北京邮电大学数据库系统实验报告

实验名称: 数据查询实验

计算机科学与技术系
2015211306班
魏晓
学号:2015211301

实验目的

- 通过实验了解通用数据库应用编程接又ODBC的基本原理和实现机制， 熟悉主要的 ODBC接又的语法和使用方法
- 利用C语言(或其它支持ODBC接又的高级程序设计语言)编程实现简单的数据库应用程序，掌握基于ODBC的数据库访问的基本原理和方法
- 学习java语言，并采用jdbc接又方式对数据库进行访问

实验要求

1. 以教科书第四章关于SQL语言相关内容为基础，课后查阅、自学ODBC接又有关内容， 包括ODBC的体系结构、工作原理、数据访问过程、主要API接又的语法和使用方法
2. 以实验二建立的学生数据库为基础，编写 C语言(或其它支持ODBC接又的高级程序 设计语言)数据库应用程序，按照如下步骤访问数据库
    - Step1. ODBC初始化，为ODBC分配环境句柄
    - Step2. 建立应用程序与ODBC数据源的连接
    - Step3. 利用SQLExecDirect语句，实现数据库应用程序对数据库的建立、查询、 修改、删除等操作
    - Step4. 检索查询结果集

- Step5. 结束数据库应用程序

实验平台及环境

> 实验平台:MySQL 14.14 with Python 3.6
> 运行环境:Mac OS High Sierra 10.13(17A405)
> 可视界面:MySQL WorkBench 6.3.10

```
Xiaos-MacBook-Pro:~ weixiao$ mysql --version
mysql  Ver 14.14 Distrib 5.7.20, for macos10.12 (x86_64) using  EditLine wrapper
```

实验步骤及结果分析

1. 连接数据库

   - 关于pymysql

   > This package contains a pure-Python MySQL client library. The goal of PyMySQL is to be a drop-in replacement for MySQLdb and work on CPython, PyPy and IronPython.

```
import pymysql

pyconn = pymysql.connect(
host="localhost",
port=3306,
user="wx",
passwd="password",
db="mytest")
```

   - 首先import pymysql后,利用**pymysql.connect**链接本地服务器的数据库,关于*connect*函数的各个参数
     - host指定了服务器所处的位置
     - port指定了访问端口
     - user指定了数据库的用户名
     - passwd是该用户的密码
     - db是数据库的名称
     - charset缺省设置为utf8

```
mysql> status
--------------
mysql  Ver 14.14 Distrib 5.7.20, for macos10.12 (x86_64) using  EditLine wrapper

Connection id:          7
Current database:       mytest
Current user:           wx@localhost
SSL:                    Not in use
Current pager:          stdout
Using outfile:          ''
Using delimiter:        ;
Server version:         5.7.20 MySQL Community Server (GPL)
Protocol version:       10
Connection:             Localhost via UNIX socket
Server characterset:    latin1
Db     characterset:    latin1
Client characterset:    utf8
Conn.  characterset:    utf8
UNIX socket:            /tmp/mysql.sock
Uptime:                 6 hours 46 min 30 sec

Threads: 3  Questions: 278  Slow queries: 0  Opens: 129  Flush tables: 1  Open t
ables: 122  Queries per second avg: 0.011
```

2. 获取数据库游标

   - 关于游标

> The MySQLCursor class instantiates objects that can execute operations such as SQL statements. Cursor objects interact with the MySQL server using a MySQLConnection object.

```
cursor = pyconn.cursor()
```

   - cursor方法返回一个游标对象,这个游标是python和mysql数据的桥梁

3. 获取数据

   i. 查询

```
sql="SELECT * FROM sc"

try:
    c=cursor.execute(sql)
    info = cursor.fetchmany(c)
    for i in info:
        print(i)
```

```python
#!/usr/bin/env python3
# -*- coding:utf-8 -*-
import pymysql

pyconn = pymysql.connect(
    host="localhost",
    port=3306,
    user="wx",
    passwd="password",
    db="mytest")

cursor = pyconn.cursor()

sql="SELECT * FROM sc"

try:
    c=cursor.execute(sql)
    info = cursor.fetchmany(c)
    for i in info:
        print(i)
```

```
Run test1
/Users/weixiao/anaconda/bin/python /Users/weixiao/PycharmProjects/MySQLtest1/test1.py
('30201', 'C03', 40)
('30201', 'C04', 88)
('30201', 'C05', 93)
('30202', 'C03', 40)
('30202', 'C04', 40)
('30203', 'C03', 57)
('30203', 'C04', 50)
('30203', 'C05', 40)
('30204', 'C03', 54)
```

- 可利用fetchmany 方法一次性获得多个数据,info的本质是一个list,list中的每一个 单元都是表中的一个元组,遍历并打印这个list即可看到查询结果
- 对于增删改这个命令,在cursor.execute之后还必须执行一次commit命令来将这个事物提交到数据库本身,否则我们虽然能得到正确的结果,但实际上数据库本身并没有发生变化(因为事务并没有被提交到数据库)

ii. 插入

```python
sql="INSERT INTO sc VALUES('40404','C11',99)"

try:
    c=cursor.execute(sql)
    print(c)
    pyconn.commit()
```

```
| 31428 | C01 |    40 |
| 31428 | C02 |    40 |
| 31428 | C03 |    40 |
| 40404 | C11 |    99 |
+-------+-----+-------+
143 rows in set (0.00 sec)
```

- 可以看到最后一行就是我们增加的数据

iii. 删除

```
cursor = pyconn.cursor()

sql="DELETE FROM sc WHERE sc.cno='C11'"

try:
    c=cursor.execute(sql)
    pyconn.commit()
finally:
    pyconn.close()
```

```
| 31427 | C01 |    40 |
| 31427 | C02 |    88 |
| 31428 | C01 |    40 |
| 31428 | C02 |    40 |
| 31428 | C03 |    40 |
+-------+-----+-------+
142 rows in set (0.00 sec)
```

- 可以看到最后的40404已经被删除了

iv. 更新

```
sql="UPDATE sc SET grade= 99 WHERE sno='31428' AND cno='C03'"

try:
    c=cursor.execute(sql)
    pyconn.commit()
finally:
    pyconn.close()
```

```
| 31428 | C01 |    40 |
| 31428 | C02 |    40 |
| 31428 | C03 |    99 |
+-------+-----+-------+
142 rows in set (0.00 sec)

mysql>
```

- 最后一行已经成功修改为99了

4. 释放游标

```
    cursor.close()
```

5. 释放相关资源并断开链接

```
    pyconn.close()
```

实验感想

1. 什么是游标(cursor)?
    - 个人觉得就是一个cursor,就是一个标识，用来标识数据去到什么地方了。你也可以把它理解成数组中的下标。
2. 游标(cursor)的特性
    - 只读的，不能更新的。
    - 不滚动的
    - 不敏感的，不敏感意为服务器可以或不可以复制它的结果
3. 游标(cursor)必须在声明处理程序之前被声明，并且变量和条件必须在声明游标或处理程序之前被声明
4. 本次实验将我们熟悉的Python和MySQL联系在一起,这种数据接口为使用脚本语言直接操作MySQL提供了方便