数据库工程作业

要求：

1. 完成一个小型的数据库信息管理系统(或部分功能)，并填写工程作业报告；程序和报告请 在规定时间之内上传。

2. 开发模式(B/S 或 C/S)、开发高级语言任选， 后台数据库使用大型数据库管理系统(SQL Server、Oracle、MySQL 等)，不要使用桌面数据库。

3. 报告中所列举的四种操作， 每种操作举一个例子即可。

4. 作业成绩按照报告中的标准评分， 程序只实现报告中涉及的部分即可。

5. 作业完成后， 请将工程作业报告和程序打包提交给助教老师，并联系助教老师进行系统说 明和演示，回答相关问题。

工程作业报告

1. 项目信息 ( 10 分)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 学号 | 2112060 | 姓名 | 孙蕗 | 专业 | 信息安全 |
| 项目名称 | 仓库管理系统 | | | | |
| 必备环境 | Python3 mysql8.0 | | | | |
| 系统主要功能 简介(4 分) | -添加商品信息、仓库信息  -删除库中的商品数为0的商品  -更新、添加人员信息  -入库出库表含有视图的查询  -添加入库出库信息，更新商品信息 | | | | |
| 系统主要页面 截图(6 分) | 截 3-4 个页面即可  0. 页面 | | | | |

2. 系统配置 ( 10 分)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 说明 | | (2 分) 请说明系统配置情况(后台数据库，高级语言)；  (8 分) 请使用连接串连接高级语言和数据库，并分析字符串的各个部分。 | | | |
| 配置 步骤 2 分 | DBMS | 1. MySQL 8.0 | | | |
| 1. 端口3306，用户名：root | | | |
| 1. 数据库warehouse\_1,字符集utf8mb4 | | | |
| 高级  语言 | 1.python | | | |
| 1. 使用 Flask 及其拓展 flask\_wtf、flask\_bootstrap、flask\_nav 编写页面、表单 | | | |
| 1. 使用库mysql.connector连接数据库 | | | |
| 连接串  分析  (6 分) | | 序 号 | 名称 | 功能说明 | 取值 |
| 1 | class addproduct(FlaskForm):  Product\_id = StringField('product\_id: ', validators=[DataRequired()])  Product\_name = StringField('product\_name: ', validators=[DataRequired()])  Product\_type\_id = StringField('product\_type\_id: ', validators=[DataRequired()])  Product\_sum = StringField('product\_sum: ', validators=[DataRequired()])  submit = SubmitField('submit', validators=[DataRequired()]) | 在class.py里定义表addproduct，继承Flaskform.  为表单添加属性Product\_id、Product\_name、Product\_type\_id、Product\_sum属性，都不能为空，填写时分别提示product\_id:、product\_name:、product\_type\_id:、product\_sum: |  |
| 2 | @app.route('/Add\_Product/', methods=['Get', 'Post'])  def Add\_Product():  form = addproduct()  if form.validate\_on\_submit():  Product\_id = form.data['Product\_id']  Product\_name = form.data['Product\_name']  Product\_type\_id = form.data['Product\_type\_id']  Product\_sum = form.data['Product\_sum']  temp = add\_product(Product\_id, Product\_name, Product\_type\_id, Product\_sum)  return temp  return render\_template('form.html', form=form) | 在app.py里定义一个路由，在127.0.0.1/Add\_Product下可以访问Add\_Product函数。  form是表单addproduct的一个实例，将表单中Product\_id、Product\_name、Product\_type\_id、Product\_sum的值分别赋值给变量 | 表单的值来自填写的值和输入的值 |
| 3 | def add\_product(Product\_id, Product\_name, Product\_type\_id, Product\_sum):  conn = mysql.connector.connect(  host="localhost",  port=3306,  user="root",  password="mysqlsystem",  database="warehouse\_1",  charset="utf8mb4"  )  cursor = conn.cursor()  sql = "insert into products(product\_id,product\_name,product\_type\_id,product\_sum) VALUES (%s,%s,%s,%s);"  str\_ = ''  try:  # 执行SQL语句  cursor.execute(sql, [Product\_id, Product\_name, Product\_type\_id, Product\_sum])  # 提交事务  conn.commit()  str\_ = 'Success'  except Exception as e:  # 有异常，回滚事务  print('traceback.format\_exc():\n%s' % traceback.format\_exc())  conn.rollback()  str\_ = str(e)  cursor.close()  conn.close()  return str\_ | 在sql\_func.py文件里。mysql.connector.connect用来连接mysql。  数据库host=localhost，port端口=3306，user名字=root，password自定义密码，自设password=mysqlsystem，database数据库名字=warehouse\_1，字符集charset=utf8mb4。  定义sql语句，其中%s为待填参数。  cursor.execute(sql, [Product\_id, Product\_name, Product\_type\_id, Product\_sum])用来执行SQL语句  conn.commit()提交事务  如果能够正常执行，返回Success，如果不能够正常执行，报错，则返回响应报错信息  关闭数据库连接 |  |
| 连接串代码 (截屏)  (2 分) | | class.py    app.py    sql\_func.py | | | |
| 备注 | | 添加商品信息 | | | |

3. 数据库设计 ( 14 分)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 说明 | (10 分) 按照数据表的创建顺序，依次给出所涉及数据表的信息， 其中参照字段以“(字 段 1，字段 2，……，字段 n)”的形式给出， 被参照字段以“表名(字段 1，字段 2，……， 字段 n)”的形式给出；  (4 分)一般 DBMS 都可以为数据库生成关系图，请将该图片截屏并粘贴到表格中。 | | | | |
| 数据表 (10) | 创建顺序 | 数据表名称 | 主键 | 参照属性 | 被参照表及属性 |
| 1 | personnel | personnel\_id |  |  |
| 2 | manager | personnel\_id | personnel\_id | personnel(personnel\_id) |
| 3 | customer | personnel\_id | personnel\_id | personnel(personnel\_id) |
| 4 | supplier | personnel\_id | personnel\_id | personnel(personnel\_id) |
| 5 | product\_type | product\_type\_id |  |  |
| 6 | products | product\_id | product\_type\_id | product\_type(product\_type\_id) |
| 7 | warehouse | warehouse\_id |  |  |
| 8 | inbound | inbound\_id | manager\_in\_personnel\_id | manager(personnel\_id) |
| supplier\_personnel\_id | supplier(personnel\_id) |
| warehouse\_id | warehouse(warehouse\_id) |
| product\_id | products(product\_id) |
| 9 | outbound | outbound\_id | manager\_out\_personnel\_id | manager(personnel\_id) |
| customer\_personnel\_id | customer(personnel\_id) |
| warehouse\_id | warehouse(warehouse\_id) |
| product\_id | products(product\_id) |
| 关系图  (4) | 物理模型  Diagram 1 物理模型  概念模型  Diagram 2 概念模型 | | | | |
| 备注 |  | | | | |

4. 含有事务应用的删除操作 ( 13 分)

|  |  |  |
| --- | --- | --- |
| 说明 | (1 分)简要说明该操作所要完成的功能；  (2 分)该操作会涉及的表(必须含有两张或两张以上的关系表，同时以“表名”的 形式给出)  (1 分)表连接涉及字段描述(描述方式为“表 1.属性=表 2.属性”) (1 分)删除条件涉及的字段描述(以“表名.属性=？ ”形式给出)  (4 分)实现该操作的关键代码(高级语言、 SQL)，截图即可；(其中如果删除语句 中不包含任何形式的事务应用将扣除 3 分)  (4 分)如何执行该操作，按所述方法能够正常演示程序则给分。 | |
| 功能描述 (1 分) | 删除商品数为0的商品 | |
| 涉及的表 (2 分) | products,inbound,outbound | |
| 表连接涉 及字段  (1 分) | inbound.product\_id=outbound.product\_id=products.product\_id | |
| 删除条件 字段描述 (1 分) | 字段 | 规则 |
| product\_id | 如果该商品在库里的数量为0，则可以删除，因为数据库选择的删除是级联模式，删除该商品时可以自动删除该商品所有的出入库历史表单和商品表中该商品的信息 |
| 代码 (4 分) | warehouse\_1.sql    class.py    app.py    sql\_func.py | |
| 程序演示 (4 分) | 1. 想要删除501商品，但501商品还有库存   50. 想要删除501商品（但501商品还有库存）   1. 删除失败，返回提示信息   51. 返回提示信息501应先全部出库   1. 在出库表单里将501所有商品全部出库，根据后面触发器控制下的操作，出库表填写信息后，products表里的商品数量product\_sum也会相应更新。   填写出库信息表，将501商品全部出库  52. 根据信息501商品全部出库   1. 成功出库   53. 成功出库   1. 查看表数据   出库前查看products  36. 添加入库信息后的商品表  出库后查看products  57. 501全部出库后的product  出库表填写前后查看inbound  54. 501全部出库前的inbound  出库表填写前outbound  55. 501全部出库前的outbound  出库表填写后的outbound  56. 501全部出库后的outbound  看到相应表里的信息都已经添加或更新   1. 删除501   58. 删除501   1. 成功删除   59. 成功删除501   1. 查看表   501的入库信息已经删除  62. 501的入库信息已被删除  501的商品信息已经删除  63. 商品表里已删除501 | |
| 备注 |  | |

5. 触发器控制下的添加操作 (20 分)

|  |  |  |
| --- | --- | --- |
| 说明 | (1 分)简要说明该操作所要完成的功能；  (2 分)简要说明该触发器所要完成的功能  (1 分)该操作会涉及的表(以“表名”的形式给出)。  (2 分) 该操作输入数据以及输入数据应该满足的条件，如： 数值范围、是否为空； (6 分)实现该操作的关键代码(高级语言、SQL)，截图即可；  (8 分)如何执行该操作，按所述方法能够正常演示程序则给分。 | |
| 功能描述 (1 分) | 当出入库表的信息插入后，更新商品表中的商品数量信息。 | |
| 触发器描述  (2 分) | 如果当前商品在products表中还没有记录、manager的信息在manager表里没有记录、sipplier的信息在supplier表里没有记录、warehouse的信息在warehouse表里没有记录，则不能添加入库表信息，需要先填写相应表的信息。  如果根据输入的仓库和商品信息，该仓库的该商品出库后该仓库存储该商品的数量可能将变成负数，则不能添加出库信息，返回错误信息。如果根据输入的仓库信息和商品信息outbound表里找不到满足要求的数据，则默认该商品在该仓库库存为0，也按出库后数量变成负数不能出库，返回错误信息处理。 | |
| 涉及的表 (1 分) | product,inbound；product,outbound | |
| 输入数据  (2 分) | 字段 | 规则 |
| inbound\_id | 非空 |
| Manager\_in\_personnel\_id | 非空，如果manager表里该信息不存在，则需要先插入manager表里的信息 |
| Supplier\_personnel\_id | 非空,如果supplier表里该信息不存在，则需要先插入supplier表里的信息 |
| Warehouse\_id | 非空,如果warehouse表里该信息不存在，则需要先插入warehouse表里的信息 |
| Product\_id | 非空，如果product表里该信息不存在，则需要先插入product表里的信息 |
| Inbound\_sum | 非空 |
| Inbound\_date | 非空 |
| 输入数据 | Outbound\_id | 非空 |
| Manager\_out\_personnel\_id | 非空 |
| Customer\_personnel\_id | 非空 |
| Warehouse\_id | 非空，如果根据输入的仓库信息和商品信息inbound表里找不到满足要求的数据，则默认该商品在该仓库库存为0，也按出库后数量变成负数不能出库，返回错误信息处理 |
| Product\_id | 非空，如果根据输入的仓库信息和商品信息inbound表里找不到满足要求的数据，则默认该商品在该仓库库存为0，也按出库后数量变成负数不能出库，返回错误信息处理 |
| Outbound\_sum | 非空 |
| Outbound\_date | 非空 |
| 插入操作 源码  (3 分) | 1. 入库表   Warehouse\_1.sql    class.py    app.py    Sql\_func.py     1. 出库表   Warehouse\_1.sql    Class.py    App.py    Sql\_func.py | |
| 触发器源码  (3 分) | 1. 入库表      1. 出库表 | |
| 程序演示 (4 分) | 说明：不违背触发器能够执行插入操作。   1. 添加入库表单 2. Product\_id在products表里已存在   22. 入库表添加信息（product_id已存在）   1. 成功插入   33. 入库表信息已插入   1. 查看表数据   添加前的inbound表  35. 添加入库信息前的入库表  添加后的inbound表  37. 添加入库信息后的入库表  添加前的products表  34. 添加入库信息前的商品表  添加后的products表     1. 添加出库表单 2. 填写出库信息表，将501商品全部出库   52. 根据信息501商品全部出库   1. 成功出库   53. 成功出库   1. 查看表数据   出库前查看products  36. 添加入库信息后的商品表  出库后查看products  57. 501全部出库后的product  出库表填写前outbound  55. 501全部出库前的outbound  出库表填写后的outbound  56. 501全部出库后的outbound | |
| 程序演示 (4 分) | 说明：违背触发器要求，不能够执行插入操作，系统报错。   1. 添加入库表   当前输入的Product\_id在products表里没有信息  38. 入库表添加信息（product_id不存在）  报错提示信息  39. 显示报错提示信息   1. 添加出库表单 2. 当前输入的warehouse\_id，product\_id在inbound表里找不到，默认该仓库该商品库存为0   40. 出库表信息添加（warehouse_id，product_id不同时在inbound表里满足）   1. 返回错误信息   43. warehouse_id，product_id不同时满足要求运行结果   1. 添加出库表单，但outbound\_sum的值大于该仓库已有该商品的库存   41. 出库表信息添加（outbound_sum太多了，不满足要求）   1. 返回错误提示信息   42. outbound_sum不满足要求运行结果 | |
| 备注 |  | |

6. 存储过程控制下的更新操作 ( 18 分)

|  |  |  |
| --- | --- | --- |
| **说明** | (1 分)简要说明该操作所要完成的功能；  (1 分)简要说明该存储过程所要完成的功能；  (2 分)说明该操作涉及操作的表(必须包含两张或两张以上的关系表，以“表名形 式”描述)  (1 分)表连接涉及字段描述(描述方式为“表 1.属性=表 2.属性”)  (2 分)该操作会修改字段(以“表名.字段名”的形式给出)，以及修改规则，如新 数值的计算方法、在何种条件下予以修改等；  (6 分)实现该操作的关键代码(高级语言、SQL)，截图即可；  (5 分)如何执行该操作，按所述方法能够正常演示程序则给分。 | |
| 功能描述（1分） | 人员信息的更新 | |
| 存储过程功能描述（1分） | 如果表里已存在这个人的personnel\_id，则使用update更新，如果不存在，则使用insert插入。  插入每个表的id应该满足该表id取值范围，不满足则返回错误提示信息 | |
| 涉及的关系表（2分） | Personnel,manager,customer,supplier | |
| 表连接涉及字段（1） | Personnel.personnel\_id=customer.personnel\_id  Personnel.personnel\_id=manager.personnel\_id  Personnel.personnel\_id=supplier.personnel\_id | |
| 更改字段 （2分） | 字段 | 规则 |
| Personnel\_name | 当表单存在输入的personnel\_id |
| Personnel\_tel |
| Manager\_password |
| Personnel\_name | 当表单存在输入的personnel\_id |
| Personnel\_tel |
| Customer\_location |
| Customer\_password |
| Personnel\_name | 当表单存在输入的personnel\_id |
| Personnel\_tel |
| Supplier\_location |
| Supplier\_password |
| 更新代码 （3分） | （截屏）  Warehouse\_1.sql        sql\_func.py | |
| 创建存储过程源码（3分） | （截屏）  Warehouse\_1.sql | |
| 存储过程执行源码  （1分） | （截屏）  Class.py    App.py | |
| 程序演示（2分） | 说明：不违背存储过程，能够执行更新操作   1. 添加 2. Manager个人信息添加，满足要求，不存在这个人的id,执行插入操作   1. 个人信息添加   1. 插入成功   2. 插入成功   1. 查看表   3. 在personnel表里的信息  4. 在manager表里的信息   1. 更新 2. supplier信息添加，personnel\_id已存在，进行更新操作   9. 302的信息原表已存在，进行更新   1. 成功操作   13. 成功更改   1. 查看表   更新前supplier    更新前personnel  8. 供货商原相关信息2  更新后supplier    更新后personnel | |
| 程序演示  （2分） | 说明：违背存储过程，系统报错；  添加manager表，但当前id不在manager\_id的取值范围  5. 在manager表里id不符合范围  返回错误提示检查输入  6. 返回提示请检查信息 | |
| 备注 |  | |

7. 含有视图的查询操作 ( 15 分)

|  |  |
| --- | --- |
| 说明 | (1 分)简要说明该操作所要完成的功能；  (1 分)简要说明建立的该视图的功能；  (2 分)简要说明该操作涉及的关系数据表(以“表名”的形式给出) (1 分)简要说明表连接涉及的字段(以“表 1.属性=表 2.属性”) (6 分)实现该操作的关键代码(高级语言、SQL)，截图即可；  (4 分)如何执行该操作，按所述方法能够正常演示程序则给分。 |
| 操作功能 描述(1分) | 输入仓库名和商品名，查询符合条件的入库记录，输出入库表单，管理员姓名，管理员电话，供货商姓名，供货商电话，供货商地址，仓库名字，仓库地址，商品类型名，商品名，入库数量，入库时间。  输入仓库名和商品名，查询符合条件的出库记录，输出出库表单，管理员姓名，管理员电话，顾客姓名，顾客电话，顾客地址，仓库名字，仓库地址，商品类型名，商品名，入库数量，入库时间。 |
| 视图功能 描述(1分) | 实现表的合并，便于查看入库出库信息 |
| 涉及的关 系表(2分) | Inbound,manager,personnel,supplier,warehouse,product\_type,product |
| 表连接字 段(1 分) | Inbound.manager\_id=manager.personnel\_id  Manager.personnel\_id=personnel.personnel\_id  Inbound.supplier\_id=supplier.personnel\_id  Supplier.personnel\_id=personnel.personnel\_id  Inbound.warehouse\_id=warehouse.warehouse\_id  Inbound.product\_type\_id=product\_type.product\_type\_id  Inbound.product\_id=products.product\_id  Products.product\_type\_id=product\_type.product\_type\_id |
| 创建视图 代码(3分) | (截屏)   1. 入库信息warehouse\_1.sql        1. 出库信息warehouse\_1.sql |
| 查询代码 (3 分) | (截屏)  Class.py    app.py    Sql\_func.py |
| 程序演示 (4 分) | 1. 入库信息查询 2. 是存在的数据   64. 查询入库视图   1. 找到信息输出   65. 入库视图成功查询（703）   1. 该信息找不到   66. 输入错误信息   1. 空视图   66.0 入库视图没找到   1. 出库信息查询 2. 是存在的数据   67. 查询出库视图（805）   1. 找到信息输出   68. 出库视图成功查询（805）   1. 不存在的数据   69. 输入错误信息   1. 空视图   70. 出库视图没找到 |
| 备注 |  |