Python

Github：https://github.com/IvenStarry

学习视频网站：B站黑马程序员 https://www.bilibili.com/video/BV1qW4y1a7fU?p=1&vd_source=6fd71d34326a08965fdb07842b0124a7

第十四章

SQL语言 数据库介绍

数据的存储：txt、excel、数据库 管理库、管理表、管理数据 数据库就是数据存储的库，是组织数据并存储数据 SQL语言是对数据库、数据进行操作、管理、查询的工具 使用数据库软件去获得库->表->数据，提供数据组织、存储的能力 使用SQL语言，完成数据的增删改查等操作

MySQL的入门使用

show databases; 查看有哪些数据库 use 数据库名; 使用某个数据库 show tables; 查看数据库里有哪些表 exit 退出MySQL的命令行环境 图形化工具：DBeaver

SQL基础和数据定义语言DDL

数据定义：DDL **数据操纵**：DML **数据控制**：DCL **数据查询**：DQL

```
# 注释：1.单行注释 -- 注释 2.单行注释 # 注释 3.多行注释 /* 注释/*
-- 注释1
# 注释2
/*
 我
 是
 注释
*/

# SQL语言特点：大小写不敏感，  ;结尾
# 查看数据库 show databases
show databases;

# 使用数据库 use 数据库名称
use world;

# 查看当前数据库 select database();
select database();

# 创建数据库 create database test [charset UTF8] []内可选编码格式
create database test charset utf8;
show databases;

# 删除数据库 drop database 数据库名称
drop database test;
show databases;
```

```
use world;
# 查看表 show tables;
show tables;

# 创建表 create table 表名称(列名称 列类型,列名称 列类型,.....)
-- 列类型: 1.int 2.float 3.varchar(长度) 文本，长度为数字，做最大长度限制（py中的字符串） 4.date 日期类型 5.timesleep  时间戳类型
create table student(
    id int,
    name varchar(10),
    age int
);

# 删除表 drop table 表名称; drop table if exists 表名称;
drop table student;
```

数据操纵语言DML

```
use world;

drop table student;

create table student(
    id int,
    name varchar(10),
    age int
);

# 插入数据 insert into 表[(列1, 列2,.....,列N)] values(值1, 值2,.....)[, (值1, 值2,.....), (值1, 值2,.....)]
insert into student(id) values(1), (2), (3);

# 字符串只支持单引号
insert into student(id, name, age) values(4, 'Iven', 11), (5, 'rosen', 21), (6, 'starry', 41);

# []代表可省略 全部列插入 输入数据顺序保持一样
insert into student values(7, 'bob', 11), (8, 'alen', 21);

# 数据删除 delete from 表名称 (where 条件判断) 不选where则全部删除
delete from student where id = 1;
delete from student where id < 4;
delete from student where id > 6;

delete from student where age = 21;

delete from student ;

# 数据更新 update 表名 set 列=值 [where 条件判断]
```

```
update student set name = 'Ivennn' where id = 4;
# 不选where则全部更新
update student set name = 'Ivennn';

# 课后作业
create table student_test(
    id int,
    name varchar(10),
    age int,
    gender varchar(10)
);

insert into student_test values(10001, 'Iven', 22, '男'), (10002, 'rosenn', 26,
'男'), (10003, 'bob', 21, '女');
```

数据查询语言DQL_基础查询

```
create table student(
    id int,
    name varchar(10),
    age int,
    gender varchar(10)
);

INSERT INTO student VALUES(10001, '周杰轮', 31, '男'), (10002, '王力鸿', 33, '男'),
(10003, '蔡依琳', 35, '女'), (10004, '林志灵', 36, '女'), (10005, '刘德滑', 33,
'男'), (10006, '张大山', 10, '男'), (10007, '刘志龙', 11, '男'), (10008, '王潇潇',
33, '女'), (10009, '张一梅', 20, '女'), (10010, '王一倩', 13, '女'), (10011, '陈一
迅', 31, '男'), (10012, '张晓光', 33, '男'), (10013, '李大晓', 15, '男'), (10014,
'吕甜甜', 36, '女'), (10015, '曾悦悦', 31, '女'), (10016, '刘佳慧', 21, '女'),
(10017, '项羽凡', 23, '男'), (10018, '刘德强', 26, '男'), (10019, '王强强', 11,
'男'), (10020, '林志慧', 25, '女');

# select 字段列表 |* from 表 [where 条件判断]
# 从表中，选择某些列进行展示
select id, name from student;
select id, name, age, gender from student;
# * 代表查询所有
select * from student;
select * from student where age > 20;
select * from student where gender = '男';
```

数据查询语言DQL_分组聚合

```
# 分组聚合
# select 字段|聚合函数 from 表 [where 条件] group by 列
# 聚合函数：sum(列)，avg(列)，min(列)，max(列)，count(列|*)
# select后非聚合函数只能有跟group by一样的对象 聚合函数无限制
```

```
select gender, avg(age), sum(age), min(age), max(age), count(*) from student group
by gender;
```

数据查询语言DQL_排序分页

```
# 结果排序 对某一列进行排序
# select 列|聚合函数|* from 表 order by ...[asc| desc]默认升序
select * from student where age > 20;
select * from student where age > 20 order by age asc;
select * from student where age > 20 order by age desc;

# 结果分页限制
# select 列|聚合函数|* from 表 limit n[, m]
select * from student limit 5;
# 跳过前10条从第11条开始 取5个
select * from student limit 10, 5;

# 综合使用 执行顺序不可变 select from必写 其他可省略
# select 列|聚合函数|* from 表 where... group by... order by ...[asc| desc] limit
n[, m]...
select age, count(*) from student where age > 20 group by age;
select age, count(*) from student where age > 20 group by age;
order by age;
select age, count(*) from student where age > 20 group by age;
order by age limit 3;
```

Python操作MySQL基础使用

```
# pymysql 除了使用图形化工具 也可以用编程语言执行SQL来操作数据库

# todo 创建到MySQL的数据库链接
from pymysql import Connection
conn = Connection(
    host = "localhost",         # 主机名(IP)
    port = 3306,                # 端口
    user = "root",              # 账户
    password = "123456"         # 密码
)
# 获取服务器连接信息
print(conn.get_server_info())

# todo 创建到MySQL的数据库链接
# 获取游标对象 conn.cursor()
cursor = conn.cursor()
# 选择数据库 use = select_db
conn.select_db("test")
# 执行SQL 用python语言可以不写;
cursor.execute("create table test_pymysql(id int);")
cursor.execute("create table test_pymysql2(id int)")
```

```
# todo 执行查询性质的SQL语句
conn.select_db("world")
cursor.execute("select * from student")
# 拿到查询结果 fetchall() 返回了一个嵌套元组
results = cursor.fetchall()
print(results)
for r in results:
    print(r)

# todo 关闭链接
conn.close()
```

Python操作MySQL数据插入

```
from pymysql import Connection

conn = Connection(
    host="localhost",
    port=3306,
    user="root",
    password="123456"
)

# 游标对象
cursor = conn.cursor()
conn.select_db("world")
# 插入数据
cursor.execute("insert into student values(10001, 'Iven', 23, '男')")
# * 插入数据必须要commit确认
conn.commit()
cursor.close()

# 若不想手动确认，可以在构建连接对象时，添加一个参数
conn = Connection(
    host="localhost",
    port=3306,
    user="root",
    password="123456",
    autocommit=True
)

cursor = conn.cursor()
conn.select_db("world")
# 自动插入数据
cursor.execute("insert into student values(10002, 'Iven', 23, '男')")
conn.close()
```

Python操作MySQL综合案例

```
import json
from pymysql import Connection

class Record:
    def __init__(self, date, order_id, money, province):
        self.date = date
        self.order_id = order_id
        self.money = money
        self.province = province

    def __str__(self):
        return f"{self.date}, {self.order_id}, {self.money}, {self.province}"

class FileReader:
    # 但这里不能正常注释Record类 最好还是纯英文文件
    # def read_data(self) -> list[Record]:
    def read_data(self) -> list:
        """读取文件数据 读的数据转为Record对象 并封装在list内返回即可"""
        pass

class TextFileReader(FileReader):
    def __init__(self, path):
        self.path = path

    # 复写
    # def read_data(self) -> list[Record]:
    def read_data(self) -> list:
        f = open(self.path, "r", encoding="UTF-8")

        record_list = []

        for line in f.readlines():
            # 消除换行符
            line = line.strip()
            data_list = line.split(",")
            record = Record(data_list[0], data_list[1], int(data_list[2]),
data_list[3])
            record_list.append(record)

        f.close()

        # 返回数据列表
        return record_list

class JsonFileReader(FileReader):
    def __init__(self, path):
        self.path = path

    # 复写
    # def read_data(self) -> list[Record]:
    def read_data(self) -> list:
        f = open(self.path, "r", encoding="UTF-8")
```

```
        record_list = []

        for line in f.readlines():
            # 消除换行符 这里可要可不要
            # line = line.strip()
            data_dict = json.loads(line)
            record = Record(data_dict['date'], data_dict['order_id'],
int(data_dict['money']), data_dict['province'])
            record_list.append(record)

        f.close()

        # 返回数据列表
        return record_list

text_file_reader = TextFileReader("related_data/2011年1月销售数据.txt")
json_file_reader = JsonFileReader("related_data/2011年2月销售数据JSON.txt")

jan_data: list[Record] = text_file_reader.read_data()
feb_data: list[Record] = json_file_reader.read_data()
# 合并为一个list储存
all_data : list[Record] = jan_data + feb_data

# todo SQL操作
conn = Connection(
    host="localhost",
    port=3306,
    password="123456",
    user='root',
    autocommit=True
)

cursor = conn.cursor()
conn.select_db("py_sql")
for record in all_data:
    # * 三种长字符串换行显示
    # 1.第一行末尾加\
    sql = f"insert into orders(order_date, order_id, money, province) \
    values('{record.date}', '{record.order_id}', {record.money},
'{record.province}')"
    # 2.三个"
    sql = f'''insert into orders(order_date, order_id, money, province) \
    values("{record.date}", "{record.order_id}", {record.money}, "
{record.province}")'''
    # 3.三个'
    sql = f"""insert into orders(order_date, order_id, money, province) \
    values('{record.date}', '{record.order_id}', {record.money},
'{record.province}')"""
    # print(sql)
    cursor.execute(sql)
conn.close()

# 课后作业
f = open("related_data/098_2011年2月销售数据JSON.txt", "a", encoding="UTF-8")
```

```
conn = Connection(
    host="localhost",
    port=3306,
    password="123456",
    user='root',
    autocommit=True
)
cursor = conn.cursor()
conn.select_db("py_sql")
cursor.execute("select * from orders")
# 返回一个元组
results = cursor.fetchall()
# print(results)

# 作业所示的文件并不是正式的json格式 所以以字典转str格式写入
# data_list = []
for data in results:
    # 从SQL里拿出的日期是datetime.date类型
    year_month = str(data[0])[:7]
    if year_month == "2011-02":
            order_date = str(data[0])
            order_id = data[1]
            money = data[2]
            province = data[3]

            data_dict = {}
            data_dict['date'] = order_date
            data_dict['order_id'] = order_id
            data_dict['money'] = money
            data_dict['province'] = province

            str_data_dict = str(data_dict) + "\n"
            f.write(str_data_dict)
            # data_list.append(data_dict)

# json_list = json.dumps(data_list, ensure_ascii=False)
# f.write(json_list)

f.flush()
f.close()
conn.close()
```

第十五章

spark是全球顶级的分布式计算框架，用于对海量数据进行大规模分布式计算，支持众多的编程语言开发，是大数据开发的核心技术。 作为python库进行数据处理或提交到spark集群进行分布式集群计算