# 1、数据库介绍与分类

## 1.1数据库介绍

### 1.1.1、什么是数据库？

简单的说，数据库（database）就是一个存放数据的仓库，这个仓库是按照一定的数据结构（数据结构是指数据的组织形式或数据之间的联系）来组织、存储的，我们可以通过数据提供的多种方法来管理数据库里的数据。

## 1.2 数据库的种类

按照早起的数据库理论，比较流行的数据库模型有三种，分别为层次式数据库、网络式数据库和关系型数据库，而在当今的互联网中，最常用的数据库模型主要是两种，即关系型数据库和非关系型数据库。

### 1.2.1 关系型数据库介绍

#### 1） 关系型数据库由来

虽然网状数据库和层次数据库已经很好的解决了数据的集中和共享问题，但是在数据独立性和抽象级别上仍有很大欠缺。用户在对这两种数据库进行存取时，仍然需要明确数据的存储结构，支出存取路径。而关系数据库就可以较好的解决这些问题。

#### 2）关系型数据库介绍

关系型数据库模型是把复杂的数据结构归结为简单的二元关系（即二维表格形式）。例如某单位的同时关系就是一个二元关系。在关系数据库中，对数据的操作几乎全部建立在一个或多个关系表格上，通过对这些关系的表格分类、合并、连接或选取等运算来实现数据的管理。

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

#### 3）关系型数据库表格之间的不安息举例



#### 小结：什么是关系型数据库

1、二维表格

2、市场占有量较大的为MySQL、oracle数据库，互联网运维最常用的就是MySQL。

3、通过SQL结构化查询语言来存取、管理数据。

4、在保持数据一致性方面很强。ACID理论（后期会讲）

### 1.2.2 非关系型数据库介绍

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

随着互联网web2.0网站的兴起，传统的关系型数据库在应付web2.0网站，特别是对于规模日益扩大的海量数据，超大规模和高并发的微博、微信、sns类型的微博2.0纯动态网站已经显得力不从心，暴露了很多难以克服的问题，例如：传统的关系型数据库IO瓶颈、性能瓶颈都难以有效突破，于是开始出现了大批针对特定场景，以高性能和使用便利为目的功能特异化的数据库产品，nosql（非关系型）累的数据库就是在这样的情景中诞生并得到了非常迅速的发展。

Nosql是非关系型数据库的广义定义。它打破了长久以来关系型数据库与ACID理论打一统的局面。Nosql数据存储不需要固定的表结构，通常也不存在连接操作。在大数据存取上具备关系型数据库无法比拟的性能优势。该术语在2009初二道了广泛认同。

当今的用用体系结构需要数据存储在横向伸缩性上能够慢走需求，而nosql存储就是为了实现这个需求而诞生的。Google的bigtable与Amazon的dynamo是非常成功的商业NoSQL实现。一些开源的nosql体系，如Facebook的Cassandra，Apache的Hbase，也得到了广泛认同，redis，mongodb也逐渐的越来越受到各类大中小型公司的欢迎和追捧。

#### Nosql非关系型数据库小结：

1、nosql不是否定关系数据库，而是作为关系数据库的一个重要补充。

2、nosql为了高性能、高并发而生，忽略影响高性能、高并发的功能。

3、nosql典型产品memcached（纯内存）、rcdis（持久化缓存），mongodb

### 1.2.3 非关系型数据库种类：

#### 1）键值（key-value）存储数据库

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

键值数据库主要是使用一个哈希表，这个表中有一个特定的键和一个指针指向特定的数据。Key/value模型对于IT相同来说的优势在于简单、易部署、高并发。

典型产品：memcached、rediscachedb、BerkeleyDB

#### 2）列存储（column-oriented）数据库

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

这部分数据库通常是用来应对分布式存储的海量数据。键仍然存在，但是它们的特点是指向了多个列。这些列是由列家族来安排的。

典型产品：Cassandra，hbase。

#### 3）面向文档（document-oriented）的数据库

文档型数据库的灵感是来自于lotus notes办公软件的，而且它同第一种键值存储相类似。该类型的数据库模型是版本化的文档，半结构化的文档以特定的格式存储，比如json，文档型数据库可以看做是键值数据库的升级版，允许之间嵌套键值。而且文档型数据库比键值数据库的查询效率更高。

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

典型产品：mongodb、couchdb

#### 4）图形（graph）数据库

图形数据库允许沃恩将数据以图的方式存储。实体会被作为顶点，而实体之间的关系则会被作为边。比如我们有三个实体，Steve jobs、Apache和next，则会有两个“founded by”的边将Apple和next连接到Steve jobs。

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

典型产品：Neo4J、infogrid

## 1.3 常用关系型数据库产品介绍

### 1.3.1 Oracle数据库



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

Oracle公司是目前全球最大的数据库软件公司，也是近年业务增长即为迅速的软件提供与服务商。

2007.7.12，甲骨文公司在古国纽约宣布瑞出数据库oracle llg，这是oracle数据库的最新版本。

介绍说，oracle llg有400多项功能，经过了1500万个小时的测试，开发工作量达到了3.6万人/月。Oracle llg在安全，XML DB，备份等方面得到了很大提升。

主要应用范围：传统大企业，大公司，政府，金融，证券等等。

版本升级：oracle8i，oracle9i，oracle10g，oracle11g

### 1.3.2 mysql数据库



Mysql数据库是一个中小型关系型数据库管理系统，软件开发者为瑞典mysql AB公司。在2008年1月16号被sun公司收购，后sun公司又被oracle公司收购。目前MySQL被广泛的应用在Internet上的大中小型网站中。由于其体积小、速度快、总体拥有成本低，尤其是开放源码这一特点，许多大中小型网站为了降低网站总体拥有成本而选择了MySQL作为网站数据库，甚至国内知名的淘宝网也选择启用oracle而更换为更开放的MySQL。

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

### 1.3.3 MariaDB数据库



MariaDB数据库管理相同时MySQL数据库的一个分支，主要由开源社区维护，采用GPL授权许可。开发这个MariaDB数据库分支的可能原因之一是：甲骨文公司收购了MySQL后，又将MySQL闭源的潜在风险，因此MySQL开源社区采用分支的方式来避开这个风险。

开发MariaDB数据库的目的是完全兼容MySQL数据库，包括API和命令行，使之能轻松的称为MySQL的代替品。在存储引擎方面，使用xtraDB来代替MySQL的innoDB,innoDB由MySQL的常使人Michael widenius主导开发。早前曾以10亿美元的价格，将自己创建的公司MySQL AB卖给了SUN，随后，随著SUN被甲骨文收购，MySQL的所有权也落入oracle的手中。MariaDB数据库的名称来自MySQL创始人Michael widenius的女儿Maria的名字。

MariaDB基于事务的Maria存储引擎，替换了MySQL的myisam存储引擎，它使用了percona的xtraDB。这个版本还包括了primeBase XT和federatedX存储引擎。

### 1.3.4 SQL server数据库

Microsoft SQL server 是微软公司开发的大型关系型数据库系统。Sql server的功能比较全面，效率高，可以作为中型企业或单位的数据库平台。SQL server可以与Windows操作系统紧密集成，不论是应用程序开发速度还是系统事务处理运行速度，都能得到较大的提升。对于在Windows平台上开发的各种企业级信息管理系统来说，不论是C/S（客户机/服务器）架构还是B/S（浏览器/服务器）架构，SQL server都是一个很好的选择。**SQL server的缺点是只能在Windows系统下运行。**

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

### 1.3.5 Access 数据库

Microsoft公司与1994年推出的微机数据库管理系统。它具有界面友好、易学易用、开发简单、接口灵活等特点，是典型的新一代桌面关系数据库管理系统。它结合了Microsoft jet database engine和图形用户界面两项特点，是Microsoft office的成员之一。和office办公软件绑定的入门级小型桌面数据库，也是关系型数据库。

Access是入门级小型桌面数据库，性能安全性都很一般，可供个人管理或小型网站使用。

Access不是数据库语言，知识一个数据库程序。目前最新版本为office 2007.其主要特点如下：

1、完善的管理各种数据对象，具有强大的数据组织、用户管理、安全检查等功能。

2、强大的数据处理功能，在一个工作组级别的网站环境中，使用access开发的多用户数据库管理系统具有传统的XBASE（DBASE、fox base统称）数据库系统所无法实现的客户服务器（cient/server）结构和相应的数据库安全机制，access具备了许多先进的大型数据库管理系统所具备的特征，如事务处理/出错回滚能力等。

早起应用领域：小型程序系统asp+access系统，留言板，校友录等。

### 1.3.6 其他不常用关系型数据库

DB2，PostgreSQL,Informix,Sybase等。这些关系型数据库逐步的淡出了普通运维的视线，特别是互联网公司几乎见不到。

## 1.4 常用非关系型数据库产品介绍

### 1.4.1 Memcached （key-value）



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

由于memcached为纯内存缓存软件，一旦重启所有数据都会丢失，因此，新浪网基于开发了一个开源项目memcachedDB。通过为memcached增加Berkeley DB的持久化存储机制和异步主辅复制机制，使memcached具备了事务恢复能力、持久化数据存储能力和分布式复制能力，memcached非常适合需要超高性能读写速度、持久化保存的应用场景，但是资金几年逐渐被其他的持久化产品替代，例如redis。

小结：

1、key-value型数据库。

2、纯内存数据库。（重启数据就会丢失）

3、较早的非关系型数据库 memcachedDB（新浪开发sina）

### 1.4.2 Redis（key-value）



和memcached类似，redis也是一个key-value型存储系统。但redis支持的存储value类型相对更多，包括string（字符串）、list（链表）、set（集合）和zset（有序集合）等。这些数据类型都支持push/pop、add/remove及取交集、并集和差集更丰富的操作，而且这些操作都是原子性的。在此基础上，redis支持各种不同方式的排序。与memcached一样，为了保证效率，redis的数据都是缓存在内存中。区别是redis会周期性的把更新的数据写入磁盘或者修改操作写入追加的记录文件，并且在此基础上实现了master-slave（主从）同步。

Redis是一个高性能的key-value数据库。Redis的出现，很大程度补偿了memcached这类key/value存储的不足，在部分场合可以对关系数据库起到很到的补充作用。它提供了Python，ruby，erlang，PHP客户端，使用很方便。

#### 特点：

1、支持内存缓存，这个功能相当于memcached。

2、支持持久化存储，这个功能相当于memcachedDB，ttserver

3、数据类型更丰富，比其他key-value库功能更强。

4、支持主从集群，分布式。

5、支持队列等特殊功能。

应用：缓存从存取memcached更改存取redis。

### 1.4.3 mongoDB（document-oriented）



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

主要功能特性：

1、面向集合存储，易存储对象类型的数据。

“面向集合”（collenction-orented），意思是数据被分组存储在数据集中，被称为一个集合（collenction）。每个集合在数据库中都有一个唯一的标识名，并且可以包含无限数目的文档。集合的概念类似关系型数据库（rdbms）里的表（table），不同的是它不需要定义任何模式（schema）

2、模式自由

模式自由（schema-free），意味着对于存储在mongodb数据库中的文件，我们不需要知道它的任何结构定义。如果需要，完全可以把不同的结构的文件存储在同一个数据库里。

3、支持动态查询。

4、支持完全索引，包含内部对象。

5、支持查询。

6、支持复制和故障恢复。

7、使用高效的二进制数据存储，包括大型对象（如视频等）。

8、自动处理碎片，以支持云计算层次的扩展性。

9、支持ruby，Python，Java，C++，PHP等多种语言。

10、支持存储格式为BSON

11、可通过网络访问

### 1.4.4 Cassandra

Apache Cassandra 是一套开源分布式key-value存储系统。它最初由Facebook开发，用于存储特别大的数据。Facebook目前在使用此系统。

主要特性：

1、分布式

2、基于column的结构化

3、高伸展性

Apache Cassandra 的主要特点就是他不是一个数据库，而是由一堆数据库节点共同构成的一个分布式网络服务，对Cassandra的一个写操作，会被复制到其他节点上去，对Cassandra的读操作，也会被路由到某个节点上面去读取。对于一个Cassandra集群来说，扩展性能是比较简单的事情，直观在群集里添加节点就可以了。

### 1.4.5 其他不常用非关系型数据库

HBase,memcachedDB，berkelcyDB、Tokyo cabinet\Tokyo Tyrant(Hserver)

# 2、MySQL数据库入门知识

## 2.1 MySQL数据库介绍

MySQL是互联网领域里一款最重要的、深受广大用户欢迎的开源关系型数据库软件之一，由软垫MySQL AB公司开发与维护。2006年，MySQL AB公司被sun公司收购，2008年，sun公司又被传统数据库领域大佬甲骨文（oracle）公司收购，因此，MySQL数据库软件目前属于oracle公司，称为传统数据库领域老大的又一个数据库产品，甲骨文公司收购MySQL后，使得自身在商业数据库与开源数据库领域市场占有份额都跃居第一的位置，这样的格局，引起了业内很多的人士的担忧，这种担忧直接导致后来的MySQL分支数据库MariaDB的诞生与发展。

### 2.1.1 MariaDB数据库诞生背景介绍

自甲骨文送死收购了MySQL数据库产品后，引起了很多业界人士担忧，因为商业数据库的老大有可能将MySQL闭源的潜在风险，为了避免oracle将MySQL闭源，而无开源的类MySQL数据库可用，MySQL社区采用分支的方式来避开这个风险。因此MariaDB数据库就这样诞生了，MariaDB是一个向后兼容、可能在以后替代MySQL的数据库的一个不错的新的数据库产品，MariaDB数据库可以先测试，等产品应用人员做多一些，社区更活跃后再考虑使用为好。

### 2.1.2、为什么选择MySQL数据库的原因：

1）MySQL性能卓越，服务稳定。很少出现异常宕机。

2）MySQL开放源代码且无版权制约，自主性及使用成本低。

3）MySQL历史悠久，社区及用户非常活跃，遇到问题，可以寻求帮助。

4）MySQL软件体积小，安装使用简单，并且易于文虎，安装及维护成本低。

5）MySQL品牌效应，使得企业无需考虑就直接用之，LAMP，LEMP流行架构

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

## 2.2 MySQL数据库的官方网站

MySQL数据库的官方网站为：<http://www.mysql.com>，其发布的MySQL版本采用双授权政策，和大多数开源产品的路线一样，分为社区版和商业版，而这两个版本又各自分四个版本依次发布，这四个版本为：alpha版、beta版、RC版和GA版本。

### 2.2.1 MySQL数据库商业版也社区版区别

#### 1）MySQL数据库商业版发布介绍

MySQL商业版是由MySQL AB公司内部专门的人员负责开发及维护，但同时也会吸纳社区人员编写的优秀代码及算法，并且由专门的人员严格按照软件测试流程对这些采纳的代码进行严格测试，没有问题再进行发布。简单的说，MySQL商业版是由MySQL公司内部发布，同时参考社区版的先进代码功能和算法，是MySQL公司盈利产品，需要付费才能使用及提供服务支持，稳定性和可靠性无疑都是最好的，当然，企业腰包要够鼓才能买得起。

#### 2）MySQL数据库社区版本发布介绍

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

#### 3）MySQL数据库商业版与社区版之间的区别

#### MySQL商业版与社区版之间的区别在于：

1、商业版组织管理与测试关节控制更严格，稳定性方面，回避社区版本更稳定。

2、MySQL是成熟产品，商业版与社区办之间性能方面相差不大。

3、商业版不遵守GPL协议，社区办遵守了GPL协议可以免费试用。

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

5、社区版本的维护服务只能靠社区提供，无法像商业版本获得故障及补丁解决服务了，但是，使用社区版死活完全免费的方式，社区版的服务质量与时效性等方面就无法与MySQL AB公司提供的服务相比了。

### 2.2.2 MySQL数据库四种发布版本介绍

MySQL的版本发布采用双授权政策，即分为社区版和商业版，而这两个版本又各自分为四个版本一次发布：alpha版、beta版、RC版和GA版。四种发布版本之间的说明及区别，如下：

#### Alpha版

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

#### Beta版

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

#### RC版

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

#### GA版

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

## 2.3 MySQL产品路线

### 2.3.1 MySQL产品路线变更历史背景

早起，MySQL也是遵循版本号主键增加的方式发展，格式例如：mysql-X.XX.XX.tar.gz，例如，DBA都非常熟悉的生产场景版本：4.1.7、5.0.56等。

近几年，为了提高MySQL产品的竞争优势、以及提高性能、降低开发维护成本等原因，同时，更方便企业用户更精准的选择适合的版本产品用于自己的企业生产环境中，MySQL在发展到5.1系列版本之后，重新规划为三条产品线。

### 2.3.2 第一条-5.0.xx到5.1.xx产品线系列介绍（最正宗的后代）

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

MySQL5.1是当前稳定（产品质量）发布系列，只针对漏洞修复中心发布；没有增加会影响稳定性的新功能。

\* MySQL 5.1：previous stable（production-quality）release

MySQL5.0是前一稳定发布系列，只针对严重漏洞修复和安全修复重新发布；没有增加会影响该系列的重要功能。

\* MySQL 5.0：older stable release nearing the end of the product lifecycle

MySQL4.0和3.23是旧的稳定发布系列。该版本不在使用，新的发布只用来修复特别严重的漏洞。  
  
社区版：  
第一条最正宗产品线：根正苗红（yum安装的默认版本）  
版本：4.0--5.0--5.1--5.2  
2.3.3 5.4.XX开始---到5.7.XX产品系列介绍  
5.4.XX开始---到5.7.XX产品系列介绍。是为了更好的整合MySQL。AB公司社区和第三方公司开发的新存储引擎，以及吸收新的实现算法等，从而更好地支持SMP架构，提高性能而做了大量的代码重构。版本号从5.4.XX开始，目前发展到了5.6.X。  
MySQL 5.5：Latest General Availability  
（Production）release  
主流：互联网公司用mysql：5.5和5.6  
2.3.4 第三条6.0.XX-到7.1.XX产品线系列介绍  
6.0.XX-到7.1.XX产品线系列介绍  
为了更好的推广MySQL Cluster版本，以及提高MySQL Cluster的性能和稳定性，以及功能改进和增加，以及改动MySQL基础功能，使其对Cluster储存引擎提供更有效的支持与优化。版本号为6.0.XX开发，目前大战大7.1.XX。  
  
操作使用MySQL版本5.5.32  
安装MySQL：  
linux软件的安装方式：  
1）yun/rpm  简单，快，无法定制  
2）编译安装，./configure；make；make install。复杂，速度慢，可定制。  
     针对MySQL,第一条产品线的编译方式5.0--5.1  
     mysql 5.5以上  
     编译安装，./cmake；gmake；gmake install  
3）二进制包&&&&&  
     直接解压，初始化数据库即可，类似绿色软件，无需安装  
     简单，快，不好定制。

## 2.4 MySQL数据库软件命名介绍

MySQL数据库软件的名字是由3个数字和一个后缀组成的版本号。例如。mysql-5.0.56.tar.gz的版本号是这样解释：

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

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

3）第三个数字（56）是在此发行系列的版本号，随每个新分发版递增。通常你需要已经选择的发行（release）的最新版本。

每次更新后，版本字符串的最后一个数字递增。如果相对于前一个版本增加了新功能或有微小的不兼容性，字符串的第二个数字递增。如果文件格式改变，第一个数字递增。

4）后缀显示发行的稳定性级别。通过一系列后缀显示如何改进稳定性。可能的后缀有：

##### alpha版

表名发行包含大量未被称帝测试的新代码。已知的缺陷应该在新闻小结被记录。在大多数alpha版本中也有新的命令和扩展。alpha版本也可能有主要代码更改等开发。但我们在发布前一定对其进行测试。

##### beta版

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

##### RC版

是发布代表；是一个发型了一段时间的beta版本，看起来应该运行正常。只增加了很小的修复（发布代表即以前所称的gamma版）

##### 没有后缀

例如：mysql-5.0.56.tar.gz，这意味着该版本语句在很多地方运行一段时间了，而且没有非平台特定的缺陷报告。这就是我们称为一个产品（稳定）或“通用”版本的东西。

2.5 生产产经如何选择MySQL版本

#### MySQL版本选择最终建议：

企业生产场景选择MySQL数据库建议：

1、稳定版：选择开源的社区版的稳定版GA版本。

2、产品线：可以选择5.1或5.5.互联网公司主流5.5，其次是5.1和5.6

3、选择MySQL数据库GA版发布后6个月以上的GA版本。

4、要选择前后几个月没有大的bug修复的版本，而不是大量修复bug的集中版本。

5、最好 较长时间没有更新发布的版本。

6、要考虑开发人员开发程序使用的版本是否兼容你选的版本。

7、作为内部开发测试数据库环境，跑大概3-6个月的时间。

8、优先企业非核心业务采用新版本的数据库GA版本软件。

9、向DBA高手请教，或者在及技术氛围好的群里和大家一起交流，使用真正的高手们用过的GA版本产品。

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

# 3、MySQL数据库安装方法

## 3.1 MySQL数据库安装

在当今的大中型互联网企业里，MySQL数据库服务几乎都是运行在Linux系列操作系统上，淡然，也可以运行在windows/Unix等商业操作系统上，大众型互联网企业使用开源领域的产品木有洒落数据库目的是，软件代码的开放性和以及无版权免费应用节约成本，因此，拓建中小企业采用lamp/lnmp网站架构，后面的数据库知识也是以Linux（centos6.6）平台为例俩讲述如何安装及使用MySQL软件的。

不同场景不同的MySQL的安装可以有多重方法。

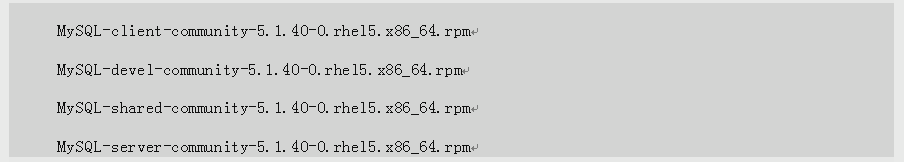
### 3.1.1 yum/rpm方式安装MySQL

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

注意：yum/rpm方式安装MySQL适合所有MySQL软件产品系列。

###### 1）rpm包方式安装MySQL

rpm包的安装非常简单，以mysql-5.1.40为例，可以光盘或者网上下载到四个安装包：



其中，我们必须安装MySQL-server\*和MySQL-client\*软件包，对于另外两个则可视实际需要，但是建议一起安装，为此我们可以把这四个RPM包至于有个目录中，然后执行shell目录：rpm –ivh MySQL\_\* 则可以完成MySQL软件安装任务，剩下的就是如何配置的问题了。

###### 2）yum方式安装MySQL

yum方式安装MySQL安装极其简单，只要执行yum install mysql-server即可。

yum/rpm方式安装MySQL应用场景

yum/rpm安装适合对数据库要求不太高的场合，例如并发不大，公司内部，企业内部的一些应用场景。

大的内湖把源码格局企业的需求制作成rpm，搭建yum仓库，yum install xxx –y。

### 3.1.2 常规方式编译安装MySQL

第一条最正宗的产品线5.1及以前：

所谓常规方式编译安装MySQL就是延续早期MySQL的3部曲安装方式，即./configure;make;make install，撑场场景的具体命令及参数为：

./configure \

--prefix=/application/mysq5.1.72 \

--with-unix-socket-path=/application/mysql5.1.72/tmp/mysql.sock \

--localstatedir=/application/mysql5.1.72/data \

--enable-assembler \

--enable-thread-safe-client \

--with-mysqld-user=mysql \

--with-big-tables \

--without-debug \

--without-debug \

--with-pthread \

--enable-assembler \

--with-extra-chrsets=complex \

--with-readline \

--with-ssl \

--with-embedded-server \

--enable-local-infile \

--with-plugins=partition,innobase \

--with-mysqld-ldflags=-all-static \

--with-client-ldflags=-all-static

make

make install

此种方式适合所有MySQL5.0.xx-5.1.xx产品系列，是最常规的编译方式。

### 3.1.3 采用cmake方式编译安装MySQL

由于MySQL5.5.xx-5.6.xx产品系列特殊性，所以编译方式也和早期的产品安装方式不同，采用cmake或gmake方式编译安装。即./cmake;make;make install，生产场景的具体命令及参数为：

tar zxf mysql-5.5.32.tar.gz

cd mysql-5.5.32

cmake . -DCMAKE\_INSTALL\_PREFIX=/application/mysql-5.5.32 \  
-DMYSQL\_DATADIR=/application/mysql-5.5.32/data \  
-DMYSQL\_UNIX\_ADDR=/application/mysql-5.5.32/tmp/mysql.sock \  
-DDEFAULT\_CHARSET=utf8 \  
-DDEFAULT\_COLLATION=utf8\_general\_ci \  
-DEXTRA\_CHARSETS=gbk,gb2312,utf8,ascii \  
-DENABLED\_LOCAL\_INFILE=ON \  
-DWITH\_INNOBASE\_STORAGE\_ENGINE=1 \  
-DWITH\_FEDERATED\_STORAGE\_ENGINE=1 \  
-DWITH\_BLACKHOLE\_STORAGE\_ENGINE=1 \  
-DWITHOUT\_EXAMPLE\_STORAGE\_ENGINE=1 \  
-DWITHOUT\_PARTITION\_STORAGE\_ENGINE=1 \  
-DWITH\_FAST\_MUTEXES=1 \  
-DWITH\_ZLIB=bundled \  
-DENABLED\_LOCAL\_INFILE=1 \  
-DWITH\_READLINE=1 \  
-DWITH\_EMBEDDED\_SERVER=1 \  
-DWITH\_DEBUG=0

#--build files have been written to /home/oldboy/tools/mysql-5.5.32

提示，编译时可配置的选项很多，具体可参考官方文档

make

make install

如果上述操作为出现错误，则MySQL5.5.32软件cmake方式安装就算成功了。

### 3.1.4 采用二进制方式免编译安装MySQL

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

5.1.5如何正确选择MySQL的安装方式

yum/rpm安装适合对数据库要求不太高的场合，例如并发不大，公司内部，企业内部的一些应用场景。二进制免安装比较简方便，适合5.0-5.1和5.5-5.6系列，是很多专业DBA的选择，普通Linux与呢我人员多采用编译的方式，5.0-5.1系列就是常规编译方式，5.5-5.6系列就是cmake编译方式。

所以综合来讲，这些安装方式都是可以使用的，只是不同层次的人的习惯不同，性能差距不是很大。

推荐选择：首先是mysql5.5及以上版本，安装方式及其数量少的话，推荐cmake编译方式，这是大多数运维人员的选择。数量多的话就用二进制免安装，也是没有问题的，这是某写DBA的偏爱，数量特别大的情况，可以选择定制rpm包的安装方式。

# 4、MySQL多实例

4．1 MySQL多实例介绍  
简单说就是，在一台机器开启多个不同的服务端口（如3306,3307），同时运行 多个MySQL服务进程，这些服务进程通过不同的socket监听不同的服务端口来 提供各自的服务。  
这些MySQL多实例同用一套MySQL安装程序，使用不同的my.cnf（也可以相同）的配置文件、启动程序（也可以相同），数据文件。在提供服务时，多实例MySQL在逻辑上看来是各自独立的，多实例的自身是根据配置文件的对应设定值，来取得服务器的相关硬件资源多少。  
作个比喻，MySQL多实例相当于房子的多个卧室一样，每个实例可以看做一间卧室，整个服务器就是一套房子，服务器的硬件资源（cpu，mem，disk）可以看做房子的卫生间、厨房、厅一样，是房子的公用资源，北漂伙伴们在一起，休息在自己卧室，出来活动肯定要共用上述共用资源。这样想我们就应该明白了MySQL多实例了吧。  
多实例补充：其实很多服务都可以配置多实例的，在门户网站用得更广泛，例如：nginx就可以多实例，apache，haproxy，redis，memcache，都可以多实例。



## 4.2 Mysql多实例在作用与问题

**有效利用服务器资源**

当单个服务器资源有剩余时，可以充分利用剩余在资源提供更多在服务，且可以实现资源的逻辑隔离。

**节约服务器资源**

当公司资金紧张，但是数据库又需要和各自尽量独立的提供服务，而且，需要只从复制等技术是，多实例就再好不过了。

MySQL多实例有它的好好处，但也有其弊端，比如，会存在资源互相抢占的问题。

**资源互相抢占问题**

当某个数据库实力并发很高或者有SQL慢查询时，整个实力会消耗整个更多的内存、CPU、磁盘IO资源，导致服务器上的其他的 数据库实例提供服务的质量一起下降。这就相当于大家住在一个房子的不同卧室一样，早晨起来上班，都要刷牙、洗脸等，这样卫生间就会长期占用，其他人就要等待一样的道理。不同实例获取的资源是相对独立的，无法像虚拟化一样完全隔离。

## 4.3 多实例的生产应用场景

**1、资金紧张型公司**

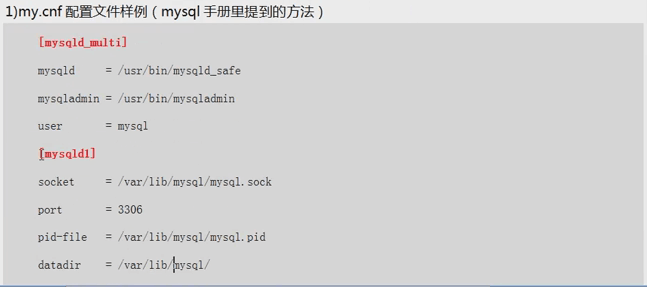
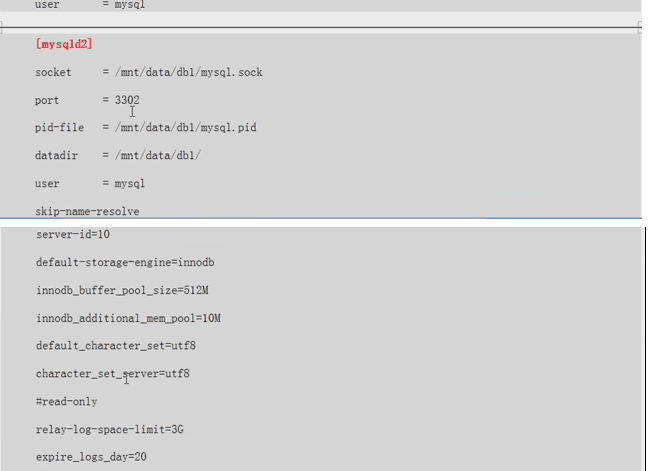
当公司业务访问量不太大, 又舍不得花钱, 但又希望不同业务的数据库服务各自尽量独立的提供服务互相不受影响;而且,需要主从复制等技术提供备份或读写分离服务时, 多实例就再好不过了 。如: 可以通过3台服务器部署6-9个实例, 交叉做主从同步各份及写分离,实现6-9台服务器才有的效果。这里要强调的是,所谓的尽量独立是相对的。

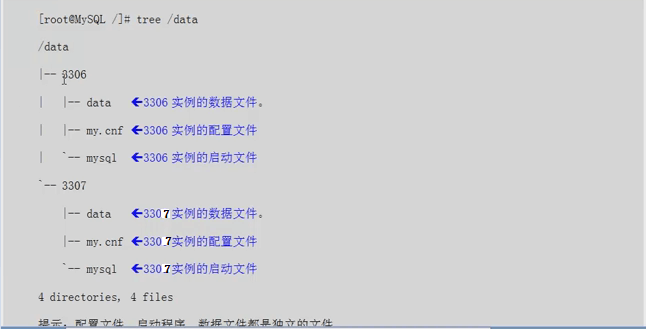
**2、并发访问不是特别大的业务**

当公司业务访问量不太大的时候1 服务器的资源基本都是浪费的, 这时就很适合多实例的应用, 如果对SQL语句优化做的比较好, MySQL多实例一个很值得使用的技术, 即使并发很大,合理分配好系统资源以及打配好服务,也不会有太大问题。

**3、门户网站应用 MySQL多实例场景**

门户网站通常都会使用多实例，因为配置硬件好的服务器，可节省IDC机柜空间、同时，跑多实例也会减少硬件资源跑不满的浪费。比如，百度搜索引擎的数据库就是多实例, 一般是从库, 例如某部门例子, IBM服务器为48核 CPU, 内存96G,跑3-4个实例; sina网也是用的多实例, 内存48G左右。门户网站使用多实例的目的是配硬件好的服务器, 节省 IDC机柜空间, 同时, 跑多实例让硬件资源不浪费。（磁盘多，性能高）

4.4、MySQL多实例常见配置方案  
4.4.1单一配置文件、单一启动程序多实例部署方案（官方的文档提供的配置）  
1）配置文件  
  
  
2）启动程序  
D:\qq\343012818\Image\C2C\$1[VM`Q_5D4Y)$PZV_B7`VN.png  
3）此方案说明  
缺点：耦合性太高，一个配置文件，不好管理。工作开发和运维的统一原则：降低耦合点。  
D:\qq\343012818\Image\C2C\{[%W379Z8@V((W%QV7[]5GS.png

4.4.2多配制文件、多启动程序部署方案  
 多配制文件、多启动程序部署方案，是通过配置多个配置文件及多个启动程序来实现多实例的方案，这就是推荐的方法。  
本文的配置：  
提示：这里的配置文件my.cnf、启动程序mysql都是独立的文件，数据文件data目录也是独立的。

## 4.5 安装MySQL多实例

###### 4.5.1安装MySQL需要的依赖包和编译软件

1）安装MySQL需要的依赖包

安装MySQL前，最好先安装MySQL需要的依赖包，不然后面会出现很多报错信息，到时候还得再回来安装MySQL的依赖包。安装目录如下：

[root@db01 ~]# yum install -y ncurses-devel

[root@db01 ~]# rpm -qa ncurses-devel libaio-devel

ncurses-devel-5.7-4.20090207.el6.x86\_64

libaio-devel-0.3.107-10.el6.x86\_64

提示：安装后检查，如果出现两行信息标识安装成功。

2）安装编译MySQL需要的软件

yum install –y cmake

4.5．2 开始安装MySQL

为了学习更多的MySQL技术，此次选择了相对复杂的源代码安装为例，MySQL多实例的安装，大型公司一般都会讲MySQL软件定制成rpm包，然后放到yum仓库里，使用yum安装，中小企业里的二进制和编译安装的区别不大。

1）建立MySQL用户账号

首先以root身份登录到Linux系统中，然后执行如下命令创建mysql用户账号：

[root@db01 ~]# useradd -s /sbin/nologin -M mysql

[root@db01 ~]# id mysql

uid=501(mysql) gid=501(mysql) 组=501(mysql)

2）获取MySQL软件包

[root@db01 ~]# mkdir -p /home/oldboy/tools

[root@db01 ~]# cd !$

rz –y

tar xf mysql-5.5.32.tar

cd mysql-5.5.32

cmake . -DCMAKE\_INSTALL\_PREFIX=/application/mysql-5.5.32 \

-DMYSQL\_DATADIR=/application/mysql-5.5.32/data \

-DMYSQL\_UNIX\_ADDR=/application/mysql-5.5.32/tmp/mysql.sock \

-DDEFAULT\_CHARSET=utf8 \

-DDEFAULT\_COLLATION=utf8\_general\_ci \

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

-DENABLED\_LOCAL\_INFILE=ON \

-DWITH\_INNOBASE\_STORAGE\_ENGINE=1 \

-DWITH\_FEDERATED\_STORAGE\_ENGINE=1 \

-DWITH\_BLACKHOLE\_STORAGE\_ENGINE=1 \

-DWITHOUT\_EXAMPLE\_STORAGE\_ENGINE=1 \

-DWITHOUT\_PARTITION\_STORAGE\_ENGINE=1 \

-DWITH\_FAST\_MUTEXES=1 \

-DWITH\_ZLIB=bundled \

-DENABLED\_LOCAL\_INFILE=1 \

-DWITH\_READLINE=1 \

-DWITH\_EMBEDDED\_SERVER=1 \

-DWITH\_DEBUG=0

make

make install

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

下面为MySQL安装路径设置不带版本号的软链接/application/mysql，操作步骤如下：

[root@db01 ~]# ln -s /application/mysql-5.5.32/ /application/mysql

如果上述操作未出现错误，查看/application/mysql/目录下有内容，则MySQL5.5.32源代码包采用cmake方式的安装就算成功了。

4.5.3 创建MySQL多实例的数据文件目录

在企业中，通常以/data目录作为MySQL多实例总的根目录，然后规划不同的数字（即MySQL实例端口号）作为/data下面的二级目录，不同的二级目录对应的数字就作为MySQL实例的端口号，以区别不同的实例，数字对应的二级目录下包含mysql的数据文件、配置文件以及启动文件等。

下面以配置3306、3307两个实例为例进行实战，创建MySQL多实例的目录如下：

[root@db01 ~]# mkdir -p /data/{3306,3307}/data

### 4.5.4 创建MySQL多实例的配置文件

MySQL数据库默认为用户提供了多个配置文件模板，用户可以根据服务器硬件配置的大小来选择。

[root@db01 mysql-5.5.32]# ls -l support-files/my\*.cnf

-rw-r--r-- 1 root root 4759 12月 27 15:34 support-files/my-huge.cnf

-rw-r--r-- 1 root root 19809 12月 27 15:34 support-files/my-innodb-heavy-4G.cnf

-rw-r--r-- 1 root root 4733 12月 27 15:34 support-files/my-large.cnf

-rw-r--r-- 1 root root 4744 12月 27 15:34 support-files/my-medium.cnf

-rw-r--r-- 1 root root 2908 12月 27 15:34 support-files/my-small.cnf

上面是单实例的默认配置文件模板，如果配置多实例，和单实例会有不同。为了让MySQL多实例之间彼此独立，因此，要为每一个实例建立一个my.cnf配置文件和一个启动文件mysql，让它们分别对应自己的数据文件目录data。

首先，通过vim命令来添加配置文件内容，命令如下：

不同的实例需要添加的my.cnf内容会有区别，具体见表，期中的配置由官方的配置模板修改而来，当然，在实际工作中我们是拿早已配置好的模板来进行修改的，可以通过rz等方式上传配置文件模板my.cnf文件到相关目录下。

rz –y data.zip

cd /

unzip data.zip

[root@db01 /]# tree data/

data/

|-- 3306

| |-- data

| |-- my.cnf

| `-- mysql

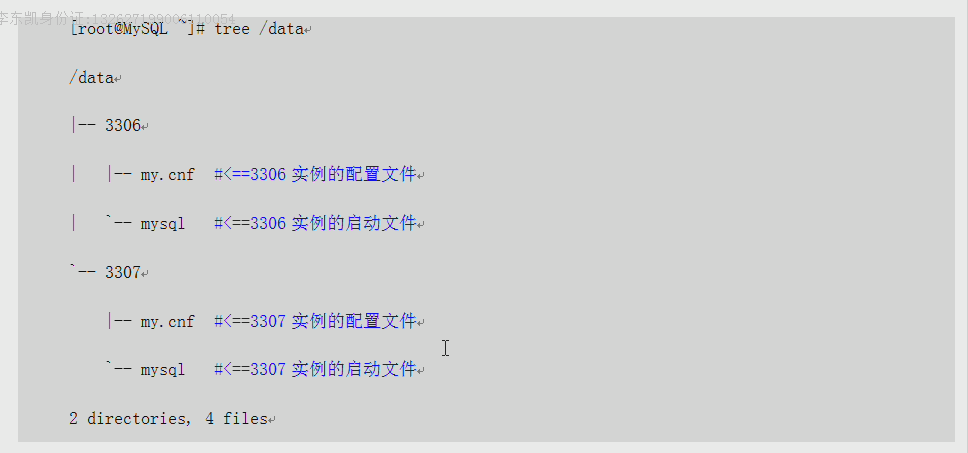
`-- 3307

|-- data

|-- my.cnf

`-- mysql

最终完成后的多实例根/data目录结果如下：



说明：在多实例启动文件中，启动MySQL不同实例服务所执行的命令实质是由区别的，例如，启动3306实例的命令如下：

mysqld\_safe –defaults-file=/data/3306/my.cnf 2>&1 > /dev/null &

**启动3307的命令如下：**

mysqld\_safe –defaults-file=/data/3307/my.cnf 2>&1 > /dev/null &

多实例启动文件中，停止MySQL不同实例服务的实质命令。

**停止3306实例命令如下：**

mysqladmin –uroot –poldboy123 –S /data/3306/mysql.sock shutdown

停止3307实例命令如下：

mysqladmin –uroot –poldboy123 –S /data/3307/mysql.sock shutdown

### 4.5.5 配置MySQL多实例的文件权限

1）通过下面的命令授权mysql用户和组管理整个多实例的根目录/data

[root@db01 /]# find /data/ -name "mysql"|xargs chmod +x

[root@db01 /]# chown -R mysql.mysql /data/

2）通过下面的命令授权mysql多实例所有启动文件的mysql可执行，设置700权限最佳，注意不要用755权限，因为启动文件里有数据库管理员密码，会被读取到。

[root@db01 /]# find /data –name mysql|xargs chmod 700

### 4.5.6 MySQL相关目录加入全局路径的配置

1、配置全局路径意义

如果不为MySQL的命令配置全局路径，就无法直接在命令行输入mysql这样的命令，只能使用全路径（/application/mysql/bin/mysql），这种带着路径输入命令的方式很麻烦。

2、配置MySQL全局路径的方法

1） 确认mysql命令所在路径

[root@db01 /]# ls /application/mysql/bin/mysql

/application/mysql/bin/mysql

2）在PATH变量前面增加/application/mysql/bin路径，并追加到/etc/profile文件中。

[root@db01 3306]# cp /application/mysql/bin/\* /usr/local/sbin/

### 4.5.7初始化MySQL多实例的数据库文件

上述步骤全部配置完毕后，就可以初始化数据库文件了，这个步骤其实也可以在编译安装MySQL之后就操作，只不过放到这里更合适一些。

1）初始化MySQL数据库

初始化命令为：

[root@db01 /]# cd /application/mysql/scripts/

[root@db01 scripts]# ./mysql\_install\_db --basedir=/application/mysql --datadir=/data/3306/data --user=mysql

[root@db01 scripts]# ./mysql\_install\_db --basedir=/application/mysql --datadir=/data/3307/data --user=mysql

[root@db01 scripts]# /data/3306/mysql start

[root@db01 scripts]# /data/3307/mysql start

[root@db01 scripts]# ss -lntup|grep 330

tcp LISTEN 0 600 \*:3306 \*:\* users:(("mysqld",41821,12))

tcp LISTEN 0 600 \*:3307 \*:\* users:(("mysqld",42539,11))

2）初始化数据库的原理及结果说明

初始化数据库的实质就是创建基础的数据库系统的库文件，例如：生成MySQL库表等。

### 4.5.8 启动MySQL多实例数据库

下面来看看启动mysql多实例的命令

第一个实例3306的启动命令为：

[root@db01 scripts]# /data/3306/mysql start

第二个实例3307的启动命令为：

[root@db01 scripts]# /data/3307/mysql start

检查MySQL多实例数据库是否成功启动：

[root@db01 scripts]# ss -lntup|grep 330

tcp LISTEN 0 600 \*:3306 \*:\* users:(("mysqld",41821,12))

tcp LISTEN 0 600 \*:3307 \*:\* users:(("mysqld",42539,11))

### 4.5.9MySQL多实例启动故障排错说明

如果MySQL多实例有服务没有被启动，排查办法如下：

1、如果发现没有显示MySQL对应实例的端口，请稍微等待几秒再检查，MySQL服务的启动比web服务等会慢一下。

2、如果还不行的话，请查看MySQL服务对应实例的错误日志，错误日志路径在my.cnf配置的最下面定义。例如，3306实例的错误日志为：

[root@db01 3306]# grep log-error my.cnf|tail -1

log-error=/data/3306/mysql\_oldboy3306.err

那么，可以执行tail-100 /data/3306/mysql\_oldboy3306.err检查MySQL错误日志：

[root@db01 3306]# tail -100 /data/3306/mysql\_oldboy3306.err

提示：

1、细看所有执行命令返回的输出，不要忽略关键的输出内容。

2、辅助查看系统日志/var/log/messages。

3、如果是MySQL关联了其他服务，要同时查看相关服务的日志。

4、仔细阅读，重新查看操作的步骤是否正确，书写的命令及字符是不是都对。

## 4.6配置及管理MySQL多实例数据库

1、配置MySQL多实例数据库开机自启动

服务的开机自启动很关键，MySQL多实例的启动也不例外，把MySQL多实例的穹顶命令加入/etc/rc.local，实现开机自启动：

[root@db01 3306]# echo "mysql multi instances" >>/etc/rc.local

[root@db01 3306]# echo "/data/3306/mysql start" >>/etc/rc.local

[root@db01 3306]# echo "/data/3307/mysql start" >>/etc/rc.local

[root@db01 3306]# tail -3 /etc/rc.local

mysql multi instances

/data/3306/mysql start

/data/3307/mysql start

2、登录MySQL测试

测试命令如下：

[root@db01 3306]# mysql -uroot -S /data/3306/mysql.sock

Welcome to the MySQL monitor. Commands end with ; or \g.

Your MySQL connection id is 6

Server version: 5.5.32-log Source distribution

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> show databases; #查看当前的数据库

+--------------------+

| Database |

+--------------------+

| information\_schema |

| mysql |

| performance\_schema |

| test |

+--------------------+

4 rows in set (0.04 sec)

mysql> select user(); #查看当前的登录用户。

+----------------+

| user() |

+----------------+

| root@localhost |

+----------------+

1 row in set (0.07 sec)

3、MySQL多实例数据库的管理方法

MySQL安装完成后，默认情况下，MySQL管理员的账号root是无密码的，登录不同的实例需要制定不同势力的sock路径及mysql.sock文件，这个mysql.sock是在my.cnf配置文件里指定的。

下面是无密码情况下登录数据库的方法，关键点是-S参数及后面指定的/data/3306/mysql.sock，注意，不同实例的sock虽然名字相同，但是路径是不同的，因此是不同的文件。

-S /data/3306/mysql.sock

-S /data/3307/mysql.sock

下面是重启对应实例数据库的命令。

/data/3306/mysql stop

/data/3306/mysql start

4、MySQL安全配置

MySQL管理员的账号root密码默认为空，极不安全，可以通过mysqladmin命令为mysql不同实例的数据库设置独立的密码。

[root@db01 3306]# mysqladmin -uroot -S /data/3306/mysql.sock password oldboy123

[root@db01 3306]# mysql -uroot -poldboy123 -S /data/3306/mysql.sock

5、多实例MySQL登录问题分析

1）多实例本地登录MySQL

多实例本地登录一般是通过socket文件来指定具体登录到哪个实例的，此文件的具体位置是在mysql编译过程或者my.cnf文件里指定的。在本地登录数据库时，登录程序会通过socket文件来判断登录的是哪个数据库实例。

例如：通过mysql –u –p –S /data/3307/mysql.sock可知，登录的是3307这个实例。MySQL.sock文件是MySQL服务端与本地MySQL客户端进行通信的Unix套接字文件。

2）远程连接登录MySQL多实例

远程登录MySQL多实例中的一个实例时，通过TCP端口（port）来指定所要登录的MySQL实例，此端口的配置是在mysql配置文件my.cnf中指定的。

例如：在mysql –uoldboy –poldoby –h 10.0.0.7 –P3307中，-P为端口参数，后面接具体的实例端口，端口是一种“逻辑连接位置”，是客户端程序被分派到计算机上特殊服务程序的一种方式，强调提前在10.0.0.7上对oldboy用户做了授权。

**MySQL基础安全**

1、启动程序设置700，属主和用户组为MySQL

2、为MySQL超级用户root设置密码。

3、如果要求严格可以删除root用户，创建其他管理用户，例如admin。

4、登录时尽量不要在命令行 暴露密码，备份脚本中过如果有密码，设置为700，属主和用户组为MySQL或root。

5、删除默认存在的test库

6、初始删除无用户的用户，只保留

7、授权用户对应的主机不要用%，权限不要给all，最小化授权，从库只给select。

8、不要一个用户管理所有的库，尽量专库专用户。

9、清理MySQL操作日志文件~/.mysql\_history

10、禁止开发获取到web连接的密码，禁止开发连接操作生成对外的库。

11、phpmyadmin安全

12、服务器禁止设置外网IP

13、防SQL注入（web），PHP.ini。

# 5．MySQL常用基础命令操作实战

## 5.1 启动与关闭MySQL

### 5.1.1 单实例MySQL启动与关闭方法

法一：常规启动关闭数据库方式（推荐）

1）常规方法启动数据库

1、启动MySQL命令：

[root@db01 3306]# /etc/init.d/mysqld start

2、查看MySQL端口：

[root@db01 scripts]# ss -lntup|grep 330

3、查看MySQL进程：

[root@db01 3308]# ps -ef|grep mysql|grep -v grep

4、MySQL启动基本原理说明

/etc/init.d/mysqld是一个shell启动脚本，启动后最终会调用mysqld\_safe脚本，最后调用mysqld主程序启动mysql，如下，/etc/init.d/mysqld脚本中调用mysqld\_safe的程序。

/etc/init.d/mysql启动：

$bindir/mysqld\_safe –datadir=”$datadir” –pid-file=”$mysqld\_file\_path” $ other\_args >/dev/null 2> &1 &

/etc/init.d/mysql 停止：

if (kill -0 $mysqld\_pid 2>/dev/null)

then

echo $echo\_n”Shutting down MySQL”

kill $mysqld\_pid

2）初始化数据库时MySQL系统输出给出的启动方法

mysqld\_safe –user=mysql &

提示：

1、当找回root密码时，会经常使用mysqld\_safe –user=mysql & 带参数启动。

2、我们自己开发脚本启动数据库时可能会用到这个启动方法。

3、/etc/init.d/mysql和mysqld\_safe –user=mysql &的启动实质是一样的。

3）常规方法关闭数据库

1、关闭MySQL命令

[root@db01 3308]# /etc/init.d/mysqld stop

提示：使用编译及二进制安装的mysql，mysqld的启动脚本由mysql-xx/support-files/mysql.server复制而来

2、MySQL常规关闭数据库原理

/etc/init.d/mysql 停止：

if (kill -0 $mysqld\_pid 2>/dev/null)

then

echo $echo\_n”Shutting down MySQL”

kill $mysqld\_pid

4）强制关闭数据库方法：

killall mysqld

pkill mysqld

killall -9 mysqld

kill -9 pid

提示：

1、用killall关闭，要知道出现mysqld：no process likked 标识完成关闭操作。

2、法2的启动与关闭命令一般生产情况下不使用，特别是关闭命令。

5） 优雅关闭数据库方法：

第一种mysqladmin方法：

mysqladmin –u –p shutdown

第二种自带的脚本

/etc/init.d/mysqld stop

第三种kill信号的方法

kill –USE2 `cat path/pid`

提示：推荐前两个方法，自己写脚本启动停止就用第一个。

### 5.1.2 多实例MySQL启动与关闭方法示例

启动：

/data/3306/mysql start  
/data/3307/mysql start  
关闭：

/data/3306/mysql stop  
/data/3307/mysql stop  
启动：/bin/sh ${CmdPath}/mysqld\_safe --defaults-file=/data/${port}/my.cnf 2>&1 > /dev/null &  
关闭：${CmdPath}/mysqladmin -u ${mysql\_user} -p${mysql\_pwd} -S /data/${port}/mysql.sock shutdown

启动3306实例命令：  
mysqld\_safe --defaults-file=/data/3306/my.cnf 2>&1 > /dev/null &  
停止3306实例命令：  
mysqladmin -uroot -poldboy123 -S /data/3306/mysql.sock shutdown

/data/3306/mysql start  
启动3306实例命令：  
mysqld\_safe --defaults-file=/data/3306/my.cnf 2>&1 > /dev/null &  
停止3306实例命令：  
/data/3306/mysql stop  
mysqladmin -uroot -poldboy123 -S /data/3306/mysql.sock shutdown  
  
/etc/init.d/mysqld start相当于mysqld\_safe 2>&1 > /dev/null &  
/etc/init.d/mysqld stop相当于kill pidnum

<http://oldboy.blog.51cto.com/2561410/1566703>**初学者学习linux运维的几个问题及老鸟建议**

## 5.2 登录MySQL方法

### 5.2.1 单实例MySQL登录的方法

1、mysql #刚装完系统无密码情况登录方式。不要密码。

2、mysql –uroot #刚装完系统无密码情况登录方式。不要密码。

3、mysql –uroot –p #这里标准的DBA命令登录命令。

4、mysql –uroot –poldboy123 #非脚本里一般不这样用，密码明文会泄露，可以掩饰history功能解决。

强制Linux不记录敏感历史命令

#HISTCONTROL=ignorespace

登录后默认提示符是：mysql>，这个提示符也是可以改的，就像Linux命令行提示符也可以改一样。

更改MySQL数据登录提示符（了解的知识）方法如下：

###### 1.命令行修改登录提示符

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

PROMPT set to ‘\u@oldboy \r:\m:\s->’

###### 2.配置文件修改登录提示符

在my.cnf配置文件中[MySQL]模块下添加如下内容（注意：不是[mysqld]），保存后，无需重启MySQL，退出当前session，重新登录即可，如果在my.cnf配置文件中加，可以用\\，避免转义带来的问题

[mysql]

Prompt=\\u@oldboy \\r:\\M;\\s->

### 5.2.2 多实例MySQL登录的方法

多实例MySQL本地登录

[root@db01 3308]# mysql -uroot -S /data/3307/mysql.sock

提示：多实例通过mysql的-S命令指定不同的sock文件登录不同的服务中。

注意：多实例的远程连接无需指定sock路径：？？

[root@db01 3308]# mysql -uroot –p –h 127.0.0.1 –P3307

## 5.3 登录MySQL后

###### a.登录后默认提示符为

mysql>

b.为了防止操作失误，可以标记是测试环境，写在配置里永久生效。

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

## 5.4 善用MySQL的帮助命令help

Linux的帮助man，help

MySQL中的help命令和Linux命令行的man是类似的，和Linux下查看内置命令的help是同名的。要想查看MySQL中的命令使用语法，就需要用help，help后面接相关命令及命令组合即可。例如：help create，默认情况MySQL中大的命令执行不区分大小写，这点和Linux命令行严格区分大小写是有区别的。

直接在mysql中输入help

mysql> help

For information about MySQL products and services, visit:

http://www.mysql.com/

For developer information, including the MySQL Reference Manual, visit:

http://dev.mysql.com/

To buy MySQL Enterprise support, training, or other products, visit:

https://shop.mysql.com/

List of all MySQL commands:

Note that all text commands must be first on line and end with ';'

? (\?) Synonym for `help'.

clear (\c) Clear the current input statement.

connect (\r) Reconnect to the server. Optional arguments are db and host.

delimiter (\d) Set statement delimiter.

edit (\e) Edit command with $EDITOR.

ego (\G) Send command to mysql server, display result vertically.

exit (\q) Exit mysql. Same as quit.

go (\g) Send command to mysql server.

help (\h) Display this help.

nopager (\n) Disable pager, print to stdout.

notee (\t) Don't write into outfile.

pager (\P) Set PAGER [to\_pager]. Print the query results via PAGER.

print (\p) Print current command.

prompt (\R) Change your mysql prompt.

quit (\q) Quit mysql.

rehash (\#) Rebuild completion hash.

source (\.) Execute an SQL script file. Takes a file name as an argument.

status (\s) Get status information from the server.

system (\!) Execute a system shell command.

tee (\T) Set outfile [to\_outfile]. Append everything into given outfile.

use (\u) Use another database. Takes database name as argument.

charset (\C) Switch to another charset. Might be needed for processing binlog with multi-byte charsets.

warnings (\W) Show warnings after every statement.

nowarning (\w) Don't show warnings after every statement.

For server side help, type 'help contents'

初学者学习Linux运维的几个问题及建议：

潜意识的问题：

1、习惯到处问，不管啥问题，反正就是一点都不思考，结果是经常被鄙视，就算会了，自己也没多大提高。

2、找不到人，闻不到答案时就去搜索，搜索关键字还不会选取，结果还是找不到答案。

3、没意识或者不习惯去查看软件的帮助、手册。久而久之养成了坏习惯，限制了自己的发展。

建立：

1、一般解决故障问题，一定要先看故障日志，根据报错输出或者日志文件自行排查下。

2、实在解决不了，然后在搜索（g>b）好些，搜索关键字要精练，且尽量唯一。

3、对于知识点的学习、用法，查看软件的帮助或官方手册最好，直接的一手资料。

4、实在看不懂英文的，可以先搜索中文自恋，然后边对照边学习英文软件的帮助或官方手册。

<http://oldboy.blog.51cto.com/2561410/1566703>

## 5.5 退出MySQL数据库方法

mysql> quit 或 exit

使用快捷键Ctrl+c

## 5.6 设置及修改mysql root用户密码

### 5.6.1 MySQL数据库用户安全策略介绍

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

a. 为root设置比较复杂的密码。

b. 删除无用的mysql库内的用户账号，只留（root localhost）。

c. 删除默认存在的test数据库。

d. 增加用户的时候，尽量授权的权限最小，允许访问的主机范围最小。

f．除了上面的方法，针对MySQL数据库的用户处理，还有更严格的做法如下。

更安全的措施例如删除root，添加新的管理员用户。

**1）删除所有mysql中的用户，包括root超级用户。**

mysql> delete from mysql.user;

提示：root可以保留，然后修改为其他用户也可以。

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

mysql> grant all privileges on \*.\* to system@’localhost’ identified by ‘123’ with grant option;

提示：flush-privileges

g. 带密码的文件或脚本权限最好700，用户和组最好是root或mysql。

### 5.6.2 为管理员root用户设置密码方法

mysqladmin -u password ‘oldboy123’ #没有密码的用户设置密码命令

mysqladmin –uroot –poldboy123 password ‘oldboy’ –S #适合多实例方式修改密码。

强调：以上的命令都是在Linux命令行，不是mysql命令行。

### 5.6.3 修改管理员root密码法1：命令行外修改法

mysqladmin -u password ‘oldboy123’ #没有密码的用户设置密码命令

mysqladmin –uroot –poldboy123 password ‘oldboy’ –S #适合多实例方式修改密码。

强调：以上的命令都是在Linux命令行，不是mysql命令行。

### 5.6.4 修改管理员root密码法2：\*\*\*SQL语句修改法

mysql> update mysql.user set password=password(“oldboy”) where user=’root’ and host=’localhost’;

flush privileges;

提示：此法适合密码丢失后通过—skip-grant-tables参数启动数据库后修改密码。

### 5.6.5 修改管理员root密码法3

MySQL安装完毕后，root默认口令为空，需要马上修改root口令：

mysql> set password=password(“oldboy”)

flush privileges;

提示：此法不适合—skip-grant-tables方式修改密码。

## 5.7 找回丢失的mysql root用户密码

### 5.7.1 启动修改丢失的MySQL单实例root密码方法：

1、首先停止MySQL

/data/3306/mysql stop

Stoping MySQL...

2、使用—skip-grant-tables启动mysql，忽略授权登录验证。

mysqld\_safe --defaults-file=/data/3306/my.cnf --skip-grant-tables &

mysql> #登录时空密码。

提示：在启动--skip-grant-tables参数，标识忽略授权表验证。

3、无需密码即可登录MySQL

[root@db01 ~]# mysql -uroot -S /data/3306/mysql.sock

mysql>

4、修改root密码为新密码

mysql>update mysql.user set password=PASSWORD("oldboy3306") where user='root' and host='localhost';

mysql>flush privileges;

mysql>select user,host,password from mysql.user;

5、重启服务在登录

[root@db01 ~]# kill 67786

[root@db01 ~]# /data/3306/mysql start

###### 小结：

1、启动与关闭MySQL命令及工作原理。

2、登录MySQL方法。

3、善用MYSQL的帮助命令HELP

4、退出MYSQL数据库方法

5、设置及修改MYSQL ROOT用户密码

6、 MySQL数据库用户安全策略介绍

7、 为管理员root用户设置密码方法

8、 找回丢失的MYSQL ROOT用户密码

## 5.8 SQL结构化查询语言

### 5.8.1 什么是sql？

SQL，英文全称structured query language，中文意思是结构化查询语言，它是一种**对关系数据库中的数据进行定义和操作的语言方法**，是大多数关系数据库管理相同所支持的工业标准。\*\*\*存取数据库使用的语言\*\*\*

结构化查询语言SQL是一种数据库查询和程序设计语言，用于存取数据以及查询、更新和管理关系数据库系统；同时也是数据库脚本文件的扩展名。结构化查询语言是高级的非过程化编程语言，允许用户在高增数据结构上工作。它不要求用户指定对数据的存放方法，也不需要用户了解具体的数据存放方式，所以，具有完全不同于底层结构的，不同数据库系统可以使用相同的结构化查询语言作为数据输入与管理的接口。结构化查询语言语句可以嵌套，这使得它具有极大的灵活性和强大的功能。不同的数据库系统的SQL语言会有一些差别。

### 5.8.2 SQL的分类

**小结：**

Sql语句最常见的分类一般就是3类：

DDL（data definition language）数据定义语言（create，alter，drop），管理基础数据，例如：库，表。 #运维要熟练，开发也要熟练。

DCL（data control language）数据控制语言（grant，revoke，commit），rollback

，用户授权，权限回收，数据提交回滚等。 #运维要熟练

DML（date manipulation language）数据操作语言（select，insert，delete，update），针对数据库里的表里的数据进行操作，记录。 #开发要熟练，运维要理解。

## 5.9 MySQL数据库常见管理应用

### 5.9.1 创建数据库

命令语法：create database <数据库名> #注意库名不能数字开头，大小写不敏感。

在MySQL默认字符集情况下简历数据库测试如下：

a. 建立一个名为oldboy的数据库

mysql> create database oldboy

查看库的语句:

mysql> show create database oldboy;

+----------+-----------------------------------------------------------------+

| Database | Create Database |

+----------+-----------------------------------------------------------------+

| oldboy | CREATE DATABASE `oldboy` /\*!40100 DEFAULT CHARACTER SET utf8 \*/ |

+----------+-----------------------------------------------------------------+

1 row in set (0.00 sec)

b. 建立一个名为oldboy\_gbk的GBK字符集数据库

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

mysql> show create database oldboy\_gbk;

+------------+--------------------------------------------------------------------+

| Database | Create Database |

+------------+--------------------------------------------------------------------+

| oldboy\_gbk | CREATE DATABASE `oldboy\_gbk` /\*!40100 DEFAULT CHARACTER SET gbk \*/ |

+------------+--------------------------------------------------------------------+

1 row in set (0.00 sec)

如果不知道如何创建请使用：

mysql> help create database; #查看如何创建库

mysql> show character set; #查看字符集规则

c. 建立一个名为lidongkai\_uft8的UTF8数据库

mysql> create database lidongkai\_utf8 character set utf8 collate utf8\_general\_ci;

Query OK, 1 row affected (0.00 sec)

mysql> show create database lidongkai\_utf8;

+----------------+-------------------------------------------------------------------------+

| Database | Create Database |

+----------------+-------------------------------------------------------------------------+

| lidongkai\_utf8 | CREATE DATABASE `lidongkai\_utf8` /\*!40100 DEFAULT CHARACTER SET utf8 \*/ |

+----------------+-------------------------------------------------------------------------+

1 row in set (0.00 sec)

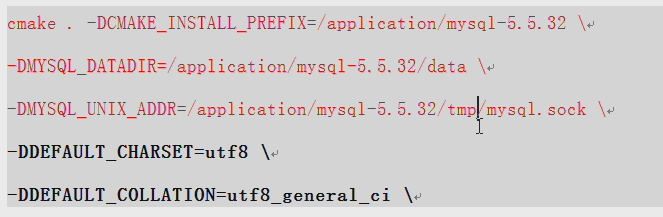
d. 创建不同字符集格式的数据库命令

create database oldboy; <==默认数据库配置，相当于创建拉UTF8字符集数据库。  
create database oldboy\_gbk DEFAULT CHARACTER SET gbk COLLATE gbk\_chinese\_ci;    <==创建gbk字符集数据库  
create database oldboy\_utf8 DEFAULT CHARACTER SET utf8  COLLATE utf8\_general\_ci;  <==创建utf8字符集数据库

mysql> show character set;

mysql> show databases like 'oldboy%'; 模糊匹配

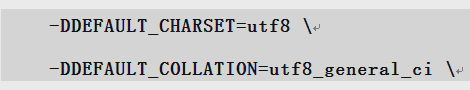
如果编译的时候指定了特定的字符集，则以后创建对应字符集的数据库就不需要指定字符集了。如下：



**企业场景创建什么字符集的数据库？**

1、根据开发的程序确定字符集（建议utf8）

2、可以编译时指定默认字符集，例如：



然后建库的时候默认创建即可，create database oldboy;

3、编译的时候没有指定字符集或者制定了和程序不同的字符集，如何解决？

指定字符集创建数据库即可。

创建utf8字符集数据库

create database lidongkai\_utf8 character set utf8 collate utf8\_general\_ci;

创建gbk字符集数据库

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

### 5.9.2 显示数据库

命令：show databases;

mysql> show databases;

+--------------------+

| Database |

+--------------------+

| information\_schema |

| mysql |

| performance\_schema |

| test |

+--------------------+

4 rows in set (2.08 sec)

模糊匹配：

mysql> show databases like 'oldboy%';

+--------------------+

| Database (oldboy%) |

+--------------------+

| oldboy |

| oldboy\_gbk |

+--------------------+

2 rows in set (0.00 sec)

## 5.9.3 删除数据库

命令：drop database <数据库名>

例如：删除名为oldboy的数据库

mysql> drop database oldboy;

Query OK, 0 rows affected (0.00 sec)

如果有特殊符号，删除不了的话，使用：

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

学习潜意识旧查看帮助，看提示：

mysql> help drop database

### 5.9.4 连接数据库

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

**例如：如果oldboy数据库存在，尝试进入oldboy库里：**

mysql> use oldboy #切换数据库，

Database changed

### 5.9.5 查看当前连接的数据库

**mysql> select database(); #查看当前数据库，相当于Linux下的pwd**

mysql> select database();

+------------+

| database() |

+------------+

| oldboy |

+------------+

1 row in set (0.00 sec)

**mysql> select user();#查看当前用户，相当于whoami**

mysql> select user();

+----------------+

| user() |

+----------------+

| root@localhost |

+----------------+

1 row in set (0.00 sec)

### 5.9.6 当前数据库包含的表信息

**切到数据库里面去查看**

mysql> show tables;

Empty set (0.00 sec) #空表，因为是新库，还没有建立表。

mysql> show tables from oldboy\_gbk; #查看指定库中的表。

Empty set (0.00 sec)

**查看数据库版本：**

mysql> select version();

+------------+

| version() |

+------------+

| 5.5.32-log |

+------------+

1 row in set (0.00 sec)

**查看当前时间**

mysql> select now();

+---------------------+

| now() |

+---------------------+

| 2015-12-15 19:17:37 |

+---------------------+

1 row in set (0.00 sec)

**查看所有用户信息**

mysql> select user,host from mysql.user;

+------+-----------+

| user | host |

+------+-----------+

| root | 127.0.0.1 |

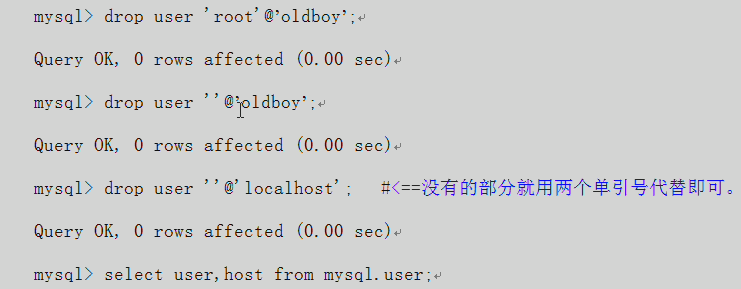
| root | localhost |

+------+-----------+

2 rows in set (0.00 sec)

## 5.10 删除MySQL系统多余账号

语法：drop user “user”@”主机域” #注意引号，可以是单或双引号，但是不能不加。



注意：删除后的结果如下面返回的结果，如果drop删除不了（一般是特殊字符或大写），可以用下面方式删除（以root用户，oldboy主机为例）：

delete from mysql.user where user=’root’ and host=’oldboy’;

## 5.11 创建MySQL用户及赋予用户权限

### 5.11.1 通过help查看grant命令帮助

1、通过在MySQL中输入”help grant”得到如下帮助信息。



通过查看grant的命令帮助，可以很容易的找到创建用户并授权的例子！

2、运维人员比较常用的创建用户的方法是，使用grant命令在创建用户的同时进行权限授权。具体授权例子为:

grant all on db1.\* to ‘jeffrey’@’localhost’ identified by ‘mypass’;

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

create user ‘jeffrey’@’localhost’ identified by ‘mypass’;

**相当于useradd Jeffrey && echo mypass|password –stdin jeffrey**

grant all on db1.\* to ‘jeffrey’@’localhost’;

**相当于sudo里授权all**

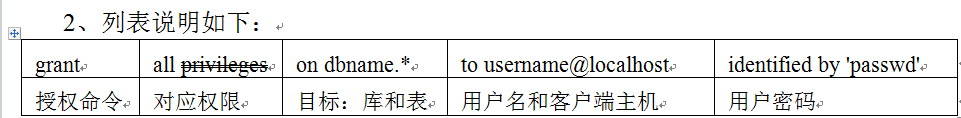
#以上两条命令相当于2中的命令

5.11.2 通过grant命令创建用户并授权

1、grant命令简单语法如下：

grant all privilege on dbname.\* to username@localhost identified by ‘passwd’

**2、列表说明如下：**



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

3、操作案例1：创建oldboy用户，对test库具备所有权限，允许从localhost主机登录管理数据库，密码是oldboy123.

a. 实现上述操作的具体命令如下：

mysql> grant all on test.\* to 'oldboy'@'localhost' identified by 'oldboy123';

Query OK, 0 rows affected (0.00 sec)

**b.实战演示：**

**查看当前数据库用户情况，然后执行对应命令授权如下：**

mysql> select user,host from mysql.user;

+--------+-----------+

| user | host |

+--------+-----------+

| root | 127.0.0.1 |

| oldboy | localhost |

| root | localhost |

+--------+-----------+

3 rows in set (0.00 sec)

**查看授权用户oldboy的具体权限命令：**

mysql> show grants for 'oldboy'@'localhost';

### 5.11.4 授权局域网内主机远程连接数据库：

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

法1：

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

**法2：**

mysql> grant all on \*.\* to 'test1'@'10.0.0.0/255.255.255.0' identified by 'test123';

提示：如果是web连接数据库的用户，尽量不要授权all，而是select、insert、update、delete。

### 5.11.5 MySQL用户可以授权的权限有哪些

通过实验获得all privileges包括哪些权限

1、取消oldboy的插入权限（insert）看看。

mysql> **revoke insert** on test.\* from 'oldboy'@'localhost';

Query OK, 0 rows affected (0.00 sec)

mysql> flush privileges;

Query OK, 0 rows affected (0.00 sec)

mysql> show grants for oldboy@'localhost';

+--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+

| Grants for oldboy@localhost |

+--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+

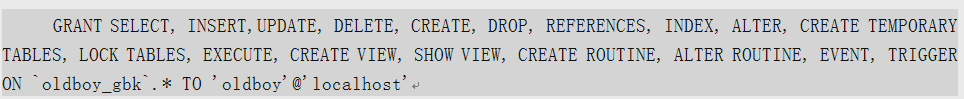
| GRANT USAGE ON \*.\* TO 'oldboy'@'localhost' IDENTIFIED BY PASSWORD '\*FE28814B4A8B3309DAC6ED7D3237ADED6DA1E515' |

| GRANT SELECT, UPDATE, DELETE, CREATE, DROP, REFERENCES, INDEX, ALTER, CREATE TEMPORARY TABLES, LOCK TABLES, EXECUTE, CREATE VIEW, SHOW VIEW, CREATE ROUTINE, ALTER ROUTINE, EVENT, TRIGGER ON `test`.\* TO 'oldboy'@'localhost' |

+--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+

2 rows in set (0.00 sec)

**因此可以说是，all privileges的权限包括：**

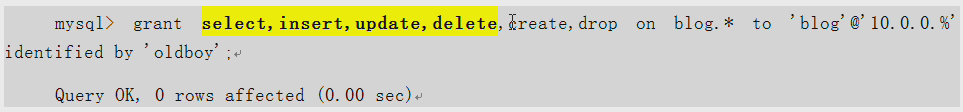


即在授权时，可以授权用户最小的满足业务需求的权限，而不是一味的授权“ALL PRIVILEGES“.

### 5.11.6 企业生产环境如何授权用户权限？

1、博客，CMS等产品的数据库授权：

对于web连接用户授权尽量采用最小化原则，很多开源软件都是web界面安装，因此，在安装期间除了select、insert、update、delete4个权限外，还需要create、drop等比较危险的权限。



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

2、生产数据库表后，要收回create、drop授权：

3、生产环境针对主库（写为主杜伟辅）用户的授权：

普通的环境：

本机：lnmp，lamp环境数据库授权

grant all on `blog`.\* to ‘blog’@’localhost’ identified by ‘oldboy123’;

**应用服务器和数据库服务器不在一个主机上的授权：**

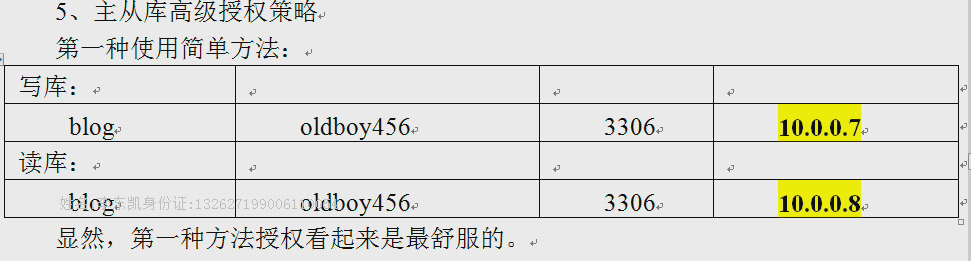
grant all on `blog`.\* to ‘blog’@’10.0.0.%’ identified by ‘oldboy123’;

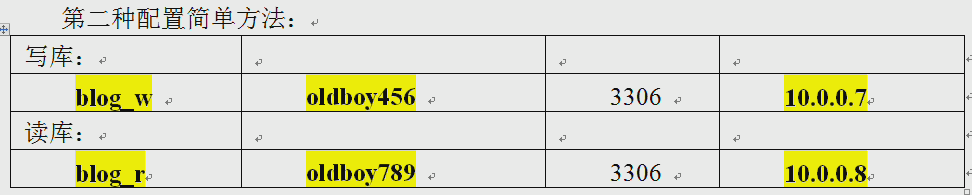
**严格的授权：重视安全，忽略了方便：**

grant select，insert，update，delete on `blog`.\* to ‘blog’@’localhost’ identified by ‘oldboy123’;

4、生产环境从库（只读）用户的授权：

grant select on `blog`.\* to ‘blog’@’localhost’ identified by ‘oldboy123’;





6、生产场景授权具体命令为：

**a.主库授权的命令：**

grant select，insert，update，delete on `blog`.\* to ‘blog’@’10.0.0.%’ identified by ‘oldboy123’;

**b．从库授权用户的命令：**

grant select on `blog`.\* to ‘blog’@’localhost’ identified by ‘oldboy123’;

当然从库除了做select的授权外，还可以加read-only等只读参数，严格控制web用户写从库。

7、这里有一个较为重要的问题

就是主从库的MySQL库和表是同步的，无法针对同一个用户授权不同的权限。因为主库授权后会自动同步到从库上，导致从库的授权只读失败。

**解决方法：**

a. 取消MySQL库的同步。

b. 授权主库权限后，从库执行收回增删改权限。

c. 不在授权商控制增删改，而是用read-only参数，控制普通用户更新从库，注意，read-only参数对超级用户无效。

**设置及修改mysql root用户密码**

Mysql数据库用户安全策略介绍

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

1、为root设置比较复杂的密码。

2、删除无用的mysql库内的用户账号

3、删除默认存在的test数据库

4、增加用户的时候，尽量授权的权限最小，允许访问的合租记范围最小。

除了上面的方法，针对mysql数据库的用户处理，还有更严格的做法如下。

更安全的弱势例如删除root，添加新的管理员用户

1）删除所有mysql中的用户，包括root超级用户

mysql> delete from mysql.user;

#root可以保留，然后修改为其他用户也可以。

2）增加system并提升为超级管理员，即和root等价的管理员用户，知识名字不同。

Grant all privileges on \*.\* to system@’localhost’ identified by ‘123456’ with grant option;

## 5.13 表操作

### 5.13.1 以默认字符集Latin1建库

mysql> create database oldboy;

mysql> show databases like "oldboy%";

mysql> show create database oldboy\G #查看建立库结构

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

Database: oldboy

Create Database: CREATE DATABASE `oldboy` /\*!40100 DEFAULT CHARACTER SET utf8 \*/

1 row in set (0.00 sec)

## 5.13.2 建立表

1）建表的基本命令语法：

create table <表名>（

<字段名1><类型1>,

….

<字段名n><类型n>;

提示：其中create table是关键字，不能更改，但是大小写可以变化。

2） 建表语句：

下面是人工写法设计的建表语句例子，表名student，

mysql> use oldboy;

mysql> create table student(

-> id（列，字段） int（整形）(4)（长度） not null,

-> name char（定长的字符串）(20) not null,

-> age tinyint（微小的整形）(2) NOT NULL default '0',

-> dept varchar（变长的字符串）(16) default NULL

-> );

Query OK, 0 rows affected (0.06 sec)

mysql> show tables;

+------------------+

| Tables\_in\_oldboy |

+------------------+

| student |

+------------------+

1 row in set (0.00 sec)

数据库服务器---》数据库（多个实例）----》  
多个库----》多个表----》多个字段行列（数据）

第二种MySQL生产的建表语句student表例子：

mysql> show create table student\G #查看建表语句

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

Table: student

Create Table: CREATE TABLE `student` (

`id` int(4) NOT NULL,

`name` char(20) NOT NULL,

`age` tinyint(2) NOT NULL DEFAULT '0',

`dept` varchar(16) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8

1 row in set (0.00 sec)

需要注意的是：MySQL5.1和MySQL5.5环境的默认建表语句中的引擎的不同，如果希望控制表的引擎，就要在建表语句里显示的指定引擎建表：

MySQL5.1及以前默认引擎为myisam，MySQL5.5.5以后默认引擎为innoDB

查看表结构的命令：

mysql> desc student;

+-------+-------------+------+-----+---------+-------+

| Field | Type | Null | Key | Default | Extra |

+-------+-------------+------+-----+---------+-------+

| id | int(4) | NO | | NULL | |

| name | char(20) | NO | | NULL | |

| age | tinyint(2) | NO | | 0 | |

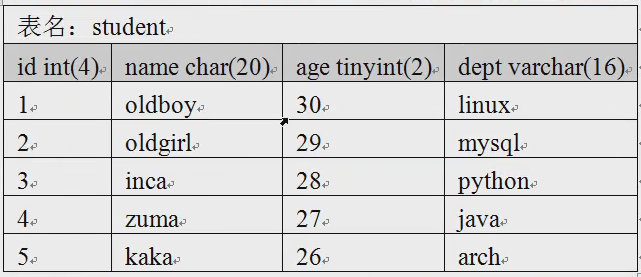
| dept | varchar(16) | YES | | NULL | |

+-------+-------------+------+-----+---------+-------+

4 rows in set (0.00 sec)



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



直观看，MySQL表的内容和word，excell表格内容显示几乎没什么区别。

## 5.13.3 MySQL表的字段类型

1）数字类型

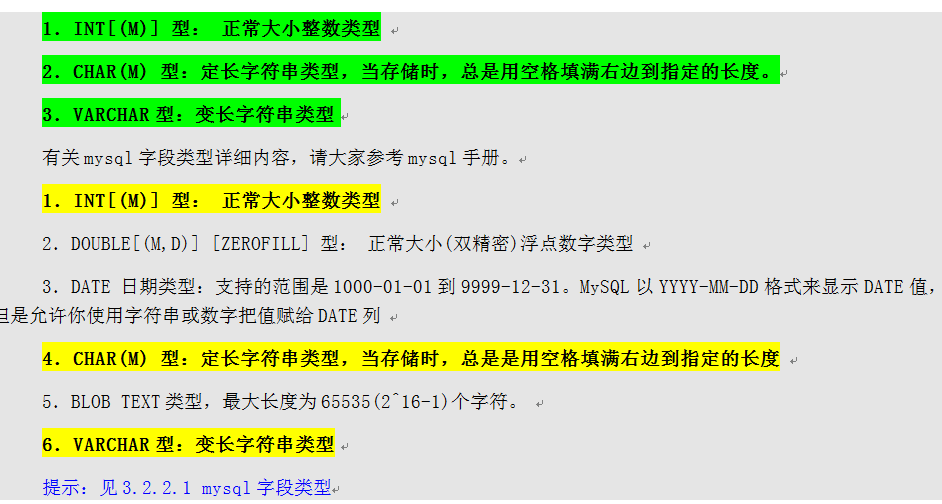


###### 2） 日期和时间类型（date日期类型：支持的范围是1000-01-01 到9999-12-31.MySQL以YYYY-MM-DD格式来显示date值，但是允许使用字符串或数字把值赋给date列）



###### 3） 字符串类型







小结：

1、char是定长，不够的用空格补全，浪费存储空间，查询速度快，多数系统表字段都是定长。

2、varchar是变长，查询速度慢。

### 5.13.4 生产环境标准的UTF8格式表结构语句

某sns产品生产正式建表语句：

use sns;  
set names gbk;  
CREATE TABLE `subject\_comment\_manager` (  
  `subject\_comment\_manager\_id` bigint(12) NOT NULL auto\_increment COMMENT '主键',  
  `subject\_type` tinyint(2) NOT NULL COMMENT '素材类型',  
  `subject\_primary\_key` varchar(255) NOT NULL COMMENT '素材的主键',  
  `subject\_title` varchar(255) NOT NULL COMMENT '素材的名称',  
  `edit\_user\_nick` varchar(64) default NULL COMMENT '修改人',  
  `edit\_user\_time` timestamp NULL default NULL COMMENT '修改时间',  
  `edit\_comment` varchar(255) default NULL COMMENT '修改的理由',  
  `state` tinyint(1) NOT NULL default '1' COMMENT '0代表关闭，1代表正常',  
  PRIMARY KEY  (`subject\_comment\_manager\_id`),  
  KEY `IDX\_PRIMARYKEY` (`subject\_primary\_key`(32)), #<==括号内的32表示对前32个字符做前缀索引。  
  KEY `IDX\_SUBJECT\_TITLE` (`subject\_title`(32))  
  KEY `index\_nick\_type` (`edit\_user\_nick`(32),`subject\_type`)#<==联合索引，此行为新加的，用于给大家讲解的。实际表语句内没有此行。  
) ENGINE=InnoDB AUTO\_INCREMENT=1 DEFAULT CHARSET=utf8;

### 5.13.5 查看简历的表结构

A、查看表结构命令：

desc 表名或show columns from表名

例：

mysql> desc student

mysql> show columns from student;

### 5.13.6 查看已建表的语句

B、查已建表的语句（可以看索引及创建表的相关信息）：

mysql> show create table student\G

C、演示整个过程

1、创建数据库：

create database oldboy;

2、查看已建库的语句：

show create database oldboy\G

3、切换到数据库

Use oldboy;

4、建立表

create table student(  
id int(4) not null,  
name char(20) not null,  
age tinyint(2)  NOT NULL default '0',  
dept varchar(16)  default NULL  
);

5、查看已建表的语句：

show create table user\G

6、查看表结构：

desc oldboy;

### 5.13.7 为表的字段创建索引\*\*\*\*\*

数据库的索引就像书的目录一样，如果在字段上建立了索引，那么以索引列为查询条件时就可以加快查询数据的速度，这是MySQL优化的重要内容之一。

#### 5.13.7.1 创建主键索引

查询数据库，按主键查询是最快的，每个表只能有一个主键列，但是可以有多个普通索引列。主键列要求列的所有内容必须唯一，而普通索引列不要求内容必须唯一。

主键就类似在学校学习时的学号一样，班级内是唯一的，整个表的每一条记录的主键值在表内都是唯一的，用来唯一标识一条记录。那么，我们该如何建立主键索引和普通索引呢？

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

建立主键索引的方法：

1、在建表时，可以增加建立主键索引的语句如下：

drop table student;  
create table student(  
id int(4) not null AUTO\_INCREMENT,  
name char(20) not null,  
age tinyint(2)  NOT NULL default '0',  
dept varchar(16)  default NULL,  
primary key(id),  
KEY index\_name(name)  
);

提示：

1、primary key（id） #主键

2、key index\_name(name) #name字段普通索引。

优化：在唯一值多的列上建索引查询效率高。

提示：只有int类型且为primary key才可以使用auto\_increment.

2、建表后通过alter命令增加主键索引（工作中不要这样干）

a. 主键列不能重复创建，必须先删除上面的配置

mysql> alter table student drop primary key;

#如果是自增主键，则无法删除。

#### 利用alter命令修改id列为自增主键列

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

#### 5.13.7.2 创建普通索引\*\*\*\*\*\*

1、在建表时，可以增加建立普通索引的语句如下：

create table student(  
id int(4) not null AUTO\_INCREMENT,  
name char(20) not null,  
age tinyint(2)  NOT NULL default '0',  
dept varchar(16)  default NULL,  
primary key(id),  
KEY index\_name(name)  
);

提示：

1、KEY index\_name(name) #name字段普通索引

优化：在唯一值多的列上建立索引查询效率高。

2、建表后利用alter增加普通索引

删除建表时创建的index\_name索引

mysql> alter table student drop index index\_name;

Query OK, 0 rows affected (0.00 sec)

Records: 0 Duplicates: 0 Warnings: 0

mysql> desc student;

+-------+-------------+------+-----+---------+----------------+

| Field | Type | Null | Key | Default | Extra |

+-------+-------------+------+-----+---------+----------------+

| id | int(4) | NO | PRI | NULL | auto\_increment |

| name | char(20) | NO | | NULL | |

| age | tinyint(2) | NO | | 0 | |

| dept | varchar(16) | YES | | NULL | |

+-------+-------------+------+-----+---------+----------------+

4 rows in set (0.00 sec)

在name列上添加索引，索引名为index\_name

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

mysql> desc student;

+-------+-------------+------+-----+---------+----------------+

| Field | Type | Null | Key | Default | Extra |

+-------+-------------+------+-----+---------+----------------+

| id | int(4) | NO | PRI | NULL | auto\_increment |

| name | char(20) | NO | MUL | NULL | |

| age | tinyint(2) | NO | | 0 | |

| dept | varchar(16) | YES | | NULL | |

+-------+-------------+------+-----+---------+----------------+

4 rows in set (0.01 sec)

提示：PRI为主键的标识，MUL为普通索引的标识。

生产场景案例：

#生产场景，数据量很大的时候，不适合建立索引，会影响用户访问。曾经有400-500万条记录的表，建立索引，花了90-180秒。（与设备硬件有直接关系，尽量选择在业务低谷时建立索引）。

查看命令

查看索引： show index from student\G

查看表结构：desc student;

删除索引：alter table student drop index index\_name;

添加索引：alter table student add index index\_name(name);

添加索引2：create index index\_dept on student(dept(8)); ##只能添加，无法删除。

法2的语句等同于alter table student add index index\_dept(dept(8));

#### 5.13.7.4 对字段的前N个字符建立索引

当遇到表中比较大的列时，列内容的前N个字符在所偶内容中已经接近唯一的时，这时可以对列的前n个字符建立索引，而无需对整个列建立索引，这样可以节省创建索引占用的系统空间，以及降低读取和更新维护索引消耗的系统资源。

对字段难道前N个字符创建普通索引的语法：

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

操作实践：

在dept系别列上，前8个字符创建索引，此列总共长度为16个。

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

这个语句等同于 alter table student index index\_dept(dept(8));

Query OK, 0 rows affected (0.07 sec)

Records: 0 Duplicates: 0 Warnings: 0

mysql> desc student;

+-------+-------------+------+-----+---------+----------------+

| Field | Type | Null | Key | Default | Extra |

+-------+-------------+------+-----+---------+----------------+

| id | int(4) | NO | PRI | NULL | auto\_increment |

| name | char(20) | NO | MUL | NULL | |

| age | tinyint(2) | NO | | 0 | |

| dept | varchar(16) | YES | MUL | NULL | |

+-------+-------------+------+-----+---------+----------------+

4 rows in set (0.00 sec)

#### 5.13.7.3 为表的多个字段创建联合索引

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

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

Query OK, 0 rows affected (0.03 sec)

Records: 0 Duplicates: 0 Warnings: 0

mysql> desc student;

+-------+-------------+------+-----+---------+----------------+

| Field | Type | Null | Key | Default | Extra |

+-------+-------------+------+-----+---------+----------------+

| id | int(4) | NO | PRI | NULL | auto\_increment |

| name | char(20) | NO | MUL | NULL | |

| age | tinyint(2) | NO | | 0 | |

| dept | varchar(16) | YES | | NULL | |

+-------+-------------+------+-----+---------+----------------+

4 rows in set (0.00 sec)

可以为多列的前n个字符列创建联合索引，实践演示如下：

mysql> create index ind\_name\_dept on student(name(8),dept(6));

mysql> show index from student\G

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

Table: student

Non\_unique: 0

Key\_name: PRIMARY

Seq\_in\_index: 1

Column\_name: id

Collation: A

Cardinality: 0

Sub\_part: NULL

Packed: NULL

Null:

Index\_type: BTREE

Comment:

Index\_comment:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 2. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Table: student

Non\_unique: 1

Key\_name: ind\_name\_dept

Seq\_in\_index: 1

Column\_name: name

Collation: A

Cardinality: 0

Sub\_part: 8

Packed: NULL

Null:

Index\_type: BTREE

Comment:

Index\_comment:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 3. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Table: student

Non\_unique: 1

Key\_name: ind\_name\_dept

Seq\_in\_index: 2

Column\_name: dept

Collation: A

Cardinality: 0

Sub\_part: 6

Packed: NULL

Null: YES

Index\_type: BTREE

Comment:

Index\_comment:

3 rows in set (0.00 sec)

提示：按条件列查询数据时，联合索引是有前缀生效特性的。

index(a,b,c)仅a,ab,abc三个查询条件列可以走索引。b,bc,ac,c等无法使用索引了。尽量把最常用作为查询条件的列，放在第一个位置。

###### 统计表记录的唯一值的命令：

Select user,host from mysql.user;

1、select distinct count(user) from mysql.user; #不建议使用

2、select count(distinct user) from mysql.user; #高化翔版本

提示：尽量在唯一值多的大表上建立索引。

提示：按条件列查询数据时，联合索引是有前缀生效特性的。

也可以多列创建联合主键

Show create table mysq.user\G

PRIMARY KEY (`Host`,`User`)

#### 5.13.7.4 创建唯一索引（非主键）

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

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

#### 5.13.7.5 索引列的创建及生效条件

问题1、既然索引可以加快査询速度, 那么就给所有的列建索引吧?

解答: 因为索引不但占用系统空间, 而且更新数据库时还需要维护索引数据的,因此,索引是一把双刃剑，并不是越多越好，例如：数十到几百行的小表上无需建立索引,更新频繁,读取比较少的表要少建立索引。

问题2、需要在哪些列上创建索引才能加快速度呢?

select user,host from mysq1.user where password=..,索引一定要创建在where后的条件列上，而不是select后的选择数据的列上。另外，我们要尽量选择在唯一值多的大表上的列建立索引，例如,男女性别列唯一值少，所以不适合建立索引。

查看唯一的数量：select count(distinct user) from mysql.user; #高化翔版本

#### 5.13.7.6 创建索引的小结：

**1、创建索引相关命令集合**

如果没有主键，修改字段id为主键，自增。

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

删除主键索引：主键列不能自增。

alter table student drop primary key;

删除普通索引：

alter table student drop index index\_name;

创建普通索引：

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

查看索引信息：

show index from student\G

建立索引：

create index index.name on test(name);

等同于：

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

创建联合索引：

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

查看表内唯一值个数：

select count(distinct user) from mysql.user;

创建联合索引的好处，符合条件的数据更少，速度更快。

创建唯一索引：

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

**创建索引的基本知识小结：**

1． 索引类似书籍的目录，会较快查询数据的速度。

2． 要在表的列（字段）上创建索引。

3． 索引会加快查询速度，但是也会影响更新的速度，因为更新要维护索引数据。

4． 索引列并不是越多越好，要在频繁查询的where后的条件列上创建索引。

5． 小表或重复值很多的列上可以不建索引，要在大表以及重复值得条件列上创建索引。

6． 多个列联合索引有前缀生效特性。

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

尽量把最常用作为查询条件的列，放在第一位置。

7． 当字段内容前N个字符已经接近唯一时，可以对字段的前N个字符创建索引。

8． 索引从工作方式区分，有主键，唯一，普通索引。

9． 索引类型会有btree（默认）和hash（适合做缓存（内存数据库）等）

### 5.13.8 往表中插入数据

###### 1）命令语法：

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

###### 2）新建一个简单的测试表test

mysql> create table `test` (

-> `id` int(4) not null auto\_increment,

-> `name` char(20) not null,

-> primary key (`id`)

-> );

Query OK, 0 rows affected (0.07 sec)

###### 3）往表中插入数据的不同的语法例子：

1、按规矩指定所有列名，并且每列都插入值

mysql> insert into test(id,name) values(1,'oldboy');  
Query OK, 1 row affected (0.08 sec)

2、由于id列为自增的，所以，可以只在name列插入值

mysql> insert into test(name) values('oldboy');  
insert into test(id,name) values (null,’zhangsan’);

3、如果不指定列，就要按规矩为每列都插入恰当的值

mysql> insert into test values(3,'inca');

4、批量插入数据方法，提升效率

mysql> insert into test values(4,'zuma'),(5,'kaka');

5、测试完毕，退出数据库，然后备份上述数据，留着备用：

[root@db01 ~]# mysqldump -uroot -poldboy123 -S /data/3306/mysql.sock -B oldboy >/opt/bak\_$(date +%F).sql

[root@db01 ~]# ll /opt/

总用量 8

-rw-r--r-- 1 root root 2758 12月 28 21:45 bak\_2015-12-28.sql

6、备份后检查备份的sql数据内容：过滤无用信息

[root@db01 ~]# grep -E -v "#|\/|^$|--" /opt/bak\_2015-12-28.sql

USE `oldboy`;

DROP TABLE IF EXISTS `student`;

CREATE TABLE `student` (

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

`name` char(20) NOT NULL,

`age` tinyint(2) NOT NULL DEFAULT '0',

`dept` varchar(16) DEFAULT NULL,

PRIMARY KEY (`id`),

KEY `ind\_name\_dept` (`name`(8),`dept`(6))

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

LOCK TABLES `student` WRITE;

UNLOCK TABLES;

DROP TABLE IF EXISTS `test`;

CREATE TABLE `test` (

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

`name` char(20) NOT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=6 DEFAULT CHARSET=utf8;

LOCK TABLES `test` WRITE;

INSERT INTO `test` VALUES (1,'oldboy'),(2,'oldgirl'),(3,'laoqiang'),(4,'zuma'),(5,'kaka');

UNLOCK TABLES;

7、一个备份错误案例：如果部件厂备份数据可能导致数据不是想要的。

[root@db01 ~]# mysqldump -uroot -poldboy123 -S /data/3306/mysql.sock -A -B oldboy >/opt/bak\_$(date +%F).sql

[root@db01 ~]# cat /opt/bak\_2015-12-28.sql

Usage: mysqldump [OPTIONS] database [tables]

OR mysqldump [OPTIONS] --databases [OPTIONS] DB1 [DB2 DB3...]

OR mysqldump [OPTIONS] --all-databases [OPTIONS]

For more options, use mysqldump –help

提示：

1、错误的点时-A表示所有库，后面不能再指定oldboy库了。

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

删除test的所有表：

truncate table test;

## 5.14 查询数据

### 5.14.1 查询表的所有数据行

1、命令语法：select<字段1，字段2，….> from <表名>where<表达式> 其中，select，from，where是不能随便改的，是关键字，支持大小写。

2、例：查看表test中所有数据

a.指定查询部分数据：

mysql> use oldboy;

mysql> select \* from test;

+----+----------+

| id | name |

+----+----------+

| 1 | oldboy |

| 2 | oldgirl |

| 3 | laoqiang |

| 4 | zuma |

| 5 | kaka |

+----+----------+

5 rows in set (0.00 sec)

b． 直接查询库下面表的数据

mysql> select \* from oldboy.test; #用点号隔开。

+----+----------+

| id | name |

+----+----------+

| 1 | oldboy |

| 2 | oldgirl |

| 3 | laoqiang |

| 4 | zuma |

| 5 | kaka |

+----+----------+

5 rows in set (0.00 sec)

c． 查看mysql库的用户

mysql> select user,host from mysql.user;

+--------+------------------------+

| user | host |

+--------+------------------------+

| test | 10.0.0.% |

| test1 | 10.0.0.0/255.255.255.0 |

| root | 127.0.0.1 |

| oldboy | localhost |

| root | localhost |

+--------+------------------------+

5 rows in set (0.06 sec)

5.14.2 根据指定条件查询表的部分数据

1） 例：查看表test中前2行数据：select \* from test limit 2;

mysql> select \* from test limit 2;

+----+---------+

| id | name |

+----+---------+

| 1 | oldboy |

| 2 | oldgirl |

+----+---------+

2 rows in set (0.00 sec)

从第1行开始查，查2行记录。select \* from test limit 1,2;

mysql> select \* from test limit 1,2;

+----+----------+

| id | name |

+----+----------+

| 2 | oldgirl |

| 3 | laoqiang |

+----+----------+

2 rows in set (0.00 sec)

2） 指定固定条件查询数据 select \* from test where id=1;

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

+----+--------+

| id | name |

+----+--------+

| 1 | oldboy |

+----+--------+

1 row in set (0.00 sec)

多条件查询数据：select \* from test where name='oldgirl' and id=2;

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

+----+---------+

| id | name |

+----+---------+

| 2 | oldgirl |

+----+---------+

1 row in set (0.00 sec)

#提示：查询字符类型的条件的值要带单引号，数字值不带引号。

3）指定固定条件范围查询数据：select \* from test where id>2 and id<5;

mysql> select id,name from test where id>2 and id<5;

+----+----------+

| id | name |

+----+----------+

| 3 | laoqiang |

| 4 | zuma |

+----+----------+

2 rows in set (0.05 sec)

4） 其他查询功能

1、排序功能查询：

正序：

select \* from test order by id asc;

mysql> select \* from test order by id asc;

+----+----------+

| id | name |

+----+----------+

| 1 | oldboy |

| 2 | oldgirl |

| 3 | laoqiang |

| 4 | zuma |

| 5 | kaka |

+----+----------+

5 rows in set (0.00 sec)

倒序：

select \* from test order by id desc;

mysql> select \* from test order by id desc;

+----+----------+

| id | name |

+----+----------+

| 5 | kaka |

| 4 | zuma |

| 3 | laoqiang |

| 2 | oldgirl |

| 1 | oldboy |

+----+----------+

5 rows in set (0.00 sec)

导出数据：

select id,name from test where id >1 and id <5 order by id desc INTO OUTFILE '/tmp/a.txt';

mysql> select \* from test where id>1 and id<5 order by id desc into outfile'/tmp/a.txt';

[root@db02 ~]# cat /tmp/a.txt

4 zuma

3 inca

2 oldgirl

### 5.14.3 多表查询

#### 5.14.3.1 建立几个关联表

要实现多表连表查询，就需要有关连表及数据，表名及字段内容如下：



实战演示：

create table student(

Sno int(10) not null comment '学号',

Sname varchar(16) not null comment '姓名',

Ssex char(2) not null comment '性别',

Sage tinyint(2) not null default '0' comment '学生年龄',

Sdept varchar(16) default null comment '学生所在系别',

primary key (Sno) ,

key index\_Sname (Sname)

) engine=innodb auto\_increment=1 default charset=utf8;

create table course(

Cno int(10) NOT NULL COMMENT '课程号',

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

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

PRIMARY KEY (Cno)

) ENGINE=InnoDB AUTO\_INCREMENT=1 DEFAULT CHARSET=utf8;

create table `SC` (

Scid int(12) not null auto\_increment comment '主键',

`Cno` int(10) not null comment '课程号',

`Sno` int(10) not null comment '学号',

`Grade` tinyint(2) not null comment '学生成绩',

primary key (`SCid`)

) engine=innodb default charset=utf8;

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

INSERT INTO student values(0002,'王硕 ','男',30,'computer application');

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

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

INSERT INTO student values(0005,'oldgirl','女',26,'计算机科学与技术');

INSERT INTO student values(0006,'莹莹','女',22,'护士');

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

INSERT INTO course values(1002,'Linux高级架构师',5);

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

######################开始查询################################

mysql> select student.Sno,student.Sname,course.Cname,SC.Grade from student,course,SC where student.Sno=SC.Sno and course.Cno=SC.Cno;

+-----+---------+----------------------+-------+

| Sno | Sname | Cname | Grade |

+-----+---------+----------------------+-------+

| 1 | 宏志 | Linux中高级运维 | 4 |

| 2 | 王硕 | Linux中高级运维 | 3 |

| 3 | oldboy | Linux中高级运维 | 4 |

| 4 | 脉动 | Linux中高级运维 | 1 |

| 5 | oldgirl | Linux中高级运维 | 5 |

| 1 | 宏志 | Linux高级架构师 | 3 |

| 2 | 王硕 | Linux高级架构师 | 2 |

| 3 | oldboy | Linux高级架构师 | 4 |

| 4 | 脉动 | Linux高级架构师 | 1 |

| 5 | oldgirl | Linux高级架构师 | 3 |

| 1 | 宏志 | MySQL高级Dba | 1 |

| 2 | 王硕 | MySQL高级Dba | 2 |

| 3 | oldboy | MySQL高级Dba | 2 |

| 4 | 脉动 | MySQL高级Dba | 2 |

| 5 | oldgirl | MySQL高级Dba | 2 |

| 1 | 宏志 | Python运维开发 | 6 |

| 2 | 王硕 | Python运维开发 | 8 |

| 3 | oldboy | Python运维开发 | 8 |

| 4 | 脉动 | Python运维开发 | 3 |

| 5 | oldgirl | Python运维开发 | 9 |

+-----+---------+----------------------+-------+

20 rows in set (0.00 sec)

mysql> select student.Sno,student.Sname,course.Cname,SC.Grade from student,course,SC where student.Sno=SC.Sno and course.Cno=SC.Cno order by Sno;

+-----+---------+----------------------+-------+

| Sno | Sname | Cname | Grade |

+-----+---------+----------------------+-------+

| 1 | 宏志 | Linux中高级运维 | 4 |

| 1 | 宏志 | Linux高级架构师 | 3 |

| 1 | 宏志 | MySQL高级Dba | 1 |

| 1 | 宏志 | Python运维开发 | 6 |

| 2 | 王硕 | Linux中高级运维 | 3 |

| 2 | 王硕 | Linux高级架构师 | 2 |

| 2 | 王硕 | MySQL高级Dba | 2 |

| 2 | 王硕 | Python运维开发 | 8 |

| 3 | oldboy | Linux中高级运维 | 4 |

| 3 | oldboy | Linux高级架构师 | 4 |

| 3 | oldboy | MySQL高级Dba | 2 |

| 3 | oldboy | Python运维开发 | 8 |

| 4 | 脉动 | Linux中高级运维 | 1 |

| 4 | 脉动 | Linux高级架构师 | 1 |

| 4 | 脉动 | MySQL高级Dba | 2 |

| 4 | 脉动 | Python运维开发 | 3 |

| 5 | oldgirl | Linux中高级运维 | 5 |

| 5 | oldgirl | Linux高级架构师 | 3 |

| 5 | oldgirl | MySQL高级Dba | 2 |

| 5 | oldgirl | Python运维开发 | 9 |

+-----+---------+----------------------+-------+

20 rows in set (0.00 sec)

### 5.14.4 使用explain查看select语句的执行计划

即使用索引情况

mysql> desc test;

+-------+----------+------+-----+---------+----------------+

| Field | Type | Null | Key | Default | Extra |

+-------+----------+------+-----+---------+----------------+

| id | int(4) | NO | PRI | NULL | auto\_increment |

| name | char(20) | NO | | NULL | |

+-------+----------+------+-----+---------+----------------+

2 rows in set (0.00 sec)

#当没有建立索引时，查询了6行

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

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

id: 1

select\_type: SIMPLE

table: test

type: ALL

possible\_keys: NULL

key: NULL

key\_len: NULL

ref: NULL

rows: 6

Extra: Using where

1 row in set (0.00 sec)

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

Query OK, 0 rows affected (0.15 sec)

Records: 0 Duplicates: 0 Warnings: 0

mysql> desc test;

+-------+----------+------+-----+---------+----------------+

| Field | Type | Null | Key | Default | Extra |

+-------+----------+------+-----+---------+----------------+

| id | int(4) | NO | PRI | NULL | auto\_increment |

| name | char(20) | NO | MUL | NULL | |

+-------+----------+------+-----+---------+----------------+

2 rows in set (0.00 sec)

;#建立索引后值查询1行

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

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

id: 1

select\_type: SIMPLE

table: test

type: ref

possible\_keys: index\_name

key: index\_name

key\_len: 60

ref: const

rows: 1

Extra: Using where; Using index

1 row in set (0.00 sec)

使用explain命令优化SQL语句（select语句）的基本流程：

1、抓慢查询的SQL语句方法：

a.救火：紧急且重要： show full processlist ; #（登录数据库现场抓，连续执行两次，超过两秒）

show full processlist ; #数据库输入

mysql -uroot -poldboy3306 -S /data/3306/mysql.sock -e "show full processlist;"|egrep -vi "sleep" #Linux命令行输入

b.未雨绸缪：重要不紧急：分析慢查询日志

配置参数记录慢查询语句

Long\_query\_time = 2 #《--超过2秒，记录到log里。

Log\_queries\_not\_using\_indexes #《--没有走索引的语句，记录到log里。

Log-slow-queries = /data/3306/slow.log #《--往哪儿记录---log文件。

2、explain语句检查索引执行情况。

Explain select \* from test where name=’oldboy’\G

Explain select SQL\_NO\_CACHE \* from test where name=’oldboy’\G

3、对需要建索引的条件列建立索引

生产场景，大表不能高峰期建立索引，例如：300万几率。

4、分析慢查询SQL的工具mysqlsla（每天早晨发邮件）切割慢查询日志。

1） mv,reload 进程。

2） cp, >清空

3）定时任务。

5、日常优化

Dba,总监，开发，CTO联合解决。

官方手册需要大家掌握的章节5,6,7,8,10,11,13,14,15。

一个对索引效率的简单测试  
创建一个测试表

编写批量插入数据的简单存储过程  
create procedure pro\_test(in para int(11))   
begin        
declare i int default 0;   
declare rand\_num int;    
while i < para do   
select cast(rand()\*10000 as unsigned) into rand\_num;   
insert into test(num) values(rand\_num);   
set i = i +1;   
end while;   
end;   
执行存储过程，插入50W测试数据  
call pro\_test(500000);  
  
然后测试有无索引的效率吧

###### 企业案例：

优化的起因：

1） 网站出问题了，访问很慢。

a.web，存储，db（负载、磁盘IO、CPU）

登录：show full processlist;

2）慢查询语句（日志文件）

Long\_query\_time = 1

Log-slow-queries - /data/3306/slow.log

Log\_queries\_not\_using\_indexes

切割、分析、发给管理员

### 5.14.5 使用profile功能优化MySQL查询

SELECT @@profiling;

来查看是否已经启用profile，如果profilng值为0，可以通过

SET profiling = 1;

变量profiling是用户变量，每次都得重新启用。

来启用。启用profiling之后，我们执行一条查询语句，比如：

mysql> use oldboy;

mysql> SET profiling = 1;

mysql> SELECT @@profiling;

+-------------+

| @@profiling |

+-------------+

| 1 |

+-------------+

1 row in set (0.00 sec)

mysql> show tables;

+------------------+

| Tables\_in\_oldboy |

+------------------+

| course |

| sc |

| student |

| test |

+------------------+

4 rows in set (0.00 sec)

mysql> select count(\*) from test;

+----------+

| count(\*) |

+----------+

| 6 |

+----------+

1 row in set (0.00 sec)

mysql> show profiles;

+----------+------------+---------------------------+

| Query\_ID | Duration | Query |

+----------+------------+---------------------------+

| 1 | 0.00013225 | SELECT @@profiling |

| 2 | 0.00034100 | show tables |

| 3 | 0.00007150 | select count(\*) from test |

+----------+------------+---------------------------+

3 rows in set (0.00 sec)

**profile查询的博文：**

http://www.cnblogs.com/adforce/archive/2012/06/02/2532287.html

## 5.15 修改表中的数据

MySQL DLL语句最重要的命令就是查询

### 5.15.1 修改表中指定条件固定列的数据

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

###### 2、修改指定的行字段内容

a.查看要修改的表

mysql> select \* from test;

+----+----------+

| id | name |

+----+----------+

| 1 | oldboy |

| 2 | oldgirl |

| 3 | laoqiang |

| 4 | zuma |

| 5 | kaka |

| 6 | lifen |

+----+----------+

6 rows in set (0.00 sec)

b. 修改id为3的行的名字为gongli

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

mysql> select \* from test where id=3;

+----+--------+

| id | name |

+----+--------+

| 3 | gongli |

+----+--------+

1 row in set (0.00 sec)

### 5.15.2 修改表中所有行的数据

###### 3、严重的案例（可能无操作导致数据对视）

a.不带条件更改所有表的记录

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

b．更改了所有记录

mysql> select \* from test;

+----+--------+

| id | name |

+----+--------+

| 1 | gongli |

| 2 | gongli |

| 3 | gongli |

| 4 | gongli |

| 5 | gongli |

| 6 | gongli |

+----+--------+

6 rows in set (0.00 sec)

c.用备份的数据恢复

mysql> system ls -l /opt

总用量 12

-rw-r--r-- 1 root root 2758 12月 28 21:50 bak\_2015-12-28.sql

-rw-r--r-- 1 root root 4578 12月 29 13:36 bak\_2015-12-29.sql

mysql> source /opt/bak\_2015-12-29.sql

防止出错的例子：

http://oldboy.blog.51cto.com/2561410/1321061 DBA防止自己出错的设置

5.16 删除表中的数据

5.16.1 实践删除表中的数据

1、命令语法：delete from 表名 where 表达式

a.实践，例如：删除表test中编号为1的记录

mysql> delete from test where id=1;

Query OK, 1 row affected (0.00 sec)

mysql> select \* from test;

+----+---------+

| id | name |

+----+---------+

| 2 | oldgirl |

| 3 | gongli |

| 4 | zuma |

| 5 | kaka |

+----+---------+

4 rows in set (0.00 sec)

delete from test where id=1;

select \* from test;

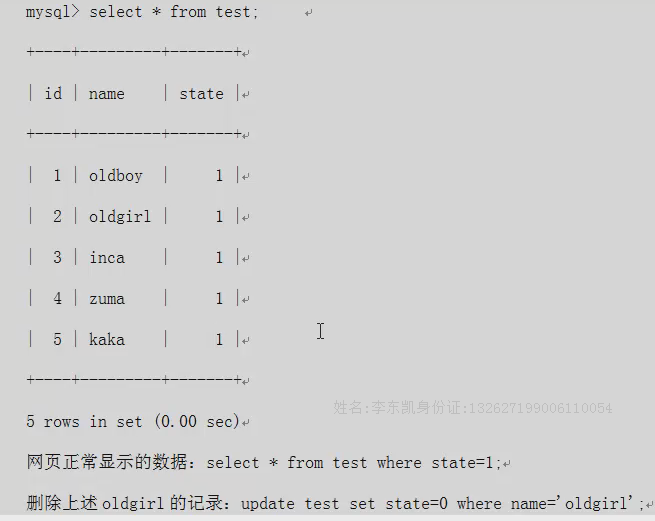
5.16.2 truncate table test;和delete from test;区别

1、truncate table test;速度更快，直接清空对应数据的物理文件。（物理删除）

2、delete from test;速度慢，逻辑清除，按行删。

5.16.3 通过update伪删除数据

在开发人员开发程序时，页面显示，一般是通过状态来判断的。举例：test表如下数据：



## 5.17 增删改表的字段

### 5.17.1 命令语法及默认添加演示

1、命令语法：alter table 表名 add 字段 类型 其他； #一般运维不需要操作

2、测试表数据

添加字段

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

mysql> desc test;

+-------+----------+------+-----+---------+----------------+

| Field | Type | Null | Key | Default | Extra |

+-------+----------+------+-----+---------+----------------+

| id | int(4) | NO | PRI | NULL | auto\_increment |

| name | char(20) | NO | | NULL | |

| sex | char(4) | YES | | NULL | |

+-------+----------+------+-----+---------+----------------+

3 rows in set (0.00 sec)

### 5.17.2 指定添加列在表里的位置

b．指定添加年龄列到name列后面的位置

alter table test add age int(4) after name;

mysql> alter table test add age int(4) after name;

mysql> desc test;

+-------+----------+------+-----+---------+----------------+

| Field | Type | Null | Key | Default | Extra |

+-------+----------+------+-----+---------+----------------+

| id | int(4) | NO | PRI | NULL | auto\_increment |

| name | char(20) | NO | | NULL | |

| age | int(4) | YES | | NULL | |

| sex | char(4) | YES | | NULL | |

+-------+----------+------+-----+---------+----------------+

4 rows in set (0.00 sec)

c. 若想添加在行首的命令：

alter table test add qq varchar(15) first;

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

mysql> desc test;

+-------+-------------+------+-----+---------+----------------+

| Field | Type | Null | Key | Default | Extra |

+-------+-------------+------+-----+---------+----------------+

| qq | varchar(15) | YES | | NULL | |

| id | int(4) | NO | PRI | NULL | auto\_increment |

| name | char(20) | NO | | NULL | |

| age | int(4) | YES | | NULL | |

| sex | char(4) | YES | | NULL | |

+-------+-------------+------+-----+---------+----------------+

5 rows in set (0.00 sec)

删除的命令：

alter table test drop qq;

mysql> alter table test drop qq;

mysql> desc test;

+-------+----------+------+-----+---------+----------------+

| Field | Type | Null | Key | Default | Extra |

+-------+----------+------+-----+---------+----------------+

| id | int(4) | NO | PRI | NULL | auto\_increment |

| name | char(20) | NO | | NULL | |

| age | int(4) | YES | | NULL | |

| sex | char(4) | YES | | NULL | |

+-------+----------+------+-----+---------+----------------+

4 rows in set (0.01 sec)            
alter table test drop age;

### 5.17.3 生产环境多个复杂添加修改多字段信息的案例

1、增加1个字段：

ALTER TABLE `etiantian` ADD `FIRSTPHOTO\_URL` varchar(255) default NULL COMMENT '第一张图片URL'

2、增2个字段：

ALTER TABLE `basic` ADD `adhtml\_top`  varchar(1024) default NULL COMMENT '顶部广告html' , ADD `adhtml\_right` varchar(1024) default NULL COMMENT '右侧广告html' ;

3、改变字段：

alter table ett\_ambiguity change ambiguity\_state  ambiguity\_state tinyint  comment '状态，默认1=正常，0=失效';  
ALTER TABLE `ett\_photo`  
MODIFY COLUMN `PHOTO\_DESCRIPTION` varchar(512) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL COMMENT '描述' AFTER PHOTO\_TITLE`;

4、修改字段类型：

mysql> alter table test drop qq;

mysql> alter table test modify age char(4) after name;          
Query OK, 6 rows affected (0.00 sec)  
Records: 6  Duplicates: 0  Warnings: 0

5、修改字段名称

mysql> alter table test change age oldboyage char(4) after name;     
Query OK, 6 rows affected (0.01 sec)  
Records: 6  Duplicates: 0  Warnings: 0

提示：工作中添加字段需求来自开发，运维或DBA拿着开发给的语句执行。

## 5.18 更改表名

### 5.18.1 rename的用法

1、命令语法：rename table 原表名 to 新表名

法1：

mysql> rename table test to oldboy;     
mysql> show tables;

+------------------+

| Tables\_in\_oldboy |

+------------------+

| course |

| oldboy |

| sc |

| student |

+------------------+

4 rows in set (0.00 sec)

法2：

mysql> alter table oldboy rename to test;  
mysql> show tables;

+------------------+

| Tables\_in\_oldboy |

+------------------+

| course |

| sc |

| student |

| test |

+------------------+

4 rows in set (0.00 sec)

## 5.19 删除表

命令语法： drop table <表名>

例如：删除表名为test的表

mysql> show tables;

+------------------+

| Tables\_in\_oldboy |

+------------------+

| course |

| sc |

| student |

| test |

+------------------+

4 rows in set (0.00 sec)

mysql> drop table test;

Query OK, 0 rows affected (0.01 sec)

mysql> show tables;

+------------------+

| Tables\_in\_oldboy |

+------------------+

| course |

| sc |

| student |

+------------------+

3 rows in set (0.00 sec)