MySQL数据库

==========================

| Tedu Python 教学部 |

| --- |

| Author：吕泽|

-----------

[TOC]

## 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 ... ...

![](img/数据库系统.png)

### 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

>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（二维表）来组织数据结构的数据库

>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

![](img/库结构.png)

2. 数据库概念解析

>数据表 ： 存放数据的表格

>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

![](img/表结构.png)

3. 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

4. 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

5. 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

>关闭连接

>> ctrl-D

>> exit

## 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];

```sql

e.g. 创建stu数据库，编码为utf8

create database stu character set utf8;

create database stu charset=utf8;

```

3.查看创建库的语句(字符集)

>show create database 库名;

```sql

e.g. 查看stu创建方法

show create database stu;

```

4.查看当前所在库

>select database();

5.切换库

>use 库名;

```sql

e.g. 使用stu数据库

use stu;

```

6.删除库

>drop database 库名;

```sql

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】 分析存储内容

【2】 确定字段构成

【3】 设计字段类型

2. 数据类型支持

>数字类型：

>>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

![](img/整型.png)

>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

----------------------------------

>字符串类型：

>>CHAR和VARCHAR类型

>>BLOB和TEXT类型

>>ENUM类型和SET类型

![](img/字符串.png)

\* 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，效率偏低 ，但是节省空间。

\* 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

\* 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

-------------------------------------------

1. 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

);

>>\* 如果你想设置数字为无符号则加上 unsigned

>>\* 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。

```sql

e.g. 创建班级表

create table class\_1 (id int primary key auto\_increment,name varchar(32) not null,age int unsigned 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'),level 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

>

>>desc 表名;

>删除表

>

>>drop table 表名;

## 数据基本操作

### 插入(insert)

```SQL

insert into 表名 values(值1),(值2),...;

insert into 表名(字段1,...) values(值1),...;

```

```sql

e.g.

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

```

### 查询(select)

```SQL

select \* from 表名 [where 条件];

select 字段1,字段名2 from 表名 [where 条件];

```

```sql

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

#### 算数运算符

![](img/算数.png)

```sql

e.g.

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

```

#### 比较运算符

![](img/比较.png)

```sql

e.g.

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

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

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

```

#### 逻辑运算符

![](img/逻辑.png)

```sql

e.g.

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

```

#### 位运算符

![](img/位.png)

![](img/运算符.png)

### 更新表记录(update)

```SQL

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

```

```sql

e.g.

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

```

###　删除表记录（delete）

```SQL

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

```

```sql

e.g.

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

```

### 表字段的操作(alter)

```SQL

语法 ：alter table 表名 执行动作;

\* 添加字段(add)

alter table 表名 add 字段名 数据类型;

alter table 表名 add 字段名 数据类型 first;

alter table 表名 add 字段名 数据类型 after 字段名;

\* 删除字段(drop)

alter table 表名 drop 字段名;

\* 修改数据类型(modify)

alter table 表名 modify 字段名 新数据类型;

\* 修改字段名(change)

alter table 表名 change 旧字段名 新字段名 新数据类型;

\* 表重命名(rename)

alter table 表名 rename 新表名;

```

```sql

e.g.

alter table interest add tel char(11) after name;

```

### 时间类型数据

>时间和日期类型:

>>DATE，DATETIME和TIMESTAMP类型

>>TIME类型

>>年份类型YEAR

![](img/时间.png)

#### 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 ：以系统时间存储

2、timestamp ：以标准时间存储但是查看时转换为系统时区，所以表现形式和datetime相同

```sql

e.g.

create table marathon (id int primary key auto\_increment,athlete varchar(32),birthday date,registration\_time datetime,performance time);

```

#### 日期时间函数

\* now() 返回服务器当前时间,格式对应datetime类型

\* curdate() 返回当前日期，格式对应date类型

\* curtime() 返回当前时间，格式对应time类型

#### 时间操作

\* 查找操作

```sql

select \* from marathon where birthday>='2000-01-01';

select \* from marathon where birthday>="2000-07-01" and performance<="2:30:00";

```

\* 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： 2 hour | 1 minute | 2 second | 2 year | 3 month | 1 day

```sql

select \* from marathon where registration\_time > (now()-interval 7 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 %来表示任意0个或多个字符，下划线\_表示任意一个字符。

使用 LIKE 子句从数据表中读取数据的通用语法：

```sql

SELECT field1, field2,...fieldN

FROM table\_name

WHERE field1 LIKE condition1

```

```sql

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

```sql

SELECT field1, field2,...fieldN

FROM table\_name

WHERE field1 REGEXP condition1

```

```sql

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：

```sql

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

```sql

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

```

#### 分页(限制)

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：

```sql

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：

```sql

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。

```sql

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

```

### 数据备份

1. 备份命令格式

>mysqldump -u用户名 -p 源库名 > ~/stu.sql

>> --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 < stu.sql

>从所有库备份中恢复某一个库(--one-database)

>

>>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(cur = db.cursor())

3. 游标方法: cur.execute("insert ....")

4. 提交到数据库或者获取数据 : db.commit()/db.fetchall()

5. 关闭游标对象 ：cur.close()

6. 断开数据库连接 ：db.close()

#### 常用函数

\*\*\*参考代码 day15/mysql.py\*\*\*

*"""*

*mysql.py*

*pymysql 操作数据库基本流程*

*"""*

**import** pymysql

*# 连接数据库*

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

port = 3306,

user=**'root'**,

password = **'123456'**,

database = **'stu'**,

charset=**'utf8'**)

*# 获取游标* *(操作数据库,执行sql语句,得到执行结果)*

cur = db.cursor()

*# 执行语句*

sql = **"insert into interest values\**

**(4,'Emma','draw','C',14800,'xxxx');"**

cur.execute(sql) *# 执行语句*

*# 提交到数据库*

db.commit()

*# 关闭游标*

cur.close()

*# 关闭数据库*

db.close()

\*\*\*参考代码 day15/read\_db.py\*\*\*

*"""*

*read\_db.py*

*pymysql 读操作演示* *(select)*

*"""*

**import** pymysql

*# 连接数据库*

db = pymysql.connect(user=**'root'**,

passwd=**'123456'**,

database=**'stu'**,

charset=**'utf8'**)

*# 获取游标*

cur = db.cursor()

*# 获取数据*

sql=**"select name,hobby from interest \**

**where hobby = 'draw';"**

cur.execute(sql)

*# 可以直接遍历游标*

*# for i in cur:*

*# print(i)*

*# 获取所有查询结果*

*# all\_row = cur.fetchall()*

*# print(all\_row)*

*# 获取多个查询结果*

many\_row = cur.fetchmany(2)

print(many\_row)

*# 获取一个查询结果*

one\_row = cur.fetchone()

print(one\_row)

cur.close()

db.close()

\*\*\*参考代码 day15/write\_db.py\*\*\*

*"""*

*write\_db.py*

*pymysql 写操作演示* *(insert update delete)*

*"""*

**import** pymysql

*# 连接数据库*

db = pymysql.connect(user=**'root'**,

passwd=**'123456'**,

database=**'stu'**,

charset=**'utf8'**)

*# 获取游标*

cur = db.cursor()

*# 执行sql语句*

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

*# age = input('Age:')*

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

**try**:

*# 插入操作*

*# 合成一个正确的sql语句才能正确commit*

*# sql = "insert into class\_1 (name,age,score) \*

*# values ('%s',%d,%f);"%(name,age,score)*

*# sql语句值参量可以通过execute传入*

*# sql = "insert into class\_1 (name,age,score) \*

*# values (%s,%s,%s)"*

*# cur.execute(sql,[name,age,score]) # 执行语句*

*# 修改操作*

sql = **"update class\_1 set score=91 \**

**where name='Abby'"**

cur.execute(sql)

*# 删除操作*

sql = **"delete from class\_1 \**

**where name = 'Jame'"**

cur.execute(sql)

db.commit() *# 同步数据库*

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

print(e)

db.rollback() *# 回滚到没有commit之前的状态*

cur.close()

db.close()

>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

>> cur = db.cursor() 返回游标对象,用于执行具体SQL命令

>> db.commit() 提交到数据库执行

>> db.rollback() 回滚，用于当commit()出错是回复到原来的数据形态

>> db.close() 关闭连接

>游标对象(cur)的方法

>>cur.execute(sql命令,[列表]) 执行SQL命令

>>cur.fetchone() 获取查询结果集的第一条数据，查找到返回一个元组否则返回None

>>cur.fetchmany(n) 获取前n条查找到的记录，返回结果为元组嵌套元组， ((记录1),(记录2))。

>>cur.fetchall() 获取所有查找到的记录，返回结果形式同上。

>>cur.close() 关闭游标对象