

Table of Contents

数据库简介	1.1
MySQL	1.2
MySQL安装与配置	1.2.1
windows	1.2.1.1
Ubuntu	1.2.1.2
三大范式	1.2.2
基础概念	1.2.3
术语	1.2.3.1
约束	1.2.3.2
字段类型	1.2.3.3
图形化界面	1.2.4
必知命令	1.2.5
连接命令	1.2.5.1
数据库操作	1.2.5.2
表操作	1.2.5.3
数据操作	1.2.5.4
备份与恢复	1.2.5.5
知识进阶	1.2.6
查询	1.2.6.1
比较运算符	1.2.6.1.1
逻辑运算符	1.2.6.1.2
模糊查询	1.2.6.1.3
范围查询	1.2.6.1.4
空判断	1.2.6.1.5
优先级	1.2.6.1.6
聚合	1.2.6.2
分组	1.2.6.3
排序	1.2.6.4
分页	1.2.6.5
Python交互	1.2.7
安装引入	1.2.7.1
Connection对象	1.2.7.2
Cursor对象	1.2.7.3
CURD	1.2.7.4
delete	1.2.7.4.1
create	1.2.7.4.2
Update	1.2.7.4.3
read	1.2.7.4.4
语句参数化	1.2.7.5
封装	1.2.7.6
实例-用户登录	1.2.7.7

数据库简介

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

更简单形象的理解，数据库和我们生活中存放杂物的储物间仓库性质一样，区别只是存放的东西不同，杂物间存放实体的物件，而数据库里存放的是数据。

数据库诞生于距现在大概六十多年前，随着信息技术的发展和人类社会的不断进步，特别是2000年以后，数据库不在仅仅是存储和管理数据了，而转变成用户所需要的各种数据管理方式。数据库有很多种类和功能，从最简单的存储有各种数据的表格到能够进行海量数据存储的大型数据库系统都在各方面得到了广泛的应用。

数据库发展

• 人工管理阶段：

硬件存储设备只有磁带、卡片和纸带

程序员在程序中不仅要规定数据的逻辑结构，还要设计其物理结构，包括存储结构、存取方法、输入输出方式等

• 文件系统阶段

出现了操作系统和高级软件，文件系统是专门管理外存的数据管理软件，操作系统为用户使用文件提供了友好界面。

• 数据库系统阶段

关系型数据库模型是把复杂的数据结构归结为简单的二元关系（即二维表格形式）

采用数据模型表示复杂的数据结构。

数据冗余小，易修改、易扩充

对数据进行统一管理和控制，提供了数据的安全性、完整性、以及并发控制。

程序和数据有较高的独立性。

具有良好的用户接口，用户可方便地开发和使用数据库。

• 数据库最新排名

Rank			DBMS	Database Model	Score		
Aug 2018	Jul 2018	Aug 2017			Aug 2018	Jul 2018	Aug 2017
1.	1.	1.	Oracle	Relational DBMS	1312.02	+34.24	-55.85
2.	2.	2.	MySQL	Relational DBMS	1206.81	+10.74	-133.49
3.	3.	3.	Microsoft SQL Server	Relational DBMS	1072.65	+19.24	-152.82
4.	4.	4.	PostgreSQL	Relational DBMS	417.50	+11.69	+47.74
5.	5.	5.	MongoDB	Document store	350.98	+0.65	+20.48
6.	6.	6.	DB2	Relational DBMS	181.84	-4.36	-15.62
7.	7.	9.	Redis	Key-value store	138.58	-1.34	+16.68
8.	8.	10.	Elasticsearch	Search engine	138.12	+1.90	+20.47
9.	9.	7.	Microsoft Access	Relational DBMS	129.10	-3.48	+2.07
10.	10.	8.	Cassandra	Wide column store	119.58	-1.48	-7.14
11.	11.	11.	SQLite	Relational DBMS	113.73	-1.55	+2.88
12.	12.	12.	Teradata	Relational DBMS	77.41	-0.82	-1.83
13.	13.	16.	Splunk	Search engine	70.49	+1.26	+9.03
14.	14.	18.	MariaDB	Relational DBMS	68.29	+0.78	+13.60
15.	16.	13.	Solr	Search engine	61.90	+0.38	-5.06
16.	15.	14.	SAP Adaptive Server	Relational DBMS	60.44	-1.68	-6.48
17.	17.	15.	HBase	Wide column store	58.80	-1.97	-4.72
18.	18.	20.	Hive	Relational DBMS	57.94	+0.32	+10.64
19.	19.	17.	FileMaker	Relational DBMS	56.05	-0.33	-3.60
20.	20.	19.	SAP HANA	Relational DBMS	51.93	+0.33	+3.96
21.	21.	22.	Amazon DynamoDB	Multi-model	51.66	+2.03	+14.04
22.	22.	21.	Neo4j	Graph DBMS	40.92	-0.95	+2.92
23.	24.	23.	Couchbase	Document store	32.96	-0.12	-0.01
24.	23.	24.	Memcached	Key-value store	32.91	-0.96	+2.95
25.	25.	26.	Microsoft Azure SQL Database	Relational DBMS	26.10	-0.74	+4.19
26.	26.	25.	Informix	Relational DBMS	25.39	-1.20	-2.04
27.	28.	30.	Firebird	Relational DBMS	20.29	-0.37	+2.22
28.	27.	27.	Vertica	Relational DBMS	20.04	-0.78	-1.77
29.	30.	39.	Microsoft Azure Cosmos DB	Multi-model	19.52	+0.07	+10.10
30.	29.	28.	CouchDB	Document store	18.44	-1.06	-2.90

MySql简介

MySQL是一个关系型数据库管理系统，由瑞典MySQL AB 公司开发，目前属于 Oracle 旗下产品。MySQL 是最流行的关系型数据库管理系统之一，在 WEB 应用方面，MySQL是最好的 RDBMS (Relational Database Management System，关系数据库管理系统) 应用软件。

MySQL是一种关系数据库管理系统，关系数据库将数据保存在不同的表中，而不是将所有数据放在一个大仓库内，这样就增加了速度并提高了灵活性。

MySQL所使用的 SQL 语言是用于访问数据库的最常用标准化语言。MySQL 软件采用了双授权政策，分为社区版和商业版，由于其体积小、速度快、总体拥有成本低，尤其是开放源码这一特点，一般中小型网站的开发都选择 MySQL 作为网站数据库。

mysql社区版和企业版区别：

1.企业版只包含稳定之后的功能，社区版包含所有Mysql的最新功能。

也就是说，社区版是企业版的测试版，但是，前者的功能要比后者多。

2.官方的支持服务只针对企业版，用户在使用社区版时出现任何问题，Mysql官方概不负责。

至于管理工具，Mysql官方提供的工具都是免费的，从官方网站都可以下载到，同样可以用在社区版的Mysql上。

MySql的安装配置

Windows安装配置。。。。。

Ubuntu安装配置。。。。。

安装下载

python2.7: 安装mysql模块

```
sudo apt-get install python-mysq
```

安装pip `apt-get install python-pip`

安装python-dev, 否则后面安装mysql模块报错

升级pip

安装libmysqlclient-dev

```
pip install MySQL-python
```

在文件中引入模块

```
import MySQLdb
```

python3.x:

Python3中不支持MySQLdb包, 使用pymysql来代替了, 用法一模一样。

```
pip install pymysql
```

在文件中引入模块:

```
import pymysql
```


数据库三大范式

为了建立冗余较小、结构合理的数据库，设计数据库时必须遵循一定的规则。在关系型数据库中这种规则就称为范式。范式是符合某一种设计要求的总结。要想设计一个结构合理的关系型数据库，必须满足一定的范式。

1. 第一范式(确保每列保持原子性)

第一范式是最基本的范式。如果数据库表中的所有字段值都是不可分解的原子值，就说明该数据库表满足了第一范式。

第一范式的合理遵循需要根据系统的实际需求来定。比如某些数据库系统中需要用到“地址”这个属性，本来直接将“地址”属性设计成一个数据库表的字段就行。但是如果系统经常会访问“地址”属性中的“城市”部分，那么就非要将“地址”这个属性重新拆分为省份、城市、详细地址等多个部分进行存储，这样在对地址中某一部分操作的时候将非常方便。这样设计才算满足了数据库的第一范式

用户信息表							
编号	姓名	性别	年龄	联系电话	省份	城市	详细地址
1	张红欣	男	26	0378-23459876	河南	开封	朝阳区新华路23号
2	李四平	女	32	0751-65432584	广州	广东	白云区天明路148号
3	刘志国	男	21	0371-87659852	河南	郑州	二七区大学路198号
4	郭小明	女	27	0371-62556789	河南	郑州	新郑市薛店北街218号

2. 第二范式(确保表中的每列都和主键相关)

第二范式在第一范式的基础之上更进一层。第二范式需要确保数据库表中的每一列都和主键相关，而不能只与主键的某一部分相关（主要针对联合主键而言）。也就是说在一个数据库表中，一个表中只能保存一种数据，不可以把多种数据保存在同一张数据库表中。

比如要设计一个订单信息表，因为订单中可能会有多种商品，所以要将订单编号和商品编号作为数据库表的联合主键，如下表所示。

订单信息表

订单编号	商品编号	商品名称	数量	单位	价格	客户	所属单位	联系方式
001	1	挖掘机	1	台	1200000¥	张三	上海玖智	020-1234567
001	2	冲击钻	8	把	230¥	张三	上海玖智	020-1234567
002	3	铲车	2	辆	980000¥	李四	北京公司	010-1234567

这样就产生一个问题：这个表中是以订单编号和商品编号作为联合主键。这样在该表中商品名称、单位、商品价格等信息不与该表的主键相关，而仅仅是与商品编号相关。所以在这里违反了第二范式的设计原则。

而如果把这个订单信息表进行拆分，把商品信息分离到另一个表中，把订单项目表也分离到另一个表中，就非常完美了。如下所示。

这样设计，在很大程度上减小了数据库的冗余。如果要获取订单的商品信息，使用商品编号到商品信息表中查询即可。

3. 第三范式(确保每列都和主键列直接相关,而不是间接相关)

第三范式需要确保数据表中的每一列数据都和主键直接相关，而不能间接相关。

比如在设计一个订单数据表的时候，可以将客户编号作为一个外键和订单表建立相应的关系。而不可以再在订单表中添加关于客户其它信息（比如姓名、所属公司等）的字段。如下面这两个表所示的设计就是一个满足第三范式的数据库表。

订单信息表

订单编号	客户	所属单位	联系方式
001	张三	上海玖智	020-1234567
002	李四	北京公司	010-1234567

订单项目表

订单编号	商品编号	数量
001	1	1
001	2	8
002	3	2

商品信息表

商品编号	商品名称	单位	商品价格
1	挖掘机	台	1200000¥
2	冲击钻	个	230¥
3	铲车	辆	980000¥

这样在查询订单信息的时候，就可以使用客户编号来引用客户信息表中的记录，也不必在订单信息表中多次输入客户信息的内容，减小了数据冗余。

基础概念

一个数据库就是一个完整的业务单元，可以包含多张表，数据被存储在表中

在表中为了更加准确的存储数据，保证数据的正确有效，可以在创建表的时候，为表添加一些强制性的验证，包括数据字段的类型、约束

相关术语

- 主键：一个记录中有若干个属性，其中一个能唯一标识该记录，该属性就是主键

比如一条记录包括身份证号，姓名，年龄，身份证号是唯一确定这个人的，它就是主键

- 外键：外键是与另一张表的关联，能确定另一个表中的记录

比如：

有三个表：

客户表：记录客户的信息，如客户编号，客户名称，地址，联系方式等

商品表：记录商品的信息，比如商品编号，商品名称，品牌，单价，库存数量等

订单表：包括订单信息

一条订单记录：包括客户编号，商品编号，商品数量，金额等属性

客户编号是客户表中的主键，它就是订单表的外键

- 约束：一种限制，通过对表的行或列的数据做出限制，来确保数据的完整性、唯一性

比如：在订单记录中，指定的客户编码，必须是客户表中存在的客户

商品编号，必须是商品表中存在的商品

表字段约束

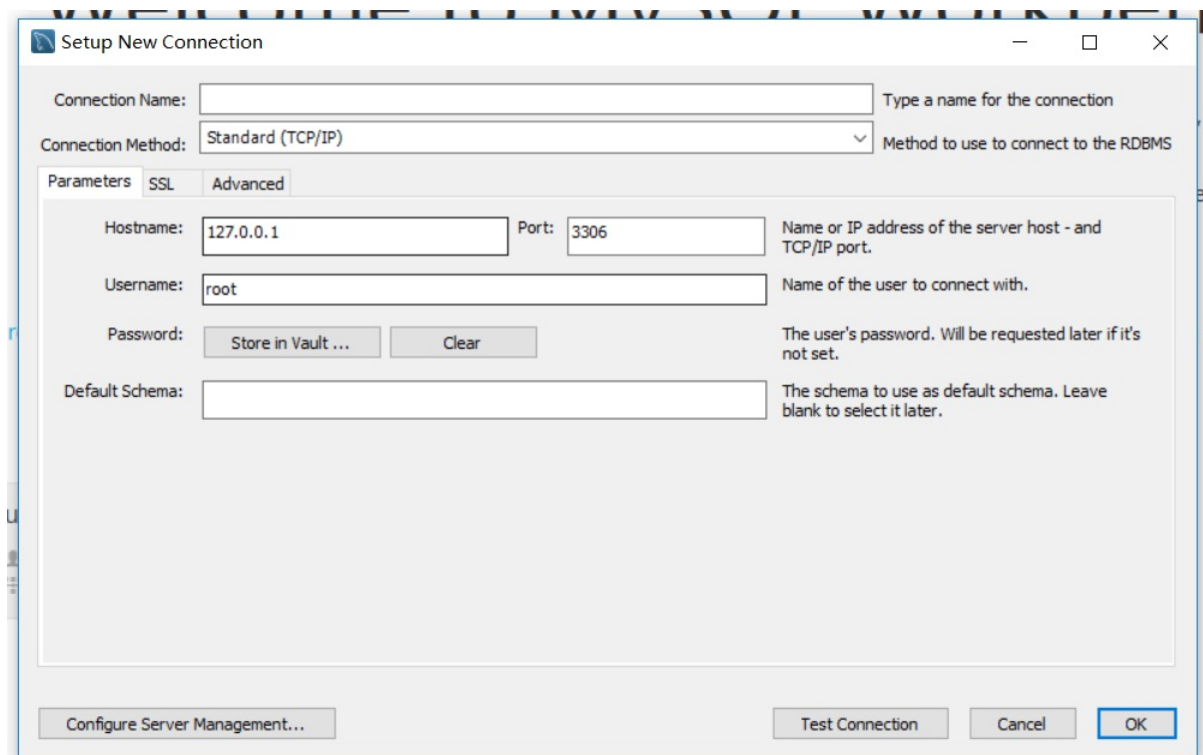
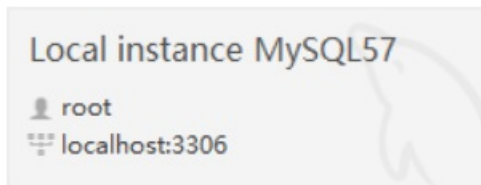
- 主键primary key
- 非空not null
- 惟一unique
- 默认default
- 外键foreign key

表字段类型

- 在mysql中包含的数据类型很多，这里主要列出来常用的几种
- 数字: `int,decimal, float`
- 字符串: `varchar,text`
- 日期: `datetime`
- 布尔: `bit`

MySqlWorkBench图形化界面

MySQL Connections



The "Setup New Connection" dialog box is shown. It has a title bar with standard window controls. The main area contains the following fields and options:

- Connection Name:** A text input field with a placeholder "Type a name for the connection".
- Connection Method:** A dropdown menu currently set to "Standard (TCP/IP)" with a placeholder "Method to use to connect to the RDBMS".
- Parameters tab:** The "Parameters" tab is selected, with "SSL" and "Advanced" tabs also visible.
- Hostname:** A text input field containing "127.0.0.1".
- Port:** A text input field containing "3306". A tooltip indicates: "Name or IP address of the server host - and TCP/IP port."
- Username:** A text input field containing "root". A tooltip indicates: "Name of the user to connect with."
- Password:** A text input field with two buttons: "Store in Vault ..." and "Clear". A tooltip indicates: "The user's password. Will be requested later if it's not set."
- Default Schema:** A text input field. A tooltip indicates: "The schema to use as default schema. Leave blank to select it later."

At the bottom of the dialog, there are four buttons: "Configure Server Management...", "Test Connection", "Cancel", and "OK".

MySQL Workbench

Local instance MySQL57

File Edit View Query Database Server Tools Scripting Help

Navigator

MANAGEMENT

- Server Status
- Client Connections
- Users and Privileges
- Status and System Variables
- Data Export
- Data Import/Restore

INSTANCE

- Startup / Shutdown
- Server Logs
- Options File

PERFORMANCE

- Dashboard
- Performance Reports
- Performance Schema Setup

SCHEMAS

Filter objects

- ludjango
- orgdemo
- sakila
- sys
 - Tables
 - sys_config
 - Views
 - Stored Procedures
 - Functions
- world

sys_config

1 select * from sys.sys_config

Result Grid

variable	value	set_time	set_by
diagnostics.allow i s tables	OFF	2018-06-11 23:03:49	mysql
diagnostics.include raw	OFF	2018-06-11 23:03:49	mysql
os thread tx info.max length	65535	2018-06-11 23:03:49	mysql
statement performance analyzer.limit	100	2018-06-11 23:03:49	mysql
statement performance analyzer.view	mysql	2018-06-11 23:03:49	mysql
statement truncate len	64	2018-06-11 23:03:49	mysql

sys_config 3 x

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	14:51:55	SELECT * FROM ludjango.stu LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec
2	14:52:09	select * from ludjango LIMIT 0, 1000	Error Code: 1146: Table 'ludjango.ludjango' doesn't exist	0.000 sec
3	14:52:22	select * from stu LIMIT 0, 1000	3 row(s) returned	0.000 sec / 0.000 sec
4	14:52:45	select * from sys.sys_config LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec

Information

Schemas: sys

Object Info Session

SQLAdditions

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Form Editor

Field Types

Context Help Snippets

SQL常用命令*****

连接命令

使用命令连接

- 命令操作方式，在工作中使用的更多一些，所以要达到熟练的程度
- 打开终端，运行命令

`mysql -uroot -p`

回车后输入密码，当前设置的密码为mysql

查看版本：`select version();`

显示当前时间：`select now();`

- 注意：在语句结尾要使用分号;
- 退出登录

`quit`或`exit`

数据库操作

- 显示数据库列表

`show databases;`

- 创建数据库

`create database 数据库名 charset=utf8;`

- 删除数据库

`drop database 数据库名;`

- 切换数据库

`use 数据库名;`

- 查看当前选择的数据库

`select database();`

表操作

- 查看当前数据库中所有表

`show tables;`

- 创建表

- `create table` 表名(列及类型);

- 如:

- `create table students(`

```
id int auto\increment primary key,  
  
sname varchar\ (10\ ) not null  
  
\);
```

- 修改表

`alter table` 表名 `add|change|drop` 列名 类型;

`alter table` 表名 `add` 列名 类型; --添加某一列

`alter table` 表名 `change` 原列名 新列名 类型; --修改表的列属性名

`alter table` 表名 `drop` 列名; --删除表的某一列

`alter table` 表名 `rename` 新表名; --修改表名

如:

`alter table students add birthday datetime;`

- 删除表

`drop table` 表名;

- 查看表结构

`desc` 表名;

数据操作

- 查询

`select * from 表名`

- 增加

全列插入: `insert into 表名 values(...)`

缺省插入: `insert into 表名(列1,...) values(值1,...)`

同时插入多条数据: `insert into 表名 values(...),(...)...;`

或`insert into 表名(列1,...) values(值1,...),(值1,...)...;`

- 主键列是自动增长，但是在全列插入时需要占位，通常使用0，插入成功后以实际数据为准

- 修改

`update 表名 set 列1=值1,... where 条件`

- 删除

`delete from 表名 where 条件`

数据备份与恢复

- 进入mysql库目录

- 运行mysqldump命令

`mysqldump -uroot -p 数据库名 > ~/Desktop/备份文件.sql;`

按提示输入mysql的密码

数据恢复

- 连接mysqk，创建数据库

- 退出连接，执行如下命令

`mysql -uroot -p 数据库名 < ~/Desktop/备份文件.sql`

根据提示输入mysql密码

知识进阶

查询

简介

- 查询的基本语法

`select * from 表名;`

- `from`关键字后面写表名，表示数据来源于这张表
- `select`后面写表中的列名，如果是`*`表示在结果中显示表中所有列
- 在`select`后面的列名部分，可以使用`as`为列起别名，这个别名出现在结果集中
- 如果要查询多个列，之间使用逗号分隔

消除重复行

- 在`select`后面列名前使用`distinct`可以消除重复的行

`select distinct gender from students;`

- 使用`where`子句对表中的数据筛选，结果为`true`的行会出现在结果集中
- 语法如下：

`select * from 表名 where 条件;`

比较运算符

- 等于=
- 大于>
- 大于等于>=
- 小于<
- 小于等于<=
- 不等于!=或<>

- 查询编号大于3的学生

```
select * from students where id>3;
```

- 查询编号不大于4的科目

```
select * from subjects where id<=4;
```

- 查询姓名不是“黄蓉”的学生

```
select * from students where sname!='黄蓉';
```

- 查询没被删除的学生

```
select * from students where isdelete=0;
```

逻辑运算符

- and

- or

- not

- 查询编号大于3的女同学

```
select * from students where id>3 and gender=0;
```

- 查询编号小于4或没被删除的学生

```
select * from students where id<4 or isdelete=0;
```

模糊查询

- like

- %表示任意多个任意字符

- _表示一个任意字符

- 查询姓黄的学生

```
select * from students where sname like '黄%';
```

- 查询姓黄并且名字是一个字的学生

```
select * from students where sname like '黄_';
```

- 查询姓黄或叫靖的学生

```
select * from students where sname like '黄%' or sname like '%靖%';
```

范围查询

- in表示在一个非连续的范围内

- 查询编号是1或3或8的学生

```
select * from students where id in(1,3,8);
```

- between ... and ...表示在一个连续的范围内

- 查询学生是3至8的学生

```
select * from students where id between 3 and 8;
```

- 查询学生是3至8的男生

```
select * from students where id between 3 and 8 and gender=1;
```

空判断

- 注意：null与"是不同的

- 判空is null

- 查询没有填写地址的学生

```
select * from students where hometown is null;
```

- 判非空is not null

- 查询填写了地址的学生

```
select * from students where hometown is not null;
```

- 查询填写了地址的女生

```
select * from students where hometown is not null and gender=0;
```

优先级

- 小括号，`not`，比较运算符，逻辑运算符
- `and`比`or`先运算，如果同时出现并希望先算`or`，需要结合`()`使用

聚合

为了快速得到统计数据，提供了5个聚合函数

- **count(*)**表示计算总行数，括号中写星与列名，结果是相同的

查询学生总数

```
select count(*) from students;
```

- **max(列)**表示求此列的最大值

查询女生的编号最大值

```
select max(id) from students where gender=0;
```

- **min(列)**表示求此列的最小值

查询学生最小编号

```
select min(id) from students ;
```

- **sum(列)**表示求此列的和

查询男生的编号之和

```
select sum(id) from students where gender=1;
```

- **avg(列)**表示求此列的平均值

查询未删除女生的编号平均值

```
select avg(id) from students where isdelete=0 and gender=0;
```

分组

按照字段分组，表示此字段相同的数据会被放到一个组中

分组后，只能查询出相同的数据列，对于有差异的数据列无法出现在结果集中

可以对分组后的数据进行统计，做聚合运算

语法：

```
select 列1,列2,聚合... from 表名 group by 列1,列2,列3...
```

查询男女生总数

```
select gender as 性别,count(*)
```

```
from students
```

```
group by gender;
```

查询各城市人数

```
select hometown as 家乡,count(*)
```

```
from students
```

```
group by hometown;
```

分组后的数据筛选

- 语法：

```
select 列1,列2,聚合... from 表名
```

```
group by 列1,列2,列3...
```

```
having 列1,...聚合...
```

- **having**后面的条件运算符与**where**的相同

- 查询男生总人数

```
select count(*)
```

```
from students
```

```
where gender=1;
```


排序

为了方便查看数据，可以对数据进行排序

语法：

select * from 表名

order by 列1 asc|desc,列2 asc|desc,...

将行数据按照列1进行排序，如果某些行列1的值相同时，则按照列2排序，以此类推

默认按照列值从小到大排列

asc从小到大排列，即升序

desc从大到小排序，即降序

查询男生学生信息，按学号降序

select * from students

where gender=1

order by id desc;

查询未删除科目信息，按名称升序

select * from subject

order by stitle;

分页

获取部分行

当数据量过大时，在一页中查看数据是一件非常麻烦的事情

语法

`select * from 表名`

`limit start,count`

从`start`开始，获取`count`条数据

`start`索引从0开始

示例：分页

已知：每页显示`m`条数据，当前显示第`n`页

求第`n`页的数据

`select * from students`

`limit (n-1)*m,m`

安装引入

python2.7: 安装mysql模块

```
sudo apt-get install python-mysql
```

安装pip apt-get install python-pip

安装python-dev, 否则后面安装mysql模块报错

升级pip

安装:libmysqlclient-dev

```
pip install MySQL-python
```

在文件中引入模块

```
import Mysqldb
```

python3.x:

Python3中不支持Mysqldb包, 使用pymysql来代替了, 用法一模一样。

```
pip install pymysql
```

在文件中引入模块:

```
import pymysql
```

Connection对象

Connection对象

用于建立与数据库的连接

创建对象：调用connect()方法

conn=connect(参数列表)

参数host：连接的mysql主机，如果本机是'localhost'

参数port：连接的mysql主机的端口，默认是3306

参数db：数据库的名称

参数user：连接的用户名

参数password：连接的密码

参数charset：通信采用的编码方式，默认是'gb2312'，要求与数据库创建时指定的编码一致，否则中文会乱码

对象的方法

close()关闭连接

commit()事务，所以需要提交才会生效

rollback()事务，放弃之前的操作

cursor()返回Cursor对象，用于执行sql语句并获得结果

Cursor对象

执行sql语句

创建对象：调用Connection对象的cursor()方法

```
cursor1=conn.cursor()
```

对象的方法

close()关闭

execute(operation [, parameters])执行语句，返回受影响的行数

fetchone()执行查询语句时，获取查询结果集的第一个行数据，返回一个元组

next() 执行查询语句时，获取当前行的下一行()

fetchall()执行查询时，获取结果集的所有行，一行构成一个元组，再将这些元组装入一个元组返回

scroll(value[,mode])将行指针移动到某个位置

mode表示移动的方式

mode的默认值为relative，表示基于当前行移动到value，value为正则向下移动，value为负则向上移动

mode的值为absolute，表示基于第一条数据的位置，第一条数据的位置为0

对象的属性

rowcount只读属性，表示最近一次execute()执行后受影响的行数

connection获得当前连接对象

CURD

delete

创建testDelete.py文件，删除学生表的一条数据

```
#encoding=utf-8
```

```
import MySQLdb
```

```
try:
```

```
    conn=pymysql.connect\((host='localhost',port=3306,db='test1',user='root',passwd='mysql',charset='utf8'\)

    cs1=conn.cursor\(\)

    count=cs1.execute\("delete from students where id=6"\)

    print count

    conn.commit\(\)

    cs1.close\(\)

    conn.close\(\)
```

```
except Exception,e:
```

```
    print e.message
```

Create

创建testInsert.py文件，向学生表中插入一条数据

```
#encoding=utf-8
```

```
import pymysql
```

```
try:
```

```
    conn=pymysql.connect\((host='localhost',port=3306,db='test1',user='root',passwd='mysql',charset='utf8'\))

    cs1=conn.cursor\(\)

    count=cs1.execute\("insert into students\((sname\) values\('张良'\)"\)

    print count

    conn.commit\(\)

    cs1.close\(\)

    conn.close\(\)
```

```
except Exception,e:
```

```
    print e.message
```


Update

创建testUpdate.py文件，修改学生表的一条数据

```
#encoding=utf-8
```

```
import MySQLdb
```

```
try:
```

```
conn=pymysql.connect\((host='localhost',port=3306,db='test1',user='root',passwd='mysql',charset='utf8'\)

cs1=conn.cursor\(\)

count=cs1.execute\("update students set sname='刘邦' where id=6"\)

print count

conn.commit\(\)

cs1.close\(\)

conn.close\(\)
```

```
except Exception,e:
```

```
print e.message
```

Read

创建testSelectOne.py文件，查询一条学生信息

```
#encoding=utf8
```

```
import pymysql
```

```
try:
```

```
    conn=pymysql.connect\((host='localhost',port=3306,db='test1',user='root',passwd='mysql',charset='utf8'\)

    cur=conn.cursor\(\)

    cur.execute\('select \* from students where id=7'\)

    result=cur.fetchone\(\)
    result=cur.fetchall()

    print\((result\)

    cur.close\(\)

    conn.close\(\)
```

```
except Exception as e:
```

```
    print\((e.message\)
```

语句参数化

创建testInsertParam.py文件，向学生表中插入一条数据

```
#encoding=utf-8
```

```
import pymysql
```

```
try:
```

```
    conn=pymysql.connect\((host='localhost',port=3306,db='test1',user='root',passwd='mysql',charset='utf8'\)

    cs1=conn.cursor\(\)

    sname=raw_input\("请输入学生姓名: "\)

    params=\[sname\]

    count=cs1.execute\('insert into students\((sname\) values\((%s\)'\),params\)

    print count

    conn.commit\(\)

    cs1.close\(\)

    conn.close\(\)
```

```
except Exception as e:
```

```
    print\((e.message\)
```

封装

定义类

```
#encoding=utf8
```

```
import pymysql
```

```
class MysqlHelper():
```

```
    def __init__(self,host,port,db,user,passwd,charset='utf8'):

        self.host=host

        self.port=port

        self.db=db

        self.user=user

        self.passwd=passwd

        self.charset=charset


    def connect(self):

        self.conn=pymysql.connect(host=self.host,port=self.port,db=self.db,user=self.user,passwd=self.passwd,charset=self.charset)

        self.cursor=self.conn.cursor()


    def close(self):

        self.cursor.close()

        self.conn.close()


    def get_one(self,sql,params=()):

        result=None

        try:

            self.connect()

            self.cursor.execute(sql, params)

            result = self.cursor.fetchone()

            self.close()

        except Exception as e:

            print(e.message)

        return result


    def get_all(self,sql,params=()):

        list=()

        try:

            self.connect()

            self.cursor.execute(sql,params)

            list=self.cursor.fetchall()

            self.close()

        except Exception as e:
```

```

        print\(\e.message\)

    return list

def insert\(\self,sql,params=\(\)\):

    return self._\_\_edit\(\sql,params\)

def update\(\self, sql, params=\(\)\):

    return self._\_\_edit\(\sql, params\)

def delete\(\self, sql, params=\(\)\):

    return self._\_\_edit\(\sql, params\)

def _\_\_edit\(\self,sql,params\):

    count=0

    try:

        self.connect\(\)

        count=self.cursor.execute\(\sql,params\)

        self.conn.commit\(\)

        self.close\(\)

    except Exception as e:

        print\(\e.message\)

    return count

```

创建用户表

创建用户表userinfos

表结构如下

id

uname

upwd

isdelete

注意：需要对密码进行加密

如果使用md5加密，则密码包含32个字符

如果使用sha1加密，则密码包含40个字符，推荐使用这种方式

```
create table userinfos(  
id int primary key auto_increment,  
uname varchar(20),  
upwd char(40),  
isdelete bit default 0  
);
```

加入数据

加入测试数据

插入如下数据，用户名为123,密码为123,这是sha1加密后的值

```
insert into userinfos values(0,'123','40bd001563085fc35165329ea1ffa5ec9d13bb5',0);
```

接受输入并验证

接收输入并验证

创建testLogin.py文件，引入hashlib模块、MysqlHelper模块

接收输入

根据用户名查询，如果未查到则提示用户名不存在

如果查到则匹配密码是否相等，如果相等则提示登录成功

如果不相等则提示密码错误

```
#encoding=utf-8  
  
from MysqlHelper import MysqlHelper  
  
from hashlib import sha1  
  
sname=input("请输入用户名: ")  
spwd=input("请输入密码:")  
s1=sha1()  
s1.update(spwd)  
spwdSha1=s1.hexdigest()  
  
sql="select upwd from userinfos where uname=%s"  
params=[sname]
```

```
sqlhelper=MysqlHelper('localhost',3306,'test1','root','mysql')
```

```
userinfo=sqlhelper.get_one(sql,params)
```

```
if userinfo==None:
```

```
    print\('用户名错误'\)
```

```
elif userinfo[0]==spwdSha1:
```

```
    print\('登录成功'\)
```

```
else:
```

```
    print\('密码错误'\)
```