

MySQL和MariaDB数据库介绍

[MySQL和MariaDB数据库介绍](#)

[数据库的基础概念](#)

[数据库的分类](#)

[什么是MySQL](#)

[如何获得MySQL相关资源](#)

[MySQL在企业中的应用场景](#)

[MySQL 数据库安装](#)

[RHEL 7.2 RPM 包安装 MariaDB 5.5](#)

[RHEL 7.2 RPM 包安装 MariaDB 10.2](#)

[RHEL 7.2 RPM 包安装 MySQL 5.7](#)

[RHEL 7.2 二进制文件安装 MySQL 5.6](#)

[MySQL客户端连接数据库](#)

[MySQL客户端的使用](#)

[python连接MySQL数据库](#)

[PHPmyAdmin在线工具使用](#)

[MySQL Workbench 连接数据库](#)

数据库的基础概念

数据库(**database**): 保存有组织的数据的容器(通常是一个文件或一组文件)。

数据库管理系统(**DBMS**): 顾名思义,既然是数据管理系统,就是用来管理数据的数据库软件

* 线上生产环境中有很多数据,举例说明,论坛、qq、微博、朋友圈等等。 *

在我们线上生产环境当中有很多数据,那么用户比如说登一个论坛发了个帖子,这个帖子里的内容就是一个数据;用户登陆一个购物网站,买了一个商品,用户购买一个商品是一个订单信息,也是一个数据,各种各样的环境,我们会发现,在我们的线上应用当中,存在很多用户信息,他购买商品也好,发帖子也好,或者他打游戏注册帐号也好,等等,这些东西其实都是信息,我们需要将这些信息保存下来,因为以后每次用户登陆时都需要反复的读写,读写你购买过的商品,读写你发过的帖子,随着我们用户数量越来越多,这种类似的数据会越来越多,从一开始的一个用户到后面的一万个用户,到成年累月下来,我们发现论坛已经有了几千万条帖子,每个帖子又有很多的回复,这时候你会发现我们的数据量越来越大。

* 数据早期保存的方式,文本文件,介绍读取方式、优缺点,举例说明,做一个小实验。 *

那么在早些年的时候呢,我们说最早数据保存的方法是文本方式保存,文本文件保存,就是说我们网站上有一千万个帖子,我们把这一千万个帖子放在一个文本文档里,当用户登陆后要读帖的时候,就要从这个文本文件中去读,后来我们发现,当文本文件行数比较少的时候,几百行几千行几万行还行,要是几十万行几百万行的话,你会发现读取的时候非常的慢,慢的话,有的时候需要半分钟,有的时候可能需要五分钟,甚至有时候半小时,那么很明显这个速度是不行的,没有哪个用户说愿意为了看个帖子,在那里等个半小时。看到你帖子出来了,好,开始回帖,然后一个回帖操作完成又要等半个小时,才能看到回帖的结果。很明显不行,所以呢,我们会发现文本文档方式保存信息有一个非常大的缺点,就是我们的读写速度,随着文件内容的增加,他的读写速度会越来越慢,关于这一点,你们可以自己做个实验测以下,写个几十万行的数据2-3个G,过滤出某一行,或者读一下,看看要花多上多少时间。

```
#!/bin/bash
touch /tmp/a
for i in `seq 1 500000`
do
    sed -n '1,$r /var/log/messages' /tmp/a
done
```

* 我们希望的数据读取方式是怎样的呢？引出数据库的概念。 *

我们在操作过程的当中，并不希望把所有的数据都读取出来，很多时候我可能只需要读取其中的一行，而文本文档的操作命令，属于牛操作，他需要从头部开始做，很多时候他过滤的话都会从头到尾寻找一遍，才能找到你需要的那一行数据，或者多行数据，那么在这个过程中会极大地消耗性能。我们希望实现的效果是，如果我们需要取其中的几行，则迅速地定位到那几行数据，也就是说我不单要知道文本文件所在的位置，我还知道文本文件当中每一行所在的系统的位置，不单是系统还有硬盘上的位置，他在硬盘的第几个磁道，第几个扇区，我要第十行到哪里去找，取第1000行到哪里去找，不需要从头部去找，这是一种文件的操作方法，或者是数据的操作方法，于是就有了我们的数据库。

数据库如果要实现这个功能，在几十G的文件中迅速定位到你需要的数据，他需要**索引**，所以尽量去保存可索引数据，这样才会让读写性能有所提高。但是总有人把图片视频（二进制数据）放到数据库中，数据库是没办法对二进制文件内容做索引的。数据分可索引数据和不可索引数据。但也不是说二进制数据就不能放在数据库中，只是说放进去对性能没有提高。比如说做个论坛，如果发帖有附件，那么附件是上传到网站根目录下的，在做备份的时候，数据库要备份，网站根目录下的文件要备份，有人呢就偷懒将所有数据都放到数据库下，备份的时候就只要备份数据库了，但是我不建议这么做，因为本身数据库已经很大了，你再把这些二进制文件往里面放会导致数据库更大，更加臃肿。而且数据库本身有很多缓冲，你放的文件太大，缓冲就用得多，缓冲用的多，内存消耗大，数据库的性能消耗会明显上升。所以自己去看，尽量不要去保存不可索引的数据。

数据库的分类

数据库类型	描述	主流产品	有谁在用	适用场景	不适用场景
关系型数据库	关系型数据库就是由二维表及其之间的联系组成的一个数据组织	Oracle、DB2、PostgreSQL、Microsoft SQL Server、Microsoft Access、MySQL、MariaDB	alibaba,sina,网易, youtube, google等	保持数据的一致性（事务处理）；由于以标准化为前提，数据更新的开销很小（相同的字段基本上都只有一处）；可以进行Join等复杂查询；能够保持数据的一致性关系型数据库的最大优势	大量数据的写入处理；为有数据更新的表做索引或表结构(schema)变更；字段不固定时应用；对简单查询需要快速返回结果的处理

数据库类型	描述	主流产品	有谁在用	适用场景	不适用场景
键值（Key-Value）数据库	键值数据库就像在传统语言中使用的哈希表。你可以通过key来添加、查询或者删除数据，鉴于使用主键访问，所以会获得不错的性能及扩展性。	Riak、Redis、Memcached、Amazon's Dynamo、Project Voldemort	GitHub（Riak）、BestBuy（Riak）、Twitter（Redis和Memcached）、StackOverFlow（Redis）、Instagram（Redis）、Youtube（Memcached）、Wikipedia（Memcached）	储存用户信息，比如会话、配置文件、参数、购物车等等。这些信息一般都和ID（键）挂钩，这种情景下键值数据库是个很好的选择。	1. 取代通过键查询，而是通过值来查询。 Key-Value 数据库中根本没有通过值查询的途径。 2. 需要储存数据之间的关系。在 Key-Value 数据库中不能通过两个或以上的键来关联数据。 3. 事务的支持。在 Key-Value 数据库中故障产生时不可以进行回滚。

数据库类型	描述	主流产品	有谁在用	适用场景	不适用场景
面向文档数据库 (Document-Oriented) 数据库	<p>面向文档数据库会将数据以文档的形式储存。每个文档都是自包含的数据单元，是一系列数据项的集合。每个数据项都有一个名称与对应的值，值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，同一个表中存储的文档属性可以是不同的，数据可以使用 XML、JSON 或者 JSONB 等多种形式存储。</p>	MongoDB、CouchDB、RavenDB	SAP (MongoDB)、Codecademy (MongoDB)、Foursquare (MongoDB)、NBC News (RavenDB)	<p>1. 日志。企业环境下，每个应用程序都有不同的日志信息。</p> <p>Document-Oriented 数据库并没有固定的模式，所以我们可以使用它储存不同的信息。</p> <p>2. 分析。鉴于它的弱模式结构，不改变模式下就可以储存不同的度量方法及添加新的度量。</p>	在不同的文档上添加事务。 Document-Oriented 数据库并不支持文档间的事务，如果对这方面有需求则不应该选用这个解决方案。

数据库类型	描述	主流产品	有谁在用	适用场景	不适用场景
列存储（Wide Column Store/Column-Family）数据库	<p>列存储数据库将数据储存在列族（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。</p>	Cassandra、HBase	<p>Ebay（Cassandra）、Instagram（Cassandra）、NASA（Cassandra）、Twitter（Cassandra and HBase）、Facebook（HBase）、Yahoo!（HBase）</p>	<p>1. 日志。因为我们可以将数据储存在不同的列中，每个应用程序可以将信息写入自己的列族中。2. 博客平台。我们储存每个信息到不同的列族中。举个例子，标签可以储存在一个，类别可以在一个，而文章则在另一个。</p>	<p>1. 如果我们不需要 ACID 事务。 Vassandra 就不支持事务。2. 原型设计。如果我们分析 Cassandra 的数据结构，我们就会发现结构是基于我们期望的数据查询方式而定。在模型设计之初，我们根本不可能去预测它的查询方式，而一旦查询方式改变，我们就必须重新设计列族。</p>

数据库类型	描述	主流产品	有谁在用	适用场景	不适用场景
图（Graph-Oriented）数据库	图数据库允许我们将数据以图的方式储存。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、Apple 和 Next，则会有两个“Founded by”的边将 Apple 和 Next 连接到 Steve Jobs。	Neo4J、Infinite Graph、OrientDB	Adobe（Neo4J）、Cisco（Neo4J）、T-Mobile（Neo4J）	1. 在一些关系性强的数据中2. 推荐引擎。如果我们将数据以图的形式表现，那么将会非常有益于推荐的制定不适合的数据模型。图数据库的适用范围很小，因为很少有操作涉及到整个图。	

关系模型就是指二维表格模型,因而一个关系型数据库就是由二维表及其之间的联系组成的一个数据组织。当前主流的关系型数据库有 Oracle、DB2、PostgreSQL、Microsoft SQL Server、Microsoft Access、MySQL、MariaDB、浪潮 K-DB 等。

什么是MySQL

什么是mysql? 摘自官方一个解释, 说 ** “MySQL 是采用客户/服务器模型的开放源码关系型 SQL 数据库管理系统,它可以在多种操作系统上运行,它是由 MySQL Ab 公司开发、发布并支持的, 后被sun收购, 现在在oracle公司旗下, 现在有一个知名的分支MariaDB。” ** 稍微有点长, 我们一段一段来解释, 里面有一些关键点, 能够帮助你非常有效地去理解MySQL。

* 第一个要点, 客户/服务器模型。 *

通过C/S和B/S的对比去介绍为什么mysql选择c/s模型, 分别从方便、性能、稳定、安全四个方面来讲。

mysql是一个c/s模型地一套应用程序, 其实讲到c/s模型大家就会联想到, b/s模型, c/s叫做客户端/服务器, b/s叫做浏览器/服务器。服务器还是服务器, 只不过客户端是装一个应用服务呢, 还是直接用浏览器。那么两种模型呢各有优缺点, 如果我们希望使用起来更方便, 那使用浏览器是更方便的, 因为不管什么系统都自带浏览器, 比如手机自带、pad自带, 各种环境都有浏览器, 这种情况比较方便; 但是浏览器主要通过网络, 那么性能、稳定、安全性就会受到一定地影响。而c/s模型就不是, 我装一个客户端软件, 客户端和服务端地通信是走自己专有地协议的, 他们可以实现一个高效、稳定的通信, 所以c/s模型他最大的优点是性能高、稳定好、功能多、安全性好, 各种各样的优点, 相对于b/s模型来说要装一个客户端软件, 用起来没有他方便。我们mysql用的是c/s模型, 因为我们不是考虑让你用起来方便, 我们的关注点就是性能更加高效, 关注的就是安全, 数据库里面的数据非常的重要, 经常我们的用户名和密码都是放在数据库里面, 数据库泄露会导致很多问题。大家平时在上网过程中有没有曾经遇到过一些

情况，比如登陆微博、QQ，登陆的时候他告诉你，你的密码不够安全，要求你修改密码，之前有没有遇到过，莫名其妙，登上来就告诉你修改密码，凭什么说你密码不安全，我告诉你哪怕你密码设得再复杂他也会跟你讲你的密码不安全，原因是因为他数据库泄露过了，他数据库泄露了又不能直接跟你讲：“我数据库泄露了，你快改密码阿！”他不能说的，只能告诉你你的密码不安全，让你改。如果你没有修改，很可能就会被盗号了。从这个例子我们就可以知道数据库的安全性问题的重要性。包括去年很火的，非常出名的一件事，12306数据库泄露，大家买过的火车票，买火车票时使用的身份证、用户名、密码、银行信息就都泄露了，泄露了大概十几个G的用户信息，知道吗？在12306上买过火车票吗？买过，那很可能你的身份证信息已经被泄露了。所以我们这里讲数据库关注的点，不是让你使用起来更方便，而是要保证性能、安全、稳定。稳定也是一个比较重要的地方，数据库不能出错，不能说连着连着，一会连不上了。今天去存钱，存完钱明天去取钱，存了多少钱不知道，为什么呢？数据库没连上，你昨天存的钱我找不到，这是一个稳定性。你要保证数据库7*24小时在线运行，不能出任何问题。好这就是我们讲的c/s模型。

* 第二个要点，开放源码。 *

介绍open source和closed source的概念和区别，介绍mysql的开源理念以及历史，介绍“去IOE”以及Mariadb的诞生。

什么叫做开源？open source和他相对应的就是闭源closed source应用，闭源的东西又叫做copyright（版权、著作权），微软的东西一般都是copyright；而开源的叫做copyleft（非盈利版权，公共版权）。公司赚钱有两个大的流派，一个流派就是以微软、oracle为首的闭源软件起家的，他们认为程序写完之后你得拿去卖，卖代码来赚钱，我卖给你使用，你给我钱，这样的话呢，我才能活得更滋润一点；开源应用不是，他认为我程序写完之后，可以免费给你使用，然后你要是觉得有问题，你告诉我你做过修改，你也应该把他开源出来，我们开源的人越多，修改的人越多，程序就会越做越好，不反对你赚钱，但是你赚钱的收入点不是在程序本身，而是在于你后期所提供的服务，提供的一些额外的操作，这是开源领域的一些理念，以redhat为首，他就是以开源为目标的。

mysql开始设计的时候，就是一个开源产品，当然这是以前的事情了，mysql当年是开源的，然后被收购了，其实我们说阿，做开源的人呢，赚钱最好的方式就是找人把他给收了，收购的话基本上能赚一大票，收购mysql的是当时比较著名的sun公司，有很多非常经典的产品都是sun开发出来的，比如他的java编程语言。除了java语言以外sun公司还开发过很多经典的技术，那么sun公司的话，有钱有实力，但是呢不太会经营，最后就倒闭了，然后被另外一家收购，现在mysql所属于oracle。当年收购完mysql之后呢，他觉得，我的oracle产品面向于大型的公司，我需要有一个产品来弥补中的端企业的空缺，于是他就收购了mysql，收购完之后本身觉得没什么位难题，后来mysql在oracle的带领下越做越好，因为oracle在数据库领域算得上是排名第一了。他有很多专有的数据库技术，能够让数据在读写过程当中，性能非常高，然后又有高可用，有很多的有点，mysql越做越好，好到什么程度呢？不单单是中小型企业在使用，大型企业也开始用mysql，用下来觉得比oracle要好，出现这种现象，以至于我们在前几年的时候有人提出了一个，俗称为IT界的革命，叫做“去IOE”。

什么叫做“去IOE”呢？三个非常著名的公司，IBM、ORACLE、EMC，这三个公司有业界一流的产品，IBM有他的小型机，从服务器的性能上来讲小型机已经算是顶尖的，比你的pc server要好的多；然后oracle的数据业界顶尖的；EMC的存储设备也是业界顶尖的。经常呢有公司说我的压力特别大，我需要去买最好的硬件，就买这三家公司的，放在一起组成一套性能非常强大的这么一套硬件环境。后来这些公司发现这样不划算，因为你每年买这些硬件要花几千万，这些硬件非常贵的阿，像IBM的小型机几十万上百万；存储，没有一百万你别谈，谈不了，以后要是老板跟你说，要你搭一套商业的高性能的存储方案，特别是数据库用的，你先问他有没有一百万，一百万都不给我，拿什么去谈，谈不了；oracle也不说了，按cpu给你卖钱，看你的服务器几个cpu是吧，现在服务器多核，多核你就多付钱吧。后来公司说我一年买几千万，都没赚多少，赚的钱一大半用来买硬件了，能不能成本低一点。

IBM的小型机换成了pc server，比我们这里的服务器稍微好一点，我们是1U的，他们起码2U的，买一些pc server，单台服务器的性能比IBM的小型机要低，但是我们现在有非常成熟的集群解决方案，有高可用集群、负载均衡集群、高性能计算集群，当多台pc server放在一起的时候，他整体价格要比IBM的便宜，而且提供的整体性能也比IBM高，无非就是浪费一些地方，但是会发现得到的是成本降低、收益更好。一台IBM上百万，我买10台pc server性能就超过你了，所以pc server划算。

oracle这里不说了，你要收钱的，我就用不收钱的mysql，用mysql跑跑，当然了，现在有很多大公司会把oracle换成mysql，也不是全部都换掉，有一些非常核心的应用还是跑在oracle上面，oracle还是有他的有点的。存储这里

呢，我们现在有isdn的存储，以后讲到存储会去讲，isdn存储比我们学过的nfs、samba的性能要高，但是成本低，不需要拿个几百万出来。针对这些情况，IBM怎么想的我不知道，EMC怎么想的我也不知道，但是我知道oracle怎么想的。oracle说我收购mysql是想让中小型企业用的，好阿，你们现在都用了是吧，oracle卖不出去了，那怎么办呢？是大家别用呢？还是怎么做比较好？把mysql掐死，你们不是觉得mysql开源不要钱吗，反正他是领养的孩子，oracle才是我自己的孩子，领养的孩子抢自己孩子的蛋糕，那我就掐死他。好了，oracle决定把mysql闭源，慢慢地将mysql的技术做逐步闭源。那他一做闭源，用户就发现问题了，我开始用mysql，结果你先在mysql也要收钱，那么最关键的是mysql的开发者，当年最早地开发者，一个开始在mysql-AB公司，后来被sun收购后去了sun公司，sun公司也是秉持开源理念地，所以也做得比较开心，oracle收购完之后呢，开始也开源，后来呢变闭源了，这些人受不了了，这些人不在乎赚多少钱，在乎地是个人理想。我要实现个人理想，我的理想就是把这个软件越做越好，然后让大家免费使用，这是个人理想，这些人称之为黑客。哪些人叫黑客，搞破坏的那些人不是黑客，崇尚开放、平等、自由的，然后又有高超的计算机水平的这些人就是黑客。就像linux的创始人，Linus Torvalds林纳斯 托瓦兹，他们都是非常著名的黑客。既然你mysql要收钱，那我不干了，所以当年mysql最早的开发者其中的一个领头，他就跳槽出来了，然后带了一批手下做了一个开源社区又吸收了一批人，开始写一个新的数据库，就是我们rhel7当中的mariadb。

Mariadb，有很多地方跟mysql很像，你会发现在操作mariadb的过程中用到的都是mysql的命令，登陆mysql、备份mysqldump，你会发现mariadb所有的接口都和mysql的接口一模一样，内部的函数名都和mysql的一模一样。所以呢，你可以很方便地从mysql的环境迁移到mariadb上面。mariadb也做得非常好，刚开始比不过mysql，但是近两年mysql有的优点他都有，甚至他还有很多自己特有的优点，oracle这时候坐不住了，我现在闭源，闭源以后呢没人来买我的oracle，大家换成mariadb了，那我这个闭源的意义就没了，所以最近呢又发表声明，说“大家放心，我以后不会再闭源了！mysql以后还会以开源的方式一直存在下去。”不会收钱了，但是也没什么人理他了，更多的人都愿意相信，愿意去使用mariadb了。mysql也有人用，mariadb也有人用，现在就变成这个市场，mysql和mariadb各占了一部分江山。这个呢就是开放源码。

* 第三个要点，关系型。 *

介绍关系型和非关系型数据库的概念和优缺点

所谓关系型是指,将数据保存在不同的表中,表与表之间支持一定的关系，而不是将所有数据放在一个大的仓库内。这样就增加了速度并提高了灵活性。

什么是关系型数据库？我们在存放数据的时候，数据很多，几百万几千万行，如果说我们把所有的数据放在一个文件里面，虽然数据库的读写性能会比一般的程序来的高，但是文件太大总归多多少少会影响一些性能，为了能够让我更快地读取数据，我会考虑把数据拆分开来，不同的数据放在不同的文件当中，而数据库里面基本的存储方式叫做表。也就是说我们的数据是放在不同的表里面。数据库管理员管什么东西呢，管表，所以我们经常是这么说的，叫数据库管理员，男的叫表哥，女的叫表姐，老大叫做大表哥。那么我现在把数据放在不同的表里面，表和表之间能不能支持一定的关系，因为我们经常在操作表里的数据的时候是有关系的，比如说，我可能有个表叫员工表，有个表叫工资表，工资表里面发钱，发钱的时候呢，每一个发工资的人应该在员工表里面存在，你不能说发钱发给不是我们公司的人，不认识的人，这不行。所以呢，这是一个表和表之间的关系。能够把这些关系实现出来的，我们一般把他叫做关系型数据库。能够以表来存储，并在表和表之间支持一定的依赖关系，各种各样的依赖关系，可能是大小值的判断，可能是存在的判断，等等。当然我们除了关系型数据库以外呢，还有一种数据库叫做，键值型数据库，比如memcache\redis数据库，典型的代表例子，优点是性能更加高，基本都是使用内存来存放数据，但是能实现的功能很简单，他不是基于表的，功能很简单，你只能查a是多少，什么值是多少。

* 第四个要点，SQL语句。 *

介绍SQL语句的概念

MySQL 中的 SQL 指的是“结构化查询语言”。SQL 是访问数据库的最常用的标准化语言。他是统一的操作语法，我们说的是数据库统一的操作语法。也就是说在mysql里面学的sql语句以后你在其它数据库中也同样适用。每个数据库自己的sql语句很少，百分之九十五都是一样的。Sql语句时间不够学没关系，后面还有oracle课程，会去详细学习sql语句。

我们再回顾一下mysql的定义。MySQL 是采用客户/服务器模型的开放源码关系型 SQL 数据库管理系统,它可以在

多种操作系统上运行,不要以为他只能在linux系统上运行, windows系统也能运行。应该这么说, Mysql在linux上运行性能更好, 稳定性更高, 包括oracle也是, oracle也能装在windows上面, 但是跟装在linux上面性能不一样, oracle官方推荐安装在linux上面, mysql也有windows版本, 整个安装过程相当简单。可以去下载一个“wamp.exe”软件, 下载下来一路next安装到底, 就类似与LAMP的环境, 就搭建好了。这是多平台运行。Mysql从mysqllab公司到sun公司再到oracle公司。目前是在oracle公司旗下。

如何获得MySQL相关资源

- 为了学习更多的MySQL知识,请访问MySQL官网 <http://dev.mysql.com/>。
- 为了下载服务器的一个副本,请访问MySQL官网 <http://dev.mysql.com/downloads/>。



MySQL在企业中的应用场景

MySQL 是世界上 ** 最受欢迎的开源数据库 **,她拥有相当大的装机量。而且 DB-Engines 的排名一直处于数据库总榜第二名的位置,仅次于 Oracle。MySQL 在开源领域排名第一,而第二大开源数据库 PostgreSQL 的分数仅仅是MySQL 的零头。

MySQL 拥有庞大的用户群,国外的有 ** Facebook、Flickr、eBay ** 等,国内的有 ** 阿里、腾讯、新浪、百度等 **。而这些互联网和大部分传统公司的服务需要 ** 7×24 ** 小时连续工作。当此类型网站的部分数据库服务器宕机时,就需要高可用技术将流量牵引至备份主机,从而此时这些公司需要通过 ** 备份和恢复 ** 手段来产生备机,并通过 ** 复制 ** 来同步主备机间的状态,同时部署各种 ** 监控 ** 软件来监控服务器状态。当异常数据库服务器宕机时,通过手工或自动化手段将主机流量切换至备机,这个动作叫作 ** failover **。而一些大型公司在面对成千上万台 MySQL 服务器时,通常使用 ** 自动化运维脚本 ** 或程序完成上述种种动作。

MySQL 数据库安装

课程要求:

- 学会MariaDB 5.5的安装
- 学会MariaDB 10.2的安装

num	安装方法	说明	是否编译好
<ul style="list-style-type: none">• 学会MySQL 5.6的安装• 学会MySQL 5.7的安装• MySQL自动安装脚本			

安装方法

num	安装方法	说明	是否编译好
a	二进制文件	解压即可	编译好的
b	包管理	rpm,deb	编译好的，有版本(推荐使用)
c	源代码	类似Mplayer	自己编译（按需选择参数：代码优化）

RHEL 7.2 RPM 包安装 MariaDB 5.5

项目	参数
软件名	mariadb-server 5.5
service	mariadb
daemon	mysqld
配置文件	/etc/my.cnf /etc/my.cnf.d/*.cnf
数据文件	/var/lib/mysql
日志文件	/var/log/mariadb/mariadb.log（错误日志，启动日志）
端口号	3306

安装必要的软件包

```
[root@mastera0 ~]# yum install -y vim net-tools wget
```

```
[root@mastera0 ~]# yum list|grep mariadb
```

Repodata is over 2 weeks old. Install yum-cron? Or run: yum makecache fast

mariadb-libs.x86_64	1:5.5.44-2.el7	@anaconda/7.2
mariadb.x86_64	1:5.5.44-2.el7	rhel_dvd
mariadb-bench.x86_64	1:5.5.44-2.el7	rhel_dvd
mariadb-devel.i686	1:5.5.44-2.el7	rhel_dvd
mariadb-devel.x86_64	1:5.5.44-2.el7	rhel_dvd
mariadb-libs.i686	1:5.5.44-2.el7	rhel_dvd
mariadb-server.x86_64	1:5.5.44-2.el7	rhel_dvd
mariadb-test.x86_64	1:5.5.44-2.el7	rhel_dvd

```
[root@mastera0 ~]# yum install -y mariadb-server
```

查看软件架构

```
[root@mastera0 ~]# rpm -ql mariadb-server
```

启动服务

```
[root@mastera0 ~]# systemctl start mariadb
```

查看守护进程

```
[root@mastera0 ~]# ps -ef|grep mysqld
```

mysql	2496	1	0	13:56	?	00:00:00	/bin/sh /usr/bin/mysqld_safe --basedir=/usr
mysql	2653	2496	0	13:56	?	00:00:00	/usr/libexec/mysqld --basedir=/usr -- datadir=/var/lib/mysql --plugin-dir=/usr/lib64/mysql/plugin --log- error=/var/log/mariadb/mariadb.log --pid-file=/var/run/mariadb/mariadb.pid -- socket=/var/lib/mysql/mysql.sock
root	2694	2347	0	13:56	pts/0	00:00:00	grep --color=auto mysqld

查看监听端口号

```
[root@mastera0 ~]# netstat -lntlp|grep mysqld
```

tcp	0	0	0.0.0.0:3306	0.0.0.0:*	LISTEN	2653/mysqld
-----	---	---	--------------	-----------	--------	-------------

我们任何一个程序安装完成之后都要去看一下他的组成 `rpm -ql mariadb-server`，我们的每个程序都有这样几个部分，配置文件目录、脚本文件目录、数据文件目录、日志文件目录等。我们可以用 `rpm -ql <程序名>` 来查看程序的组成。我们可以看到 `mariadb-server` 的配置文件为 `/etc/my.cnf`；脚本文件目录为 `/usr/local/mysql/bin/`；数据文件目录为 `/var/lib/mysql/`；日志文件在 `/var/log/mysql.log`。接下来我们详细介绍每一下脚本文件目录中的脚本。

MySQL服务器和服务启动脚本:

Mysqld MySQL服务器

mysqld_safe、mysql.server和mysqld_multi是服务器启动脚本

mysql_install_db 初始化数据目录和初始数据库

访问服务器的客户程序:

mysql 是一个命令行客户程序,用于交互式或以批处理模式执行SQL语句

mysqladmin 是用于管理功能的客户程序

mysqlcheck 执行表维护操作

mysqldump和mysqlhotcopy负责数据库备份

mysqlimport 导入数据文件

mysqlshow 显示信息数据库和表的相关信息

独立于服务器操作的工具程序:

myisamchk 执行表维护操作

myisampack 产生压缩、只读的表

mysqlbinlog 是处理二进制日志文件的实用工具

perror 显示错误代码的含义

MySQL客户端的使用

项目	参数
软件名称	mariadb 5.5
命令	mysql 登陆连接mysql服务器
	mysqladmin 修改数据库服务器用户密码
	mysqldump 备份
	mysqlbinlog 二进制日志的查看

命令的使用

mysql

1.服务启动后, mariadb5.5直接登陆, 不需要密码

2.退出 \q exit ctrl+d

-u 用户名 空格可有可无 -u root;-uroot

-p 密码 不可以有空格 -puplooking

mysql -uroot -puplooking

mysql -uroot -puplooking123

mysqladmin

1.无密码情况下添加密码

mysqladmin -uroot password 'uplooking'

-u 用户名 空格可有可无 -u root;-uroot

password 新密码 一定要有空格

2.有密码情况下修改密码

mysqladmin -uroot -puplooking password 'uplooking123'

-u 用户名 可有可无

-p 当前密码 不能有

password 新密码 有

```
[root@mastera0 ~]# mysql==>未设置密码登陆
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MariaDB connection id is 2
Server version: 5.5.41-MariaDB MariaDB Server
Copyright (c) 2000, 2014, Oracle, MariaDB Corporation Ab and others.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
MariaDB [(none)]> \q
Bye
[root@mastera0 ~]# mysqladmin -uroot password "uplooking"==>设置密码
[root@mastera0 ~]# mysql -uroot -p==>登陆
Enter password:
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MariaDB connection id is 4
Server version: 5.5.41-MariaDB MariaDB Server
Copyright (c) 2000, 2014, Oracle, MariaDB Corporation Ab and others.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
MariaDB [(none)]> show databases==>;分号提交命令
+-----+
| Database
|
+-----+
| information_schema |==>临时数据库
| mysql |==>做 mysql 初始化的库
| performance_schema |
| test |==>临时共享库,任何人都可以看
+-----+
4 rows in set (0.00 sec)
[root@serverg ~]# cat ~/.mysql_history==>查看之前的命令但是不全
Bye
```

RHEL 7.2 RPM 包安装 MariaDB 10.2

项目	参数
软件名	MariaDB-server 10.2
service	mariadb/mysql
daemon	mysqld
配置文件	/etc/my.cnf
数据文件	/var/lib/mysql/
日志文件	默认没有开启
端口号	3306

软件的获取

<http://classroom.example.com/materials/mariadb-10.2.repo>

卸载冲突包mariadb-libs再安装mariadb-10.2

```
# rpm -e mariadb-libs --nodeps
# yum install -y MariaDB-server
```

初始化MariaDB数据库并启动MariaDB服务，并作安全加固

```
# systemctl start mysql
# mysql_secure_installation
```

查看MariaDB数据库实例监听的端口

```
# netstat -ntpl|grep sql
tcp        0      0 0.0.0.0:3306          0.0.0.0:*           LISTEN     12877/mysqld
tcp        0      0 0.0.0.0:4567          0.0.0.0:*           LISTEN     12877/mysqld
```

客户端

项目名	参数
软件名	MariaDB-client 10.2
命令	mysql,mysqladmin,mysqlbinlog,mysqldump

RHEL 7.2 RPM 包安装 MySQL 5.7

项目名	参数
软件名	mysql-community-server 5.7
service	mysqld
daemon	mysqld
配置文件	/etc/my.cnf, /etc/my.cnf.d/*.cnf
数据文件	/var/lib/mysql
启动日志	/var/log/mysqld.log

软件的获取

<http://classroom.example.com/materials/mysql-5.7.repo>

卸载冲突包mariadb-libs再安装mysql-5.7

```
# rpm -e mariadb-libs --nodeps
# yum install -y mysql-community-server
```

初始化MySQL数据库并启动mysqld服务，并作安全加固

```
# systemctl start mysqld
# mysql_secure_installation
```

服务端修改初始密码

```
# grep password /var/log/mysqld.log
# mysqladmin -uroot -p'' password '(Uploo00king)'
```

客户端

项目名	内容
软件名	mysql-community-client 5.7
命令	mysql,mysqladmin,mysqlbinlog,mysqldump

客户端登陆数据库

```
mysql -uroot -p'(Uploo00king)'
```

mysql.user表的结构变化了，原先的password列改为了authentication_string列

RHEL 7.2 二进制文件安装 MySQL 5.6

mysql	软件架构
数据目录	/data/mysql/data
binlog目录	/data/mysql/log-data
pid文件	/data/tmp
临时目录	/data1/tmp/

以上目录所有者和所属组都需要为mysql.mysql


```
[root@mastera0 ~]# tar -xf mysql-5.6.20-linux-glibc2.5-x86_64.tar.gz
[root@mastera0 ~]# cd mysql-5.6.20-linux-glibc2.5-x86_64
[root@mastera0 mysql-5.6.20-linux-glibc2.5-x86_64]# ls
bin COPYING data docs include INSTALL-BINARY lib man mysql-test README scripts share
sql-bench support-files
[root@mastera0 mysql-5.6.20-linux-glibc2.5-x86_64]# cat INSTALL-BINARY
```

... ..

To install and use a MySQL binary distribution, the basic command sequence looks like this:

```
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
shell> bin/mysqld_safe --user=mysql &
# Next command is optional
shell> cp support-files/mysql.server /etc/init.d/mysql.server
... ..
```

```
[root@mastera0 mysql-5.6.20-linux-glibc2.5-x86_64]# groupadd mysql
[root@mastera0 mysql-5.6.20-linux-glibc2.5-x86_64]# cd ..
[root@mastera0 ~]# useradd -r -g mysql mysql
[root@mastera0 ~]# cd /usr/local
[root@mastera0 local]# mv /root/mysql-5.6.20-linux-glibc2.5-x86_64 .
[root@mastera0 local]# ls
bin  games  lib  libexec  sbin  src
etc  include lib64 mysql-5.6.20-linux-glibc2.5-x86_64 share
[root@mastera0 local]# ln -s mysql-5.6.20-linux-glibc2.5-x86_64 mysql
[root@mastera0 local]# ll mysql
lrwxrwxrwx. 1 root root 34 Dec 11 12:20 mysql -> mysql-5.6.20-linux-glibc2.5-x86_64
[root@mastera0 mysql]# cd mysql
[root@mastera0 mysql]# mkdir /data/mysql/data -p
[root@mastera0 mysql]# chown mysql. /data/mysql/data
[root@mastera0 mysql]# chown mysql. /data/mysql/data -R
[root@mastera0 mysql]# ll -d /data/mysql/data
drwxr-xr-x. 2 mysql mysql 4096 Dec 11 12:24 /data/mysql/data
[root@mastera0 mysql]# scripts/mysql_install_db --user=mysql --datadir=/data/mysql/data --
basedir=/usr/local/mysql
```

```
[root@mastera0 mysql]# ll /data/mysql/data
total 110604
-rw-rw----. 1 mysql mysql 12582912 Dec 11 12:28 ibdata1
-rw-rw----. 1 mysql mysql 50331648 Dec 11 12:28 ib_logfile0
-rw-rw----. 1 mysql mysql 50331648 Dec 11 12:28 ib_logfile1
drwx-----. 2 mysql mysql 4096 Dec 11 12:28 mysql
drwx-----. 2 mysql mysql 4096 Dec 11 12:28 performance_schema
drwx-----. 2 mysql mysql 4096 Dec 11 12:28 test
```

```

[root@mastera0 mysql]# vim /etc/my.cnf
[client]
#如果不认识这个参数会忽略
loose-default-character-set=utf8
loose-prompt='\u@\h:\p [\d]>'
socket=/tmp/mysql.sock

[mysqld]
basedir = /usr/local/mysql
datadir = /data/mysql/data
user=mysql
port = 3306
socket=/tmp/mysql.sock
pid-file=/data/tmp/mysql.pid
tmpdir=/data1/tmp
character_set_server=utf8

#skip
skip-external_locking=1
skip-name-resolve=1

#AB replication
server-id = 1
log-bin = /data/mysql/log-data/mastera
binlog_format=row
max_binlog_cache_size=2000M
max_binlog_size=1G
sync_binlog=1
#expire_logs_days=7

#semi_sync
rpl_semi_sync_master_enabled=1
rpl_semi_sync_master_timeout=1000

[root@mastera0 mysql]# pwd
/usr/local/mysql
[root@mastera0 mysql]# cp support-files/mysql.server /etc/init.d/mysql
[root@mastera0 mysql]# echo export PATH=$PATH:/usr/local/mysql/support-files/ >> /etc/bashrc
[root@mastera0 mysql]# service mysql start

```

MySQL客户端连接数据库

MySQL客户端的使用

上一节安装MySQL服务端时已经讲解过客户端的使用，此处不再赘述。

python连接MySQL数据库

Python 标准数据库接口为 Python DB-API，Python DB-API为开发人员提供了数据库应用编程接口。

Python 数据库接口支持非常多的数据库，你可以选择适合你项目的数据库：

- GadFly

- mSQL
- MySQL
- PostgreSQL
- Microsoft SQL Server 2000
- Informix
- Interbase
- Oracle
- Sybase

你可以访问Python数据库接口及API查看详细的支持数据库列表。

不同的数据库你需要下载不同的DB API模块，例如你需要访问Oracle数据库和Mysql数据，你需要下载Oracle和MySQL数据库模块。

DB-API 是一个规范. 它定义了一系列必须的对象和数据存取方式, 以便为各种各样的底层数据库系统和多种多样的数据库接口程序提供一致的访问接口。

Python的DB-API，为大多数的数据库实现了接口，使用它连接各数据库后，就可以用相同的方式操作各数据库。

Python DB-API使用流程：

- 引入 API 模块。
- 获取与数据库的连接。
- 执行SQL语句和存储过程。
- 关闭数据库连接。

什么是MySQLdb?

MySQLdb 是用于Python链接Mysql数据库的接口，它实现了 Python 数据库 API 规范 V2.0，基于 MySQL C API 上建立的。

如何安装MySQLdb?

安装MySQLdb，请访问 <http://sourceforge.net/projects/mysql-python>，(Linux平台可以访问：<https://pypi.python.org/pypi/MySQL-python>)从这里可选择适合您的平台的安装包，分为预编译的二进制文件和源代码安装包。

如果您选择二进制文件发行版本的话，安装过程基本安装提示即可完成。如果从源代码进行安装的话，则需要切换到MySQLdb发行版本的顶级目录，并键入下列命令：

```
$ gunzip MySQL-python-1.2.2.tar.gz
$ tar -xvf MySQL-python-1.2.2.tar
$ cd MySQL-python-1.2.2
$ python setup.py build
$ python setup.py install
```

下载安装包

[MySQL-python](#)

```
[root@localhost ~]# ls
MySQL-python-1.2.5.zip
[root@localhost ~]# unzip MySQL-python-1.2.5.zip
[root@localhost ~]# cd MySQL-python-1.2.5/
[root@localhost MySQL-python-1.2.5]# ls
oc      MANIFEST.in  _mysql_exceptions.py  README.md      setup_posix.py  site.cfg
GPL-2.0  metadata.cfg  MySQL_python.egg-info  setup.cfg      setup_posix.pyc  tests
HISTORY  _mysql.c      PKG-INFO              setup_common.py  setup.py
INSTALL  MySQLdb       pymemcompat.h         setup_common.pyc  setup_windows.py
[root@localhost MySQL-python-1.2.5]# python setup.py build
sh: mysql_config: command not found
Traceback (most recent call last):
  File "setup.py", line 17, in <module>
    metadata, options = get_config()
  File "/root/MySQL-python-1.2.5/setup_posix.py", line 43, in get_config
    libs = mysql_config("libs_r")
  File "/root/MySQL-python-1.2.5/setup_posix.py", line 25, in mysql_config
    raise EnvironmentError("%s not found" % (mysql_config.path,))
EnvironmentError: mysql_config not found
```

缺少mysql_config命令

该命令是由mysql-devel或者mariadb-devel安装生成，因此先检查是否安装，在检查site.cfg中mysql_config的路径是否正确

```
[root@localhost MySQL-python-1.2.5]# which mysql_config
/usr/bin/which: no mysql_config in (/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/root/bin)

[root@localhost MySQL-python-1.2.5]# yum install -y mariadb-devel

[root@localhost MySQL-python-1.2.5]# which mysql_config
/usr/bin/mysql_config

[root@localhost MySQL-python-1.2.5]# vim site.cfg
mysql_config=/usr/bin/mysql_config
```

开始编译安装

```
[root@localhost MySQL-python-1.2.5]# python setup.py build
[root@localhost MySQL-python-1.2.5]# python setup.py install
```

加载模块

```
In [2]: import MySQLdb
```

```
In [3]: dir(MySQLdb)
```

```
Out[3]:
```

```
['BINARY',  
 'Binary',  
 'Connect',  
 'Connection',  
 'DATE',  
 'DATETIME',  
 'DBAPISet',  
 'DataError',  
 'DatabaseError',  
 'Date',  
 'DateFromTicks',  
 'Error',  
 'FIELD_TYPE',  
 'IntegrityError',  
 'InterfaceError',  
 'InternalError',  
 'MySQLError',  
 'NULL',  
 'NUMBER',  
 'NotSupportedError',  
 'OperationalError',  
 'ProgrammingError',  
 'ROWID',  
 'STRING',  
 'TIME',  
 'TIMESTAMP',  
 'Time',  
 'TimeFromTicks',  
 'Timestamp',  
 'TimestampFromTicks',  
 'Warning',  
 '__all__',  
 '__author__',  
 '__builtins__',  
 '__doc__',  
 '__file__',  
 '__loader__',  
 '__name__',  
 '__package__',  
 '__path__',  
 '__revision__',  
 '__version__',  
 '_mysql',  
 'apilevel',  
 'connect',  
 'connection',  
 'constants',  
 'debug',  
 'escape',
```

```
'escape_dict',
'escape_sequence',
'escape_string',
'get_client_info',
'paramstyle',
'release',
'result',
'server_end',
'server_init',
'string_literal',
'test_DBAPISet_set_equality',
'test_DBAPISet_set_equality_membership',
'test_DBAPISet_set_inequality',
'test_DBAPISet_set_inequality_membership',
'thread_safe',
'threadsafety',
'times',
'version_info'
```

数据库连接

连接数据库前，请先确认以下事项：

1. 数据库服务已经启动，并已创建了数据库 **test1**
2. 在**test1**数据库中您已经创建了表 **db1**
3. t1表字段为 id,first_name,last_name,age,sex,income
4. 连接数据库t1使用的用户名为 "root" ， 密码为 "uplooking"

```
[root@localhost ~]# systemctl start mariadb
[root@localhost ~]# mysqladmin -uroot password 'uplooking'
[root@localhost ~]# mysql -uroot -puplooking
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 4
Server version: 5.5.44-MariaDB-log MariaDB Server

Copyright (c) 2000, 2015, Oracle, MariaDB Corporation Ab and others.

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

MariaDB [(none)]> create database db1;
Query OK, 1 row affected (0.01 sec)

MariaDB [(none)]> create table db1.t1 (id int auto_increment primary key,first_name
varchar(50),last_name varchar(50),age int,sex char(1),income float);
Query OK, 0 rows affected (0.09 sec)
```

实例1：连接数据库获取数据库版本信息

```
[root@localhost python-mysql]# vim getversion.py
#!/usr/bin/python
# -*- coding: UTF-8 -*-

import MySQLdb

# 打开数据库连接
db = MySQLdb.connect("localhost","root","uplooking","db1" )

# 使用cursor()方法获取操作游标
cursor = db.cursor()

# 使用execute方法执行SQL语句
cursor.execute("SELECT VERSION()")

# 使用 fetchone() 方法获取一条数据库。
data = cursor.fetchone()

print "Database version : %s " % data

# 关闭数据库连接
db.close()

# 执行结果
[root@localhost python-mysql]# python getversion.py
Database version : 5.5.44-MariaDB-log
```

实例2：查询mysql中的db1.t1表的结构（数据字典）信息

```
#!/usr/bin/python
# -*- coding: UTF-8 -*-

import MySQLdb
import sys

sql=sys.argv[1]

# 打开数据库连接
db = MySQLdb.connect("localhost","root","uplooking","db1" )

# 使用cursor()方法获取操作游标
cursor = db.cursor()

# 使用execute方法执行SQL语句
cursor.execute(sql)

# 使用 fetchone() 方法获取一条数据库。
data = cursor.fetchall()

for i in data:
    for j in i:
        sys.stdout.write(str(j)+'\t')
    sys.stdout.write('\n')

# 关闭数据库连接
db.close()
```

脚本带参数执行:

```
[root@localhost python-mysql]# python select.py "desc db1.t1"
id  int(11) NO  PRI None      auto_increment
first_name  varchar(50) YES      None
last_name   varchar(50) YES      None
age  int(11) YES      None
sex  char(1) YES      None
income float   YES      None
```

数据库查询操作

- Python查询Mysql使用 `fetchone()` 方法获取单条数据, 使用`fetchall()` 方法获取多条数据。
- `fetchone()`: 该方法获取下一个查询结果集。结果集是一个对象
- `fetchall()`:接收全部的返回结果行。
- `rowcount`: 这是一个只读属性, 并返回执行`execute()`方法后影响的行数。

实例3：获取mysql数据库的变量值


```
#!/usr/bin/python
# -*- coding: UTF-8 -*-

import MySQLdb
import sys

key=sys.argv[1]
sql="select @@"+key

# 打开数据库连接
db = MySQLdb.connect("localhost","root","uplooking","db1" )

# 使用cursor()方法获取操作游标
cursor = db.cursor()

# 使用execute方法执行SQL语句
cursor.execute(sql)

# 使用 fetchone() 方法获取一条数据库。
data = cursor.fetchall()

print "{} : {}".format(sys.argv[1],data[0])

# 关闭数据库连接
db.close()
```

获取innodb_version\innodb_buffer_pool_size

```
[root@localhost python-mysql]# python Get_variables.py innodb_version
innodb_version : 5.5.43-MariaDB-37.2
[root@localhost python-mysql]# vim Get_variables.py
[root@localhost python-mysql]# python Get_variables.py innodb_buffer_pool_size
innodb_buffer_pool_size : 134217728
```

实例4：创建数据库表

如果数据库连接存在我们可以使用execute()方法来为数据库创建表，如下所示创建表t2:

```
#!/usr/bin/python
# -*- coding: UTF-8 -*-

import MySQLdb

# 打开数据库连接
db = MySQLdb.connect("localhost","root","uplooking","db1" )

# 使用cursor()方法获取操作游标
cursor = db.cursor()

# 如果数据表已经存在使用 execute() 方法删除表。
cursor.execute("DROP TABLE IF EXISTS t2")

# 创建数据表SQL语句
sql = """CREATE TABLE t2 (
        FIRST_NAME  CHAR(20) NOT NULL,
        LAST_NAME   CHAR(20),
        AGE INT,
        SEX CHAR(1),
        INCOME FLOAT )"""

cursor.execute(sql)

# 关闭数据库连接
db.close()
```

执行该脚本

```
[root@localhost python-mysql]# python createdb.py
createdb.py:13: Warning: Unknown table 't2'
  cursor.execute("DROP TABLE IF EXISTS t2")

[root@localhost python-mysql]# ls
createdb.py  Get_variables.py  getversion.py  select.py

#查看表的结构

[root@localhost python-mysql]# python select.py "desc db1.t2"
FIRST_NAME  char(20)      NO      None
LAST_NAME   char(20)      YES     None
AGE int(11)  YES      None
SEX char(1)  YES      None
INCOME float  YES      None
```

实例5：数据库插入操作

以下实例使用执行 SQL INSERT 语句向表 db1.t1 插入记录：

```
#!/usr/bin/python
# -*- coding: UTF-8 -*-

import MySQLdb

# 打开数据库连接
db = MySQLdb.connect("localhost","root","uplooking","db1" )

# 使用cursor()方法获取操作游标
cursor = db.cursor()

# SQL 插入语句
sql = """INSERT INTO t1(FIRST_NAME,
                        LAST_NAME, AGE, SEX, INCOME)
                        VALUES ('Mac', 'Mohan', 20, 'M', 2000)"""
try:
    # 执行sql语句
    cursor.execute(sql)
    # 提交到数据库执行
    db.commit()
except:
    # Rollback in case there is any error
    db.rollback()

# 关闭数据库连接
db.close()
```

执行该脚本，并查看db1.t1表中的所有内容

```
[root@localhost python-mysql]# python insert_table.py
[root@localhost python-mysql]# python select.py "select * from db1.t1"
1   Mac  Mohan   20   M   2000.0
```

实例6：删除操作

删除操作用于删除数据表中的数据，以下实例演示了删除数据表 db1 中 AGE 大于 20 的所有数据：

```
#!/usr/bin/python
# -*- coding: UTF-8 -*-

import MySQLdb

# 打开数据库连接
db = MySQLdb.connect("localhost","root","uplooking","db1" )

# 使用cursor()方法获取操作游标
cursor = db.cursor()

# SQL 删除语句
sql = "DELETE FROM db1.t1 WHERE age > 20"
try:
    # 执行SQL语句
    cursor.execute(sql)
    # 提交修改
    db.commit()
except:
    # 发生错误时回滚
    db.rollback()

# 关闭连接
db.close()
```

实例7：执行事务

事务机制可以确保数据一致性。

事务应该具有4个属性：原子性、一致性、隔离性、持久性。这四个属性通常称为ACID特性。

- 原子性（atomicity）。一个事务是一个不可分割的工作单位，事务中包括的诸操作要么都做，要么都不做。
- 一致性（consistency）。事务必须是使数据库从一个一致性状态变到另一个一致性状态。一致性与原子性是密切相关的。
- 隔离性（isolation）。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并发的其他事务是隔离的，并发执行的各个事务之间不能互相干扰。
- 持久性（durability）。持续性也称永久性（permanence），指一个事务一旦提交，它对数据库中数据的改变就应该是永久性的。接下来的其他操作或故障不应该对其有任何影响。

Python DB API 2.0 的事务提供了两个方法 commit 或 rollback。

实例：

```
# SQL删除记录语句
sql = "DELETE FROM db1.t1 WHERE age > 20"
try:
    # 执行SQL语句
    cursor.execute(sql)
    # 向数据库提交
    db.commit()
except:
    # 发生错误时回滚
    db.rollback()
```

对于支持事务的数据库，在Python数据库编程中，当游标建立之时，就自动开始了一个隐形的数据库事务。

commit()方法游标的所有更新操作，rollback（）方法回滚当前游标的所有操作。每一个方法都开始了一个新的事务。

错误处理

DB API中定义了一些数据库操作的错误及异常，下表列出了这些错误和异常：

异常	描述
Warning	当有严重警告时触发，例如插入数据是被截断等等。必须是 StandardError 的子类。
Error	警告以外所有其他错误类。必须是 StandardError 的子类。
InterfaceError	当有数据库接口模块本身的错误（而不是数据库的错误）发生时触发。 必须是 Error 的子类。
DatabaseError	和数据库有关的错误发生时触发。 必须是 Error 的子类。
DataError	当有数据处理时的错误发生时触发，例如：除零错误，数据超范围等等。 必须是 DatabaseError 的子类。
OperationalError	指非用户控制的，而是操作数据库时发生的错误。例如：连接意外断开、数据库名未找到、事务处理失败、内存分配错误等等操作数据库是发生的错误。 必须是 DatabaseError 的子类。
IntegrityError	完整性相关的错误，例如外键检查失败等。必须是 DatabaseError 子类。
InternalError	数据库的内部错误，例如游标（ cursor ）失效了、事务同步失败等等。 必须是 DatabaseError 子类。
ProgrammingError	程序错误，例如数据表（ table ）没找到或已存在、SQL语句语法错误、 参数数量错误等等。必须是 DatabaseError 的子类。
NotSupportedError	不支持错误，指使用了数据库不支持的函数或API等。例如在连接对象上 使用.rollback()函数，然而数据库并不支持事务或者事务已关闭。 必须是 DatabaseError 的子类。

总结

- 安装模块 `mysql-python`
- 加载模块 `import MySQLdb`
- 打开数据库连接 `db = MySQLdb.connect(host,user,password,database))`
- 使用`cursor()`方法获取操作游标事务开始 `cursor = db.cursor()`
- 执行SQL语句 `cursor.execute(sql)`
- 向数据库提交 `db.commit()`
- 发生错误时回滚 `db.rollback()`
- 关闭连接 `db.close()`

PHPmyAdmin在线工具使用

PHPMYADMIN 是一个使用 PHP 语言编写的,使用 web 管理 MYSQL 的组件。严格意义上说,它也是一种 MYSQL 的客户端。最近一段时间,出现的很多依靠网站连接 MYSQL 进行管理的产品,在这些“WEB GUI”中,PHPMYADMIN 是使用范围最为广泛的,同时也受到很多 MYSQL 数据库管理员的好评。通过它,你可以非常轻松,非常方便的管理 MYSQL 数据库

- 获取 PHPMYADMIN

你可以到 PHPMYADMIN 的官方网站 <http://www.phpmyadmin.net> 下载最新的版本。

- 安装 PHPMYADMIN

```

1.yum install -y httpd php php-mysql mariadb-server php-mbstring
ftp://rpmfind.net/linux/centos/7.2.1511/os/x86_64/Packages/php-mbstring-5.4.16-
36.el7_1.x86_64.rpm

2.systemctl start httpd
3.systemctl start mariadb
4.mysqladmin -uroot password uplooking
5.mysql -uroot -puplooking
>create database phpmyadmin
>grant all on phpmyadmin.* to php@localhost identified by 'uplooking';
>flush privileges;

6.echo hi > /var/www/html/index.html
测试一下web服务是否成功

7.tar jxf /mnt/courses/db100/rhel7.2/materials/phpMyAdmin-4.4.15.5-all-languages.tar.bz2 -C
/var/www/html
8.chmod -R 755 html
9.mv config.sample.inc.php config.sample.inc;vim config.inc.php
$config['blowfish_secret'] = ''; /* YOU MUST FILL IN THIS FOR COOKIE AUTH! */

/*
 * Servers configuration
 */
$i = 0;

/*
 * First server
 */
$i++;
/* Authentication type */
$config['Servers'][$i]['user'] = 'root';
$config['Servers'][$i]['password'] = 'uplooking';
$config['Servers'][$i]['auth_type'] = 'config';

10.vim /etc/httpd/conf/httpd.conf
<IfModule dir_module>
    DirectoryIndex index.html index.php
</IfModule>

11.systemctl restart httpd
从浏览器输入172.25.0.10

```

默认安装phpMyAdmin，通常只能连一台MySQL服务器，其配置信息是保存在phpMyAdmin的配置文件里的，当我们需要在多台服务器之间进行切换登陆的时候，修改起来非常麻烦。遵照下面的配置方法，我们可以方便的使用phpMyAdmin连接多台MySQL。

登陆phpMyAdmin时只需输入用户名、密码，服务器地址为下拉列表可选，登陆后也可选择其他服务器快速切换。（推荐）

优点：登陆操作简便，登陆后切换服务器无须退出当前连接。

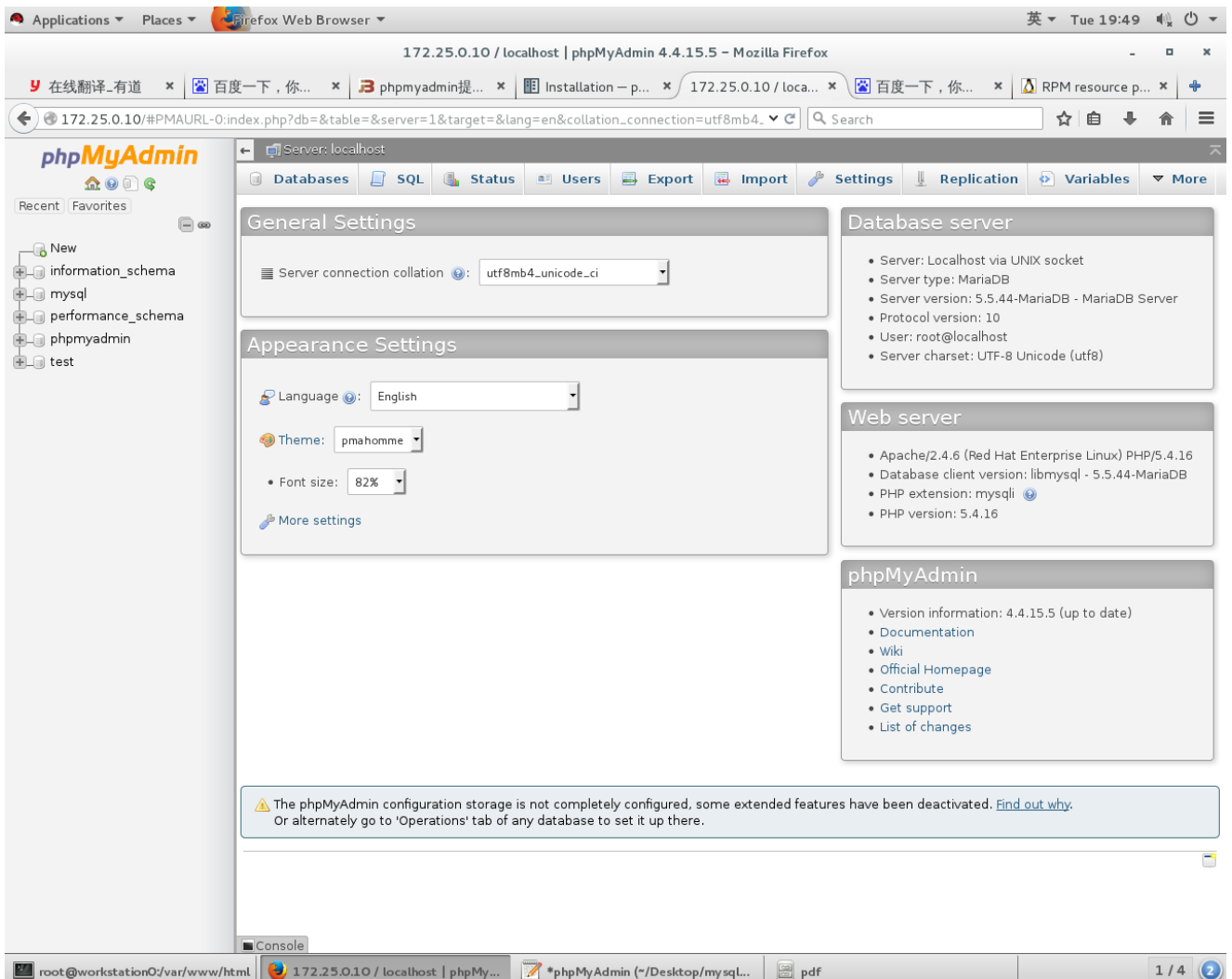
```

$hosts = array(
    '1'=>array('host'=>'localhost','user'=>'root','password'=>'uplooking'),
    '2'=>array('host'=>'172.25.0.11','user'=>'root','password'=>'uplooking')
);

for ($i=1;$i<=count($hosts);$i++){

/* Authentication type */
$config['Servers'][$i]['user']           = $hosts[$i]['user'];
$config['Servers'][$i]['password']       = $hosts[$i]['password'];
$config['Servers'][$i]['auth_type']      = 'config';
/* Server parameters */
$config['Servers'][$i]['host']           = $hosts[$i]['host'];
$config['Servers'][$i]['connect_type']   = 'tcp';
$config['Servers'][$i]['compress']       = false;
$config['Servers'][$i]['AllowNoPassword'] = false;
}

```



MySQL Workbench 连接数据库

MySQL Workbench 是一款专门为用户提供了用于创建、修改、执行和优化SQL的可视化工具，开发人员可以很轻松的管理数据库。该工具并且提供开发者一整套可视化用于创建、编辑和管理SQL 查询和管理数据库连接。在可视化SQL编辑工作模式下，用户创建表，删除表，修改表信息等只需要使用简单的可编辑列表中完成。

用户通常认为MySQL Workbench 是一个MySQL 数据库ER模型设计的工具，可以说是专门为MySQL数据库提供的数据库设计工具，用户使用MySQL Workbench可以很容易的设计、编辑数据库ER模型。这一功能可以说是MySQL Workbench的一大亮点。

软件获取

<https://dev.mysql.com/downloads/workbench/>

软件安装

```
[root@workstation software]# pwd
/software
[root@workstation software]# ls mysql-workbench-community-6.3.8-1.el7.x86_64.rpm mysql-workbench-
community-6.3.8-1.el7.x86_64.rpm
[root@workstation software]# rpm -ivh mysql-workbench-community-6.3.8-1.el7.x86_64.rpm
warning: mysql-workbench-community-6.3.8-1.el7.x86_64.rpm: Header V3 DSA/SHA1 Signature, key ID
5072e1f5: NOKEY
error: Failed dependencies:
    tinysql is needed by mysql-workbench-community-6.3.8-1.el7.x86_64
    libzip is needed by mysql-workbench-community-6.3.8-1.el7.x86_64
    python-paramiko >= 1.15.1 is needed by mysql-workbench-community-6.3.8-1.el7.x86_64
    proj is needed by mysql-workbench-community-6.3.8-1.el7.x86_64
    libodbc.so.2()(64bit) is needed by mysql-workbench-community-6.3.8-1.el7.x86_64
    libodbcinst.so.2()(64bit) is needed by mysql-workbench-community-6.3.8-1.el7.x86_64
```

需要一些依赖关系包，其中unixODBC和libzip本地yum中有，而tinysql\python-paramiko\proj需要下载

```
[root@workstation workbench]# ls
libtomcrypt-1.17-23.el7.x86_64.rpm      mysql-workbench-community-6.3.8-1.el7.x86_64.rpm
libtomcrypt-devel-1.17-23.el7.x86_64.rpm  proj-4.8.0-4.el7.x86_64.rpm
libtomcrypt-doc-1.17-23.el7.noarch.rpm    python2-crypto-2.6.1-13.el7.x86_64.rpm
libtommath-0.42.0-4.el7.x86_64.rpm        python2-ecdsa-0.13-4.el7.noarch.rpm
libtommath-devel-0.42.0-4.el7.x86_64.rpm  python2-paramiko-1.16.1-1.el7.noarch.rpm
libtommath-doc-0.42.0-4.el7.noarch.rpm    tinyxml-2.6.2-3.el7.x86_64.rpm
[root@workstation workbench]# rpm -ivh libtommath* libtomcrypt* python2-crypto* python2-ecdsa*
python2-paramiko* tinyxml* proj*
Preparing...                               ##### [100%]
package libtommath-0.42.0-4.el7.x86_64 is already installed
package libtomcrypt-1.17-23.el7.x86_64 is already installed
package python2-crypto-2.6.1-13.el7.x86_64 is already installed
package python2-ecdsa-0.13-4.el7.noarch is already installed
package python2-paramiko-1.16.1-1.el7.noarch is already installed
package libtomcrypt-devel-1.17-23.el7.x86_64 is already installed
package libtommath-devel-0.42.0-4.el7.x86_64 is already installed
package proj-4.8.0-4.el7.x86_64 is already installed
package tinyxml-2.6.2-3.el7.x86_64 is already installed
package libtomcrypt-doc-1.17-23.el7.noarch is already installed
package libtommath-doc-0.42.0-4.el7.noarch is already installed
[root@workstation workbench]# yum install -y unixODBC libzip
Loaded plugins: langpacks, product-id, search-disabled-repos, subscription-manager
This system is not registered to Red Hat Subscription Management. You can use subscription-
manager to register.
Package unixODBC-2.3.1-11.el7.x86_64 already installed and latest version
Package libzip-0.10.1-8.el7.x86_64 already installed and latest version
Nothing to do
[root@workstation workbench]# rpm -ivh mysql-workbench-community-6.3.8-1.el7.x86_64.rpm
warning: mysql-workbench-community-6.3.8-1.el7.x86_64.rpm: Header V3 DSA/SHA1 Signature, key ID
5072e1f5: NOKEY
Preparing...                               ##### [100%]
Updating / installing...
 1:mysql-workbench-community-6.3.8-1##### [100%]
[root@workstation workbench]#
[root@workstation workbench]# rpm -ql mysql-workbench-community |head
/usr/bin/mysql-workbench
/usr/bin/wbcopytables
/usr/lib64/mysql-workbench
/usr/lib64/mysql-workbench/libantlr3c_wb.so
/usr/lib64/mysql-workbench/libcdbc.so
/usr/lib64/mysql-workbench/libcdbc.so.6.3.8
/usr/lib64/mysql-workbench/libctemplate.so
/usr/lib64/mysql-workbench/libctemplate.so.3
/usr/lib64/mysql-workbench/libctemplate.so.3.0.0
/usr/lib64/mysql-workbench/libgdal.so.1
```

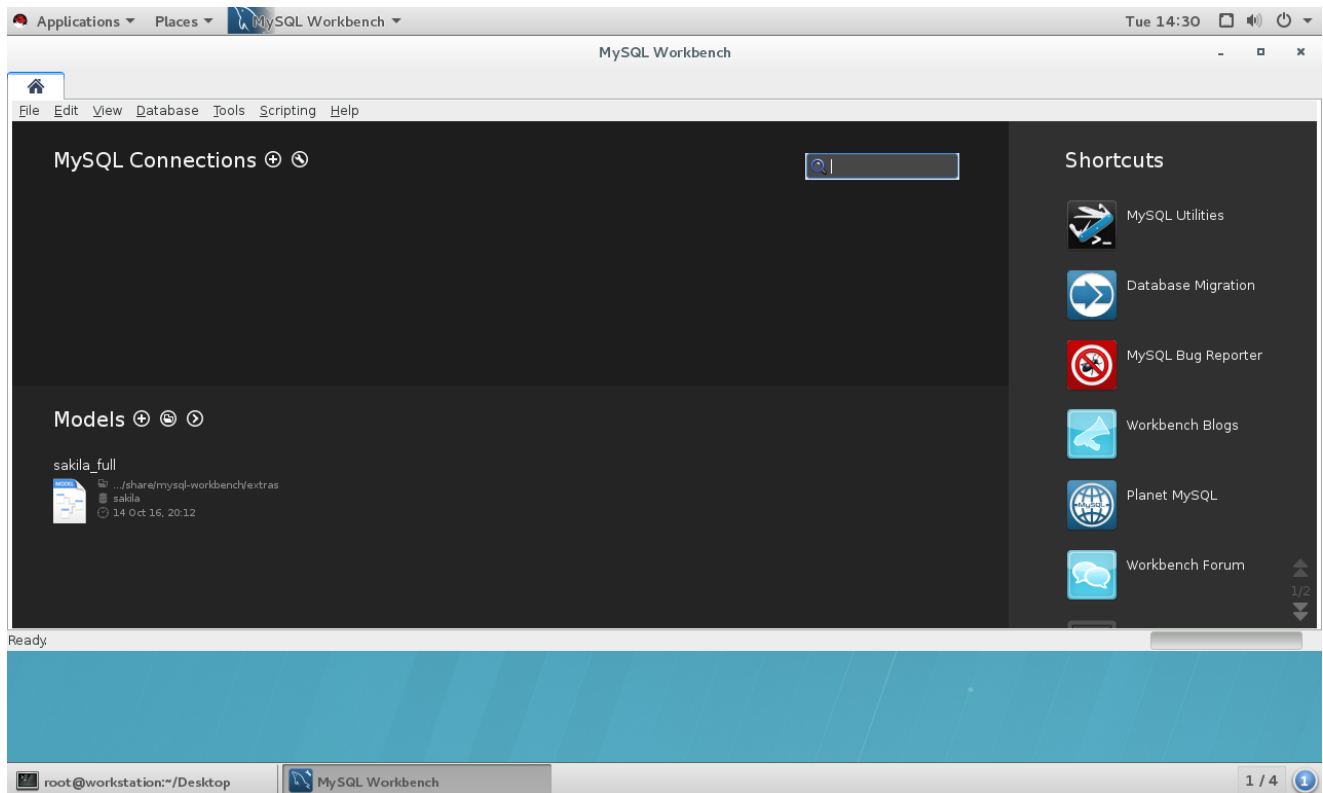
SQL Development的基本操作

- 创建数据库连接
- 创建新的数据库

- 创建和删除新的数据表
- 添加、修改表记录
- 查询表记录
- 修改表结构

启动MySQL Workbench

```
[root@workstation ~]# mysql-workbench
```



MySQL Workbench工作空间下对数据库数据进行管理之前，需要先创建数据库连接

建立连接前，需要在服务器上给MySQL Workbench授权

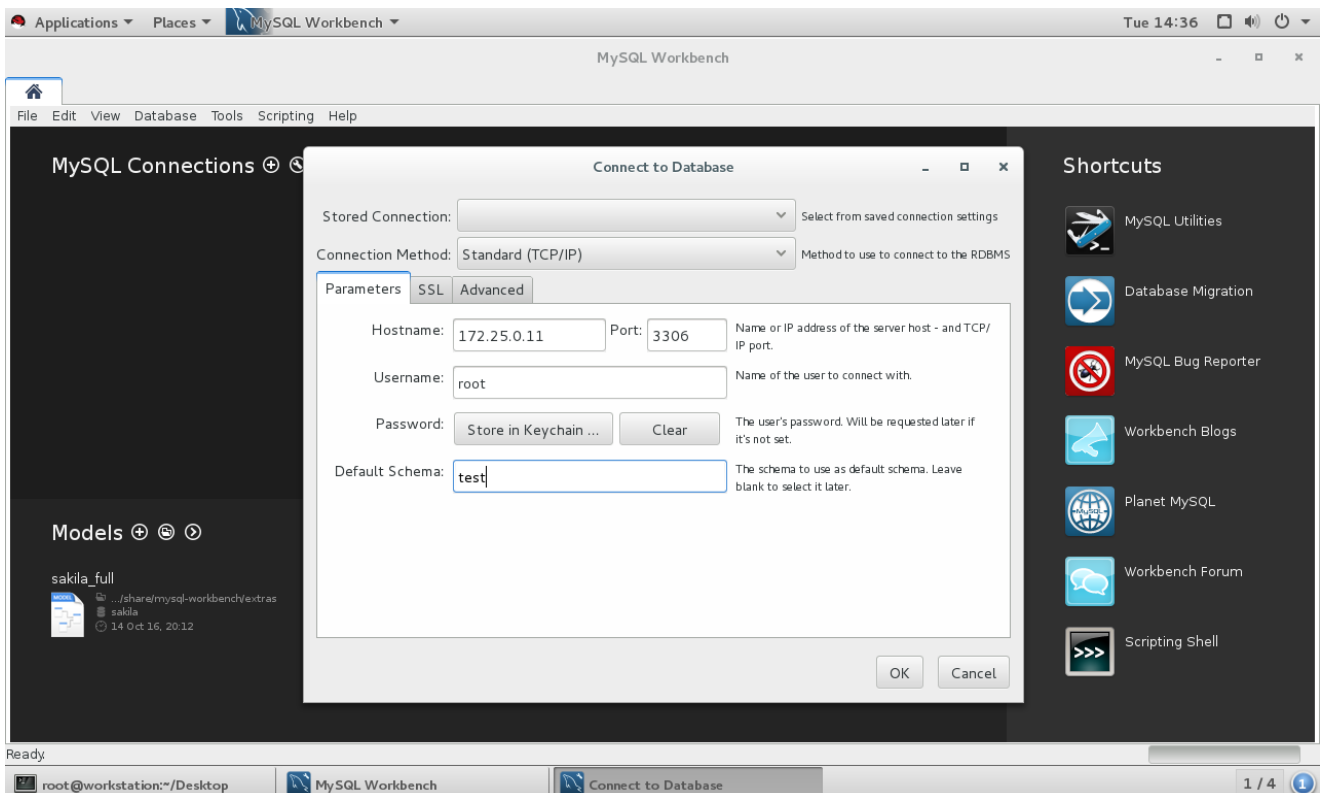
```
[root@mastera ~]# systemctl start mariadb
[root@mastera ~]# systemctl stop firewalld
[root@mastera ~]# mysql
ERROR 1045 (28000): Access denied for user 'root'@'localhost' (using password: NO)
[root@mastera ~]# mysql -uroot -puplooking
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 3
Server version: 5.5.44-MariaDB-log MariaDB Server

Copyright (c) 2000, 2015, Oracle, MariaDB Corporation Ab and others.

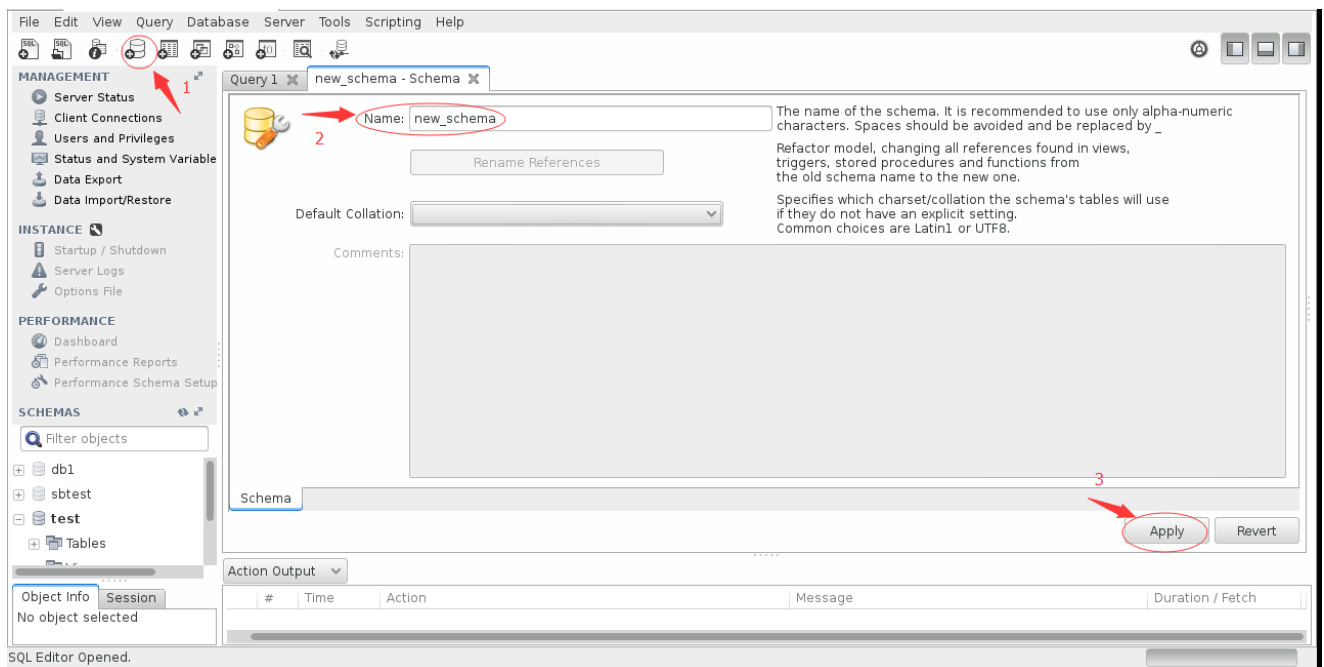
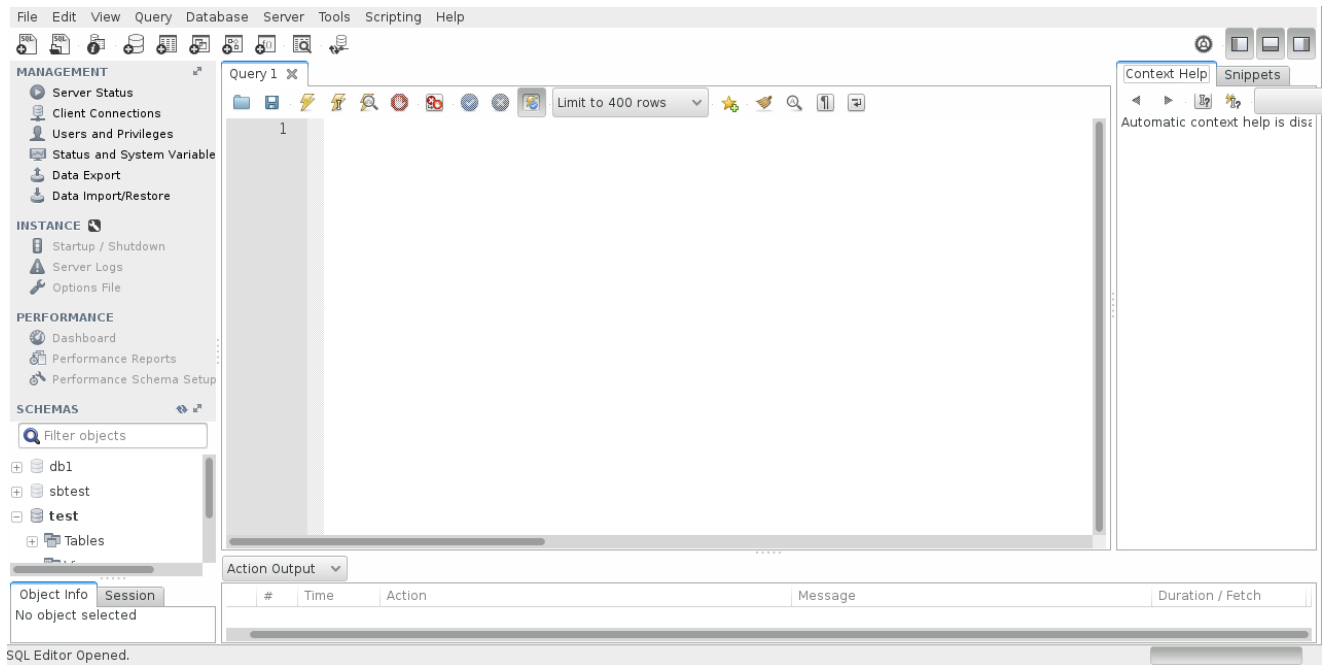
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> grant all on *.* to root@'172.25.0.10' identified by 'uplooking';
Query OK, 0 rows affected (0.01 sec)

MariaDB [(none)]> flush privileges;
Query OK, 0 rows affected (0.00 sec)
```



成功创建数据库连接后，可以创建新的数据库。



Apply SQL Script to Database

☒ Review SQL Script

☐ Apply SQL Script

Review the SQL Script to be Applied on the Database


Please review the following SQL script that will be applied to the database.

Note that once applied, these statements may not be revertible without losing some of the data.

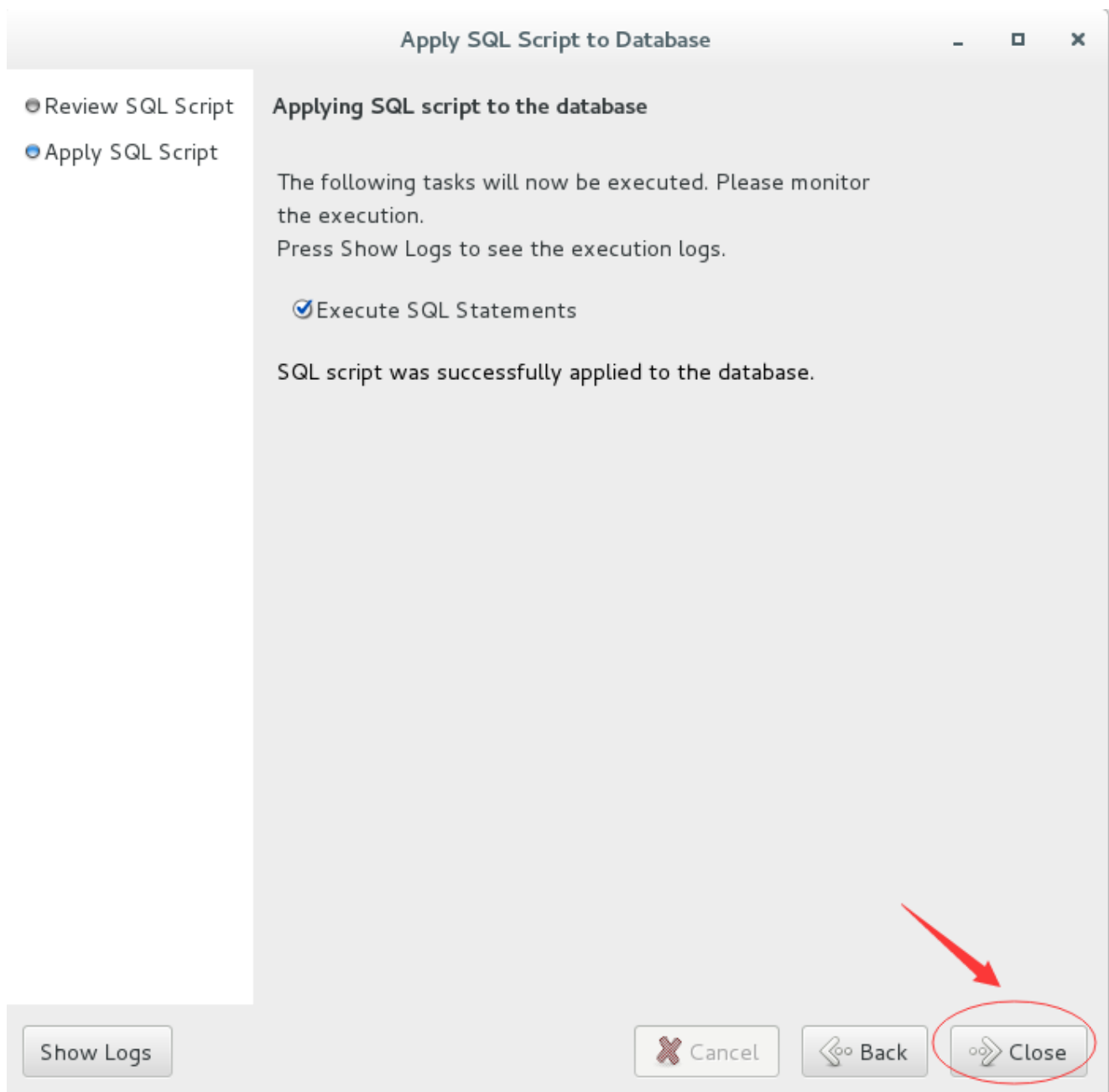
You can also manually change the SQL statements before execution.

```
1 CREATE SCHEMA `new_schema` ;  
2
```

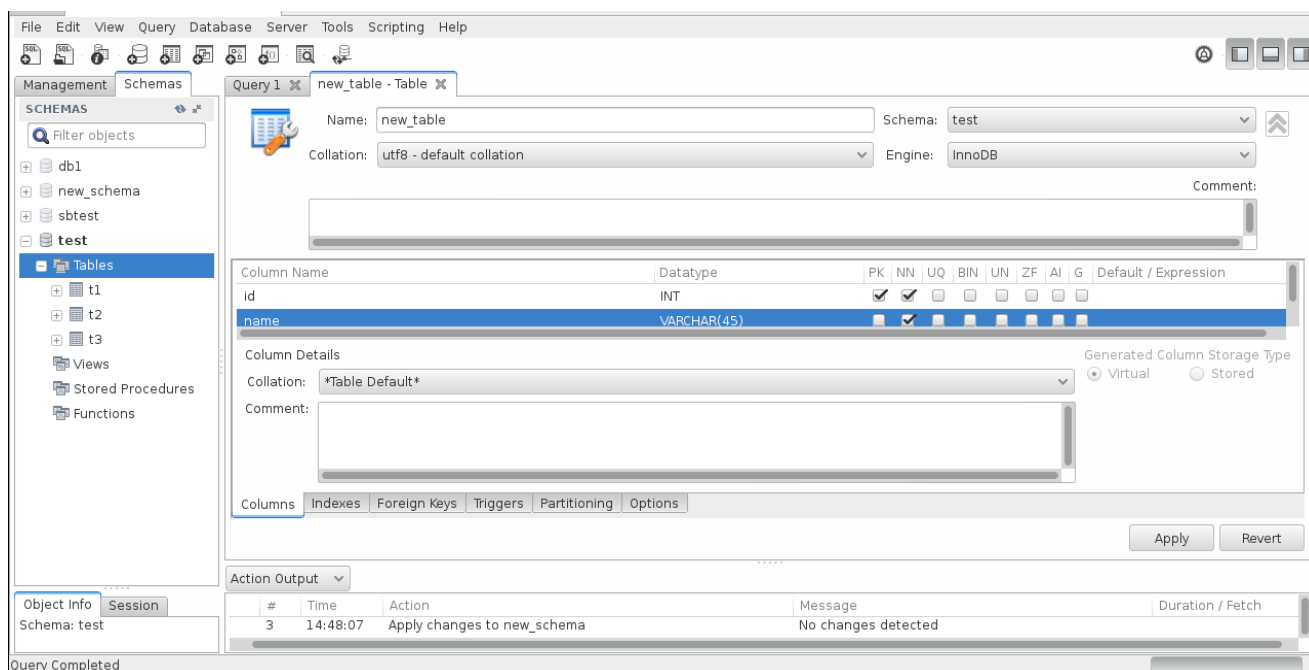
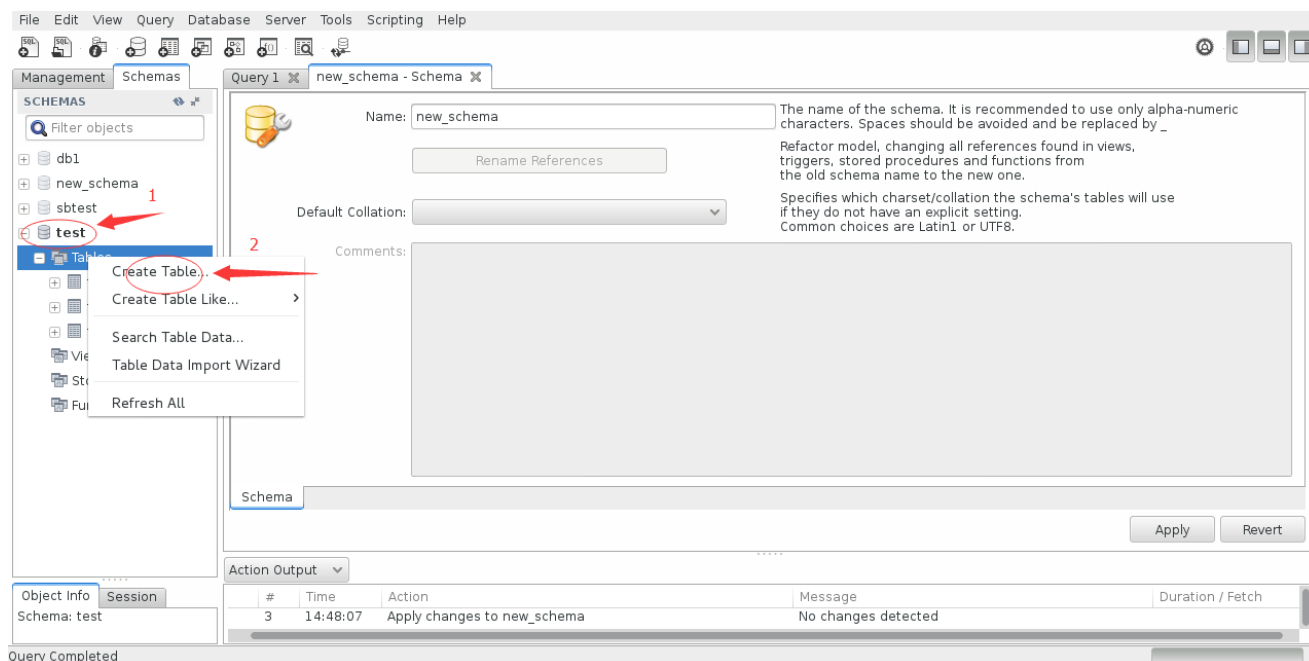
 Cancel

 Back

 Apply



成功创建数据库连接后，在左侧的SCHEMAS下面可以看到test数据库。用户可以创建新的数据库，本小节主要创建和删除新的数据表操作。



Apply SQL Script to Database

☒ Review SQL Script

☐ Apply SQL Script

Review the SQL Script to be Applied on the Database


Please review the following SQL script that will be applied to the database.

Note that once applied, these statements may not be revertible without losing some of the data.

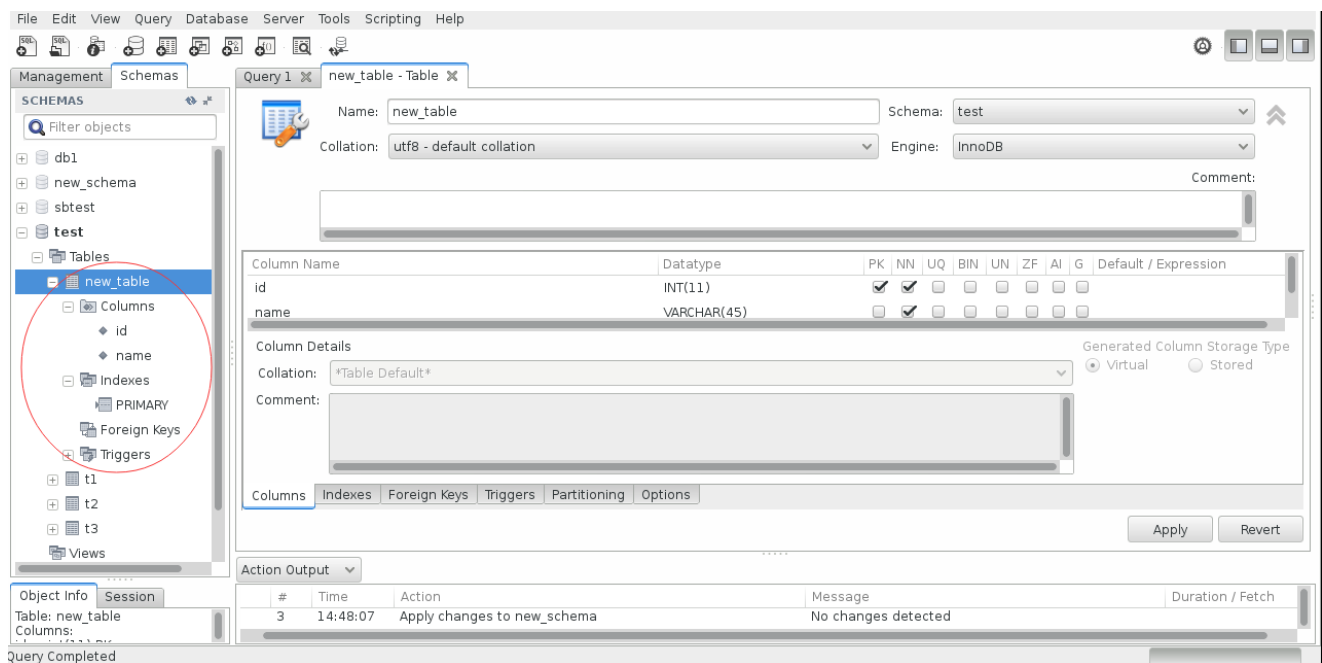
You can also manually change the SQL statements before execution.

```
1 CREATE TABLE `test`.`new_table` (  
2   `id` INT NOT NULL,  
3   `name` VARCHAR(45) NOT NULL,  
4   PRIMARY KEY (`id`))  
5 ENGINE = InnoDB  
6 DEFAULT CHARACTER SET = utf8;  
7
```

 Cancel

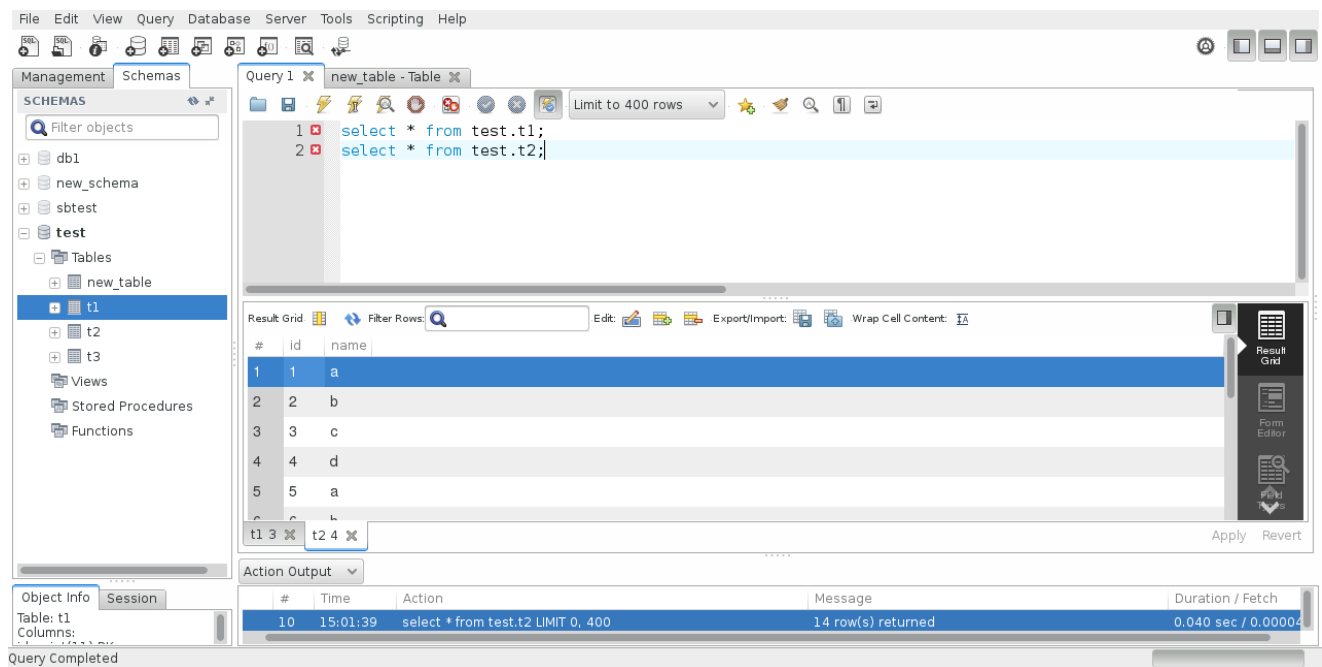
 Back

 Apply



下面简单介绍MySQL Workbench在图形界面下对数据库表的维护操作。

查询t1表中的数据操作。



使用WorkBench操作可以修改表的结构。

File Edit View Query Database Server Tools Scripting Help

Management Schemas

SCHEMAS

Filter objects

- db1
- new_schema
- sbtest
- test
 - Tables
 - new_table 1
 - t1
 - t2
 - t3
 - Views
 - Stored Procedures
 - Functions

Query 1 X new_table - Table X t2 - Table X

Name: t2 Schema: test

Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	G	Default / Expression
id	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
name_booboo	VARCHAR(10)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
t2col	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Column Details

Collation: *Table Default*

Comment:

Generated Column Storage Type: ☒ Virtual ☐ Stored

Columns Indexes Foreign Keys Triggers Partitioning Options

Apply Revert

Action Output

#	Time	Action	Message	Duration / Fetch
10	15:01:39	select * from test.t2 LIMIT 0, 400	14 row(s) returned	0.040 sec / 0.00004

Object Info Session

Table: t2

Columns:

Query Completed

Apply SQL Script to Database

☒ Review SQL Script

☐ Apply SQL Script

Review the SQL Script to be Applied on the Database


Please review the following SQL script that will be applied to the database.

Note that once applied, these statements may not be revertible without losing some of the data.

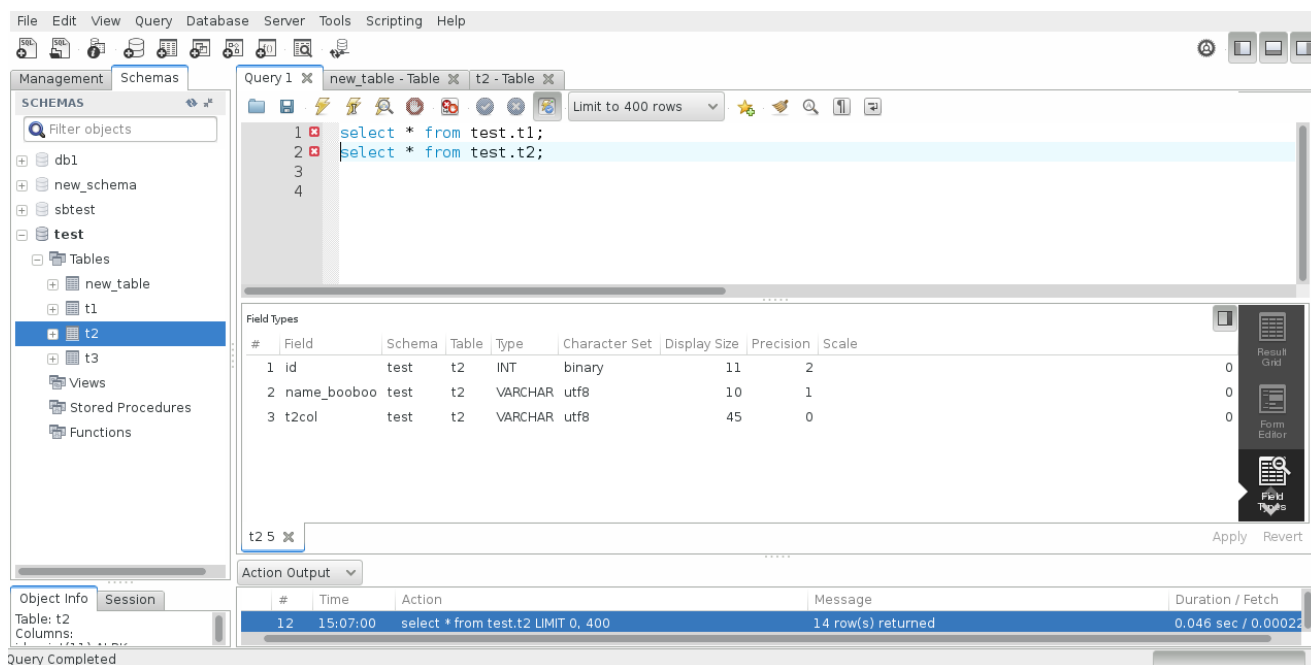
You can also manually change the SQL statements before execution.

```
1 ALTER TABLE `test`.`t2`  
2 CHANGE COLUMN `name` `name_booboo` VARCHAR(45) NULL AFTER  
3 ADD COLUMN `t2col` VARCHAR(45) NULL AFTER  
4
```

 Cancel

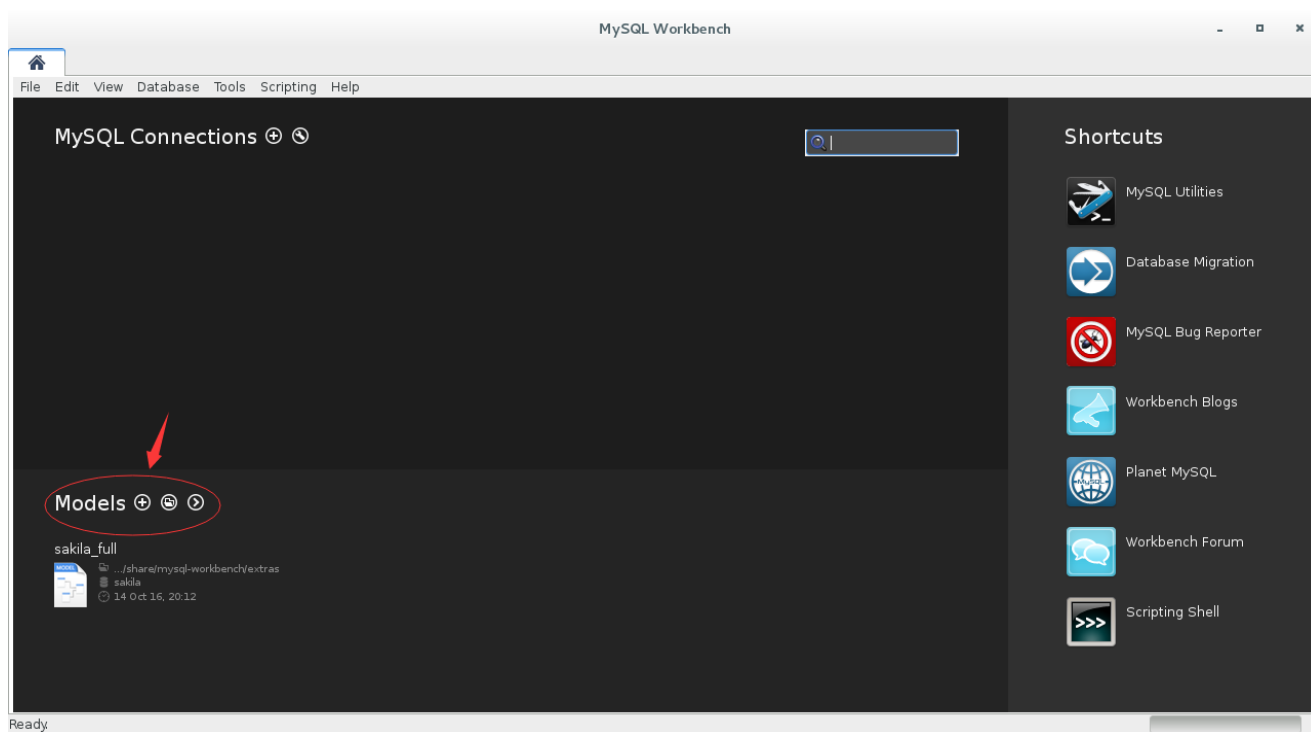
 Back

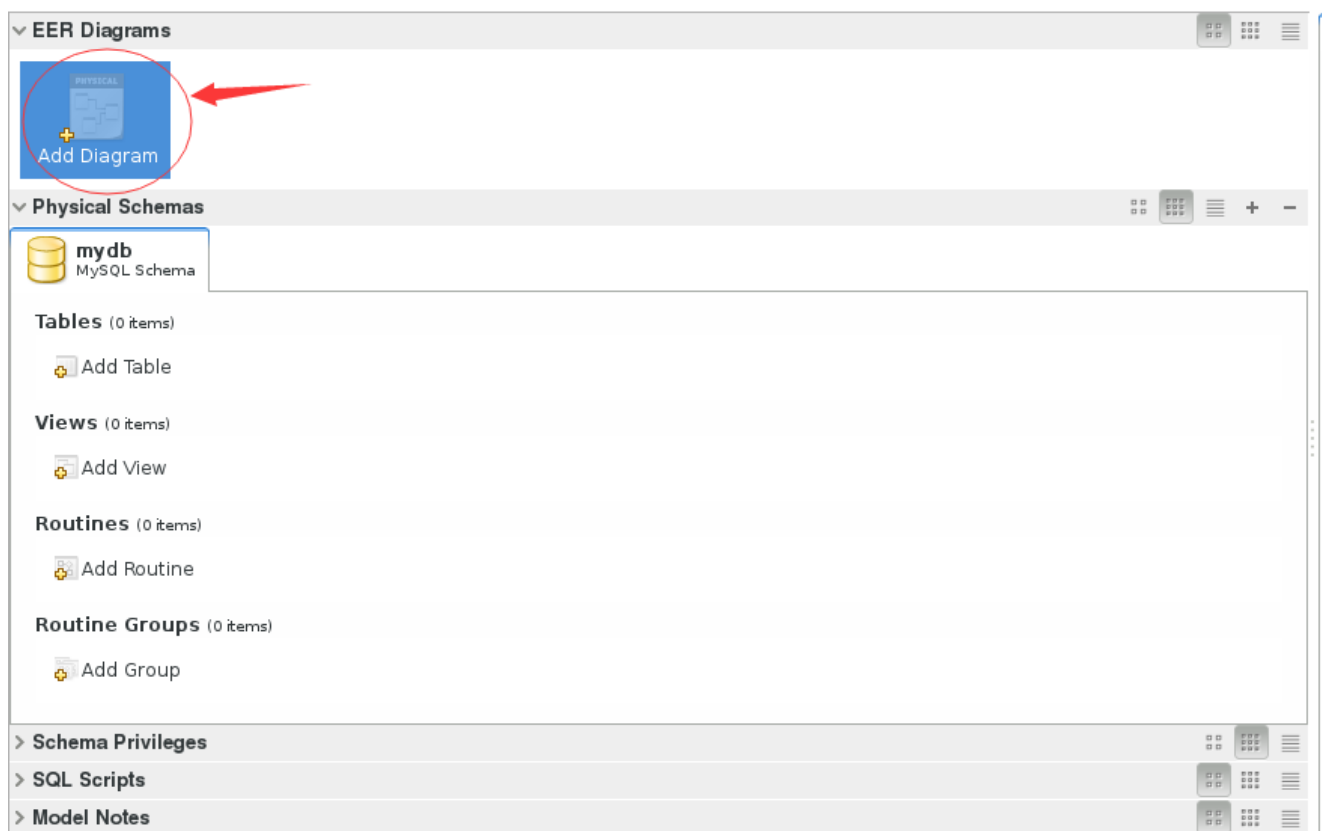
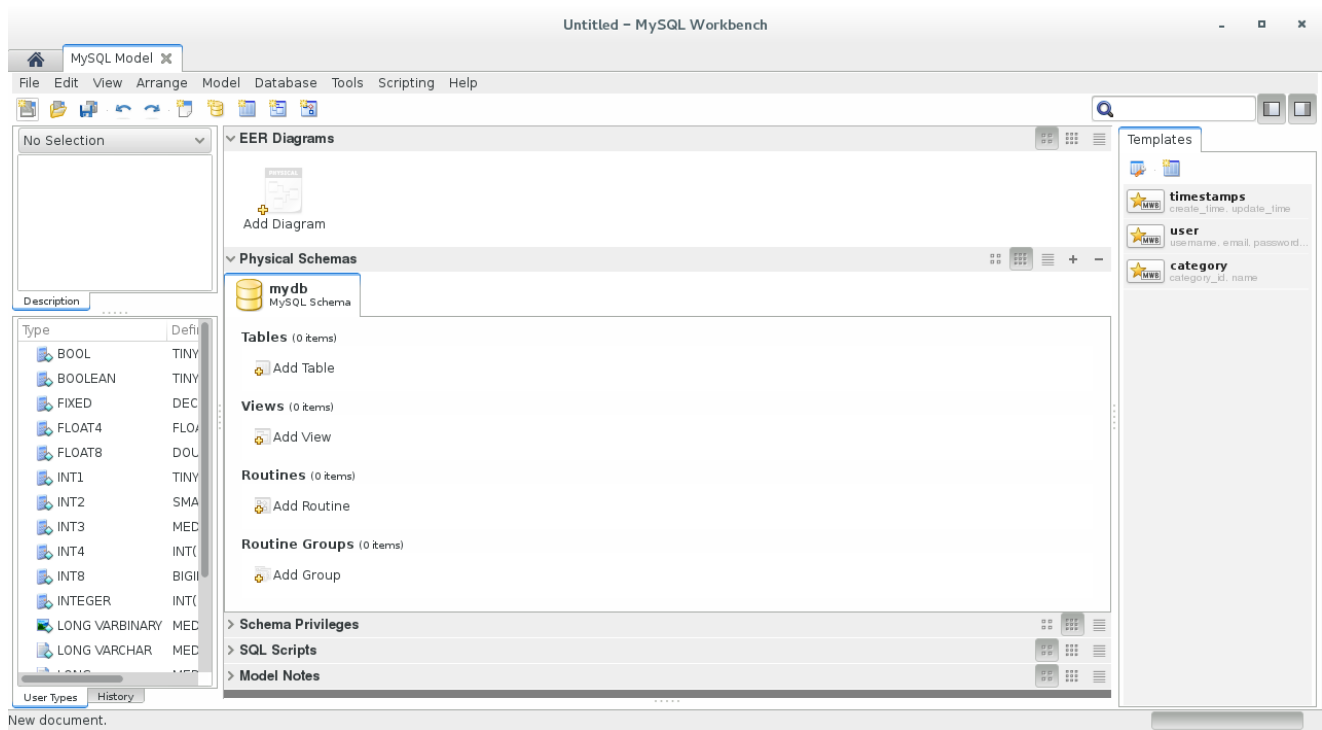
 Apply

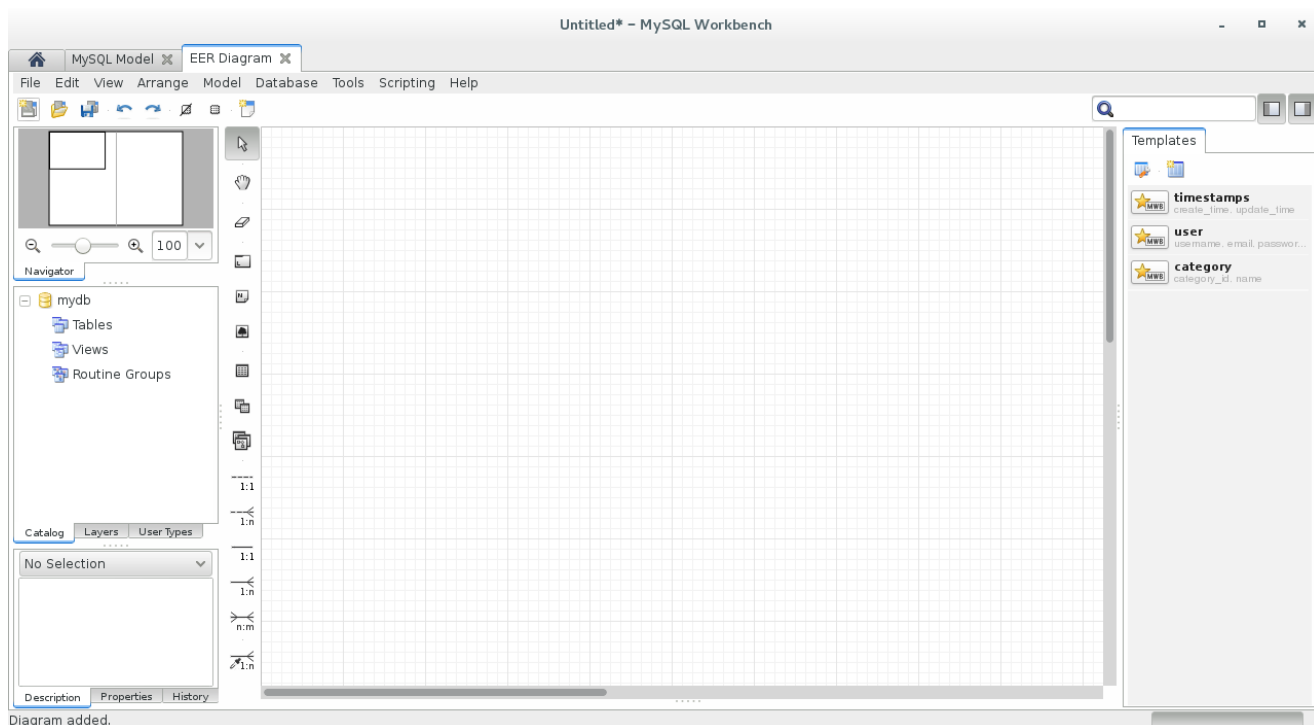


Data Modeling的基本操作

- 建立ER模型
- 导入ER模型





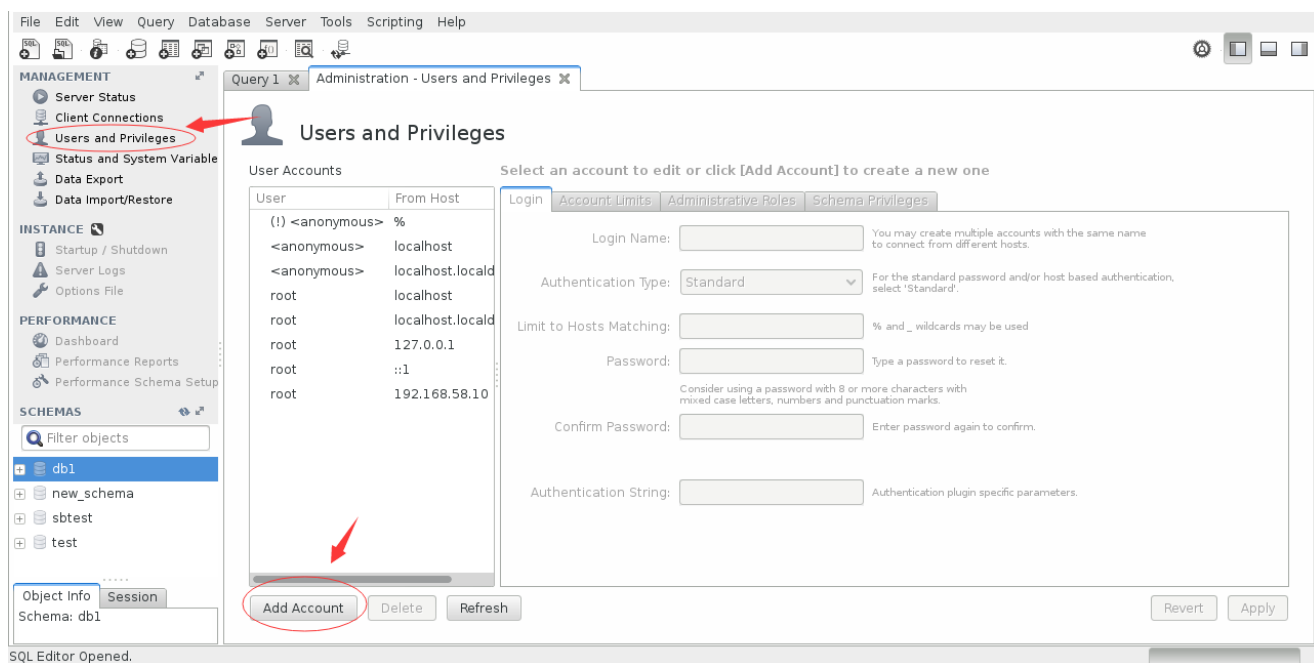


最后 执行 “File”->“Export” 按钮，选择 Forward Engineer SQL CREATE Script (ctrl+shift+G). 这样就可以把模型导出为SQL脚本文件。现在执行这个SQL文件就OK了。

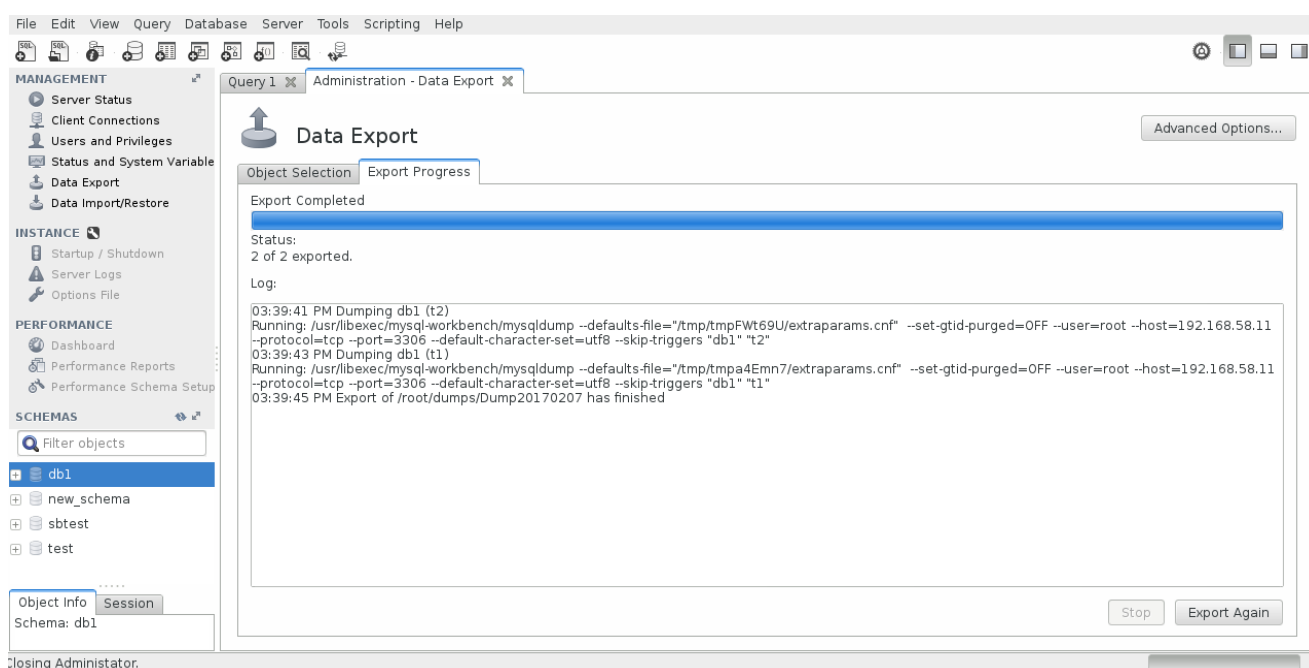
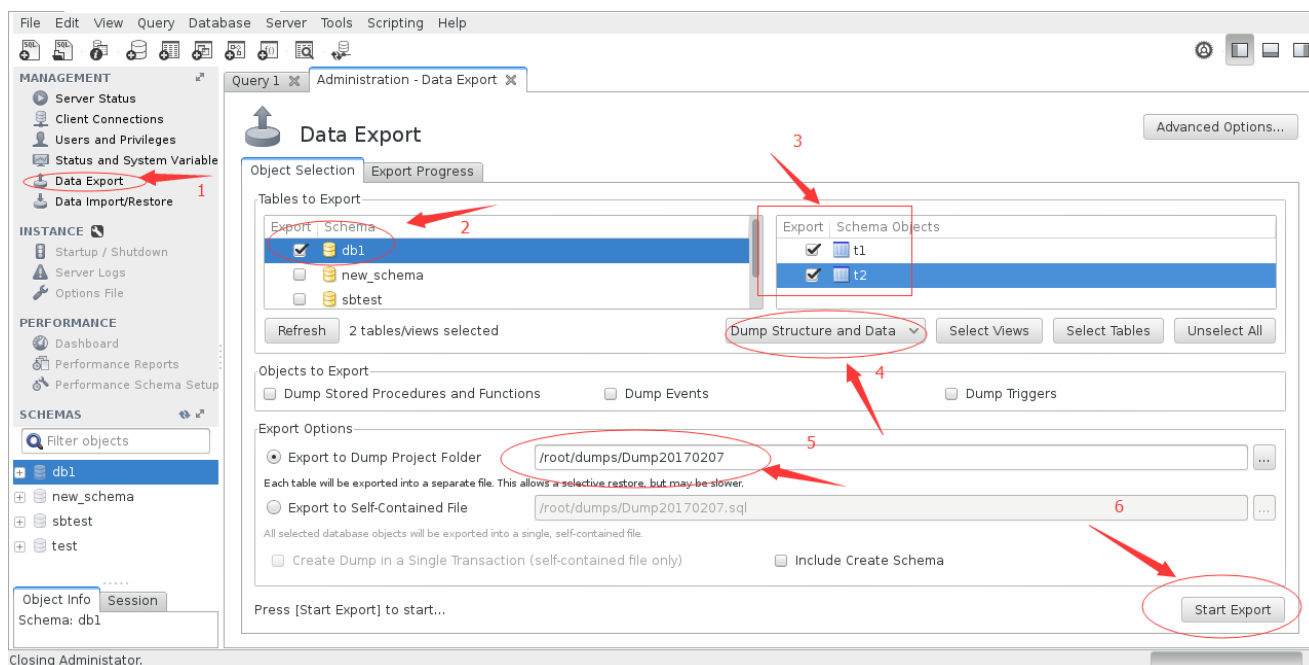
Server Administration的基本操作

- 管理MySQL用户
- 备份MySQL数据库
- 还原MySQL数据库

管理MySQL用户



备份MySQL数据库



改备份为逻辑备份，使用的是mysqldump命令进行的，备份数据存放如下：

```
[root@workstation ~]# ls /root/dumps/Dump20170207/
db1_t1.sql  db1_t2.sql
```

还原MySQL数据库

将db1库中的t1和t2表删除，进行还原

File Edit View Query Database Server Tools Scripting Help

MANAGEMENT

- Server Status
- Client Connections
- Users and Privileges
- Status and System Variable
- Data Export
- Data Import/Restore** 1

INSTANCE

- Startup / Shutdown
- Server Logs
- Options File

PERFORMANCE

- Dashboard
- Performance Reports
- Performance Schema Setup

SCHEMAS

Filter objects

- db1
 - Tables
 - Views
 - Stored Procedures

Object Info Session
No object selected

Query 1 Administration - Data Import/Restore

Data Import

Import from Disk Import Progress

Import Options

☒ Import from Dump Project Folder ... 2

Select the Dump Project Folder to import. You can do a **selective restore**.

☐ Import from Self-Contained File ...

Select the SQL/dump file to import. Please note that the whole file will be imported.

Default Schema to be Imported To

Default Target Schema: The default schema to import the dump into. NOTE: this is only used if the dump file doesn't contain its schema, otherwise it is ignored.

Select Database Objects to Import (only available for Project Folders)

Import	Schema
<input checked="" type="checkbox"/>	db1

Import	Schema Objects
--------	----------------

3

Press [Start Import] to start...

Query Completed

Query 1 Administration - Data Import/Restore

Data Import

Import from Disk Import Progress

Import Completed

Status:
2 of 2 imported.

Log:

```
03:46:20 PM Restoring db1 (t1)
Running: /usr/libexec/mysqld --defaults-file="/tmp/tmpmBd0Bo/extraparams.cnf" --protocol=tcp --host=192.168.58.11 --user=root --port=3306
--default-character-set=utf8 --comments --database=db1 < "/root/dumps/Dump20170207/db1_t1.sql"
03:46:22 PM Restoring db1 (t2)
Running: /usr/libexec/mysqld --defaults-file="/tmp/tmpPzPb1I/extraparams.cnf" --protocol=tcp --host=192.168.58.11 --user=root --port=3306 --
default-character-set=utf8 --comments --database=db1 < "/root/dumps/Dump20170207/db1_t2.sql"
03:46:24 PM Import of /root/dumps/Dump20170207 has finished
```