# Netkiller MySQL 手札

陈景峰 著

MySQL®

# Netkiller MySQL 手札

目录

安装 XtraBackup
innobackupex
备份数据库
恢复数据库
xbstream
xtrabackup
Percona Toolkit - MySQL Management Software
2. MySQL Plugin
2.1. validate_password
2.2. MySQL Images manager
2.3. MySQL fifo
2.4. 内容输出到文本插件
3. Replication
3.1. Master Slave
Master
Slave
Testing
将现有数据库迁移到主从结构数据库
主从复制安全问题
3.2. Master Master(主主)
Master A
Master B
将Master A 数据库 同步到 Master B 两端数据库内容保持一致
Master A - B 同步两端数据库
Master A 数据库解除只读权限
查看主主的工作状态
3.3. Semisynchronous Replication
Master
Slave 配置
卸载插件
my.cnf
3.4. multi-master replication
3.5. multi-source replication
3.6. 与复制有关的问题
主从不同步问题

1. mysql - the MySQL command-line client
    1.1. ~/.my.cnf
    1.2. 屏幕输出到文件
    1.3. 终端编码
    1.4. Unix Socket
    1.5. 重定向巧用
    1.6. --sigint-ignore 忽略 Ctrl + C
    1.7. mysql log
2. mysqldump - a database backup program
    2.1. 备份数据库并压缩文件
    2.2. 备份数据库/表
    2.3. 备份到文件
    2.4. 备份数据库，无结构，只有数据
    2.5. 使用完整的insert插入数据
    2.6. --extended-insert / --skip-extended-insert
    2.7. --skip-lock-tables
    2.8. --skip-add-locks
    2.9. --where
    2.10. 注释信息--comments /--skip-comments
    2.11. 不导出注释信息
    2.12. 字符集设置
3. mysqladmin - client for administering a MySQL server
    3.1. reload
    3.2. 更改密码
    3.3. status
    3.4. process list
4. myisamchk — MyISAM Table-Maintenance Utility
5. mysqlcheck — A Table Maintenance and Repair Program
6. mysqlslap - load emulation client
7. mysqldumpslow - Parse and summarize the MySQL slow query log.
8. mysql-shell
9. 
10. MySQL慢查询日志（Slow Query Log）
    10.1. MySQL 8.x
    10.2. MySQL 5.x
        查看设置

开启慢查询日志
配置慢查询日志
删除慢查询日志
11. mysql-admin
12. MySQL Workbench 数据库恢复

3. 数据库管理（Database Administration)
    1. 用户管理（User Account Management）
        1.1. 创建用户
        1.2. 删除用户
        1.3. 修改用户名
        1.4. 修改密码
    2. Access Privilege System
        2.1. SHOW GRANTS
        2.2. show privileges
        2.3. Grant privileges
        2.4. Revoke privileges
        2.5. Show Privileges
        2.6. MAX_QUERIES_PER_HOUR/MAX_UPDATES_PER_HOUR
        2.7. Table Privileges
        2.8. Column Privileges
    3. 字符集转换
        3.1. 转换 latin1 到 UTF-8
    4. 重新整理AUTO_INCREMENT字段
    5. 数据库内容替换
    6. Kill 脚本
    7. MySQL 时区管理
    8. SHOW COMMAND
        8.1. 查看版本
        8.2. status
            show status
            show master status
            show slave status
            show plugins
        8.3. show processlist

        LVM Snapshot
        Btrfs Snapshot
4. DDL - Data Definition Language
    1. 数据库管理(Database)
        1.1. 创建数据库
        1.2. 删除数据库
        1.3. 修改数据库
        1.4. 重命名数据库
        1.5. 修改字符集
        1.6. 查看数据库创建语句
    2. 表管理(Table)
        2.1. 数据类型
            SET 集合类型
        2.2. 基于现有表结构创建新空表
        2.3. 基于已存在表创建新表
        2.4. 修改表
        2.5. 临时表
        2.6. CHARACTER
        2.7. DEFAULT
            AUTO_INCREMENT
            TIMESTAMP NULL DEFAULT NULL ON UPDATE
            表存储位置(DATA DIRECTORY)
        2.8. KEY
            PRIMARY KEY
        2.9. AUTO_INCREMENT 定义初始值
        2.10. COMMENT
        2.11. 修改表名
        2.12. Engine 存储引擎
            显示当前数据库支持引擎
            切换引擎
            FEDERATED
            BLACKHOLE
            ARCHIVE
            CSV
    3. Partitioning
        3.1. RANGE

- 9.2. drop trigger
- 9.3. show triggers
  - SHOW CREATE TRIGGER
- 9.4. EXAMPLE
  - BEFORE/AFTER
  - UUID
  - CALL PROCEDURE

10. 事件调度器(EVENT)
    - 10.1. 启用 EVENT
    - 10.2. 创建 EVENT
    - 10.3. 禁用/启用
    - 10.4. 查看 events
    - 10.5. 删除 EVENT
    - 10.6. EVENT 应用案例
      - 实例·每月创建一个表
      - 案例·定时删除数据
      - 指定日期执行

5. DML (Data Manipulation Language)
    1. INSERT
       - 1.1. INSERT INTO ... SELECT
       - 1.2. INSERT IGNORE
       - 1.3. INSERT...ON DUPLICATE KEY UPDATE
    2. REPLACE
    3. DELETE
       - 3.1. 删除重复数据
6. SQL Statement Syntax
    1. DISTINCT
    2. group by
    3. HAVING
    4. REGEXP
    5. IN / NOT IN
    6. ALL / Any
    7. exists, not exists
    8. UNION
       - 8.1. UNION ALL
       - 8.2. 两张表字段不对等解决方法

8. DCL (Data Control Language)
    1. 锁
        1.1. 共享锁
        1.2. 排他锁
        1.3. 锁
            表的加锁与解锁
            禁止查询
        1.4. 锁等待与超时
            超时设置
            select for update nowait
    2. 事务处理和锁定语句
        2.1. 事务隔离级别
        2.2. 事务所用到的表
        2.3. 解决更新冲突
        2.4. 共享锁
        2.5. SAVEPOINT
9. Optimization
    1. Limit 状态
    2. 使用 Btrfs 文件系统存储mysql数据
    3. 打开表的数量
    4. Buffering and Caching
        4.1. Query Cache SELECT Options
    5. where 优化
    6. SHOW PROFILE Syntax SQL性能分析器
    7. PROCEDURE ANALYSE()
10. MySQL Connectors
    1. JDBC
    2. ODBC
    3. MySQL native driver for PHP - mysqlnd
    4. python-mysqldb
11. MySQL GUI/Web Manager
    1. HeidiSQL
    2. Toad for MySQL Freeware
    3. phpMyAdmin - MySQL web administration tool
    4. Maatkit Essential command-line utilities for MySQL
12. Miscellaneous

    1. Multi-Master Replication Manager for MySQL
    2. MHA
    3. HandlerSocket
    4. Maatkit
    5. Mroonga
    6. Amoeba
13. FAQ
    1. Reset root password 重置MySQL root密码
        1.1. MySQL 5.7.x
        1.2. MySQL 8.0
    2. 查看错误代码
        2.1. ERROR 1153 (08S01) at line 3168: Got a packet bigger than 'max_allowed_packet' bytes
        2.2. ERROR 1129 (00000): Host 'XXXXXX' is blocked because of many connection errors; unblock with 'mysqladmin flush-hosts'
    3. 临时表是否需要建索引
    4. [Warning] Changed limits: max_open_files: 5000 (requested 20480)
    5. Table 'performance_schema.session_variables' doesn't exist
    6. SQL Error (1038): Out of sort memory, consider increasing server sort buffer size
    7. this is incompatible with sql_mode=only_full_group_by
    8. ERROR 1071 (42000) at line 25: Specified key was too long; max key length is 767 bytes
    9. ERROR 1086 (HY000): File '/var/lib/mysql-files/order.txt' already exists
    10. ERROR 1114 (HY000): The table 'your_table' is full
    11. Error Code: 1146. Table 'test.CACHE_UPDATE' doesn't exist
    12. Error Code: 1175. You are using safe update mode and you tried to update a table without a WHERE that uses a KEY column.
    13. ERROR 1273 (HY000) at line 3364: Unknown collation: 'utf8mb4_0900_ai_ci'

14. ERROR 1290 (HY000): The MySQL server is running with the --secure-file-priv option so it cannot execute this statement
15. ERROR 1364: 1364: Field 'id' doesn't have a default value
16. Error Code: 1292. Incorrect datetime value: '0000-00-00 00:00:00' for column 'create_time' at row 95692
17. ERROR 1415: Not allowed to return a result set from a trigger
18. ERROR 1503 (HY000): A PRIMARY KEY must include all columns in the table's partitioning function
19. ERROR 1819 (HY000): Your password does not satisfy the current policy requirements
20. ERROR 1820 (HY000): You must reset your password using ALTER USER statement before executing this statement.
21. ERROR 1840 (HY000) at line 24: @@GLOBAL.GTID_PURGED can only be set when @@GLOBAL.GTID_EXECUTED is empty.
22. ERROR 2002 (HY000): Can't connect to local MySQL server through socket '/var/lib/mysql/mysql.sock'
23. ERROR 2013 (HY000): Lost connection to MySQL server during query
24. ERROR 2059 (HY000): Authentication plugin 'caching_sha2_password' cannot be loaded: /usr/lib64/mysql/plugin/caching_sha2_password.so: cannot open shared object file: No such file or directory
25. ERROR 3024 (HY000): Query execution was interrupted, maximum statement execution time exceeded
26. Authentication plugin 'caching_sha2_password' cannot be loaded: /usr/lib64/mysql/plugin/caching_sha2_password.so: cannot open shared object file: No such fileor directory
27. com.mysql.jdbc.exceptions.jdbc4.MySQLNonTransientConnectionException: Public Key Retrieval is not allowed

表格清单



范例清单

# Netkiller MySQL 手札

## MySQL, MariaDB, Percona ...

ISBN#

**Mr. Neo Chan, 陈景峯(BG7NYT)**

中国广东省深圳市望海路半岛城邦三期
518067
+86 13113668890

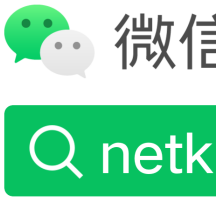
<netkiller@msn.com>

文档始创于 2010-11-18

电子书最近一次更新于 2023-03-08 22:23:34




http://www.netkiller.cn
http://netkiller.github.io
http://netkiller.sourceforge.net
微信公众号: netkiller
微信：13113668890 请注明“读者”
QQ：13721218 请注明“读者”
QQ群：128659835 请注明“读者”
知乎专栏 | 多维度架构




$Date: 2013-04-10 15:03:49 +0800 (Wed, 10 Apr 2013) $

Netkiller 数据库系列手札

数据库系列手札

| Netkiller Database 手札 | Netkiller PostgreSQL 手札 | Netkiller MySQL 手札 | Netkiller NoSQL 手札 | Netkiller LDAP 手札 |
| --- | --- | --- | --- | --- |

# 致读者

Netkiller 系列手札 已经被 Github 收录，并备份保存在北极地下250米深的代码库中，备份会保留1000年。

Preserving open source software for future generations

The world is powered by open source software. It is a hidden cornerstone of modern civilization, and the shared heritage of all humanity.

The GitHub Arctic Code Vault is a data repository preserved in the Arctic World Archive (AWA), a very-long-term archival facility 250 meters deep in the permafrost of an Arctic mountain.

We are collaborating with the Bodleian Library in Oxford, the Bibliotheca Alexandrina in Egypt, and Stanford Libraries in California to store copies of 17,000 of GitHub's most popular and most-depended-upon projects—open source's "greatest hits"—in their archives, in museum-quality cases, to preserve them for future generations.

https://archiveprogram.github.com/arctic-vault/

# 自述

Netkiller 手札系列电子书　http://www.netkiller.cn

## Netkiller MySQL 手札

陈景峰 著

《Netkiller 系列 手札》是一套免费系列电子书，netkiller 是 nickname 从1999 开使用至今，“手札” 是札记，手册的含义。

2003年之前我还是以文章形式在BBS上发表各类技术文章，后来发现文章不够系统，便尝试写长篇技术文章加上章节目录等等。随着内容增加，不断修订，开始发布第一版，第二版.....

IT知识变化非常快，而且具有时效性，这样发布非常混乱，经常有读者发现第一版例子已经过时，但他不知道我已经发布第二版。

我便有一种想法，始终维护一个文档，不断更新，使他保持较新的版本不过时。

第一部电子书是《PostgreSQL 实用实例参考》开始我使用 Microsoft Office Word 慢慢随着文档尺寸增加 Word 开始表现出力不从心。

我看到PostgreSQL 中文手册使用SGML编写文档，便开始学习 Docbook SGML。使用Docbook写的第一部电子书是《Netkiller Postfix Integrated Solution》这是Netkiller 系列手札的原型。

至于“手札”一词的来历，是因为我爱好摄影，经常去一个台湾摄影网站，名字就叫“摄影家手札”。

由于硬盘损坏数据丢失 《Netkiller Postfix Integrated Solution》 的 SGML文件已经不存在; Docbook SGML存在很多缺陷 UTF-8支持不好，转而使用Docbook XML.

目前技术书籍的价格一路飙升，动则￥80，￥100，少则￥50，￥60. 技术书籍有时效性，随着技术的革新或淘汰，大批书记成为废纸垃圾。并且这些书技术内容雷同，相互抄袭，质量越来越差，甚至里面给出的例子错误百出，只能购买影印版，或者翻译的版本。

在这种背景下我便萌生了自己写书的想法，资料主要来源是我的笔记与例子。我并不想出版，只为分享，所有我制作了基于CC License 发行的系列电子书。

本书注重例子，少理论（捞干货），只要你对着例子一步一步操作，就会成功，会让你有成就感并能坚持学下去，因为很多人遇到障碍就会放弃，其实我就是这种人，只要让他看到希望，就能坚持下去。

# 1. 写给读者

*为什么写这篇文章*

有很多想法,工作中也用不到所以未能实现，所以想写出来,和大家分享.有一点写一点,写得也不好,只要能看懂就行,就当学习笔记了.

开始零零碎碎写过一些文档，也向维基百科供过稿，但维基经常被ZF封锁，后来发现sf.net可以提供主机存放文档，便做了迁移。并开始了我的写作生涯。

这篇文档是作者20年来对工作的总结,是作者一点一滴的积累起来的，有些笔记已经丢失，所以并不完整。

因为工作太忙整理比较缓慢。目前的工作涉及面比较窄所以新文档比较少。

我现在花在技术上的时间越来越少，兴趣转向摄影，无线电。也想写写摄影方面的心得体会。

*写作动力:*

曾经在网上看到外国开源界对中国的评价，中国人对开源索取无度，但贡献却微乎其微.这句话一直记在我心中，发誓要为中国开源事业做我仅有的一点微薄贡献

另外写文档也是知识积累，还可以增加在圈内的影响力.

人跟动物的不同,就是人类可以把自己学习的经验教给下一代人.下一代在上一代的基础上再创新,不断积累才有今天.

所以我把自己的经验写出来,可以让经验传承

*没有内容的章节:*

目前我自己一人维护所有文档，写作时间有限，当我发现一个好主题就会加入到文档中，待我有时间再完善章节，所以你会发现很多章节是空无内容的.

文档目前几乎是流水帐试的写作，维护量很大，先将就着看吧.

我想到哪写到哪,你会发现文章没一个中心,今天这里写点,明天跳过本

章写其它的.
文中例子绝对多,对喜欢复制然后粘贴朋友很有用,不用动手写,也省时间.
理论的东西,网上大把,我这里就不写了,需要可以去网上查.
我爱写错别字,还有一些是打错的,如果发现请指正.
文中大部分试验是在Debian/Ubuntu/Redhat AS上完成.

写给读者

至读者:

我不知道什么时候，我不再更新文档或者退出IT行业去从事其他工作，我必须给这些文档找一个归宿，让他能持续更新下去。

我想捐赠给某些基金会继续运转，或者建立一个团队维护它。

我用了20年时间坚持不停地写作，持续更新，才有今天你看到的《Netkiller 手扎》系列文档，在中国能坚持20年，同时没有任何收益的技术类文档，是非常不容易的。

有很多时候想放弃，看到外国读者的支持与国内社区的影响，我坚持了下来。

中国开源事业需要各位参与，不要成为局外人，不要让外国人说:中国对开源索取无度，贡献却微乎其微。

我们参与内核的开发还比较遥远，但是进个人能力，写一些文档还是可能的。

系列文档

下面是我多年积累下来的经验总结，整理成文档供大家参考:

Netkiller Architect 手札
Netkiller Developer 手札
Netkiller PHP 手札
Netkiller Python 手札
Netkiller Testing 手札
Netkiller Cryptography 手札

Netkiller Linux 手札
Netkiller FreeBSD 手札
Netkiller Shell 手札
Netkiller Security 手札
Netkiller Web 手札
Netkiller Monitoring 手札
Netkiller Storage 手札
Netkiller Mail 手札
Netkiller Docbook 手札
Netkiller Version 手札
Netkiller Database 手札
Netkiller PostgreSQL 手札
Netkiller MySQL 手札
Netkiller NoSQL 手札
Netkiller LDAP 手札
Netkiller Network 手札
Netkiller Cisco IOS 手札
Netkiller H3C 手札
Netkiller Multimedia 手札
Netkiller Management 手札
Netkiller Spring 手札
Netkiller Perl 手札
Netkiller Amateur Radio 手札

# 2. 作者简介

陈景峯 (ㄔㄣˊ ㄐㄧㄥˇ ㄈㄥ)

Nickname： netkiller | English name: Neo chen | Nippon name: ちん けいほう (音訳) | Korean name: 천징봉 | Thailand name: ภูมิภาพภูเขา | Vietnam: Trần Cảnh Phong

Callsign: BG7NYT | QTH: ZONE CQ24 ITU44 ShenZhen, China

程序猿，攻城狮，挨踢民工, Full Stack Developer, UNIX like Evangelist, 业余无线电爱好者（呼号：BG7NYT）,户外运动，山地骑行以及摄影爱好者。

《Netkiller 系列 手札》的作者

**成长阶段**

1981年1月19日(庚申年腊月十四)出生于黑龙江省青冈县建设乡双富大队第一小队

1989年9岁随父母迁居至黑龙江省伊春市，悲剧的天朝教育，不知道那门子归定，转学必须降一级，我本应该上一年级，但体制让我上学前班，那年多都10岁了

1995年小学毕业，体制规定借读要交3000两银子(我曾想过不升初中)，亲戚单位分楼告别平房，楼里没有地方放东西，把2麻袋书送给我，无意中发现一本电脑书BASIC语言，我竟然看懂了，对于电脑知识追求一发而不可收，后面顶零花钱，压岁钱主要用来买电脑书《MSDOS 6.22》《新编Unix实用大全》《跟我学Foxbase》。。。。。。

1996年第一次接触UNIX操作系统，BSD UNIX, Microsoft Xinux(盖茨亲自写的微软Unix，知道的人不多)

1997年自学Turbo C语言，苦于没有电脑，后来学校建了微机室才第一次使用QBASIC(DOS 6.22 自带命令)，那个年代只能通过软盘拷贝转播，Trubo C编译器始终没有搞到，

1997年第一次上Internet网速只有9600Bps, 当时全国兴起各种信息港域名格式是www.xxxx.info.net, 访问的第一个网站是NASA下载了很多火星探路者拍回的照片，还有“淞沪”sohu的前身

1998~2000年在哈尔滨学习计算机，充足的上机时间，但老师让我们练打字（明伦五笔/WT）打字不超过80个/每分钟还要强化训练，不过这个给我的键盘功夫打了好底。

1999年学校的电脑终于安装了光驱，在一张工具盘上终于找到了Turbo C, Borland C++与Quick Basic编译器，当时对VGA图形编程非常感兴趣，通过INT33中断控制鼠标，使用绘图函数模仿windows界面。还有操作 UCDOS 中文字库，绘制矢量与点阵字体。

2000年沉迷于Windows NT与Back Office各种技术，神马主域控制器，DHCP，WINS，IIS，域名服务器，Exchange邮件服务器，MS Proxy, NetMeeting...以及ASP+MS SQL开发；用56K猫下载了一张LINUX。ISO镜像，安装后我兴奋的24小时没有睡觉。

## 职业生涯

2001 年来深圳进城打工,成为一名外来务工者. 在一个4人公司做PHP开发，当时PHP的版本是2.0, 开始使用Linux Redhat 6.2.当时很多门户网站都是用FreeBSD,但很难搞到安装盘，在网易社区认识了一个网友,从广州给我寄了一张光盘，FreeBSD 3.2

2002 年我发现不能埋头苦干,还要学会"做人".后辗转广州工作了半年，考了一个Cisco CCNA认证。回到深圳重新开始，在车公庙找到一家工作做Java开发

2003 年这年最惨,公司拖欠工资16000元,打过两次官司2005才付清.

2004 年开始加入分布式计算团队,目前成绩，工作仍然是Java开发并且开始使用PostgreSQL数据库。

2004-10月开始玩户外和摄影

2005-6月成为中国无线电运动协会会员,呼号BG7NYT,进了一部Yaesu FT-60R手台。公司的需要转回PHP与MySQL，相隔几年发现PHP进步很大。在前台展现方面无人能敌，于是便前台使用PHP，后台采用Java开发。

2006 年单身生活了这么多年,终于找到归宿. 工作更多是研究PHP各种框架原理

2007 物价上涨,金融危机，休息了4个月（其实是找不到工作），关外很难上439.460中继，搞了一台Yaesu FT-7800.

2008 终于找到英文学习方法， 《Netkiller Developer 手札》，《Netkiller Document 手札》

2008-8-8 08:08:08 结婚,后全家迁居湖南省常德市

2009《Netkiller Database 手札》,2009-6-13学车，年底拿到C1驾照

2010 对电子打击乐产生兴趣，计划学习爵士鼓。由于我对Linux热爱，我轻松的接管了公司的运维部，然后开发运维两把抓。我印象最深刻的是公司一次上架10个机柜，我们用买服务器纸箱的钱改善伙食。我将40多台服务器安装BOINC做压力测试，获得了中国第二的名次。

2011 平凡的一年，户外运动停止，电台很少开，中继很少上，摄影主要是拍女儿与家人，年末买了一辆山地车

2012 对油笔画产生了兴趣，活动基本是骑行银湖山绿道，

2013 开始学习民谣吉他，同时对电吉他也极有兴趣；最终都放弃了。这一年深圳开始推数字中继2013-7-6日入手Motorola

MOTOTRBO XIR P8668，Netkiller 系列手札从Sourceforge向Github迁移；年底对MYSQL UDF，Engine与PHP扩展开发产生很浓的兴趣，拾起遗忘10+年的C，写了几个mysql扩展（图片处理，fifo管道与ZeroMQ），10月份入Toyota Rezi 2.5V并写了一篇《攻城狮的苦逼选车经历》

2014-9-8 在淘宝上买了一架电钢琴 Casio Privia PX-5S pro 开始陪女儿学习钢琴，由于这家钢琴是合成器电钢，里面有打击乐，我有对键盘鼓产生了兴趣。

2014-10-2号罗浮山两日游，对中国道教文化与音乐产生了兴趣，10月5号用了半天时间学会了简谱。10月8号入Canon 5D Mark III + Canon Speedlite 600EX-RT香港过关被查。

2014-12-20号对乐谱制作产生兴趣（https://github.com/SheetMusic/Piano），给女儿做了几首钢琴伴奏曲，MuseScore制谱然后生成MIDI与WAV文件。

2015-09-01 晚饭后拿起爵士鼓基础教程尝试在Casio Privia PX-5S pro演练，经过反复琢磨加上之前学钢琴的乐理知识，终于在02号晚上，打出了简单的基本节奏，迈出了第一步。

2016 对弓箭（复合弓）产生兴趣，无奈天朝法律法规不让玩。每周游泳轻松1500米无压力，年底入 xbox one s 和 Yaesu FT-2DR, 同时开始关注功放音响这块

2017 7月9号入 Yamaha RX-V581 功放一台，连接Xbox打游戏爽翻了，入Kindle电子书，计划学习蝶泳，果断放弃运维和开发知识体系转攻区块链。

2018 从溪山美地搬到半岛城邦，丢弃了多年攒下的家底。11 月开始玩 MMDVM，使用 Yaesu FT-7800 发射，连接MMDVM中继板，树莓派，覆盖深圳湾，散步骑车通联两不误。

2019 卖了常德的房子，住了5次院，哮喘反复发作，决定停止电子书更新，兴趣转到知乎，B站

2020 准备找工作

职业生涯路上继续打怪升级

# 3. 如何获得文档

下载 **Netkiller** 手札 **(epub,kindle,chm,pdf)**
EPUB https://github.com/netkiller/netkiller.github.io/tree/master/download/epub
MOBI https://github.com/netkiller/netkiller.github.io/tree/master/download/mobi
PDF https://github.com/netkiller/netkiller.github.io/tree/master/download/pdf
CHM https://github.com/netkiller/netkiller.github.io/tree/master/download/chm

通过 **GIT** 镜像整个网站

https://github.com/netkiller/netkiller.github.com.git

$ git clone https://github.com/netkiller/netkiller.github.com.git

镜像下载

整站下载

```
wget -m http://www.netkiller.cn/index.html
```

指定下载

```
wget -m wget -m http://www.netkiller.cn/linux/index.html
```

**Yum** 下载文档

获得光盘介质，RPM包，DEB包，如有特别需要，请联系我

YUM 在线安装电子书

http://netkiller.sourceforge.net/pub/repo/

```
# cat >> /etc/yum.repos.d/netkiller.repo <<EOF
[netkiller]
```

```
name=Netkiller Free Books
baseurl=http://netkiller.sourceforge.net/pub/repo/
enabled=1
gpgcheck=0
gpgkey=
EOF
```

查找包

```
# yum search netkiller

netkiller-centos.x86_64 : Netkiller centos Cookbook
netkiller-cryptography.x86_64 : Netkiller cryptography Cookbook
netkiller-docbook.x86_64 : Netkiller docbook Cookbook
netkiller-linux.x86_64 : Netkiller linux Cookbook
netkiller-mysql.x86_64 : Netkiller mysql Cookbook
netkiller-php.x86_64 : Netkiller php Cookbook
netkiller-postgresql.x86_64 : Netkiller postgresql Cookbook
netkiller-python.x86_64 : Netkiller python Cookbook
netkiller-version.x86_64 : Netkiller version Cookbook
```

安装包

```
yum install netkiller-docbook
```

# 4. 打赏（Donations）

If you like this documents, please make a donation to support the authors' efforts. Thank you!

您可以通过微信，支付宝，贝宝给作者打赏。

**银行(Bank)**

招商银行(China Merchants Bank)

开户名：陈景峰

账号：9555500000007459

**微信（Wechat）**

**支付宝（Alipay）**

**PayPal Donations**

https://www.paypal.me/netkiller

# 5. 联系方式

主站 http://www.netkiller.cn/

备用 http://netkiller.github.io/

繁体网站 http://netkiller.sourceforge.net/


联系作者

Mobile: +86 13113668890

Email: netkiller@msn.com

QQ群: 128659835 请注明“读者”

QQ: 13721218

ICQ: 101888222


注：请不要问我安装问题！


博客 **Blogger**

知乎专栏 https://zhuanlan.zhihu.com/netkiller

LinkedIn: http://cn.linkedin.com/in/netkiller

OSChina: http://my.oschina.net/neochen/

Facebook: https://www.facebook.com/bg7nyt

Flickr: http://www.flickr.com/photos/bg7nyt/

Disqus: http://disqus.com/netkiller/

solidot: http://solidot.org/~netkiller/

SegmentFault: https://segmentfault.com/u/netkiller

Reddit: https://www.reddit.com/user/netkiller/

Digg: http://www.digg.com/netkiller

Twitter: http://twitter.com/bg7nyt

weibo: http://weibo.com/bg7nyt

**Xbox club**

我的 xbox 上的ID是 netkiller xbox，我创建了一个俱乐部 netkiller 欢迎加入。

**Radio**

CQ CQ CQ DE BG7NYT:

如果这篇文章对你有所帮助,请寄给我一张QSL卡片, qrz.cn or qrz.com or hamcall.net

Personal Amateur Radiostations of P.R.China

ZONE CQ24 ITU44 ShenZhen, China

Best Regards, VY 73! OP. BG7NYT

守听频率 DMR 438.460 -8 Color 12 Slot 2 Group 46001

守听频率 C4FM 439.360 -5 DN/VW

**MMDVM Hotspot:**

Callsign: BG7NYT QTH: Shenzhen, China

YSF: YSF80337 - CN China 1 - W24166/TG46001

DMR: BM_China_46001 - DMR Radio ID: 4600441

# 第 1 章 MySQL Server

## 1. MySQL 安装

### MySQL Installation

http://downloads.mysql.com/archives.php

### 1.1. Rocky Linux 9.0

```
[root@netkiller ~]# dnf install -y mysql-server

[root@netkiller ~]# systemctl enable mysqld
Created symlink /etc/systemd/system/multi-
user.target.wants/mysqld.service →
/usr/lib/systemd/system/mysqld.service.
```

备份配置文件

```
cp /etc/my.cnf{,.original}
cp /etc/my.cnf.d/mysql-server.cnf{,.original}
cp /etc/my.cnf.d/client.cnf{,.original}

cat >> /etc/my.cnf.d/mysql-server.cnf <<EOF

# Add by Neo
character-set-server=utf8mb4
collation-server=utf8mb4_general_ci
explicit_defaults_for_timestamp=true
lower_case_table_names=1
EOF
```

设置文件打开数据

```
cat >> /etc/security/limits.d/20-nofile.conf <<EOF
mysql soft nofile 65535
mysql hard nofile 65535
EOF

mkdir /etc/systemd/system/mysqld.service.d/

cat >> /etc/systemd/system/mysqld.service.d/override.conf  <<EOF
[Service]
LimitNOFILE=65000
EOF
```

启动数据库

```
[root@netkiller ~]# systemctl start mysqld
```

创建用户数据库 root 用户

CREATE USER 'root'@'%' IDENTIFIED BY 'test';

GRANT ALL ON *.* TO 'root'@'%' WITH GRANT OPTION;

```
[root@netkiller ~]# mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 10
Server version: 8.0.28 Source distribution

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or
its
affiliates. Other names may be trademarks of their respective
```

```
owners.

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

mysql> CREATE USER 'root'@'%' IDENTIFIED BY 'chen';
Query OK, 0 rows affected (0.03 sec)

mysql> GRANT ALL ON *.* TO 'root'@'%' WITH GRANT OPTION;
Query OK, 0 rows affected (0.02 sec)
```

开启防火墙

```
[root@netkiller ~]# firewall-cmd --permanent --zone=public --
add-port=3306/tcp
success
```

## 1.2. CentOS

### CentOS 8 Stream + dnf 安装 Mysql

```
[root@localhost ~]# dnf install -y mysql-server mysql mysql-test
[root@localhost ~]# systemctl enable mysqld
Created symlink /etc/systemd/system/multi-
user.target.wants/mysqld.service →
/usr/lib/systemd/system/mysqld.service.
```

备份配置文件

```
cp /etc/my.cnf{,.original}
cp /etc/my.cnf.d/mysql-server.cnf /etc/mysql-server.cnf.original
```

```
cat >> /etc/my.cnf.d/mysql-server.cnf <<EOF

# Add by Neo
character-set-server=utf8mb4
collation-server=utf8mb4_general_ci
explicit_defaults_for_timestamp=true
lower_case_table_names=1
EOF
```

设置文件打开数据

```
cat >> /etc/security/limits.d/20-nofile.conf <<EOF
mysql soft nofile 65535
mysql hard nofile 65535
EOF

mkdir /etc/systemd/system/mysqld.service.d/

cat >> /etc/systemd/system/mysqld.service.d/override.conf  <<EOF
[Service]
LimitNOFILE=65000
EOF
```

启动数据库

```
[root@localhost ~]# systemctl start mysqld
```

创建用户

```
mysql> CREATE USER 'root'@'%' IDENTIFIED BY 'test';
Query OK, 0 rows affected (0.05 sec)
```

```
mysql> GRANT ALL ON *.* TO 'root'@'%' WITH GRANT OPTION;
Query OK, 0 rows affected (0.03 sec)
```

例 1.1. MySQL 8 创建root账号

```
[root@localhost ~]# mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.21 Source distribution

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All
rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or
its
affiliates. Other names may be trademarks of their respective
owners.

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

mysql> CREATE USER 'root'@'%' IDENTIFIED BY 'test';
Query OK, 0 rows affected (0.00 sec)

mysql> GRANT ALL ON *.* TO 'root'@'%' WITH GRANT OPTION;
Query OK, 0 rows affected (0.01 sec)

mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.00 sec)

mysql>
```

### CentOS 6.2 + MySQL 5.5.25 (RPM)

准备下面的软件包

```
```

```
# ls -1
MySQL-client-5.5.25-1.el6.x86_64.rpm
MySQL-devel-5.5.25-1.el6.x86_64.rpm
MySQL-server-5.5.25-1.el6.x86_64.rpm
MySQL-shared-5.5.25-1.el6.x86_64.rpm
MySQL-shared-compat-5.5.25-1.el6.x86_64.rpm
```

使用 yum 本地安装 rpm, yum 可以帮你解决依赖于冲突

```
# yum localinstall MySQL-*
```

```
# /etc/init.d/mysql start
Starting MySQL... SUCCESS!

# /usr/bin/mysqladmin -u root password 'tUG26WSslP30bkbwtMhn'
```

## MySQL 8.0

安装

```
curl -s
https://raw.githubusercontent.com/oscm/shell/master/database/mys
ql/8.0/server.sh | bash
```

启动

```
systemctl enable mysql
systemctl start mysql
```

必须修改密码后才能使用

```
[root@netkiller ~]# grep "A temporary password"
/var/log/mysqld.log
2018-04-03T02:24:16.935070Z 1 [Note] A temporary password is
generated for root@localhost: kMA*d<e#Q3EC
2018-04-20T03:36:31.935143Z 5 [Note] [MY-010454] [Server] A
temporary password is generated for root@localhost: MqatK=hae5F#

[root@netkiller ~]# mysqladmin -u root -p'MqatK=hae5F#' password
mysqladmin: [Warning] Using a password on the command line
interface can be insecure.
New password:
Confirm new password:
Warning: Since password will be sent to server in plain text,
use ssl connection to ensure password safety.

[root@netkiller ~]# mysql -uroot -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 10
Server version: 8.0.11 MySQL Community Server - GPL

Copyright (c) 2000, 2018, Oracle and/or its affiliates. All
rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or
its
affiliates. Other names may be trademarks of their respective
owners.

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

mysql>
```

创建用户

```
```

```
mysql> CREATE USER 'root'@'%' IDENTIFIED BY
'MQiEge1ikst7S_6tlXzBOmt_4b';
Query OK, 0 rows affected (0.05 sec)

mysql> GRANT ALL ON *.* TO 'root'@'%';
Query OK, 0 rows affected (0.03 sec)
```

## 1.3. Docker

```
docker run --name mysql -d \
    --restart always \
    -e MYSQL_ROOT_PASSWORD=123456 \
    -e MYSQL_DATABASE=test \
    -e MYSQL_USER=test \
    -e MYSQL_PASSWORD=test \
    -p 127.0.0.1:3306:3306 \
    mysql:latest
```

## 1.4. Ubuntu/Debian

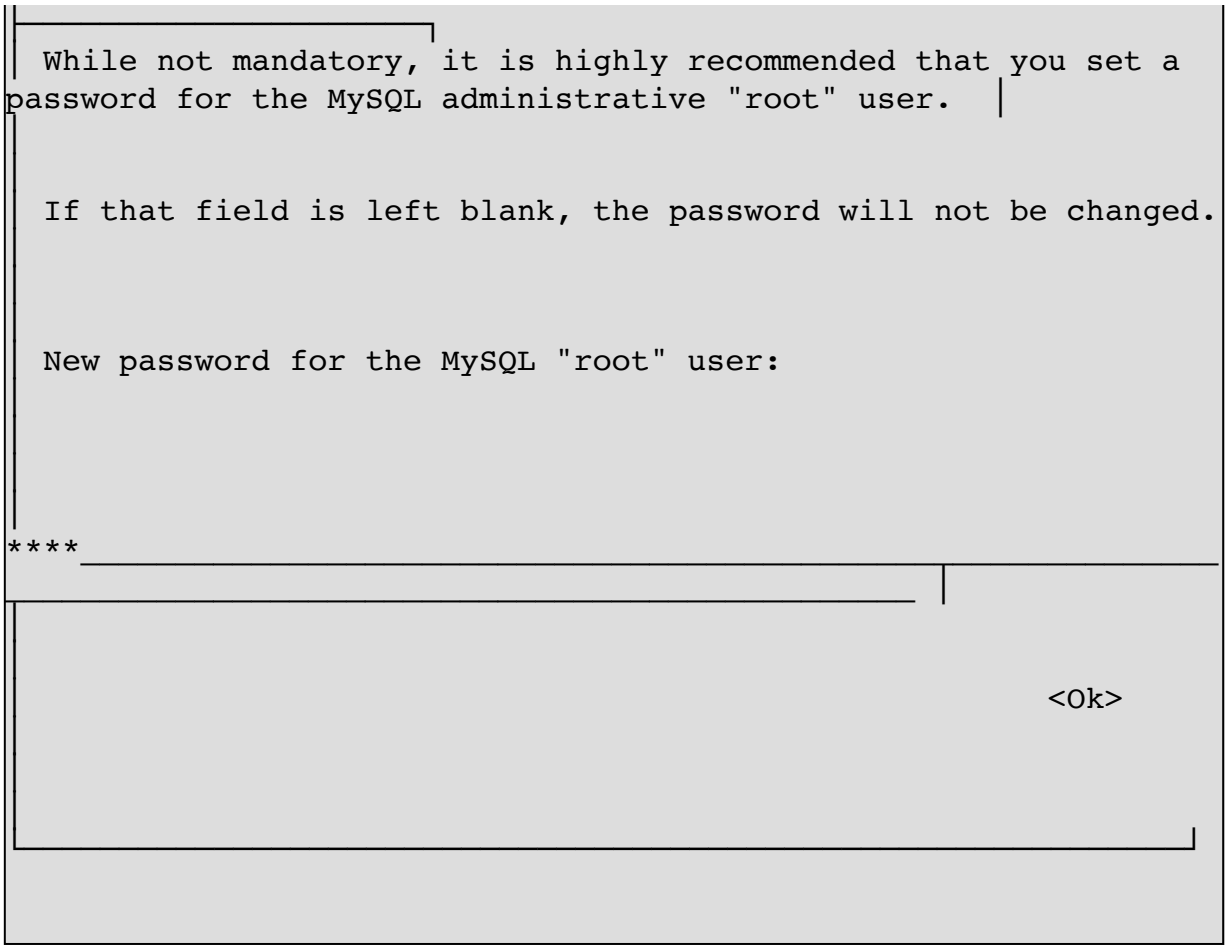
### Installation by apt under debian/ubuntu

安装环境 ubuntu 17.10

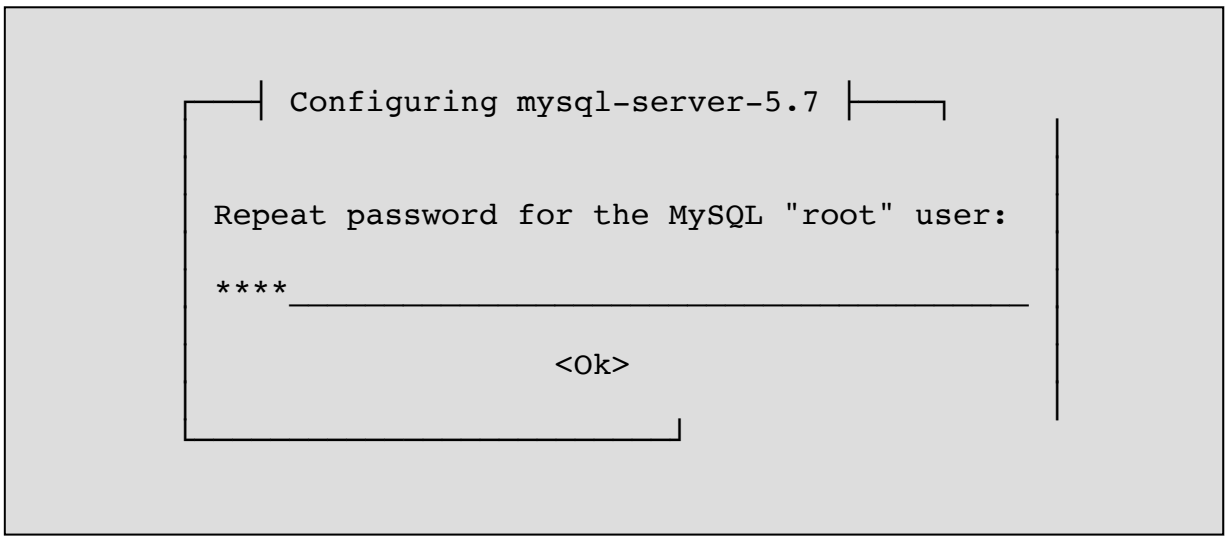
```
                        sudo apt install mysql-server mysql-
client
```

New password for the MySQL "root" user

```
┌──────────────────────┤ Configuring mysql-server-5.7
```




Repeat password for the MySQL "root" user




尝试登录，验证是否安装成功

```
# mysql -udbuser -p
Enter password:

mysql> SHOW GRANTS;
+-----------------------------------------------------------------
-----------------------------------------------------------------
---------+
| Grants for root@localhost
|
+-----------------------------------------------------------------
-----------------------------------------------------------------
---------+
| GRANT ALL PRIVILEGES ON *.* TO 'root'@'localhost' IDENTIFIED
BY PASSWORD '*C6325DAF39AE6CC34E960D3C65F1398FE467E1D0' WITH
GRANT OPTION |
+-----------------------------------------------------------------
-----------------------------------------------------------------
---------+
1 row in set (0.00 sec)

GRANT ALL PRIVILEGES ON example.* TO 'dbuser'@'localhost'
IDENTIFIED BY '******' WITH GRANT OPTION;
FLUSH PRIVILEGES;
```

配置监听IP地址，默认数据库只能从 127.0.0.1访问

```
                          neo@netkiller /etc/mysql/mysql.conf.d %
vim
                          /etc/mysql/mysql.conf.d/mysqld.cnf

                          bind-address = 0.0.0.0
```

**mysql-5.5.21-debian6.0-i686.deb**

```
sudo apt-get install libaio1
```

```
sudo groupadd mysql
sudo useradd -r -g mysql mysql

sudo dpkg -i mysql-5.5.21-debian6.0-i686.deb

cd /opt/mysql/
sudo chown -R mysql .
sudo chgrp -R mysql .

cd server-5.5/

sudo support-files/binary-configure

sudo chown -R mysql data

# Next command is optional
shell> cp support-files/my-medium.cnf /etc/my.cnf

shell> bin/mysqld_safe --user=mysql &

# Next command is optional
sudo cp support-files/mysql.server /etc/init.d/mysql
```

## 1.5. 源码安装

### Installation by source code

```
./configure \
--prefix=/usr/local/$MYSQL_DIR \
--enable-assembler \
--enable-local-infile \
--with-charset=utf8 \
--with-collation=utf8_general_ci \
--with-extra-charsets=none \
--with-openssl \
--with-pthread \
--with-unix-socket-path=/var/lib/mysql/mysql.sock \
--with-mysqld-user=mysql \
--with-mysqld-ldflags \
```

```
--with-client-ldflags \
--with-comment \
--with-big-tables \
--without-ndb-debug \
--without-docs \
--without-debug \
--without-bench

make && make install
```

/usr/local/$MYSQL_DIR/bin/mysql_install_db

other option

```
                        --without-isam
                        --without-innodb
                        --without-ndbcluster
                        --without-blackhole
                        --without-ibmdb2i
                        --without-federated
                        --without-example
                        --without-comment
                        --localstatedir=/usr/local/mysql/data
```

## 1.6. 二进制版本

### MySQL binary distribution

```
shell> groupadd mysql
shell> useradd -r -g mysql mysql
shell> cd /usr/local
shell> tar zxvf /path/to/mysql-VERSION-OS.tar.gz
shell> ln -s full-path-to-mysql-VERSION-OS mysql
shell> cd mysql
shell> chown -R mysql .
shell> chgrp -R mysql .
shell> scripts/mysql_install_db --user=mysql
```

```
shell> chown -R root .
shell> chown -R mysql data
# Next command is optional
shell> cp support-files/my-medium.cnf /etc/my.cnf
shell> bin/mysqld_safe --user=mysql &
# Next command is optional
shell> cp support-files/mysql.server /etc/init.d/mysql.server
```

install core database

```
[root@test mysql]# ./scripts/mysql_install_db
Installing MySQL system tables...
100428 23:16:20 [Warning] '--skip-locking' is deprecated and
will be removed in a future release. Please use '--skip-
external-locking' instead.
OK
Filling help tables...
100428 23:16:20 [Warning] '--skip-locking' is deprecated and
will be removed in a future release. Please use '--skip-
external-locking' instead.
OK

To start mysqld at boot time you have to copy
support-files/mysql.server to the right place for your system

PLEASE REMEMBER TO SET A PASSWORD FOR THE MySQL root USER !
To do so, start the server, then issue the following commands:

./bin/mysqladmin -u root password 'new-password'
./bin/mysqladmin -u root -h db.example.com password 'new-
password'

Alternatively you can run:
./bin/mysql_secure_installation

which will also give you the option of removing the test
databases and anonymous user created by default.  This is
strongly recommended for production servers.

See the manual for more instructions.
```

```
You can start the MySQL daemon with:
cd . ; ./bin/mysqld_safe &

You can test the MySQL daemon with mysql-test-run.pl
cd ./mysql-test ; perl mysql-test-run.pl

Please report any problems with the ./bin/mysqlbug script!
```

set root's password

```
                        [root@test mysql]# cp support-
files/mysql.server
                        /etc/init.d/mysqld
                        [root@test mysql]# /etc/init.d/mysqld
start
                        Starting MySQL. [ OK ]

                        [root@test mysql]# ./bin/mysqladmin -u
root
                        password 'chen'
                        [root@test mysql]# ./bin/mysqladmin -u
root -h
                        db.example.com password 'chen'
```

test

```
[root@test mysql]# ./bin/mysql -uroot -pchen
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3
Server version: 5.1.45 MySQL Community Server (GPL)

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

mysql>
```

## 1.7. Installing MySQL on Linux Using the MySQL Yum Repository

### MySQL 5.6

http://dev.mysql.com/doc/mysql-repo-excerpt/5.6/en/linux-installation-yum-repo.html

```
yum localinstall http://dev.mysql.com/get/mysql-community-release-el6-5.noarch.rpm
```

安装MySQL Server

```
yum install mysql-server
chkconfig mysqld on
service mysqld start
```

修改root密码

```
mysqladmin -u root password 'new-password'
```

安全设置向导

```
/usr/bin/mysql_secure_installation
```

### MySQL 5.7

```
yum localinstall -y https://dev.mysql.com/get/mysql57-community-
release-el7-11.noarch.rpm
yum install mysql-server -y
systemctl enable mysqld
systemctl start mysqld

cp /etc/my.cnf{,.original}

cat >> /etc/security/limits.d/20-nofile.conf <<EOF

mysql soft nofile 40960
mysql hard nofile 40960
EOF

cat >> /etc/my.cnf.d/default.cnf <<EOF
[mysqld]
skip-name-resolve
max_connections=8192
default-storage-engine=INNODB

#wait_timeout=30
#interactive_timeout=30

character-set-server=utf8
collation_server=utf8_general_ci
init_connect='SET NAMES utf8'

explicit_defaults_for_timestamp=true

query_cache_type=1
query_cache_size=512M

[client]
character_set_client=utf8

EOF
```

MySQL 5.7 会随机分配一个密码给用户

```
```

```
grep "A temporary password" /var/log/mysqld.log
```

登陆后修改密码

```
ALTER USER root@localhost identified by
'MQiEge1ikst7S_6tlXzBOmt_4b';
ALTER USER root@localhost PASSWORD EXPIRE NEVER;
```

## 1.8. Firewall

iptables

```
iptables -A INPUT -i eth0 -p tcp -s xxx.xxx.xxx.xxx --dport 3306
-j ACCEPT
```

## 1.9. Mac OS

```
brew install mysql
```

启动

```
brew services start mysql
```

## 1.10. MariaDB

http://mariadb.org/

## CentOS 6 YUM 安装 MariaDB

```
cat >> /etc/yum.repos.d/MariaDB.repo <<EOF
# MariaDB 5.5 CentOS repository list - created 2013-12-04 02:17
UTC
# http://mariadb.org/mariadb/repositories/
[mariadb]
name = MariaDB
baseurl = http://yum.mariadb.org/5.5/centos6-amd64
gpgkey=https://yum.mariadb.org/RPM-GPG-KEY-MariaDB
gpgcheck=1
EOF
```

```
# yum search MariaDB | grep MariaDB
============================== N/S Matched: MariaDB
==============================
MariaDB-Galera-server.x86_64 : MariaDB: a very fast and robust
SQL database
MariaDB-client.x86_64 : MariaDB: a very fast and robust SQL
database server
MariaDB-common.x86_64 : MariaDB: a very fast and robust SQL
database server
MariaDB-compat.x86_64 : MariaDB: a very fast and robust SQL
database server
MariaDB-devel.x86_64 : MariaDB: a very fast and robust SQL
database server
MariaDB-server.x86_64 : MariaDB: a very fast and robust SQL
database server
MariaDB-shared.x86_64 : MariaDB: a very fast and robust SQL
database server
MariaDB-test.x86_64 : MariaDB: a very fast and robust SQL
database server
```

安装数据库

```
# yum install -y MariaDB-server MariaDB-client
```

指定默认root密码

```
# mysqladmin -u root password 'chen'
```

数据库安全配置

```
# /usr/bin/mysql_secure_installation
/usr/bin/mysql_secure_installation: line 379: find_mysql_client:
command not found

NOTE: RUNNING ALL PARTS OF THIS SCRIPT IS RECOMMENDED FOR ALL
MariaDB
      SERVERS IN PRODUCTION USE!  PLEASE READ EACH STEP
CAREFULLY!

In order to log into MariaDB to secure it, we'll need the
current
password for the root user.  If you've just installed MariaDB,
and
you haven't set the root password yet, the password will be
blank,
so you should just press enter here.

Enter current password for root (enter for none):
OK, successfully used password, moving on...

Setting the root password ensures that nobody can log into the
MariaDB
root user without the proper authorisation.

You already have a root password set, so you can safely answer
'n'.

Change the root password? [Y/n]
New password:
Re-enter new password:
Password updated successfully!
Reloading privilege tables..
```

```
 ... Success!


By default, a MariaDB installation has an anonymous user,
allowing anyone
to log into MariaDB without having to have a user account
created for
them.  This is intended only for testing, and to make the
installation
go a bit smoother.  You should remove them before moving into a
production environment.

Remove anonymous users? [Y/n]
 ... Success!

Normally, root should only be allowed to connect from
'localhost'.  This
ensures that someone cannot guess at the root password from the
network.

Disallow root login remotely? [Y/n]
 ... Success!

By default, MariaDB comes with a database named 'test' that
anyone can
access.  This is also intended only for testing, and should be
removed
before moving into a production environment.

Remove test database and access to it? [Y/n]
 - Dropping test database...
 ... Success!
 - Removing privileges on test database...
 ... Success!

Reloading the privilege tables will ensure that all changes made
so far
will take effect immediately.

Reload privilege tables now? [Y/n]
 ... Success!

Cleaning up...

All done!  If you've completed all of the above steps, your
MariaDB
```

```
installation should now be secure.

Thanks for using MariaDB!
```

进入数据库

```
# mysql -uroot -pchen
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 12
Server version: 5.5.34-MariaDB MariaDB Server

Copyright (c) 2000, 2013, Oracle, Monty Program Ab and others.

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

MariaDB [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
+--------------------+
3 rows in set (0.00 sec)

MariaDB [(none)]> use mysql;
Reading table information for completion of table and column
names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [mysql]>
```

使用上与MySQL并无差异

**CentOS 7 安装 MariaDB**

```
yum install -y mariadb
```

## 1.11. Percona

http://www.percona.com/

### Percona yum Repository

```
# yum install http://www.percona.com/redir/downloads/percona-
release/redhat/latest/percona-release-0.1-3.noarch.rpm
```

查看所有percona软件包

```
yum search percona
```

### Percona XtraBackup

#### 安装 XtraBackup

通过YUM安装 percona-xtrabackup

```
# yum install percona-xtrabackup
```

通过RPM安装 CentOS 6

http://www.percona.com/downloads/XtraBackup/LATEST/binary/redhat/6/x86_64/

```
# yum install -y
```

```
http://www.percona.com/redir/downloads/XtraBackup/LATEST/binary/
redhat/6/x86_64/percona-xtrabackup-2.2.6-5042.el6.x86_64.rpm
```

通过RPM安装 CentOS 7

http://www.percona.com/downloads/XtraBackup/LATEST/binary/redhat/7/x86_64/

```
# yum install -y
http://www.percona.com/redir/downloads/XtraBackup/LATEST/binary/
redhat/7/x86_64/percona-xtrabackup-2.2.6-5042.el7.x86_64.rpm
```

卸载

```
# yum remove percona-xtrabackup
```

查看文件列表

```
# rpm -ql percona-xtrabackup
/usr/bin/innobackupex
/usr/bin/xbcrypt
/usr/bin/xbstream
/usr/bin/xtrabackup
/usr/share/doc/percona-xtrabackup-2.2.6
/usr/share/doc/percona-xtrabackup-2.2.6/COPYING
```

**innobackupex**

首先创建备份用户

```
mysql> CREATE USER 'backup'@'localhost' IDENTIFIED BY 's3cret';
mysql> GRANT RELOAD, LOCK TABLES, REPLICATION CLIENT ON *.* TO
'backup'@'localhost';
mysql> FLUSH PRIVILEGES;
```

```

```

备份数据库

备份所有数据库

```
# mkdir -p /backup
# innobackupex --user=backup --password=chen /backup/full
```

备份指定数据库

```
# innobackupex --user=backup --password=chen --database=test
/backup
```

--defaults-file=/etc/my.cnf 参数

```
# innobackupex --defaults-file=/etc/my.cnf --user=backup --
password=chen --database=test /backup
```

备份后打包

```
# innobackupex --user=backup --password=chen --database=test --
stream=tar /backup/  > test.tar
```

打包并压缩

```
# innobackupex --user=backup --password=chen --database=test --
stream=tar /backup/ | gzip > test.tar.gz
```

备份到远程服务器

```
# innobackupex --user=backup --password=chen --defaults-
file=/etc/my.cnf --database=test --stream=tar /backup | gzip |
ssh neo@192.168.2.1 cat ">"   /backup/backup-2014-11-12.tar.gz
```

增量备份

```
# innobackupex --user=backup --password=chen --database=test
/backup/incremental
# ls /backup/incremental
2014-11-12_13-45-26
# innobackupex --user=backup --password=chen --database=test --
incremental --incremental-basedir=/backup/incremental/2014-11-
12_13-45-26/ /backup/incremental
```

恢复数据库

恢复数据首先停止MySQL服务

```
# service mysql stop
```

恢复文件

```
# innobackupex --copy-back /path/to/BACKUP-DIR
# innobackupex --user=backup --password=chen --apply-log
/backup/full/2014-11-12_13-45-26/
```

数据恢复完成后修改权限

```
$ chown -R mysql:mysql /var/lib/mysql
```

增量备份恢复方法

```
innobackupex --apply-log --redo-only BASE-DIR
innobackupex --apply-log --redo-only BASE-DIR --incremental-
dir=INCREMENTAL-DIR-1
innobackupex --apply-log BASE-DIR --incremental-dir=INCREMENTAL-
DIR-2
innobackupex --apply-log BASE-DIR
innobackupex --copy-back BASE-DIR
```

**xbstream**

```
$ innobackupex --stream=tar /tmp
$ innobackupex --stream=xbstream /root/backup/ >
/root/backup/backup.xbstream
$ innobackupex --stream=xbstream --compress /root/backup/ >
/root/backup/backup.xbstream

$ xbstream -x <  backup.xbstream -C /root/backup/
$ innobackupex --compress --stream=xbstream /root/backup/ | ssh
user@otherhost "xbstream -x -C /root/backup/"
```

**xtrabackup**

```
# xtrabackup --user=backup --password=chen --backup --target-
dir=/backup/backup
```

## Percona Toolkit - MySQL Management Software

YUM安装

```
# yum install -y percona-toolkit
```

RPM安装

```
# yum install -y http://www.percona.com/redir/downloads/percona-
toolkit/LATEST/RPM/percona-toolkit-2.2.11-1.noarch.rpm
```

percona-toolkit 所含的文件

```
# rpm -ql percona-toolkit
/usr/bin/pt-align
/usr/bin/pt-archiver
/usr/bin/pt-config-diff
/usr/bin/pt-deadlock-logger
/usr/bin/pt-diskstats
/usr/bin/pt-duplicate-key-checker
/usr/bin/pt-fifo-split
/usr/bin/pt-find
/usr/bin/pt-fingerprint
/usr/bin/pt-fk-error-logger
/usr/bin/pt-heartbeat
/usr/bin/pt-index-usage
/usr/bin/pt-ioprofile
/usr/bin/pt-kill
/usr/bin/pt-mext
/usr/bin/pt-mysql-summary
/usr/bin/pt-online-schema-change
/usr/bin/pt-pmp
/usr/bin/pt-query-digest
/usr/bin/pt-show-grants
/usr/bin/pt-sift
/usr/bin/pt-slave-delay
/usr/bin/pt-slave-find
/usr/bin/pt-slave-restart
/usr/bin/pt-stalk
/usr/bin/pt-summary
/usr/bin/pt-table-checksum
/usr/bin/pt-table-sync
/usr/bin/pt-table-usage
/usr/bin/pt-upgrade
/usr/bin/pt-variable-advisor
/usr/bin/pt-visual-explain
/usr/share/doc/percona-toolkit-2.2.11
/usr/share/doc/percona-toolkit-2.2.11/COPYING
/usr/share/doc/percona-toolkit-2.2.11/Changelog
/usr/share/doc/percona-toolkit-2.2.11/INSTALL
```

```
/usr/share/doc/percona-toolkit-2.2.11/README
/usr/share/man/man1/percona-toolkit.1p.gz
/usr/share/man/man1/pt-align.1p.gz
/usr/share/man/man1/pt-archiver.1p.gz
/usr/share/man/man1/pt-config-diff.1p.gz
/usr/share/man/man1/pt-deadlock-logger.1p.gz
/usr/share/man/man1/pt-diskstats.1p.gz
/usr/share/man/man1/pt-duplicate-key-checker.1p.gz
/usr/share/man/man1/pt-fifo-split.1p.gz
/usr/share/man/man1/pt-find.1p.gz
/usr/share/man/man1/pt-fingerprint.1p.gz
/usr/share/man/man1/pt-fk-error-logger.1p.gz
/usr/share/man/man1/pt-heartbeat.1p.gz
/usr/share/man/man1/pt-index-usage.1p.gz
/usr/share/man/man1/pt-ioprofile.1p.gz
/usr/share/man/man1/pt-kill.1p.gz
/usr/share/man/man1/pt-mext.1p.gz
/usr/share/man/man1/pt-mysql-summary.1p.gz
/usr/share/man/man1/pt-online-schema-change.1p.gz
/usr/share/man/man1/pt-pmp.1p.gz
/usr/share/man/man1/pt-query-digest.1p.gz
/usr/share/man/man1/pt-show-grants.1p.gz
/usr/share/man/man1/pt-sift.1p.gz
/usr/share/man/man1/pt-slave-delay.1p.gz
/usr/share/man/man1/pt-slave-find.1p.gz
/usr/share/man/man1/pt-slave-restart.1p.gz
/usr/share/man/man1/pt-stalk.1p.gz
/usr/share/man/man1/pt-summary.1p.gz
/usr/share/man/man1/pt-table-checksum.1p.gz
/usr/share/man/man1/pt-table-sync.1p.gz
/usr/share/man/man1/pt-table-usage.1p.gz
/usr/share/man/man1/pt-upgrade.1p.gz
/usr/share/man/man1/pt-variable-advisor.1p.gz
/usr/share/man/man1/pt-visual-explain.1p.gz
```

# 2. MySQL Plugin

## 2.1. validate_password

插件的卸载与安装

```
uninstall plugin validate_password;
INSTALL PLUGIN validate_password SONAME 'validate_password.so';
```

查看变量设置

```
mysql> SHOW VARIABLES LIKE 'validate_password%';
+--------------------------------------+--------+
| Variable_name                        | Value  |
+--------------------------------------+--------+
| validate_password_check_user_name    | OFF    |
| validate_password_dictionary_file    |        |
| validate_password_length             | 8      |
| validate_password_mixed_case_count   | 1      |
| validate_password_number_count       | 1      |
| validate_password_policy             | MEDIUM |
| validate_password_special_char_count | 1      |
+--------------------------------------+--------+
7 rows in set (0.00 sec)

mysql>
```

修改策略与密码长度

```
mysql> set global validate_password_policy=0;
mysql> set global validate_password_length=4;
mysql> grant all privileges on test.* to 'test'@localhost
identified by 'chen';
```

## 2.2. MySQL Images manager

地址： https://github.com/netkiller/mysql-image-plugin

```
cd /usr/local/src/
git clone https://github.com/netkiller/mysql-image-plugin.git
cd mysql-image-plugin/
yum install cmake

cmake .
make && make install
                </screen>
                <screen>
Install

create function image_check returns string soname
'libimage.so';
create function image_remove returns string soname
'libimage.so';
create function image_rename returns string soname
'libimage.so';
create function image_crc32 returns string soname
'libimage.so';
create function image_move returns string soname 'libimage.so';
Uninstall

drop function image_check;
drop function image_remove;
drop function image_rename;
drop function image_crc32;
drop function image_move;
Example

select image_check('/path/filename.ext');
select image_remove('/path/filename.ext');
select image_rename('/path/oldfile.ext','/path/newfile.ext');
select image_crc32('/path/filename.ext');
select image_move('/path/filename.ext','/path/to/newfile.ext');
```

## 2.3. MySQL fifo

```
mysql-fifo-plugin

MySQL Pipes (FIFOs) Plugin

Build

cd /usr/local/src/
git clone https://github.com/netkiller/mysql-fifo-plugin.git
cd mysql-fifo-plugin/

cmake .
make
make install

or

gcc -O3  -g  -I/usr/include/mysql -I/usr/include  -fPIC -lm -lz
-shared -o libfifo.so fifo.c
sudo mv libfifo.so /usr/lib/mysql/plugin/
Plugin Install and Uninstall

Install

create function fifo_create returns string soname 'libfifo.so';
create function fifo_remove returns string soname 'libfifo.so';
create function fifo_read returns string soname 'libfifo.so';
create function fifo_write returns string soname 'libfifo.so';
Uninstall

drop function fifo_create;
drop function fifo_remove;
drop function fifo_read;
drop function fifo_write;
Testing

创建管道
```

```
mysql> create function fifo_create returns string soname
'libfifo.so';
Query OK, 0 rows affected (0.02 sec)

mysql> select fifo_create('/tmp/myfifo');
+----------------------------+
| fifo_create('/tmp/myfifo') |
+----------------------------+
| ture                       |
+----------------------------+
1 row in set (0.00 sec)
```

查看管道是否创建

```
$ ls /tmp/myfifo
/tmp/myfifo
```

覆盖测试，正确应该返回false

```
mysql> select fifo_create('/tmp/myfifo');
+----------------------------+
| fifo_create('/tmp/myfifo') |
+----------------------------+
| false                      |
+----------------------------+
1 row in set (0.00 sec)
```

删除管道

```
mysql> select fifo_remove('/tmp/myfifo');
+----------------------------+
| fifo_remove('/tmp/myfifo') |
+----------------------------+
| true                       |
+----------------------------+
1 row in set (0.00 sec)

mysql> select fifo_remove('/tmp/my');
+------------------------+
| fifo_remove('/tmp/my') |
+------------------------+
| false                  |
+------------------------+
1 row in set (0.00 sec)
```

删除不存在的管道会提示 false

读管道

在一个终端窗口中运行

```
mysql> select fifo_read('/tmp/myfifo');
+--------------------------+
| fifo_read('/tmp/myfifo') |
+--------------------------+
| Hello world              |
+--------------------------+
1 row in set (7.85 sec)
```

在另一个终端窗口中运行

```
mysql> select fifo_write('/tmp/myfifo','Hello world !!!');
+-----------------------------------------------+
| fifo_write('/tmp/myfifo','Hello world !!!') |
+-----------------------------------------------+
| true                                          |
+-----------------------------------------------+
1 row in set (0.00 sec)
```

或者

在命令行运行

```
$ echo "Hello world" > /tmp/myfifo
```

在SQL客户端中运行

```
mysql> select fifo_read('/tmp/myfifo');
+--------------------------+
| fifo_read('/tmp/myfifo') |
+--------------------------+
| Hello world
                          |
+--------------------------+
1 row in set (0.00 sec)
```

注意上面echo会自动增加换行符，-n参数可以避免

```
$ echo -n "Hello world" > /tmp/myfifo

mysql> select fifo_read('/tmp/myfifo');
+--------------------------+
| fifo_read('/tmp/myfifo') |
+--------------------------+
| Hello world              |
+--------------------------+
1 row in set (0.01 sec)
```

写管道

```
mysql> select fifo_write('/tmp/myfifo','Hello world !!!');
+-----------------------------------------------+
| fifo_write('/tmp/myfifo','Hello world !!!') |
+-----------------------------------------------+
| true                                          |
+-----------------------------------------------+
1 row in set (0.00 sec)

$ cat /tmp/myfifo
Hello world !!!

管道 /tmp/nofifo 不存在会返回false
mysql> select fifo_write('/tmp/nofifo',concat(mobile,'\r\n'))
from demo;
+-------------------------------------------------+
| fifo_write('/tmp/nofifo',concat(mobile,'\r\n')) |
+-------------------------------------------------+
| false                                           |
| false                                           |
| false                                           |
| false                                           |
| false                                           |
+-------------------------------------------------+
5 rows in set (0.01 sec)
```

## 2.4. 内容输出到文本插件

Plugin Install and Uninstall

```
# Install
create function out2file returns string soname
'liboutfile.so';

# Uninstall
drop function out2file;
```

操作演示

```
# 安装插件

mysql> create function out2file returns string soname
'liboutfile.so';
Query OK, 0 rows affected (0.00 sec)

# 调用插件

mysql> select out2file('/tmp/myoutfile',"Helloworld!!!");
+-------------------------------------------+
| out2file('/tmp/myoutfile',"Helloworld!!!") |
+-------------------------------------------+
| true                                      |
+-------------------------------------------+
1 row in set (0.00 sec)

# 查看文件

root@netkiller ~/mysql-outfile-plugin % cat /tmp/myoutfile
Helloworld!!!
```

触发器应用

```
CREATE TABLE `demo` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(45) DEFAULT NULL,
  `sex` enum('Man','Woman') DEFAULT NULL,
  `address` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8

DROP TRIGGER IF EXISTS `test`.`demo_AFTER_INSERT`;

DELIMITER $$
USE `test`$$
```

```
CREATE DEFINER=`root`@`%` TRIGGER `test`.`demo_AFTER_INSERT`
AFTER INSERT ON `demo` FOR EACH ROW
BEGIN
        set @rev = "";
        SELECT
    OUT2FILE('/tmp/demo.log',
            CONCAT_WS(',',
                    NEW.id,
                    NEW.name,
                    NEW.sex,
                    NEW.address))
        INTO @rev;
END$$
DELIMITER ;
```

# 3. Replication

## 3.1. Master Slave

### Master

过程 1.1. Master 设置步骤

1. 配置 my.cnf 文件

   确保主服务器主机上my.cnf文件的[mysqld]部分包括一个log-bin选项。该部分还应有一个server-id=Master_id选项

```
# vim /etc/mysql/my.cnf

server-id               = 1
log_bin                 = /var/log/mysql/mysql-bin.log
expire_logs_days        = 10
max_binlog_size         = 100M
binlog_do_db            = test
binlog_ignore_db        = mysql
```

   bind-address默认是127.0.0.1你必须更改它，否则Slave将无法链接到 Master

```
#bind-address           = 127.0.0.1
bind-address            = 0.0.0.0
```

   重启服务器

```
neo@netkiller:~$ sudo /etc/init.d/mysql reload
 * Reloading MySQL database server mysqld          [ OK ]
```

建议使用reload,如果不起作用再用restart

```
mysql> SHOW GLOBAL VARIABLES like 'server_id';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| server_id     | 1     |
+---------------+-------+
1 row in set (0.00 sec)
```

2. 登录slave服务器，测试主库3306工作情况，如果看到下面相关信息表示工作正常。

```
# telnet 192.168.1.246 3306
Trying 192.168.1.246...
Connected to 192.168.1.246.
Escape character is '^]'.
I
5.1.61-0ubuntu0.11.10.1-log1W<gs/*'#}p<u[J=5//:
```

3. 创建账户并授予REPLICATION SLAVE权限

```
mysql> GRANT REPLICATION SLAVE,REPLICATION CLIENT ON *.* TO
'replication'@'%.mydomain.com' IDENTIFIED BY 'slavepass';
mysql> FLUSH PRIVILEGES;
```

创建监控账号monitor（可选项），monitor 使用SHOW MASTER STATUS和SHOW SLAVE STATUS命令但没有复制权限

GRANT REPLICATION CLIENT ON *.* TO monitor@'192.168.245.131' IDENTIFIED BY 'monitorpass'

4. 锁表禁止写入新数据

```
mysql> FLUSH TABLES WITH READ LOCK;
```

5. 查看Master 工作状态

```
mysql> SHOW MASTER STATUS;
+------------------+----------+--------------+--------------
----+
| File             | Position | Binlog_Do_DB |
Binlog_Ignore_DB |
+------------------+----------+--------------+--------------
----+
| mysql-bin.000002 |      106 | test         | mysql
|
+------------------+----------+--------------+--------------
----+
1 row in set (0.00 sec)
```

如果显示下面内容表示，配置不正确

```
mysql> SHOW MASTER STATUS;
Empty set (0.02 sec)
```

取得快照并记录日志名和偏移量后，可以在主服务器上重新启用写活动

```
mysql> UNLOCK TABLES;
```

```
```

**Slave**

过程 1.2. Slave 设置步骤

1. 配置my.cnf

   从服务器的ID必须与主服务器的ID不相同,如果设置多个从服务器，每个从服务器必须有一个唯一的server-id值，必须与主服务器的以及其它从服务器的不相同。

```
# vim /etc/mysql/my.cnf

[mysqld]
server-id               = 2
```

2.

```
# service mysql restart
mysql start/running, process 22893
```

3. 指定 master 相关参数

   在从服务器上执行下面的语句，用你的系统的实际值替换选项值

```
mysql> CHANGE MASTER TO
        ->     MASTER_HOST='master_host_name',
        ->     MASTER_USER='replication_user_name',
    ->     MASTER_PASSWORD='replication_password',
    ->     MASTER_LOG_FILE='recorded_log_file_name',
    ->     MASTER_LOG_POS=recorded_log_position;
```

如果是全新服务器，空数据库可以忽略MASTER_LOG_FILE与MASTER_LOG_POS

```
CHANGE MASTER TO MASTER_HOST='192.168.245.129',
MASTER_USER='replication', MASTER_PASSWORD='slavepass';
```

如果是复制已经存在的数据库需要MASTER_LOG_FILE与MASTER_LOG_POS选项

首先到Master上运行 show master status 找到File与Position

```
mysql> show master status \G
*************************** 1. row
***************************
            File: mysql-bin.000009
        Position: 3988
    Binlog_Do_DB:
Binlog_Ignore_DB:
1 row in set (0.00 sec)
```

```
CHANGE MASTER TO
     MASTER_HOST='192.168.2.1',
     MASTER_USER='replication',
     MASTER_PASSWORD='kJZBTo3BjMx9AnmD9Ryn',
     MASTER_LOG_FILE='mysql-bin.000009',
     MASTER_LOG_POS=3988;
```

4. 启动从服务器线程

```
mysql> START SLAVE;
Query OK, 0 rows affected (0.00 sec)
```

5. SLAVE STATUS

```
mysql> SHOW SLAVE STATUS\G
*************************** 1. row
***************************
             Slave_IO_State: Connecting to master
                Master_Host: 192.168.245.129
                Master_User: repl
                Master_Port: 3306
              Connect_Retry: 60
            Master_Log_File:
        Read_Master_Log_Pos: 4
             Relay_Log_File: mysqld-relay-bin.000002
              Relay_Log_Pos: 98
      Relay_Master_Log_File:
           Slave_IO_Running: Yes
          Slave_SQL_Running: Yes
            Replicate_Do_DB:
        Replicate_Ignore_DB:
         Replicate_Do_Table:
     Replicate_Ignore_Table:
    Replicate_Wild_Do_Table:
Replicate_Wild_Ignore_Table:
                 Last_Errno: 0
                 Last_Error:
               Skip_Counter: 0
        Exec_Master_Log_Pos: 0
            Relay_Log_Space: 98
            Until_Condition: None
             Until_Log_File:
              Until_Log_Pos: 0
         Master_SSL_Allowed: No
         Master_SSL_CA_File:
         Master_SSL_CA_Path:
            Master_SSL_Cert:
          Master_SSL_Cipher:
             Master_SSL_Key:
      Seconds_Behind_Master: NULL
1 row in set (0.00 sec)
```

## Testing

1. 登录 master

   复制进程的信息

   SHOW PROCESSLIST语句可以提供在主服务器上和从服务器上发生的关于复制的信息

```
mysql> SHOW PROCESSLIST\G
*************************** 1. row
***************************
     Id: 62
   User: replication
   Host: ken-hx409.local:36465
     db: NULL
Command: Binlog Dump
   Time: 679
  State: Has sent all binlog to slave; waiting for binlog to
be updated
   Info: NULL
*************************** 2. row
***************************
     Id: 64
   User: root
   Host: localhost
     db: NULL
Command: Query
   Time: 0
  State: NULL
   Info: SHOW PROCESSLIST
2 rows in set (0.00 sec)
```

2. 登录从库，查看复制线程

```
mysql> SHOW PROCESSLIST\G
*************************** 1. row
***************************
```

```
     Id: 273
   User: root
   Host: localhost
     db: NULL
Command: Query
   Time: 0
  State: NULL
   Info: SHOW PROCESSLIST
*************************** 2. row
*************************** 
     Id: 276
   User: system user
   Host: 
     db: NULL
Command: Connect
   Time: 2
  State: Waiting for master to send event
   Info: NULL
*************************** 3. row
*************************** 
     Id: 277
   User: system user
   Host: 
     db: NULL
Command: Connect
   Time: 2
  State: Has read all relay log; waiting for the slave I/O
thread to update it
   Info: NULL
3 rows in set (0.00 sec)
```

如果没有复制进程，请使用start slave;启动

```
mysql> SHOW PROCESSLIST\G
*************************** 1. row
*************************** 
     Id: 273
   User: root
   Host: localhost
     db: NULL
Command: Query
```

```
   Time: 0
  State: NULL
   Info: SHOW PROCESSLIST
1 row in set (0.02 sec)

mysql> start slave;
Query OK, 0 rows affected (0.00 sec)
```

3. 登录 master

```
mysql> insert into foo(id,data) values(2,'Hello world!!!');
Query OK, 1 row affected (0.00 sec)
```

4. 登录 slave

```
mysql> select * from foo;
```

在master服务器上插入一条记录，你可以立刻在slave服务器上看到变化。

## 将现有数据库迁移到主从结构数据库

数据库已经存在的情况下怎么迁移

1. Master 锁表禁止写入新数据

```
mysql> FLUSH TABLES WITH READ LOCK;
```

2. Slave 停止复制进程

```
mysql> stop slave;
```

3. 备份Master数据库

```
mysqldump yourdb | gzip > yourdb.sql.gz
```

4. 恢复数据库

如果使用mysqldump备份主服务器的数据，将转储文件装载到从服务器

```
# zcat yourdb.sql.gz | mysql -u root -p yourdb
```

5. 启动 Slave 复制进程

```
mysql> start slave;
```

6. 解除 Master 表锁定

```
mysql> UNLOCK TABLES;
```

MyIASM引擎可以采用下面方法

备份数据库

```
# tar zcvf mysql-snapshot.tar.gz /var/lib/mysql/neo
```

复制给从数据库

```
scp mysql-snapshot.tar.gz  neo@192.168.245.131:/tmp
```

snapshot 恢复

```
$ tar zxvf mysql-snapshot.tar.gz
$ cd /var/lib/mysql

$ mv /tmp/var/lib/mysql/neo .
$ sudo chown mysql.mysql -R neo
```

重新启动Mysql

```
$ sudo /etc/init.d/mysql restart
```

有兴趣可以看看mysqlhotcopy

## 主从复制安全问题

复制帐号权限

```
grant replication slave on *.* to 'replication'@'192.168.1.%'
identified by '000000';
```

主库数据库操作帐号权限

```
grant DELETE, INSERT, SELECT, UPDATE ON your_user.* to
yourdb@'your_host' identified by 'password' with grant option;
```

从库数据库操作帐号权限

```
grant SELECT ON your_user.* to yourdb@'your_host' identified by
'password' with grant option;
```

从库必须收回写操作

## 3.2. Master Master(主主)

### Master A

my.cnf 文件加入下面的内容

```
cp /etc/mysql/my.cnf /etc/mysql/my.cnf.old
vim /etc/mysql/my.cnf

[mysqld]
server-id = 1
log-bin=/data/mysql/binlog/binlog
binlog-do-db = test
binlog-ignore-db=mysql

log-slave-updates
sync_binlog=1
auto_increment_offset=1
auto_increment_increment=2
replicate-do-db = test
replicate-ignore-db = mysql,information_schema
```

创建复制权限

```
grant replication slave on *.* to 'replication'@'192.168.1.%'
identified by '000000';
flush privileges;
```

```
mysql>flush tables with read lock;

mysql> show master status\G
*************************** 1. row ***************************
File: binlog.000007
Position: 107
Binlog_Do_DB: test
Binlog_Ignore_DB: mysql
1 row in set (0.00 sec)
```

**Master B**

创建复制权限

```
grant replication slave on *.* to 'replication'@'192.168.1.%'
identified by '000000';
flush privileges;
```

my.cnf 文件加入下面的内容

```
[mysqld]
server-id = 2
log-bin = /data/mysql/binlog/binlog
replicate-do-db = test
replicate-ignore-db = mysql,information_schema

binlog-do-db = test
binlog-ignore-db=mysql
log-slave-updates
sync_binlog=1
auto_increment_offset=2
```

```
auto_increment_increment=2
```

B 与 A 配置文件不同的两处。

```
server-id = 2
auto_increment_offset=2
```

```
mysql> show master status\G
*************************** 1. row ***************************
File: binlog.000005
Position: 107
Binlog_Do_DB: test
Binlog_Ignore_DB: mysql
1 row in set (0.00 sec)
```

**将Master A 数据库 同步到 Master B 两端数据库内容保持一致**

Master A，首先锁表为只读状态

```
mysqldump --databases test > /tmp/test.sql
```

Master B 创建一个与Master A同名的空数据库,然后将备份文件恢复到数据库中

```
# mysql
mysql> CREATE DATABASE test;
mysql>\q

# scp 192.168.1.1:/tmp/test.sql ./
# mysql -uroot -p test < /tmp/test.sql
```

```

```

**Master A - B 同步两端数据库**

master-A

```
mysql>change master to master_host='192.168.1.2',
master_user='replication', master_password='000000',
master_log_file='binlog.000005', master_log_pos=107;
```

master-B

```
mysql>change master to master_host='192.168.1.1',
master_user='replication', master_password='000000',
master_log_file='binlog.000007', master_log_pos=107;
```

**Master A 数据库解除只读权限**

Master A 解锁

```
mysql> UNLOCK TABLES;
```

查看主主的工作状态

分别在Master A与B 上运行

```

```

```
mysql>show slave status\G;

Slave_IO_Running: Yes
Slave_SQL_Running: Yes
```

## 3.3. Semisynchronous Replication

### Master

```
mysql> SHOW VARIABLES LIKE "have_dynamic_loading";
+----------------------+-------+
| Variable_name        | Value |
+----------------------+-------+
| have_dynamic_loading | YES   |
+----------------------+-------+
1 row in set (0.00 sec)

mysql>
```

Master 配置

```
mysql> install plugin rpl_semi_sync_master SONAME
'semisync_master.so';
mysql> set global rpl_semi_sync_master_enabled = 1;
mysql> set global rpl_semi_sync_master_timeout = 30;
mysql> select * from mysql.plugin;
+----------------------+--------------------+
| name                 | dl                 |
+----------------------+--------------------+
| rpl_semi_sync_master | semisync_master.so |
+----------------------+--------------------+
1 row in set (0.00 sec)
```

状态查看

```
mysql> SHOW VARIABLES LIKE "%semi%";
+-------------------------------------+-------+
| Variable_name                       | Value |
+-------------------------------------+-------+
| rpl_semi_sync_master_enabled        | ON    |
| rpl_semi_sync_master_timeout        | 10    |
| rpl_semi_sync_master_trace_level    | 32    |
| rpl_semi_sync_master_wait_no_slave | ON    |
+-------------------------------------+-------+
4 rows in set (0.00 sec)
```

**Slave 配置**

```
install plugin rpl_semi_sync_slave soname 'semisync_slave.so';
set global rpl_semi_sync_slave_enabled = 1;
stop slave io_thread;
start slave io_thread;
```

Slave 状态查看

```
show global status like 'rpl_semi%';
```

**卸载插件**

卸载插件 UNINSTALL PLUGIN plugin_name

```
UNINSTALL PLUGIN rpl_semi_sync_master;
UNINSTALL PLUGIN rpl_semi_sync_slave;
```

**my.cnf**

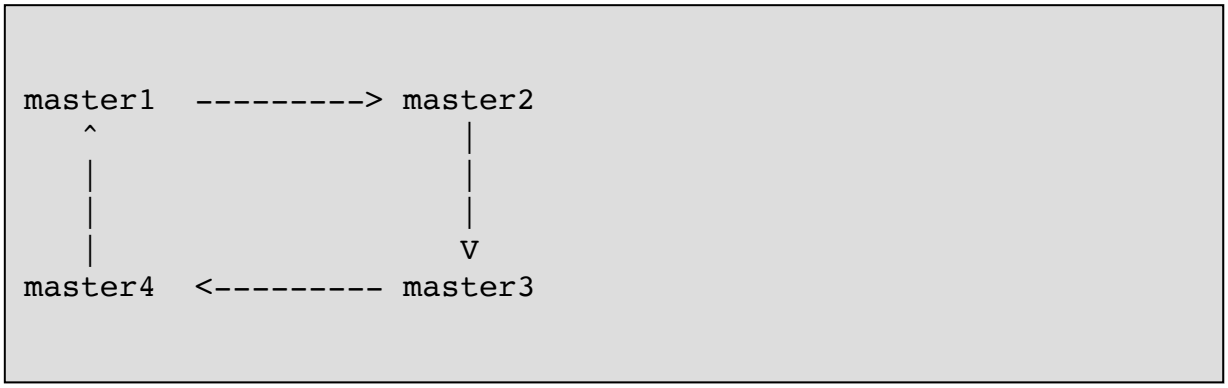
编辑my.cnf加入

```
# On Master
[mysqld]
rpl_semi_sync_master_enabled=1
rpl_semi_sync_master_timeout=1000 # 1 second

# On Slave
[mysqld]
rpl_semi_sync_slave_enabled=1
```

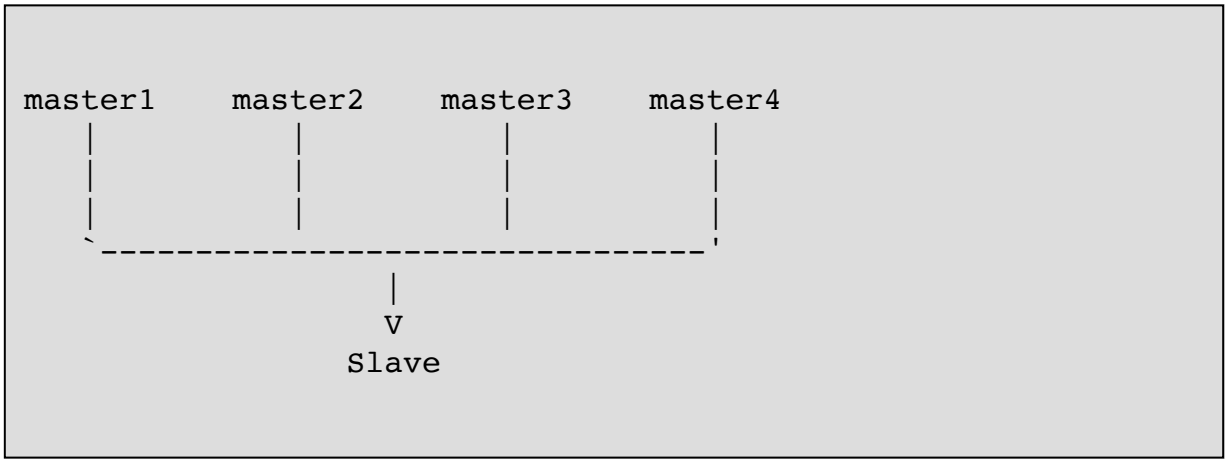
## 3.4. multi-master replication

MySQL 5.7 以上版本才能使用




## 3.5. multi-source replication

MySQL 5.7 以上版本才能使用



slave 配置

```
slave> change master to master_host="172.16.0.1",
master_port=3306,
master_user="replication",master_password="password" for
channel="master1";
slave> change master to master_host="172.16.0.2",
master_port=3306,
master_user="replication",master_password="password" for
channel="master2";

slave> start slave for channel="master1";
slave> start slave for channel="master2";
```

检查从服务器状态

```
slave > SHOW SLAVE STATUS FOR CHANNEL="master1"\G
slave > SHOW SLAVE STATUS FOR CHANNEL="master2"\G
```

测试,分别在两个主服务器上创建数据库，然后查看从数据库同步结果.

```
master1 > create database master1;
master2 > create database master2;

slave > show databases like 'master%';
+--------------------+
| Database (master%) |
+--------------------+
| master1            |
| master2            |
+--------------------+
```

```

```

## 3.6. 与复制有关的问题

### 主从不同步问题

执行下面语句

```
stop slave;
set global sql_slave_skip_counter =1 ;
start slave;

mysql> slave stop;
mysql> set GLOBAL SQL_SLAVE_SKIP_COUNTER=1;
mysql> slave start;
```

Seconds_Behind_Master 值从NULL变为大于等于0是表示已经同步

```
Seconds_Behind_Master: NULL
Seconds_Behind_Master: 8893
```

### mysql-bin 清理问题

缺省expire-logs-days为30天。这里设为7天，可根据自己情况调整。

```
[mysqld]
expire-logs-days = 7
```

通过SQL删除

```
删除某个日志:      mysql>PURGE MASTER LOGS TO ‘mysql-bin.015’;
删除某天前的日志:  mysql>PURGE MASTER LOGS BEFORE ’2010-10-25
```

```
14:00:00';
```

### 跳过 Last_Errno

修改mysql配置文件 /etc/my.cnf 在 [mysqld]下加一行

```
[mysqld]
slave_skip_errors = 1062
```

跳过所有错误

```
slave-skip-errors=all
```

### 重置Slave

```
STOP SLAVE;
RESET SLAVE;
START SLAVE;
```

## 3.7. GTID

5.6 新增功能

```
mysql> show global variables like '%gtid%';
+--------------------------------+-------+
| Variable_name                  | Value |
+--------------------------------+-------+
| binlog_gtid_simple_recovery    | OFF   |
| enforce_gtid_consistency       | OFF   |
| gtid_executed                  |       |
| gtid_mode                      | OFF   |
| gtid_owned                     |       |
| gtid_purged                    |       |
```

```
| simplified_binlog_gtid_recovery | OFF   |
+---------------------------------+-------+
7 rows in set (0.00 sec)
```

**Master**

```
[root@master mysql]# vim my.cnf

binlog-format=ROW
log-slave-updates=true
gtid-mode=on
enforce-gtid-consistency=true
master-info-repository=TABLE
relay-log-info-repository=TABLE
sync-master-info=1
slave-parallel-workers=2

binlog-checksum=CRC32
master-verify-checksum=1
slave-sql-verify-checksum=1
binlog-rows-query-log_events=1
port=3306
report-host=192.168.1.21
report-port=3306
server_id = 1
```

创建有复制权限的用户

```
GRANT REPLICATION SLAVE,REPLICATION CLIENT ON *.* TO
'replication'@'%.mydomain.com' IDENTIFIED BY 'slavepass';
FLUSH PRIVILEGES;
```

**Slave**

```
[root@slave mysql]# vim my.cnf

relay-log = relay-log
```

```
relay-log-index = relay-log.index
binlog-format=ROW
log-slave-updates=true

gtid-mode=on
enforce-gtid-consistency=true

master-info-repository=TABLE
relay-log-info-repository=TABLE
sync-master-info=1
slave-parallel-workers=2

;binlog-checksum=CRC32
master-verify-checksum=1
slave-sql-verify-checksum=1

binlog-rows-query-log_events=1
report-port=3306
port=3306
report-host=192.168.1.22
server_id = 10
```

登录到Master并进行复制

```
CHANGE MASTER TO
     MASTER_HOST='192.168.2.1',
     MASTER_USER='replication',
     MASTER_PASSWORD='kJZBTo3BjMx9AnmD9Ryn',
     MASTER_AUTO_POSITION=1;
```

就这么简单，你不再需要指定 MASTER_LOG_FILE='mysql-bin.000009', MASTER_LOG_POS=3988 两个参数。

# 4. MySQL Custer

The cluster need a lot of server for experiments, if you haven't any server for one, I have a good idea that using Vmware for you.

at first, let's create lots of virtual machine(You MUST have a third server). and then follow me step by step learning how to set up a mysql cluster on your virtual machine.


```
mgm             192.168.0.1             # Management
data            192.168.0.2             # Ndbd Node
data            192.168.0.3             # Ndbd Node
sql                     192.168.0.4             # SQL Node
sql                     192.168.0.5             # SQL Node
```

## 4.1. Management node (MGM node)

```
neo@mgm:~$ sudo vim /var/lib/mysql-cluster/config.ini

[NDBD DEFAULT]
NoOfReplicas=2
DataMemory=80M
IndexMemory=18M

[MYSQLD DEFAULT]

[NDB_MGMD DEFAULT]

[TCP DEFAULT]
portnumber=2202

[NDB_MGMD]
hostname=192.168.0.1
datadir=/var/lib/mysql-cluster
```

```
[NDBD]
hostname=192.168.0.2
datadir=/var/lib/mysql-cluster

[NDBD]
hostname=192.168.0.3
datadir=/var/lib/mysql-cluster

[MYSQLD]
hostname=192.168.0.4

[MYSQLD]
hostname=192.168.0.5
```

## 4.2. Data node

my.cnf

```
neo@data:~$ sudo vim /etc/mysql/my.cnf

[mysqld]
ndbcluster
ndb-connectstring=192.168.0.1   # the IP of the MANAGMENT
SERVER
[mysql_cluster]
ndb-connectstring=192.168.0.1   # the IP of the MANAGMENT
SERVER
```

## 4.3. SQL node

my.cnf

```
neo@sql:~$ sudo vim /etc/mysql/my.cnf

[mysqld]
ndbcluster
ndb-connectstring=192.168.0.1   # the IP of the MANAGMENT
```

```
SERVER
[mysql_cluster]
ndb-connectstring=192.168.0.1   # the IP of the MANAGMENT
SERVER
```

## 4.4. Starting

1. starting mgm

```
neo@mgm:~$ sudo ndb_mgmd -f /var/lib/mysql-
cluster/config.ini
```

2. initial ndbd

```
neo@data:~$ sudo ndbd --initial
```

首次运行需要 --initial 参数，以后不需要。

## 4.5. Shutdown

MGM

```
$ sudo ndb_mgm -e shutdown
```

## 4.6. Testing

```
neo@mgm:~$ ndb_mgm
-- NDB Cluster -- Management Client --
ndb_mgm> show
Connected to Management Server at: localhost:1186
Cluster Configuration
---------------------
```

```
[ndbd(NDB)]     2 node(s)
id=2    @192.168.0.2  (Version: 5.0.51, Nodegroup: 0)
id=3    @192.168.0.3  (Version: 5.0.51, Nodegroup: 0, Master)

[ndb_mgmd(MGM)] 1 node(s)
id=1    @192.168.0.1  (Version: 5.0.51)

[mysqld(API)]   2 node(s)
id=4    @192.168.0.4  (Version: 5.0.51)
id=5    @192.168.0.5  (Version: 5.0.51)

ndb_mgm>
```

**与没有使用簇的MySQL相比，在MySQL簇内操作数据的方式没有太大的区别。**

执行这类操作时应记住三点

1. 表必须用ENGINE=NDB或ENGINE=NDBCLUSTER选项创建，或用ALTER TABLE选项更改，以使用NDB Cluster存储引擎在 Cluster内复制它们。如果使用mysqldump的输出从已有数据库导入表，可在文本编辑器中打开SQL脚本，并将该选项添加到任何表创建语句，或用这类选项之一替换任何已有的 ENGINE（或TYPE）选项。

2. 另外还请记住，每个NDB表必须有一个主键。如果在创建表时用户未定义主键，NDB Cluster存储引擎将自动生成隐含的主键。（注释：该隐含 键也将占用空间，就像任何其他的表索引一样。由于没有足够的内存来容纳这些自动创建的键，出现问题并不罕见）。

3. 当你在一个节点上运行create database mydb;你去其他sql node上执行show databases;将不能看到mydb,你需要创建它，然后use mydb; show tables;你将看到同步的表。

SQL Node 1

```
neo@sql:~$ mysql -uroot -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 7
Server version: 5.0.51a-3ubuntu5.1 (Ubuntu)
```

```
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> create database cluster;
Query OK, 1 row affected (0.00 sec)

mysql> use cluster
Database changed
mysql> create table city( id mediumint unsigned not null
auto_increment primary key, name varchar(20) not null default
'' ) engine = ndbcluster default charset utf8;
Query OK, 0 rows affected (1.07 sec)

mysql> insert into city values(1, 'Shenzhen');
Query OK, 1 row affected (0.12 sec)

mysql> insert into city values(2, 'Guangdong');
Query OK, 1 row affected (0.00 sec)
```

SQL Node 2

```
neo@sql:~$ mysql -uroot -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 7
Server version: 5.0.51a-3ubuntu5.1 (Ubuntu)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| example            |
| mydb               |
| mysql              |
| neo                |
+--------------------+
6 rows in set (0.13 sec)
```

```
mysql> create database cluster;
Query OK, 1 row affected (0.00 sec)

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| cluster            |
| example            |
| mydb               |
| mysql              |
| neo                |
+--------------------+
6 rows in set (0.13 sec)

mysql> use cluster;
Reading table information for completion of table and column
names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+------------------+
| Tables_in_cluster |
+------------------+
| city             |
+------------------+
1 row in set (0.01 sec)

mysql> select * from city;
+----+-----------+
| id | name      |
+----+-----------+
|  1 | Shenzhen  |
|  2 | Guangdong |
+----+-----------+
2 rows in set (0.03 sec)

mysql>
```

# 5. MySQL Proxy

## 5.1. Ubuntu

安装环境 Ubuntu 13.04

```
$ sudo apt-get install mysql-proxy
```

ENABLED改为true

```
$ sudo vim /etc/default/mysql-proxy
ENABLED="true"
OPTIONS="--defaults-file=/etc/mysql/mysql-proxy.cnf"
```

配置 /etc/mysql/mysql-proxy.cnf

```
$ sudo vim /etc/mysql/mysql-proxy.cnf

[mysql-proxy]
daemon = true
user = mysql
proxy-skip-profiling = true
keepalive = true
max-open-files = 2048
event-threads = 50
pid-file = /var/run/mysql-proxy.pid
log-file = /var/log/mysql-proxy.log
log-level = debug
admin-address=:4401
admin-username=admin
admin-password=passw0rd
admin-lua-script=/usr/local/lib/mysql-proxy/lua/admin.lua
proxy-address = 0.0.0.0:3307
proxy-backend-addresses = 192.168.2.1:3306
proxy-read-only-backend-addresses=192.168.6.2:3306,
192.168.6.1:3306
```

```
proxy-lua-script=/usr/lib/mysql-proxy/lua/proxy/balance.lua
```

修改权限，这个步骤不能省略，否则无法启动。

```
$ sudo chmod 0660 /etc/mysql/mysql-proxy.cnf
```

启动mysql-proxy

```
$ sudo /etc/init.d/mysql-proxy start
 * Starting MySQL Proxy daemon...                    [ OK ]
```

测试3307端口

```
$ mysql -hlocalhost -P3307 -uroot -p
```

mysql-proxy 软件包所含文件如下:

```
$ dpkg -L  mysql-proxy
/.
/etc
/etc/default
/etc/default/mysql-proxy
/etc/init.d
/etc/init.d/mysql-proxy
/usr
/usr/share
/usr/share/mysql-proxy
/usr/share/mysql-proxy/active-queries.lua
/usr/share/mysql-proxy/active-transactions.lua
/usr/share/mysql-proxy/admin-sql.lua
/usr/share/mysql-proxy/admin.lua
/usr/share/mysql-proxy/analyze-query.lua
/usr/share/mysql-proxy/auditing.lua
/usr/share/mysql-proxy/commit-obfuscator.lua
/usr/share/mysql-proxy/histogram.lua
```

```
/usr/share/mysql-proxy/load-multi.lua
/usr/share/mysql-proxy/ro-balance.lua
/usr/share/mysql-proxy/ro-pooling.lua
/usr/share/mysql-proxy/rw-splitting.lua
/usr/share/mysql-proxy/xtab.lua
/usr/share/doc
/usr/share/doc/mysql-proxy
/usr/share/doc/mysql-proxy/README.TESTS.gz
/usr/share/doc/mysql-proxy/README
/usr/share/doc/mysql-proxy/copyright
/usr/share/doc/mysql-proxy/changelog.Debian.gz
/usr/lib
/usr/lib/libmysql-chassis-glibext.so.0.0.0
/usr/lib/libmysql-chassis-timing.so.0.0.0
/usr/lib/libmysql-chassis.so.0.0.0
/usr/lib/libmysql-proxy.so.0.0.0
/usr/lib/mysql-proxy
/usr/lib/mysql-proxy/lua
/usr/lib/mysql-proxy/lua/proxy
/usr/lib/mysql-proxy/lua/proxy/auto-config.lua
/usr/lib/mysql-proxy/lua/proxy/balance.lua
/usr/lib/mysql-proxy/lua/proxy/commands.lua
/usr/lib/mysql-proxy/lua/proxy/parser.lua
/usr/lib/mysql-proxy/lua/proxy/tokenizer.lua
/usr/lib/mysql-proxy/lua/proxy/test.lua
/usr/lib/mysql-proxy/lua/admin.lua
/usr/lib/mysql-proxy/lua/lfs.so
/usr/lib/mysql-proxy/lua/glib2.so
/usr/lib/mysql-proxy/lua/chassis.so
/usr/lib/mysql-proxy/lua/mysql.so
/usr/lib/mysql-proxy/lua/lpeg.so
/usr/lib/mysql-proxy/lua/posix.so
/usr/lib/mysql-proxy/plugins
/usr/lib/mysql-proxy/plugins/libadmin.so
/usr/lib/mysql-proxy/plugins/libproxy.so
/usr/lib/mysql-proxy/plugins/libreplicant.so
/usr/lib/mysql-proxy/plugins/libdebug.so
/usr/lib/pkgconfig
/usr/lib/pkgconfig/mysql-proxy.pc
/usr/lib/pkgconfig/mysql-chassis.pc
/usr/bin
/usr/bin/mysql-binlog-dump
/usr/bin/mysql-myisam-dump
/usr/bin/mysql-proxy
/usr/include
```

```
/usr/include/network-mysqld.h
/usr/include/network-mysqld-lua.h
/usr/include/network-mysqld-proto.h
/usr/include/network-mysqld-binlog.h
/usr/include/network-mysqld-packet.h
/usr/include/network-mysqld-masterinfo.h
/usr/include/network-conn-pool.h
/usr/include/network-conn-pool-lua.h
/usr/include/network-queue.h
/usr/include/network-socket.h
/usr/include/network-socket-lua.h
/usr/include/network-address.h
/usr/include/network-address-lua.h
/usr/include/sys-pedantic.h
/usr/include/chassis-plugin.h
/usr/include/chassis-log.h
/usr/include/chassis-keyfile.h
/usr/include/chassis-mainloop.h
/usr/include/chassis-path.h
/usr/include/chassis-filemode.h
/usr/include/chassis-limits.h
/usr/include/chassis-event-thread.h
/usr/include/chassis-gtimeval.h
/usr/include/glib-ext.h
/usr/include/glib-ext-ref.h
/usr/include/string-len.h
/usr/include/lua-load-factory.h
/usr/include/lua-scope.h
/usr/include/lua-env.h
/usr/include/network-injection.h
/usr/include/network-injection-lua.h
/usr/include/chassis-shutdown-hooks.h
/usr/include/chassis-exports.h
/usr/include/network-exports.h
/usr/include/network-backend.h
/usr/include/network-backend-lua.h
/usr/include/disable-dtrace.h
/usr/include/lua-registry-keys.h
/usr/include/chassis-stats.h
/usr/include/chassis-timings.h
/usr/include/chassis-frontend.h
/usr/include/chassis-options.h
/usr/include/chassis-win32-service.h
/usr/include/chassis-unix-daemon.h
/usr/include/my_rdtsc.h
```

```
/usr/lib/libmysql-chassis-glibext.so.0
/usr/lib/libmysql-chassis-glibext.so
/usr/lib/libmysql-proxy.so
/usr/lib/libmysql-chassis-timing.so.0
/usr/lib/libmysql-chassis-timing.so
/usr/lib/libmysql-proxy.so.0
/usr/lib/libmysql-chassis.so.0
/usr/lib/libmysql-chassis.so
```

## 5.2. CentOS

```
# yum install mysql-proxy
```

```
# cat /etc/sysconfig/mysql-proxy
# Options for mysql-proxy
ADMIN_USER="admin"
ADMIN_PASSWORD=""
ADMIN_LUA_SCRIPT="/usr/lib64/mysql-proxy/lua/admin.lua"
PROXY_USER="mysql-proxy"
PROXY_OPTIONS="--daemon --log-level=info --log-use-syslog"
```

修改PROXY_OPTIONS选项

```
#PROXY_OPTIONS="--daemon --log-level=info --log-use-syslog"
PROXY_OPTIONS="--defaults-file=/etc/mysql/mysql-proxy.cnf"
```

```
# mkdir /etc/mysql
# vim /etc/mysql/mysql-proxy.cnf

[mysql-proxy]
daemon = true
user = mysql-proxy
proxy-skip-profiling = true
keepalive = true
;max-open-files = 2048
event-threads = 512
```

```
pid-file = /var/run/mysql-proxy.pid
log-file = /var/log/mysql-proxy.log
log-level = debug
admin-address=:4401
admin-username=admin
admin-password=passw0rd
admin-lua-script=/usr/lib64/mysql-proxy/lua/admin.lua
proxy-address = 0.0.0.0:3307
proxy-backend-addresses = 192.168.2.1:3306
proxy-read-only-backend-addresses=192.168.6.2:3306,
192.168.6.1:3306
proxy-lua-script=/usr/lib64/mysql-proxy/lua/proxy/balance.lua
```

修复启动脚本BUG

```
# vim /etc/init.d/mysql-proxy

#daemon $prog $PROXY_OPTIONS --pid-file=$PROXY_PID --
user=$PROXY_USER --admin-username="$ADMIN_USER" --admin-lua-
script="$ADMIN_LUA_SCRIPT" --admin-password="$ADMIN_PASSWORD"
注视这行，改为下面的一行代码
daemon $prog $PROXY_OPTIONS --pid-file=$PROXY_PID
```

启动mysql-proxy

```
# chkconfig mysql-proxy on
# service mysql-proxy start
Starting mysql-proxy:                                      [
OK  ]
```

**FAQ**

(critical) (libevent) evsignal_init: socketpair: Too many open files

```
;max-open-files = 2048
```

注释max-open-files = 2048，使用ulimit -SHn 2048设置

# 6. MySQL Router

## 6.1. 安装 MySQL Router

```
# yum install mysql-router -y
```

MySQL Router 软件包中所含的文件。

```
[root@netkiller ~]# rpm -ql mysql-router-2.0.3-1.el7
/etc/mysqlrouter
/etc/mysqlrouter/mysqlrouter.ini
/usr/lib/systemd/system/mysqlrouter.service
/usr/lib/tmpfiles.d/mysqlrouter.conf
/usr/lib64/libmysqlharness.so
/usr/lib64/libmysqlharness.so.0
/usr/lib64/libmysqlrouter.so
/usr/lib64/libmysqlrouter.so.1
/usr/lib64/mysqlrouter
/usr/lib64/mysqlrouter/fabric_cache.so
/usr/lib64/mysqlrouter/keepalive.so
/usr/lib64/mysqlrouter/logger.so
/usr/lib64/mysqlrouter/mysql_protocol.so
/usr/lib64/mysqlrouter/routing.so
/usr/sbin/mysqlrouter
/usr/share/doc/mysql-router-2.0.3
/usr/share/doc/mysql-router-2.0.3/License.txt
/usr/share/doc/mysql-router-2.0.3/README.txt
/var/log/mysqlrouter
/var/run/mysqlrouter
```

## 6.2. 配置 MySQL Router

默认配置

```
# cat /etc/mysqlrouter/mysqlrouter.ini
```

```
# Copyright (c) 2015, Oracle and/or its affiliates. All rights
reserved.
#
# This program is free software; you can redistribute it and/or
modify
# it under the terms of the GNU General Public License as
published by
# the Free Software Foundation; version 2 of the License.
#
# This program is distributed in the hope that it will be
useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty
of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public
License
# along with this program; if not, write to the Free Software
# Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA
02110-1301 USA

#
# MySQL Router configuration file
#
# Documentation is available at
#    http://dev.mysql.com/doc/mysql-router/en/

[DEFAULT]
logging_folder = /var/log/mysqlrouter/
plugin_folder = /usr/lib64/mysqlrouter
runtime_folder = /var/run/mysqlrouter
config_folder = /etc/mysqlrouter

[logger]
level = info

# If no plugin is configured which starts a service, keepalive
# will make sure MySQL Router will not immediately exit. It is
# safe to remove once Router is configured.
[keepalive]
interval = 60
```

主备配置

适用于 MySQL Master-Master / Master-Slave 方案，当一台Master出现故障后另一台Master或者Slave接管

```
[routing:failover]
bind_address = 192.168.0.5
bind_port = 3306
max_connections = 1024
mode = read-write
destinations = 192.168.0.10:3306,192.168.0.11:3306
```

负载均衡配置

主要用于输在均衡

```
[routing:balancing]
bind_address = 192.168.0.5
bind_port = 3307
connect_timeout = 3
max_connections = 1024
mode = read-only
destinations = 192.168.0.20:3306,192.168.0.21:3306
```

## 6.3. MySQL Router ， Haproxy，LVS 的选择

MySQL Router目前仍在成长中，如果你只需要负载均衡与主备，那么LVS性能更高，Haproxy也更成熟。

# 7. my.cnf

## 7.1. bind-address

MySQL 通过 yum 安装后默认是监听 127.0.0.1 / ::1 如果你希望从其他IP访问3306端口，需要修改绑定地址为 0.0.0.0

```
bind-address=127.0.0.1
```

0.0.0.0:3306

```
bind-address = 0.0.0.0
```

指定IP地址

```
[root@localhost ~]# ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp2s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel
state UP group default qlen 1000
    link/ether 00:e0:70:81:9e:48 brd ff:ff:ff:ff:ff:ff
    inet 192.168.30.10/24 brd 192.168.30.255 scope global noprefixroute
enp2s0
       valid_lft forever preferred_lft forever
    inet 192.168.30.11/24 brd 192.168.30.255 scope global secondary
noprefixroute enp2s0
       valid_lft forever preferred_lft forever
    inet 192.168.30.12/24 brd 192.168.30.255 scope global secondary
noprefixroute enp2s0
       valid_lft forever preferred_lft forever
    inet 192.168.30.13/24 brd 192.168.30.255 scope global secondary
noprefixroute enp2s0
```

```
       valid_lft forever preferred_lft forever
    inet 192.168.30.14/24 brd 192.168.30.255 scope global secondary
noprefixroute enp2s0
       valid_lft forever preferred_lft forever
    inet6 fe80::2e0:70ff:fe81:9e48/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
3: wlp1s0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group
default qlen 1000
    link/ether 40:9f:38:b6:e0:55 brd ff:ff:ff:ff:ff:ff
4: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue
state DOWN group default
    link/ether 02:42:f0:6f:b3:4b brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
       valid_lft forever preferred_lft forever
45: br-a32fa1ca1437: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc
noqueue state DOWN group default
    link/ether 02:42:d7:ae:ea:5d brd ff:ff:ff:ff:ff:ff
    inet 172.18.0.1/16 brd 172.18.255.255 scope global br-a32fa1ca1437
       valid_lft forever preferred_lft forever
```

上面服务器上配置多个IP地址

```
bind-address=192.168.30.10
```

## 7.2. 禁用TCP/IP链接

与bind-address互斥，skip-networking 开启，只能通过UNIX SOCKET链接，而不能使用IP地址链接

```
[mysqld]
skip-networking
```

## 7.3. 配置字符集

### Configuring Database Character Encoding

```
```

```
mysql> SHOW VARIABLES LIKE 'character_set_%';
+--------------------------+----------------------------+
| Variable_name            | Value                      |
+--------------------------+----------------------------+
| character_set_client     | latin1                     |
| character_set_connection | latin1                     |
| character_set_database   | utf8                       |
| character_set_filesystem | binary                     |
| character_set_results    | latin1                     |
| character_set_server     | latin1                     |
| character_set_system     | utf8                       |
| character_sets_dir       | /usr/share/mysql/charsets/ |
+--------------------------+----------------------------+
8 rows in set (0.00 sec)
```

Server Character Set and Collation

```
shell> mysqld --character-set-server=latin1
shell> mysqld --character-set-server=latin1 \
          --collation-server=latin1_swedish_ci
```

$ vim /etc/mysql/my.cnf

```
[mysqld]
character-set-server=utf8
collation_server=utf8_general_ci
init_connect='SET NAMES utf8'

[client]
character_set_client=utf8
```

```
mysql --default-character-set=utf8 -u root -p
```

```
mysql> show variables like 'character%';
+--------------------------+----------------------------+
| Variable_name            | Value                      |
+--------------------------+----------------------------+
```

```
| character_set_client     | utf8                       |
| character_set_connection | utf8                       |
| character_set_database   | utf8                       |
| character_set_filesystem | binary                     |
| character_set_results    | utf8                       |
| character_set_server     | utf8                       |
| character_set_system     | utf8                       |
| character_sets_dir       | /usr/share/mysql/charsets/ |
+--------------------------+----------------------------+
8 rows in set (0.00 sec)
```

## 7.4. 最大链接数 max_connections

```
[mysqld]
max_connections=250
```

## 7.5. 默认引擎 storage-engine

```
[mysqld]
default-storage-engine=INNODB
```

## 7.6. max_allowed_packet

```
max_allowed_packet=8M
```

## 7.7. skip-name-resolve

跳过域名解析

```
# vim /etc/mysql/my.cnf

[mysqld]
skip-name-resolve
```

MySQL 登录缓慢，大量用户排队等待

```
mysql> SHOW FULL PROCESSLIST;
+-----+----------------------+-----------------------+------+---------
+------+-------+----------------------+
| Id  | User                 | Host                  | db   | Command |
Time | State | Info                 |
+-----+----------------------+-----------------------+------+---------
+------+-------+----------------------+
| 718 | unauthenticated user | 192.168.3.124:42075   | NULL | Connect |
NULL | login | NULL                 |
| 719 | unauthenticated user | 192.168.3.124:42073   | NULL | Connect |
NULL | login | NULL                 |
| 720 | unauthenticated user | 192.168.3.124:42074   | NULL | Connect |
NULL | login | NULL                 |
| 721 | unauthenticated user | 192.168.3.124:42077   | NULL | Connect |
NULL | login | NULL                 |
| 722 | unauthenticated user | 192.168.3.124:42076   | NULL | Connect |
NULL | login | NULL                 |
| 723 | unauthenticated user | 192.168.3.124:42079   | NULL | Connect |
NULL | login | NULL                 |
| 724 | unauthenticated user | 192.168.3.124:42078   | NULL | Connect |
NULL | login | NULL                 |
| 725 | unauthenticated user | 192.168.3.124:42081   | NULL | Connect |
NULL | login | NULL                 |
| 726 | unauthenticated user | 192.168.3.124:42080   | NULL | Connect |
NULL | login | NULL                 |
| 727 | unauthenticated user | 192.168.3.124:42082   | NULL | Connect |
NULL | login | NULL                 |
| 728 | unauthenticated user | 192.168.3.124:42083   | NULL | Connect |
NULL | login | NULL                 |
| 729 | unauthenticated user | 192.168.3.124:42085   | NULL | Connect |
NULL | login | NULL                 |
| 730 | unauthenticated user | 192.168.3.124:42084   | NULL | Connect |
NULL | login | NULL                 |
| 731 | unauthenticated user | 192.168.3.124:42086   | NULL | Connect |
NULL | login | NULL                 |
| 732 | unauthenticated user | 192.168.3.124:42087   | NULL | Connect |
NULL | login | NULL                 |
| 733 | unauthenticated user | 192.168.3.124:42088   | NULL | Connect |
NULL | login | NULL                 |
| 734 | unauthenticated user | 192.168.3.124:42089   | NULL | Connect |
NULL | login | NULL                 |
| 735 | unauthenticated user | 192.168.3.124:42090   | NULL | Connect |
NULL | login | NULL                 |
| 736 | unauthenticated user | 192.168.3.124:42091   | NULL | Connect |
NULL | login | NULL                 |
| 737 | unauthenticated user | 192.168.3.124:42092   | NULL | Connect |
NULL | login | NULL                 |
```

```
| 738 | unauthenticated user | 192.168.3.124:42093    | NULL | Connect |
NULL | login | NULL                  |
| 739 | unauthenticated user | 192.168.3.124:42094    | NULL | Connect |
NULL | login | NULL                  |
| 740 | unauthenticated user | 192.168.3.124:42095    | NULL | Connect |
NULL | login | NULL                  |
| 741 | unauthenticated user | 192.168.3.124:42096    | NULL | Connect |
NULL | login | NULL                  |
| 742 | unauthenticated user | 192.168.3.124:42097    | NULL | Connect |
NULL | login | NULL                  |
| 743 | unauthenticated user | 192.168.3.124:42098    | NULL | Connect |
NULL | login | NULL                  |
| 744 | unauthenticated user | 192.168.3.124:42099    | NULL | Connect |
NULL | login | NULL                  |
| 745 | unauthenticated user | 192.168.3.124:42100    | NULL | Connect |
NULL | login | NULL                  |
| 746 | unauthenticated user | 192.168.3.124:42101    | NULL | Connect |
NULL | login | NULL                  |
| 747 | unauthenticated user | 192.168.3.124:42102    | NULL | Connect |
NULL | login | NULL                  |
| 748 | unauthenticated user | 192.168.3.124:42103    | NULL | Connect |
NULL | login | NULL                  |
| 749 | unauthenticated user | 192.168.3.124:42104    | NULL | Connect |
NULL | login | NULL                  |
| 750 | unauthenticated user | 192.168.3.124:42068    | NULL | Connect |
NULL | login | NULL                  |
| 751 | unauthenticated user | 192.168.3.124:42064    | NULL | Connect |
NULL | login | NULL                  |
| 752 | unauthenticated user | 192.168.3.124:42071    | NULL | Connect |
NULL | login | NULL                  |
| 753 | unauthenticated user | 192.168.3.124:42072    | NULL | Connect |
NULL | login | NULL                  |
| 754 | unauthenticated user | 192.168.3.124:42067    | NULL | Connect |
NULL | login | NULL                  |
| 755 | unauthenticated user | 192.168.3.124:42070    | NULL | Connect |
NULL | login | NULL                  |
| 756 | unauthenticated user | 192.168.3.124:42069    | NULL | Connect |
NULL | login | NULL                  |
| 757 | unauthenticated user | 192.168.3.124:42065    | NULL | Connect |
NULL | login | NULL                  |
| 758 | unauthenticated user | 192.168.3.124:42112    | NULL | Connect |
NULL | login | NULL                  |
| 759 | unauthenticated user | 192.168.3.50:4872      | NULL | Connect |
NULL | login | NULL                  |
| 761 | unauthenticated user | 192.168.3.40:36363     | NULL | Connect |
NULL | login | NULL                  |
| 762 | neo                  | www.example.com:56200  | NULL | Query   |
0 | NULL  | SHOW FULL PROCESSLIST |
+-----+----------------------+------------------------+------+---------
+------+-------+----------------------+
44 rows in set (0.00 sec)
```

```
mysql> SHOW FULL PROCESSLIST;
+-----+----------------------+-----------------------+------+---------
+------+-------+----------------------+
| Id  | User                 | Host                  | db   | Command |
Time | State | Info                 |
+-----+----------------------+-----------------------+------+---------
+------+-------+----------------------+
| 718 | unauthenticated user | 192.168.3.124:42075   | NULL | Connect |
NULL | login | NULL                 |
| 719 | unauthenticated user | 192.168.3.124:42073   | NULL | Connect |
NULL | login | NULL                 |
| 720 | unauthenticated user | 192.168.3.124:42074   | NULL | Connect |
NULL | login | NULL                 |
| 721 | unauthenticated user | 192.168.3.124:42077   | NULL | Connect |
NULL | login | NULL                 |
| 722 | unauthenticated user | 192.168.3.124:42076   | NULL | Connect |
NULL | login | NULL                 |
| 723 | unauthenticated user | 192.168.3.124:42079   | NULL | Connect |
NULL | login | NULL                 |
| 724 | unauthenticated user | 192.168.3.124:42078   | NULL | Connect |
NULL | login | NULL                 |
| 725 | unauthenticated user | 192.168.3.124:42081   | NULL | Connect |
NULL | login | NULL                 |
| 726 | unauthenticated user | 192.168.3.124:42080   | NULL | Connect |
NULL | login | NULL                 |
| 727 | unauthenticated user | 192.168.3.124:42082   | NULL | Connect |
NULL | login | NULL                 |
| 728 | unauthenticated user | 192.168.3.124:42083   | NULL | Connect |
NULL | login | NULL                 |
| 729 | unauthenticated user | 192.168.3.124:42085   | NULL | Connect |
NULL | login | NULL                 |
| 730 | unauthenticated user | 192.168.3.124:42084   | NULL | Connect |
NULL | login | NULL                 |
| 731 | unauthenticated user | 192.168.3.124:42086   | NULL | Connect |
NULL | login | NULL                 |
| 732 | unauthenticated user | 192.168.3.124:42087   | NULL | Connect |
NULL | login | NULL                 |
| 733 | unauthenticated user | 192.168.3.124:42088   | NULL | Connect |
NULL | login | NULL                 |
| 734 | unauthenticated user | 192.168.3.124:42089   | NULL | Connect |
NULL | login | NULL                 |
| 735 | unauthenticated user | 192.168.3.124:42090   | NULL | Connect |
NULL | login | NULL                 |
| 736 | unauthenticated user | 192.168.3.124:42091   | NULL | Connect |
NULL | login | NULL                 |
| 737 | unauthenticated user | 192.168.3.124:42092   | NULL | Connect |
NULL | login | NULL                 |
| 738 | unauthenticated user | 192.168.3.124:42093   | NULL | Connect |
NULL | login | NULL                 |
| 739 | unauthenticated user | 192.168.3.124:42094   | NULL | Connect |
```

```
NULL | login | NULL                  |
| 740 | unauthenticated user | 192.168.3.124:42095    | NULL | Connect |
NULL | login | NULL                  |
| 741 | unauthenticated user | 192.168.3.124:42096    | NULL | Connect |
NULL | login | NULL                  |
| 742 | unauthenticated user | 192.168.3.124:42097    | NULL | Connect |
NULL | login | NULL                  |
| 743 | unauthenticated user | 192.168.3.124:42098    | NULL | Connect |
NULL | login | NULL                  |
| 744 | unauthenticated user | 192.168.3.124:42099    | NULL | Connect |
NULL | login | NULL                  |
| 745 | unauthenticated user | 192.168.3.124:42100    | NULL | Connect |
NULL | login | NULL                  |
| 746 | unauthenticated user | 192.168.3.124:42101    | NULL | Connect |
NULL | login | NULL                  |
| 747 | unauthenticated user | 192.168.3.124:42102    | NULL | Connect |
NULL | login | NULL                  |
| 748 | unauthenticated user | 192.168.3.124:42103    | NULL | Connect |
NULL | login | NULL                  |
| 749 | unauthenticated user | 192.168.3.124:42104    | NULL | Connect |
NULL | login | NULL                  |
| 750 | unauthenticated user | 192.168.3.124:42068    | NULL | Connect |
NULL | login | NULL                  |
| 751 | unauthenticated user | 192.168.3.124:42064    | NULL | Connect |
NULL | login | NULL                  |
| 752 | unauthenticated user | 192.168.3.124:42071    | NULL | Connect |
NULL | login | NULL                  |
| 753 | unauthenticated user | 192.168.3.124:42072    | NULL | Connect |
NULL | login | NULL                  |
| 754 | unauthenticated user | 192.168.3.124:42067    | NULL | Connect |
NULL | login | NULL                  |
| 755 | unauthenticated user | 192.168.3.124:42070    | NULL | Connect |
NULL | login | NULL                  |
| 756 | unauthenticated user | 192.168.3.124:42069    | NULL | Connect |
NULL | login | NULL                  |
| 757 | unauthenticated user | 192.168.3.124:42065    | NULL | Connect |
NULL | login | NULL                  |
| 758 | unauthenticated user | 192.168.3.124:42112    | NULL | Connect |
NULL | login | NULL                  |
| 759 | unauthenticated user | 192.168.3.50:4872      | NULL | Connect |
NULL | login | NULL                  |
| 761 | unauthenticated user | 192.168.3.40:36363     | NULL | Connect |
NULL | login | NULL                  |
| 762 | neo                  | www.example.com:56200  | NULL | Query   |
0 | NULL  | SHOW FULL PROCESSLIST |
+-----+----------------------+------------------------+------+---------
+------+-------+----------------------+
44 rows in set (0.00 sec)
```

解决方案 my.cnf 配置文件中加入skip-name-resolve

## 7.8. timeout

```
[mysqld]
wait_timeout=30
interactive_timeout=30
```

如果你没有修改过MySQL的配置，缺省情况下，wait_timeout的初始值是28800。

wait_timeout过大有弊端，其体现就是MySQL里大量的SLEEP进程无法及时释放，拖累系统性能，不过也不能把这个指设置的过小，否则你可能会遭遇到“MySQL has gone away”之类的问题，通常来说，我觉得把wait_timeout设置为10是个不错的选择，但某些情况下可能也会出问题，比如说有一个CRON脚本，其中两次SQL查询的间隔时间大于10秒的话，那么这个设置就有问题了：

(1)interactive_timeout 参数含义：服务器关闭交互式连接前等待活动的秒数。 参数默认值：28800秒（8小时）

(2)wait_timeout 参数含义：服务器关闭非交互连接之前等待活动的秒数。

## 7.9. 与复制有关的参数

### 用于主库的选项 Master

定义 log-bin 文件名

```
log-bin=mysql-bin
```

binlog 保留时间,过期天数设置

```
expire-logs-days = 30
```

```
binlog-do-db=需要复制的数据库名
binlog-ignore-db=不需要复制的数据库
```

用于从库的选项 **Slave**

```
replicate-do-db= 指定需要复制的数据库
replicate-ignore-db= 忽略复制的数据库
```

逃过错误

主从复制经常遇到 Last_Errno: 1062 可以使用下面配置跳过

```
slave_skip_errors=1062
```

## 7.10. 与 InnoDB 有关的配置项

```
innodb_file_per_table
```

配置后重启mysql运行下面命令将ibdata1拆分到tbl_name.ibd

```
OPTIMIZE TABLE tbl_name;
```

ls /var/lib/mysql/中查看 tbl_name.ibd文件

临时开启

```
SET @@global.innodb_file_per_table = 1;
```

## 7.11. EVENT 设置

开启EVENT定时任务

```
event_scheduler=on
```

## 7.12. 日志

操作日志

```
log = mysql.log
```

慢查询日志

```
log-slow-queries = slow.log
long_query_time = 5
```

错误日志

```
[mysqld_safe]
log-error=/var/log/mysqld.log
```

## 7.13. MySQL 5.7 my.cnf 实例

### 例 1.2. my.cnf

```
[root@netkiller ~]# cat /etc/my.cnf
# For advice on how to change settings please see
# http://dev.mysql.com/doc/refman/5.7/en/server-configuration-
defaults.html

[mysqld]
#
```

```
# Remove leading # and set to the amount of RAM for the most important
data
# cache in MySQL. Start at 70% of total RAM for dedicated server, else
10%.
# innodb_buffer_pool_size = 128M
#
# Remove leading # to turn on a very important data integrity option:
logging
# changes to the binary log between backups.
# log_bin
#
# Remove leading # to set options mainly useful for reporting servers.
# The server defaults are faster for transactions and fast SELECTs.
# Adjust sizes as needed, experiment to find the optimal values.
# join_buffer_size = 128M
# sort_buffer_size = 2M
# read_rnd_buffer_size = 2M
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock

# Disabling symbolic-links is recommended to prevent assorted security
risks
symbolic-links=0

log-error=/var/log/mysqld.log
pid-file=/var/run/mysqld/mysqld.pid

!includedir /etc/my.cnf.d
```

**例 1.3. my.cnf**

```
[root@netkiller ~]# cat /etc/my.cnf.d/default.cnf
[mysqld]
skip-name-resolve
max_connections=4096
default-storage-engine=INNODB

#wait_timeout=300
#interactive_timeout=300

character-set-server=utf8
collation_server=utf8_general_ci
init_connect='SET NAMES utf8'

explicit_defaults_for_timestamp=true
```

```
query_cache_type=1
query_cache_size=512M
table-open-cache=2000

#validate-password=OFF

sql_mode =
STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_Z
ERO,NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION

[client]
default-character-set=utf8
#character_set_client=utf8
```

## 7.14. Example for my.cnf

### 例 1.4. my.cnf

```
# Example MySQL config file for very large systems.
#
# This is for a large system with memory of 1G-2G where the system runs
mainly
# MySQL.
#
# In this file, you can use all long options that a program supports.
# If you want to know which options a program supports, run the program
# with the "--help" option.

# The following options will be passed to all MySQL clients
[client]
#password       = your_password
port            = 3306
socket          = /var/lib/mysql/mysql.sock

# Here follows entries for some specific programs
character-set-server=utf8

# The MySQL server
[mysqld]
port            = 3306
socket          = /var/lib/mysql/mysql.sock
skip-external-locking
key_buffer_size = 384M
max_allowed_packet = 1M
table_open_cache = 512
sort_buffer_size = 2M
read_buffer_size = 2M
```

```
read_rnd_buffer_size = 8M
myisam_sort_buffer_size = 64M
thread_cache_size = 8
query_cache_size = 32M
# Try number of CPU's*2 for thread_concurrency
thread_concurrency = 8

#skip-networking

# Replication Master Server (default)
# binary logging is required for replication
log-bin=mysql-bin

# required unique id between 1 and 2^32 - 1
# defaults to 1 if master-host is not set
# but will not function as a master if omitted
server-id       = 1

# Replication Slave (comment out master section to use this)
#
#
# required unique id between 2 and 2^32 - 1
# (and different from the master)
# defaults to 2 if master-host is set
# but will not function as a slave if omitted
#server-id       = 2
#
# The replication master for this slave - required
#master-host     =   <hostname>
#
# The username the slave will use for authentication when connecting
# to the master - required
#master-user     =   <username>
#
# The password the slave will authenticate with when connecting to
# the master - required
#master-password =   <password>
#
# The port the master is listening on.
# optional - defaults to 3306
#master-port     =  <port>
#
# binary logging - not required for slaves, but recommended
#log-bin=mysql-bin
#
# binary logging format - mixed recommended
#binlog_format=mixed

# Uncomment the following if you are using InnoDB tables
#innodb_data_home_dir = /var/lib/mysql
#innodb_data_file_path = ibdata1:2000M;ibdata2:10M:autoextend
```

```
#innodb_log_group_home_dir = /var/lib/mysql
# You can set .._buffer_pool_size up to 50 - 80 %
# of RAM but beware of setting memory usage too high
#innodb_buffer_pool_size = 384M
#innodb_additional_mem_pool_size = 20M
# Set .._log_file_size to 25 % of buffer pool size
#innodb_log_file_size = 100M
#innodb_log_buffer_size = 8M
#innodb_flush_log_at_trx_commit = 1
#innodb_lock_wait_timeout = 50

# Here follows entries for some specific programs
skip-name-resolve
default-storage-engine  = INNODB

character-set-server=utf8
collation_server=utf8_general_ci
init_connect='SET NAMES utf8'

max_connections                 = 4096
max_connect_errors              = 10

pid-file                                = mysql.pid
log                                     = mysql.log
log-error                               = mysql_error.log

log-slow-queries                = slow.log
long_query_time                 = 10

[mysqldump]
quick
max_allowed_packet = 16M

[mysql]
no-auto-rehash
# Remove the next comment character if you are not familiar with SQL
#safe-updates

[myisamchk]
key_buffer_size = 256M
sort_buffer_size = 256M
read_buffer = 2M
write_buffer = 2M

[mysqlhotcopy]
interactive-timeout
```

# 8. variables

```
show variables;
show global variables;
```

## 8.1. 查询多个变量

```
show variables where Variable_name like 'collation%' or
Variable_name like 'character_set_server%';
```

## 8.2. time_zone

```
SELECT @@global.time_zone, @@session.time_zone;
```

## 8.3. sql_mode

### 设置 sql_mode

```
SET GLOBAL sql_mode = 'NO_ENGINE_SUBSTITUTION';
SET SESSION sql_mode = 'NO_ENGINE_SUBSTITUTION';
```

### 查看 sql_mode

```
SELECT @@GLOBAL.sql_mode;
```

```
SELECT @@SESSION.sql_mode;
```

## 兼容早期 MySQL 版本

导入数据库遇到这样的问题

```
[root@netkiller]/tmp# cat cms.sql| mysql -uroot -p cms
```

ERROR 1067 (42000) at line 2194: Invalid default value for 'created_date'

将下面代码加入到 cms.sql 头部可以解决

```
set
@@global.sql_mode='NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION';
```

### 5.7.16

```
mysql> select version();
+-----------+
| version() |
+-----------+
| 5.7.16    |
+-----------+
1 row in set (0.00 sec)

mysql> SELECT @@global.sql_mode;
+-----------------------------------------------------------------
----------------------------------------------------------------
----------+
| @@global.sql_mode
|
+-----------------------------------------------------------------
----------------------------------------------------------------
----------+
```

```
|
ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DA
TE,ERROR_FOR_DIVISION_BY_ZERO,NO_AUTO_CREATE_USER,NO_ENGINE_SUBST
ITUTION |
+-----------------------------------------------------------------
-----------------------------------------------------------------
----------+
1 row in set (0.00 sec)
```

## 8.4. wait_timeout

```
mysql> show global variables like 'wait_timeout';
+---------------------------+-------+
| Variable_name             | Value |
+---------------------------+-------+
| wait_timeout              | 10    |
+---------------------------+-------+
```

```
mysql> use mysql;
Database changed

mysql> set wait_timeout=10;
Query OK, 0 rows affected (0.00 sec)

mysql> show variables  like '%wait_timeout%';
+--------------------------+-------+
| Variable_name            | Value |
+--------------------------+-------+
| innodb_lock_wait_timeout | 50    |
| table_lock_wait_timeout  | 50    |
| wait_timeout             | 10    |
+--------------------------+-------+
3 rows in set (0.00 sec)
```

## 8.5. table_lock_wait_timeout

```
mysql> set GLOBAL table_lock_wait_timeout=10;
Query OK, 0 rows affected (0.00 sec)

mysql> show variables  like '%table_lock_wait_timeout%';
+--------------------------+-------+
| Variable_name            | Value |
+--------------------------+-------+
| table_lock_wait_timeout | 10    |
+--------------------------+-------+
1 row in set (0.00 sec)
```

## 8.6. low_priority_updates

```
mysql> use mysql
Database changed

mysql> SET LOW_PRIORITY_UPDATES=1;
Query OK, 0 rows affected (0.00 sec)

mysql> show variables  like '%priority%';
+---------------------------+-------+
| Variable_name             | Value |
+---------------------------+-------+
| low_priority_updates      | ON    |
| sql_low_priority_updates | ON    |
+---------------------------+-------+
2 rows in set (0.00 sec)
```

## 8.7. collation_server

```
mysql> SHOW VARIABLES LIKE 'collation_server';
+------------------+-------------------+
| Variable_name    | Value             |
+------------------+-------------------+
```

```
| collation_server | latin1_swedish_ci |
+------------------+-------------------+
1 row in set (0.01 sec)

mysql>
```

## 8.8. character_set

```
mysql> show variables like 'character%';
+--------------------------+----------------------------+
| Variable_name            | Value                      |
+--------------------------+----------------------------+
| character_set_client     | utf8                       |
| character_set_connection | utf8                       |
| character_set_database   | utf8                       |
| character_set_filesystem | binary                     |
| character_set_results    | utf8                       |
| character_set_server     | utf8                       |
| character_set_system     | utf8                       |
| character_sets_dir       | /usr/share/mysql/charsets/ |
+--------------------------+----------------------------+
8 rows in set (0.00 sec)
```

链接 MySQL 指定字符集

```
mysql -uroot -h 192.168.0.10 -p --default-character-set=latin1
```

## 8.9. datadir

```
                    SHOW VARIABLES LIKE 'datadir';
```

```
mysql> SHOW VARIABLES LIKE 'datadir';
+---------------+-------------------------+
| Variable_name | Value                   |
+---------------+-------------------------+
| datadir       | /var/lib/mysql/         |
+---------------+-------------------------+
1 row in set (0.00 sec)
```

## 8.10. plugin_dir

### show variables like '%plugin_dir%';

```
mysql> show variables like '%plugin_dir%';
+---------------+-----------------------+
| Variable_name | Value                 |
+---------------+-----------------------+
| plugin_dir    | /usr/lib/mysql/plugin/ |
+---------------+-----------------------+
1 row in set (0.00 sec)
```

## 8.11. storage_engine

```
mysql> show variables like '%storage_engine%';
+------------------------+--------+
| Variable_name          | Value  |
+------------------------+--------+
| default_storage_engine | InnoDB |
| storage_engine         | InnoDB |
+------------------------+--------+
2 rows in set (0.00 sec)
```

## 8.12. timeout

```
show variables like "%timeout%";
```

## 8.13. max_connections

```
show variables like "max_connections";
```

```
set global max_connections = 200;
```

## 8.14. 自动提交 autocommit

```
select @@autocommit;

show variables like "autocommit";

set autocommit='off'
```

# 9. Monitoring

http://netkiller.sourceforge.net/monitoring/index.html

## 9.1. Analysis and Optimization

### mytop - top like query monitor for MySQL

```
sudo apt-get install mytop
```

```
mytop --host=172.16.0.7 --user=monitor --password=your_passwd
```

### mtop - MySQL terminal based query monitor

http://mtop.sourceforge.net/

```
sudo apt-get install mtop
mtop --host=172.16.0.6 --dbuser=monitor --password=your_passwd
```

mkill

```
mkill -sl 180 -fi 'select.*from bad_table' > /var/log/mkill.out
2> /var/log/mkill.kill
```

### innotop

### mysqlreport - A friendly report of important MySQL status values

```
# yum install mysqlreport -y
```

```
wget hackmysql.com/scripts/mysqlreport
```

```
[root@database ~]# mysqlreport --user root --password chen
Use of uninitialized value in multiplication (*) at
/usr/bin/mysqlreport line 829.
Use of uninitialized value in formline at /usr/bin/mysqlreport
line 1227.
MySQL 5.0.77-log         uptime 28 23:42:33     Sat Apr 10
18:15:44 2010

__ Key
_________________________________________________________________
__
Buffer used     6.54M of   8.00M  %Used:  81.75
  Current       1.49M            %Usage:  18.58
Write hit      97.65%
Read hit       99.81%

__ Questions
_____________________________________________________________
Total           2.22M     0.9/s
  DMS           1.91M     0.8/s  %Total:  86.16
  Com_        249.93k     0.1/s           11.25
  COM_QUIT     63.68k     0.0/s            2.87
  -Unknown      6.26k     0.0/s            0.28
Slow 10 s          52     0.0/s            0.00  %DMS:   0.00
Log: OFF
DMS             1.91M     0.8/s           86.16
  SELECT        1.17M     0.5/s           52.81         61.29
  INSERT      276.13k     0.1/s           12.43         14.43
  DELETE      264.78k     0.1/s           11.92         13.84
  UPDATE      158.14k     0.1/s            7.12          8.26
  REPLACE      41.74k     0.0/s            1.88          2.18
Com_          249.93k     0.1/s           11.25
  set_option   89.09k     0.0/s            4.01
  change_db    59.71k     0.0/s            2.69
  show_create  28.57k     0.0/s            1.29
```

```
__ SELECT and Sort
_____________________________________________________________
Scan          161.33k     0.1/s %SELECT:  13.76
Range           6.47k     0.0/s            0.55
Full join       1.56k     0.0/s            0.13
Range check         0       0/s            0.00
Full rng join       0       0/s            0.00
Sort scan       34.03k    0.0/s
Sort range      21.98k    0.0/s
Sort mrg pass     733     0.0/s

__ Table Locks
_____________________________________________________________
Waited             56     0.0/s  %Total:   0.00
Immediate       2.15M     0.9/s

__ Tables
_____________________________________________________________
Open               64 of   64    %Cache: 100.00
Opened         159.20k     0.1/s

__ Connections
_____________________________________________________________
Max used           36 of  200      %Max:  18.00
Total           63.75k     0.0/s

__ Created Temp
_____________________________________________________________
Disk table      32.80k     0.0/s
Table           63.69k     0.0/s    Size:  32.0M
File               319     0.0/s

__ Threads
_____________________________________________________________
Running              1 of    1
Cached               0 of    0      %Hit:      0
Created         63.75k     0.0/s
Slow                 0       0/s

__ Aborted
_____________________________________________________________
Clients            128     0.0/s
Connects           130     0.0/s
```

```
__ Bytes
_____________________________________________________________________________
Sent           23.89G    9.5k/s
Received        6.36G    2.5k/s

__ InnoDB Buffer Pool
_____________________________________________________________
Usage           8.00M of   8.00M  %Used: 100.00
Read hit       99.99%
Pages
  Free             0            %Total:   0.00
  Data           511                     99.80 %Drty:   0.00
  Misc             1                      0.20
  Latched                                 0.00
Reads           1.54M     0.6/s
  From file       135     0.0/s            0.01
  Ahead Rnd         4     0.0/s
  Ahead Sql         6     0.0/s
Writes        868.00k     0.3/s
Flushes         1.56k     0.0/s
Wait Free           0       0/s

__ InnoDB Lock
_____________________________________________________________________
Waits               0       0/s
Current             0
Time acquiring
  Total             0 ms
  Average           0 ms
  Max               0 ms

__ InnoDB Data, Pages, Rows
_____________________________________________
Data
  Reads           194     0.0/s
  Writes          628     0.0/s
  fsync           323     0.0/s
  Pending
    Reads           0
    Writes          0
    fsync           0

Pages
  Created         534     0.0/s
  Read            201     0.0/s
```

```
  Written       1.56k     0.0/s

Rows
  Deleted           0       0/s
  Inserted    423.82k     0.2/s
  Read          1.27M     0.5/s
  Updated           0       0/s
```

## mysqltuner - MySQL configuration assistant

```
# mysqltuner

 >>  MySQLTuner 1.1.1 - Major Hayden <major@mhtx.net>
 >>  Bug reports, feature requests, and downloads at
http://mysqltuner.com/
 >>  Run with '--help' for additional options and output
filtering
[!!] Successfully authenticated with no password - SECURITY
RISK!

-------- General Statistics --------------------------------
---------------
[--] Skipped version check for MySQLTuner script
[OK] Currently running supported MySQL version 5.1.69
[OK] Operating on 64-bit architecture

-------- Storage Engine Statistics -------------------------
---------------
[--] Status: -Archive -BDB -Federated +InnoDB -ISAM -NDBCluster
[!!] InnoDB is enabled but isn't being used
[OK] Total fragmented tables: 0

-------- Security Recommendations  -------------------------
---------------
[!!] User 'root' has no password set.
[!!] User 'root' has no password set.
[!!] User 'root' has no password set.
[!!] User '' has no password set.
[!!] User '' has no password set.
```

```
-------- Performance Metrics -------------------------------
--------------
[--] Up for: 18m 55s (42 q [0.037 qps], 7 conn, TX: 27K, RX:
1K)
[--] Reads / Writes: 100% / 0%
[--] Total buffers: 34.0M global + 2.7M per thread (151 max
threads)
[OK] Maximum possible memory usage: 449.2M (45% of installed
RAM)
[OK] Slow queries: 0% (0/42)
[OK] Highest usage of available connections: 0% (1/151)
[OK] Key buffer size / total MyISAM indexes: 8.0M/89.0K
[!!] Query cache is disabled
[OK] Temporary tables created on disk: 0% (0 on disk / 4 total)
[!!] Thread cache is disabled
[OK] Table cache hit rate: 76% (23 open / 30 opened)
[OK] Open file limit used: 4% (46/1K)
[OK] Table locks acquired immediately: 100% (19 immediate / 19
locks)

-------- Recommendations -----------------------------------
--------------
General recommendations:
    Add skip-innodb to MySQL configuration to disable InnoDB
    MySQL started within last 24 hours - recommendations may be
inaccurate
    Enable the slow query log to troubleshoot bad queries
    Set thread_cache_size to 4 as a starting value
Variables to adjust:
    query_cache_size (>= 8M)
    thread_cache_size (start at 4)
```

## 9.2. Cacti

## 9.3. Monitoring MySQL with SNMP

mysql-snmp - monitoring MySQL with SNMP

```
```

# 第 2 章 Client and Utility Programs

## 1. mysql - the MySQL command-line tool

### 1.1. ~/.my.cnf

```
# mysql_secure_installation config file
[mysql]

[mysqld]

[client]
user=root
password='chen'

[mysqldump]
quick

[mysqladmin]

[mysqlhotcopy]
```

### 1.2. 屏幕输出到文件

```


mysql>tee /home/neo/screen.txt
mysql>select * from member;
mysql>exit


```

### 1.3. 终端编码

```
mysql> show variables like 'char%';
```

```
+--------------------------+----------------------------+
| Variable_name            | Value                      |
+--------------------------+----------------------------+
| character_set_client     | utf8                       |
| character_set_connection | utf8                       |
| character_set_database   | utf8                       |
| character_set_filesystem | binary                     |
| character_set_results    | utf8                       |
| character_set_server     | utf8                       |
| character_set_system     | utf8                       |
| character_sets_dir       | /usr/share/mysql/charsets/ |
+--------------------------+----------------------------+
8 rows in set (0.00 sec)
```

设置终端编码 set names utf8;

```
mysql> select * from category;
+----+------+-------------+--------+-----------+-----------+
| id | name | description | status | parent_id | path      |
+----+------+-------------+--------+-----------+-----------+
|  1 | ??   | ???????     | Y      |      NULL | 1/        |
|  4 | ???  | ???         | Y      |         1 | 1/4       |
|  5 | ???  | NULL        | Y      |         4 | 1/4/5     |
|  6 | ???  | NULL        | Y      |         5 | 1/4/5/6   |
|  7 | ???  | NULL        | Y      |         6 | 1/4/5/6/7 |
+----+------+-------------+--------+-----------+-----------+
5 rows in set (0.00 sec)

mysql> set names utf8;
Query OK, 0 rows affected (0.00 sec)

mysql> select * from category;
+----+-----------+----------------------+--------+-----------
+-----------+
| id | name      | description          | status | parent_id |
path      |
+----+-----------+----------------------+--------+-----------
+-----------+
|  1 | 中国    | 中华人民共和家
| Y      |      NULL | 1/        |
|  4 | 广东省 | 广东省
| Y      |         1 | 1/4       |
|  5 | 深圳市 | NULL                       | Y      |         4 |
```

```
1/4/5     |
|  6 | 宝安区 | NULL                          | Y       |          5 |
1/4/5/6   |
|  7 | 龙华镇 | NULL                          | Y       |          6 |
1/4/5/6/7 |
+----+-----------+-----------------------+--------+-----------
+-----------+
5 rows in set (0.00 sec)
```

## 1.4. Unix Socket

```
mysql -uroot -p -S /tmp/mysql.sock
```

## 1.5. 重定向巧用

```
echo "show databases;" | mysql -uroot -pneo

cat |mysql -uroot -pneo << EOF
show databases;
EOF
```

## 1.6. --sigint-ignore 忽略 Ctrl + C

使用该选项方式用户中途通过 Ctrl + C 推出,只能通过 quit 退出

```
$ mysql --sigint-ignore
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 71
Server version: 5.5.32-0ubuntu0.13.04.1 (Ubuntu)

Copyright (c) 2000, 2013, Oracle and/or its affiliates. All
rights reserved.
```

```
Oracle is a registered trademark of Oracle Corporation and/or
its
affiliates. Other names may be trademarks of their respective
owners.

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

mysql> quit
Bye
```

## 1.7. mysql log

/etc/my.cnf 配置文件

```
在服务器上的/etc/my.cnf中的[client]加入
tee =/tmp/mysql_history.log即可.
```

查看log设置

```
show VARIABLES like '%log%';
```

命令行

```
mysql -uroot --tee=/tmp/mysql_history.log
```

```
mysql> tee mysql_history.log
Logging to file 'mysql_history.log '
```

```
或者
mysql> \T mysql_history.log
Logging to file 'mysql_history.log '
```

```
mysql> notee
Outfile disabled.
或者
mysql> \t
Outfile disabled.
```

# 2. mysqldump - a database backup program

## 2.1. 备份数据库并压缩文件

```
mysqldump -uroot -p dbname | gzip > dbname.backup
```

## 2.2. 备份数据库/表

```
mysqldump -uroot -p -d database
mysqldump -uroot -p -d database table
```

## 2.3. 备份到文件

-r, --result-file=name 输出文件

```
mysqldump --default-character-set=utf8 -r mysql.sql -h localhost -u root
-p yourdb
```

## 2.4. 备份数据库，无结构，只有数据

-t, --no-create-info Don't write table creation info.

```
mysqldump -uroot -p -t -d database
```

## 2.5. 使用完整的insert插入数据

-c, --complete-insert Use complete insert statements.

```
$ mysqldump -hlocalhost -uroot -t neo test

INSERT INTO `test` VALUES (98,'neo','chen'),(112,'jam','zheng'),
(113,'john','meng');

$ mysqldump -hlocalhost -uroot -c -t neo test
INSERT INTO `test` (`userid`, `username`, `password`) VALUES
(98,'neo','chen'),(112,'jam','zheng'),(113,'john','meng');
```

## 2.6. --extended-insert / --skip-extended-insert

--extended-insert 默认开启

```
INSERT INTO `test` VALUES (98,'neo','chen'),(112,'jam','zheng'),
(113,'john','meng');
```

每条记录使用一次insert

```
$ mysqldump -hlocalhost -uroot --skip-extended-insert -t neo test |more
INSERT INTO `test` VALUES (98,'neo','chen');
INSERT INTO `test` VALUES (111,'neo','chen');
INSERT INTO `test` VALUES (112,'jam','zheng');
INSERT INTO `test` VALUES (113,'john','meng');
```

## 2.7. --skip-lock-tables

mysqldump 时禁止锁表

使用 --skip-lock-tables 参数，不会影响正在备份的表SELECT操作。

## 2.8. --skip-add-locks

该参数mysqldump输出中包含下面

默认情况

```
LOCK TABLES `tbl_name` WRITE;
```

如果使用这个参数就不会输出 LOCK TABLE

## 2.9. --where

```
mysqldump -hlocalhost -umysql -ppasswd database table --where="id>128"
```

## 2.10. 注释信息--comments /--skip-comments

--comments附加注释信息，默认为打开。可以用--skip-comments取消

```
--
-- Table structure for table `demo`
--

DROP TABLE IF EXISTS `demo`;

...
...

-- Dump completed on 2014-02-13 13:31:05
```

使用 --skip-comments后

```
DROP TABLE IF EXISTS `demo`;
...
...
```

## 2.11. 不导出注释信息

--compact

```
mysqldump --default-character-set=utf8  -h localhost -u root -p -t
yourdb yourtable --skip-extended-insert --compact
```

运行后导出纯净的 INSERT 语句

## 2.12. 字符集设置

```
mysqldump --default-character-set=utf8 -r mysql.sql -h 192.168.0.24 -u
root -p yourdb
```

# 3. mysqladmin - client for administering a MySQL server

## 3.1. reload

```
mysqladmin --user=root --password reload
```

## 3.2. 更改密码

```
sudo mysqladmin -u root -p password '你的新密码'
```

## 3.3. status

每个10秒输出一次mysql的状态信息

```
mysqladmin -i 10 extended status
```

```
mysqladmin -h 172.16.0.1 -u monitor -ppasswd status
Uptime: 195824  Threads: 21  Questions: 57744081  Slow queries:
516230  Opens: 13202607  Flush tables: 1  Open tables: 160
Queries per second avg: 294.877
```

```
 mysqladmin -h 172.16.0.1 -u monitor -ppasswd  extended-status
+-----------------------------------+-------------+
| Variable_name                     | Value       |
+-----------------------------------+-------------+
| Aborted_clients                   | 60336       |
| Aborted_connects                  | 4599        |
| Binlog_cache_disk_use             | 36          |
| Binlog_cache_use                  | 100721      |
```

| Bytes_received | 17510873261 | |
|---|---|---|
| Bytes_sent | 92890743568 | |
| Com_admin_commands | 10026660 | |
| Com_assign_to_keycache | 0 | |
| Com_alter_db | 0 | |
| Com_alter_db_upgrade | 0 | |
| Com_alter_event | 0 | |
| Com_alter_function | 0 | |
| Com_alter_procedure | 0 | |
| Com_alter_server | 0 | |
| Com_alter_table | 418 | |
| Com_alter_tablespace | 0 | |
| Com_analyze | 0 | |
| Com_backup_table | 0 | |
| Com_begin | 0 | |
| Com_binlog | 0 | |
| Com_call_procedure | 0 | |
| Com_change_db | 4440400 | |
| Com_change_master | 1 | |
| Com_check | 0 | |
| Com_checksum | 0 | |
| Com_commit | 30089 | |
| Com_create_db | 1 | |
| Com_create_event | 0 | |
| Com_create_function | 0 | |
| Com_create_index | 1 | |
| Com_create_procedure | 0 | |
| Com_create_server | 0 | |
| Com_create_table | 211 | |
| Com_create_trigger | 4 | |
| Com_create_udf | 0 | |
| Com_create_user | 0 | |
| Com_create_view | 2 | |
| Com_dealloc_sql | 0 | |
| Com_delete | 36664 | |
| Com_delete_multi | 0 | |
| Com_do | 0 | |
| Com_drop_db | 0 | |
| Com_drop_event | 0 | |
| Com_drop_function | 0 | |
| Com_drop_index | 0 | |
| Com_drop_procedure | 0 | |
| Com_drop_server | 0 | |
| Com_drop_table | 213 | |
| Com_drop_trigger | 0 | |

| Com_drop_user | 0 | |
| --- | --- | --- |
| Com_drop_view | 4 | |
| Com_empty_query | 0 | |
| Com_execute_sql | 0 | |
| Com_flush | 9 | |
| Com_grant | 6 | |
| Com_ha_close | 0 | |
| Com_ha_open | 0 | |
| Com_ha_read | 0 | |
| Com_help | 0 | |
| Com_insert | 472260 | |
| Com_insert_select | 0 | |
| Com_install_plugin | 0 | |
| Com_kill | 12 | |
| Com_load | 0 | |
| Com_load_master_data | 0 | |
| Com_load_master_table | 0 | |
| Com_lock_tables | 209 | |
| Com_optimize | 0 | |
| Com_preload_keys | 0 | |
| Com_prepare_sql | 0 | |
| Com_purge | 0 | |
| Com_purge_before_date | 0 | |
| Com_release_savepoint | 0 | |
| Com_rename_table | 0 | |
| Com_rename_user | 0 | |
| Com_repair | 0 | |
| Com_replace | 4612 | |
| Com_replace_select | 0 | |
| Com_reset | 0 | |
| Com_restore_table | 0 | |
| Com_revoke | 0 | |
| Com_revoke_all | 0 | |
| Com_rollback | 0 | |
| Com_rollback_to_savepoint | 0 | |
| Com_savepoint | 0 | |
| Com_select | 20310686 | |
| Com_set_option | 9089818 | |
| Com_show_authors | 0 | |
| Com_show_binlog_events | 0 | |
| Com_show_binlogs | 0 | |
| Com_show_charsets | 24 | |
| Com_show_collations | 18214 | |
| Com_show_column_types | 0 | |
| Com_show_contributors | 0 | |

| Com_show_create_db | 0 | |
| --- | --- | --- |
| Com_show_create_event | 0 | |
| Com_show_create_func | 0 | |
| Com_show_create_proc | 0 | |
| Com_show_create_table | 0 | |
| Com_show_create_trigger | 0 | |
| Com_show_databases | 24 | |
| Com_show_engine_logs | 0 | |
| Com_show_engine_mutex | 0 | |
| Com_show_engine_status | 0 | |
| Com_show_events | 0 | |
| Com_show_errors | 0 | |
| Com_show_fields | 147160 | |
| Com_show_function_status | 3 | |
| Com_show_grants | 0 | |
| Com_show_keys | 2 | |
| Com_show_master_status | 1 | |
| Com_show_new_master | 0 | |
| Com_show_open_tables | 0 | |
| Com_show_plugins | 0 | |
| Com_show_privileges | 0 | |
| Com_show_procedure_status | 3 | |
| Com_show_processlist | 12483 | |
| Com_show_profile | 0 | |
| Com_show_profiles | 0 | |
| Com_show_slave_hosts | 0 | |
| Com_show_slave_status | 0 | |
| Com_show_status | 1158 | |
| Com_show_storage_engines | 0 | |
| Com_show_table_status | 2 | |
| Com_show_tables | 29915 | |
| Com_show_triggers | 0 | |
| Com_show_variables | 26295 | |
| Com_show_warnings | 0 | |
| Com_slave_start | 0 | |
| Com_slave_stop | 0 | |
| Com_stmt_close | 0 | |
| Com_stmt_execute | 0 | |
| Com_stmt_fetch | 0 | |
| Com_stmt_prepare | 0 | |
| Com_stmt_reprepare | 0 | |
| Com_stmt_reset | 0 | |
| Com_stmt_send_long_data | 0 | |
| Com_truncate | 0 | |
| Com_uninstall_plugin | 0 | |

| Com_unlock_tables | 209 |
| --- | --- |
| Com_update | 501411 |
| Com_update_multi | 23112 |
| Com_xa_commit | 0 |
| Com_xa_end | 0 |
| Com_xa_prepare | 0 |
| Com_xa_recover | 0 |
| Com_xa_rollback | 0 |
| Com_xa_start | 0 |
| Compression | OFF |
| Connections | 4555052 |
| Created_tmp_disk_tables | 421231 |
| Created_tmp_files | 1172 |
| Created_tmp_tables | 2769149 |
| Delayed_errors | 0 |
| Delayed_insert_threads | 0 |
| Delayed_writes | 0 |
| Flush_commands | 1 |
| Handler_commit | 100721 |
| Handler_delete | 133583 |
| Handler_discover | 0 |
| Handler_prepare | 0 |
| Handler_read_first | 404032 |
| Handler_read_key | 18292439681 |
| Handler_read_next | 33393351305 |
| Handler_read_prev | 77792315 |
| Handler_read_rnd | 2969739139 |
| Handler_read_rnd_next | 41965058450 |
| Handler_rollback | 0 |
| Handler_savepoint | 0 |
| Handler_savepoint_rollback | 0 |
| Handler_update | 4595750766 |
| Handler_write | 6069006380 |
| Innodb_buffer_pool_pages_data | 19 |
| Innodb_buffer_pool_pages_dirty | 0 |
| Innodb_buffer_pool_pages_flushed | 0 |
| Innodb_buffer_pool_pages_free | 493 |
| Innodb_buffer_pool_pages_misc | 0 |
| Innodb_buffer_pool_pages_total | 512 |
| Innodb_buffer_pool_read_ahead_rnd | 1 |
| Innodb_buffer_pool_read_ahead_seq | 0 |
| Innodb_buffer_pool_read_requests | 77 |
| Innodb_buffer_pool_reads | 12 |
| Innodb_buffer_pool_wait_free | 0 |
| Innodb_buffer_pool_write_requests | 0 |

| Innodb_data_fsyncs | 3 | |
| --- | --- | --- |
| Innodb_data_pending_fsyncs | 0 | |
| Innodb_data_pending_reads | 0 | |
| Innodb_data_pending_writes | 0 | |
| Innodb_data_read | 2494464 | |
| Innodb_data_reads | 25 | |
| Innodb_data_writes | 3 | |
| Innodb_data_written | 1536 | |
| Innodb_dblwr_pages_written | 0 | |
| Innodb_dblwr_writes | 0 | |
| Innodb_log_waits | 0 | |
| Innodb_log_write_requests | 0 | |
| Innodb_log_writes | 1 | |
| Innodb_os_log_fsyncs | 3 | |
| Innodb_os_log_pending_fsyncs | 0 | |
| Innodb_os_log_pending_writes | 0 | |
| Innodb_os_log_written | 512 | |
| Innodb_page_size | 16384 | |
| Innodb_pages_created | 0 | |
| Innodb_pages_read | 19 | |
| Innodb_pages_written | 0 | |
| Innodb_row_lock_current_waits | 0 | |
| Innodb_row_lock_time | 0 | |
| Innodb_row_lock_time_avg | 0 | |
| Innodb_row_lock_time_max | 0 | |
| Innodb_row_lock_waits | 0 | |
| Innodb_rows_deleted | 0 | |
| Innodb_rows_inserted | 0 | |
| Innodb_rows_read | 0 | |
| Innodb_rows_updated | 0 | |
| Key_blocks_not_flushed | 0 | |
| Key_blocks_unused | 6917 | |
| Key_blocks_used | 53585 | |
| Key_read_requests | 35870213140 | |
| Key_reads | 13788784 | |
| Key_write_requests | 35265303 | |
| Key_writes | 2467239 | |
| Last_query_cost | 0.000000 | |
| Max_used_connections | 3001 | |
| Not_flushed_delayed_rows | 0 | |
| Open_files | 238 | |
| Open_streams | 0 | |
| Open_table_definitions | 228 | |
| Open_tables | 160 | |
| Opened_files | 20864567 | |

| Opened_table_definitions | 653 |
| --- | --- |
| Opened_tables | 13202613 |
| Prepared_stmt_count | 0 |
| Qcache_free_blocks | 10480 |
| Qcache_free_memory | 38697120 |
| Qcache_hits | 17943956 |
| Qcache_inserts | 8251298 |
| Qcache_lowmem_prunes | 560647 |
| Qcache_not_cached | 11879434 |
| Qcache_queries_in_cache | 54611 |
| Qcache_total_blocks | 125193 |
| Queries | 57755205 |
| Questions | 57582352 |
| Rpl_status | NULL |
| Select_full_join | 602236 |
| Select_full_range_join | 6851 |
| Select_range | 1633467 |
| Select_range_check | 0 |
| Select_scan | 10981650 |
| Slave_open_temp_tables | 0 |
| Slave_retried_transactions | 0 |
| Slave_running | OFF |
| Slow_launch_threads | 206 |
| Slow_queries | 516237 |
| Sort_merge_passes | 548 |
| Sort_range | 293328 |
| Sort_rows | 2831414035 |
| Sort_scan | 2726547 |
| Ssl_accept_renegotiates | 0 |
| Ssl_accepts | 0 |
| Ssl_callback_cache_hits | 0 |
| Ssl_cipher | |
| Ssl_cipher_list | |
| Ssl_client_connects | 0 |
| Ssl_connect_renegotiates | 0 |
| Ssl_ctx_verify_depth | 0 |
| Ssl_ctx_verify_mode | 0 |
| Ssl_default_timeout | 0 |
| Ssl_finished_accepts | 0 |
| Ssl_finished_connects | 0 |
| Ssl_session_cache_hits | 0 |
| Ssl_session_cache_misses | 0 |
| Ssl_session_cache_mode | NONE |
| Ssl_session_cache_overflows | 0 |
| Ssl_session_cache_size | 0 |

```
| Ssl_session_cache_timeouts         | 0           |
| Ssl_sessions_reused                | 0           |
| Ssl_used_session_cache_entries     | 0           |
| Ssl_verify_depth                   | 0           |
| Ssl_verify_mode                    | 0           |
| Ssl_version                        |             |
| Table_locks_immediate              | 46406490    |
| Table_locks_waited                 | 1428430     |
| Tc_log_max_pages_used              | 0           |
| Tc_log_page_size                   | 0           |
| Tc_log_page_waits                  | 0           |
| Threads_cached                     | 33          |
| Threads_connected                  | 33          |
| Threads_created                    | 77809       |
| Threads_running                    | 7           |
| Uptime                             | 195854      |
| Uptime_since_flush_status          | 195854      |
+------------------------------------+-------------+
```

## 3.4. process list

```
[root@development ~]# mysqladmin -u root -p -h 127.0.0.1
processlist
Enter password:
+-------+--------+---------------------+--------+---------+---
---+-------+------------------+
| Id    | User   | Host                | db     | Command |
Time | State | Info             |
+-------+--------+---------------------+--------+---------+---
---+-------+------------------+
| 23648 | dbuser | 192.168.3.237:1220  | testdb | Sleep   |
2733 |       |                  |
| 23878 | dbuser | www.testdb.com:53639 | testdb | Sleep   | 7
|       |                  |
| 23881 | root   | localhost:57243     |        | Query   | 0
|       | show processlist |
+-------+--------+---------------------+--------+---------+---
---+-------+------------------+
```

```
mysql -u root -pneo -S /tmp/mysql.sock -e "show full
processlist;"|grep -v Sleep
```

# 4. myisamchk — MyISAM Table-Maintenance Utility

先停止mysqld，在--datadir目录运行

```
myisamchk */*.MYI >/dev/null  # 检查哪些表需要修复
```

修复用以下命令一个个修复:

```
myisamchk -r table.MYI
```

更省事的做法:

```
myisamchk -r /var/lib/mysql/*.MYI
```

myisamchk可用crontab定時最佳化table

```
0 * * 0 /usr/bin/myisamchk -s /var/lib/mysql/*/*.MYI
```

# 5. mysqlcheck — A Table Maintenance and Repair Program

即可最佳化所有db

```
mysqlcheck -a -c -o -r --all-databases  -uroot -p
-a = Analyse given tables.
-c = Check table for errors
-o = Optimise table
-r = Can fix almost anything except unique keys that aren't
unique
```

```
mysqlcheck -A -o -r -p
```

# 6. mysqlslap - load emulation client

```
—auto-generate-sql, -a
自动生成测试表和数据

—auto-generate-sql-load-type=type
测试语句的类型。取值包括：read，key，write，update和mixed(默认)。

—number-char-cols=N, -x N
自动生成的测试表中包含多少个字符类型的列，默认1

—number-int-cols=N, -y N
自动生成的测试表中包含多少个数字类型的列，默认1

—number-of-queries=N
总的测试查询次数(并发客户数×每客户查询次数)

—query=name,-q
使用自定义脚本执行测试，例如可以调用自定义的一个存储过程或者sql语句来执行测试。

—create-schema
测试的schema，MySQL中schema也就是database

—commint=N
多少条DML后提交一次

—compress, -C
如果服务器和客户端支持都压缩，则压缩信息传递

—concurrency=N, -c N
并发量，也就是模拟多少个客户端同时执行select。可指定多个值，以逗号或者—delimiter参数指定的值做为分隔符

—engine=engine_name, -e engine_name
创建测试表所使用的存储引擎，可指定多个

—iterations=N, -i N
测试执行的迭代次数

—detach=N
执行N条语句后断开重连
```

```
—debug-info, -T
打印内存和CPU的信息

—only-print
只打印测试语句而不实际执行
```

```
mysqlslap -u root -p -h localhost -c 10,50,100,200 -i 1 \
--engine=myisam --auto-generate-sql-load-type=mixed --number-
of-queries=50000 \
--number-char-cols=5 --number-int-cols=5 --auto-generate-sql
```

```
mysqlslap --defaults-file=/etc/my.cnf --concurrency=50,100,200
--iterations=1 \
--number-int-cols=4 --number-char-cols=4 --auto-generate-sql --
auto-generate-sql-add-autoincrement \
--auto-generate-sql-load-type=mixed --engine=myisam,innodb --
number-of-queries=200 --debug-info \
-uroot -p -S/tmp/mysql.sock
```

# 7. mysqldumpslow - Parse and summarize the MySQL slow query log.

开启记录日志，修改my.cnf加入下面几行

--log-slow-queries[=file_name]

```
long_query_time = 10
log-slow-queries =
```

long_query_time 是指执行超过10秒的sql会被记录下来。

log-slow-queries设置把日志文件的位置，如果没有给出文件名值， 默认未主机名，后缀为-slow.log。如果给出了文件名，但不是绝对路径名，文件则写入数据目录。

cat /etc/mysql/my.cnf

```
[mysqld]
set-variable=long_query_time=1
log-slow-queries=/var/log/mysql/log-slow-queries.log


You must create the file manually and change owners this way:

touch /var/log/mysql/log-slow-queries.log
chown mysql.mysql -R /var/log/mysql/log-slow-queries.log
```

```
$ mysqldumpslow /var/log/mysql/log-slow-queries.log
```

mysqldumpslow 参数

1. -s，是order的顺序，说明写的不够详细，俺用下来，包括看了代码，主要有c,t,l,r和ac,at,al,ar，t=time, l=lock time, r=rows分别是按照query次数，时间，lock的时间和返回的记录数来排序，前面加了a的时倒叙

2. -t，是top n的意思，即为返回前面多少条的数据

3. -g，后边可以写一个正则匹配模式，大小写不敏感的

4. -g，后边可以写一个正则匹配模式，大小写不敏感的

```
mysqldumpslow -s c -t 20 ubuntu-slow.log
```

```
mysqldumpslow -s r -t 20 ubuntu-slow.log
```

# 8. mysql-shell

```
dnf install https://cdn.mysql.com/archives/mysql-shell/mysql-shell-8.0.26-1.el8.x86_64.rpm
```

# 9.

https://www.mycli.net

```
$ brew update && brew install mycli
```

# 10. MySQL慢查询日志（Slow Query Log）

参数说明如下：

```
slow_query_log：                    慢查询开启状态
slow_query_log_file：     慢查询日志存放的位置（默认设置为 MySQL 的数据存放目录）
long_query_time：                   查询超过多少秒才记录
```

## 10.1. MySQL 8.x

慢查询日志状态

```
mysql> show variables like '%slow_query_log%';
+---------------------+-----------------------------------+
| Variable_name       | Value                             |
+---------------------+-----------------------------------+
| slow_query_log      | ON                                |
| slow_query_log_file | /var/lib/mysql/netkiller-slow.log |
+---------------------+-----------------------------------+
2 rows in set (0.00 sec)
```

```
set global log_output='TABLE';  -- 开启慢日志,纪录到 mysql.slow_log 表
set global long_query_time=10;  -- 设置超过10秒的查询为慢查询
set global slow_query_log='ON'; -- 打开慢日志记录
```

查询慢的 sql 日志

```
select convert(sql_text using utf8) sql_text from mysql.slow_log
```

关闭慢查询日志

```
set global slow_query_log='OFF'; -- 排查完毕后记得关闭日志
```

## 10.2. MySQL 5.x

### 查看设置

```
mysql> SHOW VARIABLES LIKE 'slow_query%';
+---------------------+-----------------------------------+
| Variable_name       | Value                             |
+---------------------+-----------------------------------+
| slow_query_log      | OFF                               |
| slow_query_log_file | /var/lib/mysql/netkiller-slow.log |
+---------------------+-----------------------------------+
2 rows in set (0.00 sec)

mysql>
```

```
mysql> SHOW VARIABLES LIKE 'long_query_time';
+-----------------+-----------+
| Variable_name   | Value     |
+-----------------+-----------+
| long_query_time | 10.000000 |
+-----------------+-----------+
1 row in set (0.01 sec)
```

```

```

## 开启慢查询日志

```
mysql> SET GLOBAL slow_query_log=ON;
Query OK, 0 rows affected (0.05 sec)

mysql> SET GLOBAL long_query_time=0.001;
Query OK, 0 rows affected (0.00 sec)
```

## 配置慢查询日志

```
[mysqld]
log-slow-queries                = /var/lib/mysql/netkiller-
slow.log
long_query_time                 = 10
```

## 删除慢查询日志

执行命令后将清空旧慢查询日志，写入新日志

```
mysqladmin -uroot -p flush-logs
```

# 11. mysql-admin

```
$ sudo apt-get install mysql-admin
```

运行mysql-admin

```
/usr/bin/mysql-admin
```

运行 mysql-query-browser

```
mysql-query-browser --query="SELECT * FROM users"
```

# 12. MySQL Workbench 数据库恢复

mysql 8.0 不再提供数据库重命名服务，无法修改数据库名，只能导出再倒入，使用 MySQL Workbench 恢复数据时也无法改变数据库名。

我们只需要下面两条命令，修改dumps数据，就可以恢复到指定的数据库下

```
rename "s/old_/new_/" old_*
sed -i "s/Database: old/Database: new/" *
```

# 第 3 章 数据库管理（Database Administration)

## 1. 用户管理（User Account Management）

### 1.1. 创建用户

```
CREATE USER user [IDENTIFIED BY [PASSWORD] 'password']
    [, user [IDENTIFIED BY [PASSWORD] 'password']] ...
```

```
CREATE USER 'test'@'xxx.xxx.xxx.xxx' IDENTIFIED BY  'your_password';
```

```
CREATE USER 'root'@'192.168.1.%' IDENTIFIED BY 'password';
```

add a new user by grant

```
GRANT ALL PRIVILEGES ON opencart.* TO 'neo'@'localhost' IDENTIFIED BY
'chen' WITH GRANT OPTION;

GRANT ALL PRIVILEGES ON *.* TO 'neo'@'localhost' IDENTIFIED BY 'chen'
WITH GRANT OPTION;

FLUSH PRIVILEGES;
```

MySQL 8.0

```
mysql> CREATE USER 'root'@'%' IDENTIFIED WITH mysql_native_password BY
'pMQiEge1ikst7S_6tlXzBOmt_4b';
```

```
Query OK, 0 rows affected (0.06 sec)

mysql> grant all on *.* to 'root'@'%';
Query OK, 0 rows affected (0.11 sec)
```

## 1.2. 删除用户

```
DROP USER user [, user] ...
```

```
mysql> drop user 'root'@'%';
Query OK, 0 rows affected (0.02 sec)

mysql> drop user admin@'localhost';
Query OK, 0 rows affected (0.00 sec)

mysql> drop user admin@'127.0.0.1';
Query OK, 0 rows affected (0.00 sec)
```

判断用户是否存，存在再删除

```
DROP USER IF EXISTS 'nacos'@'localhost';
```

## 1.3. 修改用户名

```
RENAME USER old_user TO new_user [, old_user TO new_user] ...
```

## 1.4. 修改密码

mysql 5.7 之前的版本

```
SET PASSWORD FOR 'bob'@'%.loc.gov' = PASSWORD('newpass');

SET PASSWORD FOR 'root'@'%' =
PASSWORD('co2uqAMAho1aSOS62146Xoci6ogu4I');
```

MySQL 5.7 之后

```
ALTER USER 'root'@'localhost' IDENTIFIED BY 'your_password';
```

```
mysql> ALTER user 'root'@'%' IDENTIFIED BY 'test';
Query OK, 0 rows affected (0.01 sec)

mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.00 sec)
```

# 2. Access Privilege System

```
global privileges
OR (database privileges AND host privileges)
OR table privileges
OR column privileges
OR routine privileges
```

```
Table 12.1. Permissible Privileges for GRANT and REVOKE

Privilege       Meaning
ALL [PRIVILEGES]        Grant all privileges at specified access level except
GRANT OPTION
ALTER   Enable use of ALTER TABLE
ALTER ROUTINE   Enable stored routines to be altered or dropped
CREATE  Enable database and table creation
CREATE ROUTINE  Enable stored routine creation
CREATE TABLESPACE       Enable tablespaces and log file groups to be created,
altered, or dropped
CREATE TEMPORARY TABLES Enable use of CREATE TEMPORARY TABLE
CREATE USER     Enable use of CREATE USER, DROP USER, RENAME USER, and REVOKE
ALL PRIVILEGES
CREATE VIEW     Enable views to be created or altered
DELETE  Enable use of DELETE
DROP    Enable databases, tables, and views to be dropped
EVENT   Enable use of events for the Event Scheduler
EXECUTE Enable the user to execute stored routines
FILE    Enable the user to cause the server to read or write files
GRANT OPTION    Enable privileges to be granted to or removed from other
accounts
INDEX   Enable indexes to be created or dropped
INSERT  Enable use of INSERT
LOCK TABLES     Enable use of LOCK TABLES on tables for which you have the
SELECT privilege
PROCESS Enable the user to see all processes with SHOW PROCESSLIST
PROXY   Enable user proxying
REFERENCES      Not implemented
RELOAD  Enable use of FLUSH operations
REPLICATION CLIENT      Enable the user to ask where master or slave servers are
REPLICATION SLAVE       Enable replication slaves to read binary log events from
the master
SELECT  Enable use of SELECT
SHOW DATABASES  Enable SHOW DATABASES to show all databases
SHOW VIEW       Enable use of SHOW CREATE VIEW
SHUTDOWN        Enable use of mysqladmin shutdown
SUPER   Enable use of other administrative operations such as CHANGE MASTER TO,
KILL, PURGE BINARY LOGS, SET GLOBAL, and mysqladmin debug command
TRIGGER Enable trigger operations
UPDATE  Enable use of UPDATE
```

```
USAGE   Synonym for “no privileges”
```

http://dev.mysql.com/doc/refman/5.5/en/grant.html#grant-table-privileges

REPLICATION CLIENT 与 REPLICATION SLAVE区别，前者只能使用SHOW MASTER STATUS和SHOW SLAVE STATUS命令监控复制状态，后者才能从主库复制binlog.

## 2.1. SHOW GRANTS

```
SHOW GRANTS FOR 'root'@'%';
```

```
mysql> SHOW GRANTS FOR 'root'@'localhost';
+---------------------------------------------------------------------+
| Grants for root@localhost                                           |
+---------------------------------------------------------------------+
| GRANT ALL PRIVILEGES ON *.* TO 'root'@'localhost' WITH GRANT OPTION |
+---------------------------------------------------------------------+
1 row in set (0.00 sec)
```

```
mysql> show grants;
+-------------------------------------------------------------------------------
--+
| Grants for root@localhost                                                     |
+-------------------------------------------------------------------------------
--+
| GRANT ALL PRIVILEGES ON *.* TO 'root'@'localhost' WITH GRANT OPTION |
+-------------------------------------------------------------------------------
--+
1 row in set (0.00 sec)
```

## 2.2. show privileges

```
mysql> show privileges;
+-------------------------+-----------------------------------------+-------------
----------------------------------------+
| Privilege               | Context                                 | Comment
|
+-------------------------+-----------------------------------------+-------------
```

```
----------------------------------------+
| Alter                  | Tables                                | To alter the
table                                  |
| Alter routine          | Functions,Procedures                  | To alter or
drop stored functions/procedures        |
| Create                 | Databases,Tables,Indexes              | To create
new databases and tables                |
| Create routine         | Databases                             | To use
CREATE FUNCTION/PROCEDURE                |
| Create temporary tables | Databases                            | To use
CREATE TEMPORARY TABLE                   |
| Create view            | Tables                                | To create
new views                               |
| Create user            | Server Admin                          | To create
new users                               |
| Delete                 | Tables                                | To delete
existing rows                           |
| Drop                   | Databases,Tables                      | To drop
databases, tables, and views             |
| Event                  | Server Admin                          | To create,
alter, drop and execute events          |
| Execute                | Functions,Procedures                  | To execute
stored routines                         |
| File                   | File access on server                 | To read and
write files on the server               |
| Grant option           | Databases,Tables,Functions,Procedures | To give to
other users those privileges you possess |
| Index                  | Tables                                | To create or
drop indexes                            |
| Insert                 | Tables                                | To insert
data into tables                        |
| Lock tables            | Databases                             | To use LOCK
TABLES (together with SELECT privilege) |
| Process                | Server Admin                          | To view the
plain text of currently executing queries |
| Proxy                  | Server Admin                          | To make
proxy user possible                     |
| References             | Databases,Tables                      | To have
references on tables                    |
| Reload                 | Server Admin                          | To reload or
refresh tables, logs and privileges     |
| Replication client     | Server Admin                          | To ask where
the slave or master servers are         |
| Replication slave      | Server Admin                          | To read
binary log events from the master       |
| Select                 | Tables                                | To retrieve
rows from table                         |
| Show databases         | Server Admin                          | To see all
databases with SHOW DATABASES           |
| Show view              | Tables                                | To see views
with SHOW CREATE VIEW                   |
| Shutdown               | Server Admin                          | To shut down
the server                              |
| Super                  | Server Admin                          | To use KILL
thread, SET GLOBAL, CHANGE MASTER, etc. |
| Trigger                | Tables                                | To use
triggers                                |
```

```
| Create tablespace       | Server Admin                            | To
create/alter/drop tablespaces                       |
| Update                  | Tables                                  | To update
existing rows                                |
| Usage                   | Server Admin                            | No
privileges - allow connect only                     |
+-------------------------+-----------------------------------------+-------------
-----------------------------------------+
31 rows in set (0.00 sec)
```

## 2.3. Grant privileges

Global privileges

```
GRANT ALL ON *.* TO 'someuser'@'somehost';
GRANT SELECT, INSERT ON *.* TO 'someuser'@'somehost';
```

Database privileges

```
GRANT ALL ON mydb.* TO 'someuser'@'somehost';
GRANT SELECT, INSERT ON mydb.* TO 'someuser'@'somehost';
```

Table privileges

```
GRANT ALL ON mydb.mytbl TO 'someuser'@'somehost';
GRANT SELECT, INSERT ON mydb.mytbl TO 'someuser'@'somehost';
```

Column privileges

```
GRANT SELECT (col1), INSERT (col1,col2) ON mydb.mytbl TO 'someuser'@'somehost';
```

Routine privileges

```
GRANT CREATE ROUTINE ON mydb.* TO 'someuser'@'somehost';
GRANT EXECUTE ON PROCEDURE mydb.myproc TO 'someuser'@'somehost';
```

## 2.4. Revoke privileges

```
REVOKE
    priv_type [(column_list)]
      [, priv_type [(column_list)]] ...
    ON [object_type] priv_level
    FROM user [, user] ...

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

## 2.5. Show Privileges

```
mysql> select * from user where user = 'neo'\G
*************************** 1. row ***************************
                 Host: 192.168.0.5
                 User: neo
             Password: *7564B7B0A062C9523700601CBA1DCE1F861D6270
          Select_priv: Y
          Insert_priv: Y
          Update_priv: Y
          Delete_priv: Y
          Create_priv: Y
            Drop_priv: Y
          Reload_priv: Y
        Shutdown_priv: Y
         Process_priv: Y
            File_priv: Y
           Grant_priv: N
      References_priv: Y
           Index_priv: Y
           Alter_priv: Y
         Show_db_priv: Y
           Super_priv: Y
Create_tmp_table_priv: Y
     Lock_tables_priv: Y
         Execute_priv: Y
      Repl_slave_priv: Y
     Repl_client_priv: Y
     Create_view_priv: Y
       Show_view_priv: Y
  Create_routine_priv: Y
   Alter_routine_priv: Y
     Create_user_priv: Y
           Event_priv: Y
         Trigger_priv: Y
             ssl_type:
           ssl_cipher:
          x509_issuer:
         x509_subject:
```

```
        max_questions: 0
          max_updates: 0
      max_connections: 0
 max_user_connections: 0
1 row in set (0.00 sec)

mysql>
```

## 2.6. MAX_QUERIES_PER_HOUR/MAX_UPDATES_PER_HOUR

```
GRANT USAGE ON *.* TO ...
  WITH MAX_QUERIES_PER_HOUR 500 MAX_UPDATES_PER_HOUR 100;
```

## 2.7. Table Privileges

授权tmp用户只能访问tabname表

```
GRANT ALL PRIVILEGES ON tmp.tabname TO 'tmp'@'%' IDENTIFIED BY 'chen' WITH GRANT
OPTION;
```

如果用户已经存在仅仅是分配权限可以使用下面方法

```
GRANT ALL ON mydb.mytbl TO 'someuser'@'somehost';
GRANT SELECT, INSERT ON mydb.mytbl TO 'someuser'@'somehost';
```

## 2.8. Column Privileges

mydb.mytbl 表 col1字段允许查询，col1,col2允许插入

```
GRANT SELECT (col1), INSERT (col1,col2) ON mydb.mytbl TO 'someuser'@'somehost';
```

# 3. 字符集转换

找出指定字符集的表

```
select TABLE_SCHEMA,TABLE_NAME,TABLE_COLLATION from
information_schema.tables where table_collation =
'utf8mb4_0900_ai_ci' and table_schema = 'your_schema';
```

```
SELECT
        CONCAT(
                'ALTER TABLE ',
                TABLE_NAME,
                ' CONVERT TO CHARACTER SET utf8mb4 COLLATE
utf8mb4_general_ci;'
        )
FROM
        information_schema.`TABLES`
WHERE
        TABLE_SCHEMA = 'DATABASE_NAME';
```

## 3.1. 转换 latin1 到 UTF-8

```
UPDATE category SET
name=convert(cast(convert(name using latin1) as binary) using
utf8),
description=convert(cast(convert(description using latin1) as
binary) using utf8)
```

# 4. 重新整理AUTO_INCREMENT字段

AUTO_INCREMENT 并非按照我们意愿，顺序排列，经常会跳过一些数字，例如当插入失败的时候，再次插入会使用新的值。有时会造成浪费，我们可以使用下面SQL重新编排AUTO_INCREMENT序列。

```
SET @newid=0;
UPDATE mytable SET id = (SELECT @newid:=@newid+ 1);
```

使用max()查看最大值，然后使用 alter修改起始位置。

```
select max(id) from mytable;
ALTER TABLE mytable AUTO_INCREMENT = 1000;
```

注意外键，需要 ON UPDATE CASCADE 支持，否则无法更新。CONSTRAINT `FK_group_has_contact_contact` FOREIGN KEY (`contact_id`) REFERENCES `contact` (`id`) ON UPDATE CASCADE ON DELETE CASCADE,

```
CREATE TABLE `contact` (
        `id` INT(10) UNSIGNED NOT NULL AUTO_INCREMENT COMMENT
'唯一ID',
        `name` VARCHAR(50) NOT NULL COMMENT '姓名',
        `mobile` VARBINARY(32) NULL DEFAULT NULL COMMENT '手机号
码',
        `email` VARBINARY(50) NULL DEFAULT NULL COMMENT '电子邮
件',
        `mobile_digest` VARCHAR(32) NULL DEFAULT NULL COMMENT
'摘要',
```

```
        `email_digest` VARCHAR(32) NULL DEFAULT NULL COMMENT
'邮件摘要',
        `birthday` DATE NULL DEFAULT NULL COMMENT '生日',
        `description` VARCHAR(255) NULL DEFAULT NULL COMMENT
'备注描述',
        `status`
        ENUM('Subscription','Unsubscribe') NOT NULL DEFAULT
'Subscription' COMMENT '订阅状态',
        `ctime` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP
COMMENT '创建时间',
        `mtime` TIMESTAMP NULL DEFAULT NULL ON UPDATE
CURRENT_TIMESTAMP COMMENT '修改时间',
        PRIMARY KEY (`id`),
        UNIQUE INDEX `digest` (`mobile_digest`, `email_digest`)
)
COMMENT='会员手机短信与电子邮件映射表'
COLLATE='utf8_general_ci'
ENGINE=InnoDB
AUTO_INCREMENT=43642;

CREATE TABLE `group` (
        `id` INT(10) UNSIGNED NOT NULL AUTO_INCREMENT,
        `name` VARCHAR(50) NOT NULL,
        `description` VARCHAR(512)
        NOT NULL,
        `ctime` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
        PRIMARY KEY (`id`),
        UNIQUE INDEX `name` (`name`)
)
COMMENT='短信分组'
COLLATE='utf8_general_ci'
ENGINE=InnoDB
AUTO_INCREMENT=8;

CREATE TABLE `group_has_contact` (
        `id` INT(10) UNSIGNED NOT NULL AUTO_INCREMENT,
        `group_id` INT(10) UNSIGNED NOT NULL,
        `contact_id` INT(10) UNSIGNED NOT NULL,
        `ctime` TIMESTAMP NULL DEFAULT CURRENT_TIMESTAMP,
        PRIMARY KEY (`id`),
        UNIQUE INDEX `group_contact` (`group_id`,
`contact_id`),
        INDEX `FK_group_has_contact_contact`
        (`contact_id`),
        CONSTRAINT `FK_group_has_contact_contact` FOREIGN KEY
```

```
(`contact_id`) REFERENCES `contact` (`id`) ON UPDATE CASCADE ON
DELETE CASCADE,
        CONSTRAINT `FK_group_has_contact_group` FOREIGN KEY
(`group_id`) REFERENCES `group` (`id`) ON UPDATE CASCADE ON
DELETE CASCADE
)
COMMENT='N:M'
COLLATE='utf8_general_ci'
ENGINE=InnoDB
AUTO_INCREMENT=55764;
```

# 5. 数据库内容替换

```
#!/bin/bash
HOST='localhost'
USER='neo'
PASS='chen'

SDB='neo'
TDB='netkiller'
MYSQLDUMP="mysqldump"
MYSQLDUMPOPTS="-h${HOST} -u${USER} -p${PASS}"

MYSQL="mysql"
MYSQLOPTS="-h${HOST} -u${USER} -p${PASS}"
#SED="sed -e 's/netkiller\.8800\.org/netkiller\.sf\.net/g' -e
's/陈景峰/景峰/g' -e 's/Neo/Netkiller/g'"

$MYSQL $MYSQLOPTS <<SQL
DROP DATABASE $TDB;
CREATE DATABASE $TDB DEFAULT CHARACTER SET utf8 COLLATE
utf8_general_ci;
SQL


$MYSQLDUMP $MYSQLDUMPOPTS ${SDB} | sed -e
's/netkiller\.8800\.org/netkiller\.sf\.net/g' -e 's/陈景峰/景
峰/g' -e 's/Neo/Netkiller/g' | $MYSQL $MYSQLOPTS ${TDB}
#echo "$MYSQLDUMP $MYSQLDUMPOPTS ${SDB} | $SED | $MYSQL
$MYSQLOPTS ${TDB}"
```

# 6. Kill 脚本

查询出锁定的表

SELECT concat('KILL ',id,';') FROM information_schema.processlist WHERE user='root';

SELECT concat('KILL ',id,';') FROM information_schema.processlist WHERE command='Locked' and user='root';

SELECT concat('KILL ',id,';') FROM information_schema.processlist WHERE command='Locked' and user='root' and db='test';

拼装kill命令后输入到kill.sql, source 将从kill.sql读取sql命令并执行。

```
SELECT concat('KILL ',id,';') FROM
information_schema.processlist WHERE user='root' INTO OUTFILE
'/tmp/kill.sql';
source /tmp/kill.sql;
```

```
mysqladmin -uroot -p processlist | grep Sleep |awk '{if
(length($2) > 1) print "Kill "$2}'|xargs mysqladmin -uroot kill
```

# 7. MySQL 时区管理

出现时区的问题，首先我们要检查操作系统的时区

```
[root@testing ~]# date
Tue Sep 28 10:33:03 CST 2021

[root@testing ~]# date -R
Tue, 28 Sep 2021 10:33:06 +0800

[root@testing ~]# date +"%Z %z"
CST +0800
```

CST 表示“中国标准时间” +0800

然后在查询数据库时区设置

```
mysql> show variables like "%time_zone";
+------------------+--------+
| Variable_name    | Value  |
+------------------+--------+
| system_time_zone | CST    |
| time_zone        | SYSTEM |
+------------------+--------+
2 rows in set (0.00 sec)
```

system_time_zone 当前时区是 CST，time_zone 表示数据库使用操作系统的时区设置。

临时修改时区 set time_zone='+8:00'; 设置当前时区，set global time_zone = '+8:00'; 设置全局时区。

```
#设置全局时区，即时生效，作用于所有session
mysql> set global time_zone='+8:00';
Query OK, 0 rows affected (0.00 sec)
```

永久修改时区，编辑 /etc/my.cnf 配置文件，在[mysqld]的下面添加或者修改如下内容

```
default-time_zone = '+8:00'
```

最后检查时间设置

```
mysql> select now(), UTC_TIMESTAMP;
+---------------------+---------------------+
| now()               | UTC_TIMESTAMP       |
+---------------------+---------------------+
| 2021-09-28 10:51:45 | 2021-09-28 02:51:45 |
+---------------------+---------------------+
1 row in set (0.00 sec)

mysql> SELECT TIMEDIFF(NOW(), UTC_TIMESTAMP);
+--------------------------------+
| TIMEDIFF(NOW(), UTC_TIMESTAMP) |
+--------------------------------+
| 08:00:00                       |
+--------------------------------+
1 row in set (0.00 sec)

mysql> select
timediff(now(),convert_tz(now(),@@session.time_zone,'+00:00'));
+-------------------------------------------------------------
-+
| timediff(now(),convert_tz(now(),@@session.time_zone,'+00:00'))
|
```

```
+-----------------------------------------------------------------
-+
| 08:00:00
|
+-----------------------------------------------------------------
-+
1 row in set (0.00 sec)

mysql> select curtime();
+-----------+
| curtime() |
+-----------+
| 10:52:40  |
+-----------+
1 row in set (0.00 sec)
```

JDBC 时区设置

```
# serverTimezone=UTC
jdbc.url=jdbc:mysql://localhost:3306/demo?
serverTimezone=UTC&characterEncoding=utf-8

# serverTimezone=GMT%2B8
jdbc.url=jdbc:mysql://localhost:3306/demo?
serverTimezone=GMT%2B8&characterEncoding=utf-8

# serverTimezone=Asia/Shanghai
jdbc.url=jdbc:mysql://localhost:3306/demo?
serverTimezone=Asia/Shanghai&characterEncoding=utf-8
```

# 8. SHOW COMMAND

## 8.1. 查看版本

Server

```
mysql> select version();
+-----------+
| version() |
+-----------+
| 5.0.77 |
+-----------+
1 row in set (0.00 sec)
```

```
mysql> status;
--------------
mysql Ver 14.12 Distrib 5.0.77, for redhat-linux-gnu (x86_64)
using readline 5.1

Connection id: 1533
Current database:
Current user: root@localhost
SSL: Not in use
Current pager: stdout
Using outfile: ''
Using delimiter: ;
Server version: 5.0.77 Source distribution
Protocol version: 10
Connection: Localhost via UNIX socket
Server characterset: latin1
Db characterset: latin1
Client characterset: latin1
Conn. characterset: latin1
UNIX socket: /var/lib/mysql/mysql.sock
Uptime: 1 day 21 hours 40 min 52 sec

Threads: 1 Questions: 22172 Slow queries: 0 Opens: 3130 Flush
tables: 1 Open tables: 64 Queries per second avg: 0.135
--------------
```

```
```

Client

```
[root@development ~]# mysql -V
mysql Ver 14.12 Distrib 5.0.77, for redhat-linux-gnu (x86_64)
using readline 5.1
```

## 8.2. status

```
mysql> show status;
mysql> show global status;
```

### show status

```
数据库性能状态
(1)QPS(每秒Query量)
QPS = Questions(or Queries) / seconds
mysql > show /*50000 global */ status like 'Question';

(2)TPS(每秒事务量)
TPS = (Com_commit + Com_rollback) / seconds
mysql > show status like 'Com_commit';
mysql > show status like 'Com_rollback';

(3)key Buffer 命中率
key_buffer_read_hits = (1-key_reads / key_read_requests) * 100%
key_buffer_write_hits = (1-key_writes / key_write_requests) *
100%

mysql> show status like 'Key%';

(4)InnoDB Buffer命中率
innodb_buffer_read_hits = (1 - innodb_buffer_pool_reads /
```

```
innodb_buffer_pool_read_requests) * 100%

mysql> show status like 'innodb_buffer_pool_read%';

(5)Query Cache命中率
Query_cache_hits = (Qcahce_hits / (Qcache_hits + Qcache_inserts
)) * 100%;

mysql> show status like 'Qcache%';
(6)Table Cache状态量
mysql> show status like 'open%';

(7)Thread Cache 命中率
Thread_cache_hits = (1 - Threads_created / connections ) * 100%

mysql> show status like 'Thread%';

mysql> show status like 'Connections';

(8)锁定状态
mysql> show status like '%lock%';

(9)复制延时量
mysql > show slave status

(10) Tmp Table 状况(临时表状况)
mysql > show status like 'Create_tmp%';
(11) Binlog Cache 使用状况
mysql > show status like 'Binlog_cache%';

(12) Innodb_log_waits 量
mysql > show status like 'innodb_log_waits';
```

## show master status

```
mysql> show master status;
+--------------------+----------+--------------+-------------
-----+
| File               | Position  | Binlog_Do_DB |
Binlog_Ignore_DB |
+--------------------+----------+--------------+-------------
```

```
-----+
| DBMaster-bin.000018 | 409468882 | example      |
|
+---------------------+-----------+--------------+-------------
-----+
1 row in set (0.00 sec)

mysql>
```

## show slave status

```
mysql> show slave status/G

得到的列表会有类似下面的数据：

File: mysql-bin.000001
Position: 1374
Binlog_Do_DB: test
Binlog_Ignore_DB: mysql

Slave_IO_State: Waiting for master to send event
Slave_IO_Running: Yes
Slave_SQL_Running: Yes
```

## show plugins

```
mysql> SHOW PLUGINS;
+------------+----------+----------------+---------+---------+
| Name       | Status   | Type           | Library | License |
+------------+----------+----------------+---------+---------+
| binlog     | ACTIVE   | STORAGE ENGINE | NULL    | GPL     |
| partition  | ACTIVE   | STORAGE ENGINE | NULL    | GPL     |
| ARCHIVE    | ACTIVE   | STORAGE ENGINE | NULL    | GPL     |
| BLACKHOLE  | ACTIVE   | STORAGE ENGINE | NULL    | GPL     |
| CSV        | ACTIVE   | STORAGE ENGINE | NULL    | GPL     |
```

```
| FEDERATED  | DISABLED | STORAGE ENGINE | NULL    | GPL     |
| MEMORY     | ACTIVE   | STORAGE ENGINE | NULL    | GPL     |
| InnoDB     | ACTIVE   | STORAGE ENGINE | NULL    | GPL     |
| MyISAM     | ACTIVE   | STORAGE ENGINE | NULL    | GPL     |
| MRG_MYISAM | ACTIVE   | STORAGE ENGINE | NULL    | GPL     |
+------------+----------+----------------+---------+---------+
10 rows in set (0.00 sec)
```

## 8.3. show processlist

```
show full processlist;
                </screen>
                <screen><![CDATA[
命令： show processlist;
如果是root帐号，你能看到所有用户的当前连接。如果是其它普通帐号，只能看到自己占用的连接。
show processlist;只列出前100条，如果想全列出请使用show full processlist;
mysql> show processlist;

命令： show status;

Aborted_clients 由于客户没有正确关闭连接已经死掉，已经放弃的连接数量。
Aborted_connects 尝试已经失败的MySQL服务器的连接的次数。
Connections 试图连接MySQL服务器的次数。
Created_tmp_tables 当执行语句时，已经被创造了的隐含临时表的数量。
Delayed_insert_threads 正在使用的延迟插入处理器线程的数量。
Delayed_writes 用INSERT DELAYED写入的行数。
Delayed_errors 用INSERT DELAYED写入的发生某些错误(可能重复键值)的行数。
Flush_commands 执行FLUSH命令的次数。
Handler_delete 请求从一张表中删除行的次数。
Handler_read_first 请求读入表中第一行的次数。
Handler_read_key 请求数字基于键读行。
Handler_read_next 请求读入基于一个键的一行的次数。
Handler_read_rnd 请求读入基于一个固定位置的一行的次数。
Handler_update 请求更新表中一行的次数。
Handler_write 请求向表中插入一行的次数。
Key_blocks_used 用于关键字缓存的块的数量。
Key_read_requests 请求从缓存读入一个键值的次数。
Key_reads 从磁盘物理读入一个键值的次数。
```

```
Key_write_requests 请求将一个关键字块写入缓存次数。
Key_writes 将一个键值块物理写入磁盘的次数。
Max_used_connections 同时使用的连接的最大数目。
Not_flushed_key_blocks 在键缓存中已经改变但是还没被清空到磁盘上的键块。
Not_flushed_delayed_rows 在INSERT DELAY队列中等待写入的行的数量。
Open_tables 打开表的数量。
Open_files 打开文件的数量。
Open_streams 打开流的数量(主要用于日志记载)
Opened_tables 已经打开的表的数量。
Questions 发往服务器的查询的数量。
Slow_queries 要花超过long_query_time时间的查询数量。
Threads_connected 当前打开的连接的数量。
Threads_running 不在睡眠的线程数量。
Uptime 服务器工作了多少秒。
                <</screen>
        </section>


        <section id="binary">
                <title>binary 日志</title>
                <screen>
                <![CDATA[
mysql> show binary logs;
+------------------+-----------+
| Log_name         | File_size |
+------------------+-----------+
| mysql-bin.000001 |     19544 |
| mysql-bin.000002 |    974751 |
| mysql-bin.000003 |       107 |
| mysql-bin.000004 |   3976040 |
| mysql-bin.000005 |       126 |
| mysql-bin.000006 |    350063 |
| mysql-bin.000007 |      6826 |
| mysql-bin.000008 |   3879494 |
| mysql-bin.000009 |       126 |
| mysql-bin.000010 |       494 |
| mysql-bin.000011 |  17286686 |
| mysql-bin.000012 |  15003942 |
| mysql-bin.000013 |   1709321 |
+------------------+-----------+
13 rows in set (0.00 sec)
```

## 8.4. 线程的使用情况

```
mysql> SHOW STATUS LIKE 'threads%';
+-------------------+-------+
| Variable_name     | Value |
+-------------------+-------+
| Threads_cached    | 0     |
| Threads_connected | 1     |
| Threads_created   | 1     |
| Threads_running   | 1     |
+-------------------+-------+
4 rows in set (0.01 sec)
```

## 8.5. DATABASES

```
SHOW DATABASES;
                </screen>
        </section>
        <section id="table">
                <title>TABLE</title>
                <screen><![CDATA[
SHOW TABLE STATUS FROM `dbname`;
                </screen>
        </section>
        <section id="created_tmp">
                <title>临时表</title>
                <screen>
                <![CDATA[
mysql> SHOW STATUS LIKE 'created_tmp%';
+-------------------------+-------+
| Variable_name           | Value |
+-------------------------+-------+
| Created_tmp_disk_tables | 0     |
| Created_tmp_files       | 5     |
| Created_tmp_tables      | 0     |
+-------------------------+-------+
3 rows in set (0.00 sec)
```

## 8.6. 排序统计信息

```
mysql> SHOW STATUS LIKE "sort%";
+-------------------+-------+
| Variable_name     | Value |
+-------------------+-------+
| Sort_merge_passes | 0     |
| Sort_range        | 0     |
| Sort_rows         | 0     |
| Sort_scan         | 0     |
+-------------------+-------+
4 rows in set (0.00 sec)
```

## 8.7. Key 状态

```
mysql> show status like '%key_read%';
+-------------------+-------+
| Variable_name     | Value |
+-------------------+-------+
| Key_read_requests | 6     |
| Key_reads         | 3     |
+-------------------+-------+
2 rows in set (0.00 sec)
```

## 8.8. FUNCTION

```
SHOW FUNCTION STATUS WHERE `Db`='dbname';
```

## 8.9. PROCEDURE

```
SHOW PROCEDURE STATUS WHERE `Db`='dbname';
```

## 8.10. TRIGGERS

```
SHOW TRIGGERS FROM `dbname`;
```

```
mysql> SHOW TRIGGERS LIKE '%trigger_name%'\G
Empty set (0.00 sec)

mysql> SHOW TRIGGERS LIKE '%demo%'\G
*************************** 1. row
***************************
             Trigger: demo_AFTER_INSERT
               Event: INSERT
               Table: demo
           Statement: BEGIN
        set @rev = "";
        SELECT
    OUT2FILE('/tmp/demo.log',
            CONCAT_WS(',',
                    NEW.id,
                    NEW.name,
                    NEW.sex,
                    NEW.address))
INTO @rev;
END
              Timing: AFTER
             Created: 2017-11-23 11:47:58.10
            sql_mode:
ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZER
O_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_AUTO_CREATE_USER,NO_ENGI
NE_SUBSTITUTION
             Definer: root@%
character_set_client: utf8
collation_connection: utf8_general_ci
  Database Collation: utf8_general_ci
1 row in set (0.00 sec)
```

## 8.11. EVENTS

```
SHOW EVENTS FROM `dbname`;
```

## 8.12. 引擎(ENGINES)

```
mysql> SHOW ENGINES;
+--------------------+---------+------------------------------
--------------------------------+--------------+------+-------
-----+
| Engine             | Support | Comment
| Transactions | XA   | Savepoints |
+--------------------+---------+------------------------------
--------------------------------+--------------+------+-------
-----+
| CSV                | YES     | CSV storage engine
| NO           | NO   | NO         |
| MRG_MYISAM         | YES     | Collection of identical MyISAM
tables                          | NO           | NO   | NO
|
| PERFORMANCE_SCHEMA | YES     | Performance Schema
| NO           | NO   | NO         |
| BLACKHOLE          | YES     | /dev/null storage engine
(anything you write to it disappears) | NO           | NO   |
NO         |
| MEMORY             | YES     | Hash based, stored in memory,
useful for temporary tables      | NO           | NO   | NO
|
| FEDERATED          | NO      | Federated MySQL storage engine
| NULL         | NULL | NULL       |
| ARCHIVE            | YES     | Archive storage engine
| NO           | NO   | NO         |
| MyISAM             | YES     | MyISAM storage engine
| NO           | NO   | NO         |
```

```
| InnoDB             | DEFAULT | Supports transactions, row-
level locking, and foreign keys     | YES          | YES  | YES
|
+--------------------+---------+-----------------------------
--------------------------------+--------------+------+-------
-----+
9 rows in set (0.00 sec)
```

## 8.13. 字符集(Collation)

```
mysql> SHOW COLLATION;
+--------------------------+----------+-----+---------+--------
--+---------+
| Collation                | Charset  | Id  | Default |
Compiled | Sortlen |
+--------------------------+----------+-----+---------+--------
--+---------+
| big5_chinese_ci          | big5     |   1 | Yes     | Yes
|       1 |
| big5_bin                 | big5     |  84 |         | Yes
|       1 |
| dec8_swedish_ci          | dec8     |   3 | Yes     | Yes
|       1 |
| dec8_bin                 | dec8     |  69 |         | Yes
|       1 |
| cp850_general_ci         | cp850    |   4 | Yes     | Yes
|       1 |
| cp850_bin                | cp850    |  80 |         | Yes
|       1 |
| hp8_english_ci           | hp8      |   6 | Yes     | Yes
|       1 |
| hp8_bin                  | hp8      |  72 |         | Yes
|       1 |
| koi8r_general_ci         | koi8r    |   7 | Yes     | Yes
|       1 |
| koi8r_bin                | koi8r    |  74 |         | Yes
|       1 |
| latin1_german1_ci        | latin1   |   5 |         | Yes
|       1 |
```

```
| latin1_swedish_ci        | latin1   |   8 | Yes      | Yes
|       1 |
| latin1_danish_ci         | latin1   |  15 |          | Yes
|       1 |
| latin1_german2_ci        | latin1   |  31 |          | Yes
|       2 |
| latin1_bin               | latin1   |  47 |          | Yes
|       1 |
| latin1_general_ci        | latin1   |  48 |          | Yes
|       1 |
| latin1_general_cs        | latin1   |  49 |          | Yes
|       1 |
| latin1_spanish_ci        | latin1   |  94 |          | Yes
|       1 |
| latin2_czech_cs          | latin2   |   2 |          | Yes
|       4 |
| latin2_general_ci        | latin2   |   9 | Yes      | Yes
|       1 |
| latin2_hungarian_ci      | latin2   |  21 |          | Yes
|       1 |
| latin2_croatian_ci       | latin2   |  27 |          | Yes
|       1 |
| latin2_bin               | latin2   |  77 |          | Yes
|       1 |
| swe7_swedish_ci          | swe7     |  10 | Yes      | Yes
|       1 |
| swe7_bin                 | swe7     |  82 |          | Yes
|       1 |
| ascii_general_ci         | ascii    |  11 | Yes      | Yes
|       1 |
| ascii_bin                | ascii    |  65 |          | Yes
|       1 |
| ujis_japanese_ci         | ujis     |  12 | Yes      | Yes
|       1 |
| ujis_bin                 | ujis     |  91 |          | Yes
|       1 |
| sjis_japanese_ci         | sjis     |  13 | Yes      | Yes
|       1 |
| sjis_bin                 | sjis     |  88 |          | Yes
|       1 |
| hebrew_general_ci        | hebrew   |  16 | Yes      | Yes
|       1 |
| hebrew_bin               | hebrew   |  71 |          | Yes
|       1 |
| tis620_thai_ci           | tis620   |  18 | Yes      | Yes
```

```
|       4 |
| tis620_bin                | tis620   |  89 |         | Yes
|       1 |
| euckr_korean_ci           | euckr    |  19 | Yes     | Yes
|       1 |
| euckr_bin                 | euckr    |  85 |         | Yes
|       1 |
| koi8u_general_ci          | koi8u    |  22 | Yes     | Yes
|       1 |
| koi8u_bin                 | koi8u    |  75 |         | Yes
|       1 |
| gb2312_chinese_ci         | gb2312   |  24 | Yes     | Yes
|       1 |
| gb2312_bin                | gb2312   |  86 |         | Yes
|       1 |
| greek_general_ci          | greek    |  25 | Yes     | Yes
|       1 |
| greek_bin                 | greek    |  70 |         | Yes
|       1 |
| cp1250_general_ci         | cp1250   |  26 | Yes     | Yes
|       1 |
| cp1250_czech_cs           | cp1250   |  34 |         | Yes
|       2 |
| cp1250_croatian_ci        | cp1250   |  44 |         | Yes
|       1 |
| cp1250_bin                | cp1250   |  66 |         | Yes
|       1 |
| cp1250_polish_ci          | cp1250   |  99 |         | Yes
|       1 |
| gbk_chinese_ci            | gbk      |  28 | Yes     | Yes
|       1 |
| gbk_bin                   | gbk      |  87 |         | Yes
|       1 |
| latin5_turkish_ci         | latin5   |  30 | Yes     | Yes
|       1 |
| latin5_bin                | latin5   |  78 |         | Yes
|       1 |
| armscii8_general_ci       | armscii8 |  32 | Yes     | Yes
|       1 |
| armscii8_bin              | armscii8 |  64 |         | Yes
|       1 |
| utf8_general_ci           | utf8     |  33 | Yes     | Yes
|       1 |
| utf8_bin                  | utf8     |  83 |         | Yes
|       1 |
```

```
| utf8_unicode_ci          | utf8     | 192 |         | Yes
|       8 |
| utf8_icelandic_ci        | utf8     | 193 |         | Yes
|       8 |
| utf8_latvian_ci          | utf8     | 194 |         | Yes
|       8 |
| utf8_romanian_ci         | utf8     | 195 |         | Yes
|       8 |
| utf8_slovenian_ci        | utf8     | 196 |         | Yes
|       8 |
| utf8_polish_ci           | utf8     | 197 |         | Yes
|       8 |
| utf8_estonian_ci         | utf8     | 198 |         | Yes
|       8 |
| utf8_spanish_ci          | utf8     | 199 |         | Yes
|       8 |
| utf8_swedish_ci          | utf8     | 200 |         | Yes
|       8 |
| utf8_turkish_ci          | utf8     | 201 |         | Yes
|       8 |
| utf8_czech_ci            | utf8     | 202 |         | Yes
|       8 |
| utf8_danish_ci           | utf8     | 203 |         | Yes
|       8 |
| utf8_lithuanian_ci       | utf8     | 204 |         | Yes
|       8 |
| utf8_slovak_ci           | utf8     | 205 |         | Yes
|       8 |
| utf8_spanish2_ci         | utf8     | 206 |         | Yes
|       8 |
| utf8_roman_ci            | utf8     | 207 |         | Yes
|       8 |
| utf8_persian_ci          | utf8     | 208 |         | Yes
|       8 |
| utf8_esperanto_ci        | utf8     | 209 |         | Yes
|       8 |
| utf8_hungarian_ci        | utf8     | 210 |         | Yes
|       8 |
| utf8_sinhala_ci          | utf8     | 211 |         | Yes
|       8 |
| utf8_general_mysql500_ci | utf8     | 223 |         | Yes
|       1 |
| ucs2_general_ci          | ucs2     |  35 | Yes     | Yes
|       1 |
| ucs2_bin                 | ucs2     |  90 |         | Yes
```

```
|       1 |
| ucs2_unicode_ci          | ucs2     | 128 |         | Yes
|       8 |
| ucs2_icelandic_ci        | ucs2     | 129 |         | Yes
|       8 |
| ucs2_latvian_ci          | ucs2     | 130 |         | Yes
|       8 |
| ucs2_romanian_ci         | ucs2     | 131 |         | Yes
|       8 |
| ucs2_slovenian_ci        | ucs2     | 132 |         | Yes
|       8 |
| ucs2_polish_ci           | ucs2     | 133 |         | Yes
|       8 |
| ucs2_estonian_ci         | ucs2     | 134 |         | Yes
|       8 |
| ucs2_spanish_ci          | ucs2     | 135 |         | Yes
|       8 |
| ucs2_swedish_ci          | ucs2     | 136 |         | Yes
|       8 |
| ucs2_turkish_ci          | ucs2     | 137 |         | Yes
|       8 |
| ucs2_czech_ci            | ucs2     | 138 |         | Yes
|       8 |
| ucs2_danish_ci           | ucs2     | 139 |         | Yes
|       8 |
| ucs2_lithuanian_ci       | ucs2     | 140 |         | Yes
|       8 |
| ucs2_slovak_ci           | ucs2     | 141 |         | Yes
|       8 |
| ucs2_spanish2_ci         | ucs2     | 142 |         | Yes
|       8 |
| ucs2_roman_ci            | ucs2     | 143 |         | Yes
|       8 |
| ucs2_persian_ci          | ucs2     | 144 |         | Yes
|       8 |
| ucs2_esperanto_ci        | ucs2     | 145 |         | Yes
|       8 |
| ucs2_hungarian_ci        | ucs2     | 146 |         | Yes
|       8 |
| ucs2_sinhala_ci          | ucs2     | 147 |         | Yes
|       8 |
| ucs2_general_mysql500_ci | ucs2     | 159 |         | Yes
|       1 |
| cp866_general_ci         | cp866    |  36 | Yes     | Yes
|       1 |
```

```
| cp866_bin                | cp866    |  68 |         | Yes
|       1 |
| keybcs2_general_ci       | keybcs2  |  37 | Yes     | Yes
|       1 |
| keybcs2_bin              | keybcs2  |  73 |         | Yes
|       1 |
| macce_general_ci         | macce    |  38 | Yes     | Yes
|       1 |
| macce_bin                | macce    |  43 |         | Yes
|       1 |
| macroman_general_ci      | macroman |  39 | Yes     | Yes
|       1 |
| macroman_bin             | macroman |  53 |         | Yes
|       1 |
| cp852_general_ci         | cp852    |  40 | Yes     | Yes
|       1 |
| cp852_bin                | cp852    |  81 |         | Yes
|       1 |
| latin7_estonian_cs       | latin7   |  20 |         | Yes
|       1 |
| latin7_general_ci        | latin7   |  41 | Yes     | Yes
|       1 |
| latin7_general_cs        | latin7   |  42 |         | Yes
|       1 |
| latin7_bin               | latin7   |  79 |         | Yes
|       1 |
| utf8mb4_general_ci       | utf8mb4  |  45 | Yes     | Yes
|       1 |
| utf8mb4_bin              | utf8mb4  |  46 |         | Yes
|       1 |
| utf8mb4_unicode_ci       | utf8mb4  | 224 |         | Yes
|       8 |
| utf8mb4_icelandic_ci     | utf8mb4  | 225 |         | Yes
|       8 |
| utf8mb4_latvian_ci       | utf8mb4  | 226 |         | Yes
|       8 |
| utf8mb4_romanian_ci      | utf8mb4  | 227 |         | Yes
|       8 |
| utf8mb4_slovenian_ci     | utf8mb4  | 228 |         | Yes
|       8 |
| utf8mb4_polish_ci        | utf8mb4  | 229 |         | Yes
|       8 |
| utf8mb4_estonian_ci      | utf8mb4  | 230 |         | Yes
|       8 |
| utf8mb4_spanish_ci       | utf8mb4  | 231 |         | Yes
```

```
|       8 |
| utf8mb4_swedish_ci       | utf8mb4  | 232 |         | Yes
|       8 |
| utf8mb4_turkish_ci       | utf8mb4  | 233 |         | Yes
|       8 |
| utf8mb4_czech_ci         | utf8mb4  | 234 |         | Yes
|       8 |
| utf8mb4_danish_ci        | utf8mb4  | 235 |         | Yes
|       8 |
| utf8mb4_lithuanian_ci    | utf8mb4  | 236 |         | Yes
|       8 |
| utf8mb4_slovak_ci        | utf8mb4  | 237 |         | Yes
|       8 |
| utf8mb4_spanish2_ci      | utf8mb4  | 238 |         | Yes
|       8 |
| utf8mb4_roman_ci         | utf8mb4  | 239 |         | Yes
|       8 |
| utf8mb4_persian_ci       | utf8mb4  | 240 |         | Yes
|       8 |
| utf8mb4_esperanto_ci     | utf8mb4  | 241 |         | Yes
|       8 |
| utf8mb4_hungarian_ci     | utf8mb4  | 242 |         | Yes
|       8 |
| utf8mb4_sinhala_ci       | utf8mb4  | 243 |         | Yes
|       8 |
| cp1251_bulgarian_ci      | cp1251   |  14 |         | Yes
|       1 |
| cp1251_ukrainian_ci      | cp1251   |  23 |         | Yes
|       1 |
| cp1251_bin               | cp1251   |  50 |         | Yes
|       1 |
| cp1251_general_ci        | cp1251   |  51 | Yes     | Yes
|       1 |
| cp1251_general_cs        | cp1251   |  52 |         | Yes
|       1 |
| utf16_general_ci         | utf16    |  54 | Yes     | Yes
|       1 |
| utf16_bin                | utf16    |  55 |         | Yes
|       1 |
| utf16_unicode_ci         | utf16    | 101 |         | Yes
|       8 |
| utf16_icelandic_ci       | utf16    | 102 |         | Yes
|       8 |
| utf16_latvian_ci         | utf16    | 103 |         | Yes
|       8 |
```

| utf16_romanian_ci | utf16 | 104 | | Yes | 8 |
| utf16_slovenian_ci | utf16 | 105 | | Yes | 8 |
| utf16_polish_ci | utf16 | 106 | | Yes | 8 |
| utf16_estonian_ci | utf16 | 107 | | Yes | 8 |
| utf16_spanish_ci | utf16 | 108 | | Yes | 8 |
| utf16_swedish_ci | utf16 | 109 | | Yes | 8 |
| utf16_turkish_ci | utf16 | 110 | | Yes | 8 |
| utf16_czech_ci | utf16 | 111 | | Yes | 8 |
| utf16_danish_ci | utf16 | 112 | | Yes | 8 |
| utf16_lithuanian_ci | utf16 | 113 | | Yes | 8 |
| utf16_slovak_ci | utf16 | 114 | | Yes | 8 |
| utf16_spanish2_ci | utf16 | 115 | | Yes | 8 |
| utf16_roman_ci | utf16 | 116 | | Yes | 8 |
| utf16_persian_ci | utf16 | 117 | | Yes | 8 |
| utf16_esperanto_ci | utf16 | 118 | | Yes | 8 |
| utf16_hungarian_ci | utf16 | 119 | | Yes | 8 |
| utf16_sinhala_ci | utf16 | 120 | | Yes | 8 |
| cp1256_general_ci | cp1256 | 57 | Yes | Yes | 1 |
| cp1256_bin | cp1256 | 67 | | Yes | 1 |
| cp1257_lithuanian_ci | cp1257 | 29 | | Yes | 1 |
| cp1257_bin | cp1257 | 58 | | Yes | 1 |
| cp1257_general_ci | cp1257 | 59 | Yes | Yes | 1 |
| utf32_general_ci | utf32 | 60 | Yes | Yes | |

```
|       1 |
| utf32_bin                | utf32    |  61 |         | Yes
|       1 |
| utf32_unicode_ci         | utf32    | 160 |         | Yes
|       8 |
| utf32_icelandic_ci       | utf32    | 161 |         | Yes
|       8 |
| utf32_latvian_ci         | utf32    | 162 |         | Yes
|       8 |
| utf32_romanian_ci        | utf32    | 163 |         | Yes
|       8 |
| utf32_slovenian_ci       | utf32    | 164 |         | Yes
|       8 |
| utf32_polish_ci          | utf32    | 165 |         | Yes
|       8 |
| utf32_estonian_ci        | utf32    | 166 |         | Yes
|       8 |
| utf32_spanish_ci         | utf32    | 167 |         | Yes
|       8 |
| utf32_swedish_ci         | utf32    | 168 |         | Yes
|       8 |
| utf32_turkish_ci         | utf32    | 169 |         | Yes
|       8 |
| utf32_czech_ci           | utf32    | 170 |         | Yes
|       8 |
| utf32_danish_ci          | utf32    | 171 |         | Yes
|       8 |
| utf32_lithuanian_ci      | utf32    | 172 |         | Yes
|       8 |
| utf32_slovak_ci          | utf32    | 173 |         | Yes
|       8 |
| utf32_spanish2_ci        | utf32    | 174 |         | Yes
|       8 |
| utf32_roman_ci           | utf32    | 175 |         | Yes
|       8 |
| utf32_persian_ci         | utf32    | 176 |         | Yes
|       8 |
| utf32_esperanto_ci       | utf32    | 177 |         | Yes
|       8 |
| utf32_hungarian_ci       | utf32    | 178 |         | Yes
|       8 |
| utf32_sinhala_ci         | utf32    | 179 |         | Yes
|       8 |
| binary                   | binary   |  63 | Yes     | Yes
|       1 |
```

```
| geostd8_general_ci       | geostd8  |  92 | Yes     | Yes
|       1 |
| geostd8_bin              | geostd8  |  93 |         | Yes
|       1 |
| cp932_japanese_ci        | cp932    |  95 | Yes     | Yes
|       1 |
| cp932_bin                | cp932    |  96 |         | Yes
|       1 |
| eucjpms_japanese_ci      | eucjpms  |  97 | Yes     | Yes
|       1 |
| eucjpms_bin              | eucjpms  |  98 |         | Yes
|       1 |
+--------------------------+----------+-----+---------+--------
--+---------+
```

## 8.14. SHOW GRANTS

```
MariaDB [test]> SHOW GRANTS;
+-----------------------------------------------------------------
-----------------------------------------------------------------
-----------+
| Grants for root@localhost |
+-----------------------------------------------------------------
-----------------------------------------------------------------
-----------+
| GRANT ALL PRIVILEGES ON *.* TO 'root'@'localhost' IDENTIFIED
BY PASSWORD '*C6325DAF39AE6CC34E960D3C65F1398FE467E1D0' WITH
GRANT OPTION |
| GRANT PROXY ON ''@'' TO 'root'@'localhost' WITH GRANT OPTION
|
+-----------------------------------------------------------------
-----------------------------------------------------------------
-----------+
2 rows in set (0.00 sec)
```

## 8.15. validate_password

```
```

```
SHOW VARIABLES LIKE 'validate_password.%';

mysql> SHOW VARIABLES LIKE 'validate_password.%';
+--------------------------------------+--------+
| Variable_name                        | Value  |
+--------------------------------------+--------+
| validate_password.check_user_name    | ON     |
| validate_password.dictionary_file    |        |
| validate_password.length             | 8      |
| validate_password.mixed_case_count   | 1      |
| validate_password.number_count       | 1      |
| validate_password.policy             | MEDIUM |
| validate_password.special_char_count | 1      |
+--------------------------------------+--------+
7 rows in set (0.00 sec)
```

# 9. Maintenance 数据库维护

## 9.1. CHECK 检查表

```
<![CDATA[
CHECK TABLE `dbname`.`actions`;
CHECK TABLE `dbname`.`actions` QUICK FAST MEDIUM EXTENDED
CHANGED;
```

## 9.2. ANALYZE 分析表

```
ANALYZE TABLE `dbname`.`actions`;
```

## 9.3. CHECKSUM

```
CHECKSUM TABLE `dbname`.`actions` QUICK;
```

## 9.4. OPTIMIZE 优化表

```
OPTIMIZE TABLE `dbname`.`actions`;
```

## 9.5. REPAIR 修复

```
REPAIR TABLE `dbname`.`members`;
```

```
SHOW TABLE STATUS LIKE 'members';
```

# 10. INFORMATION_SCHEMA

## 10.1. 查询表字段

```
SELECT
    GROUP_CONCAT(COLUMN_NAME) AS fields
FROM
    INFORMATION_SCHEMA.Columns
WHERE
    table_name = 'mytable'
        AND table_schema = 'test';
```

## 10.2. 列出所有触发器

```
select trigger_schema, trigger_name, action_statement from
information_schema.triggers

select * from information_schema.triggers where
information_schema.triggers.trigger_schema like '%test%';

select * from information_schema.triggers where
information_schema.triggers.trigger_name like '%trigger_name%' and
information_schema.triggers.trigger_schema like '%data_base_name%';
```

## 10.3. 查看表数据尺寸

```
SELECT
    table_name, data_length
FROM
    information_schema.tables
WHERE
    table_schema = 'netkilller'
ORDER BY data_length DESC
LIMIT 10;
```

# 11. Backup and Recovery

## 11.1. Import / Export

### Export(Backup)

```
mysqldump -hlocalhost -proot -p**** mydb > mydb.sql
```

gzip

```
mysqldump -hlocalhost -proot -p**** mydb | gzip > mydb.sql.gz
```

### Import(Recovery)

```
mysql -hlocalhost -proot -p**** mydb < mydb.sql
```

gunzip

```
gunzip mydb.sql.gz -c | mysql -hlocalhost -proot -p**** mydb
```

### xml

export xml

```
$ mysqldump -uusrname -ppasswd -X -t database table -r
filename.xml
```

备份表数据

```
SELECT * INTO OUTFILE 'file_name' FROM tbl_name
LOAD DATA INFILE 'file_name' REPLACE INTO TABLE tbl_name
```

**source**

```
mysql> use your_db
mysql> SOURCE database.sql
```

使用 **mysqlhotcopy** 备份 **MyISAM** 引擎的数据库

**shell> mysqlhotcopy db_name /path/to/some/dir**

```
mysql:~# mysqlhotcopy --user=neo --password=chen shop
/tmp/backup
Locked 100 tables in 0 seconds.
Flushed tables (`shop`.`account_log`, `shop`.`ad`,
`shop`.`ad_custom`, `shop`.`ad_position`,
`shop`.`admin_action`,
`shop`.`admin_log`, `shop`.`admin_message`,
`shop`.`admin_user`, `shop`.`adsense`, `shop`.`affiliate_log`,
...
...
...
`shop`.`user_rank`, `shop`.`users`, `shop`.`virtual_card`,
`shop`.`volume_price`, `shop`.`vote`, `shop`.`vote_log`,
```

```
`shop`.`vote_option`, `shop`.`wholesale`) in 0 seconds.
Copying 299 files...
Copying indices for 0 files...
Unlocked tables.
mysqlhotcopy copied 100 tables (299 files) in 0 seconds (0
seconds overall).
```

## AutoMySQLBackup

https://sourceforge.net/projects/automysqlbackup/

## xtrabackup - Open source backup tool for InnoDB and XtraDB.

https://launchpad.net/percona-xtrabackup

#### Percona yum Repository

```
$ rpm -Uhv http://www.percona.com/downloads/percona-
release/percona-release-0.0-1.x86_64.rpm
```

```
# yum search xtrabackup
==========================================================
= N/S Matched: XtraBackup
==========================================================
==
holland-xtrabackup.noarch : Xtrabackup plugin for Holland
percona-xtrabackup.x86_64 : XtraBackup online backup for MySQL
/ InnoDB
percona-xtrabackup-debuginfo.x86_64 : Debug information for
package percona-xtrabackup
percona-xtrabackup-test.x86_64 : Test suite for Percona
Xtrabackup
```

```
# yum install percona-xtrabackup
```

**Creating an Incremental Backup**

```
xtrabackup --backup --target-dir=/data/backups/base --
datadir=/var/lib/mysql/
```

# 11.2. Snapshot Backup

## LVM Snapshot

```
# mysql –uroot –pmysql
mysql> flush tables with read lock;
mysql>flush logs;
mysql>system lvcreate -L1024M -s -n snap0 /dev/vg00/lvol00
mysql>show master status;
mysql>unlock tables;
mysql>quit
```

## Btrfs Snapshot

```
# btrfs subvolume snapshot /data /data/backup_2013-03-20
Create a snapshot of '/data' in '/data/backup_2013-03-20'

btrfs subvolume list /data
ID 315 gen 172 top level 5 path backup_2013-03-10
ID 320 gen 178 top level 5 path backup_2013-03-20
```

# 第 4 章 DDL - Data Definition Language

## 1. 数据库管理(Database)

### 1.1. 创建数据库

Creating a UTF-8 database

```
CREATE DATABASE db_name DEFAULT CHARACTER SET utf8 COLLATE
utf8_general_ci;
```

Create a UTF-8 database with binary UTF-8 collation.

```
CREATE DATABASE dbname CHARACTER SET utf8 COLLATE utf8_bin;
```

### 1.2. 删除数据库

```
DROP DATABASE db_name;
```

### 1.3. 修改数据库

```
ALTER DATABASE dbname DEFAULT CHARACTER SET utf8 COLLATE
utf8_general_ci;
```

```

```

、

## 1.4. 重命名数据库

```
RENAME {DATABASE | SCHEMA} db_name TO new_db_name;
```

before 5.0 version

```
[neo@development ~]$ mysqldump -uroot -pchen db_old | mysql -
uroot -pchen db_new
```

## 1.5. 修改字符集

```
ALTER DATABASE <database_name> CHARACTER SET utf8;
```

## 1.6. 查看数据库创建语句

```
mysql> show create database dbname;
+----------+-----------------------------------------------------
----------------+
| Database | Create Database
|
+----------+-----------------------------------------------------
----------------+
| dbname   | CREATE DATABASE `dbname` /*!40100 DEFAULT
CHARACTER SET utf8 */   |
+----------+-----------------------------------------------------
----------------+
1 row in set (0.00 sec)
```

# 2. 表管理(Table)

## 2.1. 数据类型

### SET 集合类型

SET 集合类型，此类型适合用于多项选择场景，例如保存表单中的checkbox。

```
CREATE TABLE `QA` (
        `id` INT(10) UNSIGNED NOT NULL AUTO_INCREMENT,
        `question` VARCHAR(255) NOT NULL COMMENT '问题描述',
        `answer` SET('A','B','C','D') NOT NULL COMMENT '问题答案',
        PRIMARY KEY (`id`)
)
COMMENT='Multiple Choice'
COLLATE='utf8_general_ci'
ENGINE=InnoDB;
```

插入数据

```
INSERT INTO `QA` (`id`, `question`, `answer`) VALUES
        (1, 'Netkiller 系列手札始于那一年？ A.2000年, B.2008年, C.2010年, D.2016年', 'A'),
        (2, 'Netkiller 系列手札有哪些? A.《Netkiller Scals 手札》, B.《Netkiller Java 手札》,
C.《Netkiller Linux 手札》, D.《Netkiller EMC 手札》', 'B,C'),
        (3, 'XXXXXXXXX', 'C,D'),
        (4, 'XXXXXXXXX', 'A,B,C'),
        ...
        ...
        (1000, 'XXXXXXXXXX', 'B,C,D'),
        ...
        ...
        (5000, 'XXXXXXXXXX', 'A,B,C,D');
```

查询 SET 结果集，MySQL为SET配备了FIND_IN_SET函数

```
select * from QA where FIND_IN_SET('B',`answer`);
```

下面两种方法也能实现，但不推荐使用。

```
select question, answer from QA where locate('B',answer)>0;
select question, answer from QA where POSITION('B' in answer)>0;
```

查询多个答案

```
select question, answer from QA where answer = 'B,C';
```

## 2.2. 基于现有表结构创建新空表

```
CREATE TABLE `test`.`old_table` (
  `id` INT UNSIGNED NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(45) NULL,
  PRIMARY KEY (`id`));

CREATE TABLE new_table LIKE old_table;
```

```
mysql> show create table `test`.new_table;
+-----------+-----------------------------------------------------------------------------
-----------------------------------------------------------------------------------------
----------------------------------+
| Table     | Create Table
|
+-----------+-----------------------------------------------------------------------------
-----------------------------------------------------------------------------------------
----------------------------------+
| new_table | CREATE TABLE `new_table` (
  `id` int unsigned NOT NULL AUTO_INCREMENT,
  `name` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci |
+-----------+-----------------------------------------------------------------------------
-----------------------------------------------------------------------------------------
----------------------------------+
1 row in set (0.00 sec)
```

## 2.3. 基于已存在表创建新表

**create table ... select**

创建空表

```
create table admin_user_history select * from admin_user where 1 <> 1;
```

```

```

创建有数据的表

```
create table admin_user_history select * from admin_user;
```

## 2.4. 修改表

modifiy table

```
ALTER TABLE ecs_users add user_picture varchar(255);
```

## 2.5. 临时表

**TEMPORARY Table**

临时表将在你连接期间存在。一旦断开时将自动删除表并释放所用的空间。你在连接期间删除该表也同样释放空间。

```
CREATE TEMPORARY TABLE tmp_table (
        key VARCHAR(10) NOT NULL,
        value INTEGER NOT NULL
)
```

声明临时表是一个HEAP表，允许你指定在内存中创建它

```
CREATE TEMPORARY TABLE tmp_mem_table (
        key VARCHAR(10) NOT NULL,
        value INTEGER NOT NULL
) TYPE = HEAP
```

## 2.6. CHARACTER

```
ALTER TABLE <table_name> CONVERT TO CHARACTER SET utf8;
alter table <table_name> convert to charset utf8mb4;
```

修改表字符集

```
ALTER TABLE `tmp_cats`  COLLATE='utf8_general_ci',  CONVERT TO CHARSET utf8;
```

## 2.7. DEFAULT

### AUTO_INCREMENT

定义 AUTO_INCREMENT 起始值

```
CREATE TABLE `bank_account` (
        `id` INT(10) UNSIGNED NOT NULL AUTO_INCREMENT COMMENT '自增唯一ID',
        `name` VARCHAR(50) NOT NULL DEFAULT '0' COMMENT '帐号名称(Name on account)',
        PRIMARY KEY (`id`)
)
COMMENT='银行帐号'
COLLATE='utf8_general_ci'
ENGINE=InnoDB
AUTO_INCREMENT=2;
```

设置 AUTO_INCREMENT

```
ALTER TABLE `accounts`
        AUTO_INCREMENT=792257;
```

### TIMESTAMP NULL DEFAULT NULL ON UPDATE

```
alter table cms.article ADD  COLUMN `mtime` TIMESTAMP NULL DEFAULT NULL ON UPDATE
CURRENT_TIMESTAMP COMMENT '更改时间';
```

更新时间

`mtime` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP COMMENT '更改时间',

```
CREATE TABLE `bank_account` (
        `id` INT(10) UNSIGNED NOT NULL AUTO_INCREMENT COMMENT '自增唯一ID',
```

```
        `bank_name` VARCHAR(255) NOT NULL DEFAULT '0' COMMENT '银行名字(Bank Name)',
        `name` VARCHAR(50) NOT NULL DEFAULT '0' COMMENT '帐号名称(Name on account)',
        `account_number` VARCHAR(50) NOT NULL DEFAULT '0' COMMENT '银行帐号(Account
Number)',
        `branch_location` VARCHAR(255) NOT NULL DEFAULT '0' COMMENT '支行位置(Branch
Location)',
        `description` VARCHAR(255) NOT NULL DEFAULT '0' COMMENT '银行描述',
        `status` ENUM('Y','N') NOT NULL DEFAULT 'N' COMMENT '银行帐号状态',
        `ctime` DATETIME NOT NULL COMMENT '创建时间',
        `mtime` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
COMMENT '更改时间',
        PRIMARY KEY (`id`)
)
COMMENT='银行帐号'
COLLATE='utf8_general_ci'
ENGINE=InnoDB
AUTO_INCREMENT=2;
```

表存储位置(DATA DIRECTORY)

```
CREATE TABLE IF NOT EXISTS `tab_name` (
  `id` int(11) DEFAULT NULL,
  `purchased` date DEFAULT NULL,
  KEY `Index 1` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
/*!50100 PARTITION BY LIST (YEAR(purchased))
(PARTITION p0 VALUES IN (1990) DATA DIRECTORY = '/www/data' ENGINE = InnoDB) */;
```

## 2.8. KEY

### PRIMARY KEY

一般主键定义

```
PRIMARY KEY (`id`),
```

复合主键

```
PRIMARY KEY (`id`, `user_id`),
```

## 2.9. AUTO_INCREMENT 定义初始值

```
DROP TABLE IF EXISTS users;
CREATE TABLE user(
    id          INT   NOT NULL AUTO_INCREMENT
PRIMARY KEY(id)
)ENGINE=InnoDB AUTO_INCREMENT=100 DEFAULT CHARSET=utf8;
```

修改 auto_incremenrt 起始值

```
alter table tab1 auto_incremenrt=n
```

## 2.10. COMMENT

```
ALTER TABLE `neo`.`stuff` COMMENT = '用户表' ;
ALTER TABLE `neo`.`stuff` CHANGE COLUMN `name` `name` VARCHAR(50) NULL DEFAULT NULL
COMMENT '姓名'  ;
ALTER TABLE `neo`.`stuff` CHANGE COLUMN `password` `password` VARCHAR(50) NULL DEFAULT
NULL COMMENT '用户密码' ;


CREATE TABLE `stuff` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(50) DEFAULT NULL COMMENT ''姓名'',
  `password` varchar(50) DEFAULT NULL COMMENT ''用户密码'',
  `created` date NOT NULL DEFAULT ''0000-00-00'',
  PRIMARY KEY (`id`,`created`)
) ENGINE=MyISAM AUTO_INCREMENT=5 DEFAULT CHARSET=latin1 COMMENT=''用户表''
/*!50100 PARTITION BY HASH (year(created))
PARTITIONS 10 */
```

## 2.11. 修改表名

```
ALTER TABLE old_table_name  RENAME TO new_table_name;
RENAME old_table _name TO new_table_name;
```

## 2.12. Engine 存储引擎

显示当前数据库支持引擎

```
mysql> show engines;
+--------------------+---------+----------------------------------------------------------
```

```
--------+--------------+------+------------+
| Engine             | Support | Comment
| Transactions | XA   | Savepoints |
+--------------------+---------+-----------------------------------------------------------
--------+--------------+------+------------+
| FEDERATED          | NO      | Federated MySQL storage engine
| NULL         | NULL | NULL       |
| MRG_MYISAM         | YES     | Collection of identical MyISAM tables
| NO           | NO   | NO         |
| MyISAM             | YES     | MyISAM storage engine
| NO           | NO   | NO         |
| BLACKHOLE          | YES     | /dev/null storage engine (anything you write to it
disappears) | NO           | NO   | NO         |
| MEMORY             | YES     | Hash based, stored in memory, useful for temporary
tables      | NO           | NO   | NO         |
| CSV                | YES     | CSV storage engine
| NO           | NO   | NO         |
| ARCHIVE            | YES     | Archive storage engine
| NO           | NO   | NO         |
| PERFORMANCE_SCHEMA | YES     | Performance Schema
| NO           | NO   | NO         |
| InnoDB             | DEFAULT | Supports transactions, row-level locking, and foreign
keys     | YES          | YES  | YES        |
+--------------------+---------+-----------------------------------------------------------
--------+--------------+------+------------+
9 rows in set (0.00 sec)
```

### 切换引擎

修改与切换引擎

```
ALTER TABLE `test` ENGINE=BLACKHOLE;
ALTER TABLE `test` ENGINE=InnoDB;
```

### FEDERATED

启用 FEDERATED 引擎, 服务器环境 Ubuntu 13.04

```
$ sudo vim /etc/mysql/conf.d/federated.cnf
[mysqld]
federated

$ sudo service mysql restart
```

```
mysql> show engines;
+--------------------+---------+-----------------------------------------------------------
--------+--------------+------+------------+
| Engine             | Support | Comment
| Transactions | XA   | Savepoints |
```

```
+--------------------+---------+----------------------------------------------------------
--------+--------------+------+------------+
| FEDERATED          | YES     | Federated MySQL storage engine
| NO           | NO   | NO         |
| MRG_MYISAM         | YES     | Collection of identical MyISAM tables
| NO           | NO   | NO         |
| MyISAM             | YES     | MyISAM storage engine
| NO           | NO   | NO         |
| BLACKHOLE          | YES     | /dev/null storage engine (anything you write to it
disappears) | NO           | NO   | NO         |
| MEMORY             | YES     | Hash based, stored in memory, useful for temporary
tables      | NO           | NO   | NO         |
| CSV                | YES     | CSV storage engine
| NO           | NO   | NO         |
| ARCHIVE            | YES     | Archive storage engine
| NO           | NO   | NO         |
| PERFORMANCE_SCHEMA | YES     | Performance Schema
| NO           | NO   | NO         |
| InnoDB             | DEFAULT | Supports transactions, row-level locking, and foreign
keys     | YES          | YES  | YES        |
+--------------------+---------+----------------------------------------------------------
--------+--------------+------+------------+
9 rows in set (0.00 sec)
```

A 服务器

```
CREATE TABLE `t1` (
        `id` INT(10) UNSIGNED NOT NULL AUTO_INCREMENT,
        `name` VARCHAR(50) NOT NULL,
        `sex` ENUM('Y','N') NULL DEFAULT 'Y',
        `passwd` VARCHAR(50) NULL DEFAULT NULL,
        `ctime` TIMESTAMP NOT NULL DEFAULT '0000-00-00 00:00:00',
        `mtime` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
        PRIMARY KEY (`id`)
)
COLLATE='utf8_general_ci'
ENGINE=InnoDB
AUTO_INCREMENT=4;
```

B 服务器

```
DROP TABLE `users`;

CREATE TABLE `users` (
        `id` INT(10) UNSIGNED NOT NULL AUTO_INCREMENT,
        `name` VARCHAR(50) NOT NULL,
        `sex` ENUM('Y','N') NULL DEFAULT 'Y',
        `ctime` TIMESTAMP NOT NULL DEFAULT '0000-00-00 00:00:00',
        `mtime` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
        PRIMARY KEY (`id`)
) ENGINE=FEDERATED connection = 'mysql://www:qwer123@192.168.2.1:3306/test/t1';
```

上面字段描述是你需要的字段，并非所有字段。这里屏蔽了passwd字段

**提示**

```
connection = 'mysql://用户名:密码@主机:端口/数据库/表名'
```

```
mysql> DROP TABLE `users`;
Query OK, 0 rows affected (0.00 sec)

mysql> CREATE TABLE `users` (
    -> `id` INT(10) UNSIGNED NOT NULL AUTO_INCREMENT,
    -> `name` VARCHAR(50) NOT NULL,
    -> `sex` ENUM('Y','N') NULL DEFAULT 'Y',
    -> `ctime` TIMESTAMP NOT NULL DEFAULT '0000-00-00 00:00:00',
    -> `mtime` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
    -> PRIMARY KEY (`id`)
    -> ) ENGINE=FEDERATED connection = 'mysql://www:qwer123@192.168.2.1:3306/test/t1';
Query OK, 0 rows affected (0.06 sec)

mysql>
mysql> show tables;
+----------------+
| Tables_in_test |
+----------------+
| users          |
+----------------+
1 row in set (0.00 sec)

mysql> desc users;
+-------+------------------+------+-----+---------------------+-------------------------
----+
| Field | Type             | Null | Key | Default             | Extra
|
+-------+------------------+------+-----+---------------------+-------------------------
----+
| id    | int(10) unsigned | NO   | PRI | NULL                | auto_increment
|
| name  | varchar(50)      | NO   |     | NULL                |
|
| sex   | enum('Y','N')    | YES  |     | Y                   |
|
| ctime | timestamp        | NO   |     | 0000-00-00 00:00:00 |
|
| mtime | timestamp        | NO   |     | CURRENT_TIMESTAMP   | on update
CURRENT_TIMESTAMP |
+-------+------------------+------+-----+---------------------+-------------------------
----+
5 rows in set (0.00 sec)

mysql> select * from users;
+----+------+------+---------------------+---------------------+
| id | name | sex  | ctime               | mtime               |
+----+------+------+---------------------+---------------------+
|  1 | neo  | Y    | 0000-00-00 00:00:00 | 2013-05-17 18:05:09 |
|  2 | zen  | Y    | 0000-00-00 00:00:00 | 2013-05-17 18:05:11 |
|  3 | lily | N    | 0000-00-00 00:00:00 | 2013-05-17 18:05:22 |
+----+------+------+---------------------+---------------------+
3 rows in set (0.01 sec)
```

**FEDERATED 与 mysqldump 问题!**

切记，mysqldump 只会dump出使用FEDERATED引擎表的结构,不会包含数据。

### BLACKHOLE

```
CREATE TABLE test(id INT, val CHAR(10)) ENGINE = BLACKHOLE;
```

### ARCHIVE

归档(是适用于存放大量数据的存储引擎), 仅支持select、insert操作; 不支持delete 、update、索引等操作；使用zlib无损算法压缩数据，节省磁盘空间；

适用场景：适用于大量可查询但不能删除的历史数据保存；

基于 order 表创建 order_audit 归档表

```
create table order_audit engine=archive as select * from `order`;
```

order_audit 表结构如下

```
CREATE TABLE `order_audit` (
  `id` int(10) unsigned NOT NULL DEFAULT '0' COMMENT '订单ID',
  `name` varchar(45) NOT NULL COMMENT '订单名称',
  `price` float NOT NULL COMMENT '价格',
  `ctime` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT '创建时间'
) ENGINE=ARCHIVE DEFAULT CHARSET=utf8
```

```
mysql> show table status like 'order_audit';
+-------------+---------+---------+------------+------+----------------+-------------
+-----------------+--------------+-----------+----------------+-------------+---------
------------+------------+-----------------+----------+----------------+---------+
| Name        | Engine  | Version | Row_format | Rows | Avg_row_length | Data_length |
Max_data_length | Index_length | Data_free | Auto_increment | Create_time |
Update_time         | Check_time | Collation       | Checksum | Create_options |
Comment |
+-------------+---------+---------+------------+------+----------------+-------------
+-----------------+--------------+-----------+----------------+-------------+---------
------------+------------+-----------------+----------+----------------+---------+
| order_audit | ARCHIVE |      10 | Compressed |    4 |           2215 |        8861 |
0 |            0 |         0 |           NULL | NULL        | 2017-11-16 17:30:34 |
NULL       | utf8_general_ci |     NULL |                |          |
+-------------+---------+---------+------------+------+----------------+-------------
```

```
+-----------------+--------------+-----------+----------------+-------------+---------
-----------+------------+-----------------+----------+----------------+---------+
1 row in set (0.01 sec)
```

**CSV**

创建表

```
CREATE TABLE `csv_table` (
  `id` int(11) NOT NULL,
  `name` varchar(45) NOT NULL,
  `age` int(11) NOT NULL
) ENGINE=CSV DEFAULT CHARSET=utf8
```

查看表状态

```
mysql> show table status like 'csv_table';
+-----------+--------+---------+------------+------+----------------+-------------+---
--------------+--------------+-----------+----------------+-------------+-------------
+------------+-----------------+----------+----------------+---------+
| Name      | Engine | Version | Row_format | Rows | Avg_row_length | Data_length |
Max_data_length | Index_length | Data_free | Auto_increment | Create_time |
Update_time | Check_time | Collation       | Checksum | Create_options | Comment |
+-----------+--------+---------+------------+------+----------------+-------------+---
--------------+--------------+-----------+----------------+-------------+-------------
+------------+-----------------+----------+----------------+---------+
| csv_table | CSV    |      10 | Dynamic    |    2 |              0 |           0 |
0 |            0 |         0 |           NULL | NULL        | NULL        | NULL
| utf8_general_ci |     NULL |                |         |
+-----------+--------+---------+------------+------+----------------+-------------+---
--------------+--------------+-----------+----------------+-------------+-------------
+------------+-----------------+----------+----------------+---------+
1 row in set (0.00 sec)
```

插入数据

```
insert into csv_table values (1,'Neo',37),(2,'Jam',40);
```

查看数据

```
```

```
mysql> SELECT * FROM test.csv_table;
+----+------+-----+
| id | name | age |
+----+------+-----+
|  1 | Neo  |  37 |
|  2 | Jam  |  40 |
+----+------+-----+
2 rows in set (0.00 sec)
```

CSV 引擎是可以直接将csv文件复制出来的，表存储在 /var/lib/mysql/ 目录

```
root@netkiller /etc/nginx/conf.d % ls -1 /var/lib/mysql/test/csv*
/var/lib/mysql/test/csv_table.CSM
/var/lib/mysql/test/csv_table.CSV
/var/lib/mysql/test/csv_table.frm
```

.*CSM, *.frm 是表结构文件，*.CSV 是我们需要的文件，纯文本，可以使用Excel打开。

```
root@netkiller /etc/nginx/conf.d % cat /var/lib/mysql/test/csv_table.CSV
1,"Neo",37
2,"Jam",40
```

# 3. Partitioning

```
mysql> SHOW VARIABLES LIKE '%partition%';

+-------------------+-------+
| Variable_name     | Value |
+-------------------+-------+
| have_partitioning | YES   |
+-------------------+-------+
1 row in set (0.00 sec)
```

## 3.1. RANGE

```
18.5.1. Partitioning Keys, Primary Keys, and Unique Keys
This section discusses the relationship of partitioning keys
with primary keys and unique keys. The rule governing this
relationship can be expressed as follows: All columns used in
the partitioning expression for a partitioned table must be
part of every unique key that the table may have.

In other words, every unique key on the table must use every
column in the table's partitioning expression. (This also
includes the table's primary key, since it is by definition a
unique key. This particular case is discussed later in this
section.) For example, each of the following table creation
statements is invalid:


SQL code:
mysql> create table tx (
    ->     id int not null ,
    ->     info_time date,
    ->     primary key(id,info_time)
    -> )
    -> PARTITION BY RANGE(info_time div 100)
    -> (
    ->     PARTITION p_2008_11 VALUES LESS THAN (200812),
```

```
    ->     PARTITION p_2008_12 VALUES LESS THAN (200901),
    ->     PARTITION p_2009_01 VALUES LESS THAN (200902),
    ->     PARTITION p_2009_02 VALUES LESS THAN (200903),
    ->     PARTITION p_2009_03 VALUES LESS THAN (200904),
    ->     PARTITION p_2009_04 VALUES LESS THAN (200905),
    ->     PARTITION p_catch_all VALUES LESS THAN MAXVALUE
    -> );
Query OK, 0 rows affected (0.17 sec)

mysql>
```

```
CREATE TABLE t1 (
    year_col  INT,
    some_data INT
)
PARTITION BY RANGE (year_col) (
    PARTITION p0 VALUES LESS THAN (1991),
    PARTITION p1 VALUES LESS THAN (1995),
    PARTITION p2 VALUES LESS THAN (1999),
    PARTITION p3 VALUES LESS THAN (2002),
    PARTITION p4 VALUES LESS THAN (2006),
    PARTITION p5 VALUES LESS THAN MAXVALUE
);
```

e.g.2

```
CREATE TABLE rc (
    a INT NOT NULL,
    b INT NOT NULL
)
PARTITION BY RANGE COLUMNS(a,b) (
    PARTITION p0 VALUES LESS THAN (10,5),
    PARTITION p1 VALUES LESS THAN (20,10),
    PARTITION p2 VALUES LESS THAN (MAXVALUE,15),
    PARTITION p3 VALUES LESS THAN (MAXVALUE,MAXVALUE)
);
```

```
CREATE TABLE part_tab
(
        c1 int default NULL,
        c2 varchar(30) default NULL,
        c3 date default NULL

) engine=myisam
PARTITION BY RANGE (year(c3)) (
      PARTITION p0 VALUES LESS THAN (2000) ,
      PARTITION p1 VALUES LESS THAN (2001) ,
      PARTITION p2 VALUES LESS THAN (2002) ,
      PARTITION p3 VALUES LESS THAN (2003) ,
      PARTITION p4 VALUES LESS THAN (2004) ,
      PARTITION p12 VALUES LESS THAN (2012),
      PARTITION p13 VALUES LESS THAN MAXVALUE
);
```

## 3.2. LIST

```
CREATE TABLE client_firms (
    id   INT,
    name VARCHAR(35)
)
PARTITION BY LIST (id) (
    PARTITION r0 VALUES IN (1, 5, 9, 13, 17, 21),
    PARTITION r1 VALUES IN (2, 6, 10, 14, 18, 22),
    PARTITION r2 VALUES IN (3, 7, 11, 15, 19, 23),
    PARTITION r3 VALUES IN (4, 8, 12, 16, 20, 24)
);
```

```
CREATE TABLE lc (
    a INT NULL,
    b INT NULL
)
PARTITION BY LIST COLUMNS(a,b) (
    PARTITION p0 VALUES IN( (0,0), (NULL,NULL) ),
    PARTITION p1 VALUES IN( (0,1), (0,2), (0,3), (1,1), (1,2)
),
    PARTITION p2 VALUES IN( (1,0), (2,0), (2,1), (3,0), (3,1)
),
    PARTITION p3 VALUES IN( (1,3), (2,2), (2,3), (3,2), (3,3)
)
);
```

```
CREATE TABLE th (id INT, name VARCHAR(30), adate DATE)
PARTITION BY LIST(YEAR(adate))
(
  PARTITION p1999 VALUES IN (1995, 1999, 2003)
    DATA DIRECTORY = '/var/appdata/95/data'
    INDEX DIRECTORY = '/var/appdata/95/idx',
  PARTITION p2000 VALUES IN (1996, 2000, 2004)
    DATA DIRECTORY = '/var/appdata/96/data'
    INDEX DIRECTORY = '/var/appdata/96/idx',
  PARTITION p2001 VALUES IN (1997, 2001, 2005)
    DATA DIRECTORY = '/var/appdata/97/data'
    INDEX DIRECTORY = '/var/appdata/97/idx',
  PARTITION p2000 VALUES IN (1998, 2002, 2006)
    DATA DIRECTORY = '/var/appdata/98/data'
    INDEX DIRECTORY = '/var/appdata/98/idx'
);
```

## 3.3. HASH

```
CREATE TABLE `test` (
  `userid` int(10) unsigned NOT NULL auto_increment,
  `username` int(10) unsigned NOT NULL DEFAULT '0',
  `password` int(10) unsigned NOT NULL DEFAULT '0',

  primary key (`userid`),
  KEY `userid` (`username`)
) ENGINE=InnoDB
PARTITION BY HASH(userid)
PARTITIONS 8;
```

使用HASH (year(created)) 替代 RANGE(year(created))

```
CREATE TABLE stuff (
        id INT AUTO_INCREMENT,
        name varchar(50),
        password varchar(50),
        created DATE,
        PRIMARY KEY (id, created)
)
PARTITION BY RANGE(year(created)) (
        PARTITION p0 VALUES LESS THAN (2010),
        PARTITION p1 VALUES LESS THAN (2012),
        PARTITION p2 VALUES LESS THAN MAXVALUE
);
```

更好的方法

```
CREATE TABLE stuff (
        id INT AUTO_INCREMENT,
        name varchar(50),
        password varchar(50),
        created DATE,
        PRIMARY KEY (id, created)
)
PARTITION BY HASH (year(created)) PARTITIONS 10;
```

我们演示一下

```
mysql> CREATE TABLE stuff (
    -> id INT AUTO_INCREMENT,
    -> name varchar(50),
    -> password varchar(50),
    -> created DATE,
    -> PRIMARY KEY (id, created)
    -> )
    -> PARTITION BY HASH (year(created)) PARTITIONS 10;
Query OK, 0 rows affected (0.08 sec)


mysql> insert into stuff (name,password,created)
values('neo','test','2010-10-1');
Query OK, 1 row affected (0.06 sec)

mysql> insert into stuff (name,password,created)
values('neo1','test','2012-2-1');
Query OK, 1 row affected (0.00 sec)

mysql> insert into stuff (name,password,created)
values('neo2','test','2012-3-5');
Query OK, 1 row affected (0.00 sec)

mysql> insert into stuff (name,password,created)
values('neo4','test','2011-1-5');
Query OK, 1 row affected (0.00 sec)

mysql> SELECT
    ->   partition_name part,
    ->   partition_expression expr,
    ->   partition_description descr,
    ->   table_rows
    -> FROM
    ->   INFORMATION_SCHEMA.partitions
    -> WHERE
    ->   TABLE_SCHEMA = schema()
    ->   AND TABLE_NAME='stuff';
+------+--------------+-------+------------+
| part | expr         | descr | table_rows |
+------+--------------+-------+------------+
| p0   | year(created) | NULL  |          1 |
| p1   | year(created) | NULL  |          1 |
| p2   | year(created) | NULL  |          2 |
```

```
| p3   | year(created) | NULL  |          0 |
| p4   | year(created) | NULL  |          0 |
| p5   | year(created) | NULL  |          0 |
| p6   | year(created) | NULL  |          0 |
| p7   | year(created) | NULL  |          0 |
| p8   | year(created) | NULL  |          0 |
| p9   | year(created) | NULL  |          0 |
+------+---------------+-------+------------+
10 rows in set (0.02 sec)

mysql> EXPLAIN PARTITIONS SELECT * FROM stuff WHERE
created='2011-01-05'\G
*************************** 1. row
***************************
           id: 1
  select_type: SIMPLE
        table: stuff
   partitions: p1
         type: system
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 1
        Extra:
1 row in set (0.08 sec)

mysql> EXPLAIN PARTITIONS SELECT * FROM stuff WHERE
created='2012-03-05'\G
*************************** 1. row
***************************
           id: 1
  select_type: SIMPLE
        table: stuff
   partitions: p2
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 2
        Extra: Using where
1 row in set (0.00 sec)
```

```

```

**LINEAR HASH**

```
CREATE TABLE employees (
    id INT NOT NULL,
    fname VARCHAR(30),
    lname VARCHAR(30),
    hired DATE NOT NULL DEFAULT '1970-01-01',
    separated DATE NOT NULL DEFAULT '9999-12-31',
    job_code INT,
    store_id INT
)
PARTITION BY LINEAR HASH( YEAR(hired) )
PARTITIONS 4;
```

## 3.4. KEY分区

按照KEY进行分区类似于按照HASH分区，除了HASH分区使用的用户定义的表达式，而KEY分区的 哈希函数是由MySQL 服务器提供。MySQL 簇（Cluster）使用函数MD5()来实现KEY分区；

```
CREATE TABLE tk (
    col1 INT NOT NULL,
    col2 CHAR(5),
    col3 DATE
)
PARTITION BY LINEAR KEY (col1)
PARTITIONS 3;
```

## 3.5. Subpartitioning

```
CREATE TABLE ts (id INT, purchased DATE)
```

```
    PARTITION BY RANGE( YEAR(purchased) )
    SUBPARTITION BY HASH( TO_DAYS(purchased) )
    SUBPARTITIONS 2 (
        PARTITION p0 VALUES LESS THAN (1990),
        PARTITION p1 VALUES LESS THAN (2000),
        PARTITION p2 VALUES LESS THAN MAXVALUE
    );

CREATE TABLE ts1 (id INT, purchased DATE)
    PARTITION BY RANGE( YEAR(purchased) )
    SUBPARTITION BY HASH( MONTH(purchased) )
    SUBPARTITIONS 2 (
        PARTITION p0 VALUES LESS THAN (1990),
        PARTITION p1 VALUES LESS THAN (2000),
        PARTITION p2 VALUES LESS THAN MAXVALUE
    );
```

## 3.6. 分区管理

新增分区

mysql 5.5+

为已经存在表添加分区

```
ALTER TABLE tbl_name  ADD PARTITION PARTITIONS 6;
```

新增 RANGE分区

```
ALTER TABLE category ADD PARTITION (PARTITION p4 VALUES IN
(100,200,300,400)
                    DATA DIRECTORY = '/data/category'
                    INDEX DIRECTORY = '/data/category');
```

新增 LIST分区

```
CREATE TABLE expenses (
  expense_date DATE NOT NULL,
  category VARCHAR(30),
  amount DECIMAL (10,3)
);

ALTER TABLE expenses
PARTITION BY LIST COLUMNS (category)
(
  PARTITION p01 VALUES IN ( 'lodging', 'food'),
  PARTITION p02 VALUES IN ( 'flights', 'ground
transportation'),
  PARTITION p03 VALUES IN ( 'leisure', 'customer
entertainment'),
  PARTITION p04 VALUES IN ( 'communications'),
  PARTITION p05 VALUES IN ( 'fees')
);
```

新增 HASH分区

```
CREATE TABLE t1 (
    id INT,
    year_col INT
);

ALTER TABLE t1
    PARTITION BY HASH(id)
    PARTITIONS 8;
```

```
/* 在MySQL 5.1中*/
CREATE TABLE t2
(
  dt DATE
)
```

```
PARTITION BY RANGE (TO_DAYS(dt))
(
  PARTITION p01 VALUES LESS THAN (TO_DAYS('2007-01-01')),
  PARTITION p02 VALUES LESS THAN (TO_DAYS('2008-01-01')),
  PARTITION p03 VALUES LESS THAN (TO_DAYS('2009-01-01')),
  PARTITION p04 VALUES LESS THAN (MAXVALUE));

SHOW CREATE TABLE t2 \G
*************************** 1. row
***************************
       Table: t2
Create Table: CREATE TABLE `t2` (
  `dt` date DEFAULT NULL
) ENGINE=MyISAM DEFAULT CHARSET=latin1
/*!50100 PARTITION BY RANGE (TO_DAYS(dt))
(PARTITION p01 VALUES LESS THAN (733042) ENGINE = MyISAM,
 PARTITION p02 VALUES LESS THAN (733407) ENGINE = MyISAM,
 PARTITION p03 VALUES LESS THAN (733773) ENGINE = MyISAM,
 PARTITION p04 VALUES LESS THAN MAXVALUE ENGINE = MyISAM) */



 /*在MySQL 5.5中*/
CREATE TABLE t2
(
  dt DATE
)
PARTITION BY RANGE COLUMNS (dt)
(
  PARTITION p01 VALUES LESS THAN ('2007-01-01'),
  PARTITION p02 VALUES LESS THAN ('2008-01-01'),
  PARTITION p03 VALUES LESS THAN ('2009-01-01'),
  PARTITION p04 VALUES LESS THAN (MAXVALUE));

SHOW CREATE TABLE t2 \G
*************************** 1. row
***************************
       Table: t2
Create Table: CREATE TABLE `t2` (
  `dt` date DEFAULT NULL
) ENGINE=MyISAM DEFAULT CHARSET=latin1
/*!50500 PARTITION BY RANGE  COLUMNS(dt)
(PARTITION p01 VALUES LESS THAN ('2007-01-01') ENGINE =
MyISAM,
 PARTITION p02 VALUES LESS THAN ('2008-01-01') ENGINE =
```

```
MyISAM,
 PARTITION p03 VALUES LESS THAN ('2009-01-01') ENGINE =
MyISAM,
 PARTITION p04 VALUES LESS THAN (MAXVALUE) ENGINE = MyISAM)
*/
```

## 删除分区

删除分区 p0

```
ALERT TABLE users DROP PARTITION p0;
```

## 重建分区

使用 REORGANIZE 重建分区。

```
RANGE 分区重建
ALTER TABLE users REORGANIZE PARTITION p0,p1 INTO (PARTITION
p0 VALUES LESS THAN (6000000));

将原来的 p0,p1 分区合并起来，放到新的 p0 分区中。

LIST 分区重建
ALTER TABLE users REORGANIZE PARTITION p0,p1 INTO (PARTITION
p0 VALUES IN(0,1,4,5,8,9,12,13));
将原来的 p0,p1 分区合并起来，放到新的 p0 分区中。

HASH/KEY 分区重建
ALTER TABLE users REORGANIZE PARTITION COALESCE PARTITION 2;
分区的数量改为2，
注意：在这里数量只能减少不能增加。想要增加可以用 ADD PARTITION 方法
```

调整HASH/KEY分区数量，将分区总数扩展到8个。

```
ALTER TABLE users ADD PARTITION PARTITIONS 8;
```

## 分区维护

重建分区：这和先删除保存在分区中的所有记录，然后重新插入它们，具有同样的效果。它可用于整理分区碎片。

示例：

```
ALTER TABLE t1 REBUILD PARTITION (p0, p1);
```

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

示例：

```
ALTER TABLE t1 OPTIMIZE PARTITION (p0, p1);
```

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

· 分析分区：读取并保存分区的键分布。

示例：

```
ALTER TABLE t1 ANALYZE PARTITION (p3);
```

· 修补分区： 修补被破坏的分区。

示例：

```
ALTER TABLE t1 REPAIR PARTITION (p0,p1);
```

· 检查分区： 可以使用几乎与对非分区表使用CHECK TABLE 相同的方式检查分区。

示例：

```
ALTER TABLE trb3 CHECK PARTITION (p1);
```

## 3.7. EXPLAIN PARTITIONS

EXPLAIN PARTITIONS

```
mysql> EXPLAIN PARTITIONS SELECT * FROM users\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: users
   partitions: p0,p1,p2,p3,p4,p5,p6
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 7
        Extra:
1 row in set (0.03 sec)

mysql> EXPLAIN PARTITIONS SELECT * FROM users WHERE id < 5\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: users
   partitions: p0,p1,p2,p3,p4,p5,p6
         type: range
possible_keys: PRIMARY
          key: PRIMARY
      key_len: 4
          ref: NULL
         rows: 7
        Extra: Using where
1 row in set (0.00 sec)
```

## 3.8. SHOW CREATE TABLE

SHOW CREATE TABLE

```
mysql> SHOW CREATE TABLE users\G
*************************** 1. row ***************************
       Table: users
Create Table: CREATE TABLE `users` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `username` varchar(20) NOT NULL DEFAULT '',
  `birthday` datetime DEFAULT NULL,
  PRIMARY KEY (`id`,`username`)
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=latin1
/*!50100 PARTITION BY KEY (id,username)
PARTITIONS 7 */
1 row in set (0.00 sec)
```

## 3.9. INFORMATION_SCHEMA.partitions 表

```
SELECT
  partition_name part,
  partition_expression expr,
  partition_description descr,
  table_rows
FROM
  INFORMATION_SCHEMA.partitions
WHERE
  TABLE_SCHEMA = schema()
  AND TABLE_NAME='employees';
```

```
select
  partition_name part,
  partition_expression expr,
  from_seconds(partition_description) descr,
  table_rows
FROM
```

```
INFORMATION_SCHEMA.partitions
WHERE
    TABLE_SCHEMA = 'test'
    AND TABLE_NAME='t2';
```

## 3.10. 分区数据操作

指定分区查询

```
SELECT * FROM employees PARTITION (p0, p2);

SELECT count(1) FROM employees PARTITION (p0);
SELECT count(1) FROM employees PARTITION (p0, p2);
SELECT count(1) FROM employees PARTITION (p0, p2, p1);
```

删除分区中的记录

```
DELETE FROM employees PARTITION (p0, p1);
```

更新指定分区

```
UPDATE employees PARTITION (p0) SET store_id = 2 WHERE fname
= 'Jill';
```

指定分区连表查询

```
SELECT e.id, s.city FROM employees AS e JOIN stores PARTITION
(p1) AS s ...;
```

将某个表迁移到分区上

```
ALTER TABLE employees EXCHANGE PARTITION p0 WITH TABLE
employees2;
```

# 4. Index

## 4.1. SHOW INDEX

```
SHOW INDEX FROM tbl_name
```

垂直显示

```
SHOW INDEX FROM tbl_name\G
```

## 4.2. CREATE INDEX

```
CREATE INDEX index_name
ON table_name (column_name)
```

CREATE UNIQUE INDEX

```
CREATE UNIQUE INDEX index_name
ON table_name (column_name)
```

## 4.3. DROP INDEX

```
DROP INDEX index_name ON tbl_name
```

## 4.4. rebuild

```
SHOW INDEX FROM tbl_name
alter index IND_PK rebuild;
```

# 5. 外键(Foreign Key)

ON DELETE, ON UPDATE 事件触发限制，可选参数： RESTRICT | CASCADE | SET NULL | NO ACTION

1. RESTRICT（限制外表中的外键改动）
2. CASCADE（跟随外键改动）
3. SET NULL（设空值）
4. SET DEFAULT（设默认值）
5. NO ACTION（无动作，默认的）

## 5.1. FOREIGN KEY (RESTRICT)

```
CREATE TABLE `bank_account_group_has_bank_account` (
        `id` INT(10) UNSIGNED NOT NULL AUTO_INCREMENT,
        `bank_account_group_id` INT(10) UNSIGNED NOT NULL
DEFAULT '0',
        `bank_account_id` INT(10) UNSIGNED NOT NULL DEFAULT
'0',
        PRIMARY KEY (`id`),
        INDEX
`FK_bank_account_group_has_bank_account_bank_account`
(`bank_account_id`),
        INDEX
`FK_bank_account_group_has_bank_account_bank_account_group`
(`bank_account_group_id`),
        CONSTRAINT
`FK_bank_account_group_has_bank_account_bank_account` FOREIGN
KEY (`bank_account_id`) REFERENCES `bank_account` (`id`),
        CONSTRAINT
`FK_bank_account_group_has_bank_account_bank_account_group`
FOREIGN KEY (`bank_account_group_id`) REFERENCES
`bank_account_group` (`id`)
)
```

```
COMMENT='bank_account_group 与 bank_account 的 N:M 关系'
COLLATE='utf8_general_ci'
ENGINE=InnoDB
AUTO_INCREMENT=35;
```

# 6. 视图(View)

```
CREATE VIEW view_name AS
SELECT column_name(s)
FROM table_name
WHERE condition
```

update view

```
SQL CREATE OR REPLACE VIEW Syntax
CREATE OR REPLACE VIEW view_name AS
SELECT column_name(s)
FROM table_name
WHERE condition
```

# 7. 存储过程(PROCEDURE)

## 7.1. 存储程序

存储过程没有返回数据，需使用call proc()调用

```
CREATE DEFINER=`neo`@`%` PROCEDURE `angelfund`(IN `puid`
VARCHAR(50), IN `ptime` DATETIME)
        LANGUAGE SQL
        NOT DETERMINISTIC
        CONTAINS SQL
        SQL SECURITY DEFINER
        COMMENT ''
BEGIN

        DECLARE fusername VARCHAR(16) DEFAULT NULL;
        DECLARE fname VARCHAR(16) DEFAULT NULL;
        DECLARE fmembers_date VARCHAR(20) DEFAULT NULL;

        SELECT username,name,FROM_UNIXTIME(createtime) INTO
fusername,fname,fmembers_date FROM members WHERE username =
puid;

        IF fusername IS NOT NULL THEN
                INSERT IGNORE INTO
angelfund(username,name,members_date,accounts_date,endtime,`st
atus`,op,operator,`description`)
value(fusername,fname,fmembers_date,ptime,DATE_ADD(ptime,
INTERVAL +1 MONTH),'N','N','computer','');
        END IF;

END
```

调用过程

```
call angelfund('100','2013-10-10 10:10:10');
```

## 7.2. EXECUTE 执行 SQL

在过程中运行SQL，下面的例子是文件导出的例子。

```
DROP procedure IF EXISTS `export_file`;

DELIMITER $$
CREATE DEFINER=`dba`@`%` PROCEDURE `export_file`(IN file_name
char(64), IN tabname char(64))
BEGIN
        set @sql = concat('SELECT * INTO OUTFILE
','"'/var/lib/mysql-files/",file_name,"'",' FROM ', tabname) ;
    -- select @sql;
        PREPARE stmt FROM @sql;
        EXECUTE stmt;
        Deallocate prepare stmt;
END$$

DELIMITER ;
```

call 存储过程

```
call test.export_file('test', 'mytable');
```

## 7.3. PREPARE 传递参数

```
mysql> PREPARE stmt1 FROM 'SELECT SQRT(POW(?,2) + POW(?,2)) AS
hypotenuse';
Query OK, 0 rows affected (0.00 sec)
Statement prepared
```

```
mysql> SET @a = 3;
Query OK, 0 rows affected (0.00 sec)

mysql> SET @b = 4;
Query OK, 0 rows affected (0.00 sec)

mysql> EXECUTE stmt1 USING @a, @b;
+------------+
| hypotenuse |
+------------+
|          5 |
+------------+
1 row in set (0.00 sec)

mysql> DEALLOCATE PREPARE stmt1;
Query OK, 0 rows affected (0.00 sec)

mysql>
```

## 7.4. 存储过程返回数据

```
USE `test`;
DROP procedure IF EXISTS `test`;

DELIMITER $$
USE `test`$$
CREATE DEFINER=`dba`@`%` PROCEDURE `test`(in a int, in b int
,out num int)
BEGIN

        set num = a + b;

END$$

DELIMITER ;
```

运行后返回结果 10

```
set @num = 0;
call test(3,7,@num);
select @num;
```

## 7.5. 结果集转JSON

```
USE `netkiller`;
DROP procedure IF EXISTS `table2json`;

DELIMITER $$
USE `netkiller`$$
CREATE DEFINER=`neo`@`%` PROCEDURE `table2json`(
IN `schema` VARCHAR(32),
IN `table` VARCHAR(32),
IN `id` VARCHAR(10),
OUT rev VARCHAR(1024)
)
BEGIN
        SET @column = NULL;
        SET @str = NULL;

        SELECT
    GROUP_CONCAT(fields) AS col INTO @column FROM (
                SELECT
                        CONCAT('"', COLUMN_NAME, '",',
COLUMN_NAME) AS fields
                FROM
                        INFORMATION_SCHEMA.Columns
                WHERE
                        table_name = `table`
                                AND table_schema = `schema`)
```

```
AS tmptable;


        -- SELECT @column;


        SET @sql = CONCAT('SELECT json_object(',@column, ' )
as json INTO @str FROM ', `table`,' where id = ', `id`);

        -- SELECT @sql;

        PREPARE stmt FROM @sql;
        EXECUTE stmt;
        Deallocate prepare stmt;

        set rev = @str;

END$$

DELIMITER ;
```

使用实例

```
set @rev = '0';
call netkiller.table2json('test', 'test', '1', @rev);
select @rev;
```

## 7.6. 例子·过程返回结果

```
USE `netkiller`;
DROP procedure IF EXISTS `trigger2json`;

DELIMITER $$
USE `netkiller`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `trigger2json`(
```

```
IN `schema` VARCHAR(32),
IN `table` VARCHAR(32),
OUT rev VARCHAR(1024)
)
BEGIN
        SET @column = NULL;
        SET @str = NULL;

        SELECT
    GROUP_CONCAT(fields) AS col
INTO @column FROM
    (SELECT
        CONCAT('"', COLUMN_NAME, '", NEW.', COLUMN_NAME) AS
fields
    FROM
        INFORMATION_SCHEMA.Columns
    WHERE
        table_name = `table`
            AND table_schema = `schema`) AS tmptable;


-- SELECT @column;


        SET @sql = CONCAT('SELECT json_object(',@column, ' )
as json INTO @str ');

        -- SELECT @sql;

        PREPARE stmt FROM @sql;
        EXECUTE stmt;
        Deallocate prepare stmt;

        set rev = @str;

END$$

DELIMITER ;
```

```
set @rev = '0';
call neo.trigger2json('gw', 'member', @rev);
select @rev;
```

# 8. 函数

函数会返回数据，调用函数使用 select fun(),不能使用call调用，否则提示

```
mysql> call myfun();
ERROR 1305 (42000): PROCEDURE test.myfun does not exist
```

下面做一个实验

```
CREATE TABLE `t` (
        `id` INT(11) UNSIGNED NOT NULL AUTO_INCREMENT,
        `n` INT(11) UNSIGNED NULL DEFAULT '0',
        PRIMARY KEY (`id`)
)
COLLATE='utf8_general_ci'
ENGINE=InnoDB
AUTO_INCREMENT=5;

CREATE DEFINER=`neo`@`%` FUNCTION `myfun`()
        RETURNS int(11)
        LANGUAGE SQL
        NOT DETERMINISTIC
        READS SQL DATA
        SQL SECURITY DEFINER
        COMMENT ''
BEGIN
        INSERT INTO t (n) VALUES(rand()*100);
        RETURN LAST_INSERT_ID();
END
```

```
mysql> select myfun();
+---------+
| myfun() |
+---------+
|       9 |
+---------+
1 row in set, 2 warnings (0.07 sec)
```

## 8.1. TIMESTAMP TO ISO8601

```
USE `netkiller`;
DROP function IF EXISTS `timestamp_to_iso8601`;

DELIMITER $$
USE `netkiller`$$
CREATE DEFINER=`neo`@`db.netkiller.cn` FUNCTION
`timestamp_to_iso8601`(dt timestamp) RETURNS varchar(24) CHARSET utf8
BEGIN

        RETURN DATE_FORMAT( CONVERT_TZ(dt, @@session.time_zone,
'+00:00')  ,'%Y-%m-%dT%T.000Z');

END$$

DELIMITER ;
```

调用函数

```
mysql> select timestamp_to_iso8601(current_timestamp()) as iso8601;
+--------------------------+
| iso8601                  |
+--------------------------+
| 2017-12-07T07:21:22.000Z |
+--------------------------+
1 row in set (0.00 sec)
```

# 9. 触发器(Trigger)

## 9.1. create trigger

### Update 更新出发

实现 history 历史表功能，BEFORE update 做到数据库更新自动备份

```
CREATE TABLE user_history SELECT * FROM user WHERE 1 <> 1

DELIMITER //
CREATE TRIGGER user_history BEFORE update ON user FOR EACH ROW
BEGIN
insert into user_history SELECT * FROM user WHERE id = OLD.id;
END; //
DELIMITER ;
```

判断某字段数据修改满足条件后出发。

```
CREATE DEFINER=`dba`@`%` TRIGGER
`cms`.`jc_content_BEFORE_UPDATE` BEFORE UPDATE ON `jc_content`
FOR EACH ROW
BEGIN
        IF NEW.status = '1' THEN
                insert into `neo`.elasticsearch_trash(id)
values(OLD.content_id);
        END IF;
    IF NEW.status = '2' THEN
                delete from `neo`.elasticsearch_trash where id
= OLD.content_id;
        END IF;
END
```

**Delete 删除出发**

```
CREATE DEFINER=`dba`@`%` TRIGGER
`cms`.`jc_content_BEFORE_DELETE` BEFORE DELETE ON `jc_content`
FOR EACH ROW
BEGIN
        insert into `neo`.elasticsearch_trash(id)
values(OLD.content_id);
END
```

**Insert 插入出发**

```
```

## 9.2. drop trigger

```
DROP TRIGGER admin_user_history;

DELIMITER //
CREATE TRIGGER admin_user_history BEFORE update ON admin_user
FOR EACH ROW
BEGIN
insert into admin_user_history SELECT * FROM admin_user WHERE
user_id = OLD.user_id;
END; //
DELIMITER;
```

## 9.3. show triggers

```
show triggers;
```

### SHOW CREATE TRIGGER

```
mysql> SHOW CREATE TRIGGER ins_sum\G
*************************** 1. row ***************************
               Trigger: ins_sum
              sql_mode:
STRICT_TRANS_TABLES,NO_ENGINE_SUBSTITUTION
SQL Original Statement: CREATE DEFINER=`me`@`localhost` TRIGGER
ins_sum
                        BEFORE INSERT ON account
                        FOR EACH ROW SET @sum = @sum +
NEW.amount
  character_set_client: utf8
  collation_connection: utf8_general_ci
    Database Collation: latin1_swedish_ci
                Created: 2013-07-09 10:39:34.96
```

## 9.4. EXAMPLE

### BEFORE/AFTER

### 例 4.1. BEFORE/AFTER

```
DROP TRIGGER MY_TEST_MONITOR;
DELIMITER //
CREATE TRIGGER MY_TEST_MONITOR BEFORE insert ON MY_TEST FOR
EACH ROW
BEGIN
        INSERT INTO MY_TEST_MONITOR SELECT * FROM MY_TEST WHERE
TICKET = NEW.TICKET;
END; //
DELIMITER;
```

```
DROP TRIGGER MY_TEST_MONITOR;
DELIMITER //
CREATE TRIGGER MY_TEST_MONITOR AFTER insert ON MY_TEST FOR EACH
ROW
BEGIN
        INSERT INTO MY_TEST_MONITOR SELECT * FROM MY_TEST WHERE
TICKET = NEW.TICKET;
END; //
DELIMITER;
```

通过触发器保护数据，防止重复插入数据

```
CREATE DEFINER=`neo`@`%` TRIGGER `members_before_insert` BEFORE
INSERT ON `members` FOR EACH ROW BEGIN
        IF new.username IS NOT NULL THEN
                IF exists(select m.username from members m
where m.username = new.username) THEN
                set new.username = '';
                END IF;
        END IF;
END
```

## UUID

### 例 4.2. uuid()

```
delimiter $$

CREATE TABLE `member` (
  `uuid` char(36) NOT NULL,
  `username` varchar(20) DEFAULT NULL,
  `password` varchar(32) DEFAULT NULL,
  PRIMARY KEY (`uuid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8$$

CREATE
DEFINER=`root`@`%`
```

```
TRIGGER `test`.`member_before_insert`
BEFORE INSERT ON `test`.`member`
FOR EACH ROW
SET new.uuid = uuid()
$$
```

## CALL PROCEDURE

```
CREATE DEFINER=`neo`@`%` TRIGGER `accounts_angelfund` AFTER
INSERT ON `accounts` FOR EACH ROW BEGIN

   IF new.paymode = 'angelfund' THEN
                call angelfund(new.name,new.ctime);
   END IF;

END

CREATE DEFINER=`neo`@`%` PROCEDURE `angelfund`(IN `puid`
VARCHAR(50), IN `ptime` DATETIME)
        LANGUAGE SQL
        NOT DETERMINISTIC
        CONTAINS SQL
        SQL SECURITY DEFINER
        COMMENT ''
BEGIN

        DECLARE fusername VARCHAR(16) DEFAULT NULL;
        DECLARE fchinese_name VARCHAR(16) DEFAULT NULL;
        DECLARE fmembers_date VARCHAR(20) DEFAULT NULL;

        SELECT username,chinese_name,FROM_UNIXTIME(createtime)
INTO fusername,fchinese_name,fmembers_date FROM members WHERE
username = puid;

        IF fusername IS NOT NULL THEN
                INSERT IGNORE INTO
angelfund(username,chinese_name,members_date,accounts_date,endt
ime,`status`,op,operator,`description`)
value(fusername,fchinese_name,fmembers_date,ptime,DATE_ADD(ptim
e, INTERVAL +1 MONTH),'N','N','computer','');
   END IF;
```

```
END
```

# 10. 事件调度器(EVENT)

查看当前系统的 event 状态 SHOW VARIABLES LIKE 'event_scheduler';

```
mysql> SHOW VARIABLES LIKE 'event_scheduler';
+-----------------+-------+
| Variable_name   | Value |
+-----------------+-------+
| event_scheduler | ON    |
+-----------------+-------+
1 row in set (0.00 sec)

mysql>
```

## 10.1. 启用 EVENT

```
set GLOBAL event_scheduler=ON;
```

my.cnf 配置

```
event_scheduler=on
```

查看状态

```
mysql> select @@GLOBAL.event_scheduler;
+--------------------------+
| @@GLOBAL.event_scheduler |
+--------------------------+
| ON                       |
+--------------------------+
1 row in set (0.00 sec)

mysql> SHOW VARIABLES LIKE 'event_scheduler';
+-----------------+-------+
| Variable_name   | Value |
+-----------------+-------+
| event_scheduler | ON    |
+-----------------+-------+
```

```
1 row in set (0.01 sec)
```

## 10.2. 创建 EVENT

```
DROP EVENT IF EXISTS `captcha`;
DELIMITER //
CREATE DEFINER=`neo`@`%` EVENT `captcha` ON SCHEDULE EVERY 5 MINUTE STARTS
'2013-07-08 16:27:03' ON COMPLETION PRESERVE ENABLE DO BEGIN
        delete from captcha where ctime < DATE_ADD(now(), INTERVAL -5 MINUTE);
END//
DELIMITER ;
```

## 10.3. 禁用/启用

```
ALTER EVENT captcha DISABLE;
```

```
ALTER EVENT captcha ENABLE;
```

## 10.4. 查看 events

```
mysql> show events;
+--------+-------------+---------+-----------+-----------+------------+--------
--------+----------------+---------------------+------+---------+------------+-
--------------------+----------------------+--------------------+
| Db     | Name        | Definer | Time zone | Type      | Execute at |
Interval value | Interval field | Starts              | Ends | Status  |
Originator | character_set_client | collation_connection | Database Collation |
+--------+-------------+---------+-----------+-----------+------------+--------
--------+----------------+---------------------+------+---------+------------+-
--------------------+----------------------+--------------------+
| netkiller | captcha     | neo@%   | SYSTEM    | RECURRING | NULL       | 5
| MINUTE         | 2013-07-08 16:27:03 | NULL | ENABLED |          1 | utf8
| utf8_general_ci      | utf8_general_ci    |
| netkiller | sms_ips_log | neo@%   | SYSTEM    | RECURRING | NULL       | '0
5'          | DAY_HOUR       | 2013-07-09 14:39:51 | NULL | ENABLED |
1 | utf8                 | utf8_general_ci      | utf8_general_ci    |
```

```
+--------+-------------+---------+-----------+-----------+------------+--------
--------+----------------+---------------------+------+---------+------------+-
--------------------+----------------------+--------------------+
2 rows in set (0.00 sec)

mysql> show events \G;
*************************** 1. row ***************************
                  Db: netkiller
                Name: captcha
             Definer: neo@%
           Time zone: SYSTEM
                Type: RECURRING
          Execute at: NULL
      Interval value: 5
      Interval field: MINUTE
              Starts: 2013-07-08 16:27:03
                Ends: NULL
              Status: ENABLED
          Originator: 1
character_set_client: utf8
collation_connection: utf8_general_ci
  Database Collation: utf8_general_ci
*************************** 2. row ***************************
                  Db: netkiller
                Name: sms_ips_log
             Definer: neo@%
           Time zone: SYSTEM
                Type: RECURRING
          Execute at: NULL
      Interval value: '0 5'
      Interval field: DAY_HOUR
              Starts: 2013-07-09 14:39:51
                Ends: NULL
              Status: ENABLED
          Originator: 1
character_set_client: utf8
collation_connection: utf8_general_ci
  Database Collation: utf8_general_ci
2 rows in set (0.00 sec)

ERROR:
No query specified
```

## 10.5. 删除 EVENT

```
DROP EVENT [IF EXISTS] event_name;
```

```
DROP EVENT event_delete;
```

## 10.6. EVENT 应用案例

### 实例·每月创建一个表

每月创建一张新表，适用于分表的场景

```
CREATE DEFINER=`neo`@`netkiller` EVENT `logging`
        ON SCHEDULE
                EVERY 1 MONTH STARTS '2017-12-11 15:51:00'
        ON COMPLETION PRESERVE
        ENABLE
        COMMENT '每月自动创建表'
DO BEGIN
        declare _table_date varchar(10);
        select date_format(date_add(curdate(),interval 1 month),'%Y%m') into
_table_date;
        call logging(_table_date);
END
```

```
CREATE DEFINER=`neo`@`netkiller` PROCEDURE `logging`(
        IN `table_date` VARCHAR(10)
)
LANGUAGE SQL
NOT DETERMINISTIC
CONTAINS SQL
SQL SECURITY DEFINER
COMMENT ''
BEGIN
        set @_table_name = CONCAT('log_',table_date);
        set @_create = "CREATE TABLE If Not Exists ";
        set @_param = "(
                        `id` INT(11) NOT NULL AUTO_INCREMENT,
                        `type` VARCHAR(255) NULL DEFAULT NULL COMMENT '日志类型
1：网站 2：IOS 3:Android',
                        `url` VARCHAR(640) NULL DEFAULT NULL COMMENT '用户访问
url',
                        `serverIp` VARCHAR(255) NULL DEFAULT NULL COMMENT '服务
器ip',
                        `bodyBytesSent` VARCHAR(255) NULL DEFAULT NULL,
                        `bytesSent` VARCHAR(255) NULL DEFAULT NULL COMMENT '参
数字节数',
                        `browser` VARCHAR(255) NULL DEFAULT NULL COMMENT '浏览器
信息',
                        `ctime` TIMESTAMP NULL DEFAULT CURRENT_TIMESTAMP,
                        `mtime` TIMESTAMP NULL DEFAULT NULL ON UPDATE
```

```
CURRENT_TIMESTAMP,
                        PRIMARY KEY (`id`),
                        INDEX `ctime` (`ctime`, `deviceType`,`isFirst`),
                        INDEX `userIp` (`userIp`),
                        INDEX `deviceId` (`deviceId`),
                        INDEX `account` (`account`)
                )
                COMMENT='APP 访问记录'
                COLLATE='utf8_general_ci'
                ENGINE=InnoDB
                ;";

        SET @sql = CONCAT(@_create,@_table_name,@_param);
        PREPARE stmt FROM @sql;
        EXECUTE stmt;
        Deallocate prepare stmt;
END
```

**案例·定时删除数据**

需求：数据需要删除100万条数据，直接删除会对系统造成压力，导致主库阻塞。

解决方案：分批删除，使用 event 定时执行删除 SQL 直到删完位置。

开启 EVENT

```
set global event_scheduler = on;
```

造数据

```
CREATE EVENT IF NOT EXISTS event_test
ON SCHEDULE EVERY 2 SECOND ON COMPLETION PRESERVE
DO insert into mytable(message,ctime) value(uuid(), now());
```

备份数据

```
create table new_table_2022_7_30 select * from old_table;
```

定时删除

```
DROP EVENT event_delete;
DELIMITER $$
CREATE EVENT IF NOT EXISTS event_delete
ON SCHEDULE EVERY 1 SECOND ON COMPLETION PRESERVE
DO BEGIN
        DECLARE num integer;

        SELECT  COUNT(*) INTO num FROM mytable;

        IF num > 0 THEN
                delete from mytable limit 1;
                insert into logs(ctime) values(now());
        END IF;

END$$
```

指定日期执行

```
每个月的一号凌晨1 点执行
CREATE EVENT EVENT2
ON SCHEDULE EVERY 1 MONTH STARTS DATE_ADD(DATE_ADD(DATE_SUB(CURDATE(),INTERVAL
DAY(CURDATE())-1 DAY), INTERVAL 1 MONTH),INTERVAL 1 HOUR)
ON COMPLETION PRESERVE ENABLE
DO
BEGIN
        CALL YOUR();
END

每个季度一号的凌晨2点执行
CREATE EVENT TOTAL_SEASON_EVENT
ON SCHEDULE EVERY 1 QUARTER STARTS DATE_ADD(DATE_ADD(DATE(
CONCAT(YEAR(CURDATE()),'-',ELT(QUARTER(CURDATE()),1,4,7,10),'-',1)),INTERVAL 1
QUARTER),INTERVAL 2 HOUR)
ON COMPLETION PRESERVE ENABLE
DO
BEGIN
CALL YOUR();
END

每年1月1号凌晨2点执行
CREATE EVENT TOTAL_YEAR_EVENT
ON SCHEDULE EVERY 1 YEAR STARTS DATE_ADD(DATE(CONCAT(YEAR(CURDATE()) + 1,'-
',1,'-',1)),INTERVAL 2 HOUR)
ON COMPLETION PRESERVE ENABLE
DO
BEGIN
```

```
CALL YOUR();
END
```

# 第 5 章 DML (Data Manipulation Language)

```
SELECT - retrieve data from the a database
INSERT - insert data into a table
UPDATE - updates existing data within a table
DELETE - deletes all records from a table, the space for the
records remain
CALL - call a PL/SQL or Java subprogram
EXPLAIN PLAN - explain access path to data
LOCK TABLE - control concurrency
```

## 1. INSERT

### 1.1. INSERT INTO ... SELECT

```
SET @OLDTMP_SQL_MODE=@@SQL_MODE, SQL_MODE='';
DELIMITER //
CREATE TRIGGER `members_mobile_insert` BEFORE INSERT ON
`members_mobile` FOR EACH ROW BEGIN
        insert into members_location(id,province,city) select
NEW.id,mobile_location.province,mobile_location.city from
mobile_location where mobile_location.id = md5(LEFT(NEW.number,
7));
END//
DELIMITER ;
SET SQL_MODE=@OLDTMP_SQL_MODE;
```

### 1.2. INSERT IGNORE

INSERT IGNORE 与INSERT INTO的区别就是INSERT IGNORE会忽略数据库中已经存在 的数据，如果数据库没有数据，就插入新的数

据，如果有数据的话就跳过这条数据。

```
insert ignore into table(name)  select  name from table2
```

## 1.3. INSERT...ON DUPLICATE KEY UPDATE

```
create table foo (id serial primary key, u int, unique key
(u));

insert into foo (u) values (10);
insert into foo (u) values (10) on duplicate key update u = 20;

mysql> select * from foo;
+----+------+
| id | u    |
+----+------+
|  1 |   20 |
+----+------+
```

```
DROP TRIGGER IF EXISTS `cms`.`jc_content_BEFORE_DELETE`;

DELIMITER $$
USE `cms`$$
CREATE DEFINER=`5kwords`@`%` TRIGGER `jc_content_BEFORE_DELETE`
BEFORE DELETE ON `jc_content`
FOR EACH ROW BEGIN

    insert into `cms`.elasticsearch_trash(id)
values(OLD.content_id) on duplicate key update ctime = now();
    insert into `cms`.trash(id,`type`, site_id)
values(OLD.content_id, "delete", OLD.site_id) on duplicate key
update `type`="delete", ctime = now();

END$$
DELIMITER ;
```

# 2. REPLACE

replace 类似 ON DUPLICATE KEY UPDATE，插入过程遇到已经存在的字段，会更新处理。

```
replace into (id) value('1')
```

# 3. DELETE

## 3.1. 删除重复数据

```
delete from member group by username having count(username) > 1
```

# 第 6 章 SQL Statement Syntax

***Structured Query Language***

## 1. DISTINCT

SELECT DISTINCT user.name FROM user

```
SELECT DISTINCT user.name FROM user
```

# 2. group by

统计重复的手机号吗

```
select * from (select count(mobile) as c, mobile from member
where length(mobile) >= 11 group by mobile) as m where m.c > 1;
```

# 3. HAVING

```
select * from accounts where paymode='alipay' group by name
having count(name) >1;
```

# 4. REGEXP

## 正则匹配

判断非数字字符

```
select '看89700' regexp '^[0-9]+$'
select '89看700' regexp '^[0-9]+$'
select '89700看' regexp '^[0-9]+$'
```

应用到实际工作中

```
select count(*) from accounts a where a.name != '' and not
a.name regexp '^[0-9]+$';
select count(*) from accounts a,members m where  a.member =
m.id  and a.name != '' and  not a.name regexp '^[0-9]+$'
group by member;
SELECT * FROM tablename WHERE SUBSTRING(fieldname, 1, 1)
REGEXP '[[:digit:]]';
```

# 5. IN / NOT IN

```
select * from members where id in ('1','100','1000');
select * from members where group_id in (select id from
members_group);
```

# 6. ALL / Any

NOT IN 与 <> ALL 两个语句是相同的:

```
SELECT s1 FROM t1 WHERE s1 <> ALL (SELECT s1 FROM t2);
SELECT s1 FROM t1 WHERE s1 NOT IN (SELECT s1 FROM t2);
```

IN 与 " = ANY" 两个语句是一样的:

```
SELECT s1 FROM t1 WHERE s1 = ANY (SELECT s1 FROM t2);
SELECT s1 FROM t1 WHERE s1 IN    (SELECT s1 FROM t2);
```

**例 6.1. SQL ANY example**

```
select * from members where id = any(select members_id from
accounts where id < 100);
```

# 7. exists, not exists

```
SELECT c.id, companyname
FROM customers c
WHERE EXISTS(
    SELECT orderid FROM orders o WHERE o.customer_id = cu.id)
```

# 8. UNION

union 分页问题

```
(SELECT a FROM tbl_name_a WHERE a=10 AND B=1)
UNION
(SELECT a FROM tbl_name_b WHERE a=11 AND B=2)
ORDER BY a LIMIT 10;
```

```
select * from (
        select a from tbl_name_a WHERE a=10 AND B=1
        union all
        select a from tbl_name_b WHERE a=10 AND B=1
) tbl_name
order by a limit 0,1;
```

## 8.1. UNION ALL

UNION ALL 不会合并重复的记录

```
select a,b from tbl_name_a WHERE a=10 AND B=1
union all
select a,b from tbl_name_b WHERE a=10 AND B=1
```

## 8.2. 两张表字段不对等解决方法

```
SELECT * FROM
    (
    SELECT contract_address, decimals, name, symbol, seq,
```

```
logo FROM token
    UNION
    SELECT contract_address, decimals, name, symbol, 100,
'https://www.netkiller.cn/images/eth.jpg' FROM user_token
WHERE address = '0xB94054c174995AE2A9E7fcf6c7924635FBa8ECF7'
AND contract_address NOT IN (SELECT contract_address FROM
token)
    ) AS tmp
ORDER BY seq
```

# 9. OUTFILE/LOAD DATA INFILE

查询结果输出到文件

```
SELECT * FROM tablename INTO OUTFILE '/tmp/tablename.txt';
```

使用tee将屏幕输出到文件

```
mysql>tee /home/neo/screen.txt
mysql>select * from user;
mysql>exit
```

```
SELECT * INTO OUTFILE '/home/mark/Orders.txt'
   FIELDS
   TERMINATED BY = ','
   FROM Orders
   WHERE Order_Date >= '2000-01-01'
```

```
LOAD DATA INFILE 'data.txt' INTO TABLE db2.my_table;
```

## 9.1. Export data to CSV from MySQL

```
SELECT *
INTO OUTFILE '/tmp/products.csv'
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
```

```
ESCAPED BY '\\'
LINES TERMINATED BY '\n'
FROM products
```

## 9.2. Import data from CSV file.

```
LOAD DATA LOW_PRIORITY LOCAL INFILE 'C:\\hx.csv' IGNORE INTO
TABLE `tmp`.`creditlog`
CHARACTER SET gbk FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED
BY '"' ESCAPED BY '"' LINES TERMINATED BY '\r\n'
(`ctime`, `login`, `mode`, `type`, `prevavailcredit`, `change`,
`newavailcredit`, `comment`);
```

# 10. CASE Syntax

```
CASE case_value
    WHEN when_value THEN statement_list
    [WHEN when_value THEN statement_list] ...
    [ELSE statement_list]
END CASE

Or:

CASE
    WHEN search_condition THEN statement_list
    [WHEN search_condition THEN statement_list] ...
    [ELSE statement_list]
END CASE
```

# 11. 查询结果放入变量

```
SELECT count(*)  INTO @num FROM mytable;
SELECT @num;
```

# 12. MySQL 专有命令

## 12.1. SQL_NO_CACHE

```
SELECT /*!40001 SQL_NO_CACHE */ * FROM table
```

## 12.2. SIGNAL Syntax

```
DROP TRIGGER `members_before_insert`;
CREATE DEFINER=`neo`@`%` TRIGGER `members_before_insert`
BEFORE INSERT ON `members` FOR EACH ROW BEGIN
        IF new.username IS NOT NULL THEN
                IF not exists(select username from
members_available where username = new.username) THEN
                /*set new.username = NULL;*/
                SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT =
'An error occurred', MYSQL_ERRNO = 1001;
                END IF;
        END IF;
END;
```

```

```

# 13. SQL 92

insert + select

```
insert into product_type_commission select id,5,1,1,0,0,0,0,0,0
from product_type where title='notebook' and is_physical=0;
```

update table1,table2

```
begin;
ALTER TABLE `customer` ADD COLUMN `cutoff_time` TIMESTAMP NOT
NULL default '0000-00-00 00:00:00';
update customer,agent set customer.cutoff_time =
agent.cutoff_time where customer.id = agent.id;
ALTER TABLE `agent` DROP COLUMN `cutoff_time`;
commit;
```

update table1 set field1 = (select value from table2)

```
UPDATE
        transaction
SET
        transaction.total_sold_price = (
                SELECT
                        SUM(transaction_item.price)
        FROM
                        transaction_item
                WHERE transaction_item.transaction_id = 100
        )
WHERE
        transaction.id = 100
```

update table1, (select * from other) as table2 set table1.field1 = table2.field1

```
UPDATE
        transaction,(   SELECT

SUM(product_item.bought_price) AS total_bought_price,
transaction_item.transaction_id
                                        FROM

transaction_item
                                    WHERE

transaction_item.transaction_id IN ( '123','456' )
                                ) as total
SET
        transaction.total_bought_price =
total.total_bought_price
WHERE
        transaction.id = total.transaction_id
```

join + subquery

```
select u.*,t.category,t.items,t.[property] from
tb_sysregchkusers as u left join (select a.items as category,
b.* from (select id, items from tb_sysregchktask where
categoryid=0) as a left join tb_sysregchktask as b on
b.categoryid=a.id ) as t on u.taskID=t.id

select * from tb_sysregchklog where
CONVERT(datetime,CONVERT(varchar(10),checkTime,120)) between
convert(datetime,'2007-12-12') and convert(datetime,'2007-12-
12')
```

```
select DISTINCT user_point_history.user_id,user.username,
(select count(id) from transaction where id =
user_point_history.transaction_id) as transactions,
(SELECT SUM(u_p_h.points) FROM user_point_history as u_p_h
WHERE u_p_h.type != 'RDMP' AND u_p_h.status IN('pr','ac') AND
u_p_h.user_id = user_point_history.user_id) as
total_points_earned,
(SELECT SUM(u_p_h.points) FROM user_point_history as u_p_h
```

```
WHERE u_p_h.type = 'RDMP' AND u_p_h.status IN('pr','ac') AND
u_p_h.user_id = user_point_history.user_id) as
total_points_redeemed
from user_point_history,user where user_point_history.user_id =
user.id;
```

(total_points_earned - total_points_redeemed) as current_balance_points

```
select user_id, username, transactions, total_points_earned,
total_points_redeemed, (total_points_earned -
total_points_redeemed) as current_balance_points
from (select DISTINCT user_point_history.user_id,user.username,
(select count(id) from transaction where id =
user_point_history.transaction_id) as transactions,
(SELECT SUM(u_p_h.points) FROM user_point_history as u_p_h
WHERE u_p_h.type != 'RDMP' AND u_p_h.status IN('pr','ac') AND
u_p_h.user_id = user_point_history.user_id) as
total_points_earned,
(SELECT SUM(u_p_h.points) FROM user_point_history as u_p_h
WHERE u_p_h.type = 'RDMP' AND u_p_h.status IN('pr','ac') AND
u_p_h.user_id = user_point_history.user_id) as
total_points_redeemed
from user_point_history,user where user_point_history.user_id =
user.id) as user_performance;
```

subquery作为一个字段使用

```
select product_type_attribute.*,(select 'selected' from
product_type_attribute_set where
product_type_attribute_set.product_type_attribute_id =
product_type_attribute.id and
product_type_attribute_set.product_type_id = 26) as selected
from product_type_attribute;
```

# 第 7 章 Functions and Operators

## 1. COUNT

count()

```
SELECT (SELECT count(1) FROM ecs_category) as 'Export category
count',
        (SELECT count(1) FROM ecs_goods) as 'Goods count',
        (SELECT count(1) FROM ecs_goods_attr) as 'Attr count';
```

# 2. group_concat() 列传行

```
SELECT tags FROM neo.article;

linux
redis
mysql
java
php
```

tags字段专为一行显示

```
SELECT group_concat(tags) FROM neo.article;

linux,redis,mysql,java,php
```

distinct 去除重复数据

```
select group_concat(distinct author) from neo.article;
```

以id分组，把name字段的值打印在一行，分号分隔

```
select id,group_concat(tags separator ';') from neo.article
group by tags;
```

排序结果

```
select group_concat(distinct author order by author desc) from
neo.article;
```

# 3. UUID()

```
SELECT UUID(),LENGTH(UUID()),UUID_SHORT(),
LENGTH(UUID_SHORT());
```

# 4. String

## 4.1. LEFT/RIGHT

LEFT(str,len)

```
mysql> select left(concat('1','0000000'),5) as number;
+--------+
| number |
+--------+
| 10000  |
+--------+
1 row in set (0.00 sec)
```

RIGHT(str,len)

```
mysql> select right(concat('0000000','1'),5) as number;
+--------+
| number |
+--------+
| 00001  |
+--------+
1 row in set (0.00 sec)
```

## 4.2. RPAD/LPAD

补齐长度用'0'填充

**RPAD(str,len,padstr)**

```
mysql> select rpad('10',5,'0') as txt;
+-------+
| txt   |
+-------+
| 10000 |
+-------+
1 row in set (0.01 sec)
```

**LPAD(str,len,padstr)**

```
mysql> select lpad('10',5,'0') as txt;
+-------+
| txt   |
+-------+
| 00010 |
```

```
+-------+
1 row in set (0.00 sec)
```

## 4.3. CONCAT

CONCAT(str1,str2,...)

```
mysql> select concat('Neo',' ','Chen') as Name;
+----------+
| Name     |
+----------+
| Neo Chen |
+----------+
1 row in set (0.00 sec)
```

## 4.4. CONCAT_WS

```
SELECT CONCAT_WS(',', 'Neo', 'Chen');
Neo,Chen

SELECT CONCAT_WS('-', 'Neo', 'Chen');
Neo-Chen
```

使用逗号链接字符串

```
SELECT
    CONCAT_WS(',', id, name, age)
FROM
    mytable
```

## 4.5. 链接所有字段

当我使用 select CONCAT_WS(",", *) as string from tab 时发现不支持 * 操作。

解决方案如下

```
SET @column = NULL;
```

```
SELECT
    GROUP_CONCAT(COLUMN_NAME) AS fields INTO @column
FROM
    INFORMATION_SCHEMA.Columns
WHERE
    table_name = 'mytable'
        AND table_schema = 'test';

-- select @column;

SET @sql = CONCAT('SELECT CONCAT_WS(",",',@column, ' )  FROM mytable');

select @sql;

PREPARE stmt FROM @sql;
EXECUTE stmt;
DEALLOCATE PREPARE stmt;
```

## 4.6. GROUP_CONCAT

```
mysql> select GROUP_CONCAT(CONVERT( username , CHAR (16)) order by username
desc) as username from test;
+-----------------------------------------+
| username                                |
+-----------------------------------------+
| jam,jam2,john,john2,john3,neo,neo1,neo2   |
+-----------------------------------------+
6 rows in set, 1 warning (0.01 sec)
```

## 4.7. replace

```
select replace(goods_desc,':8000','') from ecs_goods;

update ecs_goods set goods_desc=replace(goods_desc,':8000','');
```

## 4.8. SUBSTRING

```
mysql> SELECT SUBSTRING('netkiller',4,4);
+---------------------------+
| SUBSTRING('netkiller',4,4) |
+---------------------------+
| kill                      |
+---------------------------+
1 row in set (0.00 sec)
```

```

```

与left,right 相同的用法

```
select right('M2014030615410572307:DEPOSIT', 7);
SELECT SUBSTRING('M2014030615410572307:DEPOSIT', -7);
```

## 4.9. SUBSTRING_INDEX

```
SELECT SUBSTRING_INDEX('M2014030615410572307:DEPOSIT', ':', -1);
SELECT SUBSTRING_INDEX('M2014030615410572307:DEPOSIT', ':', 1);
```

## 4.10. AES_ENCRYPT / AES_DECRYPT

简单用法

```


mysql> select AES_ENCRYPT('helloworld','key');
+----------------------------------+
| AES_ENCRYPT('helloworld','key') |
+----------------------------------+
|                                  |
+----------------------------------+
1 row in set (0.00 sec)

mysql> select AES_DECRYPT(AES_ENCRYPT('helloworld','key'),'key');
+-----------------------------------------------------+
| AES_DECRYPT(AES_ENCRYPT('helloworld','key'),'key') |
+-----------------------------------------------------+
| helloworld                                          |
+-----------------------------------------------------+
1 row in set (0.00 sec)

mysql>


```

加密数据入库

```


CREATE TABLE `encryption` (
        `mobile` VARBINARY(16) NOT NULL,
        `key` VARCHAR(32) NOT NULL
)
ENGINE=InnoDB;

INSERT INTO encryption(`mobile`,`key`)VALUES(
```

```
AES_ENCRYPT('13691851789',md5('13691851789')), md5('13691851789'))
select AES_DECRYPT(mobile,`key`), length(mobile) from encryption;
```

# 5. Date and Time

```
SELECT NOW(),CURRENT_TIMESTAMP(),SYSDATE();
```

## 5.1. year/month/day hour:minite:second

```
mysql> select year('2012-03-20');
+--------------------+
| year('2012-03-20') |
+--------------------+
|               2012 |
+--------------------+
1 row in set (0.00 sec)

mysql> select month('2012-03-20');
+---------------------+
| month('2012-03-20') |
+---------------------+
|                   3 |
+---------------------+
1 row in set (0.00 sec)

mysql> select day('2012-03-20');
+-------------------+
| day('2012-03-20') |
+-------------------+
|                20 |
+-------------------+
1 row in set (0.00 sec)

mysql> select hour('12:30:55');
+------------------+
| hour('12:30:55') |
+------------------+
|               12 |
+------------------+
1 row in set (0.00 sec)
```

```
mysql> select minute('12:30:55');
+--------------------+
| minute('12:30:55') |
+--------------------+
|                 30 |
+--------------------+
1 row in set (0.00 sec)

mysql> select second('12:30:55');
+--------------------+
| second('12:30:55') |
+--------------------+
|                 55 |
+--------------------+
1 row in set (0.00 sec)
```

## 5.2. Unix time

语法：FROM_UNIXTIME(unix_timestamp,format)
返回表示 Unix 时间标记的一个字符串，根据format字符串格式化。
format可以包含与DATE_FORMAT()函数列出的条目同样的修饰符。
根据format字符串格式化date值。
下列修饰符可以被用在format字符串中：
%M 月名字(January……December)
%W 星期名字(Sunday……Saturday)
%D 有英语前缀的月份的日期(1st, 2nd, 3rd, 等等。）
%Y 年, 数字, 4 位
%y 年, 数字, 2 位
%a 缩写的星期名字(Sun……Sat)
%d 月份中的天数, 数字(00……31)
%e 月份中的天数, 数字(0……31)
%m 月, 数字(01……12)
%c 月, 数字(1……12)
%b 缩写的月份名字(Jan……Dec)

%j 一年中的天数(001……366)
%H 小时(00……23)
%k 小时(0……23)
%h 小时(01……12)
%I 小时(01……12)
%l 小时(1……12)
%i 分钟, 数字(00……59)
%r 时间,12 小时(hh:mm:ss [AP]M)
%T 时间,24 小时(hh:mm:ss)
%S 秒(00……59)
%s 秒(00……59)
%p AM或PM
%w 一个星期中的天数(0=Sunday ……6=Saturday ）
%U 星期(0……52), 这里星期天是星期的第一天
%u 星期(0……52), 这里星期一是星期的第一天
%% 一个文字“%”。

```
mysql> SELECT UNIX_TIMESTAMP('2005-03-27 02:00:00');
+---------------------------------------+
| UNIX_TIMESTAMP('2005-03-27 02:00:00') |
+---------------------------------------+
|                            1111885200 |
+---------------------------------------+
mysql> SELECT FROM_UNIXTIME(1111885200);
+---------------------------+
| FROM_UNIXTIME(1111885200) |
+---------------------------+
| 2005-03-27 03:00:00       |
+---------------------------+
```

```
SELECT UNIX_TIMESTAMP('2012-01-01 00:00:00');
SELECT UNIX_TIMESTAMP('2012-07-30 00:00:00');
SELECT UNIX_TIMESTAMP();
```

```
SELECT UNIX_TIMESTAMP('2009-08-06') ;
SELECT UNIX_TIMESTAMP( curdate( ) );

select FROM_UNIXTIME(UNIX_TIMESTAMP('2012-07-30 00:00:00'),
'%Y-%m-%d');
SELECT FROM_UNIXTIME( 1249488000, '%Y年%m月%d日' );

SELECT FROM_UNIXTIME(time_stamp, '%Y-%m-%d %H:%i:%S') FROM
test.transaction_history;

select FROM_UNIXTIME(createtime, '%m') as month, count(1) as
count from members where createtime BETWEEN
UNIX_TIMESTAMP('2012-01-01 00:00:00') and
UNIX_TIMESTAMP('2012-12-31 00:00:00') group by
FROM_UNIXTIME(createtime, '%m');
select FROM_UNIXTIME(createtime, '%m') as month, count(1) as
count from members where createtime BETWEEN
UNIX_TIMESTAMP('2011-01-01 00:00:00') and
UNIX_TIMESTAMP('2011-12-31 00:00:00') group by
FROM_UNIXTIME(createtime, '%m');

select FROM_UNIXTIME(createtime, '%m-%d') as month, count(1)
as count from members where createtime BETWEEN
UNIX_TIMESTAMP('2011-01-01 00:00:00') and
UNIX_TIMESTAMP('2011-12-31 00:00:00') group by
FROM_UNIXTIME(createtime, '%m-%d');
select FROM_UNIXTIME(createtime, '%m-%d') as month, count(1)
as count from members where createtime BETWEEN
UNIX_TIMESTAMP('2012-01-01 00:00:00') and
UNIX_TIMESTAMP('2012-12-31 00:00:00') group by
FROM_UNIXTIME(createtime, '%m-%d');
```

## 5.3. DATE_FORMAT

DATE_FORMAT() 函数用于以不同的格式显示日期/时间数据。

```
语法
DATE_FORMAT(date,format)
date 参数是合法的日期。format 规定日期/时间的输出格式。
```

可以使用的格式有：

| 格式 | 描述 |
| --- | --- |
| %a | 缩写星期名 |
| %b | 缩写月名 |
| %c | 月，数值 |
| %D | 带有英文前缀的月中的天 |
| %d | 月的天，数值(00-31) |
| %e | 月的天，数值(0-31) |
| %f | 微秒 |
| %H | 小时 (00-23) |
| %h | 小时 (01-12) |
| %I | 小时 (01-12) |
| %i | 分钟，数值(00-59) |
| %j | 年的天 (001-366) |
| %k | 小时 (0-23) |
| %l | 小时 (1-12) |
| %M | 月名 |
| %m | 月，数值(00-12) |
| %p | AM 或 PM |
| %r | 时间，12-小时（hh:mm:ss AM 或 PM） |
| %S | 秒(00-59) |
| %s | 秒(00-59) |
| %T | 时间，24-小时 (hh:mm:ss) |
| %U | 周 (00-53) 星期日是一周的第一天 |
| %u | 周 (00-53) 星期一是一周的第一天 |
| %V | 周 (01-53) 星期日是一周的第一天，与 %X 使用 |
| %v | 周 (01-53) 星期一是一周的第一天，与 %x 使用 |
| %W | 星期名 |
| %w | 周的天 （0=星期日，6=星期六） |
| %X | 年，其中的星期日是周的第一天，4 位，与 %V 使用 |
| %x | 年，其中的星期一是周的第一天，4 位，与 %v 使用 |
| %Y | 年，4 位 |
| %y | 年，2 位 |

## 实例

下面的脚本使用 DATE_FORMAT() 函数来显示不同的格式。我们使用 NOW() 来获得当前的日期/时间：

```
DATE_FORMAT(NOW(),'%b %d %Y %h:%i %p')
DATE_FORMAT(NOW(),'%m-%d-%Y')
DATE_FORMAT(NOW(),'%d %b %y')
DATE_FORMAT(NOW(),'%d %b %Y %T:%f')
```

```
SELECT DATE_FORMAT(NOW(),'%Y-%m-%d');

select DATE_FORMAT(asctime,'%Y-%m-%d') as Date, count(1) as
Count from logging where tag='www' and facility='login' group
by DATE_FORMAT(asctime,'%Y-%m-%d') order by asctime desc;
```

## 5.4. DATE_SUB/DATE_ADD

当前时间向后推10天

```
mysql> select DATE_SUB(now(), INTERVAL 240 HOUR);
+-----------------------------------+
| DATE_SUB(now(), INTERVAL 240 HOUR) |
+-----------------------------------+
| 2012-03-09 10:26:03                |
+-----------------------------------+
1 row in set (0.00 sec)

mysql> select DATE_SUB(now(), INTERVAL 24 HOUR);
+-----------------------------------+
| DATE_SUB(now(), INTERVAL 24 HOUR) |
+-----------------------------------+
| 2012-03-18 10:28:43                |
+-----------------------------------+
1 row in set (0.00 sec)
```

```
DELETE from Message where created < DATE_sub(now(), INTERVAL
240 HOUR);

select * from PRICES_HISTORY where DATE_FORMAT(TIME
,GET_FORMAT(DATE,'ISO')) = (
select if ( WEEKDAY(CURRENT_DATE())=6 ,
DATE_SUB(CURRENT_DATE(), INTERVAL 1 DAY) , CURRENT_DATE())
)
```

**DATE_ADD**

```
SELECT DATE_ADD('1998-01-02', INTERVAL 31 DAY);
```

## 5.5. datediff / timediff

计算时间差，两个时间相减结果

```
mysql> select timediff('22:20:00','17:30:00');
+----------------------------------+
| timediff('22:20:00','17:30:00') |
+----------------------------------+
| 04:50:00                         |
+----------------------------------+
1 row in set (0.00 sec)

mysql> select datediff('2008-08-08 12:00:00', '2008-08-01
00:00:00');
+-----------------------------------------------------------+
| datediff('2008-08-08 12:00:00', '2008-08-01 00:00:00') |
+-----------------------------------------------------------+
|                                                         7 |
+-----------------------------------------------------------+
1 row in set (0.00 sec)
```

# 6. 数值函数

## 6.1. cast 类型转换

```
mysql> SELECT cast(SUBSTRING('123456789',1,4) as UNSIGNED) *
100;
+-----------------------------------------------------+
| cast(SUBSTRING('123456789',1,4) as UNSIGNED) * 100 |
+-----------------------------------------------------+
|                                              123400 |
+-----------------------------------------------------+
1 row in set (0.00 sec)
```

## 6.2. truncate 保留小数位数

```
select profit, deficit, concat(truncate((profit /
deficit)*100,2),'%') as percentage from ((select count(*) as
profit from angelfund where profit > 0) as profit, (select
count(*) as deficit from angelfund where profit < 0) as
deficit);
```

## 6.3. MOD 求余

```
mysql> select 9 mod 5;
+---------+
| 9 mod 5 |
+---------+
|       4 |
+---------+
```

```
1 row in set (0.00 sec)

mysql> select mod(5,2);
+----------+
| mod(5,2) |
+----------+
|        1 |
+----------+
1 row in set (0.00 sec)

mysql> select mod(5,2);
```

# 7. Control Flow Functions

CASE

```
mysql> SELECT CASE 1 WHEN 1 THEN 'one'
    ->     WHEN 2 THEN 'two' ELSE 'more' END;
        -> 'one'
mysql> SELECT CASE WHEN 1>0 THEN 'true' ELSE 'false' END;
        -> 'true'
mysql> SELECT CASE BINARY 'B'
    ->     WHEN 'a' THEN 1 WHEN 'b' THEN 2 END;
        -> NULL
```

IFNULL

```
mysql> SELECT IFNULL("TEST", 'OK');
+----------------------+
| IFNULL("TEST", 'OK') |
+----------------------+
| TEST                 |
+----------------------+
1 row in set (0.00 sec)

mysql> SELECT IFNULL(NULL, 'OK');
+--------------------+
| IFNULL(NULL, 'OK') |
+--------------------+
| OK                 |
+--------------------+
1 row in set (0.00 sec)
```

NULLIF()

IF

```
mysql> SELECT IFNULL("TEST", 'OK');
+----------------------+
| IFNULL("TEST", 'OK') |
+----------------------+
| TEST                 |
+----------------------+
1 row in set (0.00 sec)

mysql> SELECT IFNULL(NULL, 'OK');
+--------------------+
| IFNULL(NULL, 'OK') |
+--------------------+
| OK                 |
+--------------------+
1 row in set (0.00 sec)
```

# 第 8 章 DCL (Data Control Language)

```
COMMIT - save work done
SAVEPOINT - identify a point in a transaction to which you can
later roll back
ROLLBACK - restore database to original since the last COMMIT
SET TRANSACTION - Change transaction options like what rollback
segment to use
```

## 1. 锁

锁机制

```
    1）共享锁：由读表操作加上的锁，加锁后其他用户只能获取该表或行的共享
锁，不能获取排它锁，也就是说只能读不能写
    2）排它锁：由写表操作加上的锁，加锁后其他用户不能获取该表或行的任何
锁，典型是mysql事务中的

    锁的范围：
    行锁：对某行记录加上锁
    表锁：对整个表加上锁
```

共享锁(share mode), 排他锁(for update)

### 1.1. 共享锁

### 1.2. 排他锁

下面做作一个实验，验证锁的效果

终端一,首先进入事务状态然后运行下面语句

```
mysql> begin;
Query OK, 0 rows affected (0.00 sec)

mysql> select * from t1 where id='3' for update;
+----+--------+---------------------+---------------------+
| id | name   | ctime               | mtime               |
+----+--------+---------------------+---------------------+
|  3 | test   | 0000-00-00 00:00:00 | 2013-01-14 13:05:41 |
+----+--------+---------------------+---------------------+
1 row in set (0.00 sec)
```

终端二, 查询表中数据

```
mysql> select * from t1;
+----+--------+---------------------+---------------------+
| id | name   | ctime               | mtime               |
+----+--------+---------------------+---------------------+
|  1 | neo    | 0000-00-00 00:00:00 | 2013-01-14 13:00:00 |
|  2 | zen    | 0000-00-00 00:00:00 | 2013-01-14 13:00:43 |
|  3 | test   | 0000-00-00 00:00:00 | 2013-01-14 13:05:41 |
+----+--------+---------------------+---------------------+
3 rows in set (0.00 sec)
```

增加“for update”查询非锁定记录

```
mysql> select * from t1 where id=2 for update;
+----+------+---------------------+---------------------+
| id | name | ctime               | mtime               |
+----+------+---------------------+---------------------+
|  2 | zen  | 0000-00-00 00:00:00 | 2013-01-14 13:00:43 |
+----+------+---------------------+---------------------+
1 row in set (0.00 sec)
```

查询被锁定记录

```
mysql> select * from t1 where id=3 for update;
ERROR 1205 (HY000): Lock wait timeout exceeded; try restarting
transaction
```

查询所有记录，因为记录中包含了id=3那条，所以也不允许查询。

```
mysql> select * from t1 for update;
ERROR 1205 (HY000): Lock wait timeout exceeded; try restarting
transaction
```

测试修改记录

```
mysql> UPDATE `t1` SET `name`='testaa' WHERE  `id`=3;
ERROR 1205 (HY000): Lock wait timeout exceeded; try restarting
transaction
```

**提示**

在没有出现ERROR 1205 (HY000)的这段时间，只要终端一中执行commit,rollback锁就释放了.终端二中的语句就会运行。

select trx_query from information_schema.innodb_trx; 可以查看被锁的SQL语句

## 1.3. 锁

### 表的加锁与解锁

```
LOCK TABLES tablename WRITE;
LOCK TABLES tablename READ;

...
...

UNLOCK TABLES;
```

```
CREATE TABLE `locking` (
        `name` VARCHAR(50) NULL DEFAULT NULL
)
ENGINE=InnoDB
;

mysql> insert into locking values('test');
Query OK, 1 row affected (0.02 sec)

mysql> select * from locking;
+------+
| name |
+------+
| test |
+------+
1 row in set (0.00 sec)

mysql> UNLOCK TABLES;
```

```
mysql> LOCK TABLES locking READ;
Query OK, 0 rows affected (0.00 sec)

mysql> insert into locking values('test');
ERROR 1099 (HY000): Table 'locking' was locked with a READ lock
and can't be updated


mysql> LOCK TABLE locking WRITE;
```

```
Query OK, 0 rows affected (0.00 sec)

mysql> select * from locking;
+------+
| name |
+------+
| test |
| test |
+------+
2 rows in set (0.00 sec)

mysql> insert into locking values('test');
Query OK, 1 row affected (0.05 sec)

mysql> UNLOCK TABLES;
```

## 禁止查询

```
mysql> LOCK TABLE locking AS myalias READ;
Query OK, 0 rows affected (0.00 sec)

mysql> select * from locking;
ERROR 1100 (HY000): Table 'locking' was not locked with LOCK
TABLES

mysql> select * from locking as myalias;
+------+
| name |
+------+
| test |
| test |
| test |
+------+
3 rows in set (0.00 sec)
```

## 1.4. 锁等待与超时

当你开启了事务 begin 忘记，或者各种原因没有commit也没有rollback。悲剧了！

超时设置

```
begin;
SET SESSION wait_timeout = 60;
select * from locking for update;
```

60秒内如果没有commit/rollback将自动释放本次事务。

**select for update nowait**

使用 for update 是会遇到一个问题，就是其他用户会漫长的等待，而我们需要程序非阻塞运行，当遇到 for update 的时候应该立即返回此表已被加锁。

mysql 并没有实现 nowait 关键字（类似Oracle的功能），但又一个方法能够达到同样目的。

```
mysql> select @@innodb_version;
+------------------+
| @@innodb_version |
+------------------+
| 5.6.24           |
+------------------+
1 row in set (0.05 sec)

mysql> select * from locking;
ERROR 1100 (HY000): Table 'locking' was not locked with LOCK
TABLES
```

此时需要等待很长时间才能提示 “Table 'locking' was not locked with LOCK TABLES”

```
mysql> set session innodb_lock_wait_timeout=1;
Query OK, 0 rows affected (0.02 sec)

mysql> select * from locking for update;
ERROR 1205 (HY000): Lock wait timeout exceeded; try restarting
transaction
```

设置 innodb_lock_wait_timeout 参数后，很快就返回

```
mysql> show variables like 'innodb_lock_wait_timeout';
+--------------------------+-------+
| Variable_name            | Value |
+--------------------------+-------+
| innodb_lock_wait_timeout | 1     |
+--------------------------+-------+
1 row in set (0.00 sec)

mysql> show global variables like 'innodb_lock_wait_timeout';
+--------------------------+-------+
| Variable_name            | Value |
+--------------------------+-------+
| innodb_lock_wait_timeout | 50    |
+--------------------------+-------+
1 row in set (0.00 sec)
```

innodb_lock_wait_timeout 默认值是 50

# 2. 事务处理和锁定语句

## Transactional and Locking Statements

开始事务 begin、start transaction 或者 set autocommit=0

```
事务的特征：原子性（Atomiocity）、一致性（Consistency）、隔离性
（Isolation）和持久性（Durability），这四个特性简称ACID特性。
    原子性：事务是数据库的逻辑工作单位，事务中包括的所有操作要么都做，要么
都不做。
    一致性：事务执行的结果必须是使数据库从一个一致性的状态变到另外一个一致
性状态。
    隔离性：一个事务的执行不能被其他事务干扰。即一个事务内部的操作及使用的
数据对其他事务是隔离的，并发执行的各个事务之间互相不干扰。
    持久性：一个事务一旦成功提交，对数据库中数据的修改就是持久性的。接下来
其他的其他操作或故障不应该对其执行结果有任何影响。
```

## 2.1. 事务隔离级别

事务隔离模式

```
1) READ UNCOMMITED
SELECT的时候允许脏读，即SELECT会读取其他事务修改而还没有提交的数据。

2)READ COMMITED
SELECT的时候无法重复读，即同一个事务中两次执行同样的查询语句，若在第一次与
第二次查询之间时间段，其他事务又刚好修改了其查询的数据且提交了，则两次读到的
数据不一致。

3) REPEATABLE READ
SELECT的时候可以重复读，即同一个事务中两次执行同样的查询语句，得到的数据始
终都是一致的。实现的原理是，在一个事务对数据行执行读取或写入操作时锁定了这些
数据行。
但是这种方式又引发了幻想读的问题。因为只能锁定读取或写入的行，不能阻止另一个
事务插入数据，后期执行同样的查询会产生更多的结果。

4)SERIALIZABLE
与可重复读的唯一区别是，默认把普通的SELECT语句改成SELECT …. LOCK IN
```

```
SHARE MODE。即为查询语句涉及到的数据加上共享琐，阻塞其他事务修改真实数据。
serializable模式中，事务被强制为依次执行。这是SQL标准建议的默认行为。
```

可以通过下列语句查询全局和当前会话的事务隔离级别：

```
SELECT @@global.tx_isolation;
SELECT @@tx_isolation;
```

```
查看InnoDB系统级别的事务隔离级别：
mysql> SELECT @@global.tx_isolation;

查看InnoDB会话级别的事务隔离级别：
mysql> SELECT @@tx_isolation;

修改InnoDB系统级别的事务隔离级别：
mysql> set global transaction isolation level read committed;

修改InnoDB会话级别的事务隔离级别：
mysql> set session transaction isolation level read committed;
```

## 2.2. 事务所用到的表

information_schema

```
select * from innodb_trx;
select * from innodb_lock_waits;
select * from innodb_locks;
```

## 2.3. 解决更新冲突

```
CREATE TABLE `account` (
        `id` INT(10) UNSIGNED NOT NULL AUTO_INCREMENT,
```

```
        `user` VARCHAR(50) NOT NULL DEFAULT '0',
        `cash` FLOAT NOT NULL DEFAULT '0',
        `point` INT(10) UNSIGNED NOT NULL DEFAULT '0',
        PRIMARY KEY (`id`),
        UNIQUE INDEX `user` (`user`)
)
COLLATE='utf8_general_ci'
ENGINE=InnoDB;
```

```
INSERT INTO `test`.`account` (`user`, `cash`,`point`) VALUES
('neo', 10,10);
```

下面通过account表，我来模拟一个返点场景，例如电商网站经常会用到“返点”，购买一定数量的商品赠送一定的点数，可以通过点数买东西，这样涉及到点的加于减操作。

**表 8.1. 更新丢失演示**

| Session A | Session B |
| --- | --- |
| `select point into @point from account where user='neo';` | |
| | `select point into @point from account where user='neo';` |
| `update account set point=@point+20 where user='neo';` | |

| | |
|---|---|
| | update account set point=@point+50 where user='neo'; |

看看最后用户有多少点?

```
mysql> select point from account where user='neo';
+-------+
| point |
+-------+
|    30 |
+-------+
1 row in set (0.00 sec)
```

傻了吧，老板发火，测试不能重现，运维说这是程序计算错误，程序员说程序没有错误，这样的场景国内很多公司都出现过吧?

问题出在哪里呢？出在并发上，很多web程序员很少考虑并发是产生的问题，怎么解决？很多方案，在我的职业生涯过程就见过很多奇葩方案，都能解决问题但不太完美。

如果更新语句改为 update account set point=@point+50 where user='neo' and point=@point; 会更保险,但仍然不能解决同意时间所产生的更新操作

下面是通过事务与锁彻底解决上面的问题。

```
mysql> SELECT @@tx_isolation;
+-----------------+
| @@tx_isolation  |
+-----------------+
| REPEATABLE-READ |
```

```
+-----------------+
1 row in set (0.00 sec)
```

检查事务隔离级别为：REPEATABLE-READ

表 **8.2.** 防止更新丢失加锁演示

| Session A | Session B |
| --- | --- |
| `begin;`<br>`select point into @point from account where user='neo' for update;` | |
| | `begin;`<br>`select point into @point from account where user='neo' for update;`<br>执行到此处会挂起 |
| `update account set point=@point+20 where user='neo';`<br>`commit;` | |
| | `update account set point=@point+50 where user='neo';` |

| | |
|---|---|
| | commit; |

上面解决更新覆盖问题，但从数据库设计角度是不应该这样设计表的。仅供参考

```
CREATE TABLE `account` (
        `id` INT(10) UNSIGNED NOT NULL AUTO_INCREMENT,
        `user` VARCHAR(50) NOT NULL DEFAULT '0',
        `cash` FLOAT NOT NULL DEFAULT '0',
        `point` INT(10) NOT NULL DEFAULT '0',
        PRIMARY KEY (`id`)
)
COLLATE='utf8_general_ci'
ENGINE=InnoDB;
```

每一次数据变化新增一条数据

```
INSERT INTO `test`.`account` (`user`, `point`) VALUES ('neo',
-10);
INSERT INTO `test`.`account` (`user`, `point`) VALUES ('neo',
-5);
INSERT INTO `test`.`account` (`user`, `point`) VALUES ('neo',
30);
INSERT INTO `test`.`account` (`user`, `point`) VALUES ('neo',
-20);
```

计算剩余点数

```
select sum(point) as point from account where user='neo';
```

## 2.4. 共享锁

先加上共享锁，此时也会对mytable表加上IS锁

```
begin;
select * from mytable where id=1 for share;
```

观察IS锁

```
select * from performance_schema.data_locks;
```

```

```

## 2.5. SAVEPOINT

```
DROP PROCEDURE IF EXISTS doOrder;

DELIMITER $$

CREATE PROCEDURE doOrder(IN orderUUID VARCHAR(40))
  BEGIN
    DECLARE EXIT HANDLER FOR SQLEXCEPTION ROLLBACK TO
sp_order;

    START TRANSACTION;
    SAVEPOINT sp_order;

    -- doing my updates and selects here...
```

```
    COMMIT;

  END $$

DELIMITER ;
```

# 第 9 章 Optimization

## 1. Limit 状态

```
$ sudo cat /proc/`pidof mysqld`/limits
Limit                     Soft Limit           Hard Limit
Units
Max cpu time              unlimited            unlimited
seconds
Max file size             unlimited            unlimited
bytes
Max data size             unlimited            unlimited
bytes
Max stack size            10485760             unlimited
bytes
Max core file size        0                    unlimited
bytes
Max resident set          unlimited            unlimited
bytes
Max processes             62662                62662
processes
Max open files            20480                20480
files
Max locked memory         65536                65536
bytes
Max address space         unlimited            unlimited
bytes
Max file locks            unlimited            unlimited
locks
Max pending signals       62662                62662
signals
Max msgqueue size         819200               819200
bytes
Max nice priority         0                    0
Max realtime priority     0                    0
Max realtime timeout      unlimited            unlimited
us
```

## 2. 使用 Btrfs 文件系统存储mysql数据

```
#!/bin/sh
systemctl stop mysqld

btrfs subvolume create /srv/@mysql
btrfs subvolume list /srv/

UUID=$(blkid | grep btrfs | sed -e 's/.*UUID="\
([^"]*\)".*/\1/')
# UUID=786f570d-fe5c-4d5f-832a-c1b0963dd4e6 /srv btrfs defaults
1 1
cat << EOF >> /etc/fstab
UUID=${UUID} /var/lib/mysql  btrfs
noatime,nodiratime,subvol=@mysql 0 2
EOF

mkdir /tmp/mysql
mv /var/lib/mysql/* /tmp/mysql/

mount /var/lib/mysql/
chown mysql:mysql /var/lib/mysql

mv /tmp/mysql/* /var/lib/mysql/

systemctl start mysqld
```

# 3. 打开表的数量

```
mysql> SHOW STATUS LIKE 'open%tables';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| Open_tables   | 100   |
| Opened_tables | 1     |
+---------------+-------+
2 rows in set (0.04 sec)
```

# 4. Buffering and Caching

查看缓存是否开启

```
MySQL> select @@query_cache_type;
MySQL> show variables like 'query_cache_type';
```

开启与关闭缓存

```
MySQL> set query_cache_type=on;
MySQL> set query_cache_type=off;
```

查看缓存状态

```
show variables like 'have_query_cache';
```

查询缓存的大小

```
MySQL> select @@global.query_cache_size;
MySQL> select @@query_cache_size;
```

查看最大缓存限制，如果集大于该数则不缓存。

```
MySQL> select @@global.query_cache_limit;
```

清除缓存/重置缓存

```
MySQL> flush tables;
MySQL> flush query cache;
MySQL> reset query cache;
```

查询缓存性能

```
MySQL> show status like 'qcache%';

MySQL> show status like 'qcache_q%';
+-------------------------+-------+
| Variable_name | Value |
+-------------------------+-------+
| Qcache_queries_in_cache | 1 |
+-------------------------+-------+
1 row in set (0.00 sec)

MySQL> show status like 'qcache_f%';
+--------------------+----------+
| Variable_name | Value |
+--------------------+----------+
| Qcache_free_blocks | 1 |
| Qcache_free_memory | 16766728 |
+--------------------+----------+
2 rows in set (0.00 sec)
```

## 4.1. Query Cache SELECT Options

```
Two query cache-related options may be specified in SELECT
statements:

SQL_CACHE

The query result is cached if it is cacheable and the value
of the query_cache_type system variable is ON or DEMAND.

SQL_NO_CACHE

The query result is not cached.

Examples:

SELECT SQL_CACHE id, name FROM customer;
SELECT SQL_NO_CACHE id, name FROM customer;


SELECT /*! SQL_NO_CACHE */ stuff FROM table
```

### 例 9.1. SQL_CACHE 测试

下面的例子中你将看到缓存变化

```
flush tables;
show status like 'qcache_q%';
select sql_cache * from members limit 5;
show status like 'qcache_q%';
select sql_cache * from members limit 10;
show status like 'qcache_q%';
```

显示当前缓存中的信息数量:

```
MySQL> show status like 'qcache_q%';
```

其中各个参数的意义如下:

Qcache_free_blocks：缓存中相邻内存块的个数。数目大说明可能有碎片。FLUSH QUERY CACHE会对缓存中的碎片进行整理，从而得到一个空闲块。
Qcache_free_memory：缓存中的空闲内存。
Qcache_hits：每次查询在缓存中命中时就增大
Qcache_inserts：每次插入一个查询时就增大。命中次数除以插入次数就是不中比率。
Qcache_lowmem_prunes：缓存出现内存不足并且必须要进行清理以便为更多查询提供空间的次数。这个数字最好长时间来看;如果这个 数字在不断增长，就表示可能碎片非常严重，或者内存很少。(上面的 free_blocks和free_memory可以告诉您属于哪种情况)
Qcache_not_cached：不适合进行缓存的查询的数量，通常是由于这些查询不是 SELECT 语句或者用了now()之类的函数。
Qcache_queries_in_cache：当前缓存的查询(和响应)的数量。
Qcache_total_blocks：缓存中块的数量。

# 5. where 优化

where 条件的顺序影响查询速度

```
EXPLAIN select *,from_unixtime(sendtime) from sms where
id='461539' and content like '13%';
/* 0 rows affected, 1 rows found. Duration for 1 query: 0.218
sec. */

EXPLAIN select *,from_unixtime(sendtime) from sms where content
like '13%' and id='461539';
/* 0 rows affected, 1 rows found. Duration for 1 query: 0.717
sec. */
```

# 6. SHOW PROFILE Syntax SQL性能分析器

### 例 9.2. SHOW PROFILE Syntax

```
set profiling = 1; select * from mytab; show profile for query
1;
```

# 7. PROCEDURE ANALYSE()

数据列优化

```
SELECT ... FROM ... WHERE ... PROCEDURE ANALYSE([max_elements,
[max_memory]])
```

# 第 10 章 MySQL Connectors

## 1. JDBC

JDBC connection settings

```
jdbc:mysql://hostname:port/database?
autoReconnect=true&useUnicode=true&characterEncoding=utf8
```

confluence.cfg.xml

```
<property
name="hibernate.connection.url">jdbc:mysql://hostname:port/da
tabase?
autoReconnect=true&amp;useUnicode=true&amp;characterEncoding=
utf8</property>
```

# 2. ODBC

# 3. MySQL native driver for PHP - mysqlnd

# 4. python-mysqldb

```
$ apt-cache search python | grep mysql
python-mysqldb - A Python interface to MySQL
python-mysqldb-dbg - A Python interface to MySQL (debug
extension)

$ sudo apt-get install python-mysqldb
```

```
# -*- coding: utf-8 -*-
#mysqldb
import time, MySQLdb

#连接
conn=MySQLdb.connect(host="localhost",user="root",passwd="",db=
"test",charset="utf8")
cursor = conn.cursor()

#写入
sql = "insert into user(name,created) values(%s,%s)"
param = ("neo",int(time.time()))
n = cursor.execute(sql,param)
print n

#更新
sql = "update user set name=%s where id=3"
param = ("jam")
n = cursor.execute(sql,param)
print n

#查询
n = cursor.execute("select * from user")
for row in cursor.fetchall():
    for r in row:
        print r

#删除
sql = "delete from user where name=%s"
param =("neo")
n = cursor.execute(sql,param)
```

```
print n
cursor.close()

#关闭
conn.close()
```

# 第 11 章 MySQL GUI/Web Manager

## 1. HeidiSQL

http://www.heidisql.com/

# 2. Toad for MySQL Freeware

http://toadsoft.veriomigrations.com/

# 3. phpMyAdmin - MySQL web administration tool

homepage: http://www.phpmyadmin.net/

```
$ wget
http://nchc.dl.sourceforge.net/sourceforge/phpmyadmin/phpMyAdmi
n-3.1.3.1-all-languages.tar.bz2
$ tar jxvf phpMyAdmin-3.1.3.1-all-languages.tar.bz2
$ ln -s phpMyAdmin-3.1.3.1-all-languages phpMyAdmin
```

# 4. Maatkit Essential command-line utilities for MySQL

http://www.maatkit.org/

# 第 12 章 Miscellaneous

## 1. Multi-Master Replication Manager for MySQL

# 2. MHA

https://code.google.com/p/mysql-master-ha/

# 3. HandlerSocket

# 4. Maatkit

http://www.maatkit.org/

# 5. Mroonga

**Mroonga 是一个 MySQL 存储引擎，基于 Groonga，提供完整的全文搜索引擎。Groonga 是一款可嵌入式的全文搜寻引擎，具有储存功能和全文搜寻的检索功能。**

http://mroonga.org/

http://groonga.org/

# 6. Amoeba

http://sourceforge.net/projects/amoeba/

Amoeba(变形虫)项目,该开源框架于2008年 开始发布一款 Amoeba for Mysql软件。这个软件致力于MySQL的分布式数据库前端代理层，它主要在应用层访问MySQL的 时候充当SQL路由功能，专注于分布式数据库代理层（Database Proxy）开发。座落与 Client、DB Server(s)之间,对客户端透明。具有负载均衡、高可用性、SQL 过滤、读写分离、可路由相关的到目标数据库、可并发请求多台数据库合并结果。 通过Amoeba你能够完成多数据源的高可用、负载均衡、数据切片的功能，目前Amoeba已在很多 企业的生产线上面使用。

# 第 13 章 FAQ

## 1. Reset root password 重置MySQL root密码

忘记root密码是使用 --skip-grant-tables 启动项

CentOS 6.x

```
# vim /etc/init.d/mysqld

 $exec --skip-grant-tables  --datadir="$datadir" --socket="$socketfile"
\
    --pid-file="$mypidfile" \
    --basedir=/usr --user=mysql >/dev/null 2>&1 &
```

```
# /etc/init.d/mysqld restart
Stopping mysqld:                                           [  OK  ]
Starting mysqld:                                           [  OK  ]

# mysqladmin -u root flush-privileges password "newpassword"
```

### 1.1. MySQL 5.7.x

CentOS 7.x

添加 skip-grant-tables=1 选项，然后重启mysql

```
# cat /etc/my.cnf
# For advice on how to change settings please see
# http://dev.mysql.com/doc/refman/5.6/en/server-configuration-
defaults.html

[mysqld]
#
# Remove leading # and set to the amount of RAM for the most important
```

```
data
# cache in MySQL. Start at 70% of total RAM for dedicated server, else
10%.
# innodb_buffer_pool_size = 128M
#
# Remove leading # to turn on a very important data integrity option:
logging
# changes to the binary log between backups.
# log_bin
#
# Remove leading # to set options mainly useful for reporting servers.
# The server defaults are faster for transactions and fast SELECTs.
# Adjust sizes as needed, experiment to find the optimal values.
# join_buffer_size = 128M
# sort_buffer_size = 2M
# read_rnd_buffer_size = 2M
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
skip-grant-tables=1
# Disabling symbolic-links is recommended to prevent assorted security
risks
symbolic-links=0

# Recommended in standard MySQL setup
sql_mode=NO_ENGINE_SUBSTITUTION,STRICT_TRANS_TABLES

[mysqld_safe]
log-error=/var/log/mysqld.log
pid-file=/var/run/mysqld/mysqld.pid
```

```
# systemctl restart mysqld
```

```
update mysql.user set authentication_string=password('netkiller') where
user='root' and Host = 'localhost';
flush privileges;
quit;
```

```
# mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
```

```
Your MySQL connection id is 2
Server version: 5.7.14 MySQL Community Server (GPL)

Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights
reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

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

mysql> update mysql.user set authentication_string=password('netkiller')
where user='root' and Host = 'localhost';
Query OK, 1 row affected, 1 warning (0.03 sec)
Rows matched: 1  Changed: 1  Warnings: 1

mysql> flush privileges;
Query OK, 0 rows affected (0.00 sec)

mysql> quit;
Bye
```

删除 skip-grant-tables=1 重启MySQL

## 1.2. MySQL 8.0

```
[root@localhost log]# vim /etc/my.cnf

[mysqld]
skip-grant-table
```

```
ALTER USER root@localhost identified by 'MQiEge1ikst7S_6tlXzBOmt';
ALTER USER root@localhost PASSWORD EXPIRE NEVER;
```

# 2. 查看错误代码

```
mysql> \! perror 6
OS error code   6:  No such device or address
```

## 2.1. ERROR 1153 (08S01) at line 3168: Got a packet bigger than 'max_allowed_packet' bytes

```
max_allowed_packet=500M
```

## 2.2. ERROR 1129 (00000): Host 'XXXXXX' is blocked because of many connection errors; unblock with 'mysqladmin flush-hosts'

连接在中途被中断了的连接请求。在 max_connect_errors次失败请求后，mysql阻止该主机进一步的连接，直到管理员执行命令 mysqladmin flush-hosts。

```
mysql> flush hosts;
```

```
set global
max_connect_errors=1000;
```

```
max_connect_errors=10000
```

## 3. 临时表是否需要建索引

答案：要

# 4. [Warning] Changed limits: max_open_files: 5000 (requested 20480)

```
2018-01-08T01:34:44.515973Z 0 [Warning] Changed limits:
max_open_files: 5000 (requested 10240)
2018-01-08T01:34:44.516402Z 0 [Warning] Changed limits:
table_open_cache: 1471 (requested 2000)
```

提出出现在 CentOS 7 ulimit 配置没有问题的情况下mysql日志提示 Warning

```
# ulimit -Sa | grep "open files"
open files (-n) 40960
```

```
[root@netkiller ~]# cat /proc/`pidof
mysqld`/limits
Limit Soft Limit Hard Limit Units
Max cpu time unlimited unlimited
seconds
Max file size unlimited unlimited bytes
Max data size unlimited unlimited bytes
Max stack size 8388608 unlimited bytes
Max core file size 0 unlimited bytes
Max resident set unlimited unlimited
bytes
Max processes 63494 63494 processes
Max open files 5000 5000 files
Max locked memory 65536 65536 bytes
Max address space unlimited unlimited
bytes
Max file locks unlimited unlimited
locks
Max pending signals 63494 63494 signals
Max msgqueue size 819200 819200 bytes
```

```
                        Max nice priority 0 0
                        Max realtime priority 0 0
                        Max realtime timeout unlimited
unlimited us
```

动态改变

```
                        [root@netkiller ~]# egrep '^(Limit|Max
open files)' /proc/`pidof mysqld`/limits
                        Limit Soft Limit Hard Limit Units
                        Max open files 5000 5000 files
```

问题的出现出现原因是systemctl启动脚本覆盖了ulimit配置

```
                        # cat
/usr/lib/systemd/system/mysqld.service | grep -A2
open_files_limit
                        # Sets open_files_limit
                        LimitNOFILE = 5000
```

解决方法，直接修改上面的数值，不建议修改mysqld.service，这样会影响你下次升级。请采用下面的方案完美解决:

```
mkdir /usr/lib/systemd/system/mysqld.service.d

cat >> /usr/lib/systemd/system/mysqld.service.d/override.conf
<<EOF
[Service]
LimitNOFILE=40960
EOF
```

重启 MySQL

```
systemctl daemon-reload
systemctl restart mysqld
```

检查是否生效

```
mysql> show variables like 'open_files_limit';
+------------------+-------+
| Variable_name    | Value |
+------------------+-------+
| open_files_limit | 65535 |
+------------------+-------+
1 row in set (0.01 sec)
```

# 5. Table 'performance_schema.session_variables' doesn't exist

升级表文件即可

```
mysql_upgrade -u root -p --force
或
mysql_upgrade -h 127.0.0.1 -u root -p --force
```

# 6. SQL Error (1038): Out of sort memory, consider increasing server sort buffer size

解决方法，在 my.cnf 中增加 sort_buffer_size=512k 配置项

```
[root@gitlab ~]# cat /etc/my.cnf.d/mysql-server.cnf
#
# This group are read by MySQL server.
# Use it for options that only the server (but not clients)
should see
#
# For advice on how to change settings please see
# http://dev.mysql.com/doc/refman/en/server-configuration-
defaults.html

# Settings user and group are ignored when systemd is used.
# If you need to run mysqld under a different user or group,
# customize your systemd unit file for mysqld according to the
# instructions in http://fedoraproject.org/wiki/Systemd

[mysqld]
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
log-error=/var/log/mysql/mysqld.log
pid-file=/run/mysqld/mysqld.pid


# Add by Neo
character-set-server=utf8mb4
collation-server=utf8mb4_general_ci
explicit_defaults_for_timestamp=true
lower_case_table_names=1
sql_mode=STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR
_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION
sort_buffer_size=512k
```

# 7. this is incompatible with sql_mode=only_full_group_by

ERROR 1055 (42000): Expression #1 of SELECT list is not in GROUP BY clause and contains nonaggregated column 'mydb.contact.id' which is not functionally dependent on columns in GROUP BY clause; this is incompatible with sql_mode=only_full_group_by

```
mysql> select @@version;
+-----------+
| @@version |
+-----------+
| 5.7.10    |
+-----------+
1 row in set (0.00 sec)

mysql> select @@GLOBAL.sql_mode;
+-----------------------------------------------------------------
-----------------------------------------------------------------
-------------+
| @@GLOBAL.sql_mode
|
+-----------------------------------------------------------------
-----------------------------------------------------------------
-------------+
|
ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_
DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_AUTO_CREATE_USER,NO_ENGINE_S
UBSTITUTION |
+-----------------------------------------------------------------
-----------------------------------------------------------------
-------------+
1 row in set (0.00 sec)

mysql> SET sql_mode = '';
Query OK, 0 rows affected, 1 warning (0.00 sec)

mysql> select id,name from contact group by name limit 10;
+-------+-------------+
```

```
| id    | name        |
+-------+-------------+
| 84046 |   张伟      |
| 80259 |   张磊      |
|   784 |   王岩      |
| 87685 |  杨钞       |
+-------+-------------+
10 rows in set (0.07 sec)
```

不建议设置 SET sql_mode = ''，正确方式如下：

```
mysql> set global
sql_mode='STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERRO
R_FOR_DIVISION_BY_ZERO,NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTI
ON';
mysql> set session
sql_mode='STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERRO
R_FOR_DIVISION_BY_ZERO,NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTI
ON';
```

或者采用

```
Adding only one mode to sql_mode without removing existing
ones:

SET sql_mode=(SELECT CONCAT(@@sql_mode,',<mode_to_add>'));
Removing only a specific mode from sql_mode without removing
others:

SET sql_mode=(SELECT
REPLACE(@@sql_mode,'<mode_to_remove>',''));
In your case, if you want to remove only ONLY_FULL_GROUP_BY
mode, then use below command:

SET sql_mode=(SELECT REPLACE(@@sql_mode, 'ONLY_FULL_GROUP_BY',
''));
```

# 8. ERROR 1071 (42000) at line 25: Specified key was too long; max key length is 767 bytes

这个保存通常出现在高版本数据库向低版本数据导入数据，活着云主机例如阿里云。

Error Code: 1071. Specified key was too long; max key length is 767 bytes

```
mysql> show variables like '%innodb_large_prefix%';
+----------------------+-------+
| Variable_name        | Value |
+----------------------+-------+
| innodb_large_prefix | OFF   |
+----------------------+-------+
1 row in set (0.00 sec)
```

解决方案

```
innodb_large_prefix=ON
```

```
mysql> set global innodb_large_prefix=on;
Query OK, 0 rows affected (0.00 sec)
```

# 9. ERROR 1086 (HY000): File '/var/lib/mysql-files/order.txt' already exists

SELECT * FROM tabname INTO OUTFILE 不支持覆盖操作，这是MySQL从安全角度考虑的。

```
mysql> SELECT * FROM `order` INTO OUTFILE '/var/lib/mysql-
files/order.txt';
ERROR 1086 (HY000): File '/var/lib/mysql-files/order.txt'
already exists
```

# 10. ERROR 1114 (HY000): The table 'your_table' is full

出现这个错误一般是 tmp_table_size 设置小于 max_heap_table_size 造成的。

show variables like '%table_size%';

解决方法，将 tmp_table_size 设置等于 max_heap_table_size

```
mysql> show variables like '%table_size%';
+---------------------+----------+
| Variable_name       | Value    |
+---------------------+----------+
| max_heap_table_size | 16777216 |
| tmp_table_size      | 16777216 |
+---------------------+----------+
2 rows in set (0.00 sec)
```

# 11. Error Code: 1146. Table 'test.CACHE_UPDATE' doesn't exist

问题分析，首先确认表是存在的，但是无法读取。可以判定是 lower_case_table_names=1 选项的问题，开启后表以小写方式打开。

```
Error Code: 1146. Table 'test.CACHE_UPDATE' doesn't exist
```

如果是 MySQL 8.0 之前没有开启 lower_case_table_names=1 ，现在需要开启，加入配置后将无法启动，解决办法是，你需要重做 mysql data 目录

```
[root@localhost ~]# rm -rf /var/lib/mysql/*
[root@localhost ~]# systemctl restart mysqld
[root@localhost ~]# mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.21 Source distribution

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights
reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

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

mysql> show variables like '%lower_case_table_names%';
+------------------------+-------+
| Variable_name          | Value |
+------------------------+-------+
| lower_case_table_names | 1     |
+------------------------+-------+
1 row in set (0.01 sec)

mysql> SELECT CURRENT_SERIAL FROM CACHE_UPDATE WHERE ID=1;
+----------------+
| CURRENT_SERIAL |
+----------------+
|              7 |
```

```
+----------------+
1 row in set (0.00 sec)
```

# 12. Error Code: 1175. You are using safe update mode and you tried to update a table without a WHERE that uses a KEY column.

Error Code: 1175. You are using safe update mode and you tried to update a table without a WHERE that uses a KEY column. To disable safe mode, toggle the option in Preferences -> SQL Editor and reconnect.

解决方案

```
SET SQL_SAFE_UPDATES = 0;
delete from tables;
```

# 13. ERROR 1273 (HY000) at line 3364: Unknown collation: 'utf8mb4_0900_ai_ci'

找出指定字符集的表

```
select TABLE_SCHEMA,TABLE_NAME,TABLE_COLLATION from
information_schema.tables where table_collation =
'utf8mb4_0900_ai_ci' and table_schema = 'your_schema';
```

```
SELECT
        CONCAT(
                'ALTER TABLE ',
                TABLE_NAME,
                ' CONVERT TO CHARACTER SET utf8mb4 COLLATE
utf8mb4_general_ci;'
        )
FROM
        information_schema.`TABLES`
WHERE
        TABLE_SCHEMA = 'DATABASE_NAME';
```

# 14. ERROR 1290 (HY000): The MySQL server is running with the --secure-file-priv option so it cannot execute this statement

MySQL 不允许向 secure_file_priv 意外的目录导出文件。

```
mysql> SELECT * FROM `order` INTO OUTFILE '/tmp/order.txt';
ERROR 1290 (HY000): The MySQL server is running with the --
secure-file-priv option so it cannot execute this statement

mysql> show variables like '%secure%';
+--------------------------+-----------------------+
| Variable_name            | Value                 |
+--------------------------+-----------------------+
| require_secure_transport | OFF                   |
| secure_auth              | ON                    |
| secure_file_priv         | /var/lib/mysql-files/ |
+--------------------------+-----------------------+
3 rows in set (0.00 sec)

mysql> SELECT * FROM `order` INTO OUTFILE '/var/lib/mysql-
files/order.txt';
Query OK, 3 rows affected (0.00 sec)

root@netkiller ~ % cat /var/lib/mysql-files/order.txt
1       Tom     22      2017-11-16 17:23:15
2       Neo     34.65   2017-11-16 17:29:28
3       Cici    34.98   2017-11-16 17:30:29
```

在 my.cnf 中 加入 secure-file-priv=/tmp 可以修改为你需要的目录。

# 15. ERROR 1364: 1364: Field 'id' doesn't have a default value

```
set
@@SESSION.sql_mode='NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION'
;
SELECT @@GLOBAL.sql_mode;
UPDATE `cms`.`content` SET `source`='test' WHERE
`content_id`='1099';
```

# 16. Error Code: 1292. Incorrect datetime value: '0000-00-00 00:00:00' for column 'create_time' at row 95692

```
select @global.sql_mode;

'STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVI
SION_BY_ZERO,NO_ENGINE_SUBSTITUTION'
```

去掉 NO_ZERO_IN_DATE,NO_ZERO_DATE 即可

```
set @global.sql_mode =
'STRICT_TRANS_TABLES,NO_ENGINE_SUBSTITUTION';
```

# 17. ERROR 1415: Not allowed to return a result set from a trigger

触发器中不允许返回结果集，解决方法是顶一个变量，然后将返回值返回给变量。

```
DROP TRIGGER IF EXISTS `test`.`demo_AFTER_INSERT`;

DELIMITER $$
USE `test`$$
CREATE DEFINER=`root`@`%` TRIGGER `test`.`demo_AFTER_INSERT`
AFTER INSERT ON `demo` FOR EACH ROW
BEGIN
        set @rev = "";
        SELECT
    OUT2FILE('/tmp/demo.log',
            CONCAT_WS(',',
                    NEW.id,
                    NEW.name,
                    NEW.sex,
                    NEW.address))
        INTO @rev;
END$$
DELIMITER ;
```

# 18. ERROR 1503 (HY000): A PRIMARY KEY must include all columns in the table's partitioning function

http://dev.mysql.com/doc/refman/5.1/en/partitioning-limitations-partitioning-keys-unique-keys.html

# 19. ERROR 1819 (HY000): Your password does not satisfy the current policy requirements

MySQL 5.7 密码强度，必须含有0-9，a-z,A-Z以及“-”或“_”

https://dev.mysql.com/doc/refman/5.7/en/validate-password-options-variables.html

禁用密码安全策略（早起5.7版本可用，新版已经废弃这个选项）

```
                        # cat /etc/my.cnf | grep validate-
password
                        validate-password=OFF
```

新的方式修改策略与密码长度

```
mysql> set global validate_password_policy=0;
mysql> set global validate_password_length=4;
mysql> grant all privileges on test.* to 'test'@localhost
identified by 'chen';
```

# 20. ERROR 1820 (HY000): You must reset your password using ALTER USER statement before executing this statement.

这个错误来自 MySQL 5.7，首次登陆MySQL 5.7 必须修改密码

```
                        ALTER USER 'root'@'localhost'
IDENTIFIED BY 'your_password';
```

# 21. ERROR 1840 (HY000) at line 24: @@GLOBAL.GTID_PURGED can only be set when @@GLOBAL.GTID_EXECUTED is empty.

问题出现在 MySQL 5.7 导入数据库时候

```
[www@testing ~]$ zcat netkiller.2021-08-19.sql.gz | mysql
netkiller
ERROR 1840 (HY000) at line 24: @@GLOBAL.GTID_PURGED can only be
set when @@GLOBAL.GTID_EXECUTED is empty.
```

解决方案

执行 reset master;

```
[www@testing ~]$ mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 25669
Server version: 5.7.35 MySQL Community Server (GPL)

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All
rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or
its
affiliates. Other names may be trademarks of their respective
owners.

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

mysql> reset master;
```

```
Query OK, 0 rows affected (0.01 sec)

mysql> exit
Bye
[www@testing ~]$ zcat netkiller.2021-08-19.sql.gz | mysql
netkiller
```

# 22. ERROR 2002 (HY000): Can't connect to local MySQL server through socket '/var/lib/mysql/mysql.sock'

错误提示

```
[root@stage stage]# mysql
ERROR 2002 (HY000): Can't connect to local MySQL server through
socket '/var/lib/mysql/mysql.sock'
```

这种情况一般检查 my.cnf 中的 socket 配置项，看看 socket 被配置到了什么目录

配置文件增加 socket 指向正确路径

```
[client]
socket=/your/path/mysql/mysql.sock
```

通过参数指定

```
mysql -uroot -p --socket=/your/path/mysql/mysql.sock
```

Docker 默认是 /var/run/mysqld/mysqld.sock 我们将其改为 /var/lib/mysql/mysql.sock

```
```

```
services:
  mysql:
    command:
    - --socket=/var/lib/mysql/mysql.sock
    - --default-authentication-plugin=mysql_native_password
    - --character-set-server=utf8mb4
    - --collation-server=utf8mb4_general_ci
    - --explicit_defaults_for_timestamp=true
    - --lower_case_table_names=1
    - --max_execution_time=0
    container_name: mysql
    depends_on: []
    environment:
    - TZ=Asia/Shanghai
    - MYSQL_ROOT_PASSWORD=passw0rd
    - MYSQL_DATABASE=netkiller
    - MYSQL_USER=netkiller
    - MYSQL_PASSWORD=netkiller
    hostname: db.netkiller.com
    image: mysql:5.7
    ports:
    - 3306:3306
    restart: always
    volumes:
    - /var/lib/mysql:/var/lib/mysql
    -
/opt/netkiller.com/api.netkiller.cn/mysql/conf.d/server.cnf:/et
c/mysql/conf.d/server.cnf
    - /opt/netkiller.com/api.netkiller.cn/mysql/docker-
entrypoint-initdb.d:/docker-entrypoint-initdb.d
```

# 23. ERROR 2013 (HY000): Lost connection to MySQL server during query

```
mysql> SELECT count(*) FROM netkiller.chain_info;
ERROR 2013 (HY000): Lost connection to MySQL server during
query
No connection. Trying to reconnect...
ERROR 2002 (HY000): Can't connect to local MySQL server through
socket '/var/lib/mysql/mysql.sock' (111)
ERROR:
Can't connect to the server

mysql>
```

```
2021-11-19T09:25:15.833160Z 0 [ERROR] [MY-011906] [InnoDB]
Database page corruption on disk or a failed file read of page
[page id: space=1662, page number=1245]. You may have to
recover from a backup.
 len 16384; hex
7f5d9c6e000004dd000004db000004de000000043ab8e71f45bf00000000000
000000000067e000b3f37802c3af101593dab00050000007035102f2e294c23
641dc117ee11b40b88055300637f5d9c6e3ab8e71f; asc:   ;
InnoDB: End of page dump
InnoDB: Page may be an index page where index id is 2840
2021-11-19T09:25:15.885824Z 0 [ERROR] [MY-013183] [InnoDB]
Assertion failure: buf0lru.cc:2026:bpage->buf_fix_count == 0
thread 140289099208448
InnoDB: We intentionally generate a memory trap.
InnoDB: Submit a detailed bug report to http://bugs.mysql.com.
InnoDB: If you get repeated assertion failures or crashes, even
InnoDB: immediately after the mysqld startup, there may be
InnoDB: corruption in the InnoDB tablespace. Please refer to
InnoDB: http://dev.mysql.com/doc/refman/8.0/en/forcing-innodb-
recovery.html
InnoDB: about forcing recovery.
09:25:15 UTC - mysqld got signal 6 ;
```

```
Most likely, you have hit a bug, but this error can also be
caused by malfunctioning hardware.
Thread pointer: 0x0
Attempting backtrace. You can use the following information to
find out
where mysqld died. If you see no messages after this, something
went
terribly wrong...
stack_bottom = 0 thread_stack 0x46000
/usr/libexec/mysqld(my_print_stacktrace(unsigned char const*,
unsigned long)+0x41) [0x561905a584e1]
/usr/libexec/mysqld(handle_fatal_signal+0x313) [0x5619048651d3]
/lib64/libpthread.so.0(+0x12c20) [0x7f97af67ac20]
/lib64/libc.so.6(gsignal+0x10f) [0x7f97acb0949f]
/lib64/libc.so.6(abort+0x127) [0x7f97acaf3db5]
/usr/libexec/mysqld(+0xddf6ab) [0x5619045886ab]
/usr/libexec/mysqld(+0x2616098) [0x561905dbf098]
/usr/libexec/mysqld(buf_LRU_free_one_page(buf_page_t*, bool,
bool)+0x1e) [0x561905dc159e]
/usr/libexec/mysqld(buf_read_page_handle_error(buf_page_t*)+0x1
30) [0x561905d9f1b0]
/usr/libexec/mysqld(buf_page_io_complete(buf_page_t*,
bool)+0xc83) [0x561905da0dc3]
/usr/libexec/mysqld(fil_aio_wait(unsigned long)+0x153)
[0x561905e70f73]
/usr/libexec/mysqld(+0x25321d8) [0x561905cdb1d8]
/usr/libexec/mysqld(std::thread::_State_impl<std::thread::_Invo
ker<std::tuple<Runnable, void (*)(unsigned long), unsigned
long> > >::_M_run()+0xaa) [0x561905cdb9aa]
/lib64/libstdc++.so.6(+0xc2ba3) [0x7f97ad4f3ba3]
/lib64/libpthread.so.0(+0x817a) [0x7f97af67017a]
/lib64/libc.so.6(clone+0x43) [0x7f97acbceee3]
The manual page at
http://dev.mysql.com/doc/mysql/en/crashing.html contains
information that should help you find out what is causing the
crash.
```

解决方案

```
echo 2 > /proc/sys/vm/drop_caches
```

修复后

```
mysql> SELECT count(*) FROM netkiller.chain_info;
+----------+
| count(*) |
+----------+
|    41890 |
+----------+
1 row in set (0.00 sec)
```

可能是虚拟内存中的数据出错导致。另外也可以尝试下面方法，在 my.cnf 中加入，我没有验证过，因为这种方式需要重启数据库才能生效。我的环境不允许重启，所以没有验证。

```
[mysqld]
innodb_force_recovery = 1
```

添加 innodb_force_recovery 可以恢复数据读取，但是不能修改，此时可以 dump 备份该表，然后 去掉 innodb_force_recovery 参数，再启动数据库，删除损坏的表，导入新表。

# 24. ERROR 2059 (HY000): Authentication plugin 'caching_sha2_password' cannot be loaded: /usr/lib64/mysql/plugin/caching_sha2_password.so: cannot open shared object file: No such file or directory

错误信息

```
[root@nacos-74fb777857-qsvlx nacos]# mysql -h mysql-
0.mysql.default.svc.cluster.local -unacos -pnacos
ERROR 2059 (HY000): Authentication plugin
'caching_sha2_password' cannot be loaded:
/usr/lib64/mysql/plugin/caching_sha2_password.so: cannot open
shared object file: No such file or directory
```

原因分析，MySQL 服务器是 8.0，MySQL 客户端是 5.7 导致客户端链接服务器出错。

```
MySQL [(none)]> select version();
+-----------+
| version() |
+-----------+
| 8.0.27    |
+-----------+
1 row in set (0.00 sec)
```

解决方法，用户密码使用旧的 mysql_native_password

```
neo@Netkiller-iMac ~/workspace [1]> kubectl exec -it mysql-0 --
```

```
bash
root@mysql-0:/# mysql -uroot -p123456
mysql: [Warning] Using a password on the command line interface
can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 35
Server version: 8.0.27 MySQL Community Server - GPL

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or
its
affiliates. Other names may be trademarks of their respective
owners.

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

mysql> ALTER USER 'nacos'@'%' IDENTIFIED WITH
mysql_native_password BY 'nacos';
Query OK, 0 rows affected (0.21 sec)

mysql> ALTER USER 'nacos'@'%' IDENTIFIED BY 'nacos';
Query OK, 0 rows affected (0.16 sec)

mysql> flush privileges;
Query OK, 0 rows affected (0.07 sec)
```

解决方案二

```
vim my.cnf
[mysqld]
default_authentication_plugin=mysql_native_password
```

# 25. ERROR 3024 (HY000): Query execution was interrupted, maximum statement execution time exceeded

```
mysql> select * from cert;
ERROR 3024 (HY000): Query execution was interrupted, maximum
statement execution time exceeded
mysql> SET GLOBAL max_execution_time=10;
Query OK, 0 rows affected (0.01 sec)

mysql> select * from cert;
Empty set (0.08 sec)

mysql> show variables like 'max_execution_time';
+--------------------+-------+
| Variable_name      | Value |
+--------------------+-------+
| max_execution_time | 10    |
+--------------------+-------+
1 row in set (0.01 sec)

mysql> select /*+ max_execution_time(3000)*/ count(*) from
cert;
+----------+
| count(*) |
+----------+
|        0 |
+----------+
1 row in set (0.29 sec)
```

# 26. Authentication plugin 'caching_sha2_password' cannot be loaded: /usr/lib64/mysql/plugin/caching_sha2_password.so: cannot open shared object file: No such fileor directory

这个故障出现在 MySQL 8.0 上，用户使用 mysql client 5.7 链接 MySQL 8.0 提示如下

```
[root@netkiller ~]# mysql -h 193.112.95.53 -uroot -p
Enter password:
ERROR 2059 (HY000): Authentication plugin
'caching_sha2_password' cannot be loaded:
/usr/lib64/mysql/plugin/caching_sha2_password.so: cannot open
shared object file: No such fileor directory
```

解决方案，创建用户使用 mysql_native_password密码

```
mysql> CREATE USER 'root'@'%' IDENTIFIED WITH
mysql_native_password BY 'pMQiEge1ikst7S_6tlXzBOmt_4b';
Query OK, 0 rows affected (0.08 sec)

mysql> grant all on *.* to 'root'@'%';
Query OK, 0 rows affected (0.08 sec)
```

重新链接

```
[root@netkiller ~]# mysql -h 193.112.95.53 -uneo -p
```

```
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 24
Server version: 8.0.11 MySQL Community Server - GPL

Copyright (c) 2000, 2016, Oracle and/or its affiliates. All
rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or
its
affiliates. Other names may be trademarks of their respective
owners.

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

mysql>
```

# 27. com.mysql.jdbc.exceptions.jdbc4.MySQLNonTransientConnectionException: Public Key Retrieval is not allowed

问题出在现在 MySQL 8.0 版本

解决方法：在连接后面添加 allowPublicKeyRetrieval=true

```
spring.datasource.url=jdbc:mysql://192.168.0.1:3306/test?
useUnicode=true&characterEncoding=UTF-
8&serverTimezone=UTC&useSSL=false&allowPublicKeyRetrieval=true
```

# 28. mysqldump: Couldn't execute 'SELECT COLUMN_NAME,

问题出现在 MySQL 8.0 备份 MySQL 5.7 数据库时。

```
mysqldump: Couldn't execute 'SELECT COLUMN_NAME,
JSON_EXTRACT(HISTOGRAM, '$."number-of-buckets-specified"')
FROM information_schema.COLUMN_STATISTICS                 WHERE
SCHEMA_NAME = 'testra' AND TABLE_NAME = 'branch';': Unknown
table 'column_statistics' in information_schema (1109)
```

解决办法，使用 --column-statistics=0 选项

```
mysqldump -hdb.netkiller.cn -uroot -ptest neo --column-
statistics=0
```

# 29. this is incompatible with sql_mode=only_full_group_by

```
Expression #2 of SELECT list is not in GROUP BY clause and
contains nonaggregated column 'test.table.username' which
is not functionally dependent on columns in GROUP BY clause;
this is incompatible with sql_mode=only_full_group_by
```

问题出在 MySQL 5.7 向 MySQL 8.0 迁移。

查询 sql_mode 设置

```
select @@global.sql_mode;
'ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_
DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION'
```

临时解决方案，去掉 ONLY_FULL_GROUP_BY 即可

```
set
@@GLOBAL.sql_mode='STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_D
ATE,ERROR_FOR_DIVISION_BY_ZERO,
NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION';
set
@@SESSION.sql_mode='STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_
DATE,ERROR_FOR_DIVISION_BY_ZERO,
NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION';
```

彻底解决，有三种解决方案:

- 第一种，将 MySQL 的版本降回到 5.7
- 第二种，关闭 only_full_group_by 检查
- 第三种，修改sql使其遵守only_full_group_by语法规则

采用哪个方案？打工人的方案，就是解决眼前问题，如果你对项目负责，最好采用第三种方案。

我的职业生涯遇到无数次，因版本太低同时发现重大漏洞，厂商已经不在维护该版本，此时一边是黑客攻击，一边你又无法短时间内升级到新版本，公司又不能接受停服，你会怎么应对？我用了很多非常规手段，给开发团队争取了一周的时间，升级系统。

# 30. mysqldump: [Warning] Using a password on the command line interface can be insecure.

```
mysqldump: [Warning] Using a password on the command line
interface can be insecure.
```

```
vim ~/.my.cnf

[mysqldump]
user=root
password=123456
```

```
[root@netkiller neo]# mysqldump -h 192.168.30.40 neo
[root@netkiller neo]# mysqldump --defaults-extra-file=~/.my.cnf
netkiller > netkiller.sql
```

# 31. mysql: [Warning] Using a password on the command line interface can be insecure.

当使用 -pnetkiller 在命令行出现密码的时候，会提示下面信息。

```
[www@testing ~]$ mysql -h127.0.0.1 -uneo -pnetkiller test
mysql: [Warning] Using a password on the command line interface
can be insecure.
Reading table information for completion of table and column
names
You can turn off this feature to get a quicker startup with -A

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 25623
Server version: 5.7.35 MySQL Community Server (GPL)

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All
rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or
its
affiliates. Other names may be trademarks of their respective
owners.

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

mysql>
```

创建 ~/.my.cnf 配置文件，将密码写入配置

```
[www@testing ~]$ cat ~/.my.cnf
[mysql]
host=127.0.0.1
```

```
user=neo
password=netkiller
```

这时直接使用 mysql 命令即可进入。

```
[www@testing ~]$ mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 25622
Server version: 5.7.35 MySQL Community Server (GPL)

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All
rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or
its
affiliates. Other names may be trademarks of their respective
owners.

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

mysql>
```

# 32. 时间自动被加一秒

背景，Spring Cloud + MyBatis-Plus 插入数据库的时间被无辜加了一秒中，MyBatis-Plus 日志 2022-12-05 11:40:49，MySQL binlog 2022-12-05 11:40:50。

经过排查，发现MyBatis-Plus传递给MySQL的时间携带了毫秒，2022-12-05 11:40:49.500。

```
CREATE TABLE `test_date` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `date` datetime DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8mb4
```

```
mysql> INSERT INTO `test`.`test_date` (`date`) VALUES ('2022-
12-05 11:40:49');
Query OK, 1 row affected (0.04 sec)

mysql> INSERT INTO `test`.`test_date` (`date`) VALUES ('2022-
12-05 11:40:49.449');
Query OK, 1 row affected (0.04 sec)

mysql> INSERT INTO `test`.`test_date` (`date`) VALUES ('2022-
12-05 11:40:49.500');
Query OK, 1 row affected (0.09 sec)

mysql> SELECT * FROM test.test_date;
+----+---------------------+
| id | date                |
+----+---------------------+
|  1 | 2022-12-05 11:40:49 |
|  2 | 2022-12-05 11:40:49 |
|  3 | 2022-12-05 11:40:50 |
```

```
+----+---------------------+
3 rows in set (0.00 sec)
```

第三条数据无故被增加了一秒，这是因为 500毫秒会做四舍五入，最终成了 2022-12-05 11:40:49.500

解决方案一，不要给 mysql 传递毫秒，将 2022-12-05 11:40:49.500 改为 2022-12-05 11:40:49

解决方案二，对于时间敏感的场合不允许差一秒，可以设置时间精度解决这个问题，例如 datetime(1)，最长设置 datetime(6)

```
CREATE TABLE `test_date` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `date` datetime(3) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=utf8mb4
```

```
mysql> INSERT INTO `test`.`test_date` (`date`) VALUES ('2022-
12-05 11:40:49');
Query OK, 1 row affected (0.02 sec)

mysql> INSERT INTO `test`.`test_date` (`date`) VALUES ('2022-
12-05 11:40:49.449');
Query OK, 1 row affected (0.04 sec)

mysql> INSERT INTO `test`.`test_date` (`date`) VALUES ('2022-
12-05 11:40:49.500');
Query OK, 1 row affected (0.09 sec)

mysql> SELECT * FROM test.test_date;
+----+-------------------------+
| id | date                    |
```

```
+----+-------------------------+
|  1 | 2022-12-05 11:40:49.000 |
|  2 | 2022-12-05 11:40:49.449 |
|  3 | 2022-12-05 11:40:49.500 |
+----+-------------------------+
3 rows in set (0.00 sec)
```

影响范围还有 timestamp, time

```
CREATE TABLE `test_date` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `date` timestamp NULL DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=utf8mb4

TRUNCATE `test`.`test_date`;
INSERT INTO `test`.`test_date` (`date`) VALUES ('2022-12-05
11:40:49');
INSERT INTO `test`.`test_date` (`date`) VALUES ('2022-12-05
11:40:49.449');
INSERT INTO `test`.`test_date` (`date`) VALUES ('2022-12-05
11:40:49.500');
SELECT * FROM test.test_date;
```

time 演示

```
DROP TABLE `test`.`test_date`;
CREATE TABLE `test_date` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `date` time NULL DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=utf8mb4;
```

```
INSERT INTO `test`.`test_date` (`date`) VALUES ('11:40:49');
INSERT INTO `test`.`test_date` (`date`) VALUES
('11:40:49.449');
INSERT INTO `test`.`test_date` (`date`) VALUES
('11:40:49.500');
SELECT * FROM test.test_date;
```

time 操作演示

```
mysql> DROP TABLE `test`.`test_date`;
Query OK, 0 rows affected (0.16 sec)

mysql> CREATE TABLE `test_date` (
    ->   `id` int(11) NOT NULL AUTO_INCREMENT,
    ->   `date` time NULL DEFAULT NULL,
    ->   PRIMARY KEY (`id`)
    -> ) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT
CHARSET=utf8mb4;
Query OK, 0 rows affected (0.22 sec)

mysql>
mysql> INSERT INTO `test`.`test_date` (`date`) VALUES
('11:40:49');
Query OK, 1 row affected (0.04 sec)

mysql> INSERT INTO `test`.`test_date` (`date`) VALUES
('11:40:49.449');
Query OK, 1 row affected (0.04 sec)

mysql> INSERT INTO `test`.`test_date` (`date`) VALUES
('11:40:49.500');
Query OK, 1 row affected (0.09 sec)

mysql> SELECT * FROM test.test_date;
+----+----------+
| id | date     |
+----+----------+
|  4 | 11:40:49 |
|  5 | 11:40:49 |
|  6 | 11:40:50 |
```

```
+----+----------+
3 rows in set (0.00 sec)
```

增加时间精度设置

```
mysql> DROP TABLE `test`.`test_date`;
Query OK, 0 rows affected (0.02 sec)

mysql>
mysql> CREATE TABLE `test_date` (
        ->   `id` int NOT NULL AUTO_INCREMENT,
        ->   `date` datetime(1) DEFAULT NULL,
        ->   PRIMARY KEY (`id`)
        -> ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_general_ci;
Query OK, 0 rows affected (0.00 sec)

mysql>
mysql> INSERT INTO `test`.`test_date` (`date`) VALUES ('2022-
12-05 11:40:49');
Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO `test`.`test_date` (`date`) VALUES ('2022-
12-05 11:40:49.4');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO `test`.`test_date` (`date`) VALUES ('2022-
12-05 11:40:49.5');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO `test`.`test_date` (`date`) VALUES ('2022-
12-05 11:40:49.449');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO `test`.`test_date` (`date`) VALUES ('2022-
12-05 11:40:49.500');
Query OK, 1 row affected (0.00 sec)

mysql> select * from `test`.`test_date`;
+----+----------------------+
| id | date                 |
```

```
+----+----------------------+
|  1 | 2022-12-05 11:40:49.0 |
|  2 | 2022-12-05 11:40:49.4 |
|  3 | 2022-12-05 11:40:49.5 |
|  4 | 2022-12-05 11:40:49.4 |
|  5 | 2022-12-05 11:40:49.5 |
+----+----------------------+
5 rows in set (0.00 sec)
```

时间精度最长可以设置为 time(6)

```
DROP TABLE `test`.`test_time`;

CREATE TABLE `test`.`test_time` (
        `id` int NOT NULL AUTO_INCREMENT,
        `time` time(6) DEFAULT NULL,
        PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_general_ci;

INSERT INTO `test`.`test_time` (`time`) VALUES ('11:40:49');
INSERT INTO `test`.`test_time` (`time`) VALUES
('11:40:49.4');
INSERT INTO `test`.`test_time` (`time`) VALUES
('11:40:49.5');
INSERT INTO `test`.`test_time` (`time`) VALUES
('11:40:49.449');
INSERT INTO `test`.`test_time` (`time`) VALUES
('11:40:49.500');

select * from `test`.`test_time`;
```

timestamp 演示

```
DROP TABLE `test`.`test_timestamp`;
```

```
CREATE TABLE `test`.`test_timestamp` (
        `id` int NOT NULL AUTO_INCREMENT,
        `d1` timestamp DEFAULT NULL,
        `d2` timestamp(3) DEFAULT NULL,
        PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_general_ci;
INSERT INTO `test`.`test_timestamp` (`d1`,`d2`) VALUES
('2022-12-05 11:40:49','2022-12-05 11:40:49');
INSERT INTO `test`.`test_timestamp` (`d1`,`d2`) VALUES
('2022-12-05 11:40:49.4','2022-12-05 11:40:49.4');
INSERT INTO `test`.`test_timestamp` (`d1`,`d2`) VALUES
('2022-12-05 11:40:49.5','2022-12-05 11:40:49.5');
INSERT INTO `test`.`test_timestamp` (`d1`,`d2`) VALUES
('2022-12-05 11:40:49.449','2022-12-05 11:40:49.449');
INSERT INTO `test`.`test_timestamp` (`d1`,`d2`) VALUES
('2022-12-05 11:40:49.500','2022-12-05 11:40:49.500');
select * from `test`.`test_timestamp`;
```