# **MySQL数据库**

| **Tedu Python 教学部** |
| --- |
| Author：吕泽 |
| Days：2天 |

## **数据库概述**

### **数据存储阶段**

【1】 人工管理阶段

缺点 ： 数据无法共享,不能单独保持,数据存储量有限

【2】 文件管理阶段 （.txt .doc .xls）

优点 ： 数据可以长期保存,可以存储大量的数据,使用简单

缺点 ： 数据一致性差,数据查找修改不方便,数据冗余度可能比较大

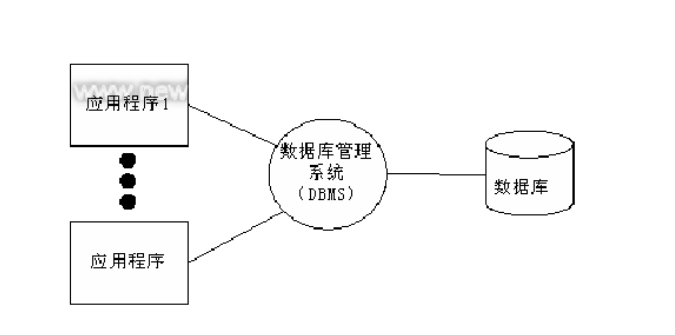
【3】数据库管理阶段

优点 ： 数据组织结构化降低了冗余度,提高了增删改查的效率,容易扩展,方便程序调用，做自动化处理

缺点 ：需要使用sql 或者 其他特定的语句，相对比较复杂

### **数据库应用**

融机构、游戏网站、购物网站、论坛网站 ... ...



### **基础概念**

数据 ： 能够输入到计算机中并被识别处理的信息集合

数据结构 ：研究一个数据集合中数据之间关系的

数据库 ： 按照数据结构，存储管理数据的**仓库**。数据库是在数据库管理系统管理和控制下，在一定介质上的数据集合。

数据库管理系统 ：管理数据库的软件，用于建立和维护数据库 （图数据库系统）

数据库系统 ： 由数据库和数据库管理系统，开发工具等组成的集合

### **数据库分类和常见数据库**

* 关系型数据库和非关系型数据库

关系型： 采用关系模型来组织数据结构的数据库 （二维表）

非关系型： 不采用关系模型组织数据结构的数据库

* 开源数据库和非开源数据库

开源：MySQL、SQLite、MongoDB

非开源：Oracle、DB2、SQL\_Server

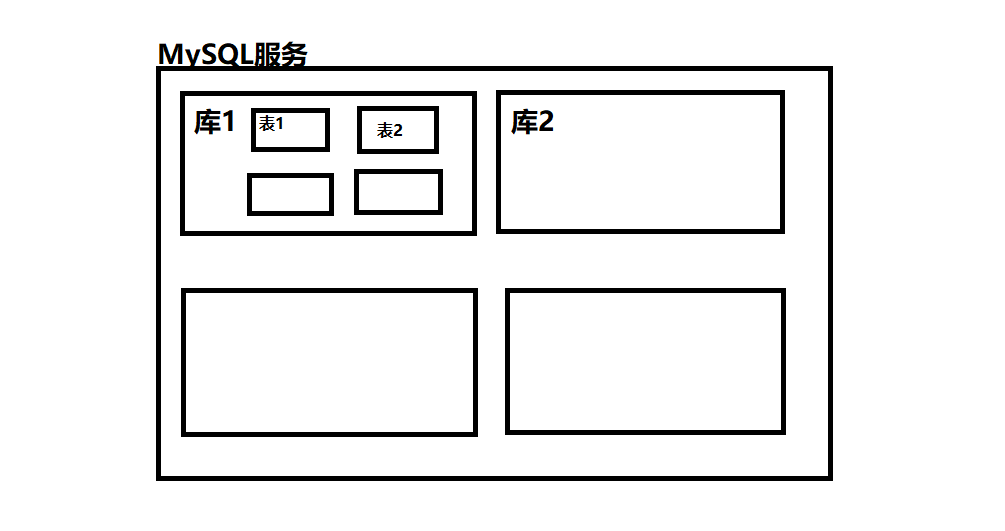
* 常见的关系型数据库

MySQL、Oracle、SQL\_Server、DB2 SQLite

### **认识关系型数据库和MySQL**

1. 数据库结构 （图库结构）

数据元素 --> 记录 -->数据表 --> 数据库

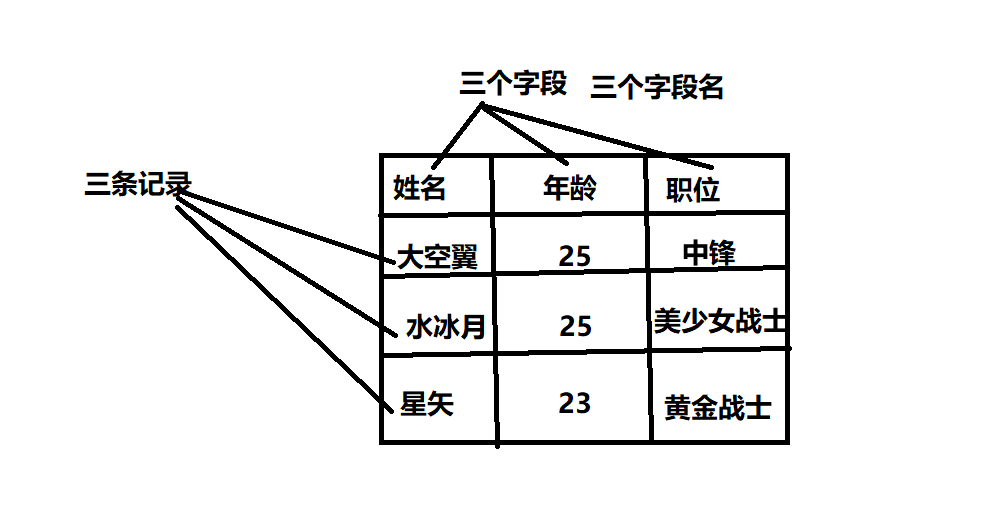


1. 数据库概念解析

数据表 ： 存放数据的表格 （图表结构）

字段： 每个列，用来表示该列数据的含义

记录： 每个行，表示一组完整的数据



1. MySQL特点

* 是开源数据库，使用C和C++编写
* 能够工作在众多不同的平台上
* 提供了用于C、C++、Python、Java、Perl、PHP、Ruby众多语言的API
* 存储结构优良，运行速度快
* 功能全面丰富

1. MySQL安装

Ubuntu安装MySQL服务

安装服务端: sudo apt-get install mysql-server  
安装客户端: sudo apt-get install mysql-client

配置文件：/etc/mysql  
命令集： /usr/bin  
数据库存储目录 ：/var/lib/mysql

Windows安装MySQL

下载MySQL安装包(windows) <https://dev.mysql.com/downloads/mysql/>  
mysql-installer\*\*\*5.7.\*\*\*.msi

安装教程去安装

1. 启动和连接MySQL服务

**服务端启动**

查看MySQL状态: sudo /etc/init.d/mysql status  
**启动/停止/重启/服务**：sudo /etc/init.d/mysql start | stop | restart

**客户端连接**

命令格式

mysql -h主机地址 -u用户名 -p密码  
mysql -hlocalhost -uroot -p123456  
**本地连接**可省略 -h 选项: mysql -uroot -p123456

## **SQL语句**

什么是SQL

结构化查询语言(Structured Query Language)，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。

SQL语句使用**特点**

* SQL语言基本上独立于数据库本身
* 各种不同的数据库对SQL语言的支持与标准存在着细微的不同
* 每条命令必须以 **;** 结尾
* SQL命令关键字不区分字母大小写

## **MySQL 数据库操作**

### **数据库操作**

#### **1.查看已有库**

**show databases;**

#### **2.创建库(指定字符集)**

**create database 库名 [character set utf8];**

e.g. 创建stu数据库，设置**默认编码格式**为utf8（这样内容可以插入中文）

**create database stu charset=utf8;**

#### **3.查看创建库的语句(字符集)**

**show create database 库名;**

e.g. 查看stu创建方法

show create database stu;

#### 4.查看当前所在库（当前使用的库）

**select database();**

#### **5.切换库**

**use 库名;**

e.g. 使用stu数据库

use stu;

#### 6.删除库

**drop database 库名;**

e.g. 删除test数据库

drop database test;

#### **7.库名的命名规则**

* 数字、字母、下划线,但不能使用纯数字
* 库名区分字母大小写
* 不能使用特殊字符和mysql关键字

### **数据表的管理**

#### １．表结构设计初步

【1】 确定存储内容  
【2】 确定字段构成  
【3】 确定字段类型

#### ２．数据类型支持

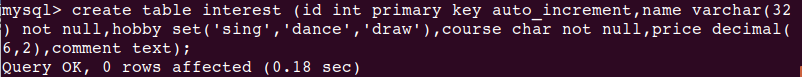
##### 整型：

整数类型（精确值） - INTEGER，INT，SMALLINT，TINYINT，MEDIUMINT，BIGINT  
定点类型（精确值） - DECIMAL  
浮点类型（近似值） - FLOAT，DOUBLE  
比特值类型 - BIT



对于精度比较高的东西，比如money，用decimal类型**提高精度减少误差**。列的声明语法是**DECIMAL(M,D)**。

**M**是数字的最大位数（精度）。其范围为1～65，M 的默认值是10。  
**D**是小数点右侧数字的数目（标度）。其范围是0～30，但不得超过M。  
比如 **DECIMAL(6,2)**最多存6位数字，小数点后占2位,取值范围-9999.99到9999.99。



比特值类型指0，1值表达2种情况，如真，假。

如果类型需要无符号（负数），在后面加上**unsigned**即可



##### 字符串类型：

CHAR和VARCHAR类型  
BINARY和VARBINARY类型  
BLOB和TEXT类型  
ENUM类型和SET类型



* **char** 和 **varchar**

char：定长，效率高，一般用于固定长度的表单提交数据存储，默认1字符  
varchar：不定长（指定一个最大长度空间，必须写一个范围），效率偏低

* **text** 和**blob**

text用来存储非二进制文本  
blob用来存储二进制字节串（图片，音频等）

* **enum** 和 **set**

enum用来存储给出的**一个值**

set用来存储给出的值中**一个或多个值**

##### **时间日期类型:**

DATE，DATETIME和TIMESTAMP类型  
TIME类型  
年份类型YEAR



#### ３．表的基本操作

##### **创建表(指定字符集)**

**create table 表名(  
字段名 数据类型,  
字段名 数据类型,  
...  
字段名 数据类型  
);**

* 如果你不想字段为 NULL 可以设置字段的属性为 NOT NULL， 在操作数据库时如果输入该字段的数据为NULL ，就会报错。
* **DEFAULT** 表示设置一个字段的默认值
* **AUTO\_INCREMENT**定义列为**自增的属性**，一般用于主键，数值会自动加1。
* **PRIMARY KEY**关键字用于定义列为**主键**。主键的值不能重复。

e.g. 创建班级表

create table class\_1 (id int **primary key auto\_increment**,name varchar(32) not null,age int not null,sex enum('w','m'),score float default 0.0);

e.g. 创建兴趣班表

create table interest (id int primary key auto\_increment,name varchar(32) not null,hobby set('sing','dance','draw'),course char not null,price decimal(6,2),comment text);

##### 查看数据表

show tables；

##### 查看已有表的字符集

show create table 表名;

##### 查看表结构（表的类型）

desc 表名;

##### 删除表

drop table 表名;

## **表数据的基本操作**

### **插入(insert)**

**方法一：insert into 表名 values(值1),(值2),...;**

**方法二：insert into 表名(字段1,...) values(值1),...;**

e.g.

insert into class\_1 values (2,'Baron',10,'m',91),(3,'Jame',9,'m',90);

### **查询(select)**

**方法一：select \* from 表名 [where 条件];**

**方法二：select 字段1,字段名2 from 表名 [where 条件];**

[]表示可写可不写

e.g.

select \* from class\_1;

select name,age from class\_1;

### **where子句**

where子句在sql语句中扮演了重要角色，主要通过一定的**运算条件**进行**数据的筛选**

MySQL 主要有以下几种运算符：

算术运算符  
比较运算符  
逻辑运算符  
位运算符

#### **１．算数运算符**



e.g.

select \* from class\_1 where age % 2 = 0;

#### **２．比较运算符**



e.g.

select \* from class\_1 where age > 8;

select \* from class\_1 where age between 8 and 10;

select \* from class\_1 where age not between 8 and 10;

select \* from class\_1 where age in (8,9);

select \* from class\_1 where age not in (8,9);

#### **３．逻辑运算符**

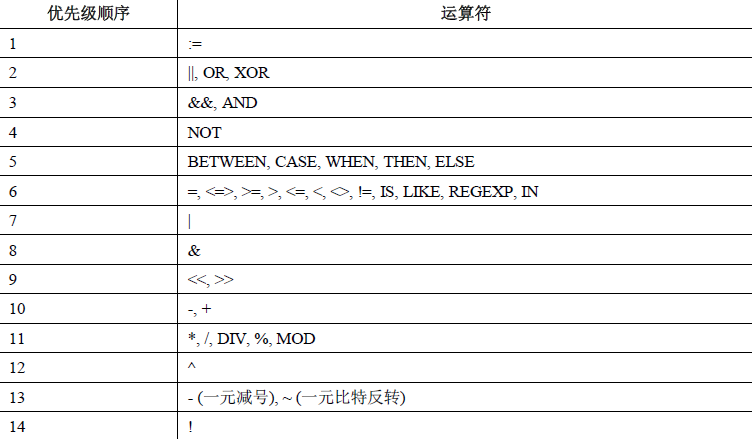


e.g.

select \* from class\_1 where sex='m' and age>9;

#### **４．位运算符**





### **更新表记录(update)**

**update 表名 set 字段1=值1,字段2=值2,... where 条件;**

e.g.

update class\_1 set age=11 where name='Abby';

### **删除表记录（delete）**

**delete from 表名 where 条件;**

**注意:**delete语句后如果不加where条件,**所有记录全部清空**

e.g.

delete from class\_1 where name='Abby';

## **表字段的操作(alter)**

语法 ：alter table 表名 执行动作;

### **１．添加字段(add)**

**方法一：alter table 表名 add 字段名 数据类型;**

**方法一：alter table 表名 add 字段名 数据类型 first;**

**方法一：alter table 表名 add 字段名 数据类型 after 字段名;**

### **２．删除字段(drop)**

**alter table 表名 drop 字段名;**

### **３．修改数据类型(modify)**

**alter table 表名 modify 字段名 新数据类型;**

**注意：修改字段数据类型需要保证该字段没有数据，否则会修改失败**

### **４．重命名字段(change)**

**alter table 表名 change 旧字段名 新字段名　类型;**

### **５．表重命名(rename)**

**alter table 表名 rename 新表名;**

e.g.

alter table interest add date Date after course;

### **时间类型数据**

#### **时间格式**

**date** ："YYYY-MM-DD"  
**time** ："HH:MM:SS"  
**datetime** ："YYYY-MM-DD HH:MM:SS"  
**timestamp** ："YYYY-MM-DD HH:MM:SS"  
注意  
1、datetime ：不给值默认返回NULL值  
2、timestamp ：不给值默认返回系统当前时间

#### **日期时间函数**

* now() 返回服务器当前时间
* curdate() 返回当前日期
* curtime() 返回当前时间
* year(date) 返回指定时间的年份
* date(date) 返回指定时间的日期
* time(date) 返回指定时间的时间

#### **时间操作**

* 查找操作

select \* from timelog where Date = "2018-07-02";

select \* from timelog where Date>="2018-07-01" and Date<="2018-07-31";

* 日期时间运算
  + 语法格式

**select \* from 表名 where 字段名 运算符 (时间-interval 时间间隔单位);**

* + 时间间隔单位： 1 day | 2 hour | 1 minute | 2 year | 3 month

select \* from timelog where shijian > (now()-interval 1 day);

### **高级查询语句**

#### 模糊查询和正则查询

LIKE用于在where子句中进行模糊查询，SQL LIKE 子句中使用百分号 %字符来表示任意字符。

使用 LIKE 子句从数据表中读取数据的通用语法：

SELECT field1, field2,...fieldN

FROM table\_name

WHERE field1 **LIKE** condition1

e.g.

mysql> select \* from class\_1 where name like 'A%';

mysql中对**正则表达式**的支持有限，只支持部分正则元字符

SELECT field1, field2,...fieldN

FROM table\_name

WHERE field1 **REGEXP** condition1

e.g.

select \* from class\_1 where name regexp 'B.+';

#### **排序**

ORDER BY 子句来设定你想按哪个字段哪种方式来进行排序，再返回搜索结果。

使用 ORDER BY 子句将查询数据排序后再返回数据：

SELECT field1, field2,...fieldN from table\_name1 where field1

**ORDER BY** field1 [ASC [DESC]]

默认情况ASC表示**升序**，DESC表示**降序**

select \* from class\_1 where sex='m' order by age;

#### **分页**

**LIMIT** 子句用于限制由 SELECT 语句返回的数据数量 或者 UPDATE,DELETE语句的操作数量

带有 LIMIT 子句的 SELECT 语句的基本语法如下：

SELECT column1, column2, columnN

FROM table\_name

WHERE field

**LIMIT** [num]

#### **联合查询**

UNION 操作符用于连接两个以上的 SELECT 语句的结果组合到一个结果集合中。多个 SELECT 语句会删除重复的数据。

UNION 操作符语法格式：

SELECT expression1, expression2, ... expression\_n

FROM tables

[WHERE conditions]

**UNION [ALL | DISTINCT]**

SELECT expression1, expression2, ... expression\_n

FROM tables

[WHERE conditions];

expression1, expression2, ... expression\_n: 要检索的列。  
tables: 要检索的数据表。  
WHERE conditions: 可选， 检索条件。  
DISTINCT: 可选，删除结果集中重复的数据。默认情况下 UNION 操作符已经删除了重复数据，所以 DISTINCT 修饰符对结果没啥影响。  
ALL: 可选，返回所有结果集，**包含重复数据**。  
前后的两个select要求**查询的字段**必须相同，否则会报错（查询的字段显示不了）

select \* from class\_1 where sex='m' **UNION ALL** select \* from class\_1 where age > 9;

#### **多表查询**

多个表数据可以联合查询，语法格式如下

select 字段1,字段2... from 表1,表2... [where 条件]

e.g.

select class\_1.name,class\_1.age,class\_1.sex,interest.hobby from class\_1,interest where class\_1.name = interest.name;

## 数据备份

### １．备份命令格式

mysqldump -u用户名 -p 源库名 > ~/\*\*\*.sql

> :导出

--all-databases 备份所有库  
库名 备份单个库  
-B 库1 库2 库3 备份多个库  
库名 表1 表2 表3 备份指定库的多张表



### ２．恢复命令格式

mysql -uroot -p 目标库名 < \*\*\*.sql  
从所有库备份中恢复某一个库(--one-database)

<：导入

mysql -uroot -p --one-database 目标库名 < all.sql

## **Python操作MySQL数据库**

### **pymysql安装**

sudo pip3 install pymysql

### **pymysql使用流程**

1. 建立数据库连接(db = pymysql.connect(...))
2. 创建游标对象(c = db.cursor())
3. 游标方法: c.execute("insert ....")
4. 提交到数据库 : db.commit()
5. 关闭游标对象 ：c.close()
6. 断开数据库连接 ：db.close()

#### **常用函数**

连接数据库：db = pymysql.connect(参数列表)

host ：主机地址,本地 localhost  
port ：端口号,默认3306  
user ：用户名  
password ：密码  
database ：库  
charset ：编码方式,推荐使用 utf8

数据库连接对象(db)的方法

db.close() 关闭连接  
db.commit() 提交到数据库执行  
db.rollback() 回滚  
cur = db.cursor() 返回游标对象,用于执行具体SQL命令

游标对象(cur)的方法

cur.execute(sql命令,[列表]) 执行SQL命令  
cur.close() 关闭游标对象  
cur.fetchone() 获取查询结果集的第一条数据 (1,100001,"河北省")  
cur.fetchmany(n) 获取n条 ((记录1),(记录2))  
cur.fetchall() 获取所有记录

### **代码实现**

#### **１．pymysql使用流程**

**import** pymysql

*# 连接数据库*

db = pymysql.connect(host=**'localhost'**,

port=3306,

user=**'root'**,

passwd=**'123456'**,

database=**'wangzherongyao'**,

charset=**'utf8'**)

*# 获取游标*

cur=db.cursor()

*# 数据操作*

cur.execute(**"insert into data values(5,'兰陵王','m','刺客',10,'致命一击');"**)

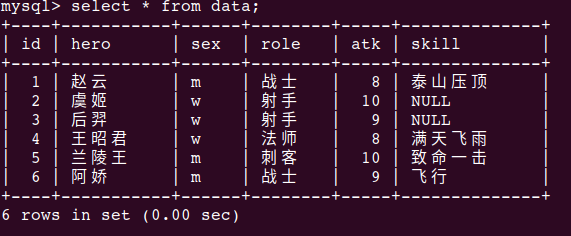
*# 将修改内容提交到数据库*

db.commit()

*# 关闭游标和数据库连接*

cur.close()

db.cursor()



#### ２．sql语句的传参

*"""*

*sql语句传参练习*

*"""*

**import** pymysql

*# 连接数据库*

db = pymysql.connect(host=**'localhost'**,

port=3306,

user=**'root'**,

passwd=**'123456'**,

database=**'stu'**, *# 数据库名*

charset=**'utf8'**)

*# 创建游标*

cur=db.cursor()

**while True**: *# 手动输入*

name = input(**"Name:"**)

age = input(**"Age:"**)

genden = input(**"m or w:"**)

score = input(**"Score:"**)

*# sql = "insert into class\_1 (name,age,genden,score) values (%s,%d,%s,%f);"%(name,age,genden,score)*

sql = **"insert into class\_1(name,age,genden,score) values(%s,%s,%s,%s);"**

print(sql)

**try**:

*# cur.execute(sql)*

*# db.commit()*

cur.execute(sql,[name,age,genden,score])*# 通过列表自动把手动输入的数据传入表中*

db.commit()

**except** Exception **as** e:

db.rollback() *# 失败回滚到操作之前的状态*

print(**"Faild:"**,e)

**else**:

print(**'添加成功'**)

cur.close()

db.close()

#### ３．数据库读操作

*"""*

*数据库读操作*

*select*

*"""*

**import** pymysql

*# 连接数据库*

db = pymysql.connect(host=**'localhost'**,

port=3306,

user=**'root'**,

passwd=**'123456'**,

database=**'stu'**,

charset=**'utf8'**)

*# 创建游标*

cur = db.cursor()

sql = **"select \* from class\_1 where age >= 19;"**

*# 执行语句* *cur拥有查询结果*

cur.execute(sql)

*# 获取查找结果第一个*

one\_row = cur.fetchone()

print(one\_row) *# (1, '老杜', 33, 'w', 59.5)*

*# 获取查找结果前n个*

many\_row = cur.fetchmany(2)

print(many\_row) *# ((1, '老杜', 33, 'w', 59.5), (2, '左哥', 20, 'm', 99.0))*

*# 获取所有查找结果*

all\_row = cur.fetchall()

print(all\_row)

*# ((1, '老杜', 33, 'w', 59.5),*

*# (2, '左哥', 20, 'm', 99.0),*

*# (3, '老徐', 19, 'm', 88.0),*

*# (10, 'Lily', 22, 'w', 75.5))*

cur.close()

db.close()

#### ４．数据库写操作

*"""*

*数据库写操作*

*"""*

**import** pymysql

*# 连接数据库*

db = pymysql.connect(host=**'localhost'**,

port=3306,

user=**'root'**,

passwd=**'123456'**,

database=**'stu'**,

charset=**'utf8'**)

*# 创建游标*

cur = db.cursor()

**try**:

*# 插入操作*

sql = **"insert into interest values(5,'Lily','sing','A','9999','优秀');"**

cur.execute(sql)

*# 修改操作*

sql = **"update interest set price=8686 where name = 'Jam';"**

cur.execute(sql)

*# 删除操作*

sql = **"delete from class\_1 where id > 7;"**

cur.execute(sql)

db.commit()

**except** Exception **as** e:

db.rollback()

print(e)

**else**:

print(**"操作成功"**)

cur.close()

db.close()

#### ５．二进制文件存储－－图片

*"""*

*二进制文件存储*

*先向数据库中存储文件，再从数据库中获取文件*

*"""*

**import** pymysql

*# 连接数据库*

db = pymysql.connect(host=**'localhost'**,

port=3306,

user=**'root'**,

passwd=**'123456'**,

database=**'stu'**,

charset=**'utf8'**)

*# 创建游标*

cur = db.cursor()

*# 向数据库中存储文件*

**with** open(**'/home/tarena/桌面/mysql.jpg'**,**'rb'**) **as** fd:

data = fd.read()

**try**:

sql = **"insert into Images values(1,'mysql.jpg',%s);"**

cur.execute(sql,[data])

db.commit()

**except** Exception **as** e:

db.rollback()

print(e)

**else**:

print(**"存储成功"**)

*# # 获取文件*

*# sql = "select \* from Images where filename='mysql.jpg';"*

*# cur.execute(sql)*

*# image = cur.fetchone()*

*# with open(image[1], 'wb') as fd:# 列表：[1,图片.jpg,data]*

*# fd.write(image[2])*

cur.close()

db.close()

#### ６．ｅｘｅｒｃｉｓｅ

*１．创建一个数据表为user*

*２．编写程序完成如下功能：*

*＊注册，从终端输入用户名和密码，将用户名密码存入到数据库中，用户名不能重复*

*＊登录，从终端输入用户名和密码，如果该用户存在则得到登录成功，不存在则得到登录失败*

*３．将数据库操作功能封装成类*

##### （１）类模块

**import** pymysql

**class** Login:

**def** \_\_init\_\_(self,

database=**'user'**,

host=**'localhost'**,

user=**'root'**,

passwd=**'123456'**,

port=3306,

charset=**'utf8'**,

table=**'login'**):

self.database = database

self.host = host

self.user = user

self.passwd = passwd

self.port = port

self.charset = charset

self.table = table

self.connect\_db() *# 连接数据库*

**def** connect\_db(self):

self.db = pymysql.connect(host=self.host,

user=self.user,

port=self.port,

database=self.database,

passwd=self.passwd,

charset=self.charset)

self.cur = self.db.cursor()

**def** close(self):

self.cur.close()

self.db.close()

**def** login(self, name, passwd):

sql = **"select \* from %s \**

**where name = '%s' and passwd='%s'"** % \

(self.table, name, passwd)

self.cur.execute(sql)

**if** self.cur.fetchone():

**return True**

**else**:

**return False**

**def** register(self, name, passwd):

sql = **"select \* from %s \**

**where name = '%s';"** % \

(self.table, name)

self.cur.execute(sql)

**if** self.cur.fetchone():

**return False**

sql = **"insert into %s (name,passwd) \**

**values ('%s','%s');"** % (self.table, name, passwd)

**try**:

self.cur.execute(sql)

self.db.commit()

**except** Exception:

self.db.rollback()

**return False**

**return True**

##### （２）调用者使用模块

**from** log\_in **import** Login　*# 调用类模块*

user = Login()

*# 　成功返回Ｔｒｕｅ失败返回Ｆａｌｓｅ*

**def** do\_login():

name = input(**"User: "**)

passwd = input(**"Passwd: "**)

**return** user.login(name, passwd)

**def** do\_register():

name = input(**"User: "**)

passwd = input(**"Passwd: "**)

**return** user.register(name, passwd)

**while True**:

print(**"====================="**)

print(**"\*\* login \*\*\*"**)

print(**"\*\* register \*\*\*"**)

print(**"====================="**)

cmd = input(**"Cmd:"**)

**if** cmd == **'login'**:

**if** do\_login():

print(**"登录成功"**)

**else**:

print(**"登录失败"**)

**break**

**elif** cmd == **'register'**:

**if** do\_register():

print(**"注册成功"**)

**else**:

print(**"注册失败"**)

**break**

**else**:

print(**"重新输入"**)

user.close()