# 

# MySQL数据库应用指南

目录

[1.1 数据库介绍 9](#_Toc7055)

[1.1.1 什么是数据库 9](#_Toc27986)

[1.2 数据库的种类 9](#_Toc12457)

[1.2.1 关系型数据库 9](#_Toc177)

[1.2.2 非关系型数据库介绍 10](#_Toc13979)

[1.2.2.1 非关系型数据库诞生背景 11](#_Toc1607)

[1.2.2.2 非关系型数据库种类 11](#_Toc22712)

[1.3. 常用关系型数据库产品介绍 13](#_Toc7164)

[1.3.1 ORALCE数据库 13](#_Toc26009)

[1.3.2 MYSQL数据库 13](#_Toc17651)

[1.3.3 MariaDB数据库 14](#_Toc20231)

[1.3.4 SQL Server数据库 15](#_Toc25173)

[1.3.5 Access数据库 15](#_Toc22012)

[1.3.6 其他不常用的关系型数据库 16](#_Toc27592)

[1.4 常用非关系型数据库产品介绍 16](#_Toc11216)

[1.4.1 memcached(key-value) 16](#_Toc11362)

[1.4.2 redis(key-value) 17](#_Toc23004)

[1.4.3 MongoDB(Document-oriented) 18](#_Toc12364)

[2.1 MYSQL数据库介绍 18](#_Toc2832)

[2.1.1 mysql数据库介绍 18](#_Toc12409)

[2.1.2 为什么选项MYSQL数据库 19](#_Toc28492)

[2.2 MYSQL数据库分类与版本升级 20](#_Toc29980)

[2.2.1 MYSQL数据库商业版与社区版区别 20](#_Toc28289)

[2.2.1.2 MYSQL数据库社区版本发布介绍 20](#_Toc11457)

[2.2.1.3 MYSQL数据库商业版与社区版区别 20](#_Toc20968)

[2.2.2 MYSQL数据库四种发布版本介绍 21](#_Toc25905)

[2.3 MYSQL产品路线 22](#_Toc31689)

[2.3.1 mysql产品路线变更历史背景 22](#_Toc31892)

[2.3.2 第一条.5.0.xx到5.1.xx产品线系列介绍 22](#_Toc19117)

[2.3.3 第二条-5.4.XX开始-到了5.7.XX产品线系列介绍 23](#_Toc21603)

[2.3.4 第三条-6.0.XX-到7.1.XX产品线系列介绍 23](#_Toc27377)

[2.4 MYSQL数据库软件命名介绍 23](#_Toc7464)

[2.5 生产环境下如何选择MYSQL版本 24](#_Toc12814)

[2.5.1 MYSQL数据库版本选择最终建议 24](#_Toc2347)

[3.1 MYSQL数据库库安装 25](#_Toc14616)

[3.1.1 YUM/rpm方式安装MYSQL 25](#_Toc25967)

[3.1.2 常规方式编译安装MYSQL 26](#_Toc22595)

[3.1.3 采用cmake方式编译安装MYSQL 27](#_Toc21421)

[3.1.4 采用二进制方式免编译安装MYSQL 27](#_Toc29545)

[3.1.5 如何正确选择MYSQL的安装方式 28](#_Toc30011)

[3.2 生产场景下常规方式安装MYSQL5.5.32 28](#_Toc18405)

[3.2.1 CMAKE软件的安装 28](#_Toc5622)

[3.2.2 依赖包ncurses-devel的安装 29](#_Toc11850)

[3.2.3 创建mysql用户和用户组 29](#_Toc5816)

[3.2.5 mysql数据库安全优化 31](#_Toc8578)

[3.3 mysql多实例介绍 31](#_Toc31958)

[3.3.1 什么是MYSQL多实例 32](#_Toc21681)

[3.3.2 MYSQL多实例的作用于问题 32](#_Toc14536)

[3.3.2.1 有效利用服务器资源 32](#_Toc3585)

[3.3.3.2 节约服务器资源 32](#_Toc31886)

[3.3.3.3 资源互相抢占问题 32](#_Toc2487)

[3.4 MYSQL多实例生产应用场景 33](#_Toc27994)

[3.4.1 资金紧张型公司的选择 33](#_Toc18763)

[3.4.2 并发访问不是特别大的业务 33](#_Toc981)

[3.4.3 门户网站应用MYSQL多实例场景 33](#_Toc241)

[3.5 MYSQL多实例常见配置方案 34](#_Toc28024)

[3.5.1 多实例文件部署方案 34](#_Toc10947)

[3.6 安装多实例MYSQL数据库 36](#_Toc16497)

[3.6.1 建立MYSQL多实例的条件 36](#_Toc21542)

[3.6.2 建立MYSQL账号 36](#_Toc8829)

[3.6.3 获得MYSQL软件 36](#_Toc17413)

[3.6.4 安装MYSQL软件 37](#_Toc32495)

[3.6.5 创建MYSQL多实例的数据目录 37](#_Toc24606)

[3.6.6 创建MYSQL多实例的配置文件 38](#_Toc30078)

[3.3.7 创建MYSQL多实例的启动文件 40](#_Toc32391)

[3.3.8.2 两种配置MYSQL路径方法 41](#_Toc27675)

[3.3.8.3 因MYSQL环境配置顺序导致的错误案例 41](#_Toc7365)

[3.3.9 初始化MYSQL多实例的数据库文件 41](#_Toc9286)

[3.3.10 启动MYSQL多实力数据库 42](#_Toc28726)

[3.3.11 MYSQL故障排除多实例数据库 42](#_Toc6205)

[3.3.12 配置MYSQL多实例数据库开机自启动 46](#_Toc8979)

[3.3.13 如何再增加一个MYSQL实例 46](#_Toc27236)

[3.3.14 小结 46](#_Toc1646)

[4.1 启动与关闭MYSQL 47](#_Toc17551)

[4.1.1 单实例MYSQL启动与关闭方法 47](#_Toc5923)

[4.1.2 多实例MYSQL启动与关闭方案实例 51](#_Toc28014)

[4.2 登陆Mysql方法 51](#_Toc15378)

[4.2.1 单实例MYSQL登陆的方法 51](#_Toc29239)

[4.2.2 多实例MYSQL登陆的方法 52](#_Toc32668)

[4.3 登陆MYSQL后的操作 53](#_Toc2085)

[4.4 善用MYSQL的帮助命令help 53](#_Toc23502)

[4.5 退出MYSQL数据库方法 54](#_Toc23222)

[4.6 MYSQL数据库用户安全策略介绍 54](#_Toc27464)

[4.6.1 为管理员root用户设置密码方法 55](#_Toc24802)

[4.6.2 修改管理员root密码方法一：命令行外修改法 55](#_Toc23694)

[4.6.3 修改管理员root密码方法二：sql语句修改法 55](#_Toc10307)

[4.6.5 修改管理员root密码方法三 56](#_Toc12162)

[4.6.6 找回丢失mysql root用户密码 56](#_Toc12863)

[4.6.6.1 启动修改丢失的MYSQL单实例root密码方法 56](#_Toc24644)

[4.6.7 多实例MYSQL启动修改丢失root密码方法 57](#_Toc29812)

[4.7 结构化查询语言 57](#_Toc23131)

[4.7.1 什么是SQL 57](#_Toc2480)

[4.7.2 SQL的分类 58](#_Toc16196)

[4.8.1 创建数据库 59](#_Toc24514)

[4.8.2 显示数据库 61](#_Toc22218)

[4.8.3 删除数据库 61](#_Toc21398)

[4.8.4 连接数据库 61](#_Toc13623)

[4.8.5 查看当前的数据库 62](#_Toc20277)

[4.8.6 当前数据库包含的表信息 62](#_Toc12234)

[4.8.7 删除MYSQL系统多余账号 62](#_Toc211)

[4.9.1 通过help查看grant命令帮助 62](#_Toc31243)

[4.9.2 通过GRANT命令创建用户并授权 63](#_Toc15191)

[4.9.3 CREATE和GRANT配合法 64](#_Toc23237)

[4.9.4 授权局域网内主机远程连接数据库 65](#_Toc21496)

[4.9.5 MYSQL用户可以授权的权限有哪些 66](#_Toc7582)

[4.9.6 企业生产环境如何授权用户权限 66](#_Toc11745)

[4.10 表操作 67](#_Toc20138)

[4.10.1 指定字符集建库 67](#_Toc27656)

[4.10.2 建立表 67](#_Toc30578)

[4.10.3 MYSQL表的字段类型 69](#_Toc17043)

[4.10.4 生产环境标准的UF8格式表结构语句 71](#_Toc4415)

[4.10.5 查看建立的表结构 71](#_Toc27775)

[4.10.6 查看已经建立的表的语句 71](#_Toc16040)

[4.10.7 为表的字段创建索引 71](#_Toc12132)

[4.11.2 根据指定条件查询表的部分数据 77](#_Toc11961)

[4.11.3 多表查询 78](#_Toc26258)

[4.12 修改表中数据 81](#_Toc24108)

[4.12.1 修改表中指定条件固定列的数据 81](#_Toc6107)

[4.12.2 修改表中所有行的数据 82](#_Toc20362)

[4.12.3 防止误操作MySQL数据库一例 83](#_Toc26553)

[4.13 删除表中数据 83](#_Toc22829)

[4.13.1 实践删除表中的数据 83](#_Toc12221)

[4.14 增删改表的字段 84](#_Toc24698)

[4.14.1 命令语法及默认添加演示 84](#_Toc20843)

[4.14.2 指定添加列在表里的位置 84](#_Toc21456)

[4.14.3 生产环境多个复杂添加修改多字段信息的案例 84](#_Toc28904)

[4.14.4 rename法 85](#_Toc249)

[4.15 删除表 85](#_Toc24104)

[4.16 MYSQL插入中文数据乱码问题 85](#_Toc5157)

[4.16.2 MYSQL命令行插入中文数据不乱码实战 86](#_Toc26751)

[4.16.3 执行SQL文件插入中文数据不乱码实战 87](#_Toc11187)

[4.16.4 通过MYSQL命令加字符集参数导入数据解决乱码 87](#_Toc30653)

[4.16.5 MYSQL插入中文不乱码5种方法小结 87](#_Toc3930)

[4.17 MYSQL数据库字符集知识 90](#_Toc24391)

[4.17.2 MYSQL如何选择合适的字符集 90](#_Toc19814)

[4.17.3 查看当前MYSQL系统支持的字符集 91](#_Toc17092)

[4.18 mysql插入中文数据不乱码深度剖析 91](#_Toc22318)

[4.18.1 MYSQL数据库默认字符集是什么 91](#_Toc24913)

[4.18.2 执行set names latin1到底做了什么 92](#_Toc15856)

[4.18.3 mysql命令参数--default-character-set=latin1在做什么 92](#_Toc874)

[4.19 确保MYSQL数据库插入数据不乱码解决方案 93](#_Toc31657)

[4.19.1 统一MYSQL数据库客户及服务端字符集 93](#_Toc26842)

[4.19.2 更改MYSQL服务端字符集 94](#_Toc20397)

[4.19.3 统一MYSQL数据库客户机服务端字符集)总结 94](#_Toc16245)

[4.19.4 彻底解决MYSQL数据库插入中文乱码的方案 95](#_Toc13578)

[4.20 如何更改生产MYSQL数据库库表的字符集 96](#_Toc12399)

[4.20.1对于已经存在表记录的字符集的修改 96](#_Toc26879)

[5.1 mysql 数据库的备份与恢复 97](#_Toc16773)

[5.1.1 备份数据的意义 97](#_Toc27138)

[5.1.2 备份单个数据库练习多种参数使用 98](#_Toc28757)

[5.1.3 mysqldump的工作原理 100](#_Toc23598)

[5.1.4 备份多个库及多个参数练习 100](#_Toc24985)

[5.1.5 如何做分库备份 101](#_Toc26143)

[5.1.6 备份单个表 102](#_Toc2855)

[5.1.7 备份多个表 102](#_Toc29043)

[5.1.8 备份数据库表结构(不包含数据) 103](#_Toc14038)

[5.1.9 刷新binlog的数据 104](#_Toc21643)

[5.1.10 MYSQL的关键参数 104](#_Toc28097)

[5.2 恢复数据库实践 105](#_Toc22351)

[5.2.1 利用source命令恢复数据库 105](#_Toc5764)

[5.2.2 其他知识 106](#_Toc21238)

[5.3 mysql生产环境案例 107](#_Toc9821)

[5.3.1 mysql数据库批量插入数据shell脚本实现 107](#_Toc27937)

[5.3.2 mysqlbinlog命令的详解和实战 108](#_Toc1114)

[6.1 MYSQL主从复制 109](#_Toc12497)

[6.1.1 普通的文件的数据同步 109](#_Toc30460)

[6.2 MYSQL主从复制 110](#_Toc7160)

[6.2.1 MYSQL主从复制介绍 110](#_Toc30715)

[6.3 MYSQL主从复制的应用场景 111](#_Toc11965)

[6.4 如何实现MYSQL主从读写分离 112](#_Toc26376)

[6.5 MYSQL数据库主从同步实战 113](#_Toc12710)

[6.5.1 定义服务器角色表 113](#_Toc7104)

[6.5.2 数据库环境准备 114](#_Toc21215)

[6.5.3 主从复制部署配置问题汇总 117](#_Toc17389)

[6.5.4 回顾MYSQL主从同步配置步骤及原理 118](#_Toc23732)

[6.5.4.1 MYSQL主从同步配置步骤 118](#_Toc31863)

[6.5.5 生产场景快速配置MYSQL主从复制方案 119](#_Toc5400)

[6.5.6 一键自动化配置主从复制脚本 119](#_Toc233)

[6.5.6.1 生产环境主从库同步配置注意事项 119](#_Toc1028)

[6.5.7 无需熬夜在工作时间轻松配置从库 120](#_Toc12283)

[6.5.8 不停主库一键批量创建从库 121](#_Toc12655)

[6.5.8.2 一键创建从库方法2 123](#_Toc28025)

[6.5.9 相关MYSQL主从复制技术技巧概览 124](#_Toc18141)

[6.5.9.1 登陆数据库查看MYSQL线程同步状态 124](#_Toc26132)

[6.5.9.2 复制主线程状态 124](#_Toc19061)

[6.5.9.3 复制从I/O线程状态 125](#_Toc19121)

[6.5.9.4 复制从SQL线程状态 126](#_Toc10132)

[6.5.9.5 查看MYSQL现场同步状态的用途 127](#_Toc1815)

[6.6 生产场景MYSQL主从复制读写分离授权方案及实战 127](#_Toc6460)

[6.6.1 生产MYSQL复制环境用户授权方案 127](#_Toc22320)

[6.6.2 通过忽略授权表的方式防止数据写从库的方法及实践 128](#_Toc27865)

[6.7 MYSQL主从复制的故障解决 130](#_Toc30525)

[6.7.1 MYSQL从库数据冲突导致同步停止 130](#_Toc2084)

[6.7.2 让MYSQL SLAVE库记录binlog方法 132](#_Toc20134)

[6.8 生产场景案例 132](#_Toc13106)

[6.8.1 主库宕机，从库切换主库角色后，实现主从同步的过程 132](#_Toc12876)

[6.8.2 从库宕机 134](#_Toc28658)

[6.8.3 双主机多主同步过程实战 135](#_Toc659)

[6.8.5 更多MYSQL数据库增量恢复总结 143](#_Toc12968)

[6.8.6 增量恢复小结 143](#_Toc19059)

[7.1 Mysql服务日志 144](#_Toc11042)

[7.2 binlog日志的三种模式 145](#_Toc28753)

[7.2.1 Statement Level模式 145](#_Toc3263)

[7.2.2 Row Level模式 146](#_Toc20085)

[7.2.3 Mixed模式 147](#_Toc29567)

[7.2.4 调整binlog日志模式方法 148](#_Toc30804)

[7.2.4.1 在配置文件中修改方法 148](#_Toc13996)

[7.2.4.2 在线修改立即生效方法 148](#_Toc23306)

[8.1 MYSQL引擎概述 149](#_Toc6833)

[8.1.1 什么是存储引擎 149](#_Toc21443)

[8.1.2 MYSQL存储引擎的架构 149](#_Toc2197)

[8.1.3 Myisam引擎介绍 149](#_Toc23773)

[8.1.3.2 MYISAM引擎特点 150](#_Toc5167)

[8.1.4 MYISAM引擎适合的生产业务场景 151](#_Toc15787)

[8.1.5 MYISAM引擎调优精要 151](#_Toc20242)

[8.2 数据库事务 152](#_Toc15543)

[8.2.1 数据库事务介绍 152](#_Toc27621)

[8.2.2 事务的四大特性(ACID) 152](#_Toc7413)

[8.2.3 事务的开启 153](#_Toc6443)

[8.3 InnoDB引擎 153](#_Toc21705)

[8.3.1 什么是InnoDB引擎 153](#_Toc13787)

[8.3.2 InnoDB引擎的特点 154](#_Toc10452)

[8.3.3 InnoDB引擎适用的场景 154](#_Toc22418)

[8.3.4 InnoDB引擎调优精要 155](#_Toc5579)

[8.3.5 如何确定MYSQL服务器又那些引擎可用 156](#_Toc6004)

[8.3.6 MYSQL引擎的my.cnf对应参数说明 156](#_Toc19046)

[8.3.7 生产环境中如何批量更改MYSQL引擎 157](#_Toc764)

[8.4 有关MYSQL引擎常见企业面试题 157](#_Toc14399)

[9.1 网站打开慢及MYSQL慢查询优化的案例 158](#_Toc19798)

[9.1.1 MYSQL慢查询优化过程 158](#_Toc12666)

[9.1.2 Memcached服务应用优化案例 162](#_Toc21063)

[9.2 MYSQL数据库安全权限控制管理思想 163](#_Toc8674)

[9.2.1 制度与流程控制 163](#_Toc4096)

[9.2.1.1 项目开发制度流程 163](#_Toc5639)

[9.2.1.2 数据库更新流程 163](#_Toc7489)

[9.2.3 DBA参与项目数据库设计 164](#_Toc23633)

[9.2.4 各种操作申请流程 164](#_Toc32469)

[9.2.5 定期对内部人员培训 164](#_Toc7773)

[9.3 账户权限控制 164](#_Toc30711)

[9.3.1 内部开发等人员权限分配 164](#_Toc6724)

[9.3.2 web账户权限分配制度 165](#_Toc24321)

[9.3.3 数据库客户端访问控制的策略 165](#_Toc8974)

[9.3.4 数据库管理思想的核心 166](#_Toc8329)

[9.4 MYSQL数据库的优化 166](#_Toc2535)

[9.5 高手分享 168](#_Toc11340)

[9.5.1 通过PHP应用程序实现MYSQL数据库读写分离 168](#_Toc24015)

[9.5.2 amoeba软件实现数据库读写分离实战 168](#_Toc8863)

[9.5.2.1 服务器架构规划 168](#_Toc7635)

[9.5.2.2 amoeba读写分离示意图 169](#_Toc3646)

[9.5.2.3 MYSQL主从复制的配置 169](#_Toc6679)

[9.5.2.4 MYSQL读写分离功能的实现 170](#_Toc5549)

[9.5.2.5 MYSQL\_proxy实现数据库的读写分离实战 176](#_Toc25149)

# 1.1 数据库介绍

# 1.1.1 什么是数据库

简单的说，数据库(英文Database)就是一个存放数据的仓库，这个仓库是按照一定的数据结构(数据结果是指数据的组织形式或数据之间的联系)来组织，存储的，我们可以通过数据库提供的多种方法来管理数据库的数据,更简单的形象理解，数据库和我们生活中存放杂物的储物间仓库性质一致，区别只是存放的东西不同，杂物间存放实体的物件，而数据库里存储的是数据，这样我们就对数据库有一个初步的了解了。

数据库诞生于距现在大约六十多年前，随着信息技术的发展和人类社会的不断进步,特别是2000年以后，数据库不再仅仅是存储和管理数据了，而转变用户所需要的各种数据管理方式，数据库有很多种类和功能，从最简单的存储有各种数据的表格能够进行海量数据存储的大型数据库系统都在各个方面得到了广泛的应用。

1.2 数据库的种类

按照早期的数据库理论，比较流行的数据库模型由三种，分别为层次式数据库，网络式数据库和关系型数据库。而在当今的互联网中，最常用的数据库模型主要有两种即关系型数据库和非关系型数据库，我们主要讲解关系型数据库和非关系型数据库这两种最重要的也是目前实际使用最多的数据库种类

1.2.1 关系型数据库

(1)关系型数据库的由来

虽然网状数据库和层次数据库已经很好地解决了数据的集中和共享问题，但是在数据库独立性和抽象性级别上仍然有很大的欠缺，用户对这两种数据库进行存取时，仍然需要明确数据的存储结构，指出存取路径，而关系型数据库就可以很好地解决这些问题

(2)关系型数据库的介绍

关系型数据库模型是把负载的数据结构归结为简单的二元关系(即二维表格形式)，例如大学里某一期的学生关系就是一个二元关系，在关系数据库中，对数据的操作几乎全部建立在一个或多个关系表格上，通过对这些关联的表格分类，合并，连接或选取等运算来实现数据的管理，

关系型数据库诞生距今已有40多年了，从理论产生发展到现实产品。例如大家最常见的MYSQL和ORACLE数据库，ORALCE在数据库领域里上升到了霸主地位，形成每年高达数百亿美元的庞大产业市场，而MYSQL也是不容忽视的数据库，以至于被ORACLE重金收购了。

(3)关系型数据库表格之间的关系举例用access或power builder完成下来关系表格



1.2.2 非关系型数据库介绍

1.2.2.1 非关系型数据库诞生背景

非关系型数据库也被称为NOSQL数据库，请读者注意，NOSQL的本意是"Not only SQL",指的是非关系型数据库，而不是"NO SQL"的意思(没有SQL语句？)，因此，NOSQL的产生并不是要彻底地否定关系型数据库，而是作为传统关系型数据库的一个有效补充，NOSQL数据库在特定的场景下可以发挥出难以想象的高效率和高性能。

随着互联网WEB2.0网站的兴起，传统的关系型数据库的应付WEB2.0网站，特别是对于规模日益扩大的海量数据，超大规模和高并发的微博，微信，SNS类型的WEB 2.0纯动态网站已经显的力不从心，保留了很多难以克服的问题，例如：传统的关系型数据库IO瓶颈，性能瓶颈都难以有效突破，于是开始出现了大批针对特定场景，以高性能和使用便利为目的功能特异化的数据库产品，NOSQL(非关系型)类的数据库就是这样的情景中诞生并得到了非常迅速的发展。NOSQL是非关系型数据库的广义定义。它打破了长久以来关系型数据库与ACID理论大一统的局面。NOSQL数据存储不需要固定的表结构，通常也不存在连接操作，在大数据存取上具备关系型数据库无法比拟的性能优势。该术语(NOSQL)在2009年初

得到了广泛认同

当今的应用体系结构需要数据存储在横向伸缩性上能够满足需求，而NOSQL存储就是为了实现这个需求而诞生的。GOOGLE的BigTable与Amazon的Dynamo是非常蟾宫的商业NOSQL实现。一些开源的NOSQL体系，如Facebook的Cassandra,Apache的HBase,也得到了广泛认同，redis,mongodb也越来越受到各类大中小型公司的欢迎和追捧。

1.2.2.2 非关系型数据库种类

(1)键值(key-value)存储数据库

键值数据库就类似传统语言中使用的哈希表。可以通过KEY来添加，查询或者删除数据，因为使用KEY主键访问，所以会获取很高的性能及扩展性。

键值(key-value)数据库主要是使用一个哈希表，这个表中有一个特定的键和一个指针指向特定的数据，KEY/VALUE模型对于IT系统来说的优势在于简单，易部署，高并发。典型产品：memcached,radis,memcacheDB,Berkeley DB

(2)列存储(Column-oriented)

列存储数据库将数据存储存在列族中，一个列族存储经常被一起查询的相关数据。举个例子，如果我们有一个Person类，我们通常会一起查询他们的姓名和年龄而不是薪资。这种情况下，姓名和年龄就会被放入一个列族中，而薪资则在另一个列族中。

这部分数据库通常是用来应付分布式存储的海量数据，键仍然存在，但是它们的特定是指向了多个列，这些列是由列家族来安排的。典型产品：Cassandra,HBase

(3)面向文档(Document-Oriented)

文档型数据库的灵感是来自于Lotus Notes办公软件的，而且它同第一种键值存储种类似。该类型的数据模型是版本化的文档，半结构化的文档以特定的格式存储，比如JSON。文档型数据库的查询效率更高。

面向文档数据库会将数据以文档的形式存储。每个文档都是自包含的数据单元，是一系列数据项的集合。每个数据项都有一个名称与对应的值，值即可以是简单的数据类型，如字符串，数字和日期等；也可以是复杂的类型，如有序列表和关联对象。数据存储的最小单位是文档，同一个表中存储的文档属性可以是不同的,数据可以使用XML,JSON或者JSONB等多种形式存储。典型产品：MongoDB,CouchDB

(4)图形(Graph)数据库

图形数据库允许我们将数据以图的方式存储。实体会被作为顶点，而实体之间的关系则会被作为边，比如我们有三个实体，stevejobs,apple和next，则会有两个"Founded by"的边将Apple和Next连接到steve Jobs.

图形结构的数据库同其他行列以及刚性结构的SQL数据库不同，它是使用灵活的图形模型，并且能够扩展到多个服务器上，NOSQL数据库没有标准的查询语言(SQL) ,因此进行数据库查询需要制定数据模型。许多NOSQL数据库都有RESET式的数据接口或者查询API。典型产品：Neo4J,InfoGrid

# 1.3. 常用关系型数据库产品介绍

## 1.3.1 ORALCE数据库



oracle前身叫SDL，由Larry Ellison和另两个编程人员在1977创办，他们开发了拳头产品，在市场上大量销售，1979年，ORACLE公司引入了第一个商用SQL关系数据管理系统。ORALCE公司是最早开发关系数据库的厂商之一，其产品支持最广泛的操作系统平台。目前ORACLE关系数据库产品的市场占有率数一数二

ORACLE公司是目前最大的数据库软件公司，也是近年业务增长极为迅速的软件提供与服务商。

2007年7月12日，甲骨文公司在美国纽约宣布退出数据库ORACLE 11G,这是ORACLE数据库的最新版本，ORACLE介绍说，ORACLE 11G有400多项功能，经过了1500万个小时的测试，开发工作量达到了3.6万人/月。ORACLE 11G在安全，XML DB,备份等方面得到了很大提升。

主要应用范围：传统大企业，大公司，政府，金融，证券等等

版本升级：ORACLE8i,ORACLE9i,ORACLE10g,ORACLE11g

## 1.3.2 MYSQL数据库



MYSQL数据库是一个中小型关系型数据库管理系统，软件开发者为瑞典MYSQL AB公司.在2008年1月16号被SUN公司收购，后SUN公司又被ORACLE公司收购。

目前MYSQL被广泛地应用在Internet上的大中型网站中。由于其体积小，速度快，总体拥有成本低，尤其是开放源码这一特点，许多大中型网站为了降低网站总体拥有成本而选择了MYSQL作为网站数据库，设置国内知名的淘宝网也选择弃用ORACLE而更换为更开放的MYSQL。

MYSQL数据库主要应用范围：互联网领域，大中小型网站，游戏公司，电商平台等等。

## 

## 1.3.3 MariaDB数据库



MariaDB数据库管理系统是MYSQL数据库的一个分支，主要由开源社区维护，采用GPL授权许可，开发这个MariaDB数据库分支的可能原因之一是：甲骨文公司收购了

MYSQL后，又将MYSQL闭源的潜在风险，因此MYSQL开源社区采用分支的方式来避开这个风险开发MariaDB数据库的目前是完全兼容MYSQL数据库，包括API和命令行，使之能轻松的称为MYSQL的代替品，在存储引擎方面，使用XtraDB(英语：XtraDB)来代替MYSQL的InnoDB,MariaDB由MYSQL的创始人Michael Widenius主导开发，他早前曾以10亿美元的价格，将自己创建的公司MYSQL AB卖给SUN，此后随着SUN被甲骨文收购MYSQL的所有权也落入ORACLE的手中，MariaDB数据库的名称来组MYSQL的创始人Micael widenius的女儿Maria的名字。

MariaDB基于事务的Maria存储引擎，替换了MYSQL的MYISAM存储引擎，它使用了Percona的XtraDB(InnoDB的变体)，这个版本还包含了PrimeBase XT(PBXT)和FederatedX存储引擎。

MariaDB数据库直到5.5版本，均依照MYSQL的版本发行。因此，使用MariaDB5.5的人会从MYSQL5.5中了解到MariaDB的所有功能

1.3.4 SQL Server数据库

Microsoft SQL Server是微软公司开发的大型关系型数据库系统，SQL server的功能比较全面，效率高，可以作为中型企业或单位的数据库平台。SQL server与windows操作系统紧密集成，不论是应用程序开发速度还是系统事务处理运行速度，都能得到较大的提升，对于在windows平台上开发的各种企业级信息管理系统来说，不论是C/S(客户机/服务器)架构还是B/S(浏览器/服务器)架构，SQL server的缺点是只能在windows系统下运行。

1987年，微软和IBM合作开发完成0S/2，IBM是其销售的OS/2 ExtendedEdition系统中绑定了OS/2 Database Manager,而微软产品线中尚缺少数据库产品，为此,微软将目光投向了Sybase，同Sybase签订了合作协议，使用Sybase的技术开发了基于OS/2平台的关系型数据库，1989年，微软发布了SQL server1.0版.Microsoft,在于SYbase分道扬镳后，随后在其6.05和7.0版本中重写了核心数据库系统，如今SQL SERVER 2008即将到来。

主要应用范围：部分企业电商(央视购物)，使用windows服务器平台的企业。

1.3.5 Access数据库

美国Microsoft公司于1994年推出的微机数据库管理系统，它具有界面友好，易学易用，开发简单，接口灵活等特点，是典型的新一代桌面关系型数据库管理系统。他结合了Microsoft Jet Database Engine和图形用户界面两项特点，是Microsoft office的成员之一。Access能够存取Access/Jet,Microsoft SQL Server,ORACLE ,或者任何ODBC兼容数据库的资料。ACCESS界面友好而且易学易用，作为OFFICE的套件的一部分，可以与OFFICE集成，实现无缝连接，ACCESS提供了表,查询,窗体,报表,宏,模块等用来建立数据库系统的对象，提供了多种向导，生成器，模板,把数据存储，数据查询，界面设计，报表生成等操作规范化。

ACCESS是入门级小型桌面数据库，性能安全性都很一般。可供个人或小型网站之用，ACCESS不是数据库语言，只是一个数据库程序，目前最新版本为OFFICE 2007。

早期应用领域：小型程序ASP+ACCESS系统，留言板，校友录等。

1.3.6 其他不常用的关系型数据库

DB2.PostgreSQL,Informix,Sybase等，这些关系型数据库逐步的淡出了普通运维的视线，特别是互联网公司几乎见不到，因此我们这里就不多介绍了。

# 1.4 常用非关系型数据库产品介绍

1.4.1 memcached(key-value)



memcached是一个开源的，高性能的，具有分布式内存对象的缓存系统，通过它可以减轻数据库负载，加速动态的WEB应用，最初版本由LiveJournal和Brad Fitzpatrick在2003年开发完成。目前全球有非常多的用户都在使用它来构建自己的大负载网站或提高自己高访问网站的响应速度。注意：memcache是这个项目的名称，而memcached是服务器端的主程序文件名。

缓存一般是用来保存一些经常被存放的对象或数据(例如，浏览器会把经常访问的网页缓存起来一样)，通过缓存来存取对象或数据要比在磁盘上存取快很多，前者是内存，后者是磁盘，memcached是一种纯内存缓存系统，把经常存取的对象或数据缓存在memcached的内存中，这些被缓存的数据被程序通过API的方式被存取，

memcached里面的数据就像一张巨大的HASH表，数据以key-value对的方式存在。

memcached通过缓存经常被存取的对象或数据，从而减轻频繁读取数据库的压力，提高网站的响应速度，构件出速度更块的可扩展的WEB应用,由于memcached为纯内存缓存软件，一旦重启后所有的数据都会丢失，因此新浪网

基于memcached开发乐意开源项目Memcachedb，通过为memcached增减了BerkeleyDB的持久化机制和异步主辅机制，使memcached具备了事务恢复能力，持久化数据存储能力和分布式复制能力，memcachedb非常适合需要超高性能读写速度，持久化保存的应用场景，但是最近几年来逐步被其他持久化产品所替代，例如redis

1.4.2 redis(key-value)



和memcached类似，redis也是key/value型存储系统。但redis支持的存储value 类型相对很多。包括String,list,set,zset等.这些数据类型都支持push/pop,add/remove及取交集，并集和差集及更丰富的操作，而且这些操作都是原子性的，在此基础上,redis支持各种不同方式的排序，与memcached一样，为了保证效率，redis的数据都是缓存在内存中，区别是redis会周期性的把更新的数据写入磁盘或修改再写入追加的记录文件，并且在此基础上实现了master-slave(主从)同步。redis是一个高性能的key-value数据库。redis的出现，很大程度补偿了memcached这类key/value存储的不足，在部分场合可以对关系数据库起到很好的补充作用。它提供了python,ruby,erlang,php客户端，使用很方便。

redis特点：

1)支持内存缓存，这个功能相当于memcached

2)支持持久化，这个功能相当于memcachedb,ttserver

3)数据类型更丰富，比其他key-value库功能更强

4)支持集群，分布式

5)支持队列等特殊功能

应用:缓存从存取memcached更改存取redis

## 1.4.3 MongoDB(Document-oriented)



MongoDB是一个介于关系数据库和非关系数据库之间的产品，是非关系数据库当中功能最丰富，最像关系数据库的。他支持的数据库结构非常松散，类似JSON的BJSON格式，因此可存储比较复杂的数据类型,Mongodb最大的特点是他支持的查询语言非常强大，其语法优点类似于面向对象的查询语言，几乎可以实现类似关系数据库单表查询的绝大部分功能，而且还支持数据建立索引，他的特点是高性能，易部署，易使用，存储数据非常方便。

主要功能特性：

1)面向集合存储，易存储对象类型的数据

2)模式自由

3)支持动态查询

4)支持完全索引，包含内部对象

5)支持查询

6)支持复制和故障恢复

7)使用高效的二进制数据存储，包括大型对象(如视频等)

8)自动处理碎片，以支持云计算层次的扩展性

9)支持RUBY,PYTHON,JAVA,C++,PHP等多种语言

10)文件存储格式为BSON(一种JSON的扩展)

11)可通过网络访问

2.1 MYSQL数据库介绍

2.1.1 mysql数据库介绍

mysql数据库是一款深受欢迎的开源领域的重要的关系型数据库产品，由瑞典MYSQL AB 公司开发与维护，2006年，MYSQL AB公司被SUN公司收购，并作了很多改进，2008年，SUN公司又被数据库龙头公司甲骨文收购，因此MYSQL数据库目前属于ORACLE公司，称为传统数据库领域老大的又一个数据库产品，甲骨文公司收购MYSQL后，使的自身在商业数据库与开源数据库领域市场占有的份额都跃居第一位的位置，这样的格局，引起了业内很多的人士的担忧，这种担忧直接导致了后来的MYSQL分支数据库MariaDB的诞生于发展

前面已经介绍过，MYSQL属于传统关系型数据库产品，它的开放式的架构使得用户选择性很强，同时社区开发与维护人数众多，其功能比较稳定，性能卓越，且在遵守GPL协议的前提下，可以免费使用与修改，也可为MYSQL的推广与使用带来了更多利好。在MYSQL成长与发展过程中，支持的功能逐渐增多，性能也在不断提高，对于平台的支持越来越多，

mysql是一种关系型数据库管理系统，关系型数据库的特点是将数据保存在不同的表中，在将这些表放入不同的数据库中，而不是将所有数据统一放在一个大仓库里，这样的设计增加了MYSQL的读取速度，灵活性可管理性也得到了很大提高，访问以及管理MYSQL数据库的最常用标准化语言为SQL结构化查询语言。

## 2.1.2 为什么选项MYSQL数据库

毫无疑问，即成事实，绝大多数的使用linux操作系统的大中小互联网网站都在使用

MYSQL作为其后端的数据库存储，从大型的BAT门户，到电商平台，分类门户等无一例外都使用MYSQL数据库，那么，MYSQL数据库到底有哪些优势和特点，让大家义无反顾的选择它呢

原因可能有以下几点

1)MYSQL性能卓越，服务稳定，很少出现异常宕机

2)MYSQL开放源代码且无版权制约，自主性及使用成本低

3)MYSQL历史悠久，社区及用户非常活跃，遇到问题，可以寻求帮助

4)MYSQL软甲体积小，安装使用简单，并且易于维护，安装维护成本低

5)MYSQL品牌口碑效应，使的企业无需考虑就直接用之，LAMP，LEMP流行架构

6)MYSQL支持多种操作系统，提供多种API接口，支持多种开发语言，特别对流行的PHP语言有很好的支持。

2.2 MYSQL数据库分类与版本升级

MYSQL数据库的官方网站http://www.mysql.com.其发布的MYSQL版本采用的双授权政策，和大多数开源产品路线一样，分为社区版和商业版，而这两个版本又各自分四个版本依次发布，这四个版本为，Alpha版，Beta版，RC版，GA版本。

2.2.1 MYSQL数据库商业版与社区版区别

MYSQL商业版是由MYSQL AB公司内部专门的人员负责开发及维护，但同时也会吸纳社区人员编写的优秀代码及算法，并且由专门的人员严格按照软件测试流程对这些采纳的代码进行严格测试，没有问题再进行发布。简单地说，MYSQL商业版是由MYSQL公司内部发布，同时参考社区版的先进代码功能和算法，是MYSQL公司的盈利产品，需要付费才能使用及提供服务支持，稳定性和可靠性无疑都是最好的，当然了，企业腰包的够鼓的起才能买的起，据作者了解到的信息，其知名分类门户网站早期就购买过MYSQL商业版，价格不必那些闭源的商业数据库便宜，也是大几十万，

### 2.2.1.2 MYSQL数据库社区版本发布介绍

MYSQL社区版则是由分散在世界各地的MYSQL开发者，爱好者以及用户参与开发与测试，并完成软件代码的管理，测试工作，社区也会设立BUG汇报机制，收集用户使用遇到BUG问题情况，相比商业版，社区版的开发测试及测试环境没有那么严格

### 2.2.1.3 MYSQL数据库商业版与社区版区别

1)商业版本组织管理与测试环节控制更严格，稳定性方面，会比社区版本更稳定

2)MYSQL是成熟产品，商业版与社区版之间性能方面相差不大

3)商业版不遵守GPL协议，社区版遵守GPL协议可以免费使用

4)使用商业版后可以购买相关的服务，享受7\*24小时技术支持以及定时打补丁等服务，但是用户必须为此支付服务费用

5)社区版本的维护服务只能靠社区提供，无法想商业版本获得故障及补救解决服务了,但是,使用社区版是完全免费的方式，社区版的服务质量与时效性等方面就无法与MYSQL AB公司提供的服务相比

2.2.2 MYSQL数据库四种发布版本介绍

在前面的内容已经阐述过，MYSQL的版本发布采用双授权政策，即分为社区版和商业版，而这两个版本又各自分四个版本依次分布：Alpha版本，Beta版本，RC版本GA版本。

这四种发布版本之间的说明及区别，如下：

1)Alpha版：

Alpha版一般只是在开发的公司内部运行，不对外公开，主要是开发者自己对产品进行测试，检查产品是否存在缺陷，错误，验证产品功能与说明书，用户手册是否一致，MYSQL是属于开发源代码的开源产品，因此需要世界各地开发者，爱好者和用户参与软件的开发，测试和手册编写等工作。所以会对外公布此版本的源码和产品，方便任何人可以参与开发或测试工作，设置编写与修改用户手册

2)Beta版：

Beta版一般是完成功能的开发和所有的测试工作之后的产品，不会存在较大的功能或性能BUG，并且邀请或提供给用户体验与测试，以便更全面地测试软件的不足之处或存在的问题

3)RC版：

RC版属于生产环境发布之前的一个小版本或称侯选版，是根据Beta版本测试结果，收集到的BUG或缺陷之处等收集的信息，进行修复和完善之后的一版产品：

4)GA版：

GA版是软件产品正式发布的版本，也称生产版本的产品，一般情况下，企业的生产环境都会选择GA版本的MYSQL软件，用于真实的生产环境中，偶尔有个别大型企业会追求新功能驱动而牺牲稳定性使用其他版本，但这是个例

MYSQL四种发布版本选择说明：

MYSQL AB官方网站会把五种数据库版本都提供下载，主要是MYSQL数据库属于开放源代码的数据库产品，鼓励全球的技术爱好者参与研发，测试，文档编写和经验分享，甚至包过产品发展规划，对于Development版本，Alpha版本和Beta版是绝对不允许使用在任何生产环境的，肯定存在重大的问题或功能未完全实现，绝大多数情况下RC版本也是不允许使用在生产环境，毕竟这是一个GA版本之前，也即生产版本发布之前的一个小版本。另外，对MYSQL数据库GA版本，也是需要慎重选择，开源社区产品毕竟不是经过严格的测试工序完成的产品，是全球开源技术人员的自愿完成的，灰存在比商业产品稳定性弱的缺陷。更严格的选项见后文

官方地址提供的数据库下载版本：http://mysql.ntu.edu.tw/Downloads/

# 2.3 MYSQL产品路线

2.3.1 mysql产品路线变更历史背景

早期，MYSQL也是遵循版本号逐渐增加的方式发展，格式例如:mysql-X.XX.XX.tar.gz,例如，DBA老鸟们都非常熟悉的生产场景版本：4.1.7,5.0.56等。

近几年，为了提高MYSQL产品的竞争优势，以及提高性能，降低开发维护成本等原因，同时，更方便企业用户更精准的选择适合的版本产品用于自己的企业生产环境中，

MYSQL在发展到5.1系列版本之后，重新规划为三条产品线

2.3.2 第一条.5.0.xx到5.1.xx产品线系列介绍

第一条产品线:5.0.xx及升级到5.1.xx的产品系列，这条产品线继续完善与改进其用户体验和性能，同时增加新功能，这条路线可以说是MYSQL早期产品的延续系列，这一系列的产品发布情况及历史版本如下：

MYSQL5.1是当前稳定(产品质量)发布系列。只针对漏洞修复重新发布：没有增加会影响稳定性的新功能

\*MYSQL 5.1:Pervious stable(production-quality)relase:MYSQL 5.0是前一稳定(产品质量)发布系列，只针对严重漏洞修复和安全修复重新发布；没有增加会影响该系列的重要功能

\*MYSQL 5.0:Older stable release nearing the end of the product lifecycle:MYSQL 4.0和3.23是旧的稳定(产品质量)发布系列。该版本不再使用，新的发布只用来修复特别严重的漏洞(以前的安全问题)

2.3.3 第二条-5.4.XX开始-到了5.7.XX产品线系列介绍

第二条产品线：为了更好地整合MYSQL AB公司和第三方公司开发的新存储引擎，以及吸收新的实现算法，从而更好地支持SMP架构，提高性能而做了大量的代码重构，版本编号为从5.4.XX开始，目前发展到了5.6.X

\*MYSQL 5.5:Latest General Availability(Production)release互联网公司使用MYSQL5.5较多

2.3.4 第三条-6.0.XX-到7.1.XX产品线系列介绍

第三条产品线：为了更好地推广MYSQL CLUSTER版本，以及提高MYSQL CLUSTER的性能和稳定性，以及功能改进和增加，以及改动MYSQL基础功能，使其对CLUSTER存储引擎提供更有效的支持与优化。版本号为6.0.XX开发，目前发展到7.1.xx

http://dev.mysql.com/downloads/mysql/

官方地址提供的数据库下载版本：<http://mysql.ntu.edu.tw/Downloads/>

# 2.4 MYSQL数据库软件命名介绍

MYSQL数据库软件的名字由3个数字和一个后缀组成的版本号。例如，像MYSQL-5.0.56.tar.gz的版本号这样解释

1)第一个数字(5)是主版本号，描述了文件格式。所有版本5发行都有相同文件格式

2)第二个数字(0)是发行级别。主版本号和发行级别组合到一起构成了发行序列号

3)第三个数字(56)是在此发行系列的版本号，随着每个新分发版递增。通常你需要选择的发行(release)的最新版本(版本).每次更新后，版本字符串的最后一个数字递增，如果相对于前一个版本增加了新功能或有微小的不兼容性，字符串的第二个数字递增，如果文件格式改变，第一个数字递增。

4)后缀显示发行的稳定性级别，通过一系列后缀显示如何改进稳定性，可能的有

.alpha版

表明发行包含大量未被彻底测试的新代码，已知的缺陷应该在新闻小节被记录，在大多数版本alpha版本中也有新的命令和扩展，alpha版本也可能有主要代码更改等开发。但是我们在发行前一定对其进行测试

.beta版

意味着该版本功能是完整的，并且所有的新代码被测试，没有增加重要的新特征，应该没有已知的缺陷，当alpha版本至少一个月没有出现报导的致命漏洞，并且没有计划增加导致已经实施的功能不稳定的新功能时，版本则从alpha版本变为beta版，在以后的beta版本，发行版本，或产品发布中，所有API,外部可是结构和SQL命令行均不再更改

RC版本

是发行代笔：是一个发行了一段时间的beta版本，看起来应该运行正常，只增加了很小的修复(发布代表即以前所称的gamma版)没有后缀

例如:mysql-5.0.56.tar.gz，这意味着该版本已经在很多地方运行一段时间了，而且没有非平台特定的缺陷报告，这就是我们成为一个产品(稳定)或通用版本的东西

# 2.5 生产环境下如何选择MYSQL版本

## 2.5.1 MYSQL数据库版本选择最终建议

企业生产场景选择MYSQL数据库建议

1)选择开源的社区版的稳定版GA版本

2)可以选择5.1或5.5,互联网公司主流5.5，其次是5.1和5.6

3)选择MYSQL数据库GA版发布后6个月以上的GA版本

4)要选择前后几个月没有大的BUG修复版本，而不是大量修复BUG的集中版本

5)最好向后较长时间没有更新发布的版本

6)要考虑开发人员开发程序使用的版本是否兼容你选的版本

7)作为内部开发测试数据库环境，跑大概3-6个月的时间

8)优先企业非核心业务采用新版本的数据库GA版本软件

9)向DBA高手请教，或者在及技术氛围好的群里和大家一起交流，使用真正的高手们使用过的好用的GA版本

10)经过上述工序之后，若是没有重要的功能BUG或性能瓶颈，则可以开始考虑作为任何业务数据服务的后端数据库软件

# 3.1 MYSQL数据库库安装

在当今的大中型互联网企业里，MYSQL数据库几乎都运行在linux系列操作系统上，当然，你也可以运行在windows/uninx等商业操作系统上，大中型互联网使用开源领域的产品MYSQL数据库目的是，软件代码的开放性和以及无版权免费应用节约成本，因此，作者推荐中小企业采用LAMP/LNMP网站架构，后面的数据库只是我们也是以linux(centos6.5)平台为例来讲述如何安装及使用MYSQL软件的。

## 3.1.1 YUM/rpm方式安装MYSQL

MYSQL官方网站提供了不同版本的RPM安装包，并且针对不同硬件或操作系统平台，又分为不同类型的安装包，大家都可以在官方网站下载页面进行选择，国内也有一些互联网公司提供了镜像，比如网易，阿里云公司提供的镜像资源就非常不错

注意：YUM/RPM方式安装MYSQL适合所有MYSQL软件产品系列

1)RPM包方式安装MYSQL

RPM包的安装非常简单，以MYSQL-5.1.40为例，我们可以通过光盘或则网上下载到四个安装包

YUM安装四个安装包图片.jpg

其中，我们必须要安装MYSQL-server\*和MYSQLclient\*软件包，对于另外二个则个视为实际需要，但是建议一起安装，为此我们可以把这四个RPM包至于某个目录中然后执行SHELL命令：rpm -ivh MySQL.\*则完成MYSQL软件安装任务，剩下的就是如何配置的问题了，

(2)YUM方式安装MYSQL

YUM方式安装MYSQL安装及其简单只要执行yum install mysql-server即可,yum/rpm方式安装MYSQL应用场景

YUN/RPM安装适合于对数据库要求不太高的场合，例如并发不大，公司内部，企业内部等一些场景大的门户网站把源码根据企业需求制作成RPM，搭建YUM仓库，自己来YUM安装

3.1.2 常规方式编译安装MYSQL

所谓常规方式编译安装MYSQL就是延续早期MYSQL的3部曲安装方式，即./configure;make,make install，

生产场景的具体命令及参数为

|  |
| --- |
| ./configure \  -prefix=/application/mysql5.1.72 \  --with-unix-socket-path=/application/mysql5.1.72/tmp/mysql.sock \  --localstatedir=/application/mysql5.1.72/data \  --enable-asembler \  --enable-thread-safe-client \  --with-mysqld-user=mysql \  --with-big-tables \  --without-debug \  --with-pthread \  --enable-assembler \  --with-extra-charsets=complex \  --with-readline \  --with-ssl \  --enable-local-infile \  --with-plugins=partition,innobase  --with-mysqld-ldflags=-all-static \  --with-client-ldflags=-all-static  make  make install |

此种方式适合所有MYSQL5.0.XX-5.1.XX产品系列，是最常规的编译方式

3.1.3 采用cmake方式编译安装MYSQL

由于MYSQL5.5.XX-5.6.XX产品系列特殊性，所以编译方式也和早期的产品安装方式不同，采用CMAKE或GMAKE方式编译安装，即cmake;make;make install,生产场景的具体命令及参数为：

|  |
| --- |
| tar zxf mysql-5.5.32.tar.gz  cd mysql-5.5.32  cmake -DCMAKE\_INSTALL\_PREFIX=/usr/local/cmake  -DMYSQL\_UNIX\_ADDR=/usr/local/mysql/mysql.sock  -DDEFAULT\_CHARSET=utf8  -DDEFAULT\_COLLATION=utf8\_general\_ci  -DWITH\_MYISAM\_STORAGE\_ENGINE=1  -DWITH\_INNOBASE\_STORAGE\_ENGINE=1  -DWITH\_MEMORY\_STORAGE\_ENGINE=1  -DWITH\_READLINE=1  -DENABLED\_LOCAL\_INFILE=1  -DMYSQL\_DATADIR=/mydata/data  -DMYSQL\_USER=mysql  -DMYSQL\_TCP\_PORT=3306  make  make install |

3.1.4 采用二进制方式免编译安装MYSQL

采用二进制方式免编译安装MYSQL，这种方法和YUM/RPM包方式安装类似，适合各类MYSQL产品系列，不需要复杂的编译设置及编译时间等待，直接解压下载的软件包，初始化即可完成MYSQL的安装启动

## 

## 3.1.5 如何正确选择MYSQL的安装方式

YUM/RPM安装适合对数据库要求不太高的场合，例如并发不大，公司内部，企业内部的一些场景，二进制免安装比较简单方便，适合5.0-5.1和5.5-5.6系列，是很多专业DBA的选择，普通运维人员多采用编译的方式，5.0-5.1系列及时常规编译方式，5.5-5.6系列就是CMAKE编译方式.所以综合来讲，这些安装方式都可以使用，只是不同层次的人的习惯不同，性能差距不是很大，

建议选择：首先是MYSQL5.5以上的版本，安装方式机器数量少的话，推荐CMAKE编译方式，这是大多数运维人员的选择，数量多的话就用二进制免安装，也是没问题的,这些是某些DBA的偏爱

# 3.2 生产场景下常规方式安装MYSQL5.5.32

## 3.2.1 CMAKE软件的安装

|  |
| --- |
| mkdir /home/oldboy/tools -p  tar xf cmake-2.8.8.tar.gz  cd cmake-2.8.8  ./configure  gmake  gmake install  cd ../ |

## 3.2.2 依赖包ncurses-devel的安装

|  |
| --- |
| yum install ncurses-devel -y |

## 3.2.3 创建mysql用户和用户组

|  |
| --- |
| groupadd mysql  useradd mysql -s /sbin/nologin -M -g mysql |

3.2.4 安装mysql数据库

|  |
| --- |
| tar zxf mysql-5.5.32.tar.gz  cd mysql-5.5.32  cmake . -DCMAKE\_INSTALL\_PREFIX=/application/mysql-5.5.32 \  -DMYSQL\_DATADIR=/application/mysql-5.5.32/data \  -DMYSQL\_UNIX\_ADDR=/application/mysql-5.5.32/tmp/mysql.sock \  -DDEFAULT\_CHARSET=utf8 \  -DDEFAULT\_COLLATION=utf8\_general\_ci \  -DEXTRA\_CHARSETS=gbk,gb2312,utf8,ascii \  -DENABLED\_LOCAL\_INFILE=ON \  -DWITH\_INNOBASE\_STORAGE\_ENGINE=1 \  -DWITH\_FEDERATED\_STORAGE\_ENGINE=1 \  -DWITH\_BLACKHOLE\_STORAGE\_ENGINE=1 \  -DWITHOUT\_EXAMPLE\_STORAGE\_ENGINE=1 \  -DWITHOUT\_PARTITION\_STORAGE\_ENGINE=1 \  -DWITH\_FAST\_MUTEXES=1 \  -DWITH\_ZLIB=bundled \  -DENABLED\_LOCAL\_INFILE=1 \  -DWITH\_READLINE=1 \  -DWITH\_EMBEDDED\_SERVER=1 \  -DWITH\_DEBUG=0  make  make install  ln -s /application/mysql-5.5.32/ /application/mysql  ll support-files/my\*cnf  2)选择配置文件  /bin/cp mysql-5.5.32/support-files/my-small.cnf /etc/my.cnf  3)配置环境变量  echo 'export PATH=/application/mysql/bin:$PATH' >>/etc/profile  tail -1 /etc/profile  echo $PATH  故障问题  http://oldboy.blog.51cto.com/2561410/1122867  4)初始化数据文件(容易出错的步骤)  mkdir -p /application/mysql/data  #建立mysql数据文件目录  chown -R mysql.mysql /application/mysql/data/  #授权mysql用户访问mysql的安装目录  chmod -R 1777 /tmp  #调整/tmp权限，否则初始化会失败  cd /application/mysql/scripts/  ./mysql\_install\_db --basedir=/application/mysql  --datadir=/application/mysql/data/ --user=mysql  cd /home/oldboy/tools/mysql-5.5.32  cp support-files/mysql.server /etc/init.d/mysqld  chmod +x /etc/init.d/mysqld  /etc/init.d/mysqld start  netstat -lntup | grep 3306 |

## 3.2.5 mysql数据库安全优化

|  |
| --- |
| mysql数据库的安全优化  mysql  mysql > select user.host from mysql.user;  mysql > delete from mysql.user where user='';  mysql > delete from mysql.user where host='www';  mysql > delete from mysql.user where host='::1';  mysql > drop database test;  mysql > SHOW DATABASES;  或者删除全部权限，添加额外管理员  mysql > delete from mysql.user;  mysql > grant all privileges on \*.\* to system@'localhost' identified by 'oldboy123' with grant option;  mysql > \q  /application/mysql//bin/mysqladmin -u root password 'oldboy123'  如果在执行上述命令出现字符集错误时，执行如下命令。  vim /etc/sysconfig/i18n  #LANG="zh\_CN.GB18030"  LANG="zh\_CN.UTF-8"  . /etc/sysconfig/i18n  mysql -uroot -poldboy123  chkconfig mysqld on  chkconfig --list mysqld |

3.3 mysql多实例介绍

## 3.3.1 什么是MYSQL多实例

简单的说，就是在一台机器上开启多个不同服务端口(3306,3307),运行多个MYSQL服务进程，这些服务进程通过不同的socket监听不同的服务端口来提供各自的服务，这些MYSQL多实例共用一套MYSQL安装程序，使用不同(也可以相同)的my.cnf配置文件，启动程序，数据文件。在提供服务时，多实例MYSQL在逻辑上看起来是各自独立的多个实例的自身是根据配置文件对应的设定值，来取得服务器的相关硬件资源多少，做个比喻，MYSQL多实例相当于房子的多个卧室一样，每个实例可以看做一个卧室，整个服务器就是一套房子，服务器的硬件资源(cpu,mem,disk)可以看做房子的卫生间，厨房，厅一样，是房子的共用资源，北漂蜗居子啊一起，肯定呀共用上述资源，这样想我们就应该明白了。

多实例补充：其实很多服务都可以有多个实例，甚至在门户网站用的很广泛，例如

nginx就可以多实例，apache,haproxy,redis,memcache，都可以多实例。

<http://edu.51cto.com/course/course_id-395.html>

3.3.2 MYSQL多实例的作用于问题

3.3.2.1 有效利用服务器资源

当单个服务器资源有剩余时，可以充分利用剩余的资源提供更多的服务。

### 3.3.3.2 节约服务器资源

### 当公司资金紧张，但是数据库又需要各自尽量独立提供服务，而且，需要主从同步等。

### 3.3.3.3 资源互相抢占问题

当某个服务实例并发很高或者慢查询时，真个实例会消耗更多的内存，CPU,磁盘IO资源，导致服务器上的其他的实例提供服务的质量下降，这就相当于大家住在同一个房子的不同卧室一样，早晨起来上班，都要刷牙，洗脸等，这样卫生间就会长期占用，其他人就要等待一样的道理。

# 3.4 MYSQL多实例生产应用场景

# 3.4.1 资金紧张型公司的选择

当公司业务访问不太大，有舍不到花钱，但又希望不同业务的数据库服务各自尽量独立的提供服务互相不受影响，而且，需要主从同步进行等技术提供备份或读写分离服务时，多实例就再好不过了，如：可以通过3台服务器部署6-9个实例，交叉做主从同步备份及读写分离，实现6-9台服务器才有的效果，这里

强调的是，所谓的尽量独立是相对的，可画图举例

## 3.4.2 并发访问不是特别大的业务

## 当公司业务访问量不太大的时候，服务器的资源基本都是浪费的，这时就很适合多实例的应用，如果对SQL语句优化做的比较好，MYSQL多实例一个很值得使用的技术，及时并发很大，赫利分配好系统资源，也不会有太大问题

3.4.3 门户网站应用MYSQL多实例场景

百度搜索引擎的数据库就是多实例，一般是从库，内存96G，跑3-4个实例，SINA网也是用的多实例，内存48G左右，门户网站使用多实例的目的是配硬件好的服务器，节省IDC机柜空间，同时，跑多实例让硬件资源不浪费。

# 3.5 MYSQL多实例常见配置方案

## 3.5.1 多实例文件部署方案

即本文主要讲解的方案，通过配置多个配置文件及多个启动程序来实现多实例的方案

这是推荐的方法。

本文的配置：

|  |
| --- |
| tree /data  /data  |--3306  | |--data <-3306实例的数据文件  | |--my.cnf <-3306实例的配置文件  | --mysql <-实例的启动文件  |--3307  | |--data <-3307实例的数据文件  | |--my.cnf <-3307实例的配置文件  | --mysql <-实例的启动文件  提示：配置文件，启动程序，数据文件分布是独立的。 |

3.5.2 单个配置文件部署方案

|  |
| --- |
| 1)my.cnf配置文件样例(mysql手册里提到的方法)  [mysqld\_multi]  mysql =/usr/bin/mysqld\_safe  mysqladmin=/usr/bin/mysqladmin  user = mysql  [mysqld1]  socket =/var/lib/mysql/mysql.sock  port = 3306  pid-file = /var/lib/mysql/mysql.pid  datadir = /var/lib/mysql/  user = mysql  [mysqld2]  socket =/mnt/data/db1/mysql.sock  port = 3302  pid-file = /mnt/data/db1/mysql.pid  datadir = /mnt/data/db1/  user = mysql  skip-name-resolve  server-id=10  default-storage-engine=innodeb  innodb\_buffer\_pool\_size=512M  innodb\_additional\_mem\_pool=10M  default\_character\_set=utf8  character\_set\_server=utf8  #read-only  relay-log-space-limit=3G  expire\_logs\_day=20  ...  2)启动方式  mysqld\_multi --config-file=/data/mysql/my\_multi.cnf start 1 2  3)缺点耦合度太高  下面我们来讲多配置文件的部署方案。  多实例MYSQL数据库的安装和前文讲解的单实例的安装过程没有任何区别，因此，读者如果有前文单实例的安装环境，那么可以直接略过单实例的安装过程  原则：降低耦合度。 |

# 3.6 安装多实例MYSQL数据库

## 3.6.1 建立MYSQL多实例的条件

(1)安装MYSQL多实例的条件

rpm -aq ncurses-devel libaio-devel openssl-devel

yum install ncurses-devel libaio-devel openssl-devel -y

rpm -aq ncurses-devel libaio-devel openssl-devel

(2)选择二进制安装方式为例

首先按照我们前面讲过的四种MYSQL的安装方式之一安装好数据库，如果是编译安装，则到make install之后截至，本文更加KISS原则，选择最简单的二进制软件为例进行讲解MYSQL多实例的安装。

## 3.6.2 建立MYSQL账号

首先以root身份登陆到linux系统中，然后执行如下命令创建MYSQL组及用户

|  |
| --- |
| groupadd mysql  useradd -s /sin/nologin -g mysql -M mysql  #useradd 命令参数简要说明  # -s /sbin/nologin 表示禁止该用户登录系统，加强安全  # -g mysql 指定属于MYSQL组  # -M 表示不创建用户家目录  #检查创建的用户  tail -1 /etc/passwd |

3.6.3 获得MYSQL软件

通过rz或sftp等命令从本地上传mysql软件包到linux服务器，名称如：mysql-5.5.32-linux2.6-x86\_64.tar.gz,当然你也可以到网上找到地址，然后到linux系统通过wget等方式下载

1)整个上传软件执行过程如下：

mkdir -p /home/oldboy/tools

cd /home/oldboy/tools/

2)查看上传后的mysql二进制软件包

ls mysql-5.5.32-linux2.6-x86\_64.tar.gz

3.6.4 安装MYSQL软件

1)解压软件包，移动到规范的软件安装目录下

|  |
| --- |
| tar xf mysql-5.5.32-linux2.6-x86\_64.tar.gz  mkdir -p /application/  mv mysql-5.5.32-linux2.6-x86\_64 /application/mysql-5.5.32 |

2)生产不带版本号的软连接/application/mysql，即mysql安装路径

|  |
| --- |
| ln -s /application/mysql-5.5.32 /application/mysql |

## 3.6.5 创建MYSQL多实例的数据目录

在企业中，采用的是/data目录作为mysql多实例的总的根目录，然后规划不同的mysql实例端口号来作为/data的下面的二级目录，不同的端口号就是不同的实例目录，以区别不同的实例，二级目录下包含MYSQL数据文件，配置文件以及启动文件的目录。

1)创建相关MYSQL多实例的目录如下

mkdir -p /data/{3306,3307}/data

tree /data

/data <-总的多实例根目录

|--3306 <-3306实例的目录

| |--data <-3306实例的数据文件目录

|--3307 <-3307实例的目录

| |--data <-3307实例的数据文件目录

提示：

1)mkdir -p /data/{3306,3307}/data 相当于 mkdir -p /data/3306/data;

mkdir-p /data/3307/data两条命令

2)如果是创建多个目录可以增加，如3308,3309,一般生产环境中2-3个实例就可以了生产硬件配置：MEM 32G 双CPU 8核，磁盘6\*600 SAS 15K 2-3个实例

## 3.6.6 创建MYSQL多实例的配置文件

mysql数据库默认为用户提供了多个配置文件模板，用户可以更加硬件配置大小来选择。

ls -l support-files/\*.cnf

提示：

1)关于MYSQL my.cnf中参数的调优，本书后面会打篇幅详解讲解

2)support-files下有mysql my.cnf的各种配置样例，懂英文的阅读下，里面的注释非常详细

3)使用cp全路径/bin/cp可实现拷贝而不出现替换的提示，即如果有重名文件会直接覆盖。

4)本例为测试安装环境，因此，选择参数配置下的my-small.cnf配置模板

上面是单实例的配置过程的默认配置文件模板，我们这里配置多实例和单实例就不同了，我们为了多实例之间彼此独立，因此，我们每一个实例建立一个配置文件，一个启动文件，对应自己的数据文件。

我们可以通过vim命令来添加 如下：

vim /data/3306/my.cnf

vim /data/3307/my.cnf

需要添加的my.cnf内容见表,(由官方的配置模板修改而成)；当然，在实际工作中我们是拿早已配置好的模板来进行修改，可以通过rz等方式上传配置模板my.cnf文件到相关目录下(上传到根目录下)

|  |
| --- |
| cd /  unzip -o data.zip  3306 3307  [client] [client]  port = 3306 port = 3307  socket = /data/3306/mysql.sock socket = /data/3307/mysql.sock  [mysql] [mysql]  no-auto-rehash no-auto-rehash  [mysqld] [mysqld]  user = mysql user = mysql  port = 3306 port = 3307  socket = /data/3306/mysql.sock socket = /data/3307/mysql.sock  basedir = /usr/local/mysql basedir = /usr/local/mysql  datadir = /data/3306/data datadir = /data/3307/data  open\_file\_limit = 1024 open\_file\_limit = 1024  back\_log = 600 back\_log = 600  .... .... |

最终完成的结果如下：

|  |
| --- |
| tree /data  /data <-总的多实例根目录  |--3306 <-3306实例的目录  | |--data <-3306实例的数据文件目录  | --my.cnf <-这个就是3306实例的配置文件  |--3307 <-3307实例的目录  | |--data <-3307实例的数据文件目录  | --my.cnf <-这个就是3307实例的配置文件 |

有关配置文件的参数内容，留个大家当作业，一定要完整注释了。

## 3.3.7 创建MYSQL多实例的启动文件

启动文件的创建，和配置我呢见创建几乎一样，当然，在实际工作中我们是拿早

配置好的模板来进行修改，可以通过rz等方式来上传配置文件模板mysql文件到相关目录

最终完成的结果如下：

|  |
| --- |
| tree /data  /data <-总的多实例根目录  |--3306 <-3306实例的目录  | |--data <-3306实例的数据文件目录  | --my.cnf <-这个就是3306实例的配置文件  | --mysql <-3306实例的启动文件  |--3307 <-3307实例的目录  | |--data <-3307实例的数据文件目录  | --my.cnf <-这个就是3307实例的配置文件  | --mysql <-3307实例的启动文件 |

多实例启动文件的启动mysql服务实质:

|  |
| --- |
| mysqld\_safe --defults-file=/data/3306/my.cnf 2>&1 >/dev/null &  mysqld\_safe --defults-file=/data/3307/my.cnf 2>&1 >/dev/null & |

多实例启动文件的停止mysql服务实质

|  |
| --- |
| mysqladmin -u root -poldboy123 -S /data/3306/mysql.sock shutdown  mysqladmin -u root -poldboy123 -S /data/3307/mysql.sock shutdown |

授权mysql用户和族管理整个多实例目录/data

|  |
| --- |
| chown -R mysql.mysql /data  find /data/ -type f -name "mysql" | xargs ls -l  或者：  find /data/ -type f -name "mysql" | xargs chmod +x |

3.3.8 配置mysql命令全局使用路径

3.3.8.1 配置全局路径意义

如果不配置MYSQL命令的全局路径，就无法直接敲MYSQL等命令，也就是只能/usr/local/mysql/bin/mysql这样敲命令

### 3.3.8.2 两种配置MYSQL路径方法

方法一：

echo "export PATH=/usr/local/mysql/bin:$PATH" >>/etc/profile

#注意，echo后是单引号，双引号是不行的，这是为什么，本文后面会讲到shell编程时会介绍

|  |
| --- |
| source /etc/profile |

方法二：

还有另外一个办法，就是把/usr/local/mysql/bin下面的命令拷贝到全局系统命令路径/usr/local/sbin也可或者做LINK

|  |
| --- |
| /bin/cp /usr/local/mysql/bin/\* /usr/local/sbin/ |

3.3.8.3 因MYSQL环境配置顺序导致的错误案例

特别强调：务必把MYSQL路径放在其他路径的前面，否则，可能导致使用的命令和安装的MYSQL不是一个而导致错误，下面是因为MYSQL路径问题导致的错误案例

http://oldboy.blog.51cto.com/2561410/1122867

3.3.9 初始化MYSQL多实例的数据库文件

|  |
| --- |
| mysql5.1.X初始化命令：  mysql\_install\_db --basedir=/application/mysql --datadir=/data/3306/data --user=mysql  mysql\_install\_db --basedir=/application/mysql  --datadir=/data/3307/data --user=mysql  mysql5.5.32初始化命令：  cd /application/mysql/scripts  ./mysql\_install\_db --basedir=/application/mysql --datadir=/data/3306/data --user=mysql  ./mysql\_install\_db --basedir=/application/mysql --datadir=/data/3307/data --user=mysql |

(2)操作过程如下。

大家还记得单实例的初始化命令:

mysql\_install\_db --user=mysql

提示：如果mysql5.5版本，即使是单实例，也无法使用mysql\_install\_db --user=mysql初始化数据库，可以指定datadir目录安装

如：mysql\_install\_db --datadir=/data/3306/data --user=mysql

(3)为什么要初始化数据库

a:初始化的主要母的就是创建基础的数据库文件，例如：生产mysql库表等

b:初始化后查看对应实例数据目录，可以看大如下文件

tree /data

需要执行结果的图示

## 3.3.10 启动MYSQL多实力数据库

|  |
| --- |
| /data/3306/mysql start  /data/3307/mysql start  netstat -lntup | grep 3306  netstat -lntup | grep 3307  或者  netstat -lntup | grep 330 |

3.3.11 MYSQL故障排除多实例数据库

如果MYSQL服务没有起来，排查办法如下

(1)如果发现没有显示MYSQL的端口，请稍微等待几秒钟在看看，MYSQL服务启动会略微有点慢

(2)如果还是不行，请查看错误日志，错误日志路径在my.cnf配置最下面

grep log-error my.cnf | tail -1

服务日志：如MYSQL错误日志一窥，如执行tail -100 /data/3306/mysql\_oldboy

3306.err检查日志报错如下

tail -100 /data/3306/mysql\_oldboy3306.err

(3)细看所有执行命令返回的屏幕输出，不要忽略关键的输出内容

(4)查看系统的/var/log/messages

(5)如果是关联服务，要同时查看相关服务的LOG

提醒：经常查看的错误日志的命令集错误日志中的完整的内容

故障1:/data/3306/mysql start启动错误如下

|  |
| --- |
| ...  /application/mysql-5.5.32/bin/mysqld: File '/data/3306/mysql-bin.index'  not found(Errcode: 13)  140925 21:01:42 [ERROR] Aborting  ... |

错误分析：没有给/data目录授权

解决办法：

|  |
| --- |
| chown -R mysql.mysql /data/ |

故障2 执行/data/3306/mysql start和/data/3307/mysql start后只有3306可以正常启动，3307无法启动

|  |
| --- |
| cat /data/3307/mysql\_oldboy3307.err |

出现如下故障现象

|  |
| --- |
| ...  141028 2:53:58 InnoDB:in this failed attempt. InnoDB only wrote those  file full of  ...  141028 2:53:58 remove old data files which contain you precious data !  ... |

|  |
| --- |
| 解决办法  使用命令 df -h 查看是否磁盘已满  可以删除以下磁盘当中没用的文件或文件包  比如：  cd /home/oldboy/tools/  rm -rf \*  >/data/3307/mysql\_oldboy3307.err  killall mysqld  cat /data/3307/mysql\_oldboy3307.err  出现如下故障现象  141028 2:57:19 InnoDB: Could not open or create data files.  ...  141028 2:53:58 InnoDB:in this failed attempt. InnoDB only wrote those  file full of  ...  141028 2:53:58 remove old data files which contain you precious data !  ...  ll /data/3307  chown -R mysql.mysql /data/  cd /data/3307  rm -rf data/  mkdir data  cd ../  chown -R mysql.mysql /data/  cd /application/mysql/scripts/  ./mysql\_install\_db --basedir=/application/mysql --datadir=/data/3307/dat  a --user=mysql  echo "web" >>/etc/hosts  /data/3307/mysql start  /data/3306/mysql start  ss -lntup | grep 330 |

停止mysql不需要密码的配置方法

|  |
| --- |
| cd /data/3306  vim mysql  <将mysql\_pwd="oldboy"改成mysql\_pwd="">  /data/3306/mysql stop  /data/3306/mysql start  mysqladmin -u root -S /data/3306/mysql.sock password 'oldboy123'  mysqladmin -u root -S /data/3307/mysql.sock password 'oldboy123'  find /data -type f -name "mysql"  授权mysql多实例服务所有启动文件mysql可执行  find /data -type f -name "mysql" -exec chmod 700 {} \;  find /data -type f -name "mysql" | xargs chown root.root {} \;  find /data -type f -name "mysql" -exec ls -l {} \;  pkill mysqld  pkill mysqld  /data/3306/mysql start  /data/3307/mysql start  ss -lntup | grep 330  登陆数据库实例的命令  mysql -S /data/3306/mysql.sock  create database d3306  quit  mysql -S /data/3307/mysql.sock  create database d3307  quit  system ls  system mysql -S /data/3306/mysql.sock |

## 3.3.12 配置MYSQL多实例数据库开机自启动

## 加入/etc/rc.local 设置为开机自启动

## 3.3.13 如何再增加一个MYSQL实例

增加一个MYSQL实例3308端口的命令过程如下

rm -rf /data/3308/data

## 3.3.14 小结

1:本地登录

2:远端登陆

远端登陆指定多实例主机中的其中一个实例，TCP/UDP端口(port)来指定所要登录的mysql实例，此端口是在mysql配置文件my.cnf中指定的

端口port <=端口是一种"逻辑连接位置"，客户端程序被分派计算机上的特使程序的一种方式

例如：mysql -uremote -p'hzcai123' -h 192.168.128.138 -P 3307

mysql -uroot -p'oldboy123' -h10.0.0.108 -P3308

# 4.1 启动与关闭MYSQL

## 4.1.1 单实例MYSQL启动与关闭方法

|  |
| --- |
| 1)常规启动关闭数据库方法(推荐)  /etc/init.d/mysqld start  mysql -uroot -poldboy123  2):查看MYSQL端口：  ss -lnt | grep 3306  3):查看MYSQL进程  ps -ef |grep mysql|grep -v grep  root 17124 1 0 04:29 pts/2 00:00:00 /bin/sh /application/mysql/bin/mysqld\_safe --defaults-file=/data/3307/my.cnf  mysql 17808 17124 0 04:29 pts/2 00:00:02 /usr/local/mysql/bin/mysqld --defaults-file=/data/3307/my.cnf --basedir=/application/mysql --datadir=/data/3307/data --plugin-dir=/usr/local/mysql/lib/plugin --user=mysql --log-error=/data/3307/mysql\_oldboy3307.err --open-files-limit=1024 --pid-file=/data/3307/mysqld.pid --socket=/data/3307/mysql.sock --port=3307  root 17875 17085 0 04:40 pts/2 00:00:00 grep mysql  4):MYSQL启动基本原理说明  /etc/init.d/mysqld是一个shell启动脚本，启动后最终会调用mysqld\_safe脚本，  最后调用mysqld服务启动mysql，如下，/etc/init.d/mysqld脚本中调用mysqld\_safe的程序  $bindir/mysqld\_safe --datadir="$datadir" --pid-file="$mysqld\_pid\_file\_path" $other\_args >/dev/null 2>&1 &  (2)初始化数据库时MYSQL系统输出给出的启动方法  mysqld\_safe --user=mysql &  提示:   1. 当找回root密码时，会经常使用mysql\_safe --user=mysql & 带参数启动 2. 我们自己开发脚本启动数据库时，可能会用到这个启动方法 3. /etc/init.d/mysqld和mysqld\_safe --user=mysql &的启动实质是一样的   (3)常规方法关闭数据库  1:关闭MYSQL命令  /etc/init.d/mysqld stop  ss -lnt | grep 3306  提示：使用编译及二进制安装MYSQL，MYSQL的启动脚本由mysql-xx/support-files  /mysql.server复制而来的  2:Mysql常规关闭数据库原理  /etc/init.d/mysqld 脚本关闭脚本  'stop)'  if test -s "$mysqld\_pid\_file\_path"  then  mysqld\_pid=`cat $mysqld\_pid\_file\_path`  if (kill -0 $mysqld\_pid 2>/dev/null)  then  echo $echo\_n "Shutting down Mysql"  kill $mysql\_pid  wait\_for\_pid removed "$mysqld\_pid" "$mysqld\_pid\_file\_path"; return\_value=$?  else  log\_failure\_msg "Mysql server process #$mysqld\_pid is not running!"  rm "$mysqld\_pid\_file\_path"  fi  if test -f "$lock\_file\_path"  then  rm -rf "$lock\_file\_path"  ....  提示：  1)用killall关闭，要直到出现mysqld:no process killed表示完成关闭操作  2)方法二的启动与关闭命令一般生产情况下不适用，特别是关闭命令。有关killall,kill,pkill等进程管理命令，请大家总结，  强调：尽量不要野蛮粗鲁杀死数据库，生产高并发环境可能会引起数据丢失野蛮粗鲁杀死数据库导致股指的企业案例  http://oldboy.blog.51.cto.com/2561410/1431161  http://oldboy.blog.51.cto.com/2561410/1431172  http://www.cnblogs.com/peida/archive/2012/12/20/2825837.html  (5)优雅关闭数据库方法  第一种mysqladmin方法  mysqladmin -uroot -poldboy123 shutdown  第二种自带的脚本  /etc/init.d/mysqld stop  第三种kill信号的方法  kill -USR2 `cat path/pid` |

4.1.2 多实例MYSQL启动与关闭方案实例

|  |
| --- |
| 启动方法：  /data/3306/mysql start  /data/3307/mysql start  关闭方法：  /data/3306/mysql start  /data/3307/mysql stop  提示：此法是双实例的使用方法，实际上就是指定不同配置文件my.cnf文件启动，具体实战见后文讲解  启动：/bin/sh ${CmdPath}/mysqld\_safe --defaults-file=/data/${port}/my.cnf 2>&1 >/dev/null &  关闭：/bin/sh ${CmdPath}/mysqladmin -u${mysql\_user} -p${mysql\_pwd} -S /data/${port}/mysql.sock shutdown |

(4)强制关闭数据库方法

|  |
| --- |
| killall mysqld  pkill mysqld  killall -9 mysqld  mysqld:no process killed  提示：  1)用killall关闭，要直到出现mysqld:no process killed表示完成关闭操作  2)方法二的启动与关闭命令一般生产情况下不适用，特别是关闭命令。  有关killall,kill,pkill等进程管理命令，请大家总结  强调：尽量不要野蛮粗鲁杀死数据库，生产高并发环境可能会引起数据丢失  野蛮粗鲁杀死数据库导致股指的企业案例  http://oldboy.blog.51.cto.com/2561410/1431161  http://oldboy.blog.51.cto.com/2561410/1431172  http://www.cnblogs.com/peida/archive/2012/12/20/2825837.html  (5)优雅关闭数据库方法  第一种mysqladmin方法  mysqladmin -uroot -poldboy123 shutdown  第二种自带的脚本  /etc/init.d/mysqld stop  第三种kill信号的方法  kill -USR2 `cat path/pid` |

## 4.1.2 多实例MYSQL启动与关闭方案实例

|  |
| --- |
| 启动方法： /data/3306/mysql start  /data/3307/mysql start  关闭方法：  /data/3306/mysql start  /data/3307/mysql stop  提示：此法是双实例的使用方法，实际上就是指定不同配置文件my.cnf文件启动，具体实战见后文讲解  启动：/bin/sh ${CmdPath}/mysqld\_safe --defaults-file=/data/${port}/my.cnf 2>&1 >/dev/null &  关闭：/bin/sh ${CmdPath}/mysqladmin -u${mysql\_user} -p${mysql\_pwd} -S /data/${port}/mysql.sock shutdown |

# 4.2 登陆Mysql方法

## 4.2.1 单实例MYSQL登陆的方法

1)mysql #刚装完系统无密码情况下登陆方式，不要密码

2)mysql -uroot #刚装完系统无密码情况登陆方式，不要密码

3)mysql -uroot -p #这里标准是DBA命令行登陆

4)mysql -uroot -p'oldboy123' #非脚本里一般不这样用，密码明文会泄漏密码

可以掩饰history功能解决登陆后默认提示符是:mysql>,这个提示符也可以改的，就像linux命令行提示符也更改MYSQL数据登陆提示符(了解的知识)方法如下：

1)强制linux不记录敏感历史命令

HISTCONTROL=ignorespace

2)情况历史记录的方法

history -c

history -d 2 #删除某一行的历史记录

cat /root/.mysql\_history #数据库的操作历史记录

1:命令行修改登录提示符<临时生效>

mysql> prompt \u@oldboy \r:\m:\s->

PROMPT set to '\u@oldboy \r:\m:\s->'

system@oldboy 12:59:03->

2:配置文件修改登录提示符

在my.cnf配置文件中[mysql]模块下添加如下内容(注意，不是[mysqld])),保存后无需重启mysql,退出当前session，冲洗登陆即可，如果在my.cnf配置文件中，可以用\\，避免转义带来的问题

[mysql]

prompt=\\u@oldboy [\\r:\\m:\\s->](\\\\r:\\m:\\s->)

## 4.2.2 多实例MYSQL登陆的方法

多实例MYSQL本地登录

mysql -uroot -poldboy123 -S /data/3306/mysql.sock

mysql -uroot -poldboy123 -S /data/3307/mysql.sock

提示：多实例通过mysql的-S命令指定不同的sock文件登陆不同的服务中

注意：多实例的远程连接无需指定sock路径

mysql -uroot -poldboy123 -h192.168.100.50 -P3307

提示：如果你使用了pkillall mysqld命令后，如果再重新使用脚本文件启动，它会出现无法启动的问题，这时候你要使用rm -rf /data/3306/mysql.sock /data/3306/\*.pid命令，然后再使用启动脚本重新启动

# 4.3 登陆MYSQL后的操作

a:登陆后默认提示符为

mysql>

b:为了防止操作失误，可以标记是测试环境，写在配置里永久生效

mysql> prompt \u@oldboy \r:\m:\s->

PROMPT set to '\u@oldboy \r:\m:\s->'

提示:prompt (\R) Change your mysql prompt.

c:登陆后的显示信息

MYSQL5.1.62版本

mysql -uroot -p

Enter password:

Welcome to the Mysql monitor Command end with; or \g. #mysql命令以分号;或者\g结束

Your MySQL connection id is 2

Server version: 5.1.62 Source distribution #版本信息

...

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

mysql>

MYSQL5.5.32版本

mysql -uroot -p'oldboy123'

登陆信息部分

# 4.4 善用MYSQL的帮助命令help

mysql中的help命令和linux命令行的man是类似的，和linux下查看那只命令的help是同名的，要想查看MYSQL中的命令语法，就需要用help，help后面接相关命令及命令组合即可，例如:help create,默认情况下MYSQL中的命令执行不区分大小写，这点和linux命令行严格区分大小写是由区别的

直接在mysql5.5中键入help:

mysql>help

信息需要补上

mysql>help show

SHOW GRANTS FOR 'oldboy'@'localhost'; #查看oldboy用户的权限

# 4.5 退出MYSQL数据库方法

mysql>quit或exit

quit (\q)Quit mysql.

exit (\q)Exit mysql. Same as quit

<=这是help帮助的结果

# 4.6 MYSQL数据库用户安全策略介绍

安装MYSQL数据库后，默认的管理员root密码为空，这很不安全。因此需要设置一个密码，其实在安装MSYQL单实例后，我们已经做了一些安全措施：例如：

a:为root设置密码

b:删除无用的mysql库内的用户账户

c:删除默认存在的test数据库

除了上面的方法，针对MYSQL数据库的用户处理，我们还有更严格的做法如下。更安全的措施例如删除root，添加新的管理员用户

1)删除所有MYSQL中的用户，包括root超级用户

mysql> delete from mysql.user;

2)增加system并提升为超级管理员，即和root等价的用户，只是名字不同

mysql>grant all privileges on \*.\* to system@'localhost' identified by 'oldboy123' with grant option;

mysql>flush privileges;

提示:最好的办法是先建立一个新的超级管理员，然后再删除root用户

## 4.6.1 为管理员root用户设置密码方法

mysqladmin -u root -password 'oldboy456' <=没有密码的用户设置密码命令

mysqladmin -u root -poldboy123 password'oldboy' -S /data/3306/mysql.sock

<=适合于多实例方式

强调：以上的命令是在linux命令行，不是mysql命令行

## 4.6.2 修改管理员root密码方法一：命令行外修改法

|  |
| --- |
| mysqladmin -u root -poldboy123 password 'oldboy' |

## 4.6.3 修改管理员root密码方法二：sql语句修改法

|  |
| --- |
| updata mysql.user SET password=PASSWORD("oldboy")WHERE user='root';  updata mysql.user SET password=PASSWORD("oldboy")WHERE user='root' and host='localhost';  flush privileges; |

提示：此方法适合密码丢失后通过--skip-grant-tables参数启动数据库后修改密码必须指定where条件

必须指定password()函数来指定更改密码

select user,host from mysql.user;

## 4.6.5 修改管理员root密码方法三

Mysql安装完毕后，root默认口令为空，需要马上修改root口令：

|  |
| --- |
| mysql> set password=password('oldboy123');  mysql>flush privileges; |

提示：此法不适合--skip-grant-tables方式修改密码,密码丢失后，

需要找回密码的环境，该方法不能使用

## 4.6.6 找回丢失mysql root用户密码

### 4.6.6.1 启动修改丢失的MYSQL单实例root密码方法

1)首先停止mysql

/etc/init.d/mysqld stop

2)使用--skip-grant-tables启动mysql,忽略授权登陆验证

mysql\_safe --skip-grant-tables --user=mysql &

mysql <=登陆时空密码

提示：在启动时加--skip-grant-tables参数，表示忽略授权表验证

3)重新修改登录的密码

updata mysql.user SET password=PASSWORD("456")WHERE user='root' and host='localhost';

flush privileges;

quit

mysqladmin -uroot -p456 shutdown

/etc/init.d/mysqld start

mysql -uroot -p456

## 4.6.7 多实例MYSQL启动修改丢失root密码方法

1)关闭mysql

killall mysqld

1. 启动时加--skip-grant-tables参数
2. 修改密码的方法

|  |
| --- |
| update mysql.user SET password=PASSWORD("456")WHERE user='root' and host='localhost';  flush privileges; |

提示：更多的密码修改方法，请参考1.6设置修改mysql root用户密码

1. 重启服务用新密码登陆

|  |
| --- |
| mysql -uroot -p456 -S /data/3306/mysql.sock |

故障1：

二进制安装故障问题：

解决办法：

|  |
| --- |
| cp /application/mysql/bin/mysqld\_safe  sed -i 's#/usr/local/mysql#/application/mysql#g' /applicaion/mysql/bin/  mysqld\_safe |

4.7 结构化查询语言

4.7.1 什么是SQL

SQL，英文全称structured query language，中文意思是结构化查询语言，它是一种对关系数据库中的数据进行定义和操作的语言方法，是大多数关系数据库管理系统所支持的工业标准结构化查询语言SQL是一种数据库查询和程序设计语言，用于存储数据以及查询，更新和管理关系数据库系统；同时也是数据库脚本文件的扩展名。结构化查询语言是高级的非过程化编程语言，允许用户在高层数据结构上工作。它不要求用户指定岁数据的存放方法，也不需要用户了解具体的数据存放方式，所以具有完全不同于底层结构的，不同数据库系统可以使用相同的结果化查询语言作为数据输入与管理的接口。结构化查询语言可以嵌套，这使它具有极大的灵活性和强大的功能，不同的数据库系统的SQL语言会有一些差别。

4.7.2 SQL的分类

SQL结构化查询语言包含6个部分

1:数据查询语言(DQL)

DQL全称Data Query Language,其语句，也称为"数据检索语言"，作用是从表中获得数据，确定数据怎样在应用程序给出，关键字SELECT是DQL(也是所有SQL)，用的最多的动词，其他DQL常用的保留子有WHERE,ORDER BY,GROUP BY和HAVING。这些

DQL保留字常与其他类型的SQL语句一起使用，具体语句例如：

|  |
| --- |
| mysql> select user,host,from mysql.user order by user;  mysql> select user,host,from mysql.user order by user asc;  mysql> select user,host,from mysql.user order by user desc; |

2:数据操作语言(DML):

DML全称Data Manipulation Language，其语句包括动词INSERT,UPDATE和DELETE。他们分别用于添加，修改和删除表中的行(数据)。也成为动作查询语言。具体语句例如：

|  |
| --- |
| mysql> delete from mysql.user where user='oldboy'; |

3:事务处理语言(TPL)

它的语句能却表被DML语句影响的表的所有行及时得以更新。TPL语句包括BEGIN

TRANSACTION,COMMIT和ROLLBACK

4：数据控制语言(DCL)

DCL全称(Data Control Language),它的语句通过GRANT和REVOKE获得许可,确定单个

用户和用户组对数据库对象的访问。某些RDBMS可用GRANT或REVOKE控制对标单个列的访问.

5：数据定义语言(DDL)

全称(Data Definition Language),其语言包括动词CREATE和DROP。在数据库中创建新表或删除表(CREATE TABLE或DROP TABLE);为表加入索引等。DDL包括许多与人数据库目录中获得数据有关的保留字。它也是动作查询的一部分。

6：指针控制语言(CCL)

全称CURSOR Control Language,它的语句，像DECLARE CURSOR,FETCH INTO和UPDATE WHERE CURRENT用于对一个或多个表单独行的操作。

小结：SQL语句最常见的分类一般就是3类

DDL-数据定义语言(CREATE,ALTER,DROP)<= 运维

DML-数据操纵语言(SELECT,INSERT,DELETE,UPDATE)

DCL-数据控制语言(GRANT,REVOKE,COMMIT,ROLLBACK)<=运维

4.8 MYSQL数据库常见管理应用

## 4.8.1 创建数据库

命令语法：create database <数据库名> #注意库名不能数字开头

在MYSQL默认字符集情况下建立数据库测试如下

a:建立一个名为oldboy\_default的数据库

mysql> create database oldboy\_default;

Query OK, 1 row affected (0.00 sec)

mysql > show database like 'oldboy%'\G;

提示：mysql默认的字符集是拉丁字符集

b:建立一个名为oldboy\_gbk的GBK字符集数据库

mysql> create database oldboy\_gbk DEFAULT CHARACTER SET gbk COLLATE gbk\_chinese\_ci

Query OK, 1 row affected(0.00 sec)

mysql> SHOW CREATE DATABASE oldboy\_gbk\G;

c:建立一个名为oldboy\_utf8的UTF8数据库

mysql> create database oldboy\_utf8 CHARACTER SET utf8 COLLATE utf8\_general\_ci

Query OK, 1 row affected (0.00 sec)

mysql > SHOW CREATE DATABASE oldboy\_utf8\G;

d:创建不同字符集的格式的数据库命令

create database oldboy #默认数据库配置，相当于创建了拉丁字符集数据库

create database oldboy\_gbk DEFAULT CHARACTER SET gbk COLLATE gbk\_chinese\_ci #创建gbk字符集数据库

create database oldboy\_utf8 DEFAULT CHARACTER SET gbk COLLATE utf8\_chinese\_ci #创建utf8字符集数据库

提示：字符集的不一致是数据库中文乱码的罪魁祸首，有关字符集的问题见下文如果编译的时候指定了特定的字符集，则以后创建对应的数据库就不需要指定字符集了

|  |
| --- |
| cmake . -DCMAKE\_INSTALL\_PREFIX=/application/mysql-5.5.32 \  -DMYSQL\_DATADIR=/application/mysql-5.5.32/data \  -DMYSQL\_UNIX\_ADDR=/application/mysql-5.5.32/tmp/mysql.sock \  -DDEFAULT\_CHARSET=utf8 \  -DDEFAULT\_COLLATION=utf8\_general\_ci \ |

企业案例：如何创建数据库

1：根据开发的程序确定字符集(建议UTF8)

2：编译时候指定字符集，例如：

-DDEFAULT\_CHARSET=utf8 \

-DDEFAULT\_COLLATION=utf8\_general\_ci \

然后创建库的时候就默认创建即可

create database oldboy

3:编译的时候没有指定字符集或者指定了和程序不同的字符集，如何解决？

指定字符集创建数据库

create database oldboy\_gbk DEFAULT CHARACTER SET gbk COLLATE gbk\_chinese\_ci #创建gbk字符集数据库

create database oldboy\_utf8 DEFAULT CHARACTER SET gbk COLLATE utf8\_chinese\_ci #创建utf8字符集数据库

数据库要支持创建库的字符集，例如：

-DEXTRA\_CHARSETS=gbk,gb2312,utf8,ascii \

## 4.8.2 显示数据库

命令：show databases;

mysql>show databases;

mysql>show databases like 'oldboy%'

mysql>show databases like '%oldboy%' #'%'为通配符，表示精确匹配

mysql>select database(); #显示当前数据库

## 4.8.3 删除数据库

命令：drop database <数据库名>

例如：删除名为oldoby的数据库

mysql> drop database test;

Query OK, 1 row affected (0.06 sec)

学习潜意识就查看帮助：

mysql>help drop database

## 4.8.4 连接数据库

命令：use <数据库名> #相当于linux下的cd切换目录的命令，use是切换数据库

例如：如果oldboy数据库的存在，尝试进入oldoby库里；

mysql> use oldboy\_gk

Database changed

## 4.8.5 查看当前的数据库

mysql> select database(); #相当于linux下的pwd

mysql> select version(); #查看当前的数据库版本

mysql> select user(); #查看当前的数据库的用户

mysql> select now(); #查看当前数据库的时间

4.8.6 当前数据库包含的表信息

切到数据库里面去查看

mysql> show tables;

Empty set (0.00 sec) #空表，因为是新库，还没建立表

mysql> show tables link 'user';

mysql> show tables from oldboy\_gbk; #查看指定库的表

mysql> show tables in oldboy gbk;

## 4.8.7 删除MYSQL系统多余账号

语法：drop user "user"@"主机域" #注意引号，可以添加单或双引号，但是不能不加

mysql> drop user'root'@'oldboy'

mysql> drop user''@'oldboy';

mysql> drop user''@'localhost'; #没有的部分就用两个单引号代替即可

注意：删除后的结果如上面返回的结果

如果drop删除不了(一般是特殊字符或大写)，可以用下面方式删除(root用户，oldoby为例)：

delete from mysql user where user='root' and host='oldboy';

flush privileges;

4.9 创建MYSQL用户及赋予用户权限

4.9.1 通过help查看grant命令帮助

1)通过在mysql中输入"help grant"得到如下帮助信息

mysql> help grant

...省略部分...

CREATE USER 'jeffery'@'localhost' IDENTIFIED BY 'mypass'

GRANT ALL ON db1.\* TO 'jeffery'@'localhost';

GRANT ALL ON db2.invoice TO 'jeffery'@'localhost';

GRANT ALL ON \*.\* TO 'jeffery'@'localhost' WITH MAX\_QUERIRS\_PER\_HOUR 90;

...省略部分...

通过查看grant的命令帮助，可以很容易的找到创建用户并授权的例子！

2)运维人眼比较常用的创建用户的方法，是用grant命令在创建用户的同时进行权限授权.具体授权例子为：

GRANT ALL ON db1.\* 'jeffery'@'localhost' IDENTIFIED BY 'mypass';

3)上述grant命令帮助里还提供了一个先用create命令创建用户，然后再用grant授权的方法，即创建用户和授权权限分开进行，例：

CRATE USER 'jeffery'@'localhost' IDENTIFIED BY 'mypass';

GRANT ALL ON db.\* TO 'jeffery'@'localhost';

以上两条命令相当于下面一条命令

GRANT ALL ON db.\* TO 'jeffery'@'localhost' IDENTIFIED BY 'mypass';

## 4.9.2 通过GRANT命令创建用户并授权

1)GRANT命令简单语法如下

GRANT ALL PRIVILEGES ON dbname.\* to username@localhost IDENTIFIED BY 'passwd'

2)列表说明如下

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| GRANT | ALLPRIVILEGES | ONdbname.\* | to username@localhost | IDENTIFIED BY'passwd' |
| 授权命令 | 对应权限 | 目标:库和表 | 用户名和客户端主机 | 用户密码 |

说明：上述命令是授权localhost主机上通过用户username管理dbname数据库的所有权限，密码为passwd.其中username,dbname,passwd可根据业务的情况修改。

3)操作案例1：创建oldboy用户，对test库具备所有的权限，允许从localhost主机登陆管理数据库，密码为oldboy123

a;实现上述操作的具体命令如下：

GRANT ALL PRIVILEGES ON test.\* TO 'oldboy'@'localhost' IDENTIFIED BY 'oldboy123' FLUSH PRIVILEGES;

b:实战演示

查看当前数据库用户情况，然后执行对应命令授权如下：

mysql> select user,host from mysql,user;

mysql> SHOW GRANTS FOR oldboy@localhost; #查看用户权限

## 4.9.3 CREATE和GRANT配合法

1)首先创建用户username及密码passwd,授权主机localhost.

|  |
| --- |
| CREATE USER 'username'@'localhost' IDENTIFIED BY 'passwd'; |

2)然后授权localhost主机上通过用户username管理dbname数据库的所有权限，无需密码

|  |
| --- |
| GRANT ALL ON dbname.\* TO 'username'@'localhost'; |

3)操作案例2：创建oldgirl用户，对test库具备所有权限，允许从localhost主机登陆管理

数据库，密码是oldboy123

4)实战演示

查看当前数据库用户情况，然后执行对用命令创建用户如下

|  |
| --- |
| mysql> select user,host from mysql.user;  mysql> create user 'oldgirl'@'localhost' identified by 'oldboy123'  mysql> grant all on test.\* to 'oldgirl'@'localhost';  mysql> SHOW GRANTS FOR oldgirl@localhost; #查看用户权限 |

## 4.9.4 授权局域网内主机远程连接数据库

根据GRANT命令语法，我们知道test@'localhost'位置为授权数据库的主机，localhost可以用域名，IP地址或IP段来替代，因此，要授权局域网内主机可以通过如下方法实现

a:百分号匹配法

|  |
| --- |
| mysql> grant all on \*.\* to 'test'@'10.0.0.%' identified by 'test123'; |

b:子网掩码匹配法

|  |
| --- |
| mysql> grant all on \*.\* to 'test'@'10.0.0.0/255.255.255.0' identified by 'test123' |

最后要记得刷新权限

mysql> flush privileges;

mysql> mysql -utest -ptest123 -h 10.0.0.7

通过MYSQL客户端连接异地数据库服务

1:本地mysql-uroot-p'oldboy123'连接数据库相当于mysql-uroot-p'oldboy123' -h localhost

2:要远程连接10.0.0.7的数据库，命令为mysql -utest -p'test123' -h 10.0.0.7,如果要能成功连接，还需要在10.0.0.7的数据库服务器上通过如下命令授权：

grant all on \*.\* test@'10.0.0.%' identified by 'test123';

提示：上述命令的意思是授权10.0.0.%，匹配这个字符串IP地址的所有主机可以连接数据库，百分号%匹配所有10.0.0.0/24内的主机。

3:通过php服务器连接MYSQL服务器的代码写法为：

<?php

$link\_id=mysql\_connect('10.0.0.7','test','test123') or mysql\_error();

if($link\_id){

echo "mysql successful by oldboy !";

}else{

echo mysql\_error();

}

?>

## 4.9.5 MYSQL用户可以授权的权限有哪些

通过实验获得ALL PRIVILEGES包括那些权限

1)先看看前面授权过的oldboy的权限

mysql> show grants for 'wpaccp'@'192.168.100.%'

mysql> REVOKE INSERT ON test.\* FROM 'oldboy'@'localhost'; #分解权限功能，收回某个权限

mysql -uroot -poldboy123 -e "show grant for oldboy@localhost;"

mysql -uroot -poldboy123 -e "show grant for oldboy@localhost;"|grep -i grant

|tail -1|tr ',' '\n'>all.txt

select \* from mysql.user\G;

提示：我们在生产环境当中，即在授权时，可以授权用户最小的满足业务需求的权限，而不一是一味的授权"ALL PRIVILEGES"

## 4.9.6 企业生产环境如何授权用户权限

1)博客，CMS等产品的数据库授权：

对于WEB连接用户授权尽量采用最小化原则，很多开源软件都是WEB界面安装，因此，在安装期间除了select,insert,update,delete 4个权限外，还需要create,drop等比较危险的权限

mysql> grant select,insert,update,delete,create,drop on blog.\* to 'blog'@'10.

0.0.%' identified by 'oldboy';

常规情况下授权select,insert,update,delete,4个权限即可，有的开源软件，例如discuz bbs，还需要create,drop等比较危险的权限。

2)生成数据库表后，要收回create,drop授权：

mysql> help revoke

mysql> REVOKE create on blog.\* FROM 'blog'@'10.0.0.%';

mysql> show grants for [blog@'10.0.0.%';](mailto:blog@'10.0.0.%';)

# 4.10 表操作

### 4.10.1 指定字符集建库

下面我们以gbk格式的oldboy库位例讲解，注意，由于我们并未特别设置数据库及客户端字符集，因此，我们先指定字符集建库

mysql> create database oldboy\_gbk DEFAULT CHARACTER SET gbk COLLATE gbk\_chinese\_ci;

mysql> show create database oldboy\G;

4.10.2 建立表

1)建表的基本命令语法：

create table <表名>(

<字段名1><类型1>,

...

<字段名n><类型n>

);

提示：其中create table是关键字，不能更改，但是大小写可以变化。

(2)建表的语句：

下面是人工写法设计的建表语句例子，表明student

create table student(

id int(4) not null,

name char(20) not null,

age tinyint(2) NOT NULL default '0'

dept varchar(16) default NUll

);

show create table student\G

第二种MYSQL生成建表语句student表的例子

create table `student` ( #CREATE TABLE 表示创建表的固定关键字，student为表名

`id` int(4) not null, #学号列，数字类型，长度为4，不为空值。

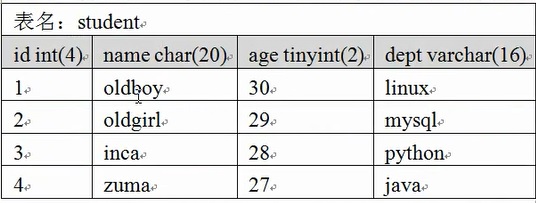
`name` char(20) not null,#名字列 定长字符类型，长度20 不为空值

`age` tinyint(2) NOT NULL default '0' #年龄列，很小的数字类型，长度为2，不为空，默认为0值

`dept` varchar(16) default NUll #系别列，变长字符类型 长度16，默认为空

)ENGINE=InnoDB DEFAULT CHARSET=latin1 #引擎和字符集，引擎默认为InnoDB,字符集，继承库的latin1

(3)student表的直观显示，可以用下表表示，字段下面的5行为表的5条记录



(4)实战演示；

执行student表建表语句

mysql> use oldboy\_gbk

mysql> create table student(id int(4) not null,name char(20) not null,

age tinyint(2) NOT NULL default '0',dept varchar(16) default NUll);

mysql> show create table student\G

需要注意的事情：MYSQL5.1和MYSQL5.5环境的默认建表语句中的引擎的不同，如果希望在建表语句里显示的指定引擎建表：

MYSQL5.1及以前默认引擎为MYISAM，MYSQL5.5以后默认引擎是InnoDB

查看表结构

mysql> desc student;

4.10.3 MYSQL表的字段类型

1)数字类型

|  |
| --- |
| 列类型 需要的储存量  TINYINT 1字节  SMALLINT 2个字节  MEDIUMINT 3个字节  INT 4个字节  INTEGER 4个字节  BIGINT 8个字节  FLOAT(x) 4 如果 X<=24或 8 如果 25<=X<=53  FLOAT 4个字节  DOUBLE 8个字节  DOUBLE 8个字节  PRECISION  REAL 8个字节  DECIMAL(M,D) M字节(D+2,如果M<D)  NUMERIC(M,D) M字节(D+2,如果M<D) |

2)日期和时间类型

|  |
| --- |
| 列类型 需要的存储量  DATE 3个字节  DATETIME 8个字节  TIMESTAMP 4个字节  TIME 3个字节  YEAR 1个字节 |

3)字符串类

|  |
| --- |
| 列类型 需要的存储量  CHAR(M) M字节，1<=M<=255  VARCHAR(M) L+1字节，L<=M和1<=M<=255  TINYBLOG,TINYTEXT L+1字节，在此L<2^8  BLOB,TEXT L+2字节，在此L<2^16  MEDIUMBLOB,MEDIUMTEXT L+3字节，在此L<2^24  LONGBLOB,LONGTEXT L+4字节，在此L<2^32  ENUM('value1','value2'...) 1或2个字节，取决于枚举值的数目(最大值65535)  SET('value1','value2'...) 1,2,3,4或8个字节，取决于集合成员的数量(最多64个成员)  1) INT[(M)]:正常大小整数类型  2) CHAR(M):定长字符串类型，当存储时，总是用空格填满右边到指定的长度  3) VARCHAR:定变长字符串类型 |

4.10.4 生产环境标准的UF8格式表结构语句

use sns;

set name gbk;

生产场景下的建表.jpg

生产场景下的建表2.jpg

4.10.5 查看建立的表结构

A:查看表结构命令

desc<表名>或者SHOW COLUMNS FROM 表名

例如：

mysql> desc test;

mysql> show columns from test;

执行结果：

4.10.6 查看已经建立的表的语句

B:查看已经建立表的语句(可以看索引及创建表的相关信息)

mysql> show create table test \G

4.10.7 为表的字段创建索引

索引就像是看书一样，如果在字段上建立了索引，那么以索引列为查询条件时可以加快查询速度，这是MYSQL优化的重要内容之一，后面课程会详细讲到

4.10.7.1 创建主键索引

查询数据库，按主键查询是最快的，每个表只能有一个主键列，但是可

以有多个普通索引列，主键列要求列的所有内容必须唯一，而索引列不要求内容必须唯一

主键类似我们在学习学习时的学号一样，班级内是唯一的，整个表的每一条记录的主键值

在表内都是唯一的，用来唯一标识一条记录，那么，我们该如何建立主键索引和普通索引呢？

首先，无论建立主键值还是普通索引，都要在表的对应列上创建，可以对单列创建索引，页可以对多列创建索引！

建立主键索引的方法：

1)在建表时，可以增加建立主键索引的语句如下

create table student(

id int(4) not null AUTO\_INCREMENT,

name char(20) not null,

age tinyint(2) not null default '0',

dept varchar(16) default NULL,

primary key(id),

KEY index\_name(name)

);

提示：

primary key (id) #主键

KEY index\_name(name)#name字段是普通索引

优化：在唯一值多的列上建立索引查询效率高

2)建表后通过alter命令增加主键索引(不建议使用)

a:主键列不能重复创建，必须先删除上面配置

mysql> alter table student drop primary key;

mysql> desc student

利用alter命令修改id列为自增主键列

mysql> alter table student change id id int primary key auto\_increment;

4.10.7.2 创建普通索引

1)在建表时，可以增加建立普通索引的语句如下

create table student(

id int(4) not null AUTO\_INCREMENT,

name char(20) not null,

age tinyint(2) NOT NULL default '0',

dept varchar(16) default NULL,

primary key(id),

KEY index\_name(name)

)

提示：KEY index\_name(name) <=name字段普通索引

优化：在唯一值多的列上建立索引查询效率高

2)建表后利用alter增加普通索引

删除建表时创建的index\_name索引

mysql> alter table student drop index index\_name

添加一个新的普通索引

mysql> alter table student add index index\_name(name);

#### 4.10.7.3 对字段的前n个字符创建普通索引

当遇到表中比较大的列时，列内容的前n个字符在所有内容中已经接近唯一时，这时可以对

的前n个字符建立索引，而无需对整个列建立索引，这样可以节省创建索引占用的系统空间

，以及降低读取和更新维护索引消耗的系统资源

对字段的前n个字符创建普通索引的语法：

create index index\_name on test(name(8)); #条件列前N个字符创建索引

操作实践

在dept系别列上，前8个字符创建索引，此列总共长度16个

mysql> create index index\_dept on student(dept(8));

mysql> desc student;

mysql> show index from student\G;

#### 4.10.7.4 为表的多个字段创建联合索引

如果查询数据的条件所多列的，我们可以为多个查询的列创建联合索引，甚至，可以为多列的前n个字符列创建联合索引，实践演示如下：

mysql> create index ind\_name\_dept on student(name,dept);

mysql> show index form student\G;

mysql> drop index ind\_name\_dept on student; #删除索引

为多列的前n个字符列创建联合索引

mysql> create index ind\_name\_dept on student(name(8),dept(10));

#为name列的前8个字符和dept列的前10个字符创建一个索引

建表语句加入联合索引：某sns产品生产正式建表语句

提示：按条件列查询数据时，联合索引是有前缀生效特征的。

index(a,b,c)仅a,ab,abc三个查询条件列可以走索引,b,bc,ac,c等无法使用索引了

#### 4.10.7.5 创建唯一非主键索引

mysql> desc student;

mysql> create unique index index\_name on student;

mysql> create unique index uni\_ind\_name on student(name);

#### 4.10.7.6 索引列的创建及生效条件

问号1:既然索引可以加快查询速度，那么就给所有的列建索引吧？

解答：索引不但占用系统空间，更新数据库时还需要维护索引数据的，因此，索引是一把

双刃剑，并不是越多越好，例如：数十个到几百行的小表上无需建立索引，更新频繁，读取比较少的业务要少建立索引。

问号2：需要在那些列上创建索引

select user,host from mysql.user where host=...,索引要创建在条件列上，而不是

select后的选择数据的列。另外我们要尽量选择在唯一值多的大表上建立索引

4.10.7.7 创建索引命令集合小结

1)创建索引相关命令集合

创建主键索引

alter table student chang id id int primary key auto\_increment;

删除主键索引

alter table student drop primary key;

创建普通索引

alter table student add index index\_dept(detp(8));

根据列的前n个字符创建索引

create index index\_dept on student(dept(8))

根据多个列创建联合查询

create index ind\_name\_dept on student(name,dept);

根据多个列的前n个字符创建联合索引

create index ind\_name\_dept on student(name(8),dept(10));

删除普通索引

alter table student drop index index\_depts;

drop index index\_dept on student;

创建唯一索引

create unique index uni\_ind\_name on student(name);

基本索引：

1:要在表的列上创建索引

2:索引会加快查询速度，但是会影响更新的速度。

3:索引不是越多越好，要在频繁查询的条件列上创建索引，因为要维护索引

4:小表或唯一值极少的列上，不建立索引，要在大表以及不同的内容多的列上创建索引。

4.10.8 往表中插入数据

1)命令语法：

insert into <表名>[(<字段名1>)][...<字段名n>])]value( 值1)[,( 值n)]

2)新建一个简单的测试表test

use oldboy\_gbk

CREATE TEBLE `test` (

`id` int(4) NOT NULL AUTO\_INCREMENT

`name` char(20) NOT NULL,

PRIMARY KEY (`id`)

)ENGINE=InnoDB DEFAULT CHARSET=latin1

3)往表中插入数据的不同的语法例子：

1:按规矩指定所有列名，并且没列都插入值

insert into test(id,name) value(1,'oldboy');

show create table test;

2:由于id为自增的，所以，可以只在name列插入数据

insert into test(name) value('oldgirl')

3:如果不指定列，就要按规矩为每列都插入恰当的值

insert into test values(3,'inca')

4:批量插入数据方法，提升效率<数据优化策略>

insert into test values(4,'zuma'),(5,'kaka'); #批量插入2条记录，提升效率

insert into test values(1,'oldboy'),(2,'oldgirl'),(3,'inca')(4,'zuma'),(5,'kaka');

5:测试完毕后，退出数据库，然后备份上述数据，留着备用

mysqldump -uroot -poldboy123 -B oldboy\_gbk >/opt/oldboy\_bak.sql

ls -l /opt/oldboy\_bak.sql

6:备份后检查备份的sql数据内容：过滤无用信息

grep -E -v "#|/\|^$|--" /opt/oldboy\_bak.sql

7:备份数据库出现的故障

mysqldump -uroot -poldboy123 -A -B oldboy\_gbk >/opt/oldboy\_bak.sql

故障分析：

因为加-A选项是备份所有的数据库，在该参数后是不加任何数据库名的，

所以如果要用-A选项，就不能在其后面添加-B oldboy\_gbk

所以在执行备份数据库命令后，一定要查看一下备份后的文件内容

8:补充强调:我们平时登陆网站发帖子，发博文，实质上都是调用WEB网站的程序连接MYS QL数据库，通过上述的insert sql语句把帖子博文数据存入数据库的。

4.11 查询数据

4.11.1 查询表的所有数据行

1)命令语法:select<字段1，字段2 ...>from<表名>where<表达式>

其中，select,from,where是不能随便改的，是关键字，支持大小写。

2)例如：查看表test中所有数据

a:进入指定库后查询

use oldboy\_gbk

select \* from test;

select id,name from test;

4.11.2 根据指定条件查询表的部分数据

1)例如：查看表test中前2行

mysql> select \* from test limit 2;

mysql> select \* from test limit 0,2;

2)指定固定条件查数据

mysql> select \* from test where id = 1;

mysql> select \* from test where name='oldgirl';

mysql> select \* from test where name='oldgirl' and id=2;

提示：字符类型的查询条件需要带引号,

and:表示两个条件的查询结果必须都有相关记录

or:表示两个查询条件只要有一个有相关记录，就可以了执行了。

3)指定固定标间范围查数据

select id,name from test where id>2 and id<4

select id,name from test where id>2 or id<4

4)其他查询功能

1：排序功能

mysql> select id,name from test order by id asc; #使用升序排列

mysql> select id,name from test order by id desc; #使用降序排列

提示，默认查询的数据结果是按升序排列的

## 4.11.3 多表查询

4.11.3.1 建立几个关联表

要实现多表连表查询，就需要有关联表及数据，表名及字段内容如下

==============学生表:student(Sno,Sname,Ssex,sage,Sdept)

drop tables student

create table student(

Sno int(10) NOT NULL COMMENT '学号',

Sname varchar(16) NOT NULL COMMENT '姓名',

Ssex char(2) NOT NULL COMMENT '性别',

sage tinyint(2) NOT NULL default '0' COMMENT '学生年龄',

Sdept varchar(16) default NULL COMMENT '学生所在的系别',

PRIMARY KEY (Sno),

key index\_Sname (Sname)

)ENGINE=lnnoDB AUTO\_INCREMENT=1 DEFAULT CHARSET=latin1;

====================课程表:Course(Cno,Cname,Ccredit)

create table course(

Cno int(10) NOT NULL COMMENT '课程表',

Cname varchar(64) NOT NULL COMMENT '课程表',

Ccredit tinyint(2) NOT NULL COMMENT '学分',

PRIMARY KEY (Cno)

)ENGINE=lnnoDB AUTO\_INCRMENT=1 DEFAULT=latin1;

===============选课表：SC(Sno,Cno,Grade)=====

create table SC(

SCid int(12) NOT NULL AUTO\_INCREMENT COMMENT '主键',

Cno int(10) NOT NULL COMMENT '课程号',

Sno int(2) NOT NULL COMMENT '学号',

Grade tinyint(2) NOT NULL COMMENT '学生成绩',

PRIMARY KEY (SCid)

)ENGINE=InnoDB DEFAULT CHARSET=latin1;

#### 4.11.3.2 往关联表中填充数据

1)学生表插入数据:

INSERT INTO student value(0001,'宏志','男',30,'计算机网络');

INSERT INTO student value(0002,'王朔','男',30,'computer application');

INSERT INTO student value(0003,'oldboy','男',28,'物流管理');

INSERT INTO student value(0004,'脉动','男',29,'computer application');

INSERT INTO student value(0004,'oldgirl','女',29,'计算机科学与技术');

INSERT INTO student value(0004,'莹莹','女',29,'护士');

2):课程表插入数据

INSERT INTO course value(1001,'Linux中高级运维',3);

INSERT INTO course value(1002,'LINUX高级架构师',5);

INSERT INTO course value(1003,'MySQL高级Dba',4);

INSERT INTO course value(1004,'Python运维开发'4);

INSERT INTO course value(1005,'Java web开发',3);

show create table course\G

3):选课表插入数据

INSERT INTO SC(Sno,Cno,Grade) values(0001,1001,4);

INSERT INTO SC(Sno,Cno,Grade) values(0001,1002,3);

INSERT INTO SC(Sno,Cno,Grade) values(0001,1003,1);

INSERT INTO SC(Sno,Cno,Grade) values(0001,1004,6);

INSERT INTO SC(Sno,Cno,Grade) values(0002,1001,3);

INSERT INTO SC(Sno,Cno,Grade) values(0002,1002,2);

INSERT INTO SC(Sno,Cno,Grade) values(0002,1003,2);

INSERT INTO SC(Sno,Cno,Grade) values(0002,1004,8);

INSERT INTO SC(Sno,Cno,Grade) values(0003,1001,4);

INSERT INTO SC(Sno,Cno,Grade) values(0003,1002,4);

INSERT INTO SC(Sno,Cno,Grade) values(0003,1003,2);

INSERT INTO SC(Sno,Cno,Grade) values(0003,1004,8);

INSERT INTO SC(Sno,Cno,Grade) values(0004,1001,1);

INSERT INTO SC(Sno,Cno,Grade) values(0004,1002,1);

INSERT INTO SC(Sno,Cno,Grade) values(0004,1003,2);

INSERT INTO SC(Sno,Cno,Grade) values(0004,1004,3);

INSERT INTO SC(Sno,Cno,Grade) values(0005,1001,5);

INSERT INTO SC(Sno,Cno,Grade) values(0005,1002,3);

INSERT INTO SC(Sno,Cno,Grade) values(0005,1003,2);

INSERT INTO SC(Sno,Cno,Grade) values(0005,1004,9);

查询SC表结果：

mysql> select \* from SC;

mysql> select student.Sno,student.Sname,course.Cname,SC.Grade from student,

course,SC where student.Sno=SC.Sno and course.Cno=SC.Cno order by Sno;

#### 4.11.3.3 使用explain查询select查询语句执行计划

即使用索引情况

mysql> show tables;

mysql> desc test;

mysql> explain select \* from test where name='oldboy'\G

mysql> create index index\_name on test(name);

mysql> explain select \* from test where name='oldboy'\G

# 4.12 修改表中数据

4.12.1 修改表中指定条件固定列的数据

1) 命令语法:update 表名 set 字段=新值,...where 条件(一定要注意条件)

2) 修改指定的行字段内容

a:查看要修改的表

mysql> select \* from test;

b:修改id为3的行的名字为gongli

mysql> update test set name='gongli' where id=3;

mysql> select \* from test;

4.12.2 修改表中所有行的数据

3)严重的案例(可能误操作导致数据丢失)

a:不带update参数更改所有表的记录

mysql> update test set name='gongli'; #如果不加update参数要十分小心，专业做法，一定要多问开发确认，如果你发给开发的语句要括号注明，防止DBA误会

b:更改了所有的记录

mysql> select \* from test;

c:用备份的数据恢复

mysql> truncate table test;

mysql> ll /opt/oldboy\_bak.sql

mysql -uroot -poldboy123 oldboy </opt/oldboy\_bak.sql

d:增量恢复数据

ll /application/mysql/data/

vim /etc/my.cnf修改以下功能选项

将(#log-bin=mysql-bin修改成log-bin=mysqlbin\_oldboy)

或者:sed -i 's@#log-bin=mysql-bin@log-bin=mysqlbin\_oldboy@' /etc/my.cnf

/etc/init.d/mysqld restart

mysql -uroot -poldboy123

mysql> use oldboy

mysql> select \* from test;

mysql> update test set name='gongli';

mysql> quit

ll /application/mysql/data

file /application/mysql/data/mysqlbin\_oldboy.000001

mysqlbinlog /application/mysql/data/mysqlbin\_oldboy.000001

mysql> insert into test values(6,'xiaozhang')

mysql> quit

ll /application/mysql/data

cp mysqlbin\_oldboy.000001 /opt/

mysqladmin -uroot -poldboy123 flush-log #二进制日志切割

ll /application/mysql/data

mysql -uroot -poldboy123 oldboy </opt/oldboy\_bak.sql

mysql -uroot -poldboy123 -e "select \* from oldboy.test";

mysqlbinlog -d oldboy mysqlbin\_oldboy.000001 >bin.sql

vi bin.sql

查找update test set name='gongli';命令，如果查找成功，就删除该语句后执行命令

mysql -uroot -poldboy123 oldboy <bin.sql

## 4.12.3 防止误操作MySQL数据库一例

http://oldboy.blog.51cto.com/2561410/1321061

alias mysql='mysql -U' #使用别名

4.13 删除表中数据

## 4.13.1 实践删除表中的数据

1)命令语法:delete from 表名 where 表达式

a:实践，例如:删除表test中编号为1的记录

mysql> delete from test where id=1;

提示：不加条件就是全部删除，也是非常危险的操作.delete from test,这里就不演示了

命令:truncate table 表名;

truncate table test; #清空表中所有内容

mysql> select \* from test; #清空前查看下.

mysql> truncate table test;

truncate table test和delete from test的区别：

1)前者的执行速度快,

delete相对于逻辑上的删除数据，按行删除,执行速度比较慢

truncate相当于清空物理文件

# 4.14 增删改表的字段

4.14.1 命令语法及默认添加演示

1)命令语法：alter table 表名 add 字段 类型 其他;

2)测试表数据

3)实践案例

例如：在表test中添加字段sex,age,qq,类型分别为char(4),int(4),varchar(15)

mysql> alter table test add sex char(4);

4.14.2 指定添加列在表里的位置

b)指定添加年龄列岛name列后面的位置

mysql> alter table test add age int(4) after name; #在name列后增加age列

mysql> desc test;

c)在第一列添加qq字段

mysql> alter table test add qq varchar(15) first;

4.14.3 生产环境多个复杂添加修改多字段信息的案例

1)增加1个字段：

ALTER TABLE `etiantian` ADD `FIRSTPHOTO\_URL` varchar(255) default NULL COMMENT

'第一张图片URL'

2)增2个字段：

ALTER TABLE `basic` ADD `adhtml\_top` varchar(1024) default NULL COMMENT '顶部

广告.html',

3)改变字段：

alter table ett\_ambiguity change ambiguity\_state ambiguity\_state tinyint

comment '状态,默认 1=正常,0=失效';

ALTER TABLE `ett\_photo`

4)修改字段类型

mysql> alter table test modify age char(4) after name;

4.14.4 rename法

1)命令语法:rename table 原表名 to 新表名

例如：在表test名字更改为oldboy

执行结果：

mysql> rename table test to test1;

mysql> show tables;

4.15 删除表

命令语法：drop table<表名>

例如：删除表名为test的表

执行结果：

mysql> show tables;

mysql> drop table test;

# 4.16 MYSQL插入中文数据乱码问题

4.16.1 MYSQL数据库插入中文数据乱码案例

a)创建一个oldboy测试数据库并查看建表语句

mysql> create database oldboy;

mysql> show create database oldboy\G;

b:在oldboy库下创建一个student表,并查看表结构和建表语句.

mysql> use oldboy

mysql> CREATE TABLE student(id int(4) NOT NULL AUTO\_INCREMENT,name char(20) NOT

NULL,PRIMARY KEY(id));

mysql> desc student;

mysql> show create table student\G

c:批量插入数据到oldboy库的student表

mysql> INSERT INTO student VALUE (1,'oldboy'),(2,'oldgirl'),(3,'inca'),(4,'zuma'),(5,'kaka');

mysql> select \* from student;

mysql> INSERT INTO student VALUES (6,'老男孩');

中文内容乱码问题出现了

为什么插入中文数据会出现乱码的问题呢

有如何解决插入中文数据乱码问题呢，带着这两个疑问，请读者跟随作者见下文

4.16.2 MYSQL命令行插入中文数据不乱码实战

1)查看建表语句，要特别注意表语句结尾的latin1

mysql> show create table student\G

2)设置MYSQL客户端的字符集和建表的字符集latin1一致

mysql> set names latin1 #设置字符集为插入数据的表的字符集，然后在插入中文语句

测试

3)继续插入一条中文数据

mysql> INSERT INTO student VALUE (7,'老男孩'); #linux命令行的字符集也要调整才行

mysql> select \* from student;

这个就是MYSQL命令行插入数据确保不乱码的方法，如果要更新的数据很多，就需要通过执行sql文件的方式更新数据了，那么，如果保证执行SQL文件也不乱码

4.16.3 执行SQL文件插入中文数据不乱码实战

1)将要更新的多个sql语句放在文本文件里如test.sql所示

mysql> system cat test.sql

set name latin1

INSERT INTO student VALUES(8,'老女孩');

提示：务必要加入set names latin1;确保插入数据不乱码

2)在MYSQL命令行通过source调用test.sql文件插入数据

mysql> use oldboy

mysql> source /root/test.sql

mysql> set names latin1;

mysql> select \* from student;

可能大家都知道了，这个步骤其实也是先执行set字符集操作,然后再执行更新语句，只不过是通过source加载文件执行的。

小结：执行DQL,DML语句之前要先执行set names系统及库表的字符集语句！

4.16.4 通过MYSQL命令加字符集参数导入数据解决乱码

1)把要更新的多个SQL语句放入文本文件中，注意，此次语句里不带set names latin1

#cat /root/test.sql;

INSERT INTO student VALUES (9,'张三');

2)通过MYSQL命令加上字符集蚕食指定latin1字符集导入test.sql语句

#mysql -uroot -p'oldboy123' --default-character-set=latin1 oldboy < /root/test.sql

3)通过"-e"参数可以在库外查看执行结果

#mysql -uroot -p'oldboy123' -e "set name latin1;select \* from oldboy.student;"

4.16.5 MYSQL插入中文不乱码5种方法小结

方法1):登陆MYSQL,先做"set names latin1;",然后再执行更新语句或执行语句文件

mysql> set name latin1;

mysql> source /root/test.sql

提示：test.sql文件格式正确

方法2):在sql文件中指定set names latin1;然后登陆mysql,通过如下命令执行

mysql> source /root/test.sql

方法3)在sql文件中指定set name latin1，然后通过mysql命令导入数据

mysql -uroot -p'oldboy123' oldboy <test.sql

mysql -uroot -p'oldboy123' oldboy -e "set name latin1;select \* from oldboy.test;"

方法4)通过指定mysql命令的字符集参数实现--default-character-set=latin1

#cat test.sql

INSERT INTO student VALUES (9,'张三');

mysql -uroot -p'oldboy123' --default-character-set=latin1 oldboy <test.sql

方法5)在配置文件里设置客户端及服务端相关参数

更改my.cnf客户端模块的参数，可以实现set names latin1的效果，并且永久生效.

[client]

default-character-set=latin1

提示：无需重启服务，退出重新登陆就生效，相当于set name lanti1;

更改服务端字符集简单的办法

2)更改my.cnf参数

[mysqld]

default-character-set=latin1 #适合5.1及以前版本

character-set-server=lantin1 #适合5.5

不乱码的思想:linux，客户端，服务端，库，表，程序的字符集统一

中英文混合环境建议使用utf8

linux字符集的配置

cat /etc/sysconfig/i18n

LANG="zh\_CN.utf8"

客户端：

set names latin1 #临时生效

更改my.cnf客户端模块的参数，可以实现set names latin1的效果，并且永久生效.

[client] #永久生效

default-character-set=latin1

提示：无需重启服务，退出重新登陆就生效，相当于set name lanti1;

服务器端

2)更改my.cnf参数

[mysqld]

default-character-set=utf8#适合5.1及以前版本

character-set-server=utf8 #适合5.5库：

create database oldboy\_utf8 DEFAULT CHARACTER SET UTF8 COLLATE utf8\_general\_ci;

show character set;

CREATE TABLE student (

id int(4) NOT NULL AUTO\_INCREMENT,

name char(20) NOT NULL,

PRIMARY KEY (id)

)ENGINE=lnnoDB AUTO\_INCREMENT=10 DEFAULT CHARSET=utf8

程序

简体UTF8

http://download.comsenz.com/DiscuzX/3.2/Discuz X3.2 SC UTF8.zip

# 4.17 MYSQL数据库字符集知识

4.17.1 MYSQL数据库字符集介绍

简单地说，字符集就是一套文字符号及其编码，比较规则的集合，第一个计算机字符集ASC2!,MYSQL数据库字符集包括字符集(CHARACTER)和校对规则(COLLATION)两个概念。其中，字符集是用来定义MYSQL数据字符串的存储方式。而校队规则则是定义比较字符串的方式,在上面命令查看已经建立的oldboy库的语句，CHARACTER SET latin1即为数据库字符集，而COLLATE latin1\_swedish\_ci为校队规则，有关字符集介绍详细内容参看mysql手册，4.17.1 MYSQL数据库常见字符集介绍

在互联网环境中，使用mysql时常用的字符集有

|  |
| --- |
| 常用字符集 长度 说明  GBK 2 不是国际标准  UTF-8 3 中英文混合环境，建议使用该字符集  latin1 1 MYSQL的默认字符集  utf8mb4 4 UTF-8 Unicode 使用在移动互联网 |

4.17.2 MYSQL如何选择合适的字符集

1)如果处理各种各样的文字，发布到不同语言国家地区，应选Uncode字符集，对MYSQL来说就是UTF-8(每个汉子三个字节),如果应用需处理英文，仅仅有少量汉子UTF-8更好。

2)如只需支持中文，并且数据量很大，性能要求也很高，可选(定长 每个汉字占双字节)英文也占双字节，如果需大量运算，比较排序等，定长字符集，更快，性能高。

3)处理移动互联网业务，可能需要使用utf8mb4字符集

## 4.17.3 查看当前MYSQL系统支持的字符集

MYSQL可支持多种字符集，同一台机器，库或表的不同字段多可以指定不同的字符集。查看MYSQL支持的字符集，更详细内容参考MYSQL手册

mysql> mysql -uroot -p'oldboy123' -e "SHOW CHARCTER SET;"

其中最常用的字符集为如下4种

mysql> mysql -uroot -p'oldboy123' -e "SHOW CHARCTER SET;"|egrep "gbk|utf8|latin1"|awk '{print $0}'

查看MYSQL当前的字符集设置情况

mysql> show variables like 'character\_set%'

提示：默认情况下character\_set\_client,character\_set\_connection,character\_set\_result三者的字符集和系统字符集一致。即为cat /etc/sysconfig/i18n

# 4.18 mysql插入中文数据不乱码深度剖析

4.18.1 MYSQL数据库默认字符集是什么

|  |
| --- |
| mysql> show variables like 'character\_set%';  不同字符集参数的意义  参数名 值  character\_set\_client latin1 #客户端字符集  character\_set\_connection latin1 #连接字符集  character\_set\_database latin1 #数据库字符集，配置文件指定或建库建表指定  character\_set\_result latin1 #返回结果字符集  character\_set\_server latin1 #服务器字符集，配置文件指定或建库建表指定更改linux系统字符集变量后，查看MYSQL中字符集的变化  echo $LANG  mysql -uroot -p'oldboy123' -e "show variables like 'character\_set%';"  我们发现character\_set\_client,character\_set\_connection,character\_set\_results三者的字符集和系统的字符集一致，都被改成了utf8了 |

4.18.2 执行set names latin1到底做了什么

无论linux系统字符集是gb2312还是utf8,默认情况下插入的数据都是乱码

1)此时查看数据就是乱码

mysql> use oldboy;

mysql> select \* from student;

mysql> show create database oldboy\G;

mysql> show create table student\G;

mysql> set names latin1;

mysql> select \* from student

提示：set name latin1 就是将character\_set\_client,character\_set\_connection,character\_set\_result三者的字符集的值修改成了latin1,这三个值默认情况下要和linux系统字符集一致但是当在mysql中执行set names字符集操作后，三者都会改变为设置的字符集,但是命令修改都是临时生效的

4.18.3 mysql命令参数--default-character-set=latin1在做什么

1)先查看下mysql的字符集

mysql -uroot -p'oldboy123' -e "show variables like 'character\_set%';"

2)登陆到mysql，注意，是带了--default-character-set=latin1

mysql -uroot -p'oldboy123' --default-character-set=latin1

3)此时在查看MYSQL的字符集

show variables like 'character\_set%';

# 4.19 确保MYSQL数据库插入数据不乱码解决方案

4.19.1 统一MYSQL数据库客户及服务端字符集

1)通常，MYSQL数据库的下面几个字符集(客户端和服务端)统一成一个字符集，才能确保插入的中文数据可以正确输出，即mysql> show variables like 'character\_set%';结果中的字符集设置尽量统一，当然，linux系统的字符集也要尽可能和数据库字符集统一。

2)show variables like 'character\_set%';命令的结果输出如下

参数名 值

character\_set\_client latin1 #客户端字符集

character\_set\_connection latin1 #连接字符集

character\_set\_database latin1 #数据库字符集，配置文件指定或建库建表指定

character\_set\_result latin1 #返回结果字符集

character\_set\_server latin1 #服务器字符集，配置文件指定或建库建表指定其中,第一个，第二个，第四个这三个参数默认情况采用linux系统字符集设置，人工登陆数据库执行set name latin1，以及MYSQL指定字符集登陆操作，都是改变了MYSQL客户端的client,connection,results 3个参数的字符集为latin1，从而解决了插入中文乱码的

问题，这个操作也可以通过更改my.cnf配置文件客户端模块的参数来实现，并且永久生效

3)通过修改my.cnf实现修改MYSQL客户端字符集，配置方法如下。

[client]

default-character-set=latin1

提示：无需重启服务，退出重新登陆就生效，此参数相当于，登陆后执行 set names latin1;

修改过程：

a:修改my.cnf配置文件

sed -n '18,19p' /etc/my.cnf

[client]

default-character-set=latin1

b:重启数据库（无需重启服务，退出重新登陆就生效）

/etc/init.d/mysqld restart

c:查看字符集设置

mysql -uroot -p'oldboy123' -e "show variables like 'character\_set%';

4.19.2 更改MYSQL服务端字符集

1)按如下要求更改my.cnf参数

[mysqld]

default-character-set=latin1 #适合5.1及以前版本

character\_set\_server=latin1 #适合5.5

重启数据库

/etc/init.d/mysqld restart

2)修改前查看数据当前字符集

mysql> show variables like 'character\_set%'

## 4.19.3 统一MYSQL数据库客户机服务端字符集)总结

1)客户端字符集设置为"set names utf8;",这样可以确保插入后的中文，不出现乱码，但是对执行set names utf8;前插入的中文无效，此命令临时生效

2)和设置客户端字符集"set names utf8"命令有相同作用的方法还有，mysql命令指定utf8字符集参数登陆，以及在my.cnf里更改参数实现

3)在MYSQL的my.cnf配置文件里[client]模块下添加字符集配置，生效时，相当于命令行"set names utf8;"的效果，由于更改的是客户端，连接和返回结果3个字符集，因此无需重启服务就生效

4)在MYSQL的my.cnf配置文件里[mysqld]模块下添加字符集配置，生效后，创建数据库和表默认都是这个设置的字符集，MYSQL5.5和5.1服务端字符集参数有变化，具体为character-set-server=utf8参数适合5.5,default-character-set=utf8参数适合5.1及以前版本

## 4.19.4 彻底解决MYSQL数据库插入中文乱码的方案

切记：字符集的不一致是数据库乱码的罪魁祸首

1)确保以下(客户端和服务端)字符集是一致，当然，字符集的选项可以有多种，未必就是gbk，还可以是utf8和latin1等

2)如何修改上述参数内容达到字符集一致呢？

a:修改下面客户端字符集的方法

|  |
| --- |
| character\_set\_client latin1 #客户端字符集  character\_set\_connection latin1 #连接字符集  character\_set\_result latin1 #返回结果字符集  临时方法 |

mysql> set names gbk;

永久方法，建议使用

在my.cnf里增加如下参数

[client]

default-character-set=gbk

b)修改下面服务端字符集的方法

character\_set\_database latin1

#数据库字符集，配置文件指定或建库建表指定

character\_set\_server latin1

#服务器字符集，配置文件指定或建库建表指定

my.cnf里修改

[mysqld]

default-character-set=gbk #适合5.1及以前版本

character-set-server=gbk #适合5.5

3)建库建表的时候要指定和上述设置的字符集相同的字符集，以gbk字符集为例

|  |
| --- |
| create database oldboy\_utf8 DEFAULT CHARACTER SET gbk COLLATE gbk\_general\_ci;  CREATE TABLE student (  id int(4) NOT NULL AUTO\_INCREMENT,  name char(20) NOT NULL,  PRIMARY KEY (id)  )ENGINE=lnnoDB AUTO\_INCREMENT=10 DEFAULT CHARSET=gbk |

# 4.20 如何更改生产MYSQL数据库库表的字符集

4.20.1对于已经存在表记录的字符集的修改

将模拟latin1字符集的数据库修改成GBK字符集的实际过程

1)导出表结构

mysqldump -uroot -p --default-character-set=latin1 -d oldboy>oldboy1.sql

#表示以GBK字符集进行连接，-d只导表结构

1. 编辑oldboy1.sql将latin1修改成GBK

3)确保数据库不再更新，导出所有数据

mysqldump -uroot -p --quick --no-create-info --extended-insert --default-character-set=latin1 oldboy >oldboy2.sql

参数说明：

--quick:用于存储大的表，强制mysqldump从服务器一次一行的检索数据而不是检索所有行，并输出前CACHE到内存中

--no-create-info:不创建CREATE TEBLE语句

--extended-insert:使用包括几个VALUES列表的多行INSERT语法，这样文件更小，IO也小，导入数据时会非常快

--default-character-set=latin1 #按照原有字符集导出数据，这样导出的文件中，所有中文都是可见，不会保存成乱码

4)打开oldboy2.sql将set names latin1 修改成 set names gbk;(或者修改系统的客户端和服务器端)

5)建库

create database oldboy default charset gbk;

6)创建表，执行oldboy1.sql

mysql -uroot -p oldboy<oldboy1.sql

7)导入数据

mysql -uroot -p oldboy<oldboy2.sql

提示：选择目标字符集时，要注意最好大于等于源字符集(字库更大)，否则，可能会丢失不被支持的数据

总结：latin1->utf8

1:建库及建表的语句导出，sed批量修改为utf8

2:导出所有MYSQL数据

3:修改MYSQL服务端和客户端编码为utf8

4:删除原有库表级别数据

5:导入建库及建表的语句

6:导入MYSQL的所有数据

# 5.1 mysql 数据库的备份与恢复

5.1.1 备份数据的意义

将此有朋友问，运维工作到底是什么工作，到底是什么，其实我个人认为，运维工作就两件事情：第一个是公司的数据的完整性和保密性，第二个就是网站7\*24小时提供服务,那么，对数据丢失一部分和网站7\*24小时提供服务那个更重要呢？ 我们知道都很重要，只是说相比哪个更重要

这个具体要看业务和公司的性质了

例如：银行，金融行业，数据最重要，一条都不能丢，可能宕机停机影响就没那么大,再例如：百度搜索，腾讯QQ聊天记录丢失几万条数据，都不算啥

对于数据来讲，数据最核心的就是数据库数据，当然也包括其他数据，本章主要讲解的是数据库的数据

对于绝大多数企业来讲，失去了数据就失去了商机，失去了客户，公司倒闭,因此，数据库的备份与恢复就显得十分重要了

举例：一个人把正式当测试导致损失数万元的故事

那么，我们要不要保护好数据？

5.1.2 备份单个数据库练习多种参数使用

mysql数据库自带了一个很好用的备份命令，就是mysqldump，他的基本使用如下：

基本语法：mysqldump -u 用户名 -p 数据库名> 备份的文件名

范例1:备份名字为oldboy的库

a:查看备份前的数据

mysql -uroot -poldbo123 -e "use oldboy;show tables;select \* from student;"

b:执行备份的命令

mysqldump -uroot -p'oldboy123' oldboy >/opt/mysql\_bak.sql

c:检查备份的结果

egrep "#|\\*|--|^$" /opt/mysql.bak.sql

提示：如果有乱码，是因为导出时的格式没加字符集，一般恢复到数据库里会正常，只是在系统外看不正常而已，另外INSERT是批量插入的方式，这样的恢复时效率很高。根据查看的结果，我们已经看到了已经备份的表结构语句以及插入的数据整合的SQL语句，但是中文数据乱码了

范例2：设置字符集参数备份解决乱码问题

a:查看备份前数据库客户端及服务端的字符集设置

mysql -u -poldboy123 -e "show variables like '%character%';"

b:指定对应的字符集备份，这里为--default-character-set=latin1

mysqldump -uroot -p'oldboy123' --default-character-set=latin1 oldboy>/opt/mysql\_

bak.sql

egrep -v "#|\\*|--|^$" /opt/myql\_bak.sql

恢复数据到指定的数据库的表中

mysql -uroot -poldboy123 -e "use oldboy;drop table student;"

mysql -uroot -p'oldboy123' oldboy </opt/mysql\_bak.sql

mysql -uroot -poldboy123 -e "select \* from oldboy.student;"

范例3 备份时加-B参数

mysqldump -uroot -p'oldboy123' -B oldboy>/opt/mysql\_bak\_B.sql

cd /opt/

diff mysql\_bak.sql mysql\_bak\_B.sql

提示：直观看，-B的作用是增加创建数据库和连接数据库的命令了。

即如下两条语句：

CREATE DATABASE /\*!32312 IF NOT EXISTS\* / `oldboy` /\*!40100 DEFAULT CHARACTER

SET latin1 \*/;

USE `oldboy`;

利用上述加 -B的备份进行恢复测试：

mysql -uroot -p'oldboy123' </opt/mysql\_bak\_B.sql

mysql -uroot -poldboy123 -e "select \* from oldboy.student;"

4)优化备份文件大小减少输出注释(debug调试)

利用mysqldump的--compact参数优化下备份结果

cat /opt/mysql\_bak.sql

--compact参数，该参数在测试时用的比较多，它可优化输出内容的大小，让容量更少，

适合调试。

该选项使的输出内容更加简洁，不包括默认选项中的各种注释，有如下几个参数的功能

--skip-add-drop-table -no-set-names --skip-disable-keys --skip-add-locks

范例5：指定压缩命令压缩备份的MYSQL数据

mysqldump -uroot -p'oldboy123' --default-character-set=latin1 -B oldboy|gzip >

/opt/myql\_bak\_B.sql.gz

ls -l /opt/mysql\_bak\_B.sql.gz

压缩效率将近3倍

通过以上的例子可以得到什么结论

1)导出数据用-B参数

2)用gzip对备份的数据压缩

## 5.1.3 mysqldump的工作原理

利用mysqldump命令备份数据的过程，实际上就是把数据从MYSQL库里以逻辑的SQL语句的形式直接输出或者生成备份的文件的过程，对，就是这么简单

可否举一反三，利用这个备份文件恢复的时候额原理

备份的数据过滤注释，都是SQL语句，结果如下：

egrep -v "#|\\*|--|^$" /opt/mysql\_bak.sql

## 5.1.4 备份多个库及多个参数练习

a:操作结果

|  |
| --- |
| mysqldump -uroot -p'oldboy123' -B oldboy oldboy\_gbk|gzip >/opt/all\_bak.sql.gz  ls -l /opt/all\_bak.sql.gz  b:-B参数说明  提示：-B参数是关键，表示接多个库，并且增加use db,和create database db的信息  生产环境常用  参数说明：该参数用于导出若干个数据库，在备份结果中会加入USE db\_name和CREATE DATABASE `db\_name`;  -B后的参数都将被作为数据库名。该参数比较常用，当-B后的数据库列全时同-A参数功能相似，  -A参数的说明 |

## 5.1.5 如何做分库备份

分库备份实际上就是执行一个备份语句备份一个库，如果数据库里有多个库，就执行多条相同的备份单个库的备份语句就可以备份多个库了，注意每个库都可以对应备份的库作为库名，结尾加.sql。备份多个库的命令如下

|  |
| --- |
| mysqldump -uroot -p'oldboy123' -B oldboy ...  mysqldump -uroot -p'oldboy123 -B oldboy\_gbk ... |

将上述命令放入一个脚本里就是脚本分库备份了，当然这样是很土的备份脚本了，更好的脚本见分库分表备份视频，我们这里学习SHELL脚本编程时，用来实现这个分库分表备份脚本

|  |
| --- |
| <http://edu.51cto.com/course/course_id-808.html>  for dbname in `mysql -uroot -p 'oldboy123' -e "show database;" | grep -Evi "database|infor|perfor"`; do  mysqldump -uroot -p'oldboy123 -B $dbname >${dbname}\_$(Date +%F).sql  done  mysql -uroot -p 'oldboy123' -e "show database;" | grep -Evi "database|infor|perfor"|sed 's#^#mysqldump -uroot -p'oldboy123' -B#g'  分库备份  mkdir /opt/bak  mysql -uroot -p 'oldboy123' -e "show database;" | grep -Evi "database|infor|perfor"|sed 's#^([a-z].\*$)#mysqldump -uroot -p'oldboy123' --events -B \1|gzip >/opt/bak/\1.sql.gz#g'|bash |

分库备份的意义何在？

有时一个企业的数据库里会有多个库，例如(www,bbs,blog)，但是出问题时，可能是某一个库，如果在备份时把所有的库都备份成一个数据文件的话，恢复某个数据时就比较麻烦了。

## 5.1.6 备份单个表

|  |
| --- |
| 语法：mysqldump -u 用户名 -p 数据库名 表名> 备份的文件名  mysql> mysqldump -uroot -p'oldboy123' oldboy student>/opt/table1.sql提示：不能加-B参数，因为库oldboy后面就是oldboy表  egrep -v "#|\\*|--|^$" /opt/tables.sql |

# 5.1.7 备份多个表

语法:mysqldump -u 用户名 -p 数据库名 表名1 表名2> 备份的文件名

操作结果

|  |
| --- |
| mysqldump -uroot -p'oldboy123' oldboy course student >/opt/bak/two\_tables.sql |

企业需求：一个库里有大表也有小表，有时候可能需要只恢复某一个小表，上述的多表备份文件很难拆开，就会像没有分库哪像导致恢复某一个小表很麻烦。

那么又如何进行分表备份呢？如下，和分库的思想一样，没执行一条语句备份一个表，生成不同的数据文件即可

|  |
| --- |
| mysqldump -uroot -p'oldboy123' oldboy test >oldboy\_test.sql  mysqldump -uroot -p'oldboy123' oldboy student >oldboy\_student.sql |

将上述命令放入一个脚本里就是脚本分库备份了，当然这样是很土的备份脚本了，更好的

脚本见分库分表备份视频，我们这里学习SHELL脚本编程时，用来实现这个分库分表备份脚本

<http://edu.51cto.com/course/course_id-808.html>

|  |
| --- |
| for dbname in `mysql -uroot -p 'oldboy123' -e "show database;" | grep -Evi "database|infor|perfor"`; do  for table in `show tables from $dbname;|sed '1d'`  mysqldump -uroot -p'oldboy123' $dbname $table > ${dbname}\_${table}\_$(Date +%F).sql  done  done |

分表备份缺点：文件多，很碎

1：备一个完整备份，在做一个分库分表备份

2：脚本批量恢复多个SQL文件

## 5.1.8 备份数据库表结构(不包含数据)

利用mysqldump -d 参数只备份表的结构，例如：备份oldboy库的所有表的结构

|  |
| --- |
| mysqldump -uroot -p'oldboy123' -B -d oldboy student >/opt/t.sql  mysqldump -uroot -p'oldboy123' --compact -t oldboy student >/opt/t.sql |

#只备份表数据。

|  |
| --- |
| mysqldump -uroot -poldboy123 -A -B --events|gzip >/opt/a.sql.gzmysqldump -uroot -poldboy123 -S /data/3306/mysql.sock -A -B --events|gzip >/server/backup/oldboy2.sql.gz  mysqldump -uroot -poldboy123 -S /data/3306/mysql.sock -A -B|gzip >/server/backup/oldboy2.sql.gz  mysqldump -uroot -poldboy123 -S /data/3306/mysql.sock -A -B >/opt/alltable.sql |

## 5.1.9 刷新binlog的数据

|  |
| --- |
| grep "log-bin=" /etc/my.cnf  sed -i 's@#log-bin=mysql-bin@log-bin=mysql-bin@' /etc/my.cnf  /etc/init.d/mysqld restart  mysqldump -uroot -p'oldboy123' -A -B -F --events|gzip >/opt/a.sql.gz  ll /application/mysql/data/  mysqldump -uroot -poldboy123 -S /data/3306/mysql.sock -A -B -F --events|gzip >/server/backup/oldboy2.sql.gz  mysqldump -uroot -p 'oldboy123' --master-data=1 --commpact oldboy  mysqldump -uroot -poldboy123 -S /data/3306/mysql.sock -A -B -F --master-data=2 --events|gzip >/server/backup/oldboy2.sql.gz  mysqldump -uroot -poldboy123 -S /data/3306/mysql.sock -A -B -t --events|gzip >/server/backup/oldboy2.sql.gz #备份整个数据库的表数据，不备份表结构提示：--master-data参数：能够显示出binlog的位置 |

5.1.10 MYSQL的关键参数

|  |
| --- |
| 关键参数:mysqldump --help  1) -B:指定多个库，增加建库语句和use语句  2) --compact:去掉注释，适合调试输出，生产场景下不使用  3) -A:备份所有的库  4) -F:刷新binlog日志  5) -x,--lock-all-table：锁表参数，一般使用在备份库操作当中  6)--master-data:如果该值为"1",表示记录并显示binlog的位置，为"2"表示不记录binlog 的位置  7) -l,--lock-table:只读模式  8) -d:只备份表的结构  9) -t:只备份表数据  10) --single-transaction:适合innodb事务数据库备份  InnoDB表在备份时，通常启用选项 --single-transaction来保证备份，实际上它的工作原理是设定本次会话的隔离级别为:REPEATABLE READ,以确保本次会话(dump)时，不会看到其他会话已经提交的数据 |

<http://imysql.cn/2008_10_24_deep_into_mysqldump_options>

更多见总结文章

1)mysql的myisam引擎的备份策略

|  |
| --- |
| mysqldump -uroot -poldboy123 -A -B -F --master-data=2 -x --events|gzip >/opt/all.sql.gz |

2)mysql的Innodb引擎的备份策略

|  |
| --- |
| mysqldump -uroot -poldboy123 -A -B -F --master-data=2 --events --single-transaction|gzip >/opt/all.sql.gz |

提示：如果是混合引擎，以myisam引擎备份策略为主

# 5.2 恢复数据库实践

## 5.2.1 利用source命令恢复数据库

|  |
| --- |
| 进入mysql数据库控制台，  mysql -uroot -poldboy123  mysql> use oldboy  然后使用source oldboy\_db.sql #这个文件是系统路径，默认是登陆MYSQL前的系统路径  mysql -uroot -p'oldboy123'  mysql> drop database oldboy  mysql> show databases;  mysql> source /opt/mysql\_bak\_B.sql  mysql> show databases;  mysql> select \* from oldboy.student;  mysql> set names latin1  提示：如果备份文件是gzip压缩过的，需要使用gzip -d 命令来解压该文件  gzip -d mysql\_bak.sql.gz  2)批量恢复脚本操作(分表恢复)  mysql -uroot -poldboy -e "show databases;"  gzip -d \*  for dbname in `ls \*.sql|sed 's#\_bak.sql##g'`;do  mysql -uroot -poldboy123 ${dbname}\_bak.sql  done |

5.2.2 其他知识

|  |
| --- |
| mysql -uroot -poldboy123 -e "show processlist;"  mysql -uroot -poldboy123 -e "show full processlist;"  mysql -uroot -poldboy123 -e "show variables;" #查询MYSQL的变量  mysql -uroot -poldboy123 -e "show variables;"|grep log\_bin #查看二进制日志是否开启  mysql -uroot -poldboy123 -e "show status" #查看MYSQL数据库的状态  mysql -uroot -poldboy123 -e "show gloabal status" #查看MYSQL数据库的全局状态  修改全局变量的案例  mysql> show variables like 'key\_buffer%';  mysql> set global key\_buffer\_size=1024\*1024\*32;  上述修改全局变量的值后，当数据库重启后，就不会不生效了  修改全局变量永久生效的案例  sed -i s'#key\_buffer\_size = .\*#key\_buffer\_size = 32K#g' /etc/my.cnf  mysql> show variables like 'key\_buffer%';  mysql> set global key\_buffer\_size=1024\*1024\*32;  命令小结：  show status #查看整个数据库运行状态信息  show global status: #查看整个数据库运行状态信息，很重要，要分析，并做好监控  show processlist；查看正在执行的SQL语句，  show full processlist；查看正在执行的完整SQL语句  set global key\_buffer\_size = 32777218;不重启数据库调整数据库参数，直接生效，重启后不丢失  show variables; 查看数据库的参数信息，例如：my.cnf里参数的生效情况 |

# 5.3 mysql生产环境案例

5.3.1 mysql数据库批量插入数据shell脚本实现

|  |
| --- |
| #!/bin/bash  mysqlCMD="mysql -uroot -poldboy123 -S /data/3306/mysql.sock"  status=`$mysqlCMD -e "use oldboy;show tables;"|grep student|wc -l`  insert(){  if [ $status -ne 0 ]; then  for i in {1..10}; do  $mysqlCMD -e "use oldboy;INSERT INTO student(name) VALUES('oldboy${i}');"  done  else  $mysqlCMD -e "USE oldboy;CREATE TABLE student (id int(4) NOT NULL AUTO\_INCREMENT,name char(20) NOT NULL,PRIMARY KEY (id));"  for i in {1..10}; do  $mysqlCMD -e "use oldboy;INSERT INTO student(name) VALUES('oldboy${i}');"  done  fi  $mysqlCMD -e "select \* from oldboy.student"  }  Main(){  insert  }  Main |

## 5.3.2 mysqlbinlog命令的详解和实战

mysql的binlog日志文件的作用：用来对记录MYSQL内部增删改查等对MYSQL数据库有更新的内容的记录

mysqlbinlog mysql-bin.000010

mysqlbinlog mysql-bin.000010 >all.sql

binlog的拆库操作

mysqlbinlog -d oldboy mysql-bin.000020 >oldboy.sql

less oldboy.sql

vimdiff oldboy.sql all.sql

-d:截取指定库的binlog

指定时间和位置点恢复的操作

1：按照指定位置点恢复

mysqlbinlog mysql-bin.000020 --start-position=365 --stop-position=456 -r pos.sql

输出：初始化位置365 结束位置为456的所有binlog日志保存到pos.sql

less pos.sql

source pos.sql #指定位置点的恢复操作

注意：结尾的日志点比较特殊，不会被包含，即输出1312pos以前的binlog,为支点信息一般要实际存在，不能乱指

2：基于时间点的恢复操作

1)基于开始时间和结束时间的恢复操作

mysqlbinlog mysql-bin.000020 --start-datetime='2014-10-16 17:14:15'--stop-datetime='2014-10-16 17:15:15' -r pos1.sql

上面语句将显示开始时间到结束时间的binlog，并输出到pos1.sql

2)基于开始时间到文件结束

mysqlbinlog mysql-bin.000020 --start-datetime='2014-10-16 17:14:15' -d oldboy -r pos2.sql

上面语句只显示从开始时间开始到日志结尾的oldboy数据库的binlog，并输出到pos2.sql

3)从文件开头到指定结束时间

mysqlbinlog mysql-bin.000020 --stop-datetime='2014-10-16 17:15:15' -d oldboy -s

上面语句只显示从日志开头到这个结束时间之间的oldboy数据库的binlog，并输出总结4个命令：mysqldump,mysql,mysqlbinlog,mysqladmin.

# 6.1 MYSQL主从复制

## 6.1.1 普通的文件的数据同步

1)NFS网络文件共享可以同步存储数据

2)samba共享数据 http://taokey.blog.51cto.com/4633273/1203553

3)定义任务或守护进程结合rsync.scp

4)inotify+rsync触发式实时数据同步。

5)ftp数据同步

6)ssh key+scp/rsync.

7)SVN版本管理

8)rsync,sersync,inotify,union(双向同步),csync2(多向同步)

mysql的主从同步方案，和上述的2,3点是类似的，只不过MYSQL无需借助第三方工具，而是其自带的同步复制功能，另外一点，MSYQL的主从同步并不是磁盘上文件直接同步。replication

6.2 MYSQL主从复制

## 6.2.1 MYSQL主从复制介绍

MYSQL支持单向，双向，链式级联，实时，异步复制。在复制过程中，一台服务器充当主服务器(MASTER),而一个或多个其他的服务器充当从服务器(SLAVE)

复制可以单向：M===>S,也可以是双向 M<===>M,当然也可以多M环状同步等。如果设置了链式级联复制，那么，从(slave)服务器本身除了充当从服务器外，业会同时

充当其下面从服务器的主服务器

链式级联复制类似A-->B--->C--->D的复制形式

下面给大家展示下各种同步的模样，引导学生一起画图，学会画图很重要！

1)单向主从同步逻辑图：

主从复制图1.jpg

2)双向主主同步逻辑图

双向主主同步逻辑图.jpg

3)线性级联单向双子主同步逻辑图

线性级联单向双子主同步逻辑图.jpg

4)环状级联单向多主同步逻辑图

环状级联单向多主同步逻辑图.jpg

5)环状级联单向多主多从同步逻辑图

环状级联单向多主多从同步逻辑图.jpg

mysql手册中的主从复制逻辑图.jpg

在当前的生产工作中，大多数应用的mysql主从同步都是异步的复制方式，即不是严格实时的数据同步。

实时和异步：

当配置好主从复制后，所有对数据库内容的更新就必须在主服务器上进行，以避免用户对主服务器上数据库的内容的更新不一致而导致发生冲突。

那么，为什么所有的更新都要在主服务器上进行呢？

我们又如何确保用户在主服务器上更新呢？通过发邮件或者口头告诉开发人员约定好，

不让其写从库，还是有别的办法？

其实可行的办法有很多，下面我们给大家一一道来。

从技术手段让开发人员写不了，而不是约定不让他写

# 6.3 MYSQL主从复制的应用场景

MYSQL主从复制有利于数据库架构的健壮性，提升访问速度和易于维护管理

1)主从服务器互为备份

主从服务器架构的设置，可以大大地加强数据库架构的健壮性，例如：当主服务器出现问题时，我们可以人工或自动(http://oldboy.blog.51cto.com/2561410/1240412)切换到服务器继续提供服务.

这类似我们课程中的NFS存储数据通过inotify+rsync同步到备份的NFS非常类似，只不过MY

SQL的同步方案，是其自带的工具

非人力的硬件，服务故障，对于认为执行drop,delete无能为力。

2)主从服务器读写分离分担网站压力

主从服务器架构可通过程序(PHP,java)或代理软件(mysql-proxy,amoeba)对用户(客户端)的请求实现读写分离，即通过在从服务器上仅仅处理用户的select查询请求，降低用户

查询响应时间及读写同时在主服务器带来的压力，对于更新的数据(update,insert,delete)仍然交给主服务器处理，确保主服务器和从服务器保存实时同步，

如果网站是以非同步更新(以浏览器)为主的业务，如blog,www首页展示等业务，查询请求比较多，这时从服务器的读写分离负载均衡策略就很有效了，这就是传说中的读写分离数据库结构。

中大型公司：通过程序(php,java)

测试环境：代理软件(mysql-proxy,amoeba)

门户网站：分布式dbproxy(读写分离，HASH负载均衡，健康检查)。

3)根据场景拆分业务

可以把几个不同的从服务器，根据公司的业务进行拆分。例如：有为外部用户提供查询服务的从服务器，有DBA用来备份的从服务器，还有提供公司内部人员访问的后台，脚本，

日志分析及开发人员服务的从服务器，这样的拆分除了减轻主服务器的压力外，使的对外用户浏览，对内处理公司内部用户业务，及DBA备份业务互不影响，具体可以用下面的简单

架构来说明

主从架构生产环境从服务器分业务拆分使用案例：

M--->S1===>对外部用户提供服务(浏览帖子，浏览博客，浏览文章)负载均衡

>S2===>对外部用户提供服务(浏览帖子，浏览博客，浏览文章)负载均衡

>S3===>对外部用户提供服务(浏览帖子，浏览博客，浏览文章)负载均衡

>S4===>对外部用户提供服务(后台访问，脚本任务，数据分析，开发人员浏览)负载均衡

>S5===>对外部用户提供服务(开启从服务器binlog功能，可实现增量备份及恢复)负载均衡

停掉Sql线程，不停io线程

提示：其中S1,S2,S3可以做出负载均衡的模式

逻辑图：

mysql主从同步架构根据业务拆分从库的方案.jpg

6.4 如何实现MYSQL主从读写分离

1)通过程序实现读写分离(性能，效率最佳，推荐)

PHP和JAVA程序都可以通过设置多个连接文件轻松实现对数据库的读写分离，即当select时，就会连接读库的连接文件，当update,insert,delete时就连接写库的连接文件

2)通过软件实现读写分离

MYSQL-proxy,Amoeba等代理软件业可以实现读写分离功能，但最常用好用的还是程序实现读写分离

3)超大型公司开发dbproxy

读写分离逻辑展示如下：

主从同步的原理总结

1)同步的时候，从库有两个线程来完成实时同步(io,sql),而主库有一个线程来完成(io)

2)要在从库上配置连接主库的IP,用户名,账号,密码,同步文件的位置，POS点

3)在实现同步之前，要将主库的数据直接导入到从库中.(同步文件的位置，POS点之前的那部分数据)

4)在同步开关没有开启之前，要在主库上建立主从数据库同步复制数据的账号

5)主库要打开binlog开关

6)从库要打开开关的过程，实际就是要这两个线程工作的过程(从库开启start slave)

# 6.5 MYSQL数据库主从同步实战

6.5.1 定义服务器角色表

|  |  |  |
| --- | --- | --- |
| 服务器名 | IP地址:端口号 | 角色描述 |
| mysql master | 192.168.100.50 3306 | mysql主库 |
| mysql slave | 192.168.100.50 3307 | mysql从库 |

提别提示

1)一般常规做主从复制，主从服务器多数在不同的机器上，并且监听的端口均为默认的3306

2)本文主从复制及时是针对前面的实例以单数据库多实例环境来讲解的。

3)第一点虽然不在一个机器上，但是步骤和过程和第二点是一样的。

4)大家在掌握了本文单数据库多实例的同步方法后，可以自己适当扩展，完成异机相同端口之间的主从复制

## 6.5.2 数据库环境准备

1)具备单机单数据库多实例的环境

2)主库打开log-bin的开关

vim /etc/3306/my.cnf

1:启用如下功能选项

[mysqld]

log-bin=/data/3306/mysql-bin

提示：

(1):上面两参数要放在my.cnf的[mysqld]模块下，否则会出错

(2):sever-id的值使用服务器IP地址的最后8位如19，目的是避免不同机器和实例ID

重复(不适合多实例)

0<server-id<2的32次方-1的自然数

(3):要先在my.cnf配置文件中查找相关参数，并按要求修改，不存在的时在添加参数，

切记，参数不要重复

(4)：修改my.cnf配置后需要重启数据库，命令为:/data/3306/mysql restart,注意要确认真正重启了

2:确保server-id的值与从库不一样，保存唯一性

server-id = 1

配置完成后，保存退出配置文件,并检查配置内容

egrep "log-bin|server-id" /data/3306/my.cnf

3)重启主库数据库实例

/data/3306/mysql restart

4)检查配置是否生效

ll /data/3306/

或者：

mysql -uroot -poldboy123 -S /data/3306/mysql.sock -e "show variables like 'log\_bin';"

3)从库修改server-id的值与主库不相同

vim /data/3307/my.cnf

修改server-id值与主库不一样

比如：server-id = 3

配置完成后，保存退出配置文件,并检查配置内容

egrep "log-bin|server-id" /data/3307/my.cnf

4)在主库上建立用于主从同步的账号req

mysql -uroot -p'oldboy123' -S /data/3306/mysql.sock

mysql> grant replication slave on \*.\* to 'backup'@'192.168.100.%' identified by 'redhat'

mysql> flush privileges;

# replication slave:为MYSQL同步必须权限，此处不要授权all

# \*.\*:表示所有库和所有表，你也可以指定具体的库和表进行复制。oldboy.text

# 'req'@'10.0.0.%':req为同步账号，10.0.0.%为授权主机网段，使用了%表示允许

整个10.0.0.0网段以req用户访问

5)对主库的数据进行备份

mysql> flush table with read lock; #锁库操作

1:对同步的点进行记录

mysql> show master status;

或者

mysql> show master logs;

2:在重开一个连接数据库的窗口主执行主库备份命令

mysqldump -uroot -poldboy123 -S /data/3306/mysql.sock -A -B --events --master-data=2 >/server/backup/backup.sql

vim /server/backup/backup.sql

3)在主库上解除数据库锁

mysql> show master status;

mysql> unlock tables;

6)将主库备份后的数据导入到从库中

mysql -uroot -poldboy123 -S /data/3307/mysql.sock </server/backup/backup.sql

7)在从库上执行以下命令

mysql -uroot -poldboy123 -S /data/3307/mysql.sock

mysql> CHANGE MASTER TO MASTER\_HOST='192.168.100.50',MASTER\_PORT=3306,MASTER\_USER='backup',MASTER\_PASSWORD='redhat',MASTER\_LOG\_FILE='mysql-bin.000005', MASTER\_LOG\_POS=423;

#MASTER\_HOST=10.0.0.7:这里是主库的IP地址

#MASTER\_PORT=3306：这里主库的端口，从库端口可能和主库不同

#MASTER\_USER='req'：这里是主库上建立的用于主从复制的用户req

#MASTER\_PASSWORD='oldboy123:这里是req用户的密码

#MASTER\_LOG\_FILE='mysql-bin.000003：这里是show master status命令时看到的二

进制日志文件名称

#MASTER\_LOG\_POS=333:这里是show master status时看到的二进制日志偏移量

mysql> start slave;

mysql> show slave status\G

8)测试数据库主从复制功能

mysql -uroot -poldboy123 -S /data/3306/mysql.sock

mysql> show databases;

mysql> create database oldboy;

mysql -uroot -poldboy123 -S /data/3307/mysql.sock

mysql> show databases;

注意：MYSQL5.1的锁表命令:flush tables with read lock

MYSQL5.5的锁表命令:flush table with read lock

这个锁表命令的时间，在不同的引擎的情况下，会受下面参数的控制，锁表时，如果

超过设置时间不操作会自动解锁。

interactive\_time =60

wait\_timeout = 60

默认情况下的时长

mysql> show variables link '%timeout%';

实现策略思想：

将整个架构做读写分离，主库只执行写入操作，多个从库做读操作，可以将多个从库做集群。

6.5.3 主从复制部署配置问题汇总

1:主库 show master status后没有任何结构，主库binlog功能开关没开启或开启后没

生效

排除思路和步骤：

1)查看配置文件或数据库中的配置是否修改正确

egrep "server-id|log-bin" /data/3306/my.cnf

mysql> show variables like 'server-id';

mysql> show variables link 'log-bin';

提示：配置文件里的参数和show variables里的参数不一样，例如配置文件里的配置

选项是log-bin,而使用show命令查看到的结果是log\_bin

2)故障现象

Last\_IO\_Error: Got fatal error 1236 from master when reading data from binary

log:'Could not find first log file name in binary log index file'

解决方法和步骤：

1）查看你配置的CHANGE MASTER TO命令中MASTER\_LOG\_FILE的值是否正确(注意空格)

2）查看binlog.index文件中是否有该mysql-bin的相关记录

3）查看/application/mysql/data中是否有该mysql-bin的日志文件

提示：由于切换binlog导致show master status位置变化无影响

6.5.4 回顾MYSQL主从同步配置步骤及原理

6.5.4.1 MYSQL主从同步配置步骤

1)准备两台数据库环境，或者单台多实例环境，能否正常启动和登陆

2)配置my.cnf文件，主库配置log-bin和server-id参数，从库配置server-id，不能和

主库及其他库一样，一般不开启从库log-bin功能，注意：配置参数后要重启生效

3)登陆主库增加用于从库连接主库同步的账号：req,并授权replication slave同步的

权限。

4)登陆主库，整库锁表flush table with read lock(窗口关闭后即失效,超时参数到了

也失效);,然后show master status查看binlog的位置状态

5)新开窗口，linux命令行备份或导出原有的数据库数据，并拷贝到从库所在的服务器

目录

如果数据量很大，并且允许停机，可以停机打包，而不用mysqldump.

6)解锁主库，unlock tables;

7)把主库导出的原有数据恢复从库

8)根据主库的show master status查看binlog的位置状态，在从库执行change master

to...

9)从库开启同步开关 start slave

10)从库show slave status\G 检查同步状态，并在主库进行更新测试

## 6.5.5 生产场景快速配置MYSQL主从复制方案

1:安装好配置从库的数据库，配置好log-bin和server-id参数

2:无需配置主库my.cnf文件，主库的log-bin和server-id参数默认就是配置好的

3:登陆主库增加用于从库连接主库同步的账户例如:req,并授权replication slave同步

的权限

4:使用半夜mysqldump带--master-data=1备份的全备数据恢复到从库

5:在从库执行change master to ...语句，无需binlog文件及对应位置点

6:从库开启同步开关，start slave

7:从库show slave status\G,检查同步状态，并在主库进行更新测试。

# 6.5.6 一键自动化配置主从复制脚本

6.5.6.1 生产环境主从库同步配置注意事项

1)第一次做从库如何做？

1:申请设备资源，用来做从库服务器。

2:攥写方案文档和实施步骤

假如你的服务器只有主库，而且已经跑了生产线上应用了，现在由于业务需要第一次做

从库，此时可能需要和公司申请维护时间(要确认这个时间段内可以做一次全背)，即在

用户访问量最小，且不影响内部其他业务运转的时间来停机(包括锁表)配置主从复制，

一般都是凌晨进行。注意：停机(锁表，停库)的最小时间段，为锁表后备份的时间，也

就是说无需等待主从配置好

当然，也可以不申请停机时间，在定时任务备份时，每天的夜里的定时备份时做一些措施即可实现，

## 6.5.7 无需熬夜在工作时间轻松配置从库

当然了，也可以不申请停机时间，在定时任务备份时，每天夜里服务器压力小的时刻定时备份时做一些措施即可，如：模拟主从同步的步骤，获取到全备过程中binlog位置的信息或者直接用--master-data 参数解决

6.5.7.1 一键做主从复制脚本

1)锁表备份全备一份

2)锁表前后取得show master status值记录日志里

这样就可以在白天从容的实现主从同步，这个脚本还是很容易实现的，如

脚本方法1

cat mysql\_bak.sh

#!/bin/sh

MYUSER=root

MYPASS="oldboy"

MYSOCK=/data/3306/mysql.sock

MAIN\_PATH=/server/backup

DATA\_PATH=/server/backup

LOG\_FILE=${DATA\_PATH}/mysqllogs\_`date +%F`.log

DATA\_FILE=${DATA\_PATH}/mysql\_backup\_`date +%F`.sql.gz

MYSQL\_PATH=/application/mysql/bin

MYSQL\_CMD="$MYSQL\_PATH/mysql -u$MYUSER -p$MYPASS -S $MYSOCK"

MYSQL\_DUMP="$MYSQL\_PATH/mysqldump -u$MYUSER -p $MYPASS -S $MYSOCK -A -B --single-transaction -e" cat | $MYSQL\_CMD<<EOF

flush tables with read lock;

system echo "-------show master status result--------" >>$LOG\_FILE;

system $MYSQL\_CMD -e "show master status"|tail -1 >>$LOG\_FILE;

system ${MYSQL\_DUMP} | gzip >$DATA\_FILE;

EOF

$MYSQL\_CMD -e "unlock tables;"

执行过程与结果：

sh mysql\_bak.sh

脚本2：

#!/bin/sh

MYUSER=root

MYPASS="oldboy"

MYSOCK=/data/3306/mysql.sock

MAIN\_PATH=/server/backup

DATA\_PATH=/server/backup

LOG\_FILE=${DATA\_PATH}/mysqllogs\_`date +%F`.log

DATA\_FILE=${DATA\_PATH}/mysql\_backup\_`date +%F`.sql.gz

MYSQL\_PATH=/application/mysql/bin

MYSQL\_CMD="$MYSQL\_PATH/mysql -u$MYUSER -p$MYPASS -S $MYSOCK"

MYSQL\_DUMP="$MYSQL\_PATH/mysqldump -u$MYUSER -p$MYPASS -S $MYSOCK -A -B --master-data=1 --single-transcation -e"

${MYSQL\_DUMP} | gzip >$DATA\_FILE

有了mysql\_backup\_2012-04-17.sql.gz和mysqllogs\_2014-04-07.log内容作从库还远

吗?

6.5.8 不停主库一键批量创建从库

6.5.8.1 一键创建从库方法1

这步就不难了。首先可使用前面讲过的批量分发数据方案，把mysql\_backup 2012-04-\

17.sql.gz发布到想做从库同步的机器上，可以多台

然后在制作如下批量做从库脚本，一键完成多台机器从库的制作。

#!/bin/sh

MYUSER=root

MYPASS="oldboy"

MYSOCK=/data/3306/mysql.sock

MAIN\_PATH=/server/backup

DATA\_PATH=/server/backup

LOG\_FILE=${DATA\_PATH}/mysqllogs\_`date +%F`.log

DATA\_FILE=${DATA\_PATH}/mysql\_backup\_`date +%F`.sql.gz

MYSQL\_PATH=/application/mysql/bin

MYSQL\_CMD="$MYSQL\_PATH/mysql -u$MYUSER -p$MYPASS -S $MYSOCK"

#recover

cd ${DATA\_PATH}

gzip -d mysql\_backup\_`data +%F`.sql.gz

$MYSQL\_CMD< mysql\_backup\_`date +%F`.sql

#config slave

cat |$MYSQL\_CMD<<EOF

CHANGE MASTER TO

MASTER\_HOST='10.0.0.19',

MASTER\_PORT=3306,

MASTER\_USER='req',

MASTER\_PASSWORD='oldboy123',

MASTER\_LOG\_FILE='mysql-bin.000008',

MASTER\_LOG\_POS=342;

EOF

$MYSQL\_CMD -e "start slave;"

$MYSQL\_CMD -e "show slave status\G|egrep "IO\_Running|SQL\_Running" >$LOG\_FILE

mail -s "mysql slave result" 286937899@qq.com< $LOG\_FILE

特别提示：

1)此脚本中change master参数要根据mysqllogs\_2012-04-17.log修改

2)此脚本未经调试，请大家自行调试

6.5.8.2 一键创建从库方法2

#!/bin/sh

#

MYUSER=root

MYPASS="oldboy"

MYSOCK=/data/3306/mysql.sock

MAIN\_PATH=/server/backup

DATA\_PATH=/server/backup

LOG\_FILE=${DATA\_PATH}/mysqllogs\_`date +%F`.log

DATA\_FILE=${DATA\_PATH}/mysql\_backup\_`date +%F`.sql.gz

MYSQL\_PATH=/application/mysql/bin

MYSQL\_CMD="$MYSQL\_PATH/mysql -u$MYUSER -p$MYPASS -S $MYSOCK"

#recover

cd ${DATA\_PATH}

gzip -d mysql\_backup\_`date +%F`.sql.gz

$MYSQL\_CMD< mysql\_backup\_`date +%F`.sql

#config slave

cat | $MYSQL\_CMD<<EOF

CHANGE MASTER TO

MASTER\_HOST='10.0.0.19',

MASTER\_PORT=3306

MASTER\_USER='req',

MASTER\_PASSWORD='oldboy123',

MASTER\_LOG\_FILE='mysql-bin.000008',

MASTER\_LOG\_POS=342;

EOF

$MYSQL\_CMD -e "start slave;"

$MYSQL\_CMD -e "show slave status\G"|egrep "IO\_Running|SQL\_Running" >$LOG\_FILE

mail -s "mysql slave result"286937899@qq.com< $LOG\_FILE

6.5.9 相关MYSQL主从复制技术技巧概览

6.5.9.1 登陆数据库查看MYSQL线程同步状态

主库

mysql> show processlist\G

从库

mysql> show processlist\G

6.5.9.2 复制主线程状态

下面列出了主服务器的Binlog Dump线程的state列的最常见的状态，如果你没有在主服

务器上个看见任何Binlog Dump线程，这说明复制没有在运行一即，目前没有连接任何

从服务器

.Sending binlog event to slave

二进制日志由各种事件组成，一个事件通常为一个更新加一些其他信息，线程已经从二

进制日志读取了一个事件并且正将它发送到从服务器

.Finished reading one binlog; switching to next binlog

线程已经读完了二进制日志文件并且正打开下一个要发送到从服务器的日志文件。

.Has sent all binlog to slave;waitting for binlog to be update

线程已经从二进制日志读取所有主要的更新并已经发送到了从服务器。线程现在

正空闲，等待由主服务器上新的更新导致的出现在二进制日志中的新事件

.Waiting to finalize termination

线程停止时发生的一个很简单的状态

### 6.5.9.3 复制从I/O线程状态

下面列出了从服务器的I/O线程的state列的最常见的状态，改状态也出现在

Slave\_IO\_State列，由SHOW SLAVE STATUS显示，着说明你可以只通过给语句仔细浏览所

发生的事情。

.Connectiong to master

线程正试图连接主服务器

.Checking master version

建立同主服务器之间额连接后立即临时出现的状态

.Registering slave on master

建立同主服务器之间的连接后立即临时出现的状态，线程向主服务器发送一条请求，索取从请求的二进制日志文件名和位置开始的二进制日志的内容

.Waiting to reconnect after a failed binlog dump request

如果二进制日志转储请求失败(由于没有连接)，线程进入睡眠状态，然后定期尝试重新

连接，可以使用--master-connect-retry选项指定重试之间的间隔

.Reconnecting after a failed binlog dump request

线程正尝试重新连接主服务器

.Waiting for master to send event

线程已经连接上主服务器，正等待二进制日志事件到达，如果主服务器空闲，会持续较长的时间，如果等待持续slave\_read\_timeout秒，则发生超时，此时，线程认为连接被中断并企图重新连接

.Queueing master\_event to the relay log

线程已经读取一个事件，正将它复制到中继日志共SQL线程来处理

.Waiting to reconnect after a failed master event read

读取时(由于没有连接)出现错误，线程企图重新连接前将睡眠master-connecte

-retry秒

.Reconnecting after a failed master event read

线程正尝试重新连接主服务器，当连接冲洗建立后，状态变为waiting for master

to send event

.Waiting for the slave SQL thread to free enough relay log space

正使用一个非零relay\_log\_space\_limit值，中继日志已经增长到其组合大小超过该

值。I/O线程正等待直到SQL线程处理中继日志内容并删除部分中继日志文件来释放

足够的空间

.Waiting for slave mutex on exit

线程停止时发生的一个很简单的状态

6.5.9.4 复制从SQL线程状态

下面列出了从服务器的SQL线程的State列的最常见的状态

.Reading event from the relay log

线程已经从中继日志读取一个事件，可以对事件进行处理了

.Has read all relay log;waiting for the slave I/O thread to update it

线程已经处理了中继日志文件的所有事件，现在正等待I/O线程将新事件写入中继日志

.Waiting for slave mutex on exit

线程停止时发生的一个很简单的状态

I/O线程的State列也可以显示语句的文本，这说明线程已经从中继读取了一个事件，从中提取了一个语句，并且正在执行该语句

### 6.5.9.5 查看MYSQL现场同步状态的用途

通过MYSQL现场同步状态查看数据库同步是否完成，用于主库宕机或者人工数据库主从切换迁移等

主库宕机选择最快的从库提升为主，就需要查看，当然也可以利用MYSQL的半同步功能，选择固定的库提升为主

# 6.6 生产场景MYSQL主从复制读写分离授权方案及实战

当配置好MYSQL主从复制后，所有对数据库内容的更新就必须在主服务器上进行，那么，为什么所有的更新都要在主服务器进行呢？这是因为数据复制是单向的，只存在主库上更新，才能避免用户对主服务器上数据内容的更新与对从服务器上数据内容一致，而不会发生冲突

那么，我们又如何确保用户在主服务器上更新呢？

通过邮件或者口头告诉开发人员约定好，不让其写从库？

通过授权用户控制

其实可行的方法可以有很多，下面我们给大家一一道来。

## 6.6.1 生产MYSQL复制环境用户授权方案

案例1：假设为博客库数据库授权，大家可以做下面的授权：

|  |  |  |  |
| --- | --- | --- | --- |
| 用户 | 密码 | 端口 | IP |
| 写库:log\_w | oldboy456 | 3306 | 10.0.0.7 |
| 读库:log\_r | oldboy789 | 3307 | 10.0.0.8 |

缺点与风险：log\_w会连接从库，有多套用户的密码，不是很专业

案例2：生产工作中的一般做法

|  |  |  |
| --- | --- | --- |
| 用户 | 密码 | ip:端口 |
| 写库:blog | oldboy456 | 192.168.100.50:3306 |
| 读库:blog | oldboy456 | 192.168.100.50:3307 |

缺点与风险: 收回权限

生产实战案例配置：

1)生产环境主库用户授权

GRANT SELECT,INSERT,UPDATE,DELETE ON oldboy.\* TO 'blog'@'192.168.100.%' identified by 'oldboy456' #这里表示给192.168.100.0/24的用户blog管理blog数据库的所有表(\*表示所有表)

拥有(SELECT,INSERT,UPDATE,DELETE)四种权限,密码为oldboy456

提示：特殊业务可能权限会略多，如果业务安全性要求不高，也可以all privileges权限

2)生产环境从库用户授权

GRANT SELECT,INSERT,UPDATE,DELETE ON oldboy.\* TO 'blog'@'192.168.100.%' identified by 'oldboy456'; #这里表示给192.168.100/24的用户blog管理blog数据库的所有表(\*表示所有表)

拥有(SELECT,INSERT,UPDATE,DELETE)四种权限,密码为oldboy456

REVOKE INSERT,UPDATE,DELETE ON oldboy.\* FROM 'blog'@'192.168.100.%'; #收回blog用户的

INSERT,UPDATE,DELETE三种权限，只让其拥有INSERT一种权限

或者：

GRANT SELECT ON oldboy.\* TO 'blog'@'10.0.0.%' identified by 'oldboy456'

##这里表示给10.0.0.0/24的用户blog管理blog数据库的所有表只拥有INSERT只读权限

提示：可以结合read-only参数共同做

最简单的办法是在主库配置binlog-ignore-db=mysql

## 6.6.2 通过忽略授权表的方式防止数据写从库的方法及实践

从技术手段上让开发人员写不了，而不是约定不让他写。

生产环境中一般会采取忽略授权表方式同步，然后对从服务器(slave)上的用户仅授权

select读权限，不同步MYSQL库，这样我们就保证主库和从库在相同的用户可以授权不同的

权限。

忽略MYSQL库和information\_schema库的主从同步

该配置在主库上配置：vim /data/3306/my.cnf,添加如下功能选项

[mysqld]

replicate-ignore-db = mysql

#binlog-do-db = testdb

binlog-ignore-db = mysql

binlog-ignore-db = preformance\_schema

binlog-ignore-db = information\_schema

保存配置后，要重新启动mysql服务器

/data/3306/mysql stop

/data/3306/mysql start

相关参数说明

DO:同步少量库

binlog-do-db=db\_oldboy

replicate-do-db=db\_oldboy

replicate-do-table=db\_oldboy

replicate-wild-do-table=db\_oldboy

ignore:排除

binlog-ignore-db=mysql

replicate-ignore-db=mysql

replicate-ignore-table=mysql

replicate-wild-ignore-table=db\_oldboy

提示：忽略纪录binlog日志的参数binlog-ignore-db一般用于系统的库和表

mysql5.0-mysql5.1,对于MYSQL5.5大家卡查询一下相关的参数

结论1：只在从库上配置 replicate-ignore-db=mysql 并不能做到从库不同步MYSQL库

结论2：只在主库上配置 replicate-ignore-db=mysql 并不能做到从库不同步MYSQL库

然后在主库和从库上做如下授权

主库上的操作：GRANT SELECT,INSERT UPDATE,DELETE ON blog.\* TO 'blog'@'10.0.0.%' identified by 'oldboy456'

从库上的操作：GRANT SELECT ON blog.\* TO 'blog'@'10.0.0.%' identified by 'oldboy456'

风险：当从库切换到主库时，连接用户就有权限问题了(只有SELECT权限没有其他的权限)

解决办法：保留一个从库准备接替主库，而这个库就可以不配置不同步主库的功能。

# 6.7 MYSQL主从复制的故障解决

## 6.7.1 MYSQL从库数据冲突导致同步停止

模拟错误：模拟重现故障的能力是运维人员最重要能力

执行show slave status命令后出现以下故障现象

...

Slave\_IO\_Running:Yes

Slave\_SQL\_Running:NO

Seconds\_Behind\_Master:NULL

Last\_Error: Error 'Can't create database 'xiaoliu'; database exists' on query.

Default database:'xiaoliu'.Query:'create database xiaoliu'

mysql -uroot -poldboy123 -S /data/3307/mysql.sock

mysql> create database butongbu;

mysql -uroot -podlboy123 -S /data/3306/mysql.sock

mysql> create database butongbu;

mysql -uroot -poldboy123 -S /data/3307/mysql.sock

mysql> show slave status\G;

解决办法一(在从库上执行如下功能)：

mysql> stop slave

mysql> set global sql\_slave\_skip\_counter =1;

mysql> start slave

1)对于普通的互联网业务，忽略问题不是很大，当然，要确认不影响公司业务的前提下。

2)企业场景解决主从同步，比主从不一致对当前业务更重要，然后如果主从数据一致也很

重要，在找个时间恢复这个从库

方法二：根据错误号跳过指定的错误

slave-skip-errors = 1032,1062,1007 #一般由于入库重复导致的失败就可以忽略对于错误代码，你应使用SLAVE服务器错误日志中错误消息提供的编号和SHOW SLAVE STATUS的输出，服务器错误代码列于附录B：错误代码和消息,你也可以(但不应)使用不推荐的all值忽略所有错误消息，不考虑所发生的错误，无需而言，如果使用该值，我们不能保证数据的完整性，在这种情况下，如果SLAVE服务器的数据与MASTER服务器上的不想尽情，不要抱怨(或编写BUG报告)，已经警告你了。

在从库上执行如下功能

vim /data/3307/my.cnf

[mysqld]

...

slave-skip-errors = 1032,1062,1007

/data/3307/mysql stop

/data/3307/mysql start

其他原因引起的同步问题

1)mysql自身的原因

2)不同的数据库版本引起不同步，低版本到高版本可以，但是高版本不能往低版本同步

3)mysql的错误

## 6.7.2 让MYSQL SLAVE库记录binlog方法

需要记录binlog的情况

1)当前从库还要作为其他库的主库.也就是级联同步

2)把从库作为备份服务器时需要开启binlog

案例实战

1)在从库的my.cnf中加入如下参数

vim /data/3307/my.cnf

[mysqld]

log-slave-updates

log-bin = mysql3307-bin

expire\_logs\_days = 7 #find /data/3306/-type f -name "mysql-bin.000\*" -mtime +7

|xargs rm -f

/data/3307/mysql stop

/data/3307/mysql start

应用场景：级联复制或 从库做数据备份 A-->B-->C B服务器要开启记录binlog日志功能

# 6.8 生产场景案例

6.8.1 主库宕机，从库切换主库角色后，实现主从同步的过程

1)登陆到要切换的从库上执行如下命令

mysql> show processlist;

查看State：状态值

2)查看所有从库的master.info文件内容,查看到那个从库的记录位置更靠前

或者利用半同步功能，直接选择做了实时同步的那个从库作为主库

cat /data/3307/master.info

提示：授权同步用户和主库要一致

3)将记录位置值最大的库作为主库使用

4)实现主从库之间的切换

1：在每个从库上执行stop slave io\_thread; show processlist;

直到看到Has read all relay log;这个状态信息为止，就说明

从库更新都执行完毕了

2：登陆那个要切换成主库的从库上执行如下命令

mysql -uroot -p'oldboy' -S /data/3307/mysql.sock

mysql> stop slave

mysql> retset master;

mysql> quit

3:进入切换后主库的数据库数据目录中，删除master.info和relay-log.info这两个文件

cd /data/3307/data

rm -f master.info relay-log.info

检查授权表，read-only参数

4:提升从库为主库

vim /data/3307/my.cnf

开启如下功能选项

log-bin = /data/3307/mysql-bin

#如果存在log-slave-updates和read-only参数，一定要将该参数注释掉

/data/3306/mysql restart

提示补充：

如果主库服务器没有宕机，需要去主库拉取binlog，将缺失的binlog记录在从库中补全，然后提升该从库为主库

5:其他从库的操作

已经检查(同步user req均存在)

登陆到其他的从库上都执行以下功能比如

mysql -uroot -poldboy123 -S /data/3308/mysql.sock

mysql> stop slave

mysql> change master to master\_host='10.0.0.20;'#如果不同步，就指定位置点

mysql> start slave;

mysql> show slave status\G;

## 6.8.2 从库宕机

1)重新配置一个新的MYSQL SLAVE从库

2)将备份的数据直接导入到从库中

mysql -uroot -poldboy -S /data/3308/mysql.sock

mysql> stop slave

mysql> quit

gzip -d oldboy\_date\_5.sql.gz

mysql -uroot -p'oldboy' -S /data/3308/mysql.sock < oldboy\_data\_5.sql &

mysql -uroot -poldboy -S /data/3308/mysql.sock

mysql> change master to master\_host='10.0.0.19',master\_user='rep' master\_password='oldboy123',master\_log\_file='mysql-bin.001440',master\_log\_pos=

68824

mysql> start slave;

mysql> show slave status\G

## 6.8.3 双主机多主同步过程实战

1)主从做双向同步操作

vim /data/3307/my.cnf

添加如下功能参数

auto\_increment\_increment

auto\_increment\_offset

例如：

[mysqld]

auto\_increment\_increment = 2

auto\_increment\_offset = 2

user = mysql

port = 3307

......

log-bin = /data/3307/mysql-bin

log-slave-updates

......

/data/3307/mysql stop

/data/3307/mysql start

vim /data/3306/my.cnf

[mysqld]

auto\_increment\_increment = 2

auto\_increment\_offset = 1

user = mysql

port = 3307

......

log-bin = /data/3306/mysql-bin

log-slave-updates

......

/data/3306/mysql stop

/data/3306/mysql start

mysql -uroot -p'oldboy123' -S /data/3307/mysql.sock

mysql> show slave status\G

mysql> quit

mysqldump -uroot -poldboy123 -S /data/3307/mysql.sock -A -B --master-data=1 -x

--events >/opt/3307bak.sql

mysql -uroot -poldboy123 -S /data/3306/mysql.sock </opt/3307bak.sql

mysql -uroot -poldboy123 -S /data/3306/mysql.sock

mysql> CHANGE MASTER TO MASTER\_HOST='1O.O.O.7',MASTER\_PORT=3307,MASTER\_USER='req',

MASTER\_PASSWORD='oldboy123';

mysql>START SLAVE

mysql>SHOW SLAVE STATUS\G

2:数据库双主模式功能测试

mysql> user oldboy

mysql> show tables;

mysql> drop table t;

mysql> create table student(id int(4) not null AUTO\_INCREMENT,name char(20) not

null,primary key(id));

mysql> insert into student(name) values('oldboy1');

mysql> insert into student(name) values('oldboy2');

mysql> insert into student(name) values('oldboy3');

mysql> select \* from oldboy.student;

mysql> quit

mysql -uroot -poldboy123 -S /data/3307/mysql.sock

mysql> select \* from oldboy.student;

mysql> use oldboy

mysql> insert into student(name) values('oldgirl1');

mysql> insert into student(name) values('oldgirl2');

mysql> insert into student(name) values('oldgirl3');

mysql> select \* from oldboy.studnet;

使用主主前提：

1：表的主键自增

2：程序与库指定ID(通过程序指定ID写库，比如M1,1,3,5,M2:2,4,6)

6.8.4 数据库数据丢失的灾难性恢复的案例实战

6.8.4.1 625某电商网站数据库特大故障解决实录

1)全量备份与增量备份

1:全量数据就是数据库中所有的数据，全量备份就是把数据库中所有的数据进行备份

例如:

备份所有库：

mysqldump -uroot -poldboy -S /data/3306/mysql.sock -F -B -A|gzip >

/server/backup/mysqlbak\_$(date +%F).sql.gz

备份一个库：

mysqldump -uroot -poldboy -S /data/3306/mysql.sock -F -B oldboy|gzip >

/server/backup/mysqlbak\_$(date +%F).sql.gz

2:增量数据就是上次全量备份之后，更新的新数据，对于MYSQL来说，binlog日志就是

MYSQL的增量数据。

1:按天备份情况

周一00全量备份 周二00全量备份

01.sql.gz 02.sql.gz

周一增量备份 周二增量备份

mysql-bin.000024 mysql-bin.000037

mysql-bin.000025 mysql-bin.000038

mysql-bin.000026 mysql-bin.000039

... ...

优点：

恢复时间:短:

维护时间:低

缺点：

占有空间:多，占用系统资源多，经常要锁表，影响用户体验

2:按周备份情况

周六00全量备份

01.sql.gz

周一增量备份 周二增量备份 周三增量备份 周五增量备份

mysql-bin.000024 mysql-bin.000037 mysql-bin.000040 mysql-bin.000048

mysql-bin.000025 mysql-bin.000038 mysql-bin.000041 mysql-bin.000049

mysql-bin.000026 mysql-bin.000039 mysql-bin.000042 mysql-bin.000050

... ... ... ...

mysql-bin.index mysql-bin.index mysql-bin.index mysql-bin.index

优点:占用空间小，无需锁表，用户体验好

缺点：恢复时间长，复杂，维护成本大

企业场景全量和增量的频率是如何做的呢？

1)中小型公司，全量一般是每天一次，业务流量低估执行全备，执行前要锁表。

2)单台数据库，如何增量。用rsync(配合定时任务效率大点，或者inotify,主从复制)把

所有binlog备份到远程服务器。尽量做主从复制

增量备份的例子:rsync -avz /data/3306/mysql-bin.000\* rsync\_backup@10.0.0.18:

backup --password-file=/etc/rsync.password

3)大公司周备，每周六00点一次全量，下周日-下周六00点前都是增量

优点：节省备份时间，减少备份压力，缺点：增量的binlog文件副本太多，还原会很麻烦

4)一主五从，会有一个从库做备份，延迟同步。

mysql的mysqldump备份什么派上用场

1)迁移或者升级数据库时

2)增加从库时候

3)如果因为硬件或特殊异常，主库或从库宕机，主从可以互相切换，无需备份。

4)在主库上人为错误的删除数据时，需要备份

5)跨机房灾备，需要将备份文件拷贝走。需要全量备份

6)只有一个主库是需要增量恢复操作的

2)MYSQL增量备份的必备条件

1:开启MYSQL log-bin日志功能

MYSQL数据库开启了log-bin参数记录binlog日志功能如下:

grep log-bin /data/3306/my.cnf

log-bin = /data/3306/mysql-bin

提示：主库和备库的从库都要开启binlog记录功能

小结：增量恢复的条件

存在一份全备加上全备之后的时刻出问题时刻的所有增量binlog文件备份

3)存在MYSQL数据库全备

1:生产环境mysqldump备份命令

在凌晨某一个时刻进行数据库全备(生产场景一般通过定时任务每日凌晨执行),备份命令

如下：

mysqldump -uroot -p'oldboy' -S /data/3306/mysql.sock --default-character-set=

gbk --single-transaction -F -B oldboy|gzip >/server/backup/mysql\_$(data +%F).

sql.gz

innodb引擎备份:

mysqldump -uroot -poldboy123 -S /data/3306/mysql.sock -F --single--transaction

-A -B|gzip >/server/backup/mysql\_$(data +%F).sql.gz

myisam引擎备份:

mysqldump -uroot -poldboy123 -S /data/3306/mysql.sock -F -A -B --lock-all-table|gzip >/server/backup/mysql\_$(data +$F).sql.gz

4)生产环境mysqldump备份脚本

下面是全备加上获取备份后最新起始binlog文件名的脚本

cat /server/scripts/mysql\_bak.sh

5)MYSQL增量恢复案例图解

MYSQL增量恢复案例图解.jpg

6.8.4.2 恢复数据库场景

6.8.4.2.1 多实例数据库主库为例

查看端口

netstat -lnt | grep 3306

数据库全备份操作过程

mysql -uroot -poldboy123 -S /data/3306/mysql.sock

mysql> show variables like '%character-set%';

mysql> use oldboy;

mysql> show tables;

mysql> quit

date -S '2017/02/19'

mkdir /server/backup

mysqldump -uroot -p'oldboy123' -S /data/3306/mysql.sock -F -B --master-data=2 oldboy|gzip >/server/backup/quanbei\_$(date +%F).sql

#数据库数据添加模拟

mysql -uroot -poldboy123 -S /data/3306/mysql.sock

mysql> use oldboy;

mysql> desc student;

mysql> insert into student(name) values('oldboy101');

mysql> insert into student(name) values('oldboy102');

mysql> select \* from student;

mysql> quit

#数据库故障操作模拟(删除数据库造成故障)

mysql -uroot -poldboy123 -S /data/3306/mysql.sock

mysql> show databases;

mysql> drop database oldboy;

mysql> show databases;

mysql> select \* from oldboy.student;

mysql> quit

提示：公司运营发现用户无法连接到oldboy库了，打电话给公司运维，运维人员经过故障盘查，发现oldboy数据库被人在10:00钟左右意外删除，准备实施数据库数据灾难恢复工作

#数据库数据恢复全过程

1)通过防火墙禁止web等应用程序向主库写数据或者锁表，让主库暂时停止更新，然后进行恢复。

2)检查全备及binlog日志

ls /server/backup #检查凌晨全备份的数据文件

ls -lrt /data/3306/mysql-bin.\* #检查增量备份的binlog文件

grep -i "change" /server/backup/quanbei\_$(date +%F).sql

3)刷新一下binlog

mysqladmin -uroot -p'oldboy123' -S /data/3306/mysql.sock flush-logs

刷新binlog之后，你所恢复的binlog应该是倒数第二个mysql-bin文件

而倒数第一个mysql-bin文件是新生成的二进制日志文件

cp mysql-bin.000014 /server/backup/

cd /server/backup/

mysqlbinlog -d oldboy mysql-bin.000014 >bin.sql #将二进制binlog文件解析成sql文件

vim bin.sql

在该文件中招到drop database oldboy这个命令后，从该文件中删除该命令后，保存退出

1)MYSQL对外不写入的情况,如果是这个环境，那么倒数第一个binlog文件的内容是空的，

那么该文件的数据是不需要再导入到数据库的

1:先执行全量备份数据的恢复操作

mysql -uroot -poldboy123 -S /data/3306/mysql.sock <quanbei\_$(date +%F).sql

2:做增量备份数据的恢复操作

mysql -uroot -poldboy123 -S /data/3306/mysql.sock oldboy <bin.sql

mysql -uroot -poldboy123 -S /data/3306/mysql.sock

mysql> select \* from oldboy.student;

2)MYSQL对外可写入的情况，

mysql -uroot -p'oldboy123' -S /data/3306/mysql.sock

mysql> show variables like '%sql\_log\_bin%'

## 6.8.5 更多MYSQL数据库增量恢复总结

第一种方案：

1)停止一个从库，然后在主库上刷新binglog，把倒数第二个二进制文件mysql-bin恢复成bin.sql(删除drop语句)

2)把全备备份文件quanbei\_$(date +%F).sql及10点前的增量备份文件bin.sql恢复到从库

3)切换到从库提供服务，数据丢多少？10:10分刷新BINLOG以后的数据(倒数第二个以后的二进制日志文件)

4)把倒数第二个以后的二进制日志文件(有可能是一个，或多个)解析成sql，恢复到从库

缺点：以上方案会有主库冲突问题

或者

停止主库，把倒数第二个以后的二进制日志文件(有可能是一个，或多个)解析成sql，恢复到从库，然后切换到从库提供服务

## 6.8.6 增量恢复小结

1)人为SQL造成的误操作,

2)全备和增量

3)恢复时建议对外停止更新

4)恢复全量，然后把增量日志中有问题的SQL语句删除，恢复到数据库。

增量恢复的核心思想

1:流程制度控制,防止问题发生，如果不做，面临服务和数据的丢失

2:通过延迟备份来解决，或者通过监控，黑名单，白名单机制

3:选择停库(取决于业务需求容忍度，可量化的目标，根据需求选择停库或锁表或者丢失

部分数据)

7.1 Mysql服务日志

MYSQL生成或相关联的日志文件种类繁多，本节重点关注与MYSQL数据库相关的几类日志文件

1)错误日志(error log)的介绍与调整

1:错误日志(error log)的介绍

错误日志(error log):记录MYSQL服务进程mysqld在启动/关闭或运行过程中遇到的错误

信息

2:错误日志(error log)的调整

cat /data/3306/my.cnf

...

[mysqld\_safe]

log-error=/data/3306/mysql\_oldboy3306.err

2)查询日志(query log):又可分为两类

1:普通查询日志(general query log)的介绍与调整

1)普通查询日志(general query log)的介绍

普通查询日志(general query log):记录客户端连接信息和执行的SQL语句信息:

2)普通查询日志(general query log)的调整

mysql -uroot -poldboy123 -S /data/3306/mysql.sock

mysql> show variables like '%general\_log%'

mysql> show variables like '%general\_log\_file%'

mysql> set global general\_log = ON; #查询日志的开启

2:慢查询日志(slow query log)介绍与调整

1)慢查询日志(slow query log)的介绍

慢查询日志(slow query log):记录执行时间超出指定值(long\_query\_time)的SQL语句

2)慢查询日志的调整

vim /data/3306/my.cnf

[mysqld]

...

long\_query\_time = 1

log-slow-queries = /data/3306/slow.log

mysql> show variables like '%index%';

Variable\_name Value

log\_queries\_not\_using\_indexes OFF

3:二进制日志(binary log)的介绍与调整

1)二进制日志(binary log)的介绍

二进制日志(binary log):记录数据被修改的相关信息:

2)二进制日志(binary log)的调整

cat /data/3306/my.cnf

...

[mysqld]

log-bin=/data/3306/mysql-bin

mysql -uroot -poldboy123 -S /data/3306/mysql.sock

mysql> show variables like '%log\_bin%';

Variable\_name Value

log\_bin ON #记录binlog

sql\_log\_bin ON #临时不记录binlog

7.2 binlog日志的三种模式

## 7.2.1 Statement Level模式

每一条会ixugai数据的SQL都会记录到MASTER的bin-log中，SLAVE在复制的时候SQL进程会

解析成和原来MASTER端执行过的相同的SQL来再次执行

优点:statement level下的优点首先就是解决了row level下的缺点,不需要记录每一行数

数据的变化，减少bin-log日志量，节约IO，提高性能，因为他只需要记录在MASTER上所

执行的语句的细节，以及执行语句时候的上下文的信息

缺点：由于他是记录的执行语句，所以，为了让这些语句在SLAVE端也能正确执行，那么

他还必须记录每条语句在执行的时候的一些相关信息，也就是上下文信息，以保证所有

语句在SLAVE端杯执行的能够得到和在MASTER端执行时候相同结果，另外就是，由于MYSQ

L现在发展比较快，很多的新功能不断加入，使MYSQL的复制遇到了不小的挑战，自然复制

的时候涉及到越复制的内容，BUG也就是容易出现，在statement level下，目前已经发现

的就有不少情况会造成MYSQL的复制出现问题，主要是修改数据的时候使用了某些特定的

函数或者功能的时候会出现，比如：sleep()函数在有些版本中就不能正确复制，在存储

过程中使用了last\_insert\_id()函数，可能会使slave和master上得到不一致的id等等，

由于row level是基于每一行来记录的变化，所以不会出现类似的问题

## 7.2.2 Row Level模式

日志中会记录成每一行数据被修改的形式，然后在slave端再对相同的数据进行修改

优点：在row level模式下，bin-log中可以不记录执行的sql语句的上下文相关的信息

仅仅只需要记录哪一条记录被修改了，修改成什么样，所以row level的日志内容会非常

清楚的记录下没一行数据修改的细节，非常容易理解，二期不会出现某些特定情况下的存储过程，或function,以及trigger的调用和触发无法被正确复制的问题

缺点：row level下，所有的执行语句当记录到日志中的时候，都将以每行记录的修改来

记录，这样可能会产生大量的日志内容，比如有这样一条update语句：update product set owner\_member\_id='b' where owner\_member\_id='a',执行之后，日志中记录的不是

这条update语句所对应的事件(MYSQL以事件的形式来记录bin-log日志)，而是这些语句

更新的每一条记录的变化情况，这样就记录成很多条记录被更新的很多个事件，自然，

bin-log日志的量就会很大，尤其是当执行alter table之类的语句的时候，产生的日志

量是惊人的。因此MYSQL对于alter table之类的表结构变更语句的处理方式是整个表的每

一条记录都需要变动，实际上就是重建了整个表，那么该表的每一条记录都会被记录到

日志。

## 7.2.3 Mixed模式

实际上就是前两种模式的结合。在Mixed模式下，MYSQL会根据执行的每一条具体的SQL语句

来区分对待记录的日志形式，也就是statement和row之间选项一种，新版本由statement

level还是和以前一样，仅仅记录执行的语句，而新版本的MYSQL中对row level模式也被做

了优化，并不是所有的修改都会以row level来记录，像遇到表结构变更的时候就会以

statement模式来记录，如果SQL语句确实就是update或delete等修改数据的语句，那么还是会记录所有行的变更

## 7.2.4 调整binlog日志模式方法

### 7.2.4.1 在配置文件中修改方法

在配置文件中参数如下

log-bin=mysql-bin

#binlog\_format="STATEMENT"

#binlog\_format="ROW"

binlog\_format="MIXED"

### 7.2.4.2 在线修改立即生效方法

运行时在线修改：

mysql> SET SESSION binlog\_format = 'STATEMENT';

mysql> SET SESSION binlog\_format = 'ROW';

mysql> SET SESSION binlog\_format = 'MIXED';

全局生效：

mysql> SET GLOBAL binlog\_format = 'STATEMENT';

mysql> SET GLOBAL binlog\_format = 'ROW';

mysql> SET GLOBAL binlog\_format = 'MIXED';

mysqlbinlog --base64-output=decode-rows -v mysql-bin.000016

#查看执行效果

# 8.1 MYSQL引擎概述

## 8.1.1 什么是存储引擎

在讲清楚什么是存储引擎之前，我们先来个比喻，我们都知道录制一个视频，可以转换成不同的格式文件如.mp4.avi.wmv等，而存在我们电脑的磁盘也会存在于不同类型的文件系统中如windows里常见的ntfs,fat32,存在于linux里常见的ext3,ext4,xfs,但是，给我们或者用户看到的实际是屁民内容都是一样的，直观区别是，占用系统的空间大小与清晰程度可能不一样

那么，数据库表里的数据存储在数据库里及磁盘上和上述的视频格式

也有很多种存储方式，但是，对于用户和应用程序来说同样一张表的数据，无论用什么引擎来存储，用户看到的数据都是一样的，不同的引擎存取，引擎功能，占用的空间大小，读取性能等可能有区别。

MYSQL最常用引擎为:Myisam和Innodb.全文索引：目前5.5版本，Myisam和innodb都已经支持

## 8.1.2 MYSQL存储引擎的架构

MYSQL的存储引擎是MYSQL数据库的重要组成部分，mysql常用的表的引擎为MYISAM和INNODB两种，MYSQL的每种存储引擎在MYSQL里是通过插件的方式使用的，MYSQL可以同时支持多种

存储引擎，下面是MYSQL存储引擎体系结构简图

MYSQL存储引擎体系结构简图(一).jpg

MYSQL存储引擎体系结构简图(二).jpg

## 8.1.3 Myisam引擎介绍

#### 8.1.3.1 什么是MYISAM引擎

MYISAM引擎是MYSQL关系型数据库管理系统的默认存储引擎(MYSQL5.5以前).这种MYSQL表存储结构从旧的ISAM代码扩展出许多有用的功能，在新版本的MYSQL中，INNODB引擎由于其对事物参照完整性，以及更高的并发性等优点开始逐步取代MYISAM引擎，

查看MYSQL5.1数据库的存储引擎：

范例：

mysql> select version();

mysql> show databases;

mysql> show create table test1\G

提示：MYSQL5.1数据库默认存储引擎为MYISAM.

每一个MYISAM表对应于硬盘的三个文件，这三个文件有一样的文件名，但都具有不同的扩展名指示其类型用途:.frm文件保存表的定义，这个文件并不是MYISAM引擎的一部分，而是服务器的一部分，.MYD保存表的数据，.MYI是表的索引文件，.MYD和.MYI是MYISAM的关键点。

范例：

ll /data/3306/data/oldboy/test1.\*

## 8.1.3.2 MYISAM引擎特点

1)不支持事务(事务是指逻辑上的一组操作，组成这组操作的各个单元，要么全成功，要么全失败。)

2)表级锁定(更新时锁整个其锁定机制是表级索引，这虽然可以让锁定的实现成本很小但是大大降低了其并发性能

3)读写互相阻塞:不仅会再写入的时候阻塞读取,MYISAM还会在读取的时候阻塞写入，但读本身并不会阻塞另外的读

4)只会缓存索引：MYISAM可以通过key\_buffer\_size缓存索引，以大大提高访问性能，减少磁盘IO，但是这个缓存区只会缓存索引，而不会缓存数据

grep key\_buffer my.cnf

key\_buffer\_size = 16M

5)读取速度较快，占用资源相对少

6)不支持外键约束，但不支持全文索引

7)MYISAM引擎是MYSQL5.5版本前的缺省引擎

## 8.1.4 MYISAM引擎适合的生产业务场景

1)不需要事务支持的业务(例如转账就不行，充值付款)，一般为读数据比较多的应用读写多频繁不适合，读多或者写多的都适合

2)并发相对较低的业务(纯度纯写高并发也可以)(锁定机制问题)

3)数据修改相对较少的业务

4)以读为主的业务，例如：www blog 图片信息数据库，用户数据库，商品库等业务。

5)对数据一致性要求不是非常高的业务

6)硬件资源比较差的机器都可用MYISAM

小结：单一数据库的操作都可以使用MYISAM，所谓单一就是尽量纯读，或者纯写(insert,update,delete)等

## 8.1.5 MYISAM引擎调优精要

1)设置合适的索引(缓存机制)

2)调整读写优先级，根据实际需求确保重要操作更优先执行

3)启用延迟插入改善大批量写入性能(降低写入频率，尽可能多条数据一次性写入)

4)尽量顺序操作让insert数据都写入尾部，减少阻塞

5)分解大的操作，降低单个操作的阻塞

6)降低并发数(减少对MYSQL访问)，某些高并发场景通过应用排队队列机制Q队列

7)对于相对静止(更改不频繁)的数据库数据，充分利用Query Cache或memcached缓存服务可以极大的提高访问效率

grep query my.cnf

query\_cache\_size = 2M

query\_cache\_limit = 1M

query\_cache\_min\_res\_unit = 2k

8)MYISAM的COUNT只是在全表扫描的时候特别高效，带有其他条件的COUNT都需要进行实际的数据访问

mysql> select count(\*) from oldboy.student;

9)把主从同步的主库使用Innodb，从库使用MYISAM引擎(不推荐)

# 8.2 数据库事务

## 8.2.1 数据库事务介绍

简单的说，事务就是指逻辑上的一组SQL语句，组成这组操作的各个SQL语句，要么全成功要么全失败。

例如:oldboy给oldgirl转账5块钱，流程如下

a:从oldboy银行卡取出5块钱，计算式money-5

b:把上面5块钱打入oldgirl的账户上，oldgirl收到5块钱，money+5

上述转账的过程，对应的SQL语句为

update oldboy\_account set money=money-5 where name='oldboy';

update oldgirl\_account set money=money+5 where name='oldgirl';

上述的两条SQL操作，在事务中的操作要么都执行，要么都不执行，这就是事物的原子性,MYSQL5.5支持事务的引擎:innodb/ndb

## 8.2.2 事务的四大特性(ACID)

1)原子性

事务的一个不可分割的单位，事务中的所有SQL等操作要么都发生，要么都不发生

2)一致性

事务发生前和发生后，数据的完整性必须保持一致

3)隔离性

当并发访问数据库时，一个正在执行的事务在执行完毕前，对于其他的会话是不可见的，多个并发事务之间的数据是相互隔离的，还记得备份参数吗？

4)持久性

一个事务一旦被提交，他对数据库中的数据改变是永久的。如果出现错误，事务也不允许撤销，只能通过"补偿性事务"

## 8.2.3 事务的开启

数据默认事务是自动提交的，也就是发一条SQL他就执行一条，如果想多条SQL放在一个事务中执行，则需要使用事务进行处理，当我们开启一个事务，并且没有提交，MYSQL会自动回滚事务，或者我们使用rollback命令手动回滚事务

数据库开启事务的命令：

start transcation开启事务

rollback回滚事务

commit 提交事务

案例一：

mysql -uroot -poldboy -S /data/3306/mysql.sock

mysql> SET GLOBAL autocommit=OFF;

mysql> quit

mysql -uroot -poldboy -S /data/3306/mysql.sock

mysql> use oldboy

mysql> insert into student values(1,'oldboy');

mysql> select \* from student;

mysql> quit

mysql -uroot -poldboy -S /data/3306/mysql.sock

mysql> select \* from oldboy.student

# 8.3 InnoDB引擎

8.3.1 什么是InnoDB引擎

InnoDB引擎是MYSQL数据库的另一个重要的存储引擎，正成为目前MYSQL AB所发行新版的标准，被包含在所有二进制安装包里，和其他的存储引擎相比，InnoDB引擎的优点是支持兼容ACID的事务(类似于PORTGRESQL)，以及参数完整性(即对外键的支持)。ORACLE公司与2005年10月收购了Innobase.Innobase采用双认证授权，它使用GNU发行，业允许其他想将InnoDB结合到商业软件的团体获得授权。

更多参考rfeman-5.5-en.html-chapter/storage-engines.html

MYSQL5.5以后数据库的默认引擎是InnoDB.

范例：

mysql> select version();

mysql> use oldboy;

mysql> create table student;

msyql> show create tables student\G

## 8.3.2 InnoDB引擎的特点

1)支持事务:支持4个事务隔离级别，支持多版本读

2)行级锁定(更新时一般是锁定当前行)：通过索引实现，全表扫描仍然会是表锁

3)读写阻塞与事务隔离级别相关

4)具有非常高的缓存特性：能缓存索引，也能存储数据

5)整个表和主键以Cluster方式存储，组成一个平衡树

6)所有Secondary Index都会保存主键信息

7)支持分区，表空间，类似ORACLE数据库

8)支持外键约束，5.5以前不支持全文索引，以后支持了

9)和MYISAM引擎相比，InnoDB对硬件资源要求比较高

## 8.3.3 InnoDB引擎适用的场景

1)需要事务的支持业务(具有较好的事务特性)

2)行级锁定对高并发有很好的适应能力，但需要确保查询是通过索引完成。

3)数据读写及更新较为频繁的场景，如：BBS,SNS,微博等。

4)数据一致性要求较高的业务，例如：充值转账，银行转账

5)硬件设备内存较大，可以利用InnoDB较好的缓存能力来提高内存利用率，尽可能减少磁盘IO

grep -i innodb my.cnf

[mysqld]

innodb\_addition\_mem\_pool\_size = 16M

innodb\_buffer\_pool\_size = 2048M

innodb\_data\_file\_path = ibdata1:1024:autoextend

innodb\_file\_io\_threads = 4

innodb\_thread\_concurrency = 4

innodb\_flush\_log\_at\_trx\_commit = 2

innodb\_log\_buffer\_size = 16M

innodb\_file\_size = 128M

innodb\_log\_file\_in\_group = 3

innodb\_lock\_wait\_timeout = 120

innodb\_file\_per\_table = 0

innodb\_file\_per\_table

innodb\_data\_home\_dir = /data/xxx

innodb\_log\_group\_home\_dir = /data/xxx

共享表空间对应物理数据文件

ll /data/3306/data/ibdata1

独立表空间对应物理数据文件

innodb\_file\_per\_table

innodb\_data\_home\_dir = /data/xxx

innodb\_data\_file\_path = ibdata1:1024:autoextend

6)相比MYISAM引擎，innodb引擎更消耗资源，速度没有MYISAM引擎快

## 8.3.4 InnoDB引擎调优精要

1) 主键尽可能小，避免Secondary index带来过大的空间负担

2) 避免全表扫描，因为会使用表锁

3) 仅可能缓存所有的索引和数据，提高响应速度，减少磁盘IO消耗

4) 在大批量小插入的时候，仅可能自己控制事务而不要用autocommit自动提交，有开关可以控制提交方式

5) 合理设置innodb\_flush\_log\_at\_trx\_commit参数值，不要过度追求安全性。如果innodb\_flush\_log\_at\_trx\_commit的值为0,log buffer每秒就会被刷写日志文件到磁盘，提交事务的时候不会做任何操作

6)避免主键更新，因为这回带来大量数据移动

## 8.3.5 如何确定MYSQL服务器又那些引擎可用

你可以在MYSQL中使用显示引擎的命令得到一个可用引擎的列表

mysql> show engines\G;

以上的输出结果显示了可用的数据库引擎的全部名单以及在当前的数据库服务器中是否支持这些引擎

## 8.3.6 MYSQL引擎的my.cnf对应参数说明

InnoDB引擎重要参数

innodb\_addition\_mem\_pool\_size = 16M

innodb\_buffer\_pool\_size = 2048M

innodb\_data\_file\_path = ibdata1:1024:autoextend

innodb\_file\_io\_threads = 4

innodb\_thread\_concurrency = 8

innodb\_flush\_log\_at\_trx\_commit = 2

innodb\_log\_buffer\_size = 16M

innodb\_file\_size = 128M

innodb\_log\_file\_in\_group = 3

innodb\_max\_dirty\_pages\_pct = 90

innodb\_lock\_wait\_timeout = 120

innodb\_file\_per\_table = 0

innodb\_file\_per\_table

innodb\_data\_home\_dir = /data/xxx

innodb\_log\_group\_home\_dir = /data/xxx

参数的解释，可以参考官方的手册

MYISAM 引擎重要参数：

key\_buffer\_size = 2048M

## 8.3.7 生产环境中如何批量更改MYSQL引擎

## 一般来说这样的需求不多见，但偶尔也会有，在这里我么推荐使用sed对备份内容进行引擎转换的方式，当然了，不要忘记了修改my.cnf使之支持并能够高效的使用对应的引擎

方法一：MYSQL命令语句的修改

创建表后引擎的更改，5.0以上

ALTER TABLE oldboy ENGINE = INNODB

ALTER TABLE oldboy ENGINE = MyISAM

方法二：使用sed对备份内容进行引擎转换

mysqldump > oldboy\_1.sql

nohub sed -e 's#MyISAM#InnoDB#g' old2boy.sql > oldboy\_1.sql &

mysql <oldboy\_1.sql

提示：数据量大，该方法效果不好

方法三 mysql\_convert\_table\_format 命令修改

mysql\_convert\_table\_format --host=10.0.0.107 --user=root --password=

oldboy123 --sokcet=/data/3306/mysql.sock --engine=MyISAM oldboy student批量修改数据库引擎脚本

## 8.4 有关MYSQL引擎常见企业面试题

1)MYSQL有哪些引擎，各自有什么特点和区别

2)生产场景如何选用MYSQL引擎

## 9.1 网站打开慢及MYSQL慢查询优化的案例

## 9.1.1 MYSQL慢查询优化过程

方法一

mysql -uroot -poldboy -S /data/3306/mysql.sock

mysql> SHOW FULL PROCESSLIST; #通过该命令来观察查询语句的执行情况

方法二：在MYSQL的配置文件里开启慢查询的日志文件功能

cat /data/3306/my.cnf

#超过一秒的查询日志都记录到日志文件中

long\_query\_time = 1

log-slow-queries = /data/3306/slow.log

==============================================

1)通过uptime命令观察到当前服务器的负载很高

#uptime

13:50:37 up 120 days, 21:21 1 user, load average: 6.14,5.76,5.38

#uptime

13:51:39 up 120 days, 21:22 1 user, load average: 8.01,6.30,5.58

2)登陆MYSQL数据库后，查看到当前有大量的线程处于等待状态

mysql -uroot -poldboy123 -S /data/3306/mysql.sock

mysql> show full processlist;

Id User Host db command Time State Info

212542838 root localhost advertise Query 0 NULL show processlit

212555221 root localhost advertise Query 0 Locked update .....

212555222 root localhost advertise Query 0 cleaning up select ....

212555223 root localhost advertise Query 0 cleaning up select ....

212555225 root localhost advertise Query 0 cleaning up update ....

.... ... ... ... ... .. ... ...

3)通过查看MYSQL服务器线程，找到需要优化的MYSQL语句比如：找到以下语句需要进行索引优化

select id from ad\_oldboy\_detail where ader='ibm\_esd-jazz\_flash' and deteline='2008-12-22' and pos=developer\_banner';

查看利用索引情况：

explain SQL\_NO\_CACHE select id from ad\_oldboy\_detail where ader='ibm\_esd-jazz\_flash' and deteline='2008-12-22' and pos=developer\_banner';

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

id: 1

select\_type: SIMPLE

table: ad\_oldboy\_detail

type: ALL

possible\_keys: NULL

key: NULL

key\_len: NULL

ref: NULL

rows: 13238 全表扫描，没有任何索引

Extra: Using where

1 row in set (0.0.4 sec)

根据以上查看结果，可以看到全表仅仅扫描了13238行，就出现了慢查询，需要对该语句添加

索引功能

4）查看表结构，确定在那些表字段上添加索引

mysql> SHOW CREATE TABLE ad\_oldboy\_detail\G

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

TABLE: ad\_oldboy\_detail

Create Table: CREATE TABLE `ad\_oldboy\_detail` (

`id` int(10) unsigned NOT NULL auto\_increment,

`ader` varchar(100) default NULL,

`pos` varchar(100) default NULL,

`clicks` int(10) default '0',

`view` int(10) default '0',

`deteline` varchar(20) default NULL

PRIMARY KEY ('id') #只有一个主键索引

) ENGINE=MyISAM AUTO\_INCREMENT=24521 DEFAULT CHARSET=latin1 COMMENT'?????????'

1 row in set (0.0.3 sec)

5) 查看条件字段列的唯一性

select count(distinct ader) from ad\_oldboy\_detail;

--------------------

count(distinct ader)

154

--------------------

1 row in set(0.0.8 sec)

select count(distinct dateline) from ad\_oldboy\_detail;

--------------------

count(distinct ader)

531

--------------------

1 row in set(0.0.8 sec)

select count(distinct pos) from ad\_oldboy\_detail;

--------------------

count(distinct ader)

421

--------------------

1 row in set(0.0.8 sec)

select count(\*) from ad\_oldboy\_detail;

--------------------

count(\*)

--------------------

13239

--------------------

1 row in set(0.0.0 sec)

6)根据以上和咨询研发经理其他语句的情况后，创建如下索引。

mysql> create index d\_a\_p on ad\_oldboy\_detail(deteline,ader(20),pos(20));

提示：在生产环境中如果访问频繁的大表，创建索引会很耗费时间，也许需要几分钟

应该在业务流量低谷时建立索引

mysql> explain SQL\_NO\_CACHE select id from ad\_oldboy\_detail where ader='ibm\_esd-jazz\_flash' and deteline='2008-12-22' and pos=developer\_banner';

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

id: 1

select\_type: SIMPLE

table: ad\_oldboy\_detail

type: ref

possible\_keys: d\_a\_p

key: d\_a\_p

key\_len: 63

ref: const,const,const #完全利用上了索引

rows: 12 全表扫描

Extra: Using where

1 row in set (0.0.0 sec)

7)建立索引后，查看MYSQL线程，几乎看不多了慢查询记录了，说明效果还是很明显的

mysql> show full processlist;

Id User Host db command Time State Info

212585564 root localhost advertise Query 0 NULL show processlit

8)优化后，查看服务器负载情况，负载已经很小了

#uptime

13:59:09 up 120 days, 21:29 1 user, load average: 2.40,4.62,5.09

#uptime

13:59:29 up 120 days, 21:29 1 user, load average: 1.72,4.32,4.98

## 9.1.2 Memcached服务应用优化案例

案例：某公司案例:网站打开慢

数据库负载很高，load值20-30

#uptime

10:41:01 up 15:02, 4 users, load average: 20,22,27

1)登陆数据库通过命令show full processlist查看数据库执行过程中的线程数

mysql -uroot -p'1111' -e "show full processlist" | grep -vi sleep

说明：通过该命令查看数据库执行的线程数发现数据库中像LIKE'%杜冷丁%这样的语句特别多，导致数据库负载很高，LIKE'%杜冷丁%这样的语句对于数据库没有太大的优化余地 ，我打开网站首页看了一下，应该是首页的搜索框搜索带来的结果,根据公司业务和我个人的理解，给出如下优化思路，供领导参考

优化方案思路：

1. 从业务上实现用户登录后再搜索，这样减少搜索次数，从而减轻数据库服务的压力。
2. 如果有大量频繁的搜索，一般是由爬虫在爬你的网站，分析WEB日志IP封掉之(AWSTATS)

3)配置多个主从同步，程序上实现读写分离(LIKE'%杜冷丁%这样的语句去从库上查询)，减轻主库读写压力

4)在数据库前段加上memcached缓存服务器

5)LIKE'%杜冷丁%这样的语句，一般在MYSQL里很难优化，可以通过搜索服务Sphinx实现搜索

6)当然还可以利用C,ruby开发程序，实现每日读库计算搜索索引，保证在服务器上提供搜索，然后，每5分钟多一个从库做一次增量。

这是大公司对站内搜索采取的比较好的方案

从短期来看，我更倾向于1和2方案，简单，好实施，3和4是后期要发展的目标。 提示：LIKE'%杜冷丁%这样的SQL语句是不通过索引查询的，所以对数据库的优化没有太大的作用，如果是LIKE '杜冷丁%有可能通过索引查询

## 9.2 MYSQL数据库安全权限控制管理思想

9.2.1 制度与流程控制

9.2.1.1 项目开发制度流程

办公开发环境---->办公测试环境----->IDC测试环境---->IDC正式环境。通过这种完善的项目开发制度及流程控制，尽可能的防止潜在的问题隐患发生

### 9.2.1.2 数据库更新流程

开发人员提交需求---->开发主管审核--->部门领导审核----->DBA(运维)审核--->DBA(运维)执行项目开发制度及流程控制的数据库更新步骤(每个步骤都要测试)，最后在IDC正式环境执行

需要说明的是，在开发人员一开始提交需求时，就可以同时抄给以上的领导及审核人员，然后，审核人员依次审核，对于特殊紧急需求，可以根据紧急程度特殊处理，这里可以指定个紧急处理流程，比如：开发人员提交需求---->DBA(运维)审核，然后操作完后汇报个其他的审核人员

通过完善的数据库更新控制流程，可以防止很多潜在的数据丢失，破坏的问题发生

## 9.2.3 DBA参与项目数据库设计

在项目开发环节上，DBA或资深运维人员最好参考数据库设计与审核工作，这样可以从源头上减少降低不良的数据库设计及不良SQL语句的发生，还可以做所有语句的审核工作，包括select，但这个需要评估工作量是否允许，一般的互联网公司实施全审核比较困难

## 9.2.4 各种操作申请流程

1)开发等人员权限申请流程

2)数据库更新执行流程

3)烂SQL语句计入KPI考核

## 9.2.5 定期对内部人员培训

定期给开发及相关人员培训，目的还是从源头上降低不良数据库设计及不良SQL语句的发生，并通过培训让大家知晓大家数据库性能的重要性，让他提升开发时照顾数据库性能的意识

# 9.3 账户权限控制

## 9.3.1 内部开发等人员权限分配

1)权限申请流程要设置规范，合理，让需求不明确者知难而退

2)办公开发和测试环境可以放开权限，IDC测试和正式环境要严格控制数据库写权限，并且读权限和对外业务服务分离

3)开发人员正式环境数据库权限分配规则：给单独的不对外服务的正式从库只读权限，不能分配线上正式业务服务分离

4)特殊人员(如领导)，需要权限时，我们要问清楚他做什么，发邮件回复，注明用户名，密码，权限范围，多提醒操作注意事项，如果有可能由DBA人员代替其操作5)特权账号(all privliges)，由DBA控制，禁止在任何客户端上执行特权账号操作(如只能localhost或其他策略)

## 9.3.2 web账户权限分配制度

1)写库账号默认权限select,insert,update,delete,不要给建表改表(create,alter)等权限，更不能使用ALL权限

2)读库账号默认权限为select(配合mysql read-only参数用)。确保从库对所有非SUPER权限是只读的

3)最好专库专账户，不要一个账户管理多个库，碎库特别多的小公司更加情况特殊对待处理

4)如果是LAMP,LNMP一体化在一台服务器的环境，DB权限主机要设置为localhost，避免用root作为web的连接用

## 9.3.3 数据库客户端访问控制的策略

1)更改默认mysql client端口，如phpmyadmin管理端口为9999，其他客户端也一样

2)数据库web client端统一部署在1-2台不对外服务的web上，限制IP及9999端口只能从办公室内网访问

3)不做公网域名解析，用host实现访问(限制任何IP直接访问)或者用内部IP访问

4)phpadmin站点目录独立于所有其他站点根目录下，只能由指定的域名或IP地址访问

5)限制使用web连接的账号管理数据库，根据开发人员用户角色分配指定账户访问。

6)按开发及相关人员根据职位角色分配适合的管理账号

7)设置指定账户访问权限层次，WEB层使用apache/nginx账号验证，数据库层使用MYSQL用户登录验证。

8)统一所有数据库账号登陆地址，禁止所有开发人员私自上传phpadmin等数据库管理的程序。

9)开通VPN，跳板机，只能通过局域网内部IP管理数据库

## 9.3.4 数据库管理思想的核心

1)未雨绸缪，不要停留在制度上，而是，实际做出来

2)亡羊补牢，举一反三，切记，不要好了伤疤忘了疼，

3)完备的架构设计及备份，恢复策略

4)定期思考，并实战模拟以上策略演练

# 9.4 MYSQL数据库的优化

1)硬件的优化：物理机

a:一台机器8-16颗，2-4颗

b:mem 96G-128G,一台服务器跑3-4个MYSQL数据库实例 32-64G 跑2个实例

c:硬盘是数量越多越好，从性能上讲ssd(高并发) >ssa(普通业务上) >sata(线上),如果是高并发场景，用ssd硬盘，raid 0和raid 10(raid0>raid10>raid5>raid1)

d:网卡 多块网卡bond，以及buffer,tcp优化

2)软件优化：

a:操作系统：X86\_64

b:软件:mysql 编译优化。

3)my.cnf里参数的优化

优化的幅度很小，大部分是架构以及SQL语句的优化,要做监控，生产参数是一般情况下的参数，如果不合适，还有进行调整可以通过show global status\G命令来观察需要调整那些参数,也可以通过mysqlreport来自动计算参数的值的合理性(调优工具)

http://www.day32.com/MySQL/tuning-primer.sh

4)SQL语句的优化

a:索引的优化

1)抓出慢SQL,需要配置my.cnf,或者使用白名单机制-百度，项目开发，DBA参与，减少上线后的慢SQL数量

long\_query\_time = 2

log-slow-queries=/data/3306/slow-log.log

按天轮询 slow-log.log

2)慢查询日志分析工具---mysqlsla

mysqldumpslow,mysqlsla,myprofile,mysql-explain-slow-log,mysqllogfilter

3)每台晚上0点定时分析慢查询，发到开发，DBA分析,及高级运维，CTO的邮箱里,DBA分析给出优化建议-->核心开发确认更改--->DBA线上操作处理

b:大的复杂的SQL语句拆分多个小的SQL语句

子查询，JOIN连表查询需要拆分SQL语句，大表(某个表大于4000万条记录)需要拆库拆表

c:数据库是存储数据库的地方，但不是计算数据的地方

对数据计算，应用类处理，都要拿到前端应用解决，禁止在数据库上处理

d:搜索的功能，

比如 like %wp%,一般不要用MYSQL数据库

5)架构上的优化

1：从业务上进行拆分，比如 like %wp%,一般不要用MYSQL数据库

2：某些业务用用使用nosql持久化存储，例如：memcachedb,redis,ttserver

(比如像：粉丝关注，好友关系)

3：数据库前端必须要加缓存(比如可以在数据库前端添加memcached服务器)

(比如：用户登录，商品查询 )

4：把动态的数据静态化,(整个文件静态化，页面片段静态化)

5: 数据库集群与读写分离(一主多从，通过程序或者dbproxy进行集群读写分离)

6: 如果出现单表存储超过2000万以上，拆库拆表，人工拆表，拆库(登陆，商品，订单)

7：大公司会这样搞

6)流程，制度，安全优化

任何一次人为数据库记录的更新，都要走流程，

a:人的流程:(开发--->核心开发---->运维或DBA)

b:测试流程:(内网测试---->IDC测试---->线上执行)

c:客户端的管理(phpmyadmin)

# 9.5 高手分享

9.5.1 通过PHP应用程序实现MYSQL数据库读写分离

1)通过程序判断SQL语句实现读写分离

2)通过调用方法实现读写分离

## 9.5.2 amoeba软件实现数据库读写分离实战

### 9.5.2.1 服务器架构规划

服务器名 IP地址 服务器角色

C65-1 10.0.0.7 负载均衡服务器 amoeba

C65-2 10.0.0.8 主从MYSQL服务器，MYSQL多实例

3306作为主服务器，3307作为从服务器

### 9.5.2.2 amoeba读写分离示意图

amoeba读写分离示意图.jpg

### 9.5.2.3 MYSQL主从复制的配置

1)在C65-2上实现MYSQL的主从复制的配置步骤

mysql -uroot -poldbo123 -S /data/3306/mysql.sock

mysql> show master status; #通过该命令查看到当前binlog的文件名和位置(pos)记录下当前的文件名和位置的值

登陆从库上查看是否配置了主从服务

mysql -uroot -poldboy123 -S /data/3307/mysql.sock

mysql> show slave status\G

登陆到主库上创建用于主从复制的用户

mysql -uroot -poldbo123 -S /data/3306/mysql.sock

mysql> grant replication slave on \*.\* 'backup'@'10.0.0.%' identified by 'redhat'

mysql> flush privileges;

mysql> quit

2)登陆到从库上创建主从复制

mysql -uroot -poldboy123 -S /data/3307/mysql.sock

mysql> slave stop;

mysql>change master to master\_host='10.0.0.18',master\_user='backup' master\_password='redhat',master\_log\_file='mysql-bin.000027',master\_log\_pos=343;

mysql> slave start;

mysql> show slave status\G;

3)测试主从复制功能

mysql -uroot -poldboy123 -S /data/3306/mysql.sock

mysql> create roudy;

mysql -uroot -poldboy123 -S /data/3307/mysql.sock

mysql> show databases;

### 9.5.2.4 MYSQL读写分离功能的实现

1)下载和安装JDK

wget http:/www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html

2)配置授权用户,分别在主\从服务器上添加一个授权用户amoeba

mysql -uroot -poldboy123 -S /data/3306/mysql.sock

mysql> grant select,insert,update,delete on \*.\* to amoeba@'192.168.100.%' identified by 'redhat';

mysql> flush privileges;

如果开启了主从同步后，这时当主服务器创建一个用户后会同步到从服务器，所以我们要在从服务器上删除主服务器建立的用户后，再新建用户

mysql -uroot -poldboy123 -S /data/3307/mysql.sock

mysql> delete from mysql.user where user='amoeba';

mysql> grant select on \*.\* to amoeba@'192.168.100.%' identified by 'redhat';

mysql> flush privileges;

mysql> quit

配置JDK功能

rpm -ivh jdk-7u45-linux-x64.rpm

ln -s /usr/java/jdk1.7.0\_45/ /usr/java/jdk1.7

vim /etc/profile

export PATH=/application/mysql/bin:$PATH

export JAVA\_HOME=/usr/java/jdk1.7

export PATH=$JAVA\_HOME/bin:$JAVA\_HOME/jre/bin:$PATH

source /etc/profile

故障解决：

JAVA环境变量配置错误导致系统命令无法使用

解决办法

使用/bin/vim /etc/profile命令进入系统变量配置文件中删除配置错误的JAVA环境变量的相关配置，然后使用source /etc/profile,重启linux操作系统后，故障解决

wget http://blogimg.chinaunix.net/blog/upfile2/101027160252.zip

mkdir /usr/local/amoeba

mv 101027160252.zip /usr/local/amoeba/

cd /usr/local/amoeba

unzip 101027160252.zip

rm -rf 101027160252.zip

cd conf/

修改amoeba的主配置文件内容

vim amoeba.xml

<?xml version="1.0" encoding="gbk"?>

<!DOCTYPE amoeba:configuration SYSTEM "amoeba.dtd">

<amoeba:configuration xmlns:amoeba="http://amoeba.meidusa.com/">

<server>

<property name="port">3306</property>

<property name="ipAddress">192.168.100.51</property>

<property name="readThreadPoolSize">20</property>

<property name="clientSideThreadPoolSize">30</property>

<property name="serverSideThreadPoolSize">30</property>

<property name="netBufferSize">128</property>

<property name="tcpNoDelay">true</property>

<property name="user">root</property>

<property name="password">mysql</property>

<property name="queryTimeout">60</property>

</server>

<connectionManagerList>

<connectionManager name="defaultManager" class="com.meidusa.amoeba.net.MultiConnectionManagerWrapper">

<property name="subManagerClassName">com.meidusa.amoeba.net.AuthingableConnectionManager</property>

<!--

default value is avaliable Processors

<property name="processors">5</property>

-->

</connectionManager>

</connectionManagerList>

<dbServerList>

<dbServer name="server1">

<factoryConfig class="com.meidusa.amoeba.mysql.net.MysqlServerConnectionFactory">

<property name="manager">defaultManager</property>

<property name="port">3306</property>

<property name="ipAddress">192.168.100.50</property>

<property name="schema">test</property>

<property name="user">amoeba</property>

<property name="password">redhat</property>

</factoryConfig>

<poolConfig class="com.meidusa.amoeba.net.poolable.PoolableObjectPool">

<property name="maxActive">200</property>

<property name="maxIdle">200</property>

<property name="minIdle">10</property>

<property name="minEvictableIdleTimeMillis">600000</property>

<property name="timeBetweenEvictionRunsMillis">600000</property>

<property name="testOnBorrow">true</property>

<property name="testWhileIdle">true</property>

</poolConfig>

</dbServer>

<dbServer name="server2">

<factoryConfig class="com.meidusa.amoeba.mysql.net.MysqlServerConnectionFactory">

<property name="manager">defaultManager</property>

<property name="port">3307</property>

<property name="ipAddress">192.168.100.50</property>

<property name="schema">test</property>

<property name="user">amoeba</property>

<property name="password">redhat</property>

</factoryConfig>

<poolConfig class="com.meidusa.amoeba.net.poolable.PoolableObjectPool">

<property name="maxActive">200</property>

<property name="maxIdle">200</property>

<property name="minIdle">10</property>

<property name="minEvictableIdleTimeMillis">600000</property>

<property name="timeBetweenEvictionRunsMillis">600000</property>

<property name="testOnBorrow">true</property>

<property name="testWhileIdle">true</property>

</poolConfig>

</dbServer>

<dbServer name="master" virtual="true">

<poolConfig class="com.meidusa.amoeba.server.MultipleServerPool">

<property name="loadbalance">1</property>

<property name="poolNames">server1</property>

</poolConfig>

</dbServer>

<dbServer name="slave" virtual="true">

<poolConfig class="com.meidusa.amoeba.server.MultipleServerPool">

<property name="loadbalance">1</property>

<property name="poolNames">server2</property>

</poolConfig>

</dbServer>

</dbServerList>

<queryRouter class="com.meidusa.amoeba.mysql.parser.MysqlQueryRouter">

<property name="ruleConfig">${amoeba.home}/conf/rule.xml</property>

<property name="functionConfig">${amoeba.home}/conf/functionMap.xml</property>

<property name="ruleFunctionConfig">${amoeba.home}/conf/ruleFunctionMap.xml</property>

<property name="LRUMapSize">1500</property>

<property name="defaultPool">master</property>

<property name="writePool">master</property>

<property name="readPool">slave</property>

<property name="needParse">true</property>

</queryRouter>

</amoeba:configuration>

启动amoeba服务器

chmod -R 700 /usr/local/amoeba/bin

/usr/local/amoeba/bin/amoeba &

测试读写分离

在从库上关闭主从复制

mysql> mysql -uroot -poldboy123 -S /data/3307/mysql.sock

mysql> slave stop;

建立一个数据库和表，对表进行插入数据操作，观察主库和从库的变化

mysql -uroot -poldboy123 -h10.0.0.8

mysql> create database oldboy;

mysql> use oldboy;

mysql> create table student(id int(4) not null AUTO\_INCREMENT,name char(20) not null,primary key(id));

mysql> insert into student(name) values('oldboy101')

mysql> insert into student(name) values('oldboy102')

mysql> quit

mysql -uroot -poldboy123 -S /data/3306/mysql.sock

mysql> use oldboy

mysql> select \* from student;

如果配置正确，只能从主库上看到插入的记录，从库是无记录的开启从库上的主从服务按钮

mysql> mysql -uroot -poldboy123 -S /data/3307/mysql.sock

mysql> slave start;

mysql> show slave status\G;

mysql> use oldboy

mysql> select \* from student;

如果配置正确，从主库和从库上都能看到插入的记录

9.5.2.5 MYSQL\_proxy实现数据库的读写分离实战

一:什么是mysql.proxy

mysql-proxy是一个通过网络利用MYSQL的网络协议，并且提供一个或多个MYSQL服务器与一个多个MYSQL客户端相互沟通的应用程序，因为MYSQL PROXY使用MYSQL网络协议，所以他兼容任何MYSQL客户端并且无需修改除了基本的传递，MYSQL PROXY可以在查询队列发送到服务器之前插入一些查询请求，也可以在服务器的应答中将对应的应答删除，这个功能可以使得管理员可以对每个查询进行跟踪并获取报告，例如：监测其执行时间或其他调试信息，并分别记录结果，同时还能将正确的应答返回给客户端.虽然MYSQL PROXY功能强大，但目前仍然处于alpha版本，存在许 多问题，不建议生产环境使用

二：MYSQL PROXY 形象图

三：MYSQL PROXY 安装

在安装前先说明一下，MYSQL PROXY的安装会遇见各种各样的问题，遇到问题可以搜，也可以找我问，要是搭建没有这个耐心，建议还是别折腾了。

官方给出软件版本需求

libevent 1.x or higher(1.3 or later is perferred)

lua 5.1x or higher

glib2.2.6.0 or higher

pkg-config

libtool 1.5 or higher

mysql 5.0x or higer developer files

四 安装基础依赖包

#直接YUM安装即可，方便快捷

yum -y install gcc gcc-c++ autoconf libevent-devel pkgconfig libtool mysql-devel

3：安装glib

#下载glib源码包并编译安装

mkdir /home/oldboy/tools

cd /home/oldboy/tools

wget http://ftp.gnome.org/pub/gnome/sources/glib/2.22/glib-2.22.5.tar.gz

tar zvxf glib-2.22.5.tar.gz

cd glib-2.22.5

./configure --prefix=/usr/local/glib-2.22.5

make && make install

4:安装LUA

#mysql proxy使用lua语言编写相应脚本，编译安装

wget http://www.lua.org/ftp/lua-5.1.4.tar.gz

tar zxvf lua-5.1.4.tar.gz

cd lua-5.14

sed -i 's#^INSTALL\_TOP=.\*#INSTALL\_TOP=/usr/local/lua-5.1.4#gi' ./Makefile

如果是64位系统需要执行

sed -i 's#^CFLAGS=.\*#CFLAGS= -02 -fPIC -Wall $(MYCFLAGS)#gi' ./src/Makefile

make linux install

mkdir /usr/local/lua-5.1.4/lib/pkgconfig

cp ./etc/lua.pc /usr/local/lua-5.1.4/lib/pkgconfig

sed -i 's#^prefix=.\*#prefix= /usr/local/lua-5.1.4/#gi' /usr/local/lua-5.1.4/lib/pkgconfig/lua.pc

export

PKG\_CONFIG\_PATH=$PKG\_CONFIG\_PATH:/usr/local/glib-2.22.5/lib/pkconfig:

/usr/local/lua-5.1.4/lib/pkgconfig

5)安装mysql proxy

mkdir /application

./configure --prefix=/application/mysql-proxy-0.8.2

make && make install

mkdir -p /application/mysql-proxy-0.8.2/lib/mysql-proxy/lua/

cp ./lib/rw-splitting.lua /application/mysql-proxy-0.8.2/lib/mysql-proxy/lua/ #rw-splitting.lua是用来实现读写分离的脚本

#建立mysql proxy配置文件存放目录

mkdir -p /application/mysql-proxy-0.8.2/conf

#建立mysql proxy日志存放目录

mkdir -p /application/mysql-proxy-0.8.2/logs/

cat >/application/mysql-proxy-0.8.2/conf/mysql-proxy.cnf<<EOF

[mysql-proxy]

plugins=admin,proxy

admin-username=admin

admin-password=admin

admin-lua-script=/application/mysql-proxy-0.8.2/lib/mysql-proxy/lua/admin.lua

proxy-backend-addresses=10.0.0.18:3306 #读写库的配置

proxy-read-only-backend-addresses=10.0.0.18:3307 #只读库

proxy-lua-script = /application/mysql-proxy-0.8.2/lib/mysql-proxy/lua/rw-splitting.lua

log-file=/application/mysql-proxy-0.8.2/logs/proxy.log

log-level=debug

daemon=true

keepalive=true

EOF

6)启动mysql-proxy服务

chmod 0660 /application/mysql-proxy-0.8.2/conf/mysql-proxy.cnf

/application/mysql-proxy-0.8.2/bin/mysql-proxy --defaults-file=/application/mysql-proxy-0.8.2/conf/mysql-proxy.cnf &

netstat -lnt

7)测试读写分离的效果

mysql -uadmin -padmin -h 10.0.0.18 -P 4041

mysql> select \* from backends

mysql>

cd /application/mysql-proxy-0.8.2

vim ./lib/mysql-proxy-0.8.2/lua/rw-splitting.lua

将min\_idle\_connections和max\_idle\_connections两个参数的值都修改成"1".

/application/mysql-proxy-0.8.2/bin/mysql-proxy --defaults-file=/application/mysql-proxy-0.8.2/conf/mysql-proxy.cnf &

常见故障解决

mysql提示Can't find messagefile/usr/share/mysql/english/errmsg.sys

解决办法:

1、查看本机安装的mysql版本，然后到http://downloads.mysql.com/archives.php寻找下载对应的mysql版本，解压后，将mysq-x.x.x/sql/share/english/errmsg.sys拷贝到系统的/usr/share/mysql/english/下，覆盖同名errmsg.sys文件（建议覆盖前备份一份）即可。

2.mysql\_instal\_db

3.service mysqld start