误的操作方式描述	需要清楚描述导致出错的执行步骤
出现的错误截图	用于直观查看错误情况
出现错误的操作次数	数字
出错的次数	数字
出错日志文件	各个出错相关服务器的“/usr/local/dossql/data/*.err”文件
段错文件	“/usr/local/dossql/data/core”文件

我们需要使用“出现错误的操作次数”和“出现错误的操作次数”来确定该错误属于“总是”还是“偶尔”；利用“出错日志文件”和“段错文件”快速查找错误，以便排除错误。

错误汇报的邮件地址是：tianxinyue@126.com。

14 限制与约束

DosSQL 是在开源 MySQL 5.7 数据库之上研发，DosSQL 几乎全部支持 MySQL 5.7 的功能，但还有少数几个功能暂时不能支持，或者支持的不是很好，我们计划在后续版本的工作库中实现或优化这些功能。

限制与约束	描述
MySQL 复制	MySQL 的复制是以“计算机”为单位的，适用于集群，不再适用于 DosSQL，这里不再支持它。相应地，DosSQL 实现了批量同步机制，他类似于 MySQL 的复制机制，不同点在于它是以“数据库”为单位的，并且更加智能，无需人工干预。MySQL 复制相关指令也不再支持。 这里不再支持复制相关的命令。
包含多个数据库的 SQL	MySQL 中我们可以同时操作多个数据库，比如将 A 库的 a 表插入 B 库的 b 表（insert B.b select * from A.a），但是在 DosSQL 版本中我们暂时不支持这种操作，我们只允许用户操作一个数据库，多库操作计划在下版本中实现。
非授权语句对权限表的操作	为了使用户在系统中的任意一台服务器上的授权对整个系统都有效和增强系统权限的安全性，请用户在授权时使用 grant\revoke\drop user 标准的授权语句，而不要使用普通的 insert\delete\update 等语句。
handler 语法	DosSQL 不支持 handler 语法，因为 DosSQL 必须保证所有数据的实时有效，也就是不允许有读取脏数据的情况出现。同时 DosSQL 数据库对查询语句提供服务的能力非常强大，用户完全可以使用普通的查询语句进行查询而没有必要使用可能读取到无用甚至有害的脏数据的 handler 读取方法。当然为了满足某些需要使用 handler 方法的用户，DosSQL 数据库计划在下版本里实现支持该功能。
临时表	不支持，但支持临时表的语法。使得“CREATE TEMPORARY TABLE”和“CREATE TABLE”表示相同的含义。使得“DROP TEMPORARY TABLE”和“DROP TABLE”表示相同的含义。
XA 事务	不支持。
SQLCOM_LOAD	该指令类型表示 LOAD 相关的指令，包括 LOAD DATA FROM MASTER、LOAD DATA INFILE、LOAD TABLE FROM MASTER。这些命令在使用时受到一些限制：首先，需要在工作库副本 IP 地址最小的节点上使用；其次，操作的数据的内容不宜太多，太多可能导致系统内存耗尽而出错。因此，建议在但副本时使用这些指令。
变量	变量包括系统变量和自定义变量。DosSQL 的变量的使用和 MySQL 的使用有一些不一样。DosSQL 在使用变量（包括设置和引用）时，务必先要连接一个使用的数据库，否则，变量可能不能有效地使用。

	DosSQL 支持大多数的 MySQL 原有系统变量，但部分不支持。所有支持的系统变量可以通过“show variables”指令来查看。比如变量 log-bin、innodb_file_per_table 不再支持。
分片表的管理	不支持表的 HASH 分片。 不支持对分片表的列的修改和重新定义。 不支持重新分片后，把新的分片数据连接到原来的分片。 分片表将大表数据分割，分别放置在多个表中。使用分片表的情况适用于大型表，较小的表使用反而会降低表的使用性能。 分片表的分片放置在不同的服务器上，重新分片可能会花销大量的时间，建议不要随便使用。 执行分片管理 SQL 后，原来不再使用的分片还存在，这里的目的是防止数据丢失，如果用户不再需要，需要手动删除。 分片表的实现继承自表分区，因此表分区的诸多限制也适用于表分片。
SQLCOM_BACKUP_TABLE	BACKUP TABLE 是 MyISAM 存储引擎类型的命令，不支持。
SQLCOM_RESTORE_TABLE	RESTORE TABLE 是 MyISAM 存储引擎类型的命令，不支持。
SET PASSWORD	不支持，可以由 GRANT 来实现修改用户密码。

15 常见问题与解答

1) 安装失败

解决方法：如果用 rpm 安装失败，则确定是否安装了 rpm 软件，并且使用。同时检查系统是否已经安装了该系统，如果已安装，则先卸载再安装。

2) 清除本机数据库内容

解决方法：删除/usr/local/dossql/data 下所有目录（除开 mysql 目录外）。

3) DosSQL 不一致、某个数据库数据遭到破坏或者副本全部为非可用状态

解决方法：

①选定一个数据库为标准数据库，关闭所有其它不一致数据库的服务器；

②重新启动这些数据库服务器，通过 DosSQL 自动恢复到数据库一致状态。

4) DosSQL 服务器不能启动

解决方法：

①停止不能正常启动节点的数据库服务；

②重新启动该数据库节点。如果此时数据库能够正常启动则结束；否则，执行以下步骤；

③停止该数据库节点；

④清除所有数据库内容；

⑤重新启动该数据库节点。

5) 无法连接 DosSQL 服务器

解决方法：

①在数据库服务器所在位置运行 DosSQL 连接数据库；

②如果连接成功，则检查网络是否连接正常；

③如果连接失败，检查是否授权；

④如果提示需要刷新缓存的用户表，这通常是错误连接过多造成的，请运行 DosSQLadmin flush-hosts。

6) DosSQL 运行期间发生连接丢失的情况

解决方法：

①检查系统是否有数据库节点正在进行重新启动数据库的操作。如果有，则等待该节点启动正常；否则，执行后面步骤；

②停止该数据库节点；

③如果有节点与其它节点没有同步，则检查配置文件中的端口设置是否一致，如果端口设置不一致则修改成一致后重新启动该节点；

④如果是 blob 数据在插入时丢失，则检查 my.cnf 中[dossqld]下的 max_allow_packet 是否设置过小，如果是，修改后重新启动该节点即可。

7) 如果运行时发现数据库操作速度降低

解决方法：检查是否有数据库节点正在重新启动。如果是，因为在节点启动后，整个系统会降低速度进行数据同步，因此速度降低是正常的。否则，重新启动 DosSQL。

8) 创建数据库失败

解决方法：DosSQL 创建数据库时如果提示：some node execute sql error，表明在某些节点上该操作请求失败。此时，检查是否有节点已经操作成功，如果系统中没有该库的信息则重新执行创建数据库的操作。

9) 客户端访问被拒绝

出现提示：Access denied for user: “username@servername” （using password: YES）或 Host “servername” is not allowed to connect to this DosSQL server 表明用户没有访问该数据库服务器的权限，同时表明用户使用了密码。

解决方法：检查用户权限是否正确，如果没有权限应向数据库管理员申请。如果用户权限正确而连接失败，请检查数据库的连接数是否达到最大连接数，或者是否出现网络错误。

10) 部分服务器线程已经存在

出现提示：Can’t start server: Bind on TCP/IP port: Address already in use;Do you already hare another dossqld server running on port: 3306?

这表明数据库服务器有部分线程还在运行，没有完全退出。

解决方法：查看 dossqld 的进程 id，kill 后重启。

如果通过以上的处理后仍然存在其它问题，请联系成都天心悦高科技发展有限公司。

16 附录一：DosSQL 系统保留字

系统指令保留字一览表

ADD	ALL	ALTER
ANALYZE	AND	AS
ASC	ASENSITIVE	BEFORE
BETWEEN	BIGINT	BINARY
BLOB	BOTH	BY
CALL	CASCADE	CASE
CHANGE	CHAR	CHARACTER
CHECK	COLLATE	COLUMN
CONDITION	CONNECTION	CONSTRAINT
CONTINUE	CONVERT	CREATE
CROSS	CURRENT_DATE	CURRENT_TIME
CURRENT_TIMESTAMP	CURRENT_USER	CURSOR
DATABASE	DATABASES	DAY_HOUR
DAY_MICROSECOND	DAY_MINUTE	DAY_SECOND
DEC	DECIMAL	DECLARE
DEFAULT	DELAYED	DELETE
DESC	DESCRIBE	DETERMINISTIC
DISTINCT	DISTINCTROW	DIV
DOUBLE	DROP	DUAL
EACH	ELSE	ELSEIF
ENCLOSED	ESCAPED	EXISTS
EXIT	EXPLAIN	FALSE
FETCH	FLOAT	FLOAT4
FLOAT8	FOR	FORCE
FOREIGN	FROM	FULLTEXT
GOTO	GRANT	GROUP
HAVING	HIGH_PRIORITY	HOUR_MICROSECOND
HOUR_MINUTE	HOUR_SECOND	IF
IGNORE	IN	INDEX
INFILE	INNER	INOUT
INSENSITIVE	INSERT	INT

INT1	INT2	INT3
INT4	INT8	INTEGER
INTERVAL	INTO	IS
ITERATE	JOIN	KEY
KEYS	KILL	LABEL
LEADING	LEAVE	LEFT
LIKE	LIMIT	LINEAR
LINES	LOAD	LOCALTIME
LOCALTIMESTAMP	LOCK	LONG
LONGBLOB	LONGTEXT	LOOP
LOW_PRIORITY	MATCH	MEDIUMBLOB
MEDIUMINT	MEDIUMTEXT	MIDDLEINT
MINUTE_MICROSECOND	MINUTE_SECOND	MOD
MODIFIES	NATURAL	NOT
NO_WRITE_TO_BINLOG	NULL	NUMERIC
ON	OPTIMIZE	OPTION
OPTIONALLY	OR	ORDER
OUT	OUTER	OUTFILE
PRECISION	PRIMARY	PROCEDURE
PURGE	RAID0	RANGE
READ	READS	REAL
REFERENCES	REGEXP	RELEASE
RENAME	REPEAT	REPLACE
REQUIRE	RESTRICT	RETURN
REVOKE	RIGHT	RLIKE
SCHEMA	SCHEMAS	SECOND_MICROSECOND
SELECT	SENSITIVE	SEPARATOR
SET	SHOW	SMALLINT
SPATIAL	SPECIFIC	SQL
SQLEXCEPTION	SQLSTATE	SQLWARNING
SQL_BIG_RESULT	SQL_CALC_FOUND_ROWS	SQL_SMALL_RESULT
SSL	STARTING	STRAIGHT_JOIN
TABLE	TERMINATED	THEN
TINYBLOB	TINYINT	TINYTEXT
TO	TRAILING	TRIGGER
TRUE	UNDO	UNION

UNIQUE	UNLOCK	UNSIGNED
UPDATE	USAGE	USE
USING	UTC_DATE	UTC_TIME
UTC_TIMESTAMP	VALUES	VARBINARY
VARCHAR	VARCHARACTER	VARYING
WHEN	WHERE	WHILE
WITH	WRITE	X509
XOR	YEAR_MONTH	ZEROFILL
以下为 DosSQL 扩展语法涉及到的保留字：		
ADMIN_SECTION	ADMIN_STATION	ADMIN_NODE
RANDOM_MASTER	WAIT_TIME	RELAY_TIME_THRESHOLD
MAX_SUM_DUP	MIN_SUM_DUP	OUT_SECTION
OUT_STATION	SLOW_LOG_EXE	MASTER_ADDR

17 附录二：DosSQL 服务器调优参数概述

调优参数一览表：

调整参数	说明
max_connections	dossqld 服务器允许的最大连接数。该值越大应用层发生连接获取失败的可能性越低。
binlog_cache_size	在事务过程中容纳二进制日志 SQL 语句的缓存大小。二进制日志缓存是服务器支持事务存储引擎并且服务器启用了二进制日志(--log-bin 选项)的前提下为每个客户端分配的内存。如果你经常使用大的，多语句事务，你可以增加该值以获得更有的性能。Binlog_cache_use 和 Binlog_cache_disk_use 状态变量可以用来调整该变量的大小。
join_buffer_size	用于完全联接的缓冲区的大小(当不使用索引的时候使用联接操作)。一般情况获得快速联接的最好方法是添加索引。当增加索引时不可能通过增加 join_buffer_size 值来获得快速完全联接。将为两个表之间的每个完全联接分配联接缓冲区。对于多个表之间不使用索引的复杂联接，需要多联接缓冲区。
key_buffer_size	MyISAM 表的索引块分配了缓冲区，由所有线程共享。key_buffer_size 是索引块缓冲区的大小。键值缓冲区即为键值缓存。对数字有机体系统无效。因为没有使用 MyISAM 表。
query_cache_size	为缓存查询结果分配的内存的数量。默认值是 0，即禁用查询缓存。请注意即使 query_cache_type 设置为 0 也将分配此数量的内存。
query_prealloc_size	用于查询分析和执行的固定缓冲区的大小。在查询之间该缓冲区不释放。如果你执行复杂查询，分配更大的 query_prealloc_size 值可以帮助提高性能，因为它可以降低查询过程中服务器分配内存的需求。
read_buffer_size	每个线程连续扫描时为扫描的每个表分配的缓冲区的大小(字节)。如果进行多次连续扫描，可能需要增加该值， 默认值为 131072。
read_rnd_buffer_size	当排序后按排序后的顺序读取行时，则通过该缓冲区读取行，避免搜索硬盘。将该变量设置为较大的值可以大大改进 ORDER BY 的性能。但是，这是为每个客户端分配的缓冲区，因此你不应将全局变量设置为较大的值。相反，只为需要运行大查询的客户端更改会话变量。
sort_buffer_size	每个排序线程分配的缓冲区的大小。增加该值可以加快 ORDER BY 或 GROUP BY 操作。
max_heap_table_size	单内存表（heap table）的最大允许值。该值用于防止在该值过小的情况下意外创建的大型 heap table 会耗尽内存。

tmp_table_size	如果内存内的临时表超过该值，dossqld 自动将它转换为硬盘上的 MyISAM 表。如果你执行许多高级 GROUP BY 查询并且有大量内存，则可以增加 tmp_table_size 的值。
thread_concurrency	并发运行的线程数，设为总 CPU 核数乘以 2
innodb_file_io_threads	InnoDB 中文件 I/O 线程的数量。正常地，这个参数是用默认的，默认值是 4。可以是 CPU 数目。
innodb_flush_log_at_trx_commit	当 innodb_flush_log_at_trx_commit 被设置为 0，日志缓冲每秒一次地被写到日志文件，并且对日志文件做到磁盘操作的刷新，但是在一个事务提交不做任何操作。当这个值为 1（默认值）之时，在每个事务提交时，日志缓冲被写到日志文件，对日志文件做到磁盘操作的刷新。当设置为 2 之时，在每个提交，日志缓冲被写到文件，但不对日志文件做到磁盘操作的刷新。尽管如此，在对日志文件的刷新在值为 2 的情况也每秒发生一次。我们必须注意到，因为进程安排问题，每秒一次的刷新不是 100%保证每秒都发生。你可以通过设置这个值不为 1 来获得较好的性能，但随之你会在一次崩溃中损失二分之一价值的事务。如果你设置这个值为 0，那么任何 **dossqld** 进程的崩溃会删除崩溃前最后一秒的事务，如果你设置这个值为 2，那么只有操作系统崩溃或掉电才会删除最后一秒的事务。尽管如此，InnoDB 的崩溃恢复不受影响，而且因为这样崩溃恢复开始作用而不考虑这个值。注意，许多操作系统和一些磁盘硬件会欺骗刷新到磁盘操作。尽管刷新没有进行，你可以告诉 **dossqld** 刷新已经进行。即使设置这个值为 1，事务的持久程度不被保证，且在最坏情况下掉电甚至会破坏 InnoDB 数据库。在 SCSI 磁盘控制器中，或在磁盘自身中，使用有后备电池的磁盘缓存会加速文件刷新并且使得操作更安全。
innodb_thread_concurrency	innodb 核心允许驻留的线程数量，太高的值会导致线程混乱。通常为总 CPU 核数至 2 倍值。
innodb_buffer_pool_size	innodb 缓冲池大小。该值越大越少访问磁盘 IO。如果设置太高又会发生内存不足，通常设置成物理内存的 50%-80%，在 32 位系统上不超过 2-3G。