今日目标

- 能够使用pymysql库对mysql数据库进行增删改查操作

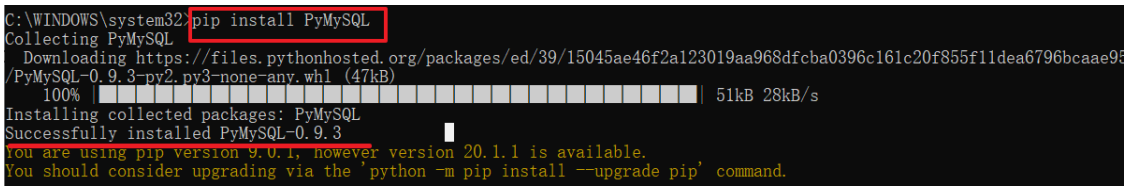
一、数据库介绍

- 概念：一个存放数据的仓库（Database），这个仓库按照一定的数据结构组织、存放、管理数据。
- 分类：
  - 关系型数据库：mysql、sql server、oracle、DB2等
  - 非关系型数据库：redis等
- python操作数据库的方式
  - pymysql：纯python开发，支持python2和python3，**简单易用**

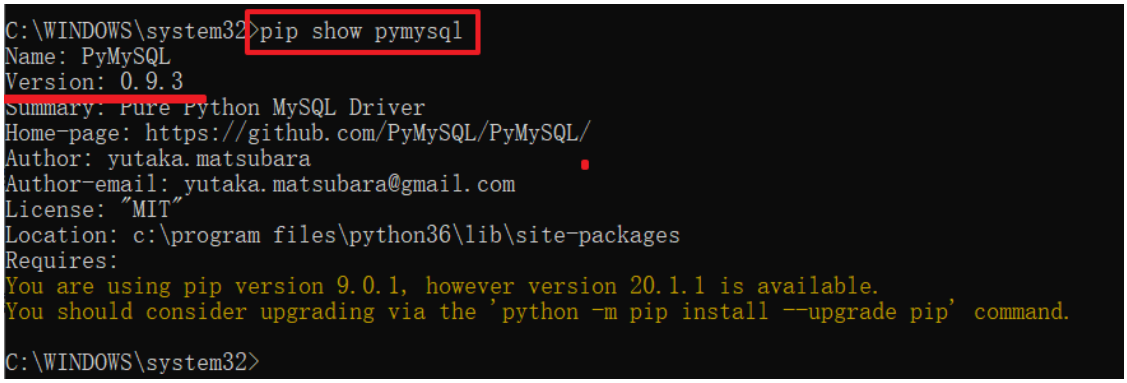
二、数据库基本操作

1、安装

- 安装：`pip install PyMySQL`



- 验证：`pip show PyMySQL`



2、操作流程（重点）

1. 创建连接
2. 获取游标
3. 执行sql
    1. 查询操作（select）
    2. 非查询操作（insert/update/delete）
        1. 事务提交（连接对象.commit()）

2. 事务回滚（连接对象.rollback()）

4. 关闭游标
5. 关闭连接

- 什么是游标？（了解）
  - 游标是SQL 的一种数据访问机制 ， 游标是一种处理数据的方法。

    众所周知，使用SQL的select查询操作返回的结果是一个包含一行或者是多行的数据集，如果我们要对查询的结果再进行查询，比如（查看结果的第一行、下一行、最后一行、前十行等等操作）简单的通过select语句是无法完成的，因为这时候索要查询的结果不是数据表，而是已经查询出来的结果集。

    游标就是针对这种情况而出现的。

    我们可以将“ 游标 ”简单的看成是结果集的一个指针，可以根据需要在结果集上面来回滚动，浏览我需要的数据。

3、数据准备

```
CREATE DATABASE if not EXISTS books DEFAULT charset utf8;
use books;

Drop TABLE if EXISTS `t_book`;
CREATE TABLE `t_book`(
    `id` int(11) not null auto_increment,
    `title` VARCHAR(20) not NULL COMMENT '图书名称',
    `pub_date` date not NULL COMMENT '发布日期',
    `read` int(11) not null default '0' comment '阅读量',
    `comment` int(11) not null default '0' comment '评论量',
    `is_delete` TINYINT(1) not NULL DEFAULT '0' COMMENT '逻辑删除',
    PRIMARY KEY(`id`)
)ENGINE=INNODB DEFAULT CHARSET=utf8 COMMENT='图书表';

INSERT into `t_book` VALUES ('1','射雕英雄传','1980-05-01','12','34','0');
INSERT into `t_book` VALUES ('2','天龙八部','1986-07-24','36','40','0');
INSERT into `t_book` VALUES ('3','笑傲江湖','1995-12-24','20','80','0');


Drop TABLE if EXISTS `t_hero`;
CREATE TABLE `t_hero`(
    `id` int(11) not null auto_increment,
    `name` VARCHAR(20) not NULL COMMENT '姓名',
    `gender` SMALLINT(6) not NULL COMMENT '性别',
    `description` VARCHAR(200) default NULL comment '描述',
    `is_delete` TINYINT(1) not NULL DEFAULT '0' COMMENT '逻辑删除',
    `book_id` int(11) not null comment '所属图书ID',
    PRIMARY KEY(`id`),
    key `t_hero_book_id`(`book_id`)
)ENGINE=INNODB DEFAULT CHARSET=utf8 COMMENT='英雄人物表';

INSERT into `t_hero` VALUES ('1','郭靖','1','降龙十八掌','0','1');
INSERT into `t_hero` VALUES ('2','黄蓉','0','打狗棍法','0','1');
INSERT into `t_hero` VALUES ('3','乔峰','1','降龙十八掌','0','2');
INSERT into `t_hero` VALUES ('4','令狐冲','1','独孤九剑','0','3');
```

```
INSERT into `t_hero` VALUES ('5','任盈盈','0','弹琴','0','3');
```

注意事项：

- 直接使用本地localhost数据库即可
    1. 请先通过phpstudy启动本地的数据库
    2. 然后通过Navicat连接本地数据库
    3. 执行数据库初始化语句

4、数据库基本操作

4.1 连接数据库

```
1).连接到数据库（host:localhost user:root password:root database:books）
2).获取数据库服务器版本信息
```

```
# 导包
import pymysql

# 创建连接
conn = pymysql.connect(host="localhost",
                       port=3306,
                       user="root",
                       password="root",
                       database="books")

# 获取游标
cursor = conn.cursor()

# 执行sql
cursor.execute("select version()")
result = cursor.fetchall()
print(result)

# 关闭游标
cursor.close()

# 关闭连接
conn.close()
```

4.2 数据库查询操作

```
1).连接到数据库（host:localhost user:root password:root database:books）
2).查询图书表的数据（包括：图书id、图书名称、阅读量、评论量）
3).获取查询结果的总记录数
4).获取查询结果的第一条数据
5).获取全部的查询结果
```

```
"""
```

```
1).连接到数据库（host:localhost user:root password:root database:books）
2).查询图书表的数据（包括：图书id、图书名称、阅读量、评论量）
3).获取查询结果的总记录数
4).获取查询结果的第一条数据
5).获取全部的查询结果
"""

# 导包
import pymysql

# 创建连接
# 1).连接到数据库（host:localhost user:root password:root database:books）
conn = pymysql.connect(host="localhost",
                       port=3306,
                       user="root",
                       password="root",
                       database="books")

# 获取游标
cursor = conn.cursor()

# 执行sql
# 2).查询图书表的数据（包括：图书id、图书名称、阅读量、评论量）
sql = "select id, title, `read`, `comment` from t_book;"
cursor.execute(sql)

# 3).获取查询结果的总记录数
print("获取的查询结果记录行数为：", cursor.rowcount)

# # 4).获取查询结果的第一条数据
# print(cursor.fetchone())

# 5).获取全部的查询结果
print(cursor.fetchall())

# 关闭游标
cursor.close()

# 关闭连接
conn.close()
```

4.3 数据库插入操作

```
1).连接到数据库（host:localhost user:root password:root database:books
autocommit:True）
2).新增一条图书数据（id:4 title:西游记 pub_date:1986-01-01 ）
```

```
"""
1).连接到数据库（host:localhost user:root password:root database:books）
2).查询图书表的数据（包括：图书id、图书名称、阅读量、评论量）
3).获取查询结果的总记录数
4).获取查询结果的第一条数据
5).获取全部的查询结果
```

```
"""

# 导包
import pymysql

# 创建连接
# 1).连接到数据库（host:localhost user:root password:root database:books）
conn = pymysql.connect(host="localhost",
                       port=3306,
                       user="root",
                       password="root",
                       database="books",
                       autocommit=True)

# 获取游标
cursor = conn.cursor()

# 执行sql
# 新增一条图书数据（id:4 title:西游记 pub_date:1986-01-01 ）
sql = "insert into t_book(id, title, pub_date) values(4, '西游记', '1986-01-
01');"
cursor.execute(sql)

# 3).获取受影响的结果记录数
print("影响的结果记录数为：", cursor.rowcount)

# 关闭游标
cursor.close()

# 关闭连接
conn.close()
```

4.4 数据库更新操作

```
1).连接到数据库（host:localhost user:root password:root database:books
autocommit:True）
2).更新[西游记]图书名称为（title:东游记）
```

```
"""
1).连接到数据库（host:localhost user:root password:root database:books
autocommit:True）
2).更新[西游记]图书名称为（title:东游记）
"""

# 导包
import pymysql

# 创建连接
conn = pymysql.connect(host="localhost",
                       port=3306,
                       user="root",
                       password="root",
                       database="books",
                       autocommit=True)
```

```
# 获取游标
cursor = conn.cursor()

# 执行sql
sql = "update t_book set title='东游记' where title = '西游记';"
cursor.execute(sql)
print(cursor.rowcount)

# 关闭游标
cursor.close()

# 关闭连接
conn.close()
```

4.5 数据库删除操作

```
1).连接到数据库（host:localhost user:root password:root database:books
autocommit:True）
2).删除图书（title:东游记）
```

```
"""
1).连接到数据库（host:localhost user:root password:root database:books
autocommit:True）
2).删除图书（title:东游记）
"""

# 导包
import pymysql

# 创建连接
conn = pymysql.connect(host="localhost",
                       port=3306,
                       user="root",
                       password="root",
                       database="books",
                       autocommit=True)

# 获取游标
cursor = conn.cursor()

# 执行sql
sql = "delete from t_book where title = '东游记';"
cursor.execute(sql)
print(cursor.rowcount)

# 关闭游标
cursor.close()

# 关闭连接
conn.close()
```

查询与非查询（插入、更新、删除）操作小结：

- 相同点：基本操作流程是一样
  - 创建连接
  - 获取游标
  - 执行sql
  - 关闭游标
  - 关闭连接
- 不同点
  - 要执行sql语句不一样
  - 非查询操作需要开启事务（在创建连接时，指定参数autocommit=True）

三、数据库事务操作

- 引入案例

```
1).连接到数据库（host:localhost user:root password:root database:books），
并开启自动提交事务
2).新增一条图书数据（id:4 title:西游记 pub_date:1986-01-01 ）
3).故意抛出一个异常（模拟代码出现异常）
4).新增一条英雄人物数据（name:孙悟空 gender:1 book_id:4）
```

```
"""
1).连接到数据库（host:localhost user:root password:root database:books），
并开启自动提交事务
2).新增一条图书数据（id:4 title:西游记 pub_date:1986-01-01 ）
3).故意抛出一个异常（模拟代码出现异常）
4).新增一条英雄人物数据（name:孙悟空 gender:1 book_id:4）
"""
# 导包
import pymysql

# 创建连接
conn = pymysql.connect(host="localhost",
                       port=3306,
                       user="root",
                       password="root",
                       database="books",
                       autocommit=True)

# 获取游标
cursor = conn.cursor()

# 执行sql
sql = "insert into t_book(id, title, pub_date) values(4, '西游记', '1986-01-
01');"
cursor.execute(sql)
print(cursor.rowcount)
print("-" * 200)

# 主动抛出异常
```

```
raise Exception("程序出错啦。。。。。。")

# 4).新增一条英雄人物数据（name:孙悟空 gender:1 book_id:4）
sql = "insert into t_hero(name,gender,book_id) values('孙悟空', 1, 4)"
cursor.execute(sql)
print(cursor.rowcount)

# 关闭游标
cursor.close()

# 关闭连接
conn.close()
```

- 概念【理解】
  - 基于代码的角度：一段实现了具体业务单元功能的代码，这段代码要么都执行，要么都不执行
  - 基于业务的角度：最小的业务单元，要么都成功，要么都失败

- 特点【了解】**ACID**
  - 原子性：事务中的一系列操作、他是最基本的工作单元。
  - 一致性：在数据库看到的结果要么是执行之前的结果，要么是执行之后的结果。
  - 隔离性：事务的内部状态对其他事务是不可见的。
  - 持久性：通过事务对数据库中数据做的改变，永久有效。

- 操作【理解】
  - 自动提交（不推荐）：`autocommit=True`
  - 手动提交（推荐）：
    - 提交事务：`conn.commit()`
    - 回滚事务：`conn.rollback()`

- 解决导入案例中的问题：数据不一致性问题

```
1).连接到数据库（host:localhost user:root password:root database:books），
并开启自动提交事务
2).新增一条图书数据（id:4 title:西游记 pub_date:1986-01-01 ）
3).故意抛出一个异常（模拟代码出现异常）
4).新增一条英雄人物数据（name:孙悟空 gender:1 book_id:4）
```

思路：

```
1.导包

try:
    程序前期，需要执行的代码
```

```
    2.创建连接对象
    3.获取游标对象
    4.执行sql
        + 在图书表中插入一行数据
        + 主动抛出异常
        + 在英雄人物表中插入一行数据

        调用提交事务：conn.commit()
except:
    程序出现异常后，处理代码
    调用事务回滚：conn.rollback()

finally:
    程序结束时，需要执行的代码
    5.关闭游标
    6.关闭连接
```

代码实现：

```
# 导包
import pymysql

# 初始化
conn = None
cursor = None

# 业务处理
try:
    # 创建连接
    conn = pymysql.connect(host="localhost",
                           port=3306,
                           user="root",
                           password="root",
                           database="books",
                           autocommit=False)
    # 获取游标
    cursor = conn.cursor()

    # 执行sql
    sql = "insert into t_book(id, title, pub_date) values(4, '西游记', '1986-
01-01');"
    cursor.execute(sql)
    print(cursor.rowcount)
    print("-" * 200)

    # 主动抛出异常
    raise Exception("程序出错啦。。。。。。")

    # 4).新增一条英雄人物数据（name:孙悟空 gender:1 book_id:4）
    sql = "insert into t_hero(name,gender,book_id) values('孙悟空', 1, 4)"
    cursor.execute(sql)
    print(cursor.rowcount)

    # 提交事务
    conn.commit()
```

```
except Exception as e:
    # 回滚数据
    conn.rollback()

    # 打印异常信息
    print(e)

finally:
    # 关闭游标
    if cursor:
        cursor.close()
    # 关闭连接
    if conn:
        conn.close()
```

四、数据库工具封装

- 需求

```
需求分析:
1. sql = "select * from t_book"
2. 调用数据库工具方法
result = exe_sql(sql)
print("结果: ", result)
```

- 思路:

```
1、创建连接
2、创建游标
3、执行sql
    try:
        # 获取游标对象
        # 调用游标对象.execute(sql)
        # 如果是 查询:
            # 返回所有数据
        # 否则:
            # 提交事务
            # 返回受影响的行数
    except:
        # 回滚事务
        # 抛出异常
    finally:
        # 关闭游标
        # 关闭连接
4、关闭游标
5、关闭连接
```

- 实现

```python
# 导包
import pymysql

# 创建工具类
class DBUtil():
    # 初始化
    __conn = None
    __cursor = None

    # 创建连接
    @classmethod
    def __get_conn(cls):
        if cls.__conn is None:
            cls.__conn = pymysql.connect(host="localhost",
                                         port=3306,
                                         user="root",
                                         password="root",
                                         database="books")
        return cls.__conn

    # 获取游标
    @classmethod
    def __get_cursor(cls):
        if cls.__cursor is None:
            cls.__cursor = cls.__get_conn().cursor()
        return cls.__cursor

    # 执行sql
    @classmethod
    def exe_sql(cls, sql):
        try:
            # 获取游标对象
            cursor = cls.__get_cursor()
            # 调用游标对象的execute方法，执行sql
            cursor.execute(sql)
            #  如果是查询
            if sql.split()[0].lower() == "select":
                # 返回所有数据
                return cursor.fetchall()
            #  否则:
            else:
                # 提交事务
                cls.__conn.commit()
                # 返回受影响的行数
                return cursor.rowcount
        except Exception as e:
            # 事务回滚
            cls.__conn.rollback()
            # 打印异常信息
            print(e)
        finally:
            # 关闭游标
            cls.__close_cursor()
            # 关闭连接
            cls.__close_conn()

    # 关闭游标
    @classmethod
```

```
    def __close_cursor(cls):
        if cls.__cursor:
            cls.__cursor.close()
            cls.__cursor = None

    # 关闭连接
    @classmethod
    def __close_conn(cls):
        if cls.__conn:
            cls.__conn.close()
            cls.__conn = None
```

- 验证

```
from test10_dbutil import DBUtil

# sql = "select * from t_book"
# sql = "insert into t_book(id, title, pub_date) values(4, '西游记', '1986-01-01');"
# sql = "update t_book set title='东游记' where title = '西游记';"
sql = "delete from t_book where title = '东游记';"
result = DBUtil.exe_sql(sql)
print(result)
```

今日总结

- 能够说出pymysql操作数据库的基本流程
- 能够通过游标对象实现数据库的增删改查操作
- 能够结合具体事例（如转账）说出数据库事务的概念
- 知道pymysql处理数据库事务的主要方法

每日作业

1、请简述安装pymysql的流程

2、请简述事务的4个特性

3、请使用pymysql完成以下需求：

- 插入一本书，书名为‘python从入门到放弃’，阅读量为50，评论量为0，发布日志为：2020-01-01
- 测试工程师人员发现一个bug，这个本书的评论数与实际不符，要求你把评论量修改为修正后的值：250

- 老板投资了python，觉得这本书名太不吉利，需要下架，请删除这本书。
- 你删除后，心中不放心到底有没有删除，想确认是否真正删除了，你需要怎么做?

4、小明正在练习pymysql，他写入了插入数据库的代码，在代码中能查到，但是查询数据库时，却查不到他插入的数据，请帮他找一找原因，并写出改正代码。

```
# 导包
import pymysql
# 建立连接
conn = pymysql.connect("localhost",'root','root','books')
# 获取游标
cursor = conn.cursor()
sql_insert = "insert into t_book(title,`read`,`comment`,pub_date) values('小明的哥哥叫大明',50,0,'2020-01-01');"
cursor.execute(sql_insert)
cursor.execute("select * from t_book where title='小明的哥哥叫大明';")
print("小明插入的书：", cursor.fetchall())
# 关闭游标
cursor.close()
# 关闭连接
conn.close()

# 输出
小明插入的书： ((8, '小明的哥哥叫大明', datetime.date(2020, 1, 1), 50, 0, 0),)
# 通过数据库连接查询结果：
1   射雕英雄传   1980-05-01  12  34  0
2   天龙八部    1986-07-24  36  40  0
3   笑傲江湖    1995-12-24  20  80  0
```