**数据库系统project报告**

2022-2023学年第2学期（CST21118）

|  |  |
| --- | --- |
| 数据库系统project任务书 | |
| 名称 | 数据库SQL执行器设计与模拟实现 |
| 类型 | □验证性 □设计性 综合性 |
| 内容 | 针对关系型数据库SQL语句执行，进行需求分析，并设计一个简单的数据库系统SQL执行器模块，根据设计模拟实现SQL基本操作等，主要实现：数据库创建、表格创建、数据添加、删除、更新、查询等操作。  模拟实现采用：python或者Java实现具体功能，设计中若有数据的存储可以使用文本文件或者excel文件，SQL语句执行采用函数实现。 |
| 要求 | （1）设计方案要合理；  （2）能基于该SQL执行器模块完成SQL的模拟实现；  （3）设计方案有一定的效率分析。 |
| 任务时间 | 2023年4月17日至2023年5月20日 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 小组成员 | | | | | | |
| 易千喜 | | 谭俊枫 | | 韩昊辰 | | 杨松鸣 |
| 20214090 | | 20214220 | | 20214272 | | 20215314 |
| 项目评分表 | | | | | | |
| 序号 | 评分项 | | 分值 | | 得分 | |
| 1 | 需求分析 | | 3分 | |  | |
| 2 | 综合设计与实现 | | 4分 | |  | |
| 3 | 团队协作 | | 3分 | |  | |
| 项目总得分： | | | | | | |

课程项目评分标准（总分10分）

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 评分项目 | 完成情况 | 得分 |
| 1 | 需求分析 | 分析合理 | 3分 |
| 分析较合理 | 2分 |
| 分析不合理 | 1分 |
| 分析完全错误 | 0分 |
| 2 | 综合设计与实现 | 设计完整，设计合理，工具使用熟练 | 4分 |
| 设计较完整，设计合理，工具使用较熟练 | 3分 |
| 设计较完整，设计较合理，工具使用较熟练 | 2分 |
| 设计较完整，设计不合理，工具使用不熟练 | 1分 |
| 抄袭、被抄袭 | 0分 |
| 3 | 团队协作 | 有团队，分工合理，密切协作 | 3分 |
| 有团队，分工合理，有一定协作 | 2分 |
| 有团队，分工不合理，无协作 | 1分 |
| 无团队，无协作 | 0分 |

1. **小组讨论和会议纪要**

**1.** **第一次讨论**

讨论时间：2023年4月28日 参会人数：4人，缺席人数：0人

**（一）主要讨论内容**

1.为了完成本project，小组成员首先进行了需求分析；

2.研究和探讨此次数据库project的大致思路。

**（二）主要决定项**

1.按照讨论所得的结果，小组成员按照该项目的需求进行相应的分析；

2.共同进行有关python中xlwings的相关操作。

**2. 第二次讨论**

讨论时间：2023年5月8日 参会人数：4人，缺席人数：0人

**（一）主要讨论内容**

1.基本sql语法实现；

2.讨论使用正则化作为交互。

**（二）主要决定项**

1.针对该数据库项目所需要实现的创建和删除库和表、增加数据、删除数据、查询数据、修改数据等相关功能，进行基本sql语法的实现；

2.讨论使用正则化进行用户交互操作，将对应字符串或数据传入实例化的对象，以执行对应逻辑的sql语句。

**3. 第三次讨论**

讨论时间：2023年5月16日 参会人数：4人，缺席人数：0人

**（一）主要讨论内容**

1.外码约束的实现；

**（二）主要决定项**

1.小组成员共同决定在基本sql语法的基础上，进行对外键约束的实现，从而实现该程序数据库功能的不断完善。

**4. 第四次讨论**

讨论时间：2023年5月23日 参会人数：4人，缺席人数：0人

**（一）主要讨论内容**

1. 着手完成实验报告的撰写；

2. 对该程序实现各项数据库功能进行效率分析；

3. 对该数据库程序的各项功能进项验证和记录。

**（二）主要决定项**

1.会后分模块共同撰写完成实验报告。

1. **需求分析**

**1. 数据库的相关存储：**

本项目要求我们需要支持的关系型数据库类型。我们小组选择调用python中自带的xlwings库，连接excel作为数据存储的容器，从而实现“创建、增、删、查、改”等关系在数据库中的持久化存储。

**2. 数据库连接与断开：**

本项目要求我们实现数据库连接的建立和关闭功能，以确保在执行SQL操作时能够正确地连接到数据库并在操作完成后断开连接。

**3.正则化处理：**

用户交互界面，使用python中的re模块进行正则化处理，将本次项目所要实现的sql语句划分为对应的正则化字符串模式，然后用while和input（）函数，让用户不断输入需要执行的sql语句，然后将字符串匹配对应的正则化模式，再将字符串中的所有有效字符串进行划分并储存为列表，然后将对应字符串或数据传入实例化的DataBase类的db对象，以执行对应逻辑的sql语句。

**4.外键约束：**

为了更好的模拟数据库的真实情况，本研究计划为本数据库添加外键。在增加了外键约束之后，该模拟数据库的增、删、改、创建等方面的相应操作则会存在限制，从而更加符合模拟数据库的要求。

1. **sql语句执行：**

对于数据库的创建、增、删、查、改，输入经过正则化处理后，能很好的模拟sql语句的输入。按要求可以实现增、删、查、改、创建和删除数据库都需要相应的外码约束检测，根据条件，可以一条或多条数据的删除、查询、修改等sql中所要求的操作。

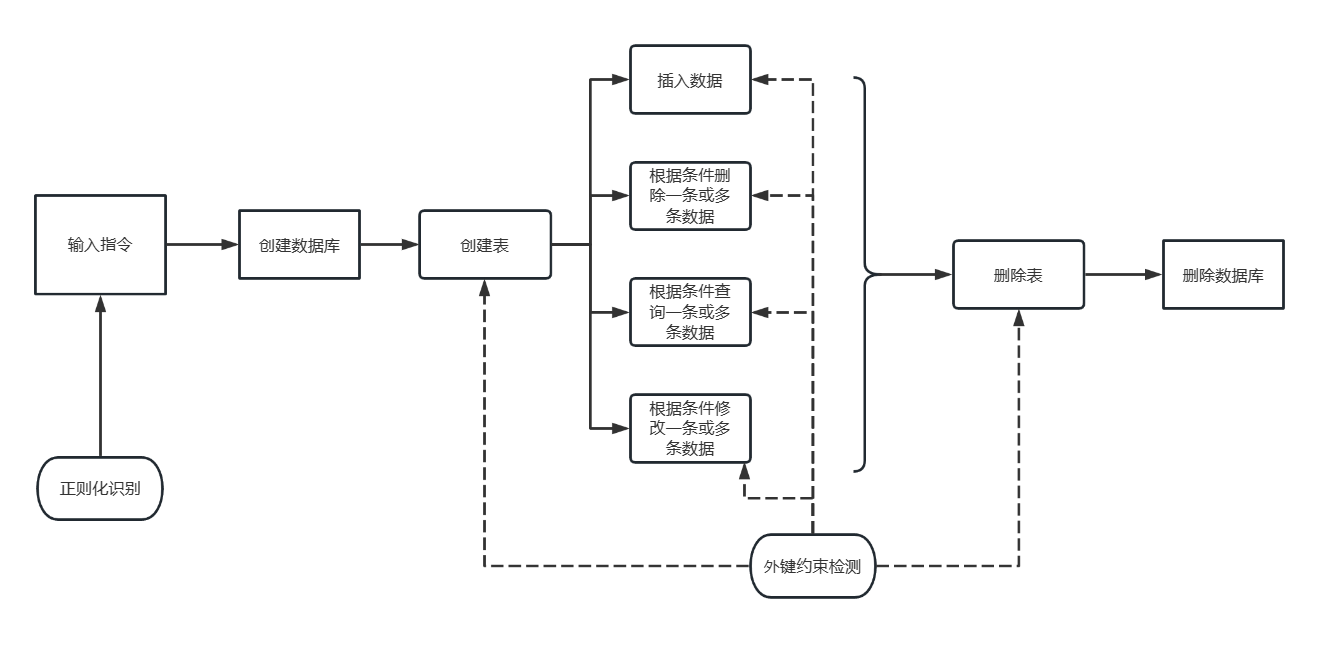
**6.错误处理和异常情况：**

在执行SQL语句的函数中，对处理数据库连接失败、SQL语句执行失败等异常情况，提供适当的错误处理机制，捕获数据库操作可能抛出的异常，并根据需要进行错误处理，例如。当数据库程序的增、删、改等操作会与外键约束相冲突时，则会提示相应的外键异常；试图创建一个与现有表同名的表，则提示相应异常；或者查找一条数据库中不存在的数据，也就是没有主键匹配的数据，则打印“没有找到相应属性”等等。

**6. 结果的输出：**

验证数据库系统SQL执行器模块的各种功能是否正常工作，测试创建遇删除数据库、创建和删除表格，增、删、查、该相应的数据，外键约束等操作，并检查执行结果是否符合预期，返回执行结果，例如返回查询、更新等操作的结果或者异常处理的错误信息。

**需求分析流程图如下：**



1. **代码展示**

**3.1初始化代码**

def \_\_init\_\_(self, db\_name: str):

        # 创建数据库模块 数据库名

        self.field\_col = ['C', 'D', 'E', 'F',

                          'G', 'H', 'I', 'J',

                          'K', 'L', 'M', 'N',

                          'O', 'P', 'Q', 'R',

                          'S', 'T', 'U', 'V']

        self.ff\_col = ['A', 'B', 'C', 'D', 'E', 'F',

                       'G', 'H', 'I', 'J',

                       'K', 'L', 'M', 'N',

                       'O', 'P', 'Q', 'R',

                       'S', 'T', 'U', 'V']

        self.length\_col = 'W'

        try:

            self.path = f'{db\_name}.xlsx'

            self.wb = xw.Book(self.path)

        except FileNotFoundError:

            self.wb = xw.Book()

            self.wb.save(self.path)

            print(f"已创建数据库：{db\_name}.xlsx")

        self.config = self.wb.sheets["Sheet1"]

        try:

            \_ = self.wb.sheets["cons"]

        except:

            self.wb.sheets.add("cons")

        self.cons = self.wb.sheets["cons"]

**3.2删表操作**

def drop\_table(self, table\_name: str):

        # 删除表模块 表名

        row = 2

        while True:

            x = self.cons.range('A' + str(row)).value

            if x is None:

                break

            if (x==table\_name):

                print("存在外键约束，无法删除！")

                return

            else:

                row = row + 1

        table=self.wb.sheets[table\_name]

        table.delete()

        self.delet\_con1(table\_name)

        self.wb.save()

        print(f"{table\_name} 已经成功删除")

        return

**3.3建表操作**

    def create\_table(self, table\_name: str, pr\_key: str, fk\_col: list[str], fk\_table: list[str], field: list[str]):

        # 创建表模块 表名，主码，外码中的属性约束，外码中的对应的表约束，属性（不算入主码）

        table\_length = len(field)

        if len(field) > 15:

            print("create table: 属性数量超过十三个了，已超过最大上限！")

            return

        try:

            \_ = self.wb.sheets[table\_name]

            print("create table: 已经存在同名表了！")

        except:

            rows = self.config.used\_range.last\_cell.row

            self.config.range('A' + str(rows + 1)).value = table\_name

            self.config.range('B' + str(rows + 1)).value = pr\_key

            for i in range(len(field)):

                self.config.range(self.field\_col[i] + str(rows + 1)).value = field[i]

            self.config.range(self.length\_col + str(rows + 1)).value = table\_length

            if(len(fk\_col)!=0):

                for i in range(len(fk\_col)):

                    try:

                        \_ = self.wb.sheets[fk\_table[i]]

                        # 将外键列和外键对应的表名添加到表格中

                        self.config.range(self.field\_col[i+16] + str(rows + 1)).value = fk\_table[i]

                        self.config.range(self.field\_col[i+12] + str(rows + 1)).value = fk\_col[i]

                    except:

                        # 如果外键对应的表不存在，则打印错误消息并返回

                        print("外码相关的表不存在，请重新尝试！")

                        return

            self.wb.sheets.add(table\_name)

            self.wb.save()

            print(f"create table: {table\_name} 创建成功...")

**3.4插入操作**

def insert(self, table\_name, pr\_key, field):

        # 插入模块 表名，主码，属性（不算主码）

        try:

            table = self.wb.sheets[table\_name]

        except:

            print("insert: 没有该表!")

            return

        if not self.check\_field(table\_name, field):

            return

        row\_=2

        while(True):

            if(self.config.range('A'+str(row\_)).value==None):

                print("找不到该表!")

                return

            elif(self.config.range('A'+str(row\_)).value==table\_name):

                break

            else:

                row\_=row\_+1

        table\_mem=[] # 哪些表跟我当前表有外键约束

        idx\_mem=[]

        for i in range(3):

            if(self.config.range(self.field\_col[i+16] + str(row\_)).value!=None):

                table\_mem.append(self.config.range(self.field\_col[i+16] + str(row\_)).value)

                idx\_mem.append(self.config.range(self.field\_col[i+12] + str(row\_)).value)

            else:

                break

        for i in range(len(table\_mem)): # 一个一个约束验证

            mem\_ck = self.check\_pr(table\_mem[i])

            flag=False

            for j in range(12): # 上限是12个，找到对应的列的idx赋值给j

                if(idx\_mem[i]==self.config.range(self.field\_col[0+j] + str(row\_)).value):

                    break

            ys\_value=field[j]

            for value\_ in mem\_ck:

                if(ys\_value==value\_):

                    flag=True

                    self.insert\_con(table\_mem[i], value\_, table\_name, pr\_key)

                    break

            if(flag==True):

                continue

            else:

                print("外码约束出错")

                return

        row = 1

        while True:

            x = table.range('B' + str(row)).value

            if x is None:

                table.range('B' + str(row)).value = pr\_key

                for i in range(len(field)):

                    table.range(self.field\_col[i] + str(row)).value = str(field[i])

                self.wb.save()

                print(f"insert: 插入成功！pr\_key={pr\_key}")

                return

            if x == pr\_key:

                print(f"insert: 已经存在相同主键{pr\_key}")

                return

            else:

                row = row + 1

**3.5修改操作的顶层模块**

def update(self, table\_name: str, SX, SX\_v, update\_SX: dict, type='=') -> None:

        # SX:属性名称，SX\_v:属性取值，cond：条件，update\_SX:需要更新的属性,type：条件类型(1:== 2:> 3:< 4:!=）

        # update table\_name

        # set update\_SX

        # where SX cond(type) SX\_v

        print('update: 更新', SX, type, SX\_v)

        try:

            table = self.wb.sheets[table\_name]

        except:

            print("update: 没有该表")

            return

        # 判断属性所在列以及更新的属性所在列

        update\_col = {}

        Col = 'Z'

        rows = self.config.used\_range.last\_cell.row

        for i in range(rows):

            if self.config.range('A' + str(i + 1)).value == table\_name:

                field\_length = int(self.config.range(self.length\_col + str(i + 1)).value)

                for j in range(field\_length + 1):

                    col\_tmp = chr(66 + j)

                    print(self.config.range(col\_tmp + str(i + 1)).value)

                    if SX == self.config.range(col\_tmp + str(i + 1)).value:

                        Col = col\_tmp

                    for sx in update\_SX.keys():

                        if sx == self.config.range(col\_tmp + str(i + 1)).value:

                            update\_col[col\_tmp] = sx

        if (Col == 'Z'):

            print("select: 没有找到属性")

            return

        # 如果更新了主码，判断主码是否和已有数据主码重复

        if 'B' in update\_col.keys():

            row = 1

            while True:

                x = table.range('B' + str(row)).value

                if x is None:

                    break

                if x == update\_SX[update\_col['B']]:

                    print(f"insert: 主码{x}重复")

                    return

                row += 1

        row = 1

        flag = 0  # 是否找到属性

        while True:

            x = table.range(Col + str(row)).value

            if x is None:

                if flag == 0:

                    print("update: 没有找到属性值")

                return

            if cond(x, SX\_v, type):

                # 主码不可重复

                if flag == 1 and 'B' in update\_col.keys():

                    print(f"update: 主码重复")

                    return

                flag = 1

                update\_row = []

                pr\_key\_old = table.range('B' + str(row)).value

                # 单独判断主码

                if table.range('B' + str(row)).value is None:

                    break

                if 'B' in update\_col.keys():

                    update\_row.append(update\_SX[update\_col['B']])

                    # table.range('B' + str(row)).value = update\_SX[update\_col['B']] # 更改excel里的数据

                    print('update: 修改数据')

                else:

                    update\_row.append(table.range('B' + str(row)).value)

                for i in range(len(self.field\_col)):

                    if table.range(self.field\_col[i] + str(row)).value is None:

                        break

                    if self.field\_col[i] in update\_col:

                        update\_row.append(update\_SX[update\_col[self.field\_col[i]]])

                        # table.range(self.field\_col[i] + str(row)).value = update\_SX[update\_col[self.field\_col[i]]]

                        print('update: 修改数据')

                    else:

                        update\_row.append(table.range(self.field\_col[i] + str(row)).value)

                print(update\_row)

                self.update1(table\_name,pr\_key\_old,update\_row)

            row += 1

**3.6删除数据库**

    def drop\_database(self, db\_name):

        self.close()

        # 删除excel文件

        os.remove(f'{db\_name}.xlsx')

        print('删除成功！')

**3.7修改操作的底层模块**

def update1(self, table\_name: str, pr\_key, field: list) -> None:

        # 更新模块，表名， 主码 ， 属性（注意：！！！！！这里的属性中是包含了主码的，主码位于第0个）

        # 此处得field是以主码pr\_key为第0个，其他的接在后面的新列表

        try:

            table = self.wb.sheets[table\_name]

        except:

            print("update: 没有该表")

            return

        if not self.check\_field(table\_name, field[1:]):

            return

        # 如果主码存在被更改的情况

        if (pr\_key!=field[0]):

            row=2

            while (True):

                x=self.cons.range("A"+str(row)).value

                y=self.cons.range("B"+str(row)).value

                if(x==None):

                    break

                row+=1

                if(x==table\_name and y==pr\_key):

                    print("当前主码存在外码约束限制, 更新失败")

                    return

            # 对于主码更改的情况，直接使用删除再添加的方式实现

            self.delete(table\_name, pr\_key)

            self.insert(table\_name, field[0], field[1: ])

            self.wb.save()

            return

        # 对于其他情况，则是当前某个属性依赖了其他属性，所以对有依赖的属性进行外码判断

        # 首先读出所有有依赖的属性

        row\_=2

        while(True):

            if(self.config.range('A'+str(row\_)).value==None):

                print("找不到该表!")

                return

            elif(self.config.range('A'+str(row\_)).value==table\_name):

                break

            else:

                row\_=row\_+1

        # 好像不需要这个

        roww=1

        field\_past=[]

        while(True):

            if(table.range("B"+str(roww)).value==None):

                print("原数据不存在")

                return

            elif(table.range("B"+str(roww)).value==pr\_key):

                for i in range(int(self.config.range(self.length\_col+str(row\_)).value)):

                    field\_past.append(table.range(self.field\_col[i]+str(roww)).value)

                break

            else:

                roww +=1

        table\_mem=[] # 哪些表跟我当前表有外键约束

        idx\_mem=[]

        for i in range(3):

            if(self.config.range(self.field\_col[i+16] + str(row\_)).value!=None):

                table\_mem.append(self.config.range(self.field\_col[i+16] + str(row\_)).value)

                idx\_mem.append(self.config.range(self.field\_col[i+12] + str(row\_)).value)

            else:

                break

        for i in range(len(table\_mem)): # 一个一个约束验证

            mem\_ck = self.check\_pr(table\_mem[i])

            flag=False

            for j in range(12): # 上限是12个，找到对应的列的idx赋值给j

                if(idx\_mem[i]==self.config.range(self.field\_col[0+j] + str(row\_)).value):

                    break

            if(field[j+1]!=field\_past[j]):

                ys\_value=field[j+1]

                self.delet\_con2(table\_mem[i],table\_name, pr\_key)

                for value\_ in mem\_ck:

                    if(ys\_value==value\_):

                        flag=True

                        self.insert\_con(table\_mem[i], value\_, table\_name, pr\_key)

                        break

                if(flag==True):

                    continue

                else:

                    print("外码约束出错(此时excel未保存，请直接关闭excel文档从而撤销错误操作)")

                    return

        row = 1

        while True:

            x = table.range('B' + str(row)).value

            if x is None:

                print("update: 没有相应的pr\_key")

                return

            if x == pr\_key:

                for i in range(len(self.field\_col)):

                    if table.range(self.field\_col[i] + str(row)).value is None:

                        break

                    table.range(self.field\_col[i] + str(row)).value = field[i+1]

                break

            else:

                row = row + 1

        self.wb.save()

        return

**3.8删除操作的顶层模块**

def delete\_zonghe(self, table\_name: str, SX, SX\_v, type='=') -> None:

        # 删除表模块 表名， where属性 属性值 条件

        try:

            table = self.wb.sheets[table\_name]

        except:

            print("delete: 没有该表")

            return

        print('delete: 删除',SX,type,SX\_v)

        Col = 'Z'

        rows = self.config.used\_range.last\_cell.row

        for i in range(rows):

            if self.config.range('A' + str(i + 1)).value == table\_name:

                field\_length = int(self.config.range(self.length\_col + str(i + 1)).value)

                for j in range(field\_length + 1):

                    col\_tmp = chr(66 + j)

                    if SX == self.config.range(col\_tmp + str(i + 1)).value:

                        Col = col\_tmp

        if (Col == 'Z'):

            print("select: 没有找到属性")

            return

        row = 1

        while True:

            x = table.range(Col + str(row)).value

            if x is None:

                return

            if cond(x,SX\_v,type):

                pr\_k=table.range("B" + str(row)).value

                if(self.delete(table\_name, pr\_k)==1):

                    row += 1

                continue

            row += 1

**3.9删除操作的底层模块**

def delete(self, table\_name: str, pr\_key) :

        # 实际工作的被调用的delete模块

        try:

            table = self.wb.sheets[table\_name]

        except:

            print("delete: 没有该表")

            return

        row=2

        while (True):

            x=self.cons.range("A"+str(row)).value

            y=self.cons.range("B"+str(row)).value

            if(x==None):

                break

            row += 1

            if(x==table\_name and y==pr\_key):

                print("存在外码约束限制, 删除失败")

                return 1

        row = 1

        while True:

            x = table.range('B' + str(row)).value

            if x is None:

                print("delete: 没有相应的pr\_key")

                return 1

            if x == pr\_key:

                # 删除第row行数据

                table.range(f"{str(row)}:{str(row)}").api.Delete()

                self.delet\_con(table\_name,pr\_key)

                self.wb.save()

                print("delete: 已更新")

                return 0

            else:

                row = row + 1

**3.10判断属性个数**

def check\_field(self, table\_name, field):

        # 判断属性个数是否符合要求

        rows = self.config.used\_range.last\_cell.row

        for i in range(rows):

            if self.config.range('A' + str(i + 1)).value == table\_name:

                field\_length = self.config.range(self.length\_col + str(i + 1)).value

                if field\_length != len(field):

                    print(f"字段数量错误! 应该是{field\_length}")

                    return False

                else:

                    return True

**3.11辅助判断外码约束**

    def check\_pr(self, tar\_table):

        # 辅助帮助判断外码约束是否存在

        mem\_ck=[]

        table = self.wb.sheets[tar\_table]

        row = 1

        while True:

            x = table.range('B' + str(row)).value

            if x is None:

                break;

            mem\_ck.append(x)

            row=row+1

        return mem\_ck

    def insert\_con(self, yue\_table:str, fir\_key, bei\_table:str, sec\_key):

        row\_con=self.cons.used\_range.last\_cell.row

        self.cons.range("A" + str(row\_con + 1)).value = yue\_table

        self.cons.range("B" + str(row\_con + 1)).value = fir\_key

        self.cons.range("C" + str(row\_con + 1)).value = bei\_table

        self.cons.range("D" + str(row\_con + 1)).value = sec\_key

        return

    def delet\_con(self, bei\_table, sec\_key):

        #   辅助删除外码记录

        row = 2

        while True:

            x = self.cons.range('C' + str(row)).value

            y = self.cons.range('D' + str(row)).value

            if x is None:

                return

            if x == bei\_table and y == sec\_key:

                self.cons.range(f"{str(row)}:{str(row)}").api.Delete()

            else:

                row = row + 1

    def delet\_con1(self, bei\_table):

        #   辅助删除外码记录

        row = 2

        while True:

            x = self.cons.range('C' + str(row)).value

            if x is None:

                return

            if x == bei\_table:

                self.cons.range(f"{str(row)}:{str(row)}").api.Delete()

            else:

                row = row + 1

    def delet\_con2(self, tar\_table,bei\_table, sec\_key):

        #   辅助删除外码记录

        row = 2

        while True:

            t = self.cons.range('A' + str(row)).value

            x = self.cons.range('C' + str(row)).value

            y = self.cons.range('D' + str(row)).value

            if x is None:

                return

            if t==tar\_table and x == bei\_table and y == sec\_key:

                self.cons.range(f"{str(row)}:{str(row)}").api.Delete()

            else:

                row = row + 1

**3.12正则化实现的模块**

if \_\_name\_\_ == '\_\_main\_\_':

    #use database db\_test

    use\_instr=r"use\b\sdatabase\b\s[a-zA-Z0-9\_]+"

    # create database if not exists db\_name

    create\_db\_instr=r"create\b\sdatabase\b\sif\b\snot\b\sexists\b\s[a-zA-Z0-9\_]+"

    # create table tb\_name (attribute1 type1,attribute2 type2,...,

    #   primary key(attribute),foreign key(attribute1,attribute2,...) references tb\_name(attribute,attribute2,...))

    create\_table\_instr = r"create\b\stable\b\s[a-zA-Z0-9\_]+\s"\

        r"\([a-zA-Z0-9\_]+\s[a-zA-Z0-9\_]+(,[a-zA-Z0-9\_]+\s[a-zA-Z0-9\_]+)\*?" \

        r"(,primary\b\skey\b\([a-zA-Z0-9\_]+\))"\

        r"(,foreign\b\skey\b\([a-zA-Z0-9\_]+\)\sreferences\b\s[a-zA-Z0-9\_]+\([a-zA-Z0-9\_]+\))\*\)"

    # drop database/table db\_name/tb\_name

    drop\_instr=r"drop\b\s(database|table)\b\s[a-zA-Z0-9\_]+"

    # insert into tb\_name values (value1,value2,...)

    insert\_instr=r"insert\b\sinto\b\s[a-zA-Z0-9\_]+\s"\

        r"values\s\((([0-9]\*)|([a-zA-Z0-9\_]\*))?(,([0-9]\*)|([a-zA-Z0-9\_]\*))\*\)"

    # delete from tb\_name where attribute=value

    delete\_instr=r"delete\b\sfrom\b\s[a-zA-Z0-9\_]+\s"\

        r"where\b\s[a-zA-Z0-9\_]+(?P<judge>!?=|[><]=?)[a-zA-Z0-9\_]+"

    # select (atribute1,attribute2,...) from tb\_name where attribute=value

    select\_instr=r"select\b\s\(?[a-zA-Z0-9\_]+(,[a-zA-Z0-9\_]+)\*\)?\s"\

        r"from\b\s[a-zA-Z0-9\_]+\s"\

        r"where\b\s[a-zA-Z0-9\_]+(?P<judge>!?=|[><]=?)[a-zA-Z0-9\_]+"

    # update tb\_name set attribute1=value1,attribute2=value2,... where attribute=value

    update\_instr=r"update\b\s[a-zA-Z0-9\_]+\s"\

        r"set\b\s[a-zA-Z0-9\_]+=[a-zA-Z0-9\_]+(,[a-zA-Z0-9\_]+=[a-zA-Z0-9\_]+)\*\s"\

        r"where\b\s[a-zA-Z0-9\_]+(?P<judge>!?=|[><]=?)[a-zA-Z0-9\_]+"

**3.13 正则化的与数据库操作模块对接的顶层模块**

    db=None

    while(True):

        instr=input("请输入指令：")

        if (instr=="exit"):

            break

        elif re.fullmatch(use\_instr, instr):

            tokens=[i for i in re.split(r"([ ,();=!<>])", instr.strip()) if i not in[' ',',',')','(','','=','!','>','<']]

            #print(tokens)

            db\_name=tokens[tokens.index('database')+1]

            db=Database(db\_name)

            print("当前数据库：",db\_name)

        elif re.fullmatch(create\_db\_instr, instr):

            tokens=[i for i in re.split(r"([ ,();=!<>])", instr.strip()) if i not in[' ',',',')','(','','=','!','>','<']]

            #print(tokens)

            db\_name=tokens[tokens.index('exists')+1]

            db=Database(db\_name)

            print("数据库创建成功！")

        elif re.fullmatch(create\_table\_instr, instr):

            tokens=[i for i in re.split(r"([ ,();=!<>])", instr.strip()) if i not in[' ',',',')','(','','=','!','>','<']]

            #print(tokens)

            tb\_name=tokens[tokens.index('table')+1]

            pr\_key=tokens[tokens.index('primary')+2]

            fk\_num=tokens.count('foreign')

            fk\_col=[]

            fk\_tb=[]

            field=[]

            for i in range(2, 6\*fk\_num, 6):

                fk\_col.append(tokens[tokens.index('foreign')+i])

                fk\_tb.append(tokens[tokens.index('foreign')+i+2])

            for i in range(tokens.index('table')+2,tokens.index('primary'),2):

                field.append(tokens[i])

            field.remove(pr\_key)

            db.create\_table(tb\_name,pr\_key,fk\_col,fk\_tb,field)

            #print(tb\_name,pr\_key,fk\_col,fk\_tb,field)

        elif re.fullmatch(drop\_instr, instr):

            tokens=[i for i in re.split(r"([ ,();=!<>])", instr.strip()) if i not in[' ',',',')','(','','=','!','>','<']]

            #print(tokens)

            if(tokens[1]=='table'):

                tb\_name=tokens[tokens.index('table')+1]

                db.drop\_table(tb\_name)

            else:

                db\_name=tokens[tokens.index('database')+1]

                db.drop\_database(db\_name)

        elif re.fullmatch(insert\_instr, instr):

            tokens=[i for i in re.split(r"([ ,();=!<>])", instr.strip()) if i not in[' ',',',')','(','','=','!','>','<']]

            #print(tokens)

            tb\_name=tokens[tokens.index('into')+1]

            pr\_key=tokens[tokens.index('values')+1]

            field=[]

            for i in range(tokens.index('values')+1,len(tokens),1):

                field.append(tokens[i])

            field.remove(pr\_key)

            #print(tb\_name,pr\_key,field)

            db.insert(tb\_name,pr\_key,field)

        elif re.fullmatch(delete\_instr, instr):

            tokens=[i for i in re.split(r"([ ,();=!<>])", instr.strip()) if i not in[' ',',',')','(','','=','!','>','<']]

            #print(tokens)

            tb\_name=tokens[tokens.index('from')+1]

            SX=tokens[tokens.index('where')+1]

            SX\_v=tokens[tokens.index('where')+2]

            type=re.match(delete\_instr,instr).group('judge')

            #print(tb\_name,SX,SX\_v,type)

            db.delete\_zonghe(tb\_name,SX,SX\_v,type)

        elif re.fullmatch(select\_instr, instr):

            tokens=[i for i in re.split(r"([ ,();=!<>])", instr.strip()) if i not in[' ',',',')','(','','=','!','>','<']]

            #print(tokens)

            tb\_name=tokens[tokens.index('from')+1]

            sel\_SX=[]

            for i in range(tokens.index('select'),tokens.index('from'),1):

                sel\_SX.append(tokens[i])

            SX=tokens[tokens.index('where')+1]

            SX\_v=tokens[tokens.index('where')+2]

            type=re.match(select\_instr,instr).group('judge')

            db.select(tb\_name,sel\_SX=sel\_SX,SX=SX,SX\_v=SX\_v,type=type)

        elif re.fullmatch(update\_instr, instr):

            tokens=[i for i in re.split(r"([ ,();=!<>])", instr.strip()) if i not in[' ',',',')','(','','=','!','>','<']]

            #print(tokens)

            tb\_name=tokens[tokens.index('update')+1]

            #

            # pr\_key\_v=tokens[tokens.index('where')+2]

            # field\_v=[]

            # for i in range(tokens.index('set')+2,tokens.index('where'),2):

            #     field\_v.append(tokens[i])

            SX=tokens[tokens.index('where')+1]

            SX\_v=tokens[tokens.index('where')+2]

            type=re.match(update\_instr,instr).group('judge')

            update\_SX={}

            for i in range(tokens.index('set')+1,tokens.index('where'),2):

                update\_SX[tokens[i]]=tokens[i+1]

            #print(tb\_name,SX,SX\_v,update\_SX,type)

            db.update(tb\_name,SX,SX\_v,update\_SX,type)

        else:

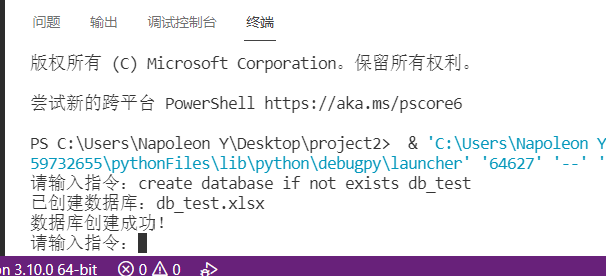
            print("输入指令有误")

1. **功能验证**

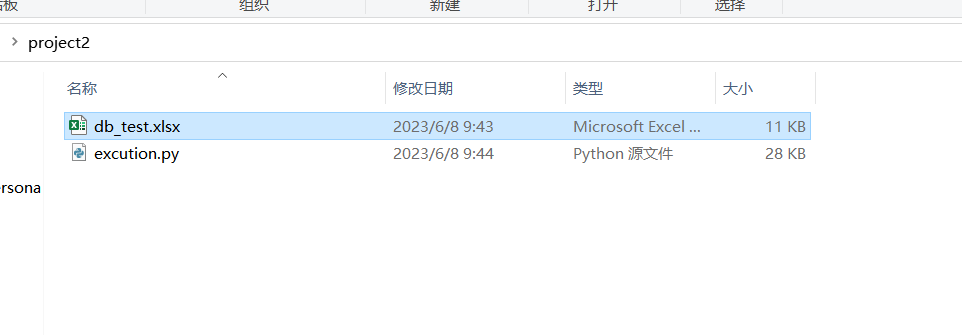
**1.数据库的创建和删除**

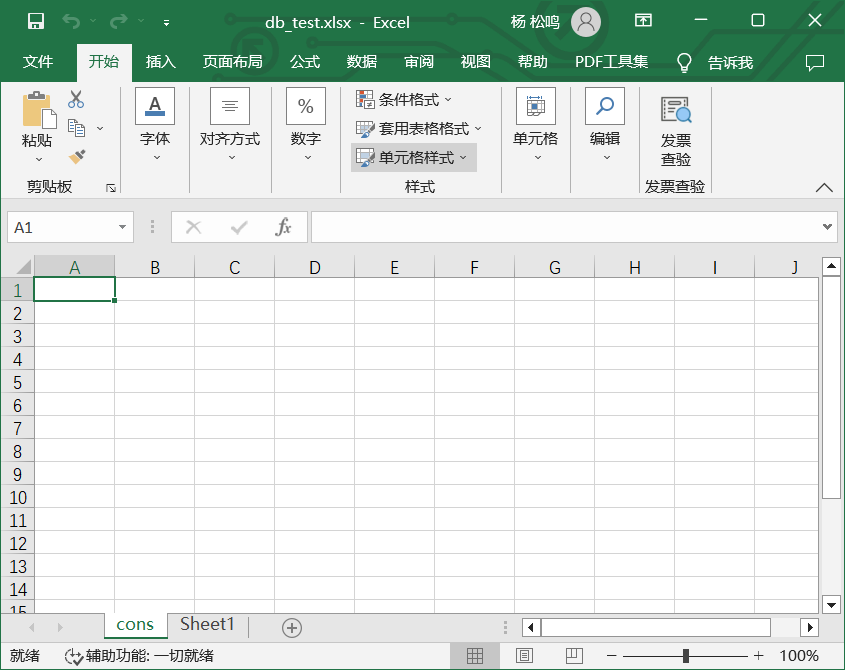
1. **1创建数据库**

输入create database if not exists db\_test指令，创建数据库

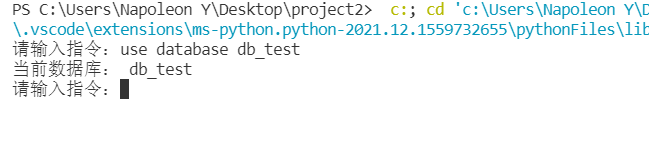


可以发现，在输入相关指令后，excel表形式的数据库创建成功。同时，自动创建了cons外键约束表，用于存储相应的外键约束。



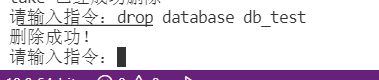


在退出该程序后重新运行该程序进行相应操作时，需要输入use database db\_test指令，以重新调用数据库

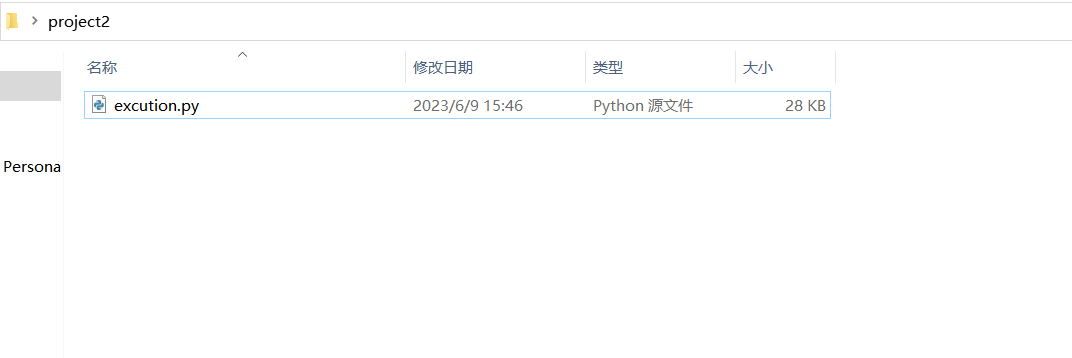


**1.2删除数据库**

输入drop database db\_test，即可删除表

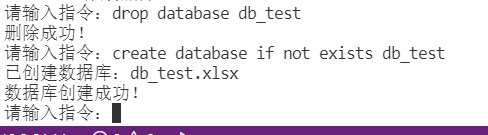


可以发现，在相关文件夹中，名为db\_test的excel表已经被删除

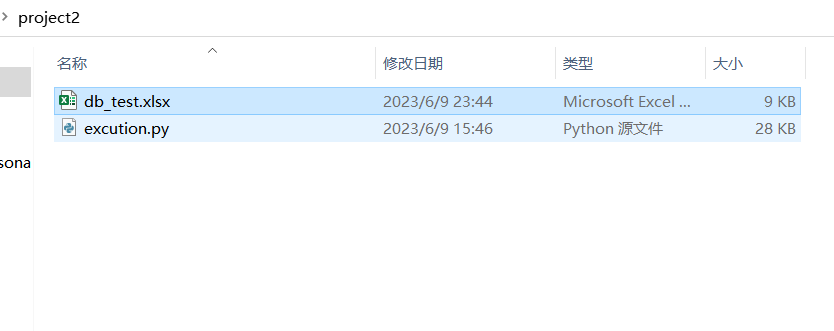


**1.3删除数据库后，重新创建数据库**

再次输入create database if not exists db\_test指令，重新创建数据库



可以发现，在相关文件夹中，重新生成了名为db\_test的新的excel表，说明之前的数据库删除操作时彻底的。



**2.创建表**

**2.1验证没有外码约束，创建成功**

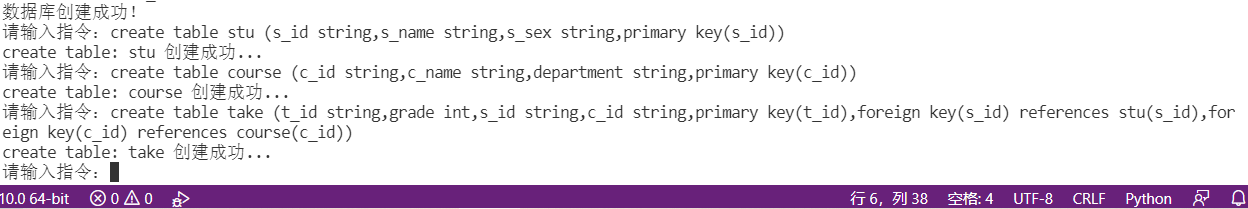
输入：

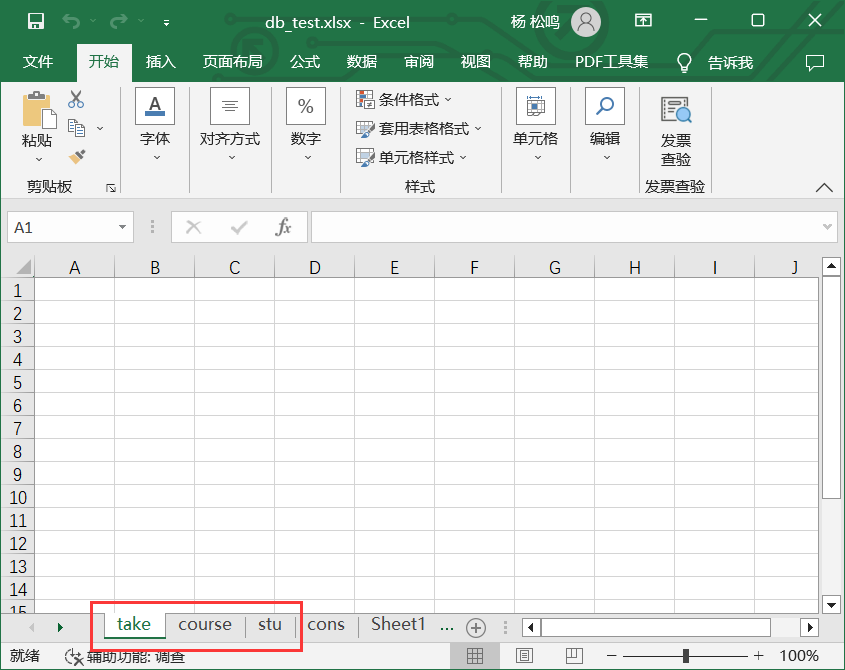
1.create table stu (s\_id string,s\_name string,s\_sex string,primary key(s\_id))

2.create table course (c\_id string,s\_name string,department string,primary key(c\_id))

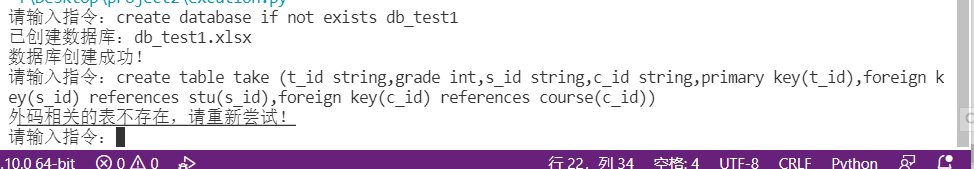
3.create table take (t\_id string,grade int,s\_id string,c\_id string,primary key(t\_id),foreign key(s\_id) references stu(s\_id),foreign key(c\_id) references course(c\_id))

可以发现，stu表，course表和take表都创建成功





**2.2验证如果有外码约束，创建失败**



相应的外键需要提前导入，而如果直接创建take表，则不存在s\_id、c\_id等相应的外码约束

**3.插入数据**

**3.1 正常插入数据**

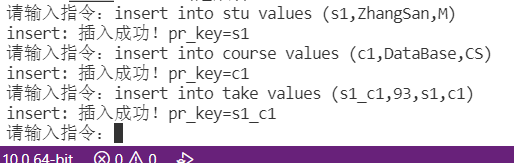
输入：

insert into stu values (s1,ZhangSan,M)

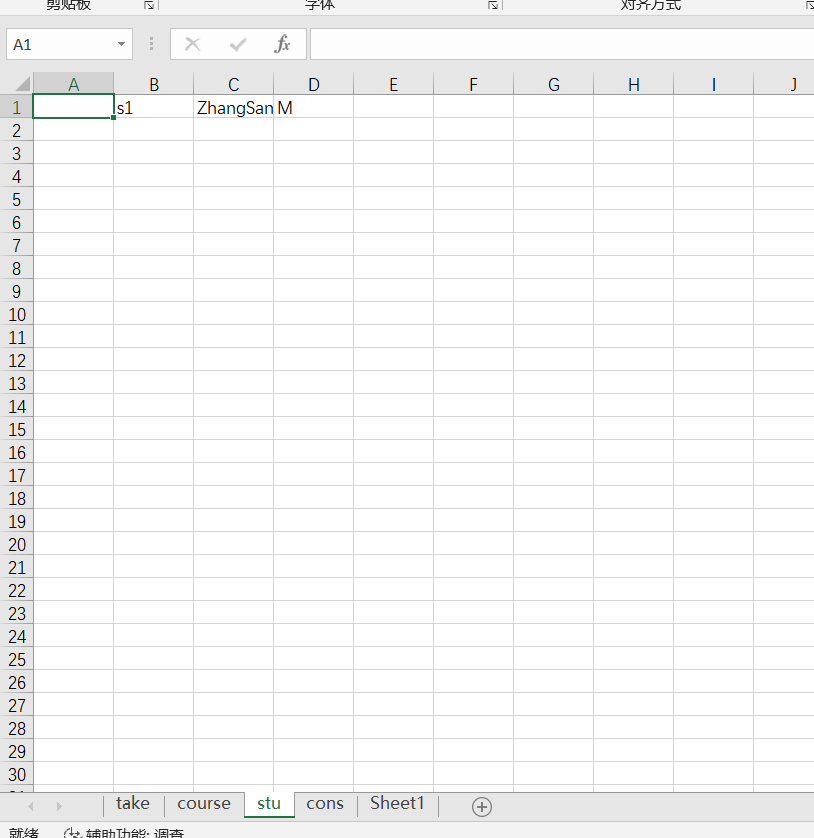
insert into course values (c1,DataBase,CS)

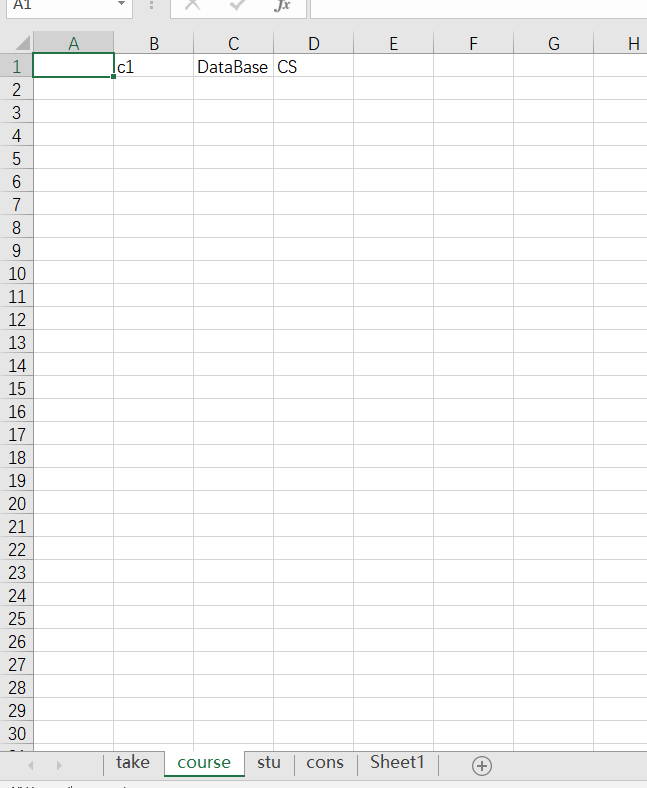
insert into take values (s1\_c1,93,s1,c1)

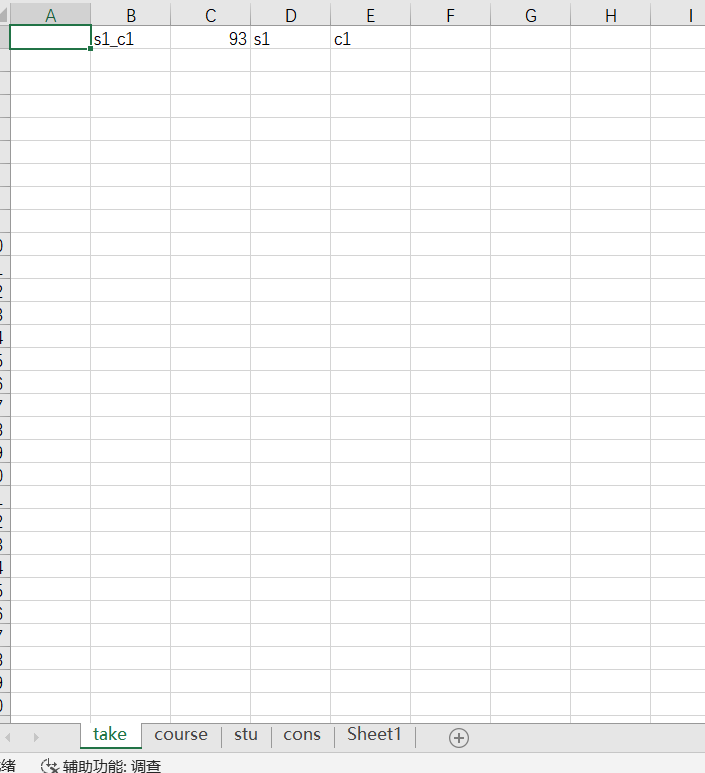
则可以成功将数据插入表中



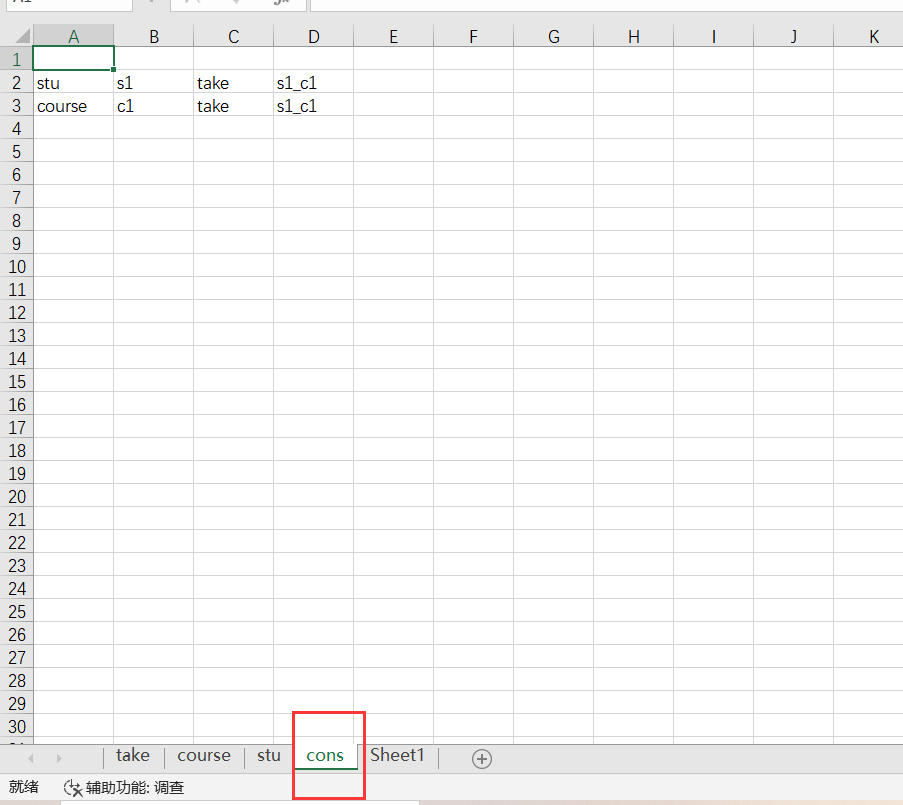
插入效果如下列图所示：





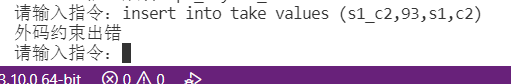


查询cons表，即外键约束表，可以找到所有存在的外键约束：

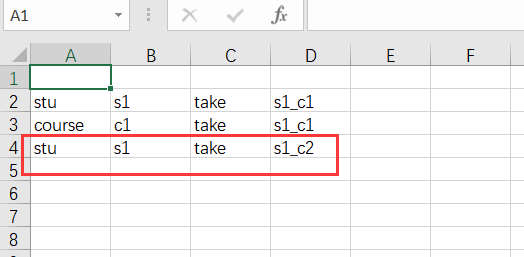


**3.2验证插入数据的的外码约束**

如果输入insert into take values (s1\_c2,93,s1,c2) 指令，但是course表中没有c2的数据，则会出现“外码约束错误”的报错信息。

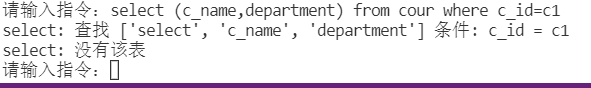


在cons表中，新增了s1\_c2的外键约束

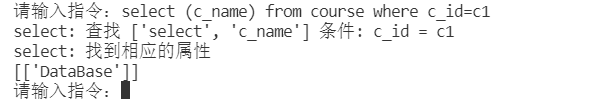


1. **查询数据**

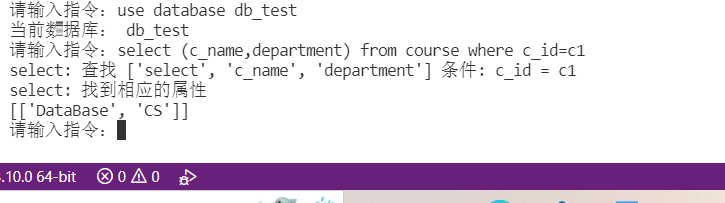
**4.1假如不存在某表，进行查询则会提示报错信息**



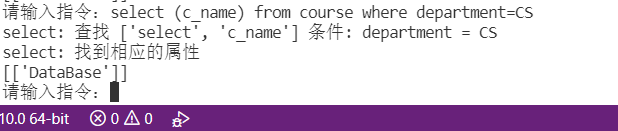
**4.2使用主码索引查一行的一个数据**



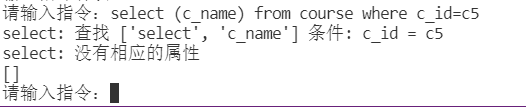
**4.3使用主码索引查一行的所有数据**



**4.4使用非主码索引查询数据**

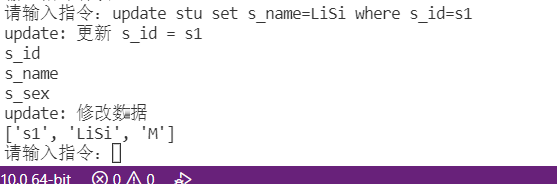


**4.5 查询不存在的数据**

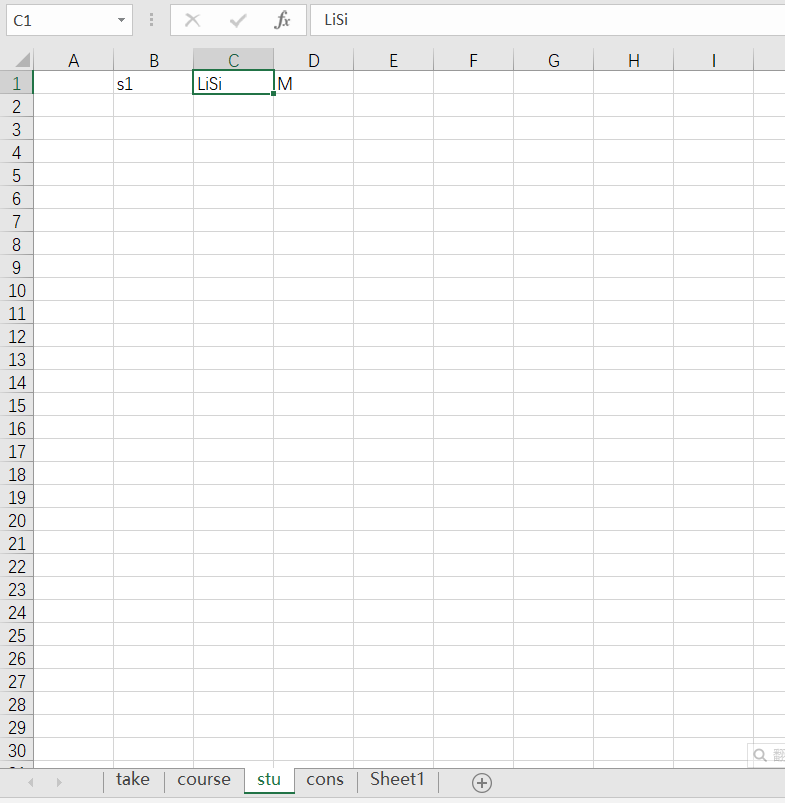


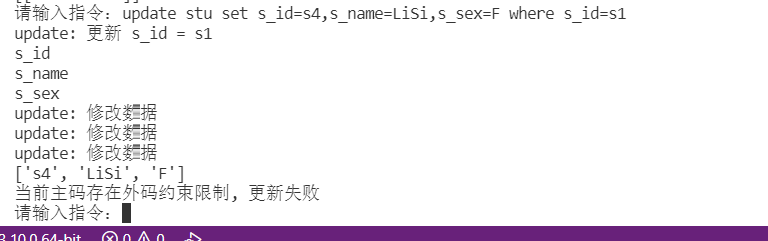
**5.修改数据**

**5.1改一个数据，（非主码，无约束）**

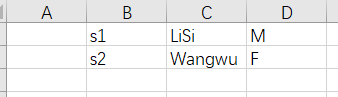


可以看到，原本为“ZhangSan”的s\_name被修改成了“LiSi”



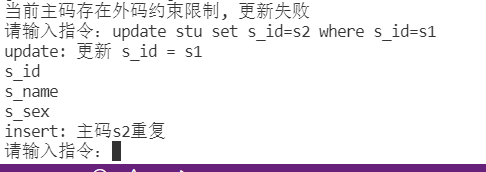
**5.2改一个数据 （非主码，有约束）**

因为在”stu”表中没有s4数据，所以存在相应的约束，无法进行更新操作

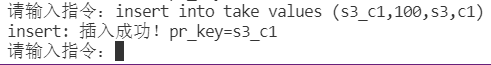


**5.3 改一个数据 （主码，有约束）**

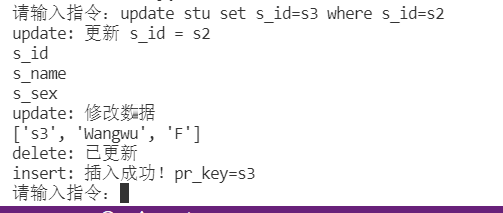
如果表中已有主码s2，将s1的主码修改为s2会有相应提示。

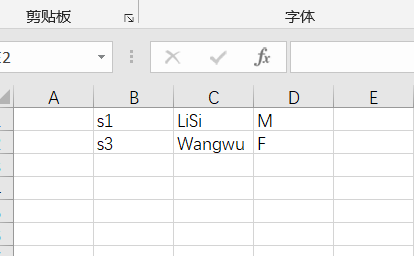


**5.4改一个数据 （主码，无约束）**



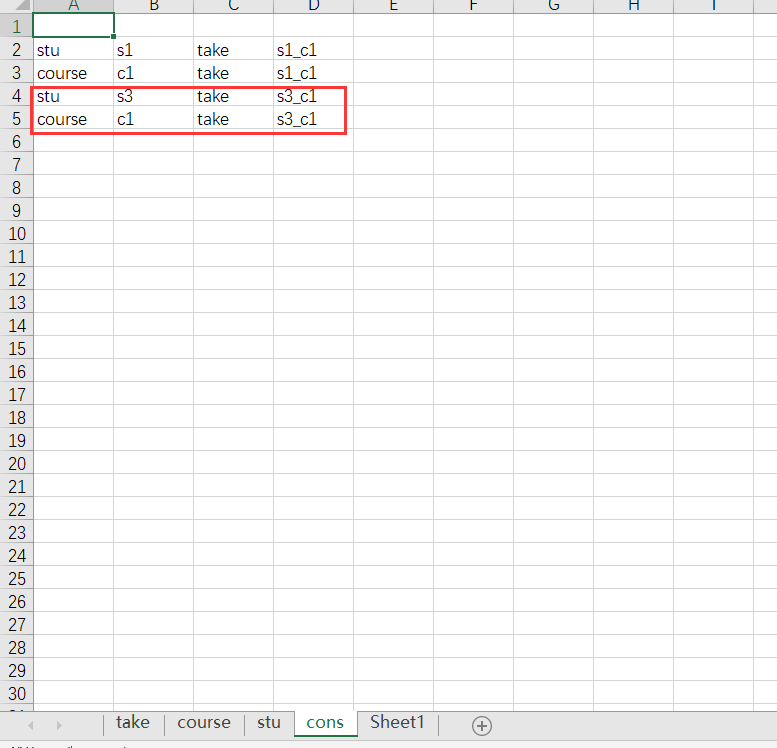
再插入一个take的数据





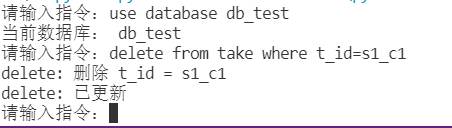
注意到，在修改数据成功的同时，程序进行了删除成功和修改成功的提示。因为主码涉及外键约束的存储，所以调用删除和添加操作可以简单的实现外码约束的更新。

与此同时，cons外键表也有了相应的外键约束的增加。

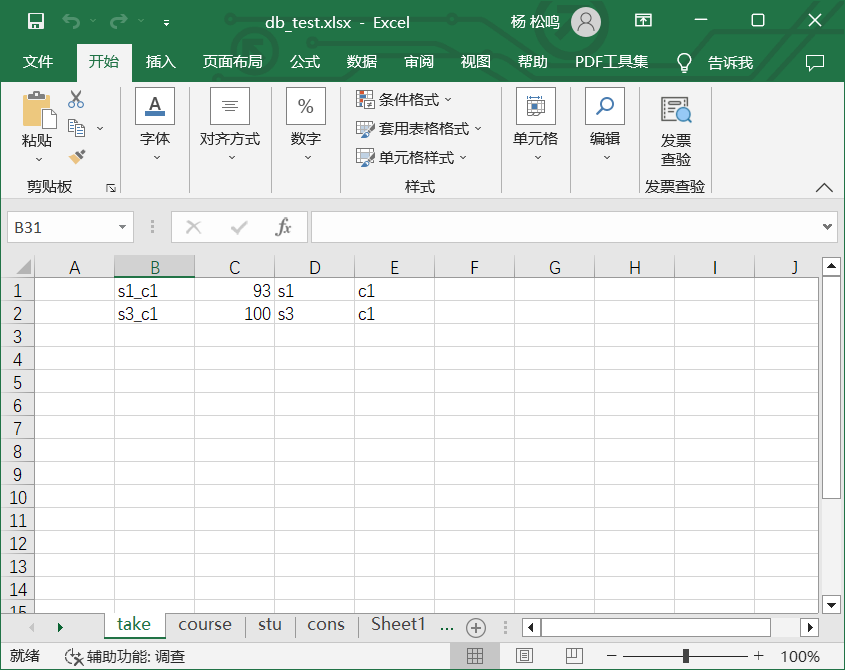


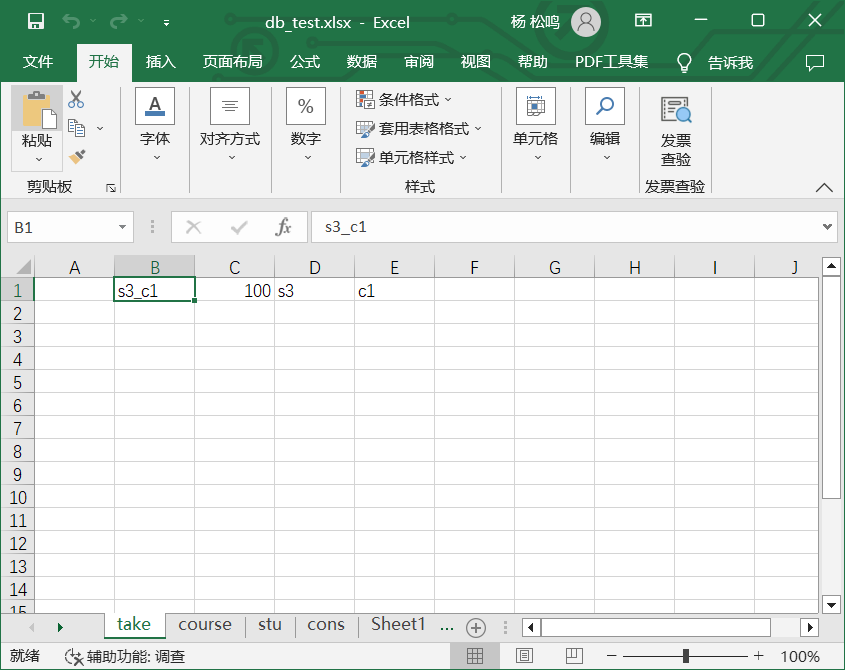
**6.删除数据**

**6.1删除一组数据 （使用其他属性索引，无任何外码约束）**

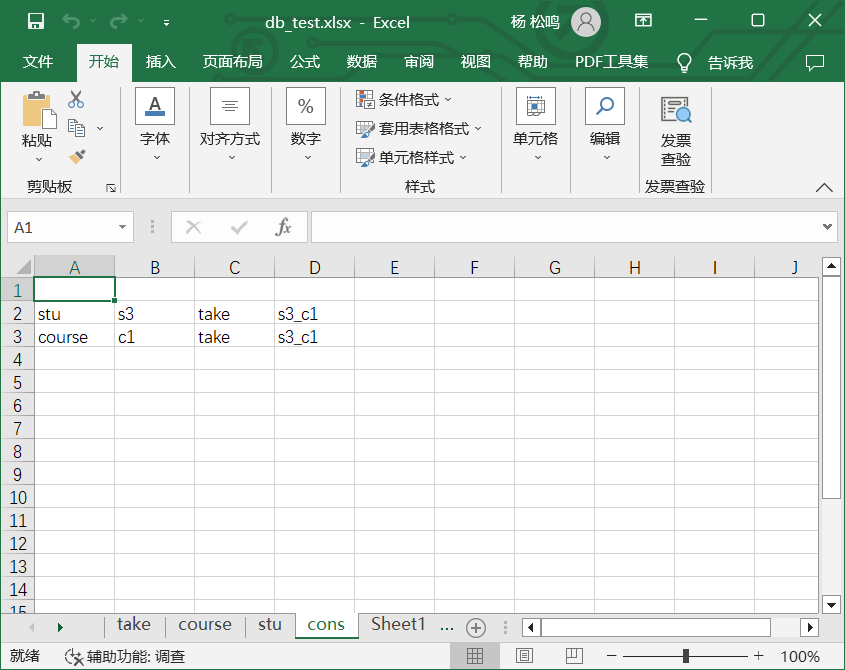


使用非主码进行索引，之后进行删除操作

****

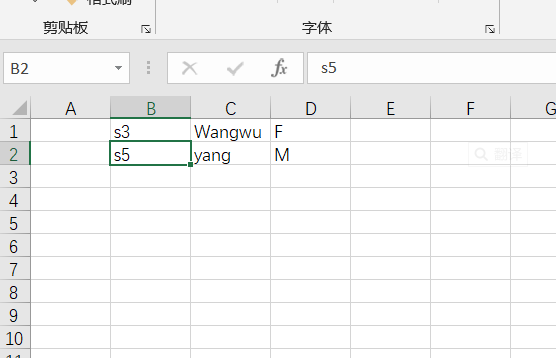


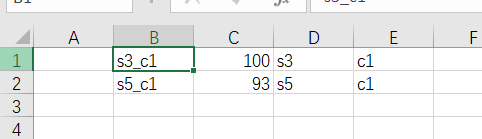
可以同时发现，原本的外键也被同时删除



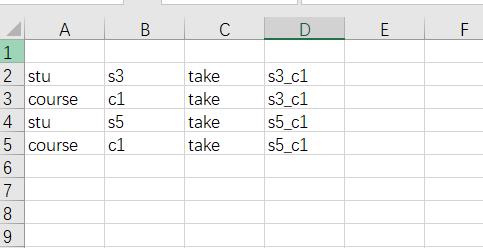
**6.2删除一个数据 （使用主码索引，有外码约束）**

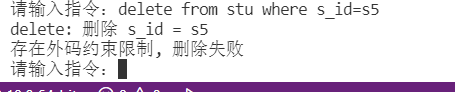
为了演示此模块，额外插入了s\_id=s5的学生，并插入了其上课信息





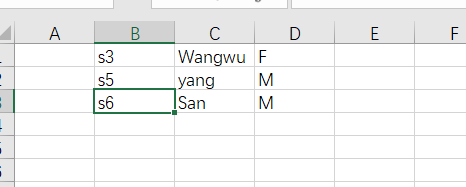
插入后形成的外键约束如图

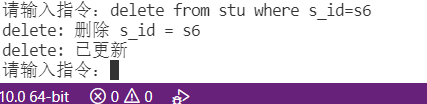


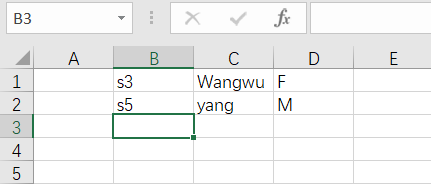


**6.3删除一个数据 （使用主码索引，无任何外码约束）**

对比6.2中的情况，在插入一个没有选课信息的学生s\_id=s6，则其不存在任何的外码约束，之后则可以对其成功删除

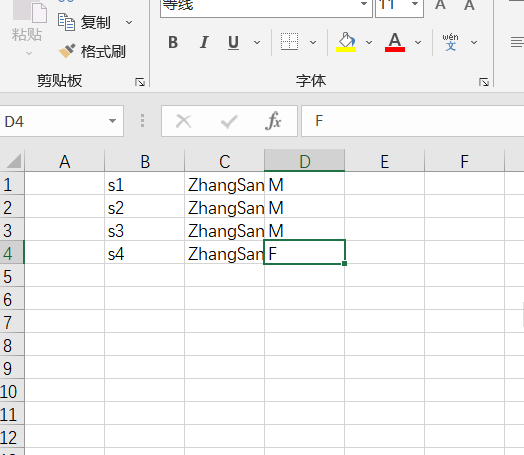


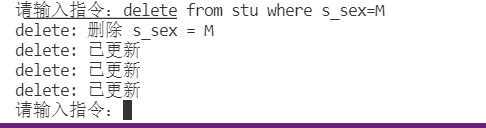




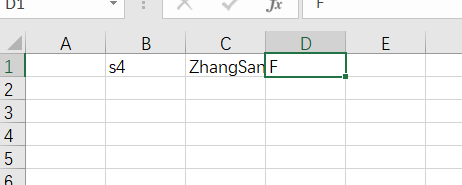
6.4 删除多个数据（无外码约束）

在这里，为了更好的演示，我们导入了以下性别不尽相同的四个stu表数据



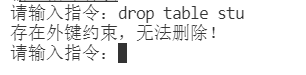


可以发现，所有的男性同学都被删除，输出后的结果如下图：

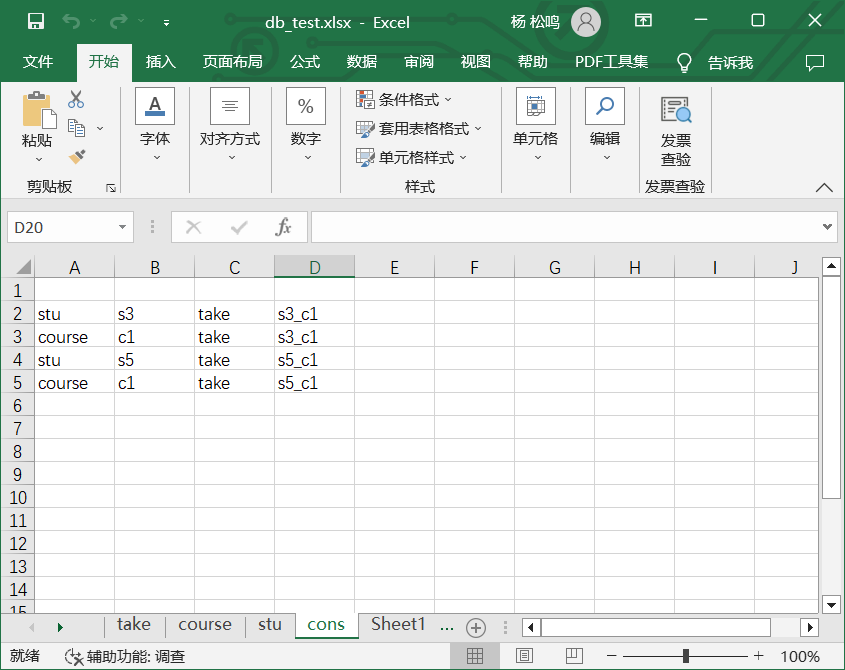


**7.删除表**

**7.1删除一个有外码约束的表（不能删）**

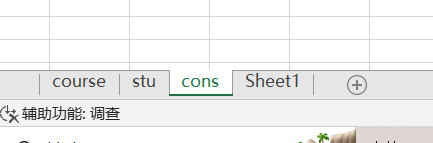


删除stu表时可以发现，因为在cons外键约束中存在对s\_id的约束，所以不能删除此表。

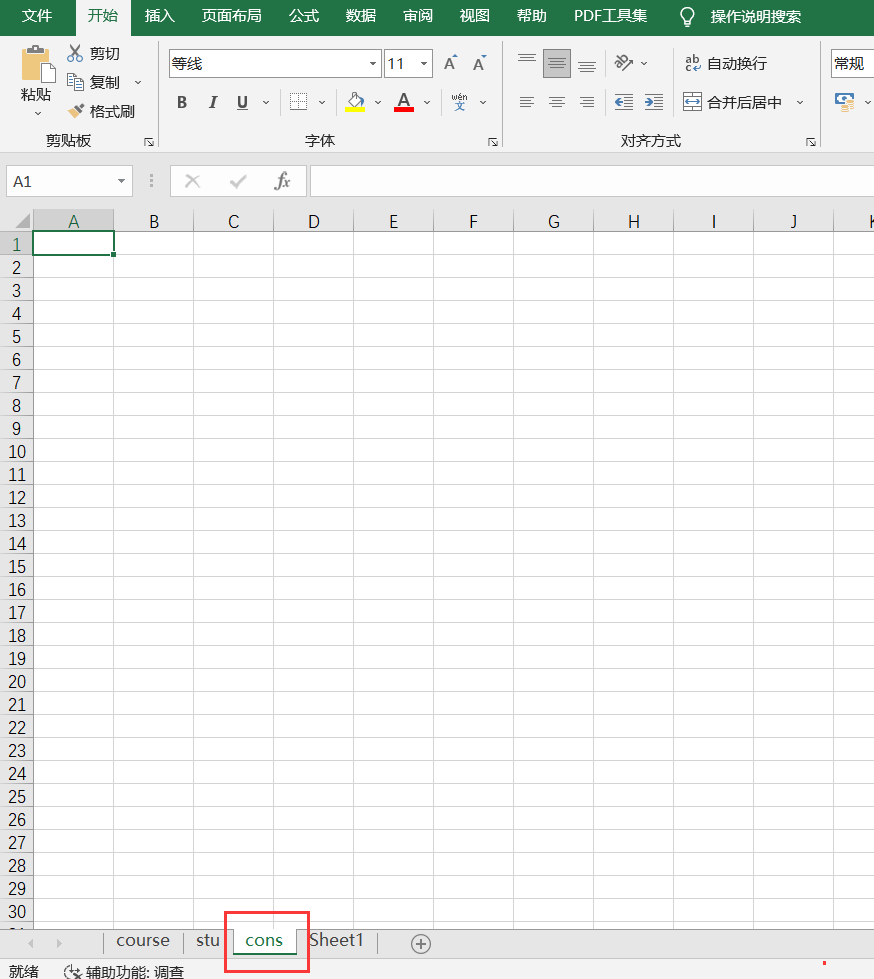


**7.2删除一个无外码约束的表**

可以发现，take表已经被删除



与此同时，观察cons外键约束，发现相应的外键约束也全部被删除。



1. **效率分析**

**5.1正则化的时间复杂度**：  
正则化的具体逻辑为，对输入序列的每个字母和正则化模式字符串进行逐个比较，所以时间复杂度为O（n）,其中的n为输入指令字符串的长度。  
  
**5.2创建数据库：**  
因为创建数据库的过程需要创建一个新的excel文件，（由于具体python内部调用excel的方式未知，excel创建所需的时间复杂度也未知），所以假设创建excel文件的时间复杂度为O(1)，那么整个创建新的数据库的时间复杂度为O(1)  
  
**5.3删除数据库：**  
同创建数据库，时间复杂度也为O(1)。  
创建表：创建表的过程中相当于在已有的excel文件中加入一个子表，假设这个过程的时间复杂度为O(1),那么创建表的时候要检查是否有外码约束存在，这个外码检测是对数据字典表进行遍历，找是否存在与当前表存在外码约束的表是否已经存在，所以这个时间复杂度为O(n)，所以最终的时间复杂度为O(n)。  
  
**5.4创建表和删除表：**  
创建表时间复杂度为线性时间，因为创建表过程中会检测当前表有关的外码约束的表是否存在，所以需要线性O（n）时间。同理，删除表也是如此，删除时要检测是否存在与当前表有外码约束的表，时间复杂度为O(n)，同时还要去数据字典中将所有的与当前表相关的约束记录全部删除，时间复杂度也为线性时间O(m)。所以综上所述，创建表和删除表的时间复杂度都为线性时间O(n)。  
  
**5.5插入数据：**  
插入数据的过程中首先要找到当前子表有关的外码约束的表，判断当前子表相关的外码约束的表是否存在，所以这个遍历数据字典判断是否存在外码相关表是否存在的时间复杂度就是O(n)，n为当前已经存在的表的个数。判断好后，将直接在对应的表后增加一行数据即可，同时在数据字典中的约束表上加入对应的约束记录即可，这个过程时间复杂度为O(1)。综上，插入数据时间复杂度为O(n)。  
 **5.6删除数据：**  
这里只讨论删除一个元祖的情况，删除数据的过程中首先需要判断当前删除的数据是否在目标的表中，这个时间复杂度为O(n)，其中n为当前目标表的元祖的个数，其次，还需要判断是否存在外码约束，所以需要遍历数据字典中，这个时间复杂度为O(m)，其中m是数据字典的表的个数，然后如果存在约束，则要判断数据字典中的约束表是否存在关于当前数据的约束，时间复杂度为O(p)，其中p为数据字典中的约束表中的约束记录个数。所以综合来看时间复杂度为O(n)，线性时间复杂度。  
**5.7更新数据：**  
这里分成两个部分进行讨论，（因为我们实现的update分成了两个模块，第一个顶层update模块，找出所有需要更新的元祖的主码，然后调用底层模块的update去一一更新元祖的数据）顶层的update模块应该是O(n\*m)，其中n是当前目标表的元祖个数，m是当前需要修改的元祖个数，所以也可以认为是O(n^2),因为修改的个数最多就是表中存的那n个元祖。  
然后分成两个情况进行进一步讨论：  
情况一：主码不更改，更改非主属性。首先还是需要判断是否存在外码约束，判断的方法跟之前插入的方法是一致的，时间复杂度为线性时间O(p)，然后进行更改数据即可。  
情况二：主码修改，对于主码需要修改的情况，由于主码涉及到了数据字典中的约束表的修改，因为主码修改后，对应的约束记录的主码也要修改，（当前是被别人引用自己主码有外码约束的表的时候，不允许修改数据。但是当前如果是引用别人主码的时候的外码约束，允许当前修改，并修改约束表中的约束记录）此时我们直接使用delete底层模块加上insert模块实现该功能，由于插入和删除模块的时间复杂度都是线性时间O(n)，所以此时时间复杂度是线性时间O（n）  
综上所述，update模块的时间复杂度为O（n^2）+O(n)，即为O(n^2)。  
  
**5.8查询数据：**

查询即定位到当前子表，然后遍历子表所有数据即可，时间复杂度为O(n)。

1. **小组分工**

杨松鸣：进行需求分析，实验报告的撰写、功能实现的验证和异常修改

易千喜：进行需求分析，外码约束实现和基础功能的补充

韩昊辰：进行需求分析，基础功能（创建、增、删、查、改）的实现

谭俊枫：进行需求分析，正则化的实现和顶层模块的实现

1. **附录（sql语句验证）**

**注意：由于是正则化检验，指令的输入不能有任何多余的空格等不合规操作**

create database if not exists db\_test

use database db\_test

create table stu (s\_id string,s\_name string,s\_sex string,primary key(s\_id))

create table course (c\_id string,c\_name string,department string,primary key(c\_id))

create table take (t\_id string,grade int,s\_id string,c\_id string,primary key(t\_id),foreign key(s\_id) references stu(s\_id),foreign key(c\_id) references course(c\_id))

insert into stu values (s1,ZhangSan,M)

insert into course values (c1,DataBase,CS)

insert into take values (s1\_c1,93,s1,c1)

delete from take where t\_id=s1\_c1

delete from stu where s\_id=s1

delete from course where c\_id=c1

update stu set s\_id=s4,s\_name=LiSi,s\_sex=F where s\_id=s1

update stu set s\_name=LiSi where s\_id=s1

update stu set s\_name=LiSi,s\_sex=dog where s\_sex=F

select (c\_name,department) from course where c\_id=c1

select (c\_name) from course where c\_id=c1

select (c\_name) from course where department=CS

drop table take

drop table stu

drop table course

drop database db\_test