千锋教育
www.mobiletrain.org

原创经典 程序员典藏

3天学会 MySQL®

千锋教育李文凯 编著

MySQL入门圣经

- √ 数据库的用途和概念
- √ 娱乐化讲解表关系
- √ mysql 数据库安装
- √ 数据语句操作类型

- ■ 适合0基础入门学习，简单易懂
- ■ 本书知识点为实际工作中最实用的知识
- ■ 丰富实例，代码步骤化
- ■ 本书开源免费，将不断更新最新的实例和调整优化

目　录

前言
01. 数据库的用途和概念
02. 娱乐化讲解表关系
03. mysql 数据库安装
04. 数据语句操作类型
05. 连接数据库
06. 数据库结构定义语句
    6.1 数据库操作
    6.2 数据表操作
    6.3 数据字段操作
07. 类型、字符集、引擎和索引
    7.1 数据类型
    7.2 字符集
    7.3 表引擎
    7.4 索引
08. 增删改查
    12.8.1 插入记录
    12.8.2 查询记录
    12.8.3 多表联合查询
    12.8.4 更新记录
    12.8.5 删除记录
09. DCL语句
10. 常用工具简介
附录1 . 学习MySQL常用的英文单词

前言

##前言

能够看到本书，你很幸运，也是我们相逢的缘分。

学习本书的过程中你将得到：

1. 完全免费，你可以随意打印下载，切无版权问题
2. 在官网获得视频支持
3. QQ群学习帮你解决问题
4. 认识很多大牛
5. 大量不定期的学习优惠券
6. 大量的代码实例下载

   … …

为什么选择这本书？

1. 适合0基础和入门学习，简单易懂
2. 所有的知识为实际工作中最实用的知识
3. 大量的实例
4. 现实生活的代码全都步骤化了
5. 本书的例子结合所以
6. 本书由于是开源免费的，将不断更新最新的实例和调整优化

版权声明

免责声明

本书贡献者

主要作者： `李文凯, bob`

欢迎加入我们共同创作本书、提出意见，让更多的初学者受益，让大学和工作迷茫的人，不再迷茫！

本书QQ群：535647412

01. 数据库的用途和概念

##数据库的用途

很多朋友最开始学习数据库的时候，很难理解数据库的作用。理不清楚，数据库与我们现实生活、虚拟生活到底有什么样的关系。

我们通过本章来加快初学者对于数据库和数据库系统的理解。

现如今，我们所有见到的跟日常生活有关、需要记录的基本全部放在数据库里面：

1. 身份证信息放在公安部的系统
2. 银行卡的余额和交易记录、转帐信息
3. 在酒店的开房信息（所有出现了某些方面的数据库被盗和信息泄漏）
4. 飞机、火车、汽车联网购票记录
5. 各个不同的网站、QQ、网上购物、贴吧、喜欢听的音乐、电影的收藏信息
6. 手机电话机录、余额、公交卡余额、水费、电费、彩票的购买记录
7. 打游戏的装备、等级、魔力、力量、攻击能力等信息
8. 美国航空母舰也在使用mysql数据库在管理航母的相关信息

… …等等

我们生活的一切全都记录在数据库里面。你可以想想，数据库有多么重要！

####在21世纪，人类没有了数据库，世界将会怎样？

##数据库的五个基本单位

1. 数据库服务器
2. 数据库
3. 数据表
4. 数据字段
5. 数据行

我们现在来对上面的五个基本单位进行说明：

1. 数据库服务器。是指用来运行数据库服务的一台电脑。在中小型企业通常为一台。在数据存储量计算量很大的时候可以存在多台。多台数据库服务器共同来存储或计算。由于数据安全非常重要，我们经常会对数据库服务器里面的数据进经备份。
2. 数据库。一个数据库服务器里面有可以有多个数据库。主要用来分类使用。我们可以建立交通信息数据库、游戏数据库、酒店开房数据库… … 主要用来将各个不同用途的数据，按照业务进行大块的划分。
3. 数据表。例如在游戏数据库中。根据这一款游戏又分为了不同的数据表。专门用来区分游戏不同的数据。例如：用户数据（用户、密码）；人物数据；所有装备和装备信息；用户的充值信息；药品、魔力药水信息…

...等

4. 数据字段，也叫数据列。就是我们日常所见表格里面的列。在表格中，我们会将一张用户表分成多个列。如下（表一）所示：用户编号、用户名、性别、年龄是字段。在真正的数据库中数据字段需要换成英文需要写成：id、username、sex、年龄。
5. 数据行。真正的数据存在每一个表的行里面。字段（列）划分出来了一个表应该按照什么样的格式存数据。而行，是真正的数据。每一行需要遵循数据字段（列）的规范和要求进行存入数据。

(表一)

用户编号	用户名	性别	年龄
1	李文凯	男	18
2	景田	女	16
3	宁泽涛	男	22

02. 娱乐化讲解表关系

上一章我们讲到了数据库的应用范围非常广泛。如果没有了数据库，可能我们未来寸步难行。

学计算机的男孩、女孩现在都挺多。特别是80、90后互联网原住民，很多人都特别爱玩游戏。我们通过游戏里面的用户装备信息讲解表的关系。

银行取钱、转账、发红包也是我们日常中最常用银行卡操作，我们还用银行卡的存取讲解表的关系。

##游戏里装备和用户的关系

在游戏里面的某个人物有头盔、衣服、靴子、武器、项链。

并且，每一个不同的武器会增加上不同的攻防值。那我们就可以这么来模拟游戏的表设计。

> 注：以下仅为了让大家更加理解游戏里、用户和装备的关系。

用户表

用户编号	用户名	面具	靴子	武器
1	骷髅王	1		6
2	混沌骑士	4		2
3	半人马		3	7

装备表

装备ID	装备名	恢复血	恢复魔	防御
1	死亡面具	10	3	5
2	逃脱匕首	22	4	1
3	速度之靴	3	4	5
4	艺人面具	1	3	4
5	法师斗篷	5	6	3
6	魔棒	13	32	32
7	幽魂权杖	11	33	45

用户表中骷髅王带上了装备表中编号为1（死亡面具）和使用了编号为6（魔棒）的武器。

而用户3（半人马），使用了装备表中编号为3（的速度之靴）和7（幽魂权杖）。

这样就实现了游戏中某些用户戴上了装备。如果用户的行和列数据里面没有这个选项的话，则没有这个数据。

通过装备的属性值，与用户等级属性值相加就实现了用户穿上装备好的回血、防御值增加等不同的效果。

##银行开户、取现和转帐

我们在日常生活中经常进行的一个活动就是使用银行卡付钱，在银行的账单中，忠实的记录着我们每一笔交易。

我们来通过表格的方式来模拟：

银行卡ID	银行卡号	密码	是否冻结	余额
1	6222 0202 0002 66014	e10adc3949ba59abbe56e057f20f883e	0	12345.00
2	5423 4321 4567 8889	5bd2026f128662763c532f2f4b6f2476	1	45.58

上表中设计了用户的银行卡ID、卡号、密码、是否冻结和余额。

场景模拟：

1. 用户插入卡、输入密码正确后。则可以取钱
2. 社会工程学中，人们喜欢用相同的密码。因此，用户的密码必须要进行再次加密，不可逆向解密。因为害怕看到了某个用户的密码后，用这个密码去尝试用户的其他银行卡。
3. 我们自行规定：冻结状态可以设置为0（未冻结）和 1 （冻结了）。如果银行收到法院的通知。则将冻结状态设置为1。有钱也不让取钱。（这块业务逻辑需要在程序中实现）。
4. 用户若取钱了，或者存钱了将用户的余额增加或者减少。同时将记录，记录至交易流水中。

交易流水表

银行卡ID	操作	地点
1	-1000.00	北京市百度大厦ATM
1	+34000.00	上海外滩xx银行营业室
2	-12.08	淘宝网购

每当用户的余额发生变化的时候，我们都会忠实的记录到交易流水表中。让交易可查、可追述。

这样就模拟了银行的冻结、取现、存钱等流程。

注：

密码必须使用md5等加密方式帮用户进行加密。用户输入原密码如：123456。我们使用md5将用户输入的123456加密后与数据库的密码进行对比。

一致则密码通过。不一致则用户将密码输入错误了。

这样就实现保证用户密码安全，防止内部人员泄漏用户密码的可能性。

更多的密码知识，我们在下册和进阶项目中更多的为大家讲解。

03. mysql 数据库安装

##XAMPP中的MySQL

若你不是使用的XAMPP安装包安装的MySQL服务器请略过此块。看一块的《全新安装MySQL服务器》。

如果你通过《PHP入门圣经》来学习的，我们在安装时教大家安装的是XAMPP集成环境包。

在环境包中大家已经将MySQL数据库服务器安装好了。不需要再进行安装。

全新安装MySQL服务器

MySQL是跨平台的服务器,windows操作系统下的使用与linux下的使用几乎一模一样。

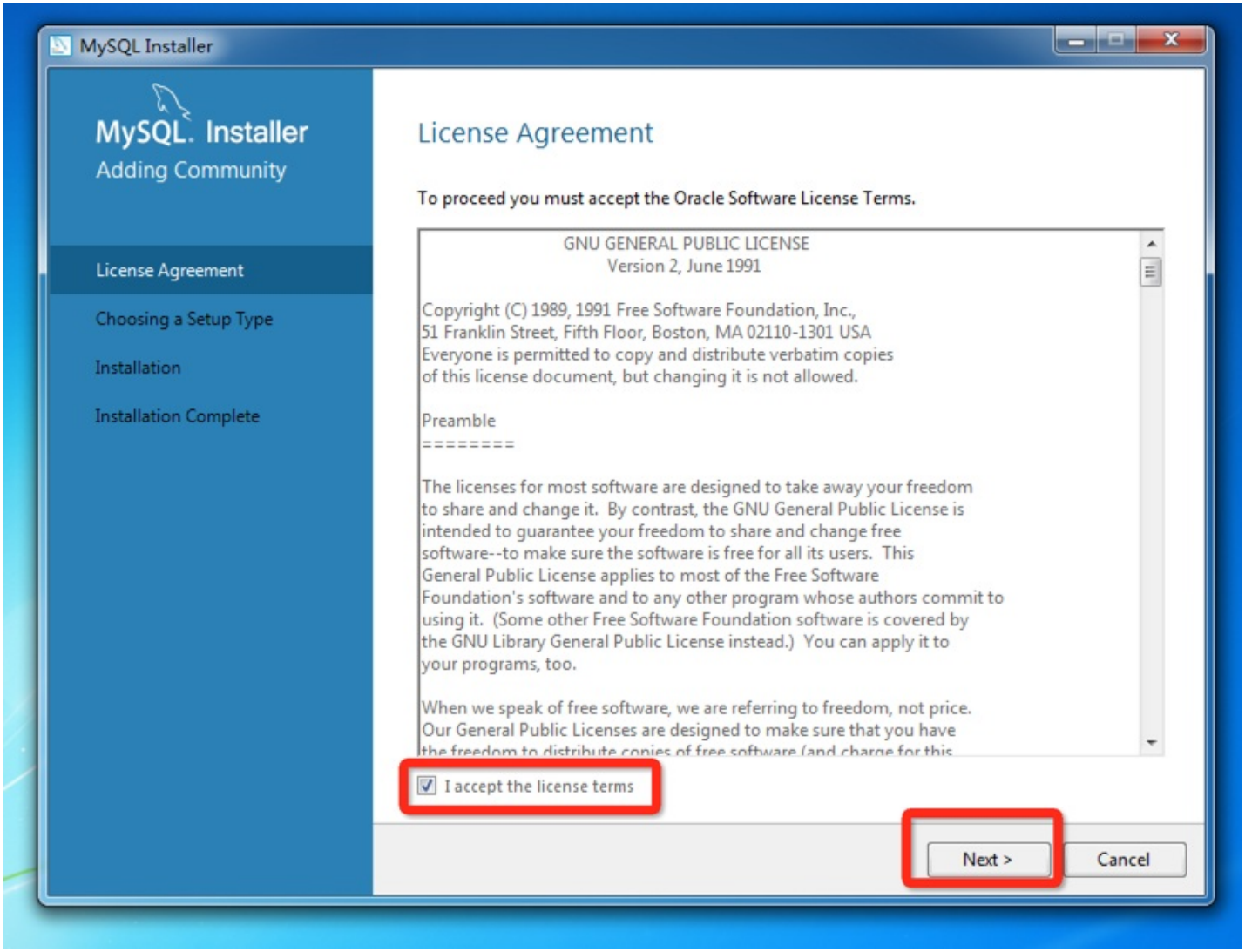
我们将会以windows平台为基础来讲解MySQL 服务器的安装。

> 注：Linux操作系统对很多初学者有困难。并且windows具有最广泛的用户，本章讲解以windows安装为主。
> 若您熟悉Linux操作系统，也可自行安装后接着下面课程的学习。

####一、下载安装包

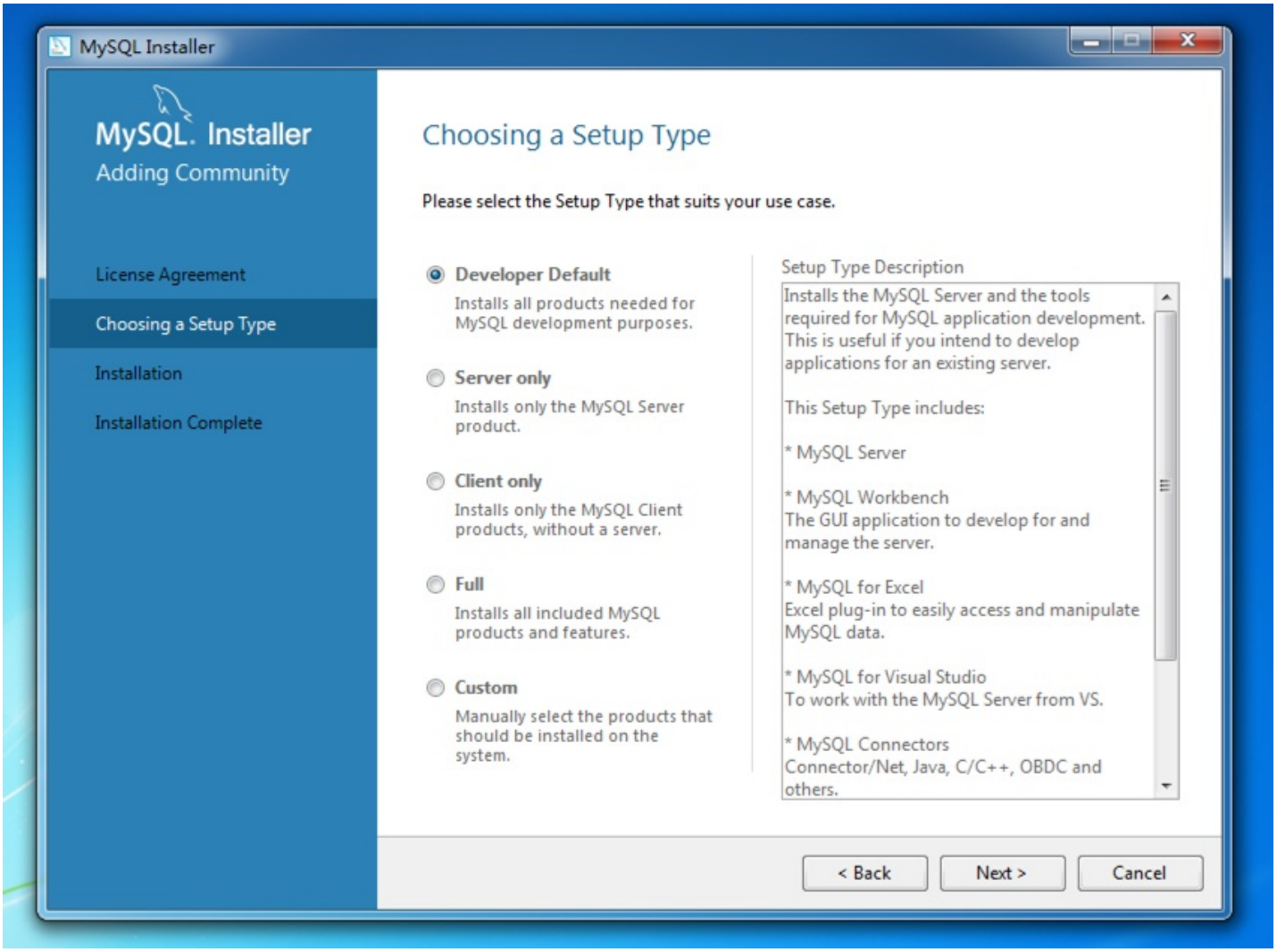
1. 百度搜索关键词：mysql server下载
2. 访问官网下载：http://dev.mysql.com/downloads/mysql/

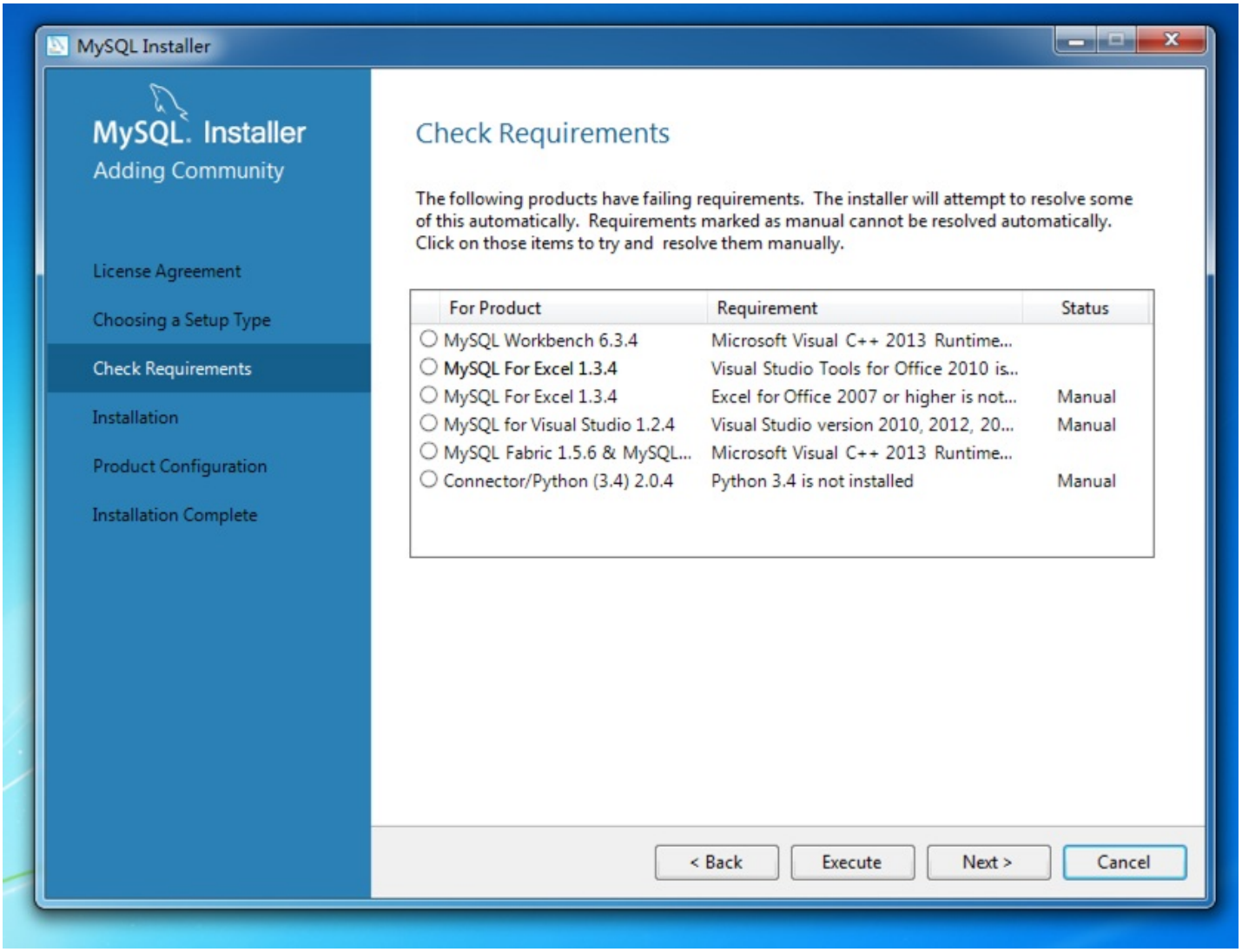
####二、打开安装包，同意协议，下一步

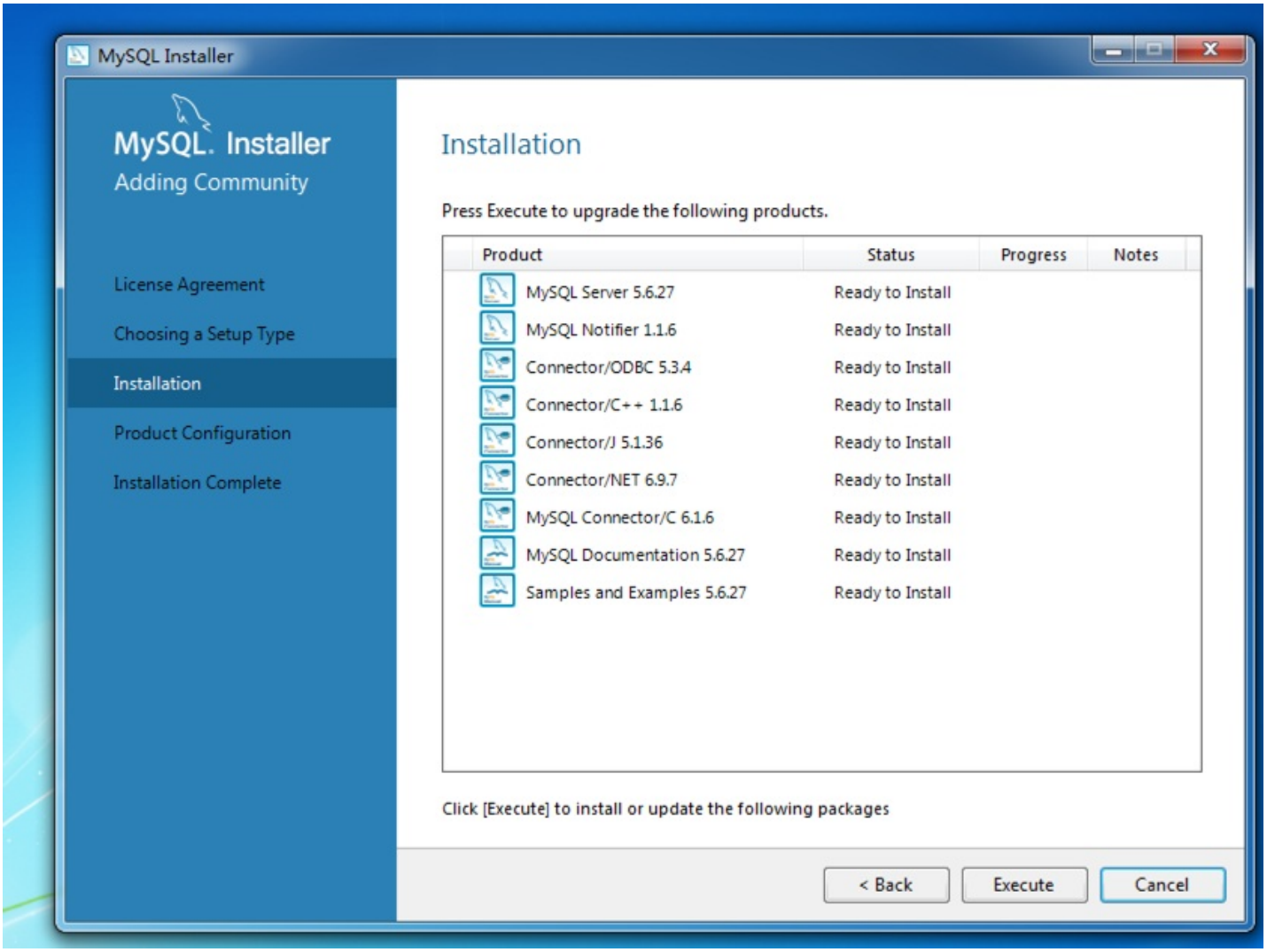


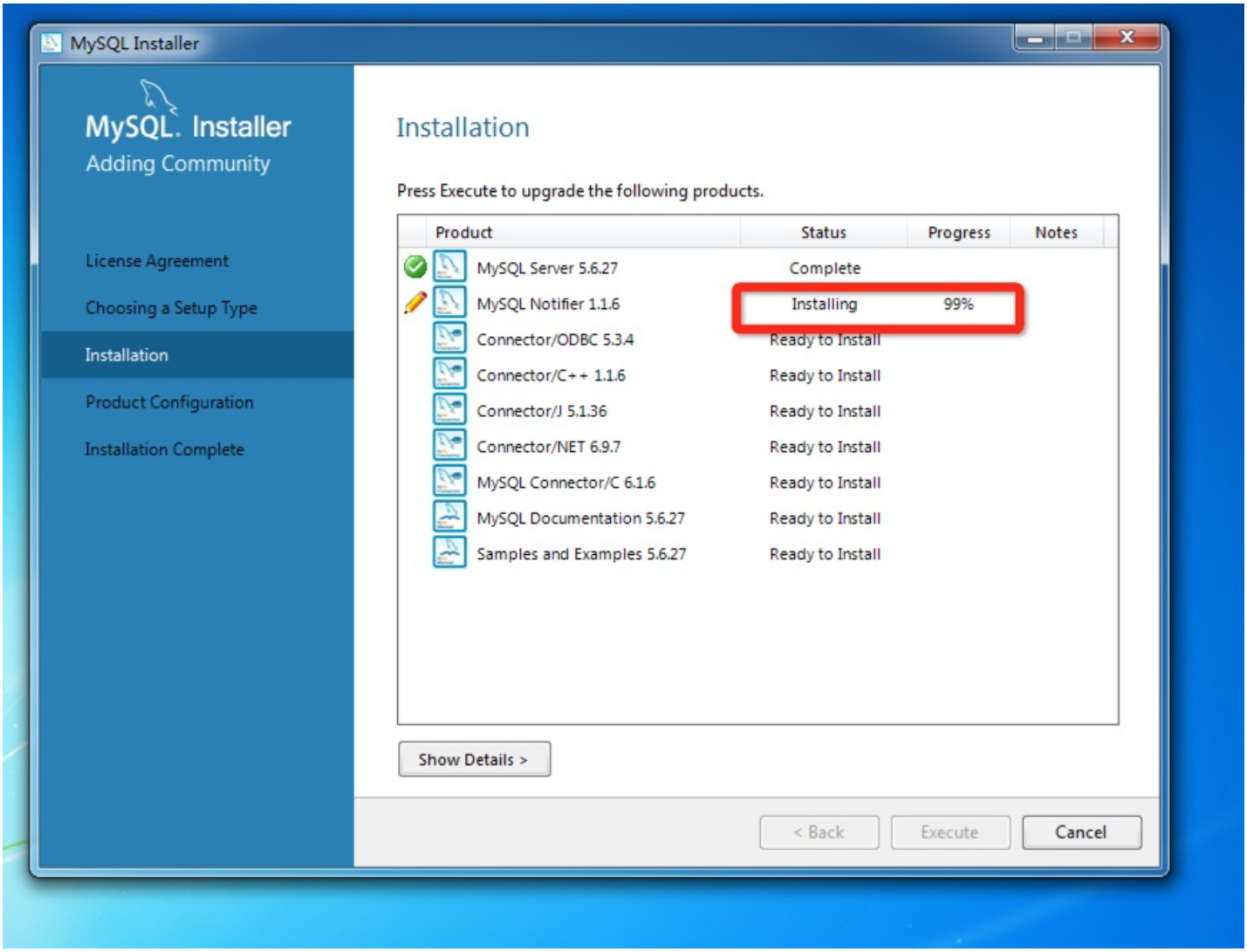
####三、选择服务器安装

1. develop 是指服务器为开发机【推荐】
2. server only只安装服务器
3. client only 只安装客户操作端
4. full 全部安装
5. custom 自定义


MySQL Installer
MySQL. Installer
Adding Community
License Agreement
Choosing a Setup Type
Installation
Installation Complete
Choosing a Setup Type
Please select the Setup Type that suits your use case.
Developer Default
Installs all products needed for MySQL development purposes.
Server only
Installs only the MySQL Server product.
Client only
Installs only the MySQL Client products, without a server.
Full
Installs all included MySQL products and features.
Custom
Manually select the products that should be installed on the system.
Setup Type Description
Installs the MySQL Server and the tools required for MySQL application development. This is useful if you intend to develop applications for an existing server.
This Setup Type includes:
* MySQL Server
* MySQL Workbench
The GUI application to develop for and manage the server.
* MySQL for Excel
Excel plug-in to easily access and manipulate MySQL data.
* MySQL for Visual Studio
To work with the MySQL Server from VS.
* MySQL Connectors
Connector/Net, Java, C/C++, OBDC and others.
< Back
Next >
Cancel


MySQL Installer
MySQL. Installer
Adding Community
License Agreement
Choosing a Setup Type
Check Requirements
Installation
Product Configuration
Installation Complete
Check Requirements
The following products have failing requirements. The installer will attempt to resolve some of this automatically. Requirements marked as manual cannot be resolved automatically. Click on those items to try and resolve them manually.
For Product
Requirement
Status
MySQL Workbench 6.3.4
Microsoft Visual C++ 2013 Runtime...
MySQL For Excel 1.3.4
Visual Studio Tools for Office 2010 is...
MySQL For Excel 1.3.4
Excel for Office 2007 or higher is not...
Manual
MySQL for Visual Studio 1.2.4
Visual Studio version 2010, 2012, 20...
Manual
MySQL Fabric 1.5.6 & MySQL...
Microsoft Visual C++ 2013 Runtime...
Connector/Python (3.4) 2.0.4
Python 3.4 is not installed
Manual
< Back
Execute
Next >
Cancel


MySQL Installer
MySQL. Installer
Adding Community
License Agreement
Choosing a Setup Type
Installation
Product Configuration
Installation Complete
Installation
Press Execute to upgrade the following products.
Product
Status
Progress
Notes
MySQL Server 5.6.27
Ready to Install
MySQL Notifier 1.1.6
Ready to Install
Connector/ODBC 5.3.4
Ready to Install
Connector/C++ 1.1.6
Ready to Install
Connector/J 5.1.36
Ready to Install
Connector/NET 6.9.7
Ready to Install
MySQL Connector/C 6.1.6
Ready to Install
MySQL Documentation 5.6.27
Ready to Install
Samples and Examples 5.6.27
Ready to Install
Click [Execute] to install or update the following packages
< Back
Execute
Cancel



####五、服务器参数修改

这一步MySQL server会在界面中显示，由用户勾选完成最后的安装配置文件的自动配置。

其中：

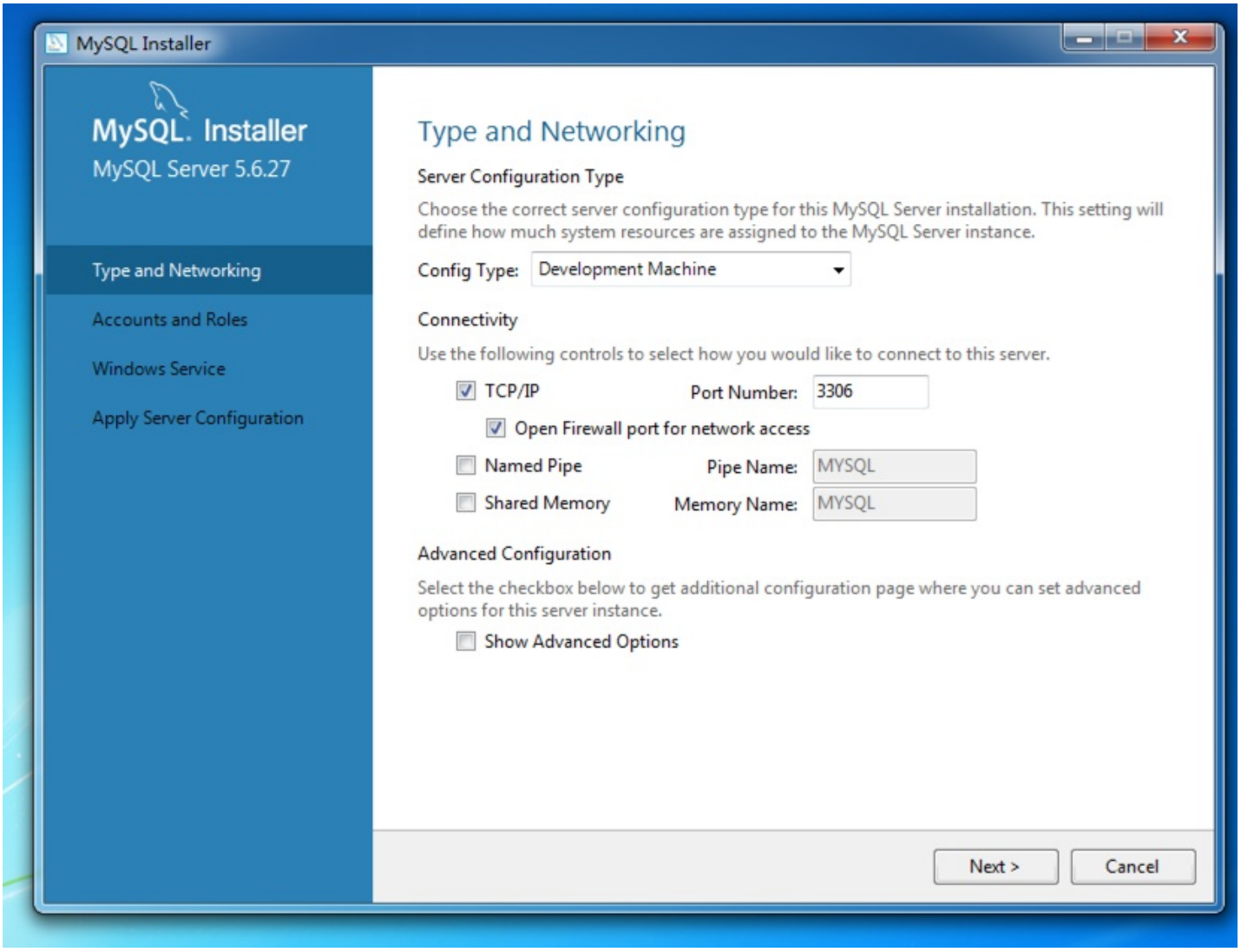
Config Type（配置文件类型）

1. Development Machine 是指开发服务器
2. Server Machine 服务器
3. Dedicated Machine 专用服务器仅作为数据库服务器使用

**TCP/IP **

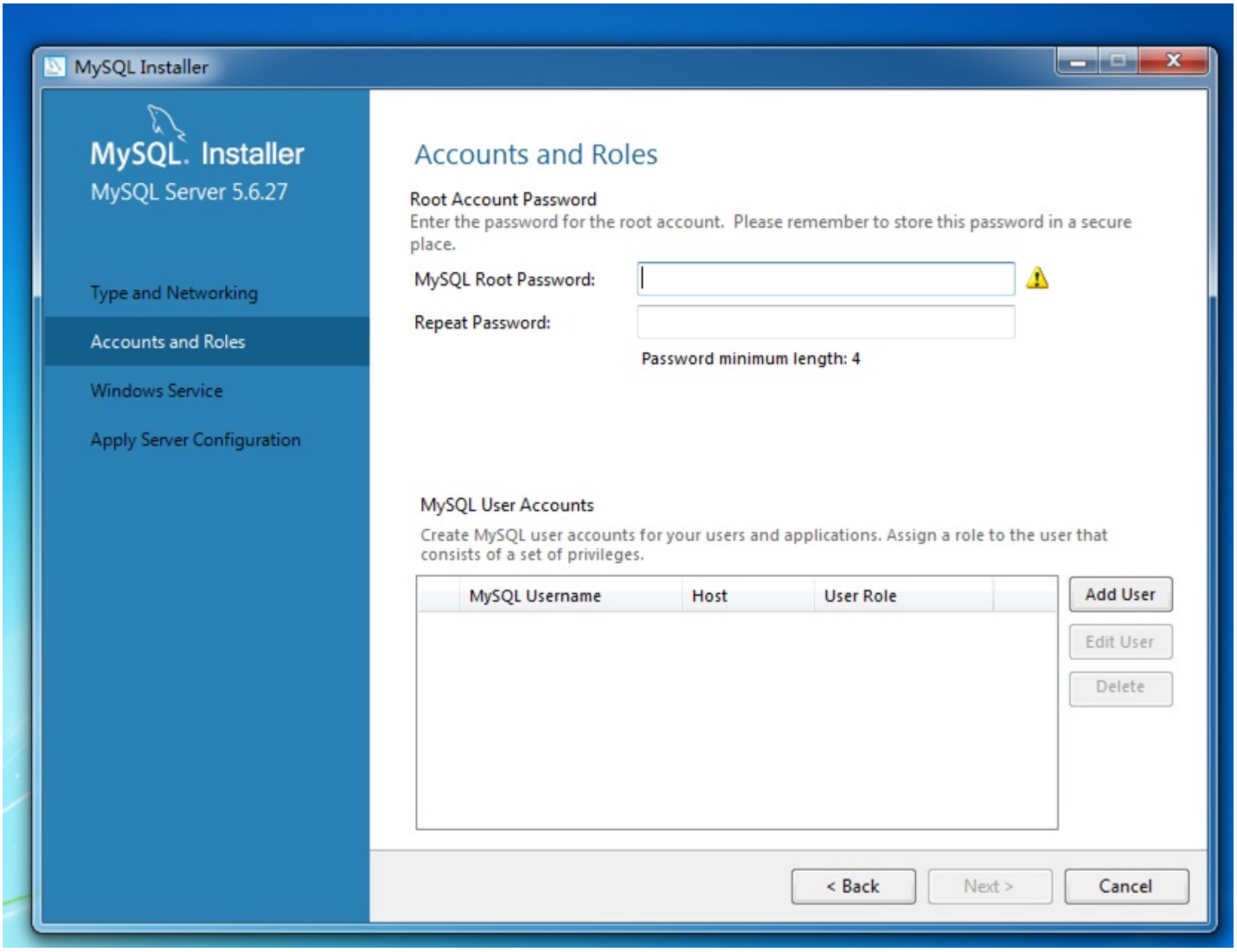
是指开放的端口

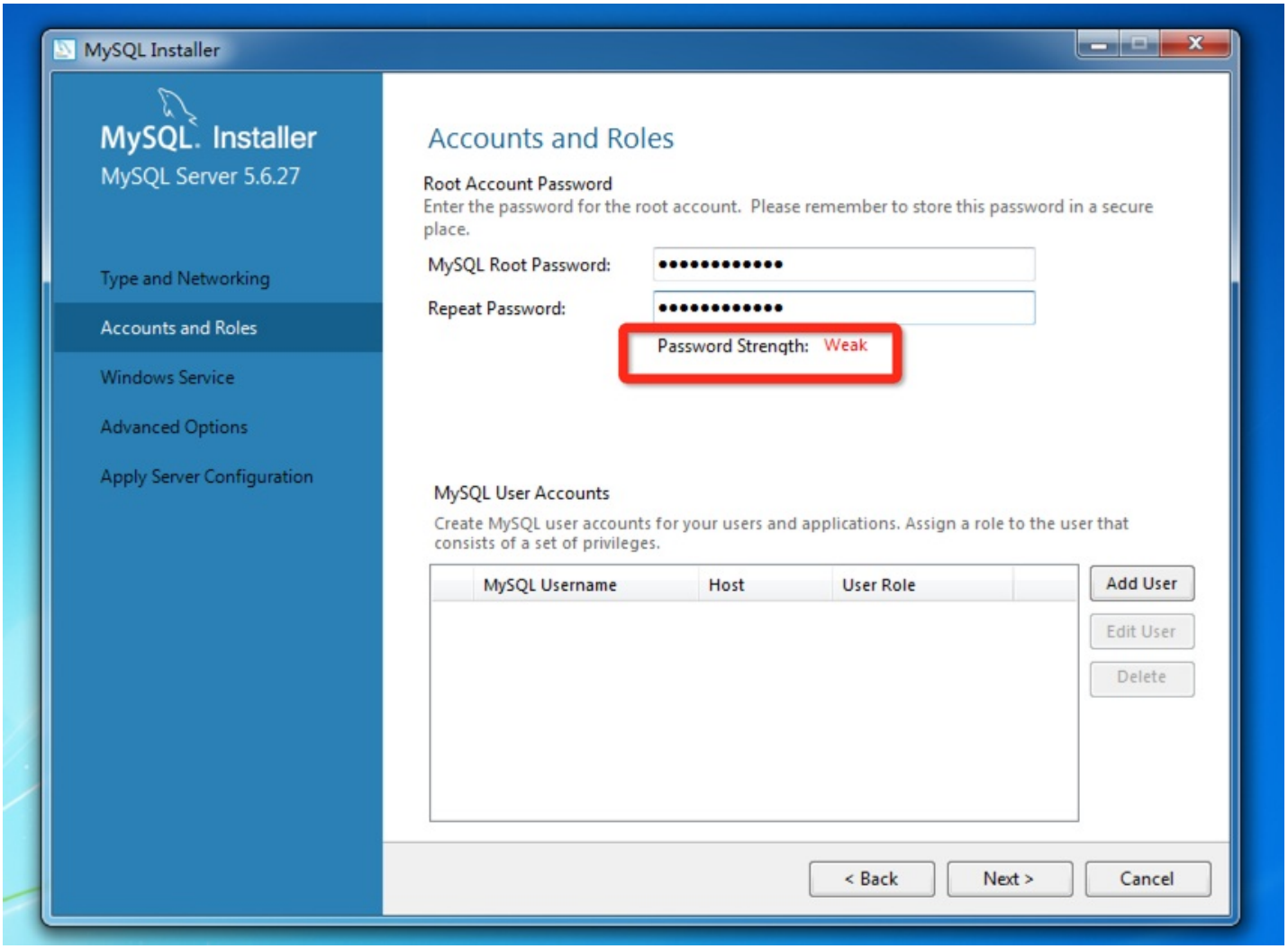
mysql 默认的端口是3306

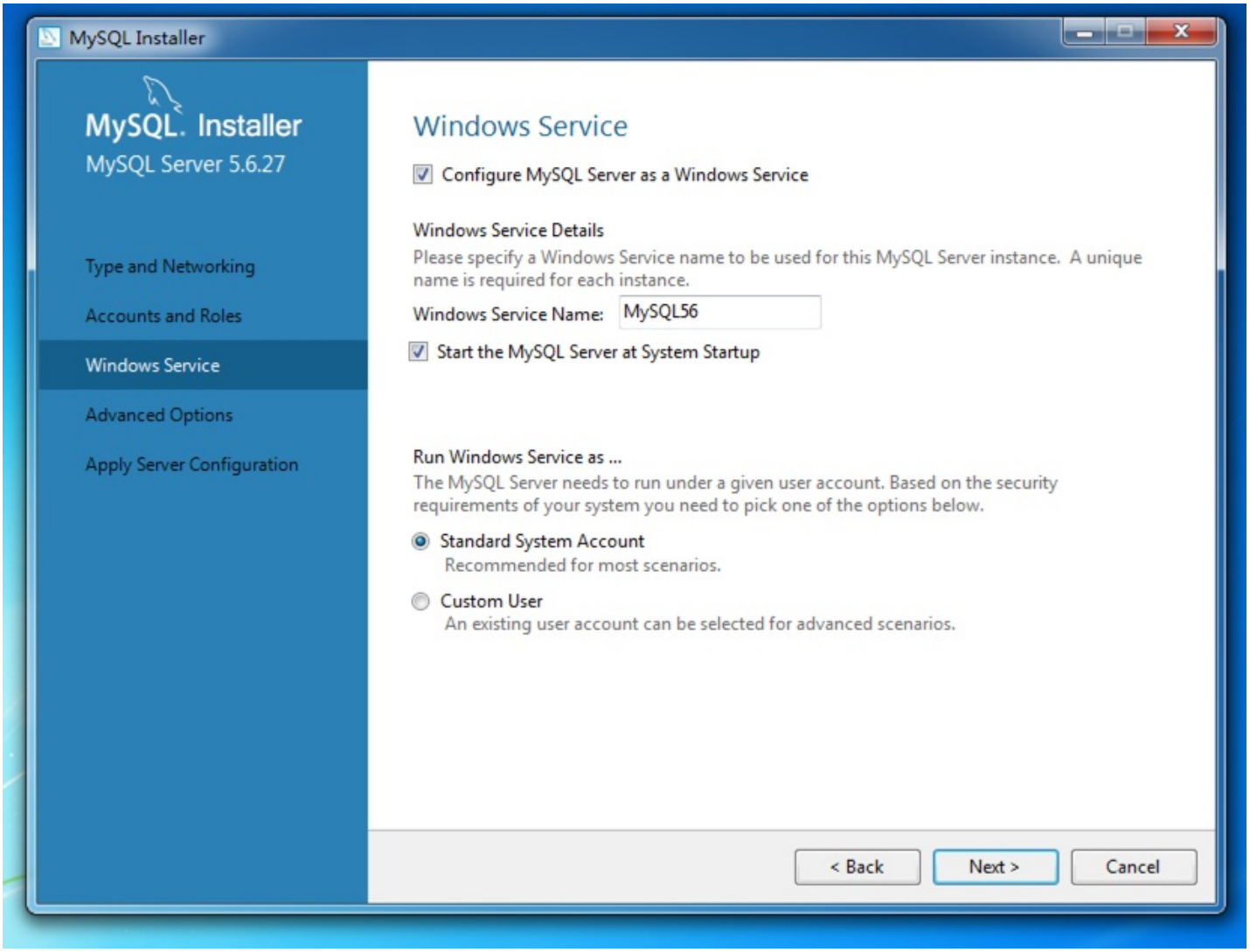


####六、设置密码

会提示密码的安全状态

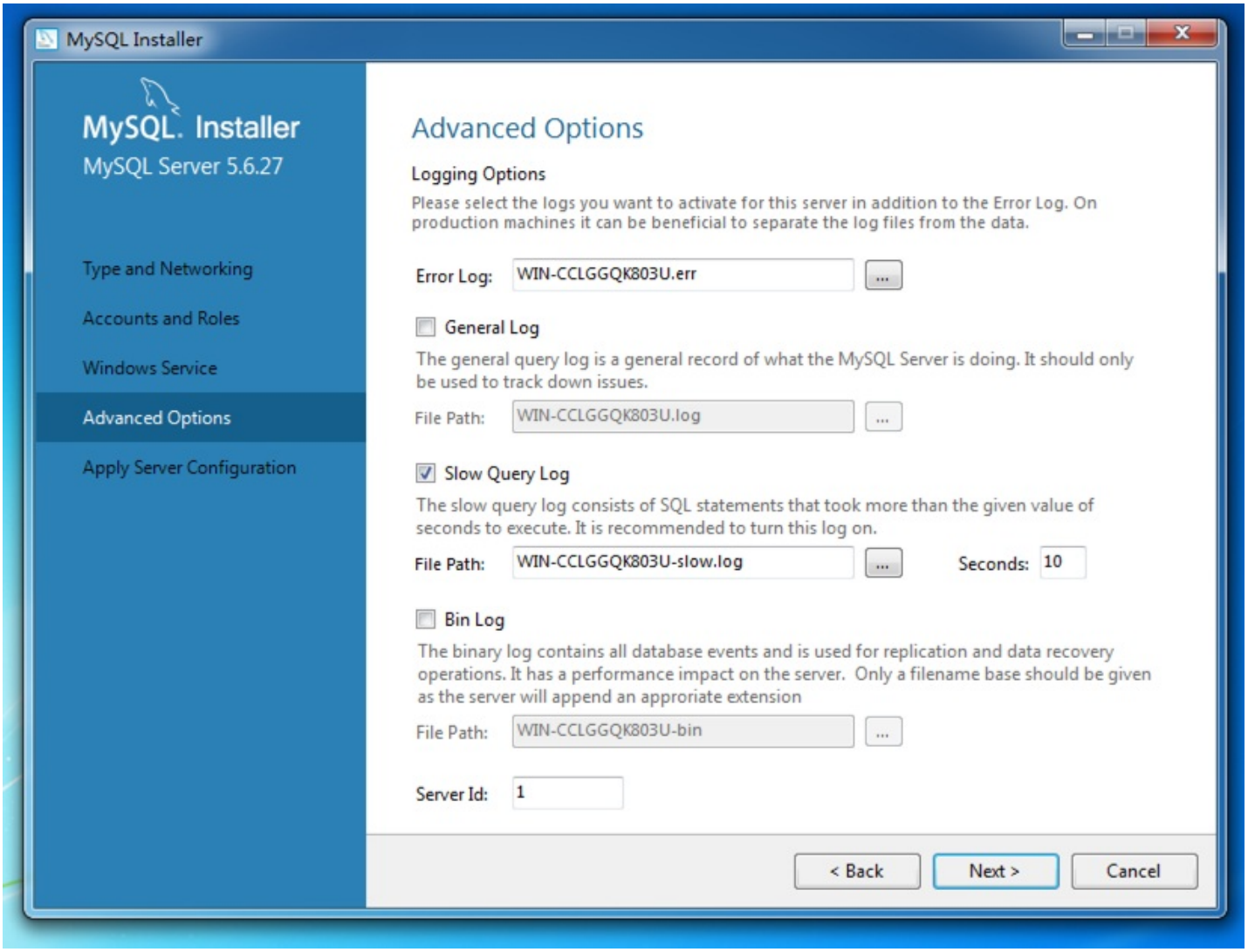

MySQL Installer
MySQL. Installer
MySQL Server 5.6.27
Type and Networking
Accounts and Roles
Windows Service
Apply Server Configuration
Accounts and Roles
Root Account Password
Enter the password for the root account. Please remember to store this password in a secure place.
MySQL Root Password:
Repeat Password:
Password minimum length: 4
MySQL User Accounts
Create MySQL user accounts for your users and applications. Assign a role to the user that consists of a set of privileges.
MySQL Username
Host
User Role
Add User
Edit User
Delete
< Back
Next >
Cancel


MySQL Installer
MySQL. Installer
MySQL Server 5.6.27
Type and Networking
Accounts and Roles
Windows Service
Advanced Options
Apply Server Configuration
Accounts and Roles
Root Account Password
Enter the password for the root account. Please remember to store this password in a secure place.
MySQL Root Password:
Repeat Password:
Password Strength: Weak
MySQL User Accounts
Create MySQL user accounts for your users and applications. Assign a role to the user that consists of a set of privileges.
MySQL Username
Host
User Role
Add User
Edit User
Delete
< Back
Next >
Cancel

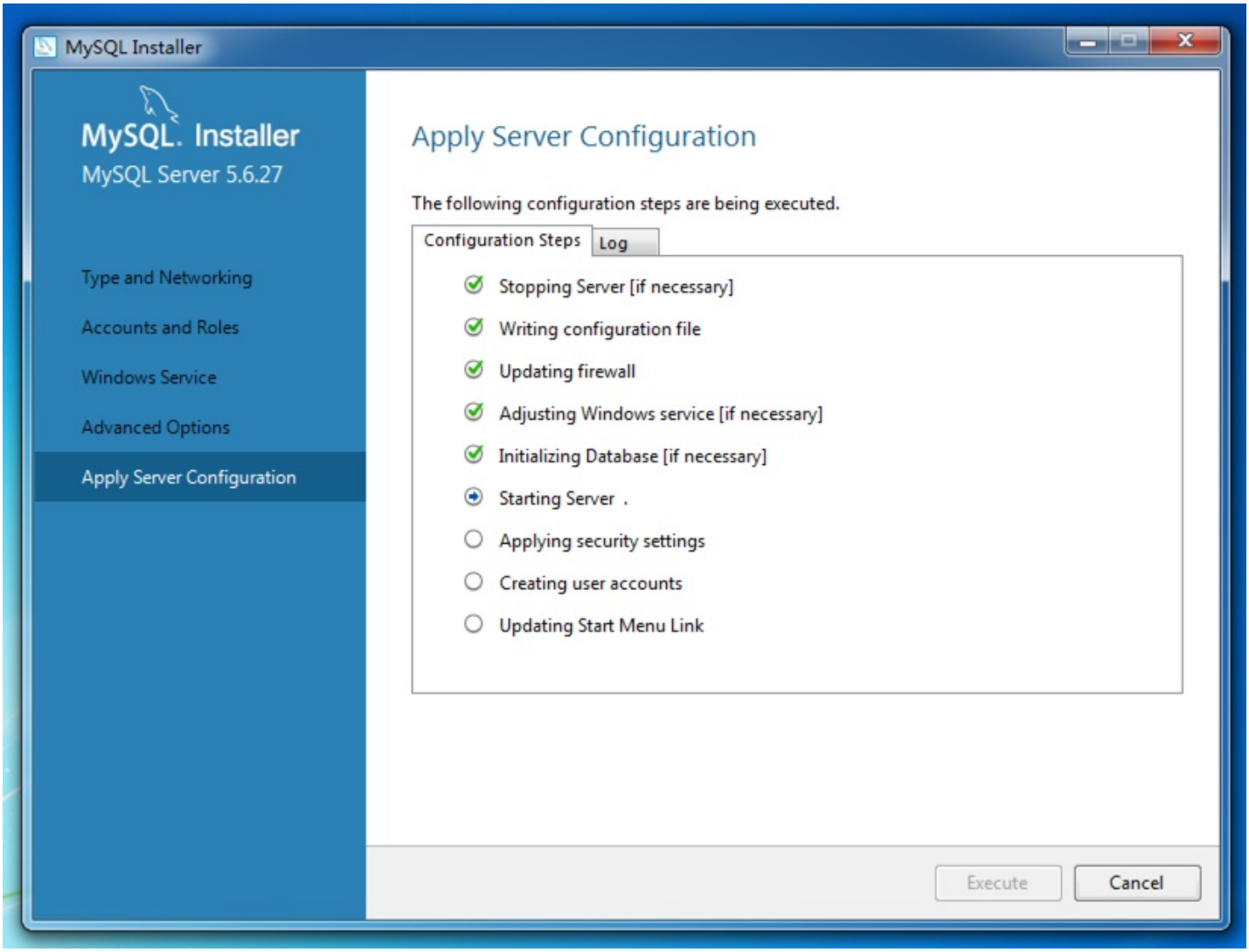


####七、日志文件存放位置确认

1. Bin log是二进制文件
2. Error Log 是指错误文件
3. slow Query Log 是慢查询日志


MySQL Installer
MySQL. Installer
MySQL Server 5.6.27
Type and Networking
Accounts and Roles
Windows Service
Advanced Options
Apply Server Configuration
Advanced Options
Logging Options
Please select the logs you want to activate for this server in addition to the Error Log. On production machines it can be beneficial to separate the log files from the data.
Error Log:
WIN-CCLGGQK803U.err
General Log
The general query log is a general record of what the MySQL Server is doing. It should only be used to track down issues.
File Path:
WIN-CCLGGQK803U.log
Slow Query Log
The slow query log consists of SQL statements that took more than the given value of seconds to execute. It is recommended to turn this log on.
File Path:
WIN-CCLGGQK803U-slow.log
Seconds:
10
Bin Log
The binary log contains all database events and is used for replication and data recovery operations. It has a performance impact on the server. Only a filename base should be given as the server will append an approriate extension
File Path:
WIN-CCLGGQK803U-bin
Server Id:
1
< Back
Next >
Cancel

####八、启动服务


MySQL Installer
MySQL. Installer
MySQL Server 5.6.27
Type and Networking
Accounts and Roles
Windows Service
Advanced Options
Apply Server Configuration
Apply Server Configuration
The following configuration steps are being executed.
Configuration Steps
Log
Stopping Server [if necessary]
Writing configuration file
Updating firewall
Adjusting Windows service [if necessary]
Initializing Database [if necessary]
Starting Server .
Applying security settings
Creating user accounts
Updating Start Menu Link
Execute
Cancel

###恭喜你安装成功了哟！

04. 数据语句操作类型

学习数据库安装后，最重要的就是学习SQL语句。

SQL是操作数据库的核心，也是本章开始的一句话：MySQL对于PHP程序员来说就是将业务转化成表结构。做好业务中的增、删、改、查。

结构化查询语言(Structured Query Language)简称SQL，是一种特殊目的的编程语言，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。

SQL是最重要的关系数据库操作语言，并且它的影响已经超出数据库领域，得到其他领域的重视和采用，如人工智能领域的数据检索等。

SQL是关系模型的数据库应用语言，由IBM在20世纪70年代为其关系型数据库 System R 所开发。

SQL 是1986年10 月由美国国家标准局（ANSI）通过的数据库语言美国标准，接着，国际标准化组织（ISO）颁布了SQL正式国际标准。1989年4月，ISO提出了具有完整性特征的SQL89标准，1992年11月又公布了SQL92标准。

虽然各个数据库系统略有不同，但是他们基本均遵循SQL 92标准。或者在SQL 92上做了一些简单的扩展和变化。

学好了MySQL 的SQL 语法，其他的SQL语法学习起来均是万变不离其中。

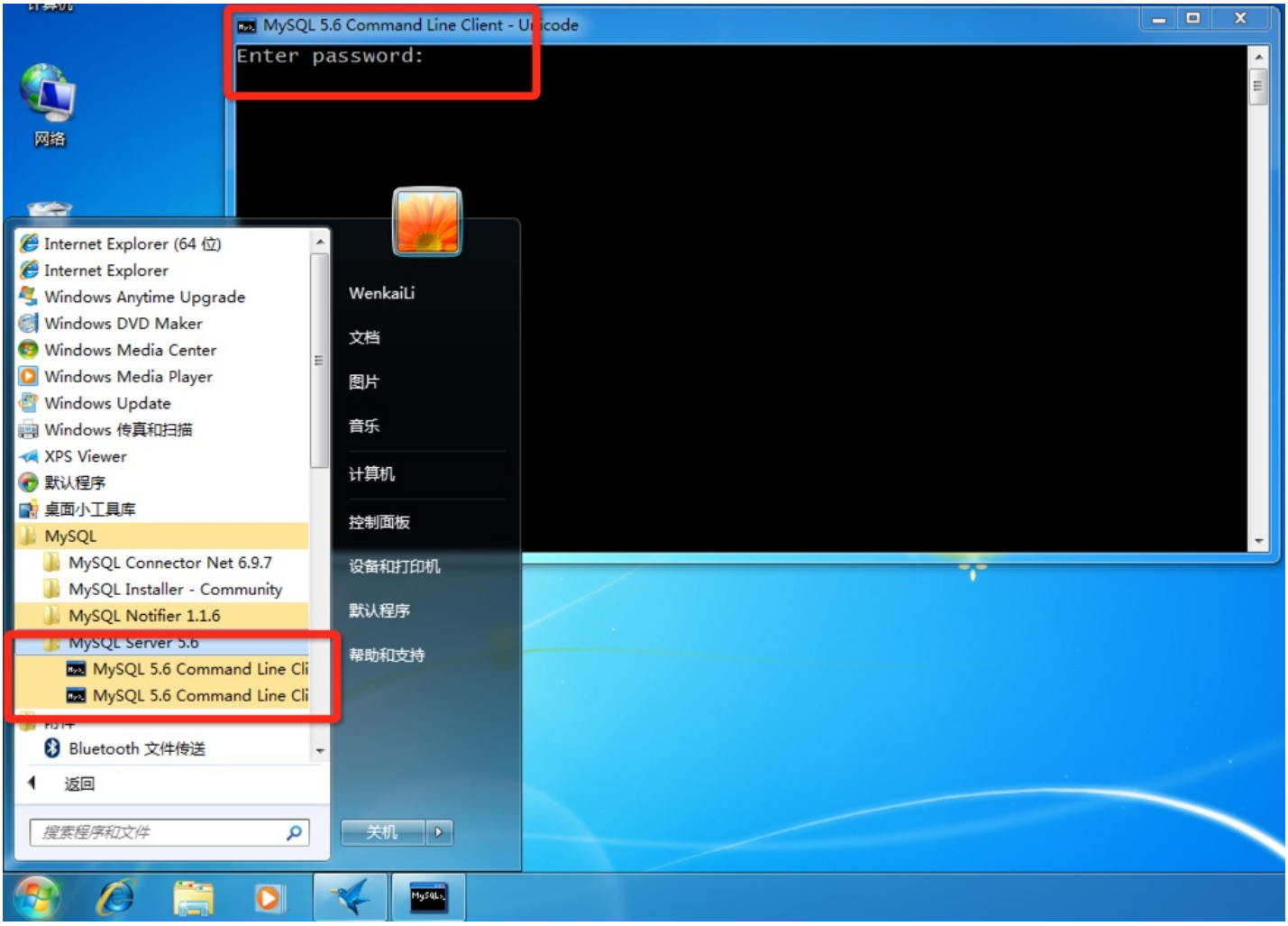
SQL语句按照其功能范围不同可分为3个类别：

1. 数据定义语言(DDL ，Data Defintion Language)语句：数据定义语句，用于定义不同的数据段、数据库、表、列、索引等。常用的语句关键字包括create、drop、alter等。
2. 数据操作语言(DML ， Data Manipulation Language)语句：数据操纵语句，用于添加、删除、更新和查询数据库记录，并检查数据的完整性。常用的语句关键字主要包括insert、delete、update和select等。
3. 数据控制语言(DCL， Data Control Language)语句：数据控制语句，用于控制不同数据段直接的许可和访问级别的语句。这些语句定义了数据库、表、字段、用户的访问权限和安全级别。主要的语句关键字包括grant、revoke等。

05. 连接数据库

##方法一

安装后，可以在开始菜单的列表中找到MySQL Command Line 点击操作的命令行终端操作。效果如图：



##方法二

如果加入到了windows的环境变量中，可以在命令行下直接操作。

在命令行下，通过以下命可以连接到数据库服务器：

```
mysql -h localhost -u root -p
```

上面的命令中： mysql 表示 mysql数据库启动工具。

参数说明：

参数	说明
-h	表示数据库连接地址，连接本机可不填
-u	表示要登录的用户
-p	表示使用密码登录

注：通常我们不直接输入密码。而是在回车之后，输入密码。因为，密码输入时的字符是不可见的，输完密码直接回车登录。防止旁边有人把重要的密码看走。
如果没有什么别的问题，登陆成功之后会出现下面内容：

> mysql -u root -p
> Enter password:
> Welcome to the MySQL monitor. Commands end with ; or \g.
> Your MySQL connection id is 7
> Server version: 5.6.25 MySQL Community Server (GPL)

> Copyright (c) 2000, 2015, 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>

上面的中文意思翻译过来是说，欢迎使用MySQL的命令行操作工具。每一个命令结束可以输入\g 或者；
mysql当前是第 7次连接。
当前数据库的版本是5.6.25社区支持版。遵循GPL协议。
版权所有：2000至 2015。归Oracle及其子机构拥有所有权。
如果需要帮助的话，通过 ‘help;’ 或者 ‘\h’ 命令来显示帮助内容，通过 ‘\c’ 命令来清除命令行历史。
mysql>
mysql > 表示等待输入指令。

注：
在登陆成功后有这么一句提示，可能大家不太理解：
Your MySQL connection id is 7
表示第7次连接登陆，每登陆一次这个id为加1。下一次显示的会是第8次。
你可以实验试试哟：)

06. 数据库结构定义语句

DDL是数据定义语言，简单来说，就是对数据库、数据表、数据字段进行创建、删除、修改和操作语言，它和数据操作语句（DML）最大的区别在于DML（数据操作语句）是对表内部数据的操作，不涉及表的定义、结构的修改，也不涉及其他对象。

我们在本章在讲解这一块时将其分为了三块：

1. 数据库操作
2. 数据表操作
3. 数据字段操作

6.1 数据库操作

##创建数据库

类别	详细解示
基本语法	create database 数据库名;
示例	create database liwenkai;
示例说明	创建一个数库，数据库的名字为liwenkai

示例：

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

“Query OK” 表示上面的命令执行成功，所有的 DDL 和 DML(不包 括 SELECT)操作执行成功后都显示“Query OK”，这里理解为执行成功就可以了；“1 row affected” 表示操作只影响了数据库中一行的记录，“0.00 sec”则记录了操作执行的时间。

如果已经存在这个数据库，系统会示:

```
mysql> create database liwenkai;
ERROR 1007 (HY000): Can't create database 'liwenkai'; database exists
```

##查看数据库

基本语法：

类别	详细解示
基本语法	show databases;
示例说明	显示当前服务器的所有数据库

注意：

show是指显示

database 是指数据库

databases 是数据库的复数形式，指全部数据库。

示例：

```
mysql> show databases;
+--------------------+
```

```
| Database |
+--------------------+
| information_schema |
| mysql |
| performance_schema |
| user |
+--------------------+
4 rows in set (0.00 sec)
```

##选中数据库

基本语法：

类别	详细解示
基本语法	use 库名;
示例	use liwenkai
示例说明	使用数据库liwenkai

注意：

use 是指使用；

库名 是存在当前数据库系统中的具体的数据库的名称；

示例：

```
mysql> use liwenkai;
Database changed
```

这样就进入到了 liwenkai 数据库中了。当然你可以使用 use 语句随时切换要操作的数据库，刚刚选中了 liwenkai ，现在我们切换到mysql内容的 mysql 数据库看看：

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

出现 " Database changed " 表示切换成功。然后，看看 mysql数据库里面有什么内容（和查看当前数据库服务器数据库一样使用 show 语句）

##查看数据库中的表

进入到库后我们可以看这个库里面有多少个数据表。

类别	详细解示
基本语法	show tables;
示例说明	显示当前数据库下所有的表

使用use 进入到某个数据库后可以使用show tables

示例，查看当前数据库的表：

```
mysql> show tables;
+--------------------------+
| Tables_in_mysql |
+--------------------------+
| columns_priv |
| db |
| event |
| func |
| general_log |
| help_category |
| help_keyword |
| help_relation |
| help_topic |
| innodb_index_stats |
| innodb_table_stats |
| ndb_binlog_index |
| plugin |
| proc |
| procs_priv |
| proxies_priv |
| servers |
| slave_master_info |
| slave_relay_log_info |
| slave_worker_info |
| slow_log |
| tables_priv |
| time_zone |
| time_zone_leap_second |
| time_zone_name |
```

```
| time_zone_transition |
| time_zone_transition_type |
| user |
+--------------------------+
28 rows in set (0.00 sec)
```

这些表里面的内容是关系数据库服务器相关的用户、权限、数据库状态、设置等相关的信息数据。

##删除数据库

类别	详细解示
基本语法	drop database 库名;
示例	drop database liwenkai;
示例说明	删除一个数库，数据库的名字为liwenkai

注意：

drop 是汉语可以翻译为指掉下来，不要了的意思

database 是指库

库名 是指要删掉的库的名称

示例：

```
mysql> drop database liwenkai;
Query OK, 0 rows affected (0.01 sec)
```

【切记】注：数据库删除后，下面的所有数据都会全部删除，所以删除前一定要慎重并做好相应的备份。（若重要数据未备份，而实际中产生的数据风险与本书无关。）

6.2 数据表操作

##创建表

类别	详细解释
基本语法	create table 表名(字段名1 字段类型,....字段名n 字段类型n);
示例	create table user(username varchar(20),password varchar(32));
示例说明	创建一个表名叫user的表，第一个字段为username、表的字段类型为varchar长度为32个长度。第二个字段为password，类型也为varchar，长度也为32个长度。

注意：

1. 为了更好的让大家入门，数据类型暂时不在我们这一章的讲解范围。害怕大家顾此失彼。快速学习数据库的管理和操作语句非常的重要，数据类型、字段、字符集、引擎都属于了解的知识点。
2. . 字段类型大家现在只需要学会int,代表整型。float，代表浮点。char和varchar代表字符串即可。
3. 我们可以在类型后接上长度如：varchar(20)。

其他示例：

```
mysql> create table emp(
ename varchar(10),
hiredate date,
sal float(10,2),
deptno int(2)
);
Query OK, 0 rows affected (0.63 sec)
```

```
mysql> create table dept( deptno int(4), deptname varchar(20));
Query OK, 0 rows affected (0.12 sec)
查看表字段结构信息
```

类别	详细解示
基本语法	desc 表名;

示例	desc emp
示例说明	查看emp表的表结构

操作显示如下：

```
mysql> desc emp;
+----------+--------------+------+-----+---------+-------+
| Field | Type | Null | Key | Default | Extra |
+----------+--------------+------+-----+---------+-------+
| ename | varchar(10) | YES | | NULL | |
| hiredate | date | YES | | NULL | |
| sal | decimal(10,2) | YES | | NULL | |
| deptno | int(2) | YES | | NULL | |
+----------+--------------+------+-----+---------+-------+
4 rows in set (0.39 sec)
查看表的创建SQL语句
查看表创建语句
```

类别	详细解示
基本语法	show create table 表名 \G;
示例	show create table emp \G;
示例说明	查看表emp的创建语句

执行完整示例：

```
mysql> show create table emp \G;
Table: emp
Create Table: CREATE TABLE emp (
ename varchar(10) DEFAULT NULL,
hiredate date DEFAULT NULL,
sal decimal(10,2) DEFAULT NULL,
deptno int(2) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
1 row in set (0.00 sec)
ERROR:
No query specified
```

上面表的创建 SQL 语句中，除了可以看到表定义以外，还可以看到表的 engine（存储引擎) 和 charset(字符集)等信息。 “\G”选项的含义是使得记录能够按照字段竖着排列，对于内 容比较长的记录更易于显示。

##删除表

类别	详细解示
基本语法	drop table 表名;
示例	drop table emp;
示例说明	删除表emp

```
mysql> drop table emp;
Query OK, 0 rows affected (0.34 sec)
```

注：删除表。表和数据均会丢失，请勿必删除重要表之前备份数据。

##指定表引擎和字符集

在创建表最后，我们常用MyISAM或者InnoDB引擎。在指定引擎时，我们可以使用：

```
ENGINE=InnoDB
```

指定表默认字符集：

```
DEFAULT CHARSET=utf8
```

效果如下：

```
CREATE TABLE emp (
useraname varchar(10) DEFAULT NULL,
password date DEFAULT NULL,
)ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

6.3 数据字段操作

假设我们存在user表，user结构如下：

```
mysql> desc user;
+----------+--------------+------+-----+---------+-------+
| Field | Type | Null | Key | Default | Extra |
+----------+--------------+------+-----+---------+-------+
| username | varchar(10) | YES | | NULL | |
| password | varchar(32) | YES | | NULL | |
| createtime | int(10) | YES | | NULL | |
| createip | int(10) | YES | | NULL | |
+----------+--------------+------+-----+---------+-------+
4 rows in set (0.01 sec)
```

修改表字段类型 modify

类别	详细解示
基本语法	alter table 表名 modify 字段名 varchar(20);
示例	alter table user modify username varchar(20);
示例说明	将user表的username的类型改为varchar(20)

我们执行一下，看看结果：

```
mysql> alter table user modify username varchar(20);
Query OK, 0 rows affected (0.48 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

```
mysql> desc user;
+----------+--------------+------+-----+---------+-------+
| Field | Type | Null | Key | Default | Extra |
+----------+--------------+------+-----+---------+-------+
| username | varchar(20) | YES | | NULL | |
| password | varchar(32) | YES | | NULL | |
```

```
| createtime | int(10) | YES | | NULL | |
| createip | int(10) | YES | | NULL | |
+----------+--------------+------+-----+---------+-------+
4 rows in set (0.01 sec)
```

##增加表字段

类别	详细解示
基本语法	alter table 表名 add column 字段名 类型;
示例	alter table user add column age int(3);
示例说明	添加一个字段为age，类型为整型长度为3

```
mysql> alter table emp add column age int(3);
Query OK, 0 rows affected (0.40 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

```
mysql> desc user;
+----------+--------------+------+-----+---------+-------+
| Field | Type | Null | Key | Default | Extra |
+----------+--------------+------+-----+---------+-------+
| username | varchar(20) | YES | | NULL | |
| password | varchar(32) | YES | | NULL | |
| createtime | int(10) | YES | | NULL | |
| createip | int(10) | YES | | NULL | |
| age | int(3) | YES | | NULL | |
+----------+--------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

##增加字段时控制字段顺序

我们刚刚学了增加字段。如果你仔细实验发现每次都是增加在最后面，如何在第一个增加或者在指字字段之后增加呢？

类别	详细解释
基本语法	ALTER TABLE 表名 ADD 字段名 字段类型 AFTER 字段名;

示例	ALTER TABLE user ADD email VARCHAR(60) AFTER createip;
示例说明	user表中，在createip后增加一个字段为email，类型为varchar，长度为60

类别	详细解示
基本语法	ALTER TABLE 表名 ADD 字段名 字段类型;
示例	ALTER TABLE user ADD id INT(10) FIRST;
示例说明	user表中在最开始的位置增加一个字段为id,类型为int，长度为10

```
ALTER TABLE user ADD email VARCHAR(60) AFTER createip;
Query OK, 0 rows affected (0.40 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

```
mysql> desc user;
+----------+--------------+------+-----+---------+-------+
| Field | Type | Null | Key | Default | Extra |
+----------+--------------+------+-----+---------+-------+
| username | varchar(20) | YES | | NULL | |
| password | varchar(32) | YES | | NULL | |
| createtime | int(10) | YES | | NULL | |
| createip | int(10) | YES | | NULL | |
| email | varchar(60) | YES | | NULL | |
| age | int(3) | YES | | NULL | |
+----------+--------------+------+-----+---------+-------+
6 rows in set (0.00 sec)
```

##删除表字段

类别	详细解示
基本语法	alter table 表名 drop column 字段名;
示例	alter table user drop column age;
示例说明	在user表中删除字段age

```
mysql> alter table user drop column age;
```

```
Query OK, 0 rows affected (0.27 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

```
mysql> desc user;
+----------+--------------+------+-----+---------+-------+
| Field | Type | Null | Key | Default | Extra |
+----------+--------------+------+-----+---------+-------+
| username | varchar(20) | YES | | NULL | |
| password | varchar(32) | YES | | NULL | |
| createtime | int(10) | YES | | NULL | |
| createip | int(10) | YES | | NULL | |
| email | varchar(60) | YES | | NULL | |
+----------+--------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

##表字段改名

类别	详细解示
基本语法	alter table 表名 change 字段原名 字段新名 字段类型;
示例	alter table user change email em varchar(60);
示例说明	在user表中将字段中的email字段名字为em

详细示例：

```
mysql> alter table user change email em varchar(60);
Query OK, 0 rows affected (0.38 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

```
mysql> desc user;
+----------+--------------+------+-----+---------+-------+
| Field | Type | Null | Key | Default | Extra |
+----------+--------------+------+-----+---------+-------+
| username | varchar(20) | YES | | NULL | |
| password | varchar(32) | YES | | NULL | |
| createtime | int(10) | YES | | NULL | |
```

```
| createip | int(10) | YES | | NULL | |
| em | varchar(60) | YES | | NULL | |
+----------+--------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

##修改表字段排列顺序

在前的字段增加和修改语句（add/change/modify）中，最后都可以加一个可选项 first|after。

增加表字段时我们已经学过了如何调整顺序。我们现在在来看看另外的change或modify如何来调整顺序。

我们用first做个小实验。

###使用modify调整顺序

```
mysql> alter table user modify em varchar(60) first;
Query OK, 0 rows affected (0.41 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

```
mysql> desc user;
+----------+--------------+------+-----+---------+-------+
| Field | Type | Null | Key | Default | Extra |
+----------+--------------+------+-----+---------+-------+
| em | varchar(60) | YES | | NULL | |
| username | varchar(20) | YES | | NULL | |
| password | varchar(32) | YES | | NULL | |
| createtime | int(10) | YES | | NULL | |
| createip | int(10) | YES | | NULL | |
+----------+--------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

##修改表名

类别	详细解示
基本语法	alter table 旧表名 rename 新的表名;
示例	alter table user rename new_user;
示例说明	将user表名改为new_user

```
mysql> alter table user rename new_user;
Query OK, 0 rows affected (0.35 sec)
```

```
mysql> desc new_user;
+----------+--------------+------+-----+---------+-------+
| Field | Type | Null | Key | Default | Extra |
+----------+--------------+------+-----+---------+-------+
| em | varchar(60) | YES | | NULL | |
| username | varchar(20) | YES | | NULL | |
| password | varchar(32) | YES | | NULL | |
| createtime | int(10) | YES | | NULL | |
| createip | int(10) | YES | | NULL | |
+----------+--------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

07. 类型、字符集、引擎和索引

[7.1 数据类型](7.1 数据类型.md)

[7.2 字符集](7.2 字符集.md)

[7.3 表引擎](7.3 表引擎.md)

[7.4 索引](7.4 索引.md)

7.1 数据类型

MySQL中存的是数据。只要是数据，我们就会规定数据的类型。在表的字段中规定了使用的是某个数据类型。那么，在插入的数据中就要使用对应的数据类型。并且，遵守数据类型的长度要求。
在MySQL里面我们将数据类型分为了以下一些类型：

1. 数值类型（整型、浮点）
2. 字符串类型
3. 日期时间类型
4. 复合类型
5. 空间类型（非科学性工作基本不用，不做讲解）

##整型

MySQL数据类型	所占字节	值范围
tinyint	1字节	-128~127
smallint	2字节	-32768~32767
mediumint	3字节	-8388608~8388607
int	4字节	范围-2147483648~2147483647
bigint	8字节	+-9.22*10的18次方

整型的长度不同，在实际使用过程也就不同。
MySQL 以一个可选的显示宽度指示器的形式对 SQL 标准进行扩展，这样当从数据库检索一个值时，可以把这个值加长到指定的长度。例如，指定一个字段的类型为 INT(6)，
就可以保证所包含数字少于 6 个的值从数据库中检索出来时能够自动地用空格填充。需要注意的是，使用一个宽度指示器不会影响字段的大小和它可以存储的值的范围。
注意：

1. 在创建表字段时，性别我们可以使用无符号的微小整型（tinyint）来表示。用0表示女、用1表示男。用2表示未知。
2. 同样人类年龄也是，在创建表字段时可用用无符号的整型。因为人类的年龄还没有负数
3. 在实际使用过程中。我们业务中最大需要存储多大的数值。我们创建表时，就选择什么样的类型来存储这样的值。

##浮点类型

MySQL数据类型	所占字节	值范围
float(m, d)	4字节	单精度浮点型，m总个数，d小数位
double(m, d)	8字节	双精度浮点型，m总个数，d小数位
decimal(m, d)		decimal是存储为字符串的浮点数

注意：

1. 浮点是非精确值，会存在不太准确的情况
2. 而decimal叫做定点数。在MySQL内部，本质上是用字符串存储的。实际使用过程中如果存在金额、钱精度要求比较高的浮点数存储，建议使用decimal（定点数）这个类型。

##字符类型

MySQL数据类型	所占字节	值范围
CHAR	0-255字节	定长字符串
VARCHAR	0-255字节	变长字符串
TINYBLOB	0-255字节	不超过255个字符的二进制字符串
TINYTEXT	0-255字节	短文本字符串
BLOB	0-65535字节	二进制形式的长文本数据
TEXT	0-65535字节	长文本数据
MEDIUMBLOB	0-16 777 215字节	二进制形式的中等长度文本数据
MEDIUMTEXT	0-16 777 215字节	中等长度文本数据
LOGNGBLOB	0-4 294 967 295字节	二进制形式的极大文本数据
LONGTEXT	0-4 294 967 295字节	极大文本数据
VARBINARY(M)	允许长度0-M个字节的定长字节符串	值的长度+1个字节
BINARY(M)	M	允许长度0-M个字节的定长字节符串

**CHAR **

类型用于定长字符串，并且必须在圆括号内用一个大小修饰符来定义。这个大小修饰符的范围从 0-255。比指定长度大的值将被截短，而比指定长度小的值将会用空格作填补。

**VARCHAR **

把这个大小视为值的大小，在长度不足的情况下就用空格补足。而 VARCHAR 类型把它视为最大值并且只使用存储字符串实际需要的长度

类型不会被空格填补，但长于指示器的值仍然会被截短。

因为 VARCHAR 类型可以根据实际内容动态改变存储值的长度，所以在不能确定字段需要多少字符时使用 VARCHAR 类型可以大大地节约磁盘空间、提高存储效率。

text类型与blob类型

对于字段长度要求超过 255 个的情况下，MySQL 提供了 TEXT 和 BLOB 两种类型。根据存储数据的大小，它们都有不同的子类型。这些大型的数据用于存储文本块或图像、

声音文件等二进制数据类型。

TEXT 和 BLOB 类型在分类和比较上存在区别。BLOB 类型区分大小写，而 TEXT 不区分大小写。大小修饰符不用于各种 BLOB 和 TEXT 子类型。

##时间类型

MySQL数据类型	所占字节	值范围
date	3字节	日期，格式：2014-09-18
time	3字节	时间，格式：08:42:30
datetime	8字节	日期时间，格式：2014-09-18 08:42:30
timestamp	4字节	自动存储记录修改的时间
year	1字节	年份

注意：

1. 时间类型在web系统中用的比较少，很多时候很多人喜欢使用int来存储时间。插入时插入的是unix时间戳，因为这种方式更方便计算。在前端业务中用date类型的函数，再将unix时间戳转成人们可识别的时间。
2. 上面的类型你可以根据实际情况实际进行选择
3. 有些人为了在数据库管理中方便查看，也有人使用datetime类型来存储时间。

##复合类型

MySQL数据类型	说明	举例
set	集合类型	set（“member”，“member2”, … “member64”)
enum	枚举类型	enum（“member1”, “member2”, … “member65535”)

一个 ENUM 类型只允许从一个集合中取得一个值；而 SET 类型允许从一个集合中取得任意多个值。

ENUM 类型

ENUM 类型因为只允许在集合中取得一个值，有点类似于单选项。在处理相互排拆的数据时容易让人理解，比如人类的性别。ENUM 类型字段可以从集合中取得一个值或使用null值，除此之外的输入将会使 MySQL 在这个字段中插入一个空字符串。另外如果插入值的大小写与集合中值的大小写不匹配，MySQL会自动使用插入值的大小写转换成与集合中大小写一致的值。

ENUM 类型在系统内部可以存储为数字，并且从1开始用数字做索引。一个 ENUM 类型最多可以包含 65536 个元素，其中一个元素被 MySQL 保留，用来存储错误信息，这个错误值用索引 0 或者一个空字符串表示。

MySQL 认为 ENUM 类型集合中出现的值是合法输入，除此之外其它任何输入都将失败。这说明通过搜索包含空字符串或对应数字索引为 0 的行就可以很容易地找到错误记录的位置。

SET 类型

SET 类型与 ENUM 类型相似但不相同。SET类型可以从预定义的集合中取得任意数量的值。并且与 ENUM 类型相同的是任何试图在 SET 类型字段中插入非预定义的值都会使MySQL插入一个空字符串。如果插入一个即有合法的元素又有非法的元素的记录，MySQL 将会保留合法的元素，除去非法的元素。

一个 SET 类型最多可以包含 64 个元素。在 SET 元素中值被存储为一个分离的“位”序列，这些“位”表示与它相对应的元素。“位”是创建有序元素集合的一种简单而有效的方式。

并且它还去除了重复的元素，所以SET类型中不可能包含两个相同的元素。

希望从 SET 类型字段中找出非法的记录只需查找包含空字符串或二进制值为 0 的行。

##类型使用

我们学习了这么多类型，在创建表的语句的时候使用对应的类型即可。

举例如下：

```
CREATE TABLE IF NOT EXISTS `demo` (
  `id` int(11) NOT NULL,
  `username` varchar(50) NOT NULL,
  `password` char(32) NOT NULL,
  `content` longtext NOT NULL,
  `createtime` datetime NOT NULL,
  `sex` tinyint(4) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

##字段其他属性设置

UNSIGNED（无符号）

主要用于整型和浮点类型，使用无符号。即，没有前面面的-（负号）。

存储位数更长。tinyint整型的取值区间为，-128~127。而使用无符号后可存储0-255个长度。

创建时在整型或浮点字段语句后接上：

> unsigned

ZEROFILL（0填充）

0（不是空格）可以用来真补输出的值。使用这个修饰符可以阻止 MySQL 数据库存储负值。

创建时在整型或浮点字段语句后接上：

> zerofill

default

default属性确保在没有任何值可用的情况下，赋予某个常量值，这个值必须是常量，因为MySQL不允许插入函数或表达式值。此外，此属性无法用于BLOB或TEXT列。如果已经为此列指定了NULL属性，没有指定默认值时默认值将为NULL，否则默认值将依赖于字段的数据类型。

创建时在整型或浮点字段语句后接上：

> default '值'

not null

如果将一个列定义为not null，将不允许向该列插入null值。建议在重要情况下始终使用not null属性，因为它提供了一个基本验证，确保已经向查询传递了所有必要的值。

创建时在整型或浮点字段语句后接上：

> not null

null

为列指定null属性时，该列可以保持为空，而不论行中其它列是否已经被填充。记住，null精确的说法是“无”，而不是空字符串或0。

创建时在整型或浮点字段语句后不要声明not null即可。

7.2 字符集

##字符集是什么？

为了更好的识别中文、日文、英文、希腊语。对于常用的符号进行了编码，这个编码就是字符集。

字符集确定了文字的存储方式。

字符集相当于是计算机中人类的语言。

举个例子：

我说的是英文，所以我存储的时候要用英文文字来存储。

如果我说的是中文，用英文字符来存储的话。那么人们就看不懂也看不明白，就是我们所说的乱码。

因为字符集太多了，足够有几十种上百种之多。所以我们不需要了解太多的字符集的知识，甚至不需要了解字符集到底是如何编成人类可见字符的。

##字符集的重点知识

我们只需要了解：

1. 常用字符集
2. 数据库中我们用什么字符集

英文字符集：

字符集	说明	字节长度
ASCII	美国标准信息交换代码	单字节
GBK	汉字内码扩展规范	双字节
unicode	万国码	4字节
UTF-8	Unicode的可变长度字符编码	1到6个字节

###ASCII

ASCII 码使用指定的7 位或8 位二进制数组合来表示128 或256 种可能的字符。标准ASCII 码也叫基础ASCII码，使用7 位二进制数来表示所有的大写和小写字母，数字0 到9、标点符号， 以及在美式英语中使用的特殊控制字符。

其中：

0～31及127(共33个)是控制字符或通信专用字符（其余为可显示字符），如控制符：LF（换行）、CR（回车）、FF（换页）、DEL（删除）、BS（退格)、BEL（响铃）等；通信专用字符：SOH（文头）、EOT（文尾）、ACK（确认）等；ASCII值为8、9、10 和13 分别转换为退格、制表、换行和回车字符。它们并没有特定的图形显示，但会依不同的应用程序，而对文本显示有不同的影响。

32～126(共95个)是字符(32是空格），其中48～57为0到9十个阿拉伯数字。

65～90为26个大写英文字母，97～122号为26个小写英文字母，其余为一些标点符号、运算符号等。

###GBK

GBK 向下与 GB 2312 编码兼容。是中华人民共和国定义的汉字计算机编码规范。早期版本为GB2312。

###Unicode

Unicode（统一码、万国码、单一码）Unicode是国际组织制定的可以容纳世界上所有文字和符号的字符编码方案。以满足跨语言、跨平台进行文本转换、处理的要求。

###UTF-8

是一种针对Unicode的可变长度字符编码，也是万国码。因为UNICODE比ASCII占用大一倍的空间，而对ASCII来说高字节的0对他毫无用处。为了解决这个问题，就出现了一些中间格式的字符集，他们被称为通用转换格式，即UTF（Universal Transformation Format）

##实际工作中要使用的编码

在中文中常用的字符集分为utf-8和GBK。

实际使用的如下：

字符集	说明
gbk_chinese_ci	简体中文, 不区分大小写
utf8_general_ci	Unicode (多语言), 不区分大小写

观察（图一）的特点你会发现，MySQL字符集由三个部分组成：

1.字符集

2.语言

3.类型

最后的bin是指二进制字符集，后面的ci是指存储排序时不区分字符的大小写。

> 注意：
> mysql在写utf-8的时候写的是utf8。不加中间的中横线。

（图一）

排序规则	说明
	armscii8 (*ARMSCII-8 Armenian*)
armscii8_bin	亚美尼亚语，二进制
armscii8_general_ci	亚美尼亚语，不区分大小写
	ascii (*US ASCII*)
ascii_bin	西欧 (多语言)，二进制
ascii_general_ci	西欧 (多语言)，不区分大小写
	big5 (*Big5 Traditional Chinese*)
big5_bin	繁体中文，二进制
big5_chinese_ci	繁体中文，不区分大小写
	binary (*Binary pseudo charset*)
binary	二进制
	cp1250 (*Windows Central European*)
cp1250_bin	中欧 (多语言)，二进制
cp1250_croatian_ci	克罗地亚语，不区分大小写
cp1250_czech_cs	捷克语，区分大小写
cp1250_general_ci	中欧 (多语言)，不区分大小写
cp1250_polish_ci	波兰语，不区分大小写
	cp1251 (*Windows Cyrillic*)
cp1251_bin	西里尔语 (多语言)，二进制
cp1251_bulgarian_ci	保加利亚语，不区分大小写
cp1251_general_ci	西里尔语 (多语言)，不区分大小写
cp1251_general_cs	西里尔语 (多语言)，区分大小写
cp1251_ukrainian_ci	乌克兰语，不区分大小写

7.3 表引擎

MySQL的强大之处在于它的插件式存储引擎，我们可以基于表的特点使用不同的存储引擎，从而达到最好的性能。

如果你足够熟悉，并且有一定工作经验后。你还可以使用阿里巴巴和网易开源出来的MySQL引擎在自己的服务器中使用。

大家在后面的一节《数据库结构定义语句》中可以学到创建表的语句。mysql在创建表的时候，可以指定对应的引擎。

在mysql命令中使用：

> show engines;

可以查看到当前服务器支持的所有引擎。

我们介绍几种常用的引擎和了解几个不常用的引擎。避免未来在实际工作中看到一些引擎不知道概念。

引擎名称	特别
MyISAM	常用。读取效率很高的引擎
InnoDB	常用。写入，支持事处等都支持
Archive	不常用。归档引擎，压缩比高达1:10，用于数据归档
NDB	不常用。主要在MySQL 集群服务器中使用，不做介绍

##MyISAM

不支持事务，表锁(表级锁，加锁会锁住整个表)，支持全文索引,操作速度快。常用于读取多的业务。

1. myisam存储引擎表由myd和myi组成。.myd用来存放数据文件，.myi用来存放索引文件。
2. 对于myisam存储引擎表，mysql数据库只缓存其索引文件，数据文件的缓存由操作系统本身来完成。

##InnoDB

1. 支持事务，主要面向在线事务处理(OLTP)方面的应用。
2. 行锁设计，支持外键，即默认情况下读取操作不加锁。

InnoDB是为处理巨大数据量时的最大性能设计。

注：

行锁：写入、更新操作的时候将这一行锁起来，不让其他人再操作了。

表锁：写入、更新操作时，将表给锁起来不让其他人再操作了。

事务：同时操作多个数据，若其中的一个数据操作失败。可回滚到操作之前。常用于银行、电商、金融等系统中。

7.4 索引

##索引优点

索引看着挺高大上的一个名字，说白了就是我们书最新面的目录。

假如你用新华字典来查找“张”这个汉字，不使用目录的话，你可能要从新华字典的第一页找到最后一页，可能要花二个小时。字典越厚呢，你花的时间就越多。现在你使用目录来查找“张”这个汉字，张的首字母是z，z开头的汉字从900多页开始，有了这条线索，你查找一个汉字可能只要一分钟，由此可见索引的重要性。

索引用于快速找出在某个列中有一特定值的行。

不使用索引，MySQL必须从第1条记录开始然后读完整个表直到找出相关的行。表越大，花费的时间越多。如果表中查询的列有一个索引，MySQL能快速到达一个位置去搜寻到数据文件的中间，没有必要看所有数据。

当然索引也不易过多，索引越多写入，修改的速度越慢。因为，写入修改数据时，也要修改索引。

##MySQL的索引类型

索引类型	功能说明
普通索引	最基本的索引，它没有任何限制
唯一索引	某一行企用了唯一索引则不准许这一列的行数据中有重复的值。针对这一列的每一行数据都要求是唯一的
主键索引	它是一种特殊的唯一索引，不允许有空值。一般是在建表的时候同时创建主键索引，常用于用户ID。类似于书中的页码
全文索引	对于需要全局搜索的数据，进行全文索引

注意：以下部分请学习完12.7后再进行学习。

###普通索引

类型	详细说明
基本语法	alter table 表 add index(字段)
示例	ALTER TABLE `money` ADD INDEX(`username`);
示例解释	为money表的username字段增加索引

###唯一索引

类型	详细说明
基本语法	alter table 表 add UNIQUE(字段)

示例	ALTER TABLE `money` ADD UNIQUE(`email`);
示例解释	为money表的email字段增加唯一索引

###全文索引

类型	详细说明
基本语法	alter table 表 add FULLTEXT(字段)
示例	ALTER TABLE `money` ADD FULLTEXT(`content`);
示例解释	为money表的content字段增加唯一索引

###主键索引

类型	详细说明
基本语法	alter table 表 add PRIMARY KEY(字段)
示例	ALTER TABLE `money` ADD PRIMARY KEY(`id`);
示例解释	为money表的id字段增加主键索引

###创建表时也可以声明索引

创建表时可在创建表语句后加上对应的类型即可声明索引：

> PRIMARY KEY(字段)
> INDEX [索引名] (字段)
> FULLTEXT [索引名] (字段)
> UNIQUE[索引名] (字段)

注：中括号中的索引名，代表可选。

整体示例如下：

```
CREATE TABLE `test` (
 `id` INT NOT NULL ,
 `username` VARCHAR(20) NOT NULL ,
 `password` INT NOT NULL ,
 `content` INT NOT NULL ,
```

> PRIMARY KEY (`id`),
> INDEX pw (`password`),
> UNIQUE (`username`),
> FULLTEXT (`content`)
>) ENGINE = InnoDB;

08. 增删改查

我们通过《12.2 娱乐化讲解表关系》。我们知道以下一些结论：

1. 增加装备需要插入
2. 修改装备需要更新
3. 增加金额需要更新
4. 删除装备需要删除
5. 显示余额、装备需要用到查询
6. 显示取钱历史需要用到查询

   … …

我们日常所有的业务建立好表结构后，基本上都可以通过增、删、改、查来实现。

如果上一章的基本语法能背下来最好。实在背不下来以后可以用工具来协助创建表结构会更加方便。

那么，这一章节是重点重的重点，所有语法都要求能够默写。

12.8.1 插入记录

插入记录有两种个基本语法

###插入基本语法一

类别	详细解示
基本语法	insert into 表 values(值1,值2,值n);
示例	insert into user values(2,'李文凯','男')
示例说明	向user表中插入值id为2，姓名为李文凯，性别为男

###插入基本语法二

类别	详细解示
基本语法	insert into 表(字段1,字段2,字段n) values(值1,值2,值n);
示例	insert into user(id,username,sex) values(213,'小沈阳',1);
示例说明	向user表中插入id为213，username为小沈阳，性别为1

####说明

基本语法1和基本语法2的区别是：

1. 基本语法1的插入语句，表中有多少个字段就必须要插入多少个值。一个不能多，一个也不能少。若有默认值，不想传，可以写上null。
2. 基本语法2中，除非有必填字段必须要写入值外。如果有默认值的不想写可以忽略不写。mysql会自动补主默认值。
3. 基本语法2中，以user(id,username,sex)字段顺序为值的顺序。

假设有一张表为user表，我们对字段、字段说明、类型和字段选填和必须状态进行说明，表结构如下：

字段	id	username	email	password	sex
中文说明	编号	用户名	邮箱	密码	性别
类型说明	int	varchar(50)	varchar(60)	varchar(32)	tinyint
默认值说	自增	必填	选填字段，默认值为	选填字段	必填字段

			123@phpxy.com		

##按照基本语法一写上表中的插入语句：

```
insert into user values(null,'李文凯','liwenkai@phpxy.com',null ,1);
```

注意

1. 可以不指定字段名称，但是 values 后面的顺序应该和表字段的排序一致。
2. 有默认值的字段可以不写，则为默认值。
3. 如果有默认值或者可空字段不想传入具体值，可写入null。
4. 数据格式必须要与表规定的数据格式一致。

##按照基本语法二写上表中的插入语句：

```
insert into user(username,sex) values('李文凯',1);
```

注意

1. ID为自增的自段可以不用传入值，每插入一次这个字段的值会自动向上加1。
2. 有默认值和可为空的字段可不传
3. 以表user(username,sex)的插入顺序为准
4. 基本语法二为更常用的用法

##基本语法变形：一次插入多条记录

```
insert into user(username,password,sex)
values('黄晓明', 'abcdef', 1),
( 'angelababy', 'bcdeef', 0),
( '陈赫', '123456', 1),
('王宝强', '987654', 1);
```

12.8.2 查询记录

在讲解查询前，我为大家准备了一个数据表。这个表中存放着银行的余额和用户的基本信息。

我们定义了一个表结构，表名为money。

创建表的语句如下：

```
CREATE TABLE money (
 id INT NOT NULL AUTO_INCREMENT,
 username VARCHAR(50) NOT NULL,
 balance FLOAT NOT NULL,
 province VARCHAR(20) NOT NULL,
 age TINYINT UNSIGNED NOT NULL,
 sex TINYINT NOT NULL,
PRIMARY KEY ( id (10))
) ENGINE = InnoDB CHARACTER SET utf8;
```

表结构和数据展示如下：

id	username	balance	province	age	sex
1	李文凯	120.02	湖北	29	1
2	范冰冰	260.23	山东	40	0
3	黄晓明	150.86	山东	40	1
4	井柏然	810	辽宁	27	1
5	李冰冰	20.15	黑龙江	43	0
6	成龙	313	山东	63	1
7	杨幂	123	北京	30	0
8	刘诗诗	456	北京	29	1
9	柳岩	23.4	湖南	36	0
10	赵本山	3456	辽宁	63	1
11	汪峰	34.32	北京	44	1
12	郭德纲	212	天津	43	1

注：

balance 是指余额

province 是指省份

##基础查询

类别	详细解示
基本语法	select * from 表;
示例	select * from money;
示例说明	查询money表中所有字段中的所有结果

注：” *” 是一种正则表达式的写法，表示匹配所有，上面的查询语句和下面的是等价：

```
mysql> select * from money;
+----+----------+---------+----------+-----+-----+
| id | username | balance | province | age | sex |
+----+----------+---------+----------+-----+-----+
| 1 | 李文凯 | 120.02 | 湖北 | 29 | 1 |
| 2 | 范冰冰 | 260.23 | 山东 | 40 | 0 |
| 3 | 黄晓明 | 150.86 | 山东 | 40 | 1 |
| 4 | 井柏然 | 810 | 辽宁 | 27 | 1 |
| 5 | 李冰冰 | 20.15 | 黑龙江 | 43 | 0 |
| 6 | 成龙 | 313 | 山东 | 63 | 1 |
| 7 | 杨幂 | 123 | 北京 | 30 | 0 |
| 8 | 刘诗诗 | 456 | 北京 | 29 | 1 |
| 9 | 柳岩 | 23.4 | 湖南 | 36 | 0 |
| 10 | 赵本山 | 3456 | 辽宁 | 63 | 1 |
| 11 | 汪峰 | 34.32 | 北京 | 44 | 1 |
| 12 | 郭德纲 | 212 | 天津 | 43 | 1 |
+----+----------+---------+----------+-----+-----+
12 rows in set (0.00 sec)
```

##指定字段查询

类别	详细解示
基本语法	select 字段 from 表;
示例	select id,username, balance from money;
示例说明	查询money表中id,username, balance字段中的所有结果

```
mysql> select id,username, balance from money;
+----+----------+---------+
| id | username | balance |
```

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

| 1 | 李文凯 | 120.02 |

| 2 | 范冰冰 | 260.23 |

| 3 | 黄晓明 | 150.86 |

| 4 | 井柏然 | 810 |

| 5 | 李冰冰 | 20.15 |

| 6 | 成龙 | 313 |

| 7 | 杨幂 | 123 |

| 8 | 刘诗诗 | 456 |

| 9 | 柳岩 | 23.4 |

| 10 | 赵本山 | 3456 |

| 11 | 汪峰 | 34.32 |

| 12 | 郭德纲 | 212 |

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

12 rows in set (0.00 sec)

查询单个字段不重复记录 distinct

类别	详细解示
基本语法	select distinct 字段 from 表;
示例	select distinct age deptno from money;
示例说明	查询money表中年龄唯一的所有结果

mysql> select distinct age deptno from money;

+--------+

| deptno |

+--------+

| 29 |

| 40 |

| 27 |

| 43 |

| 63 |

| 30 |

| 36 |

| 44 |

+--------+

8 rows in set (0.00 sec)

条件查询 where

类别	详细解示
基本语法	select 字段 from 表 where where条件;
示例	select * from money where age = 29;
示例说明	查询money表中年龄为29的所有结果

mysql> select * from money where age = 29;

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

| id | username | balance | province | age | sex |

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

| 1 | 李文凯 | 120.02 | 湖北 | 29 | 1 |

| 8 | 刘诗诗 | 456 | 北京 | 29 | 1 |

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

2 rows in set (0.00 sec)

##where后可接的条件

比较运算符

结果集中将符合条件的记录列出来。上面的例子中，where 后面的条件是一个字段的 ‘=’ 。

除此之外，还可以使用>、<、>=、<=、!=等比较运算符；

符号	说明
>	大于
<	小于
>=	大于等于
<=	小于等于
!=	不等于
=	等于

逻辑运算符

多个条件还可以使用 or 、 and 等逻辑运算符进行多条件联合查询

符号	说明
or	或者

and	并且

我们来看一下多个条件的例子：

类型	详细内容
示例	select * from money where id <10 and
说明	查询所有字段 要求id小于10 并且 province='湖北'

```
mysql> select * from money where id <10 and province='湖北';
+----+----------+---------+----------+-----+-----+
| id | username | balance | province | age | sex |
+----+----------+---------+----------+-----+-----+
| 1 | 李文凯 | 120.02 | 湖北 | 29 | 1 |
+----+----------+---------+----------+-----+-----+
1 row in set (0.00 sec)
```

##结果集排序

类别	详细解示
基本语法	select 字段 from 表 order by 字段 排序关键词
示例	select id,username, balance from money order by balance desc;
示例说明	查询money表中的 id,username,balance字段，按照余额进行降序排序

排序用到的关键词：

关键词	说明
asc	升序排列，从小到大（默认）
desc	降序排列，从大到小

在 select 出来之后的结果集中排序使用 order by ，其中 desc 和 asc 是排序顺序中的关键字。desc 表示按照字段进行降序排列，asc 表示升序排列，如果不写关键字默认升序排列。

```
mysql> select id,username, balance from money order by balance desc;
+----+----------+---------+
| id | username | balance |
```

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

| 10 | 赵本山 | 3456 |

| 4 | 井柏然 | 810 |

| 8 | 刘诗诗 | 456 |

| 6 | 成龙 | 313 |

| 2 | 范冰冰 | 260.23 |

| 12 | 郭德纲 | 212 |

| 3 | 黄晓明 | 150.86 |

| 7 | 杨幂 | 123 |

| 1 | 李文凯 | 120.02 |

| 11 | 汪峰 | 34.32 |

| 9 | 柳岩 | 23.4 |

| 5 | 李冰冰 | 20.15 |

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

12 rows in set (0.00 sec)

##多字段排序

order by 后面可以跟多个不同的字段排序，并且排序字段的不同结果集的顺序也不同，如果排序字段的值一样，则值相同的字段按照第二个排序字段进行排序。

类别	详细解示
基本语法	select 字段 from 表 order by 字段1 排序关键词,... ...字段n desc|asc;
示例	select id,username, balance from money order by balance desc,age asc;
示例说明	查询money表中的 id,username,balance字段，按照余额进行降序排序,若余额全都一样，则再使用age进行升序排序

** 注：如果第一个字段已经将结果给排好。第二个字段排序字段不生效。本例中，第二个字段无效。**

> mysql> select id,username, balance from money order by balance desc,age asc;
> +----+----------+---------+
> | id | username | balance |
> +----+----------+---------+
> | 10 | 赵本山 | 3456 |

> | 4 | 井柏然 | 810 |
> | 8 | 刘诗诗 | 456 |
> | 6 | 成龙 | 313 |
> | 2 | 范冰冰 | 260.23 |
> | 12 | 郭德纲 | 212 |
> | 3 | 黄晓明 | 150.86 |
> | 7 | 杨幂 | 123 |
> | 1 | 李文凯 | 120.02 |
> | 11 | 汪峰 | 34.32 |
> | 9 | 柳岩 | 23.4 |
> | 5 | 李冰冰 | 20.15 |
> +----+-----------+---------+
> 12 rows in set (0.00 sec)

##结果集限制

对于查询或者排序后的结果集，如果希望只显示一部分而不是全部，则可以使用 limit 关键字对结果集进行数量限制。

类别	详细解示
基本语法	select 字段 from 表 limit 数量;
示例	select id,username, balance from money limit 5;
示例说明	显示前五个用户

mysql> select * from money limit 5;

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

| id | username | balance | province | age | sex |

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

| 1 | 李文凯 | 120.02 | 湖北 | 29 | 1 |

| 2 | 范冰冰 | 260.23 | 山东 | 40 | 0 |

| 3 | 黄晓明 | 150.86 | 山东 | 40 | 1 |

| 4 | 井柏然 | 810 | 辽宁 | 27 | 1 |

| 5 | 李冰冰 | 20.15 | 黑龙江 | 43 | 0 |

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

5 rows in set (0.00 sec)

##限制结果集并排序

类别	详细解示
基本语法	select 字段 from 表 order by 字段 关键词 limit 数量
示例	select id,username, balance from money order by balance desc limit 5;
示例说明	按照钱来排序，显示前五个最有钱的用户

mysql> select id,username, balance from money order by balance desc limit 5;

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

| id | username | balance |

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

| 10 | 赵本山 | 3456 |

| 4 | 井柏然 | 810 |

| 8 | 刘诗诗 | 456 |

| 6 | 成龙 | 313 |

| 2 | 范冰冰 | 260.23 |

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

5 rows in set (0.00 sec)

##结果集区间选择

假设我从第0条开始取了3条记录。又想再从第3条开始取3条记录。再想从第6条开始取4条记录怎么办？

这时候就需要使用到结果集区间选择。

类别	详细解示
基本语法	select 字段 from 表 limit 偏移量,数量
示例	select id,username, balance from money limit 0,3;
示例说明	从第一条开始取三条记录

注：第一条记录为0。

> mysql> select id,username, balance from money limit 0,3;
>
> +----+----------+---------+
>
> | id | username | balance |
>
> +----+----------+---------+
>
> | 1 | 李文凯 | 120.02 |
>
> | 2 | 范冰冰 | 260.23 |

```
| 3 | 黄晓明 | 150.86 |
+----+-----------+---------+
3 rows in set (0.00 sec)
```

从第三条开始再取三条呢？

```
mysql> select id,username, balance from money limit 3,3;
+----+-----------+---------+
| id | username | balance |
+----+-----------+---------+
| 4 | 井柏然 | 810 |
| 5 | 李冰冰 | 20.15 |
| 6 | 成龙 | 313 |
+----+-----------+---------+
3 rows in set (0.00 sec)
```

通过上面的这个思路，显示就完成了分页。

每页显示10条记录，那么：

第1页为 limit 0,10

第2页为 limit 10,10

第3页为 limit 20,10

依此类推... ...

##统计类函数使用

1. 如果我们想知道总用户数怎么办？
2. 查询谁是数据表里的首富怎么办？
3. 如果我们想知道用户的平均金额怎么办？
4. 如果我们想知道所有用户的总金额怎么办？

统计类函数最常用的我们有四个：

函数	说明
sum	求和
count	统计总数
max	最大值
min	最小值
avg	平均值

> 注：当然你知道其他的mysql函数也可以使用。不过，在实际工作中，大公司的很多大中型项上很少使用，他们都有专门的计数服务器。因为，mysql的计算量本身很大，为了减少压力通常我们将实际的计算任务交给业务服务器或其他服务器来完成。

类别	详细解示
基本语法	select 函数(字段) from 表
示例	select count(id) from money
示例说明	查询money表的id总数

```
mysql> select count(id) from money;
+-----------+
| count(id) |
+-----------+
| 12 |
+-----------+
1 row in set (0.00 sec)
```

你还可以给字段取别名哟！使用as关键字。

```
mysql> select count(id) as zongshu from money;
+---------+
| zongshu |
+---------+
| 12 |
+---------+
1 row in set (0.00 sec)
```

####查询平均金额

```
mysql> select avg(balance) from money;
+--------------------+
| avg(balance) |
+--------------------+
| 498.24833393096924 |
+--------------------+
1 row in set (0.00 sec)
```

####查询总金额

```
mysql> select sum(balance) from money;
+------------------+
| sum(balance) |
+------------------+
| 5978.980007171631 |
+------------------+
1 row in set (0.00 sec)
```

####查询最大金额

```
mysql> select max(balance) from money;
+--------------+
| max(balance) |
+--------------+
| 3456 |
+--------------+
1 row in set (0.00 sec)
```

####查询最小金额

```
mysql> select min(balance) from money;
+--------------------+
| min(balance) |
+--------------------+
| 20.149999618530273 |
+--------------------+
1 row in set (0.00 sec)
```

##分组 group by

我们拿金额表里面的省份进行分组数据，分组数据后你会发现。有相同的省份会去掉。即，一个省份为一个组。

类别	详细解示
基本语法	select * from 表 group by 字段
示例	select * from money group by province;
示例说明	按照地区进行分组

```
mysql> select * from money group by province;
+----+----------+---------+----------+-----+-----+
| id | username | balance | province | age | sex |
+----+----------+---------+----------+-----+-----+
| 7 | 杨幂 | 123 | 北京 | 30 | 0 |
| 12 | 郭德纲 | 212 | 天津 | 43 | 1 |
| 2 | 范冰冰 | 260.23 | 山东 | 40 | 0 |
| 1 | 李文凯 | 120.02 | 湖北 | 29 | 1 |
| 9 | 柳岩 | 23.4 | 湖南 | 36 | 0 |
| 4 | 井柏然 | 810 | 辽宁 | 27 | 1 |
| 5 | 李冰冰 | 20.15 | 黑龙江 | 43 | 0 |
+----+----------+---------+----------+-----+-----+
```

统计分组（分类）各总数：

```
mysql> select deptno, count(1) from emp group by deptno;
+--------+----------+
| deptno | count(1) |
+--------+----------+
| 1 | 1 |
| 2 | 5 |
| 3 | 1 |
| 5 | 4 |
+--------+----------+
4 rows in set (0.04 sec)
```

####统计省份数量后再进行分组显示

```
mysql> select count(province),province from money group by province;
+-----------------+-----------+
| count(province) | province |
+-----------------+-----------+
| 3 | 北京 |
| 1 | 天津 |
| 3 | 山东 |
| 1 | 湖北 |
| 1 | 湖南 |
| 2 | 辽宁 |
```

```
| 1 | 黑龙江 |
+-----------------+-----------+
7 rows in set (0.00 sec)
```

##在分组基础上进行统计

with rollup用的很少。这个知识点设置为了解级别。

它的主要功能是对于分组的数据进行统计后，再进行一次总数统计。

类别	详细解示
基本语法	select * from 表 group by 字段 with rollup
示例	select count(province),province from money group by province with rollup;
示例说明	对分组的数再次进行统计

在上面的基础上统计总数，下例结果中，最后多了一个12 NULL。

```
mysql> select count(province),province from money group by province with rollup;
+-----------------+-----------+
| count(province) | province |
+-----------------+-----------+
| 3 | 北京 |
| 1 | 天津 |
| 3 | 山东 |
| 1 | 湖北 |
| 1 | 湖南 |
| 2 | 辽宁 |
| 1 | 黑龙江 |
| 12 | NULL |
+-----------------+-----------+
8 rows in set (0.00 sec)
```

##结果再过滤having

having子句与where有相似之处但也有区别,都是设定条件的语句。

having 是筛选组 而where是筛选记录。

类别	详细解示

基本语法	select * from 表 group by 字段 having 条件
示例	select count(province) as result ,province from money group by province having result >2;
示例说明	对地区分组并统计总数，将分组结果中大于2的分组地区显示出来

```
mysql> select count(province) as result ,province from money group by province having result >2;
+--------+----------+
| result | province |
+--------+----------+
| 3 | 北京 |
| 3 | 山东 |
+--------+----------+
2 rows in set (0.00 sec)
```

#整体使用SQL

我们在上面的语句中都是单一使用的某些语句，没有整体使用过。

我们现在将语句进行整合后，配合使用一次。整体的SQL语句配合使用的语法结构如下：

```
SELECT
[字段1 [as 别名1],[函数(字段2) ,].....字段n]
FROM 表名
```

[WHERE where条件]

[GROUP BY 字段]

[HAVING where_contition]

[order 条件]

[limit 条件]

注：上面的语句中可以[] 代表可选。

最终的语法总结如下：

关键词	说明
select	选择的列
from	表
where	查询的条件

group by	分组属性 having 分组过滤的条件
order by	排序属性
limit	起始记录位置，取记录的条数

我们进行一次整体的给合使用，查询money表字段：id,username,balance,province 要求id>1 余额大于50，使用地区进行分组。我们使用用户id进行降序，要求只准显示3条。

最后将SQL语句写成，查询出来的结果如下：

```
mysql> select id,username,balance,province from money where id > 1 and balance > 50 group by province order by id desc limit 3;
+----+----------+---------+----------+
| id | username | balance | province |
+----+----------+---------+----------+
| 12 | 郭德纲 | 212 | 天津 |
| 7 | 杨幂 | 123 | 北京 |
| 4 | 井柏然 | 810 | 辽宁 |
+----+----------+---------+----------+
3 rows in set (0.00 sec)
```

12.8.3 多表联合查询

很多时候在实际的业务中我们不只是查询一张表。

1. 在电子商务系统中，查询哪些用户没有购买过产品。
2. 银行中可能查询违规记录，同时查询出用户的
3. 查询中奖信息和中奖人员的基本信息。

以上只是列的情况我们就需要把两张表在一起进行查询。
而上述业务中需要多表联合在一起查询才能有结果，而多表联合查询的本质是：表连接。
##表连接
当需要查询多个表中的字段时，就可以使用表连接来实现。表联接分为内连接和外连接。

1. 内联结：将两个表中存在联结关系的字段符合联结关系的那些记录形成记录集的联结。
2. 外连接：会选出其他不匹配的记录，分为外左联结和外右联结。

在学习实验前，我为大家准备了两个模拟的数据表：

1. 用户表，存放用户信息
2. 订单表，存放哪个用户购买过哪个商品

user表创建语句

```
CREATE TABLE IF NOT EXISTS `user` (
 `uid` int(11) NOT NULL,
 `username` varchar(30) NOT NULL,
 `password` char(32) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
CREATE TABLE IF NOT EXISTS `order_goods` (
 `oid` int(11) NOT NULL,
 `uid` int(11) NOT NULL,
 `name` varchar(50) NOT NULL,
 `buytime` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

user表数据如下：
| uid | username | password |

--	--	
1	景甜	123456
2	王小二	245667
3	李文凯	1235531
4	井柏然	123455
5	范冰冰	5abcwa
6	黄晓明	abcdeef
7	anglebaby	caption
8	TFBOYS	abcdwww
9	安小超	12tfddwd
10	高小峰	3124qwqw
11	李小强	323fxfvdvd
12	李小超	311aqqee
13	韩小平	121rcfwrfq
14	宋小康	123123tcsd
15	佟小刚	3cxvdfs

order_goods数据如下：

oid	uid	name	buytime
1	10	苹果鼠标	1212313
2	3	iphone 12s	123121241
3	12	雪碧	13232333
4	15		34242123
5	3	iphone 键盘	12123413

注意：

在上表order_goods表中uid是指user表中的uid字段。上表中oid为1的数据行，uid为10的用户。为user表中uid为10的用户：高小峰。该用户购买了商品为苹果鼠标。购买时间buytime为一个unix时间戳。

##内连接

基本语法一：

类别	详细解示
基本语法	select 表1.字段 [as 别名],表n.字段 from 表1 [别名],表n where 条件;
示例	select user.uid ,user.username as username,order_goods.oid,order_goods.uid,order_goods.name as

	shopname from user,order_goods where user.uid = order_goods.uid;
示例说明	查询商品表中哪些用户购买过商品，并将用户信息显示出来

> 注：下例中from 表使用到了表别名。

由于表名太长，每次写的时候容易写错。我们可以在表后直接跟上一个简写英文字符串。在前面拼接字段时，直接使用简写字符串.字段即可。

```
mysql> select u.uid ,u.username as username,o.oid,o.uid,o.name as shopname from user
u,order_goods o where u.uid = o.uid;
+-----+-----------+-----+-----+--------------+
| uid | username | oid | uid | shopname |
+-----+-----------+-----+-----+--------------+
| 10 | 高小峰 | 1 | 10 | 苹果鼠标 |
| 3 | 李文凯 | 2 | 3 | iphone 12s |
| 12 | 李小超 | 3 | 12 | 雪碧 |
| 15 | 佟小刚 | 4 | 15 | |
| 3 | 李文凯 | 5 | 3 | iphone 键盘 |
+-----+-----------+-----+-----+--------------+
5 rows in set (0.00 sec)
```

基本语法二：

类别	详细解示
基本语法	select 表1.字段 [as 别名],表n.字段 from 表1 INNER JOIN 表n on 条件;
示例	select user.uid ,user.username as username,order_goods.oid,order_goods.uid,order_goods.name as shopname from user inner join order_goods on user.uid = order_goods.uid;
示例说明	查询商品表中哪些用户购买过商品，并将用户信息显示出来

结果与基本语法1中一致。

```
mysql> select user.uid ,user.username as username,order_goods.oid,order_goods.uid,order_goods.name as shopname from user inner join order_goods on user.uid = order_goods.uid;
+-----+----------+-----+-----+--------------+
| uid | username | oid | uid | shopname |
+-----+----------+-----+-----+--------------+
| 10 | 高小峰 | 1 | 10 | 苹果鼠标 |
| 3 | 李文凯 | 2 | 3 | iphone 12s |
| 12 | 李小超 | 3 | 12 | 雪碧 |
| 15 | 佟小刚 | 4 | 15 | |
| 3 | 李文凯 | 5 | 3 | iphone 键盘 |
+-----+----------+-----+-----+--------------+
5 rows in set (0.00 sec)
```

##外连接

说明	详解
基本语法	select 表1.字段 [as 别名],表n.字段 from 表1 LEFT JOIN 表n on 条件;
示例	select * from user left join order_goods on user.uid = order_goods.uid;
示例说明	以左边为主，查询哪些用户未购买过商品，并将用户信息显示出来

外连接又分为左连接和右链接，具体定义如下。

左连接：包含所有的左边表中的记录甚至是右边表中没有和它匹配的记录

```
mysql> select * from user left join order_goods on user.uid = order_goods.uid;
+-----+----------+-----------+------+------+--------------+----------+
| uid | username | password | oid | uid | name | buytime |
+-----+----------+-----------+------+------+--------------+----------+
| 10 | 高小峰 | 3124qwqw | 1 | 10 | 苹果鼠标 | 1212313 |
| 3 | 李文凯 | 1235531 | 2 | 3 | iphone 12s | 123121241 |
| 12 | 李小超 | 311aqqee | 3 | 12 | 雪碧 | 13232333 |
| 15 | 佟小刚 | 3cxvdfs | 4 | 15 | | 34242123 |
| 3 | 李文凯 | 1235531 | 5 | 3 | iphone 键盘 | 12123413 |
| 1 | 景甜 | 123456 | NULL | NULL | NULL | NULL |
```

| 2 | 王小二 | 245667 | NULL | NULL | NULL | NULL |

| 4 | 井柏然 | 123455 | NULL | NULL | NULL | NULL |

| 5 | 范冰冰 | 5abcwa | NULL | NULL | NULL | NULL |

| 6 | 黄晓明 | abcdeef | NULL | NULL | NULL | NULL |

| 7 | anglebaby | caption | NULL | NULL | NULL | NULL |

| 8 | TFBOYS | abcdwww | NULL | NULL | NULL | NULL |

| 9 | 安小超 | 12tfddwd | NULL | NULL | NULL | NULL |

| 11 | 李小强 | 323fxfvdvd | NULL | NULL | NULL | NULL |

| 13 | 韩小平 | 121rcfwrfq | NULL | NULL | NULL | NULL |

| 14 | 宋小康 | 123123tcsd | NULL | NULL | NULL | NULL |

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

16 rows in set (0.00 sec)

右连接：包含所有的右边表中的记录甚至是右边表中没有和它匹配的记录

类别	详细解示
基本语法	select 表1.字段 [as 别名],表n.字段 from 表1 right JOIN 表n on 条件;
示例	select * from user right join order_goods on user.uid = order_goods.uid;
示例说明	查询商品表中哪些用户购买过商品，并将用户信息显示出来

mysql> select * from user right join order_goods on user.uid = order_goods.uid;

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

| uid | username | password | oid | uid | name | buytime |

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

| 10 | 高小峰 | 3124qwqw | 1 | 10 | 苹果鼠标 | 1212313 |

| 3 | 李文凯 | 1235531 | 2 | 3 | iphone 12s | 123121241 |

| 12 | 李小超 | 311aqqee | 3 | 12 | 雪碧 | 13232333 |

| 15 | 佟小刚 | 3cxvdfs | 4 | 15 | | 34242123 |

| 3 | 李文凯 | 1235531 | 5 | 3 | iphone 键盘 | 12123413 |

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

5 rows in set (0.00 sec)

##子查询

有时候，当我们查询的时候，需要的条件是另外一个select语句的结果，这时就需要使用子查询。用于子查询的关键字包括in、not in、=、!=、exists、not exists等。

类别	详细解示
基本语法	select 字段 from 表 where 字段 in(条件)
示例1	select * from user where uid in (1,3,4);
示例1说明	按照id 查询指定用户
示例2	select * from user where uid in (select uid from order_goods);
示例2说明	将购买过商品的用户信息显示出来

示例1：

```
mysql> select * from user where uid in (1,3,4);
+-----+-----------+----------+
| uid | username | password |
+-----+-----------+----------+
| 1 | 景甜 | 123456 |
| 3 | 李文凯 | 1235531 |
| 4 | 井柏然 | 123455 |
+-----+-----------+----------+
3 rows in set (0.00 sec)
```

示例2：

```
mysql> select * from user where uid in (select uid from order_goods);
+-----+-----------+----------+
| uid | username | password |
+-----+-----------+----------+
| 10 | 高小峰 | 3124qwqw |
| 3 | 李文凯 | 1235531 |
| 12 | 李小超 | 311aqqee |
| 15 | 佟小刚 | 3cxvdfs |
+-----+-----------+----------+
4 rows in set (0.00 sec)
```

mysql> select * from emp where deptno in (select deptno from dept);

##记录联合

使用 union 和 union all 关键字，将两个表的数据按照一定的查询条件查询出来后，将结果合并到一起显示。两者主要的区别是把结果直接合并在一起，而 union 是将 union all 后的结果进行一次distinct，去除重复记录后的结果。

类别	详细解示
基本语法	select语句1 union[all] select语句2
示例	select * from user where uid in (1,3,4);
示例说明	将商品表中的用户信息和用户表中的用户信息的结果组合在一起

```
mysql> select uid from user union select uid from order_goods;
+-----+
| uid |
+-----+
| 1 |
| 2 |
| 3 |
| 4 |
| 5 |
| 6 |
| 7 |
| 8 |
| 9 |
| 10 |
| 11 |
| 12 |
| 13 |
| 14 |
| 15 |
+-----+
15 rows in set (0.00 sec)
```

12.8.4 更新记录

##更新记录

更新数据我们已经说过。需要修改内容，修改银行卡余额，修改装备信息的时候都需要使用到update，修改语句。

修改(也叫更新)语句的基本语语法如下：

类别	详细解示
基本语法	update 表名 set 字段1=值1,字段2=值2,字段n=值n where 条件
示例	update money set balance=balance-500 where userid = 15;
示例说明	修改money表，将balance余额减500。要求userid为15

假设我们有下面这一个表，表结构如下：

userid	username	balance
1	李文凯	50000.00
2	黄晓明	150000000.00
15	马云	15000.00
16	陈赫	1234131.00

mysql> select * from emp where deptno=15;

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

| userid |username| balance |

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

| 15 | 马云 | 15000.00 |

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

1 row in set (0.00 sec)

使用 update 语句进行记录更新

> mysql> update money set balance=balance-500 where userid = 15;
>
> Query OK, 1 row affected (0.35 sec)
>
> Rows matched: 1 Changed: 1 Warnings: 0

```
mysql> select * from emp where deptno=15;
+------+----------+----------+
| userid |username| balance |
+------+----------+----------+
| 15 | 马云 | 14500.00 |
+------+-------+-------------+
1 row in set (0.00 sec)
```

##修改多个字段

```
mysql> update money set balance=balance-500,username='李文凯' where userid = 15;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1 Changed: 1 Warnings: 0
```

```
mysql> select * from emp where deptno=15;
+------+----------+----------+
| userid |username| balance |
+------+----------+----------+
| 15 |李文凯 | 14500.00 |
+------+-------+-------------+
1 row in set (0.00 sec)
```

##同时对两个表进行更新

类别	详细解示
基本语法	update 表1,表2 set 字段1=值1,字段2=值2,字段n=值n where 条件
示例	update money m,user u m.balance=m.balance*u.age where m.userid=u.id;
示例说明	修改money，将money表的别名设置为m；user表的别名设置为u；将m表的余额改为m表的balance*用户表的age。执行条件是：m.userid = u.id

```
mysql> update money m,user u m.balance=m.balance*u.age where m.userid=u.id;
```

12.8.5 删除记录

使用 delete 删除记录

类别	详细解示
基本语法	delete from 表 [where 条件];
示例	delete from user where id > 10;
示例说明	删除掉用户表中id大于10的所有用户

user表，表结构如下：

id	username	balance
1	李文凯	50000.00
2	黄晓明	150000000.00
15	马云	15000.00
16	陈赫	1234131.00

```
mysql> delete from user where id = 1;
Query OK, 1 row affected (0.08 sec)
```

删除掉了id为1的，李文凯这一行的记录。

##清空表记录

delete和truncate是一样的，但它们有一点不同，那就是DELETE可以返回被删除的记录数，而TRUNCATE TABLE返回的是0。

如果一个表中有自增字段，使用truncate table 这个自增字段将起始值恢复成1.

类别	说明
基本语法	TRUNCATE TABLE 表名;
示例	TRUNCATE TABLE user;
示例说明	清空表的数据，并且让自增的id从1开始自增

【切记】

1. 删除时一定要记住加上where条件，不然会清空掉整个表的记录。
2. 删除重要数据前一定要备份、备份、备份。

09. DCL语句

##创建库用户

###添加权限

类别	详细解示
基本语法	grant 权限 on 库.表 to '用户'@'主机' identified by '密码';
示例	grant select, insert on test.* to 'liwenkai'@'localhost' identified by '4311';
示例说明	给予liwenkai用户，在本机连接test库所有表的权限。操作的这些表具有查询和写入权限

> 注：可以针对一个用户增加多条权限。

###删除权限

类别	详细解示
基本语法	revoke 权限 on 库.表 from '用户'@'主机';
示例	grant select, insert on test.* to 'liwenkai'@'localhost' identified by '4311';
示例说明	给予liwenkai用户，在本机连接test库所有表的权限。操作的这些表具有查询和写入权限

###参数说明

符号	说明
grant all	在grant后接all说明给予所有权限
revoke all	在revoke后接all说明删除所有权限
权限 on .	. 所明给予所有库所有表的操作权限
'用户'@'主机'	主机里面若为%。任意来源的主机均可以使用这个用户来访问

创建数据库用户liwenkai ，具有对test数据库中所有标的 select / insert 权限

示例：增加权限

```
mysql> grant select, insert on test.* to 'liwenkai'@'localhost' identified by '4311';
Query OK, 0 rows affected (0.00 sec)
```

示例：移除权限

```
mysql> revoke insert on test.* from 'liwenkai'@'localhost';
Query OK, 0 rows affected (0.30 sec)
```

注：

上面的一些语句用的较少。你可以将知识点的掌握级别设置为了解级别。

更多的时候，权限设置项特别多，人们往往记不住具体的命令。更多 的时候人们使用专门的工具来操作权限。

10. 常用工具简介

MySQL我们可以使用官方提供的工具和第三方工具来进行管理。工具各有优缺点。

并且工具的使用办法也各有不同，工具的使用，中文、可视化、窗口操作简单。所以我们不会专门花几十页去讲解工具的使用。

有一些复杂的SQL语句完成不用再记忆。例如：权限、建表、备份等。直接使用可视化的工具更加有利于提高工作效率。

我们会为大家推荐最方便使用的几个工具。

常用的工具有：

1. phpMyAdmin(中文，推荐)
2. Navicat(中文，推荐)
3. mysql workbench(英文，官方出品，在设计E-R图时推荐)

##phpMyAdmin

phpMyAdmin 是一个以PHP为基础，以Web-Base方式架构在网站主机上的MySQL的数据库管理工具，让管理者可用Web接口管理MySQL数据库。借由此Web接口可以成为一个简易方式输入繁杂SQL语法的较佳途径，尤其要处理大量资料的汇入及汇出更为方便。其中一个更大的优势在于由于phpMyAdmin跟其他PHP程式一样在网页服务器上执行，但是您可以在任何地方使用这些程式产生的HTML页面，也就是于远端管理MySQL数据库，方便的建立、修改、删除数据库及资料表。也可借由phpMyAdmin建立常用的php语法，方便编写网页时所需要的sql语法正确性。

你的服务器直接php运行环境。下载安装包，解压访问地址即可开始使用。

##Navicat

Navicat提供多达 7 种语言供客户选择，被公认为全球最受欢迎的数据库前端用户界面工具。

它可以用来对本机或远程的 MySQL、SQL Server、SQLite、Oracle 及 PostgreSQL 数据库进行管理及开发。

Navicat的功能足以符合专业开发人员的所有需求，而且对数据库服务器的新手来说又相当容易学习。有了极完备的图形用户界面 (GUI)，Navicat 让你可以以安全且简单的方法创建、组织、访问和共享信息。

Navicat适用于三种平台 - Microsoft Windows、Mac OS X 及Linux。它可以让用户连接到任何本机或远程服务器、提供一些实用的数据库工具如数据模型、数据传输、数据同步、结构同步、导入、导出、备份、还原、报表创建工具及计划以协助管理数据。

##MySQL WorkBench

MySQL Workbench是一款专为MySQL设计的ER/数据库建模工具。它是著名的数据库设计工具DBDesigner4的继任者。你可以用MySQL Workbench设计和创建新的数据库图示，建立数据库文档，以及进行复杂的MySQL迁移。

附录1 . 学习MySQL常用的英文单词

select

update

delete

drop

where

insert

order

by

order by

limit

database

table

function

sql

language

script

API

office

word

html

web

server

windows

back

next

cancel

folder

choose

setup

install

administrator/root/admin

finish

stop
start
config
log/logs
help
quit
module
service
port
Explorer
linu
mac
os
studio
zend studio
eclipse
notepad
note
pad
vim
gvim
down
download
code
info
phpinfo
dollar
var/variable
echo
int
integer
bool
boolearn
string
title

float
double
if
else
null
result
dump
set
unset
object
array
resource
call
back
callback
type
is
get
numeric
mixed
auto
check
define
line
method
class
version
dir
name
space
include
user
my
test
demo

password
text
get
post
submit
value
input
body
address
file
request
fire
fox
bug
action
software
content
home
role
length
protocol
interface
status
time
connection
remote
switch
case
default
break
date
while
go
to
goto

count

table

continue

declare

function

plus

cookie

session

static

match

max

min

rand

year

uninx

timezone

secnods

minutes

hours

day

weekday

month

mirco

first

end

tags

replace

encoding

pop

push

list

each

key

prev

reset

current
sort
regex
read
create
write
move
copy
data
exists
clear
cache
able
lock
seek
close
group
own
owner
path
base
build
parse
discuz
upload
size
limit
memory
enabled
progress
temp
done
error
field
style

png
jpeg/jpg
gif
header
width
height
ascii
display
report
level
notice
warning
all
core
STRICT
DEPRECATED
trigger
mysql
command
monitor
or
oracle
Copyright
engine
index
charset
execute
fetch
row
assoc
db
database
edit
delete
update

alter

modify

change

add

unsigned

ZEROFILL

enum

stamp

union

order

goods

left

right

join

from

inner

outer

shop

cms

system

manger

money

access

agent

token

thread

thread-safe

throw

video