day05_mysql扩展

学习目标：

1、掌握Python和mysql的交互

2、了解枚举类型和集合类型

3、了解存储引擎

4、了解事务

5、了解索引

6、了解视图

1.python与mysql的交互

pymysql是可用于连接mysql数据库，且能够提供mysql与python窗口交互创立通道的工具库。可以通过创建引擎，建立游标直接通过python编程实现mysql数据库操作。

PyMySQL 是一个纯 Python 实现的 MySQL 客户端库，支持兼容 Python 3，用于代替 MySQLdb。

安装pymysql库

```
pip install PyMySQL -i https://pypi.tuna.tsinghua.edu.cn/simple
```

什么是游标

> 游标，通俗来说就是“游动的标志”，
>
> 有时候，我们执行一条查询语句的时候，往往会得到N条返回结果，沿着这个游标，我们可以一次取出一行记录。
> 举个例子：
> 当不使用游标功能，去执行 select * from product where price > 3000 ; 这条语句时，如果有1000条返回结果，系统会一次性将1000条记录返回到界面中，你不能做选择，也不能做其他操作。
> 当我们开启了游标功能，再去执行这条语句的时候，系统会先帮你找到这些行，先给你存放起来。当你需要数据的时候，就借助这个游标去一行行的取出数据，你每取出一条记录，游标指针就朝前移动一次，一直到取完最后一行数据。

pymysql.connect()函数参数：

参数	说明
host	数据库连接地址
user	数据库用户名
password	数据库用户密码
database	要连接的数据库名
charset	连接数据库的字符编码
port	端口号，默认3306

常用方法:

方法	说明
cursor()	获取游标对象，操作数据库
fetchone()	获取一行数据
fetchall()	获取所有数据
close()	关闭连接
commit()	提交事务
rollback()	回滚事务

创建数据库

```
import pymysql
# pymysql创建数据库
# 创建连接
db = pymysql.connect(host='localhost', port=3306, user='root', password='root',
charset='utf8')
# 创建游标
cursor = db.cursor()
# 写sql语句
sql = 'create database test_pymysql;'
# execute()方法执行sql语句
cursor.execute(sql)
```

查询数据

```
import pymysql
# 查询day04数据中的products表的数据
# 获取MySQL连接
conn = pymysql.connect(host='localhost', port=3306, user='root',
password='root', database='day04', charset='utf8')
# 获取游标
cursor = conn.cursor()

# 执行sql语句 返回值就是sql语句在执行过程中影响的行数
sql = "select * from products;"
```

```
row_count = cursor.execute(sql)
print(f"sql语句执行影响的行数{row_count}")

# # fetchone()取出结果集中一行   返回的结果是一行
# print(cursor.fetchone())

# fetchall()函数返回所有数据
print(cursor.fetchall())

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

增删改数据

```
import pymysql

#获取MySQL连接
conn = pymysql.connect(host='localhost', port=3306,
user='root',password='root',database='day04', charset='utf8')
# 获取游标
cursor = conn.cursor()
# 1.创建一个hero_kongfu表
# sql = 'create table hero_kongfu(hid int,hname varchar(50),kname varchar(50));'
# cursor.execute(sql)

# 2.插入数据
# 第一种方式
# sql = "insert into hero_kongfu values(%s,%s,%s)"
# data = (1, '乔峰', '降龙十八掌')
# cursor.execute(sql, data)  #sql和data之间以","隔开
# 第二种方式
# sql = "insert into hero_kongfu values(2,'段誉','六脉神剑');"
# cursor.execute(sql)
# 第三种
# sql = "insert into hero_kongfu values (3,'慕容复','斗转星移'),(4,'虚竹','天山折梅
手');"
# cursor.execute(sql)

# 3.修改数据
# sql = "update hero_kongfu set kname='打狗棒法' where hname='乔峰';"
# cursor.execute(sql)

# 4.删除数据
# sql = "delete from hero_kongfu where hname='虚竹';"
# cursor.execute(sql)

conn.commit()   # 提交，不然无法保存插入或者修改或删除的数据(这个一定不要忘记加上)
# 关闭游标
cursor.close()
# 关闭连接
conn.close()
```

2.枚举类型和集合类型

知识点:

```
枚举类型: enum(x,y): 类似单选,一次只能插入一个值
注意: 枚举类型需要提前设定好值,插入的时候如果不是提前设定好的就报错

集合类型: set(x,y): 类似多选,一次可以插入多个值
注意: 集合类型需要提前设定好值,插入的时候如果不是提前设定好的就报错
```

示例:

```
# 创建库
create database day05;
# 使用库
# use day05;  -- 注意: 如果不写use 库名,那么以后所有操作都需要库名.表名操作
# 创建表
create table day05.user(
    id int primary key auto_increment,
    uname varchar(100),
    pwd varchar(100),
    age int,
    sex enum('男','女'),
    hobby set('吃饭','学习','敲代码')
);
# 验证枚举类型特点: 只能插入提前设置好的,一次只能插入一个
insert into day05.user(uname,sex) values('张三','妖'); -- 报错,因为妖没有提前设置
insert into day05.user(uname,sex) values('张三','男,女'); -- 报错,因为一次只能插入一个
insert into day05.user(uname,sex) values('张三','男'); -- 插入成功

# 验证集合类型特点: 只能插入提前设置好的,一次可以插入多个
insert into day05.user(uname,hobby) values('李四','游戏'); -- 报错,因为游戏没有提前设置
insert into day05.user(uname,hobby) values('李四','吃饭,学习'); -- 插入成功
insert into day05.user(uname,hobby) values('李四','敲代码'); -- 插入成功

# 假设妖被人们所认可,那就需要修改sex字段数据
alter table day05.user change sex sex enum('男','女','妖');
insert into day05.user(uname,sex) values('张三','妖'); -- 插入成功,因为妖已经被设置到了枚举中
```

3.mysql存储引擎

> MySQL提供了多种不同的存储引擎供我们使用，并且不同的表可以使用不同的存储引擎。
> 常见的存储引擎有: InnoDB、MyISAM、Memory、Archive、CSV等
> 可以使用**show engines;** 查看数据库支持的存储引擎:

Engine	Support	Comment	Transactions	XA	Savepoints
MEMORY	YES	Hash based, stored in memory, useful for temporary tables	NO	NO	NO
MRG_MYISAM	YES	Collection of identical MyISAM tables	NO	NO	NO
CSV	YES	CSV storage engine	NO	NO	NO
FEDERATED	NO	Federated MySQL storage engine	*<null>*	*<null>*	*<null>*
PERFORMANCE_SCHEMA	YES	Performance Schema	NO	NO	NO
MyISAM	DEFAULT	MyISAM storage engine	NO	NO	NO
InnoDB	YES	Supports transactions, row-level locking, and foreign keys	YES	YES	YES
BLACKHOLE	YES	/dev/null storage engine (anything you write to it disappears)	NO	NO	NO
ARCHIVE	YES	Archive storage engine	NO	NO	NO

查看建表语句中默认添加的存储引擎: show create table day05.user;
查看所有存储引擎: show engines;

在MySQL中最常用存储引擎的是: InnoDB和MyISAM
InnoDB和MyISAM的区别是:
1、innodb支持事务，而myisam不支持事务。
2、innodb支持外键，而myisam不支持外键。
3、innodb默认行锁，而myisam是表锁（每次增加删除更新都会锁住表）。
4、innodb是聚集索引，而myisam是非聚集索引(索引和数据是分离的)。
5、innodb不存储表的总行数，select count(*) from 表名;的时候会全表查询，执行相对较慢。
而myisam会存放表的总行数，所以select count(*) from 表名 ;的时候会查的很快。

4.mysql事务

> 事务(Transaction)就是指逻辑上的一组sql语句操作，组成这组操作的各个sql语句，执行时要么全成功要么全失败。
> MySQL 事务主要用于处理操作量大，复杂度高的数据。在 MySQL 中只有使用了 InnoDB 数据库引擎的数据库或表才支持事务。
> 事务处理可以用来维护数据库的完整性，保证成批的 SQL 语句要么全部执行，要么全部不执行。
> 事务用来管理 insert,delete,update 语句。

事务ACID特性

事务应该具有4个属性：原子性、一致性、隔离性、持久性。这四个属性通常称为ACID特性。

原子性（atomicity）：一个事务是一个不可分割的工作单位，事务中包括的操作要么都做，要么都不做。

一致性（consistency）：事务必须是使数据库从一个一致性状态变到另一个一致性状态。一致性与原子性是密切相关的。

隔离性（isolation）：一个事务的执行不能被其他事务干扰。即一个事务内部的操作及使用的数据对并发的其他事务是隔离的，并发执行的各个事务之间不能互相干扰。

持久性（durability）：一个事务一旦提交，它对数据库中数据的改变就应该是永久性的。接下来的其他操作或故障不应该对其有任何影响。

事务分类

事务的分类主要是两种: 隐式事务和显式事务

隐式事务:该事务没有明显的开启和结束标记，它们都具有自动提交事务的功能；我们的DML语句（insert、update、delete）就是隐式事务。

显式事务:该事务具有明显的开启和结束标记。使用显式事务的前提是你得先把自动提交事务的功能给禁用,禁用自动提交功能就是设置autocommit变量值为0（0:禁用 1:开启）

查看自动提交事务状态: select @@autocommit;

禁用自动提交事务的方式: set autocommit = 0;

开启自动提交事务的方式: set autocommit = 1;

```
# 查看自动提交事务状态:
```

```
select @@autocommit; -- 结果1代表自动执行事务操作
# 禁用自动提交事务的方式:
set autocommit = 0;


# 演示手动操作事务
# 创建账目表
create table student(
    id int,
    name varchar(100),
    money double
);
# 插入数据
# 演示自动提交事务,前提autocommit = 1
# 查看自动提交事务状态:
select @@autocommit; -- 结果0代表关闭了自动提交功能
# 开启自动提交事务的方式:
set autocommit = 1;
# 测试插入数据
insert into student values(1,'阿三',10000);
insert into student values(2,'阿四',10000);

# 转换大小写: ctrl+shift+U
# 演示手动提交事务,前提autocommit = 0
# 查看自动提交事务状态:
select @@autocommit; -- 结果1代表开启了自动提交功能
# 禁用自动提交事务的方式:
set autocommit = 0;
# 需求: 阿三给阿四转账
# 开启事务
start transaction;
# 阿三账户先减1000
update student set money = money-1000 where name='阿三';
# 阿四账户再加1000
update student set money = money+1000 where name='阿四';
# 提交事务
commit ;

# 演示事务如何保证数据完整性
# 开启事务
start transaction;
# 假设阿三账户在atm机器上做了扣款操作后,瞬间突然断电了
update student set money = money-1000 where name='阿三';
# 检测到错误后,不能提交,立刻回滚事务
rollback; -- 注意: 一旦提交了就不能回滚了
# 断电后,此操作就不会执行到了
update student set money = money+1000 where name='阿四';
```

5.mysql索引

索引概述

MySQL官方对索引的定义是：索引（Index）是帮助MySQL高效获取数据的数据结构。索引最形象的比喻就是图书的目录。
注意：只有在大量数据中查询时索引才显得有意义。
按字段特性分类：主键索引(PRIMARY KEY)、唯一索引(UNIQUE INDEX)、普通索引(INDEX)、全文索引(FULLTEXT)

主键索引(PRIMARY KEY)：建立在主键上的索引被称为主键索引，一张数据表只能有一个主键索引，索引列值不允许有空值，不允许重复,通常在创建表时一起创建。

唯一索引(UNIQUE)：建立在UNIQUE字段上的索引被称为唯一索引，一张表可以有多个唯一索引，索引列值允许为空，不允许重复,注意：列值中出现多个空值不会发生重复冲突。

普通索引(INDEX)：建立在普通字段上的索引被称为普通索引。

全文索引(FULLTEXT)：MyISAM 存储引擎支持Full-text索引，用于查找文本中的关键词，而不是直接比较是否相等。Full-text索引一般使用倒排索引实现，它记录着关键词到其所在文档的映射。

MySQL 5.6 以前的版本，只有 MyISAM 存储引擎支持全文索引；
MySQL 5.6 及以后的版本，MyISAM 和 InnoDB 存储引擎均支持全文索引；
只有字段的数据类型为 char、varchar、text 及其系列才可以建全文索引。

索引弊端

友情提醒:虽然索引可以加快查询速度，提高 MySQL 的处理性能，但是过多地使用索引也会造成以下弊端：
1.创建索引和维护索引要耗费时间，这种时间随着数据量的增加而增加。
2.除了数据表占数据空间之外，每一个索引还要占一定的物理空间。如果要建立聚簇索引，那么需要的空间就会更大。
3.当对表中的数据进行增加、删除和修改的时候，索引也要动态地维护，这样就降低了数据的维护速度。
4.对于那些在查询中很少使用或参考的列不应该创建索引。因为这些列很少使用到，所以有索引或者无索引并不能提高查询速度。相反，由于增加了索引，反而降低了系统的维护速度，并增大了空间要求。

添加和删除索引

```
# 演示索引的创建和删除
# 创建表
create table student_index(
    id int,
    name varchar(100) ,
    age int ,
    gender enum('男','女','妖'),  # 单选框
    hobby set('吃饭','学习','打豆豆'), # 多选框
    birthday datetime
);
# 查看表结构
desc student_index;
# 查看表中索引
show index from student_index; -- 首次查看为空

#  1.添加各种索引
# 添加主键约束(主键索引)
alter table student_index add primary key (id);
# 添加唯一约束(唯一索引)
# 方式1: change方式添加约束 : 默认字段名就是索引名
alter table student_index change name name varchar(100) unique ;
# 方式2: create方式创建索引: 可以给索引起名
create unique index test_age_i on student_index(age);
# 添加普通索引
create index test_gender_i on student_index(gender);
# 添加全文索引: 只能给字符串类型的字段添加
create fulltext index test_name_i on student_index(name);
```

```
# 添加组合索引: 查询的时候需要遵循最左前缀原则,否则不会生效
create  index test_hobby_birthday_i on student_index(hobby,birthday);

# 添加完索引后查看表中索引
show index from student_index;

# 2.删除各种索引
# 删除组合索引
drop index test_hobby_birthday_i on student_index;
# 删除全文索引
drop index test_name_i on student_index;
# 删除普通索引
drop index test_gender_i on student_index;
# 删除唯一索引
drop index test_age_i on student_index;
drop index name on student_index;
# 删除主键索引
alter table student_index drop primary key ;

# 删除完索引后查看表中索引
show index from student_index;
```

6.mysql视图

视图概述

视图:（view）是一个虚拟表，非真实存在，其本质是根据SQL语句获取动态的数据集，并为其命名，用户使用时只需使用视图名称即可获取结果集，并可以将其当作表来使用。

注意:数据库中只存放了视图的定义，而并没有存放视图中的数据。这些数据存放在原来的表中。

使用视图查询数据时，数据库系统会从原来的表中取出对应的数据。因此，视图中的数据是依赖于原来的表中的数据的。一旦表中的数据发生改变，显示在视图中的数据也会发生改变

视图作用: 简单 安全 数据独立

总而言之，使用视图的大部分情况是为了保障数据安全性，提高查询效率。

视图操作

```
创建视图:create view 视图名 as select语句;

删除视图:drop view 视图名;

修改视图数据:alter view 视图名 as select语句;
修改视图名:rename table 旧视图名 to 新视图名;

查询视图:select * from 视图名;
查看表视图和对应类型:show full tables ;
查看视图字段信息:desc 视图名;
```

```
# 演示视图操作
# 查看湖南省下所有的城市
create view day04.hbs_areas as
```

```sql
select * from day04.areas where pid = (select id from day04.areas where title='河北省');
# 创建视图后如果以后还想查询湖南省的所有城市
# 查询视图
select * from day04.hbs_areas;

# 查看株洲市的基本信息
create view day04.handan_areas as
select * from day04.areas where title='邯郸市';

# 删除视图
drop view day04.handan_areas;

# 修改视图数据
alter view day04.hbs_areas as select * from day04.areas where title='河北省';

# 修改视图名
rename table day04.hbs_areas to day04.areas_hbs;

# 查询视图
select * from day04.areas_hbs;

# 查看所有表包含视图
use day04;
show tables;
# 查看类型信息需要加full
show full tables;
# 查看视图中字段信息
desc day04.areas_hbs;
```