



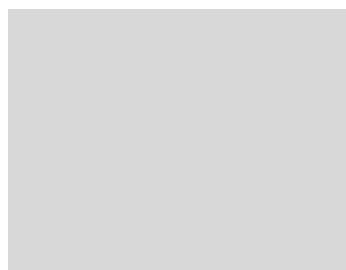
数据库系统课程设计报告

小组名称：园圃草木综理治所

小组成员：石亚男、王雅楠、朱莘菲、杨媛

撰写时间：2023年12月6日

文档版本：V1.0



目录

目录

数据库系统课程设计报告	1
撰写时间：2023年12月6日	1
文档版本：V1.0	1
一、 成员分工 (观测点 9.2)	6
成员 1：石亚男 (210502123)	6
成员 2：王雅楠 (211002725)	6
成员 3：朱莘菲 (211002130)	6
成员 4：杨媛 (211002427)	7
二、 需求分析 (观测点 2.4)	8
(一) 登录业务	8
1. 用例图	9
2. 鲁棒图	10
3. 数据项	10
4. 安全性控制与完整性分析	11
(二) 园林植物基本信息管理	11
1. 用例图	11
2. 鲁棒图	12
3. 数据项	13
4. 安全性控制与完整性分析	14
(三) 园林植物分类管理	14
1. 用例图	14
2. 鲁棒图	15
3. 数据项	15
(四) 园林植物养护管理	14

4. 安全性控制与完整性分析	17
(六) 园林植物监测管理	17
1. 用例图	17
2. 鲁棒图	18
3. 数据项	18
4. 安全性控制与完整性分析	19
三、系统概念结构设计 (观测点 2.4)	20
(一) 园林植物基本信息管理	20
(二) 园林植物分类管理	20
(三) 园林植物养护管理	21
(四) 园林植物病虫害防治管理	21
(五) 园林植物监测管理	22
四、系统的逻辑结构设计 (观测点 3.4)	24
(一) 园林植物基本信息管理	24
(二) 园林植物分类管理	24
(三) 园林植物养护管理	25
(三) 园林植物病虫害防治管理	26
(五) 园林植物监测管理	26
五、系统物理结构设计、持久层设计和关键业务代码 (观测点 3.4)	28
(一) 登录系统	28
(二) 园林植物基本信息管理	28
(三) 园林植物分类管理	44
(四) 园林植物病虫害防治管理	71
(五) 园林植物监测管理	81
六、项目运维管理和优化 (观测点 11.1)	101

(一)管理工具和实施过程	101
(二)数据优化管理	104
植物信息和植物分类交互	105
植物信息和养护管理交互	105
植物信息和防治管理交互	105
具体植物防治视图：适用于具体子货植物防治状态的查询。错误！未定义书签。	
植物信息和监测管理交互	106
(三)数据库高级编程	106
1.园林植物基本信息管理	106
1.1 存储过程	106
1.2 触发器	107
2.园林植物分类管理	108
2.1 存储过程	109
2.2 触发器	109
3.园林植物养护管理	110
3.1 存储过程	110
3.2 触发器	110
4.园林植物病虫害防治管理	111
4.1 存储过程	111
4.2 触发器	111
5.园林植物监测管理	112
5.1 存储过程	112
5.2 触发器	112
(四)数据安全和数据备份	113
1. 备份工作	113

2. 避免 SQL 注入	114
3. 输出信息控制	114
(五)关键问题和解决方法	114
七、附件内容 (观测点 11.1)	115

一、成员分工（观测点 9.2）

成员 1：石亚男（210502123）

职位：组长

任务分工：园林植物病虫害防治管理

- 完成园林植物病虫害防治管理任务的设计与实现，包括概念结构、逻辑结构、物理结构的设计，负责病虫害、药剂、防治三个实体，创建对应表，编写实体类，DAO 层以及对应实现类的代码。
- 负责病虫害管理模块具体 DAO 层和 service 层代码的编写与测试
- 与王雅楠、杨媛同学完成养护、检测、防治业务的需求分析。
- 与杨媛同学完成病虫害防治管理，植物检测管理数据集构建。
- 创建 GitHub 数据仓库，编写 WPS 共享文档，负责整体项目数据整合工作。
- 和朱莘菲同学进行构建云服务器实现多人同时协作 sql server 在线创表，录入数据。
- 负责组内分工协调与任务推进
- 绘制、整理全局 ER 图

成员 2：王雅楠（211002725）

职位：组员

任务分工：园林植物基本信息管理

- 完成园林植物基本信息数据表及数据集的构建、登录业务的数据表及数据集的构建
- 绘制园林植物基本信息部分的需求分析、E-R 图、用例图、鲁棒图和 DDL 操作撰写，登录业务的需求分析
- 负责登录系统和园林植物基本信息管理系统的实体类、DAO 层以及对应实现类的相关代码
- 与朱莘菲同学完成植物基本信息与分类信息协同设计
- 与石亚男同学和杨媛同学一起完成了养护、监测、防治业务的需求分析和数据录入
- 在小组协作方面负责 github 的教学和 java 连接数据库内容的教学
- 小组答辩 PPT 的制作和最终代码文件的整合

成员 3：朱莘菲（211002130）

职位：组员

数据

库系统课程设计报告

任务分工：园林植物分类管理

- 完成园林植物分类管理系统的概念结构、逻辑结构、物理结构的设计，创建实体
- 与王雅楠同学完成植物基本信息与分类信息协同设计，使其可进行连接与交互
- 根据业务需求设计了植物分类信息的相关视图、索引和删除相关的触发器
- 与小组成员协作，寻找数据库的连接方法以进行小组间的协同操作
- 进行数据表与数据集构建，为植物分类信息表建立树状存储结构、在经过选择后用父子节点方法为科属种的分类信息进行相关插入
- 创建了园林植物分类信息部分的 E-R 图、鲁棒图、用例图，根据基本的业务需求进行在数据库中创建各种表结构等的 DDL 操作撰写
- 负责园林植物分类系统在 java 中实体类的定义构建，DAO 层与 service 层实体类的实现代码
- 和石亚男同学进行构建云服务器实现多人同时协作 sql server 在线创表，录入数据。

成员 4：杨媛（211002427）

职位：组员

任务分工：园林植物养护管理、园林植物监测管理

- 和王雅楠同学和石亚男同学协作分析园林植物养护管理、园林植物病虫害防治管理和园林植物监测管理的相关需求
- 和小组成员协作讨论，整理多个任务之间的共性特性，绘制整体用例图
- 和小组成员协作讨论，负责用例图绘制的教学
- 绘制园林植物养护管理、园林植物监测管理的用例图、鲁棒图和 ER 图
- 设计园林植物养护管理、园林植物监测管理相关数据表和数据内容
- 和王雅楠同学针对园林植物养护管理和园林植物监测管理连接园林植物基本信息进行协作
- 设计园林植物养护管理、园林植物监测管理实际工作情景下的索引和视图
- 设计园林植物养护管理、园林植物监测管理实际工作情景下的索引和视图触发器
- 编写园林植物养护管理、园林植物监测管理的实体类定义、DAO 层和 service 层

二、需求分析（观测点 2.4）

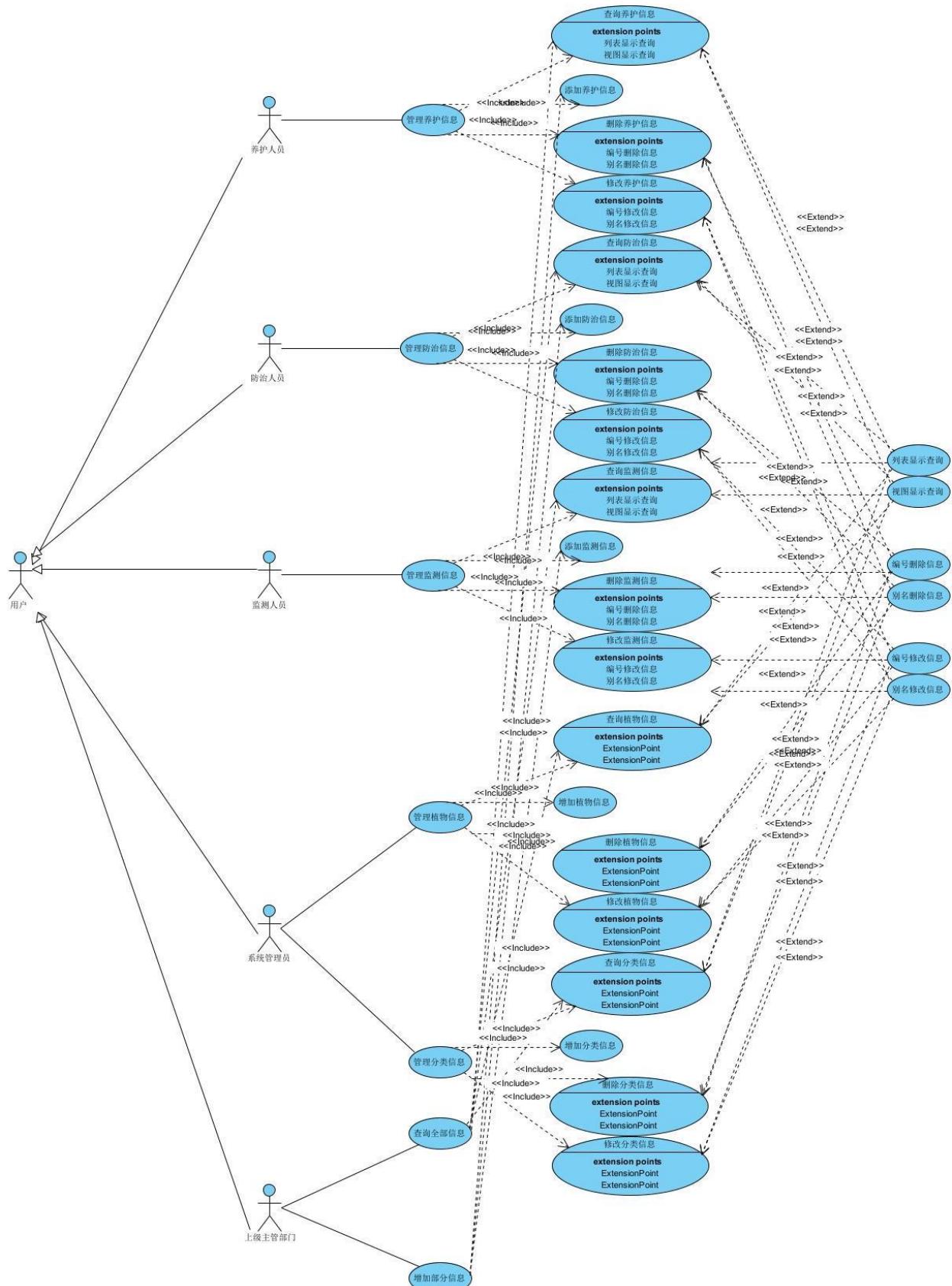
（一）登录业务

登录模块作为整个系统的入口，需要完成用户名、密码的校验，并且由于本系统中用户可能具有多个角色，所以在用户登录时除输入用户、密码外还要选定以哪个角色进行登录。在进行校验时，会详细地校验用户的登录名、密码和登录角色。如果三者对应正确，则将用户角色、用户在该角色表中对应的 id 传递给子系统，涉及数据项如下：

用户表登录校验：用户编号 no（字符型）、用户名 username（字符型）、密码 password（字符型）、角色 role（字符型）

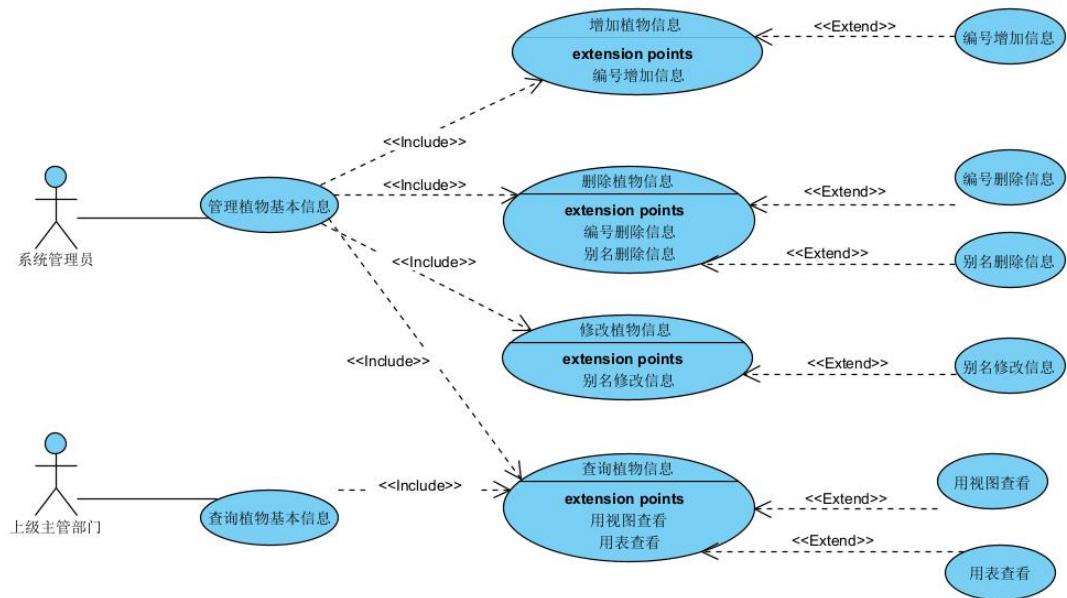
安全性控制与完整性分析：no 为用户表的主键，作为用户的唯一标识，分配账号时控制用户编号不能为空且不能重复，保证实体完整性，用户登录校验表的设计，限制用户的权限，保证用户必须以存在的角色进行登录，有效进行安全控制，防止权限越界。

整体用例图

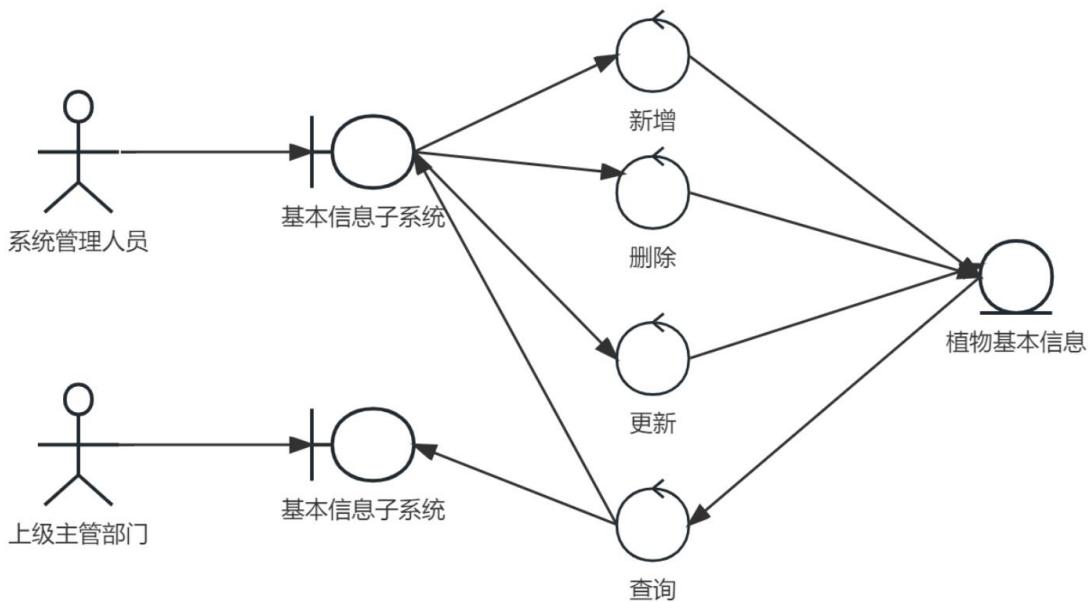


(二) 园林植物基本信息管理

1. 用例图



2. 鲁棒图



基本信息管理模块主要涉及两类角色，分别是系统管理员和上级主管部门，系统管理员可以对植物基本信息进行管理，包括增加植物基本信息（可以已有植物缺少的基本信息进行添加，也可以增加植物所有基本信息），删除植物基本信息（可以删除已有植物的部分信息，也可以删除整个植物条目，但完全删除之前需要先检查该植物是否存在养护、监测和防治中任一任务未完成，存在则不允许删除），增加和查询植物。上级主管部门只可以对植物基本信息进行联合查询，不可以进行修改操作。

3. 数据项

数据

库系统课程设计报告

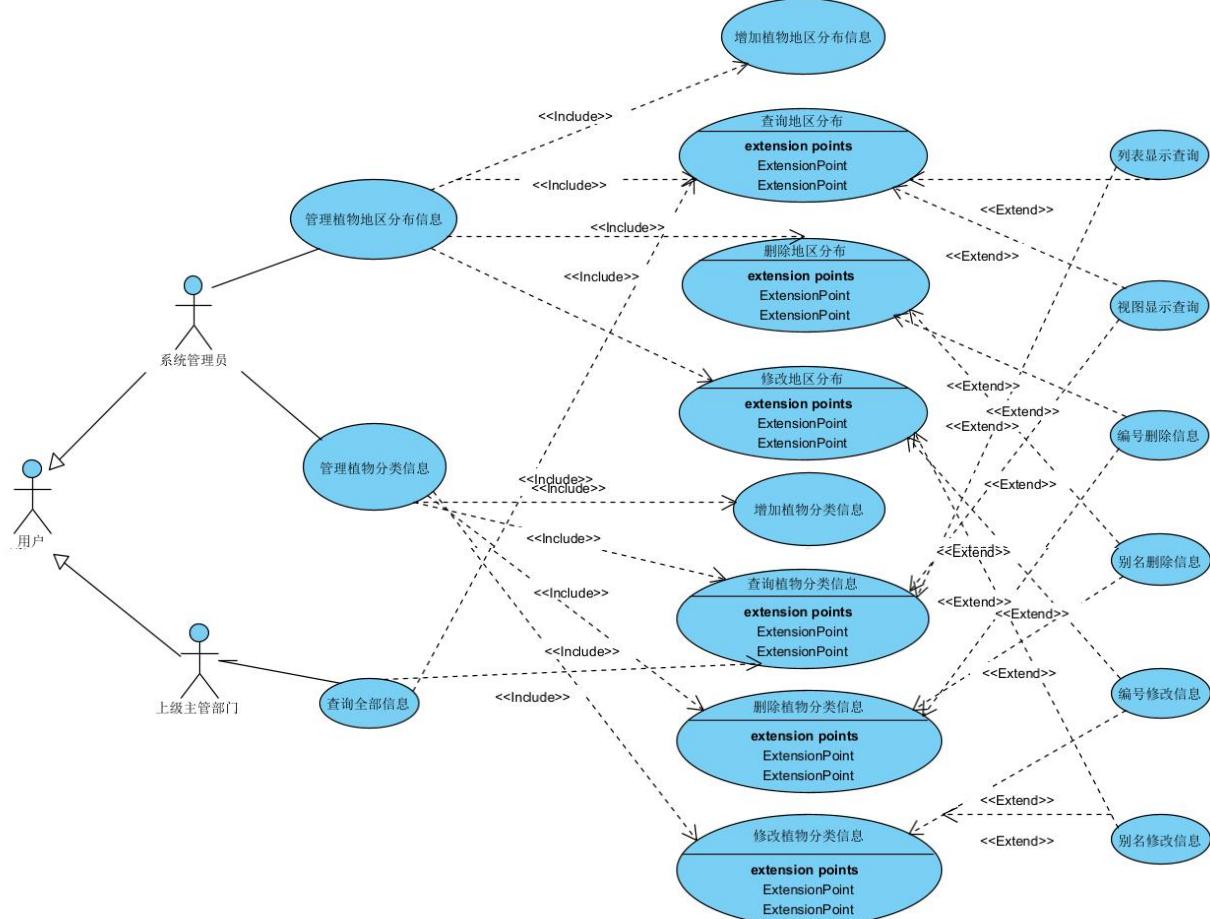
数据名称	数据类型	数据简要描述
植物编号: id	字符型	植物信息表主码
植物名称: taltername	字符型	植物名称
植物科名: familyname	字符型	植物对应科名
植物形态特征: feature	字符型	植物形态特征
植物应用价值: value	字符型	植物应用价值
栽培要点: skill	字符型	植物的栽培要点
分类编号: classifyID	数字型	分类表的主码基本表外码
分布地区编号: disAreaID	字符型	地区表的主码基本表外码
图片编号: photoId	字符型	图表的主码
配图拍摄地点: photoPlace	字符型	图片拍摄地点
配图描述: photoDes	字符型	对配图的描述
配图拍摄人: photoPer	字符型	配图拍摄人
存储路径: photoPath	字符型	配图的路径

4. 安全性控制与完整性分析

在植物基本信息管理部分，引入了植物编号作为主键保证实体完整性，由于植物和配图是一对多的关系，一个植物可以有多张配图。在植物配图表里采取了植物编号和图片编号作为植物配图关系的主码，并且分别设置对应的外码，由于设置了对应的外码，故为了参照完整性，在对植物配图进行增删改时需要同时在图表表和植物配图表里操作。在安全性控制方面，通过触发器进行判断，当要删除植物基本信息时，会先查看当前植物是否存在养护、监测、防治任务，若存在则不允许删除。

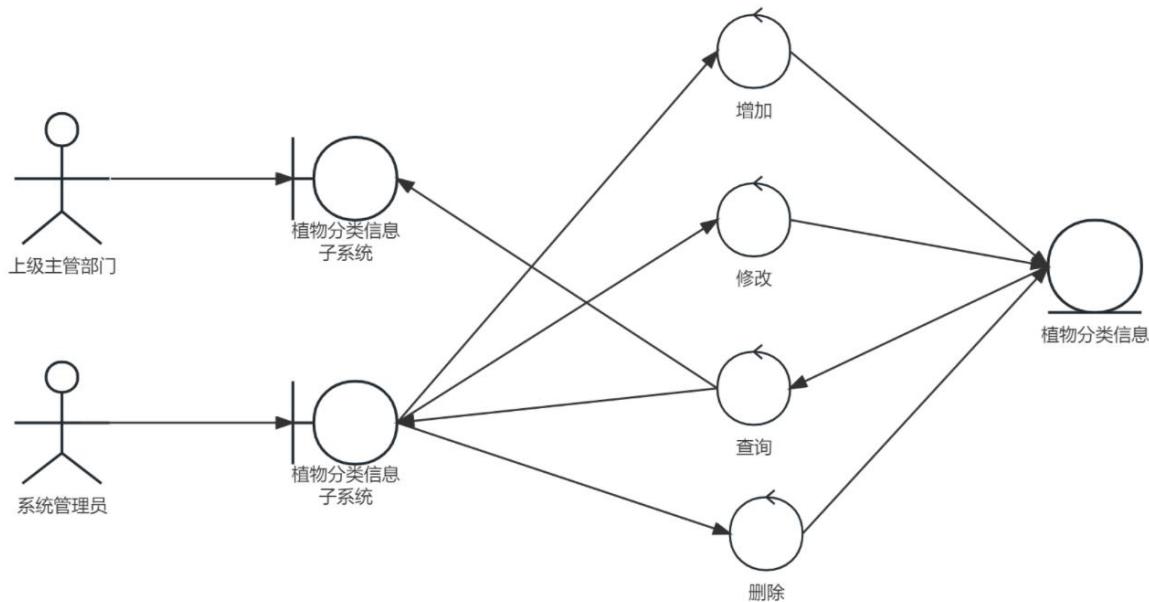
(三) 园林植物分类管理

1. 用例图



该用例图表示出了两类人员对于植物分类系统进行的一系列操作，首先是上级主管部门，他可以在该系统中查询全部的信息，包括植物地区分布信息和植物分类信息；系统管理员可以对该系统的进行管理可以进行增加植物地区分布、修改植物地区分布、删除植物地区分布、查询植物地区分布，可以进行增加植物分类信息、修改植物分类信息、删除植物分类信息、查询植物分类信息，而且在修改的时候，因为有树状分布，所以还要保证对于该数据没有其他项在引用。

2. 鲁棒图



园林植物分类信息可以被上级主管部门与系统管理员访问，对于上级主管部门，只有访问植物分类子系统的功能，可以对指定的信息进行查询，得到其分布信息（即所在的科属种）还有它所在的分布区域与生长环境；对于系统管理员，他可以对植物分类信息管理系统进行增加、修改、删除、查询，并且可以在植物的生长分布区域表中增加它所在的区域信息。

3. 数据项

数据名称	数据类型	数据简要描述
分类编号: classifyID	数字形	植物分类表主码
植物名称: alterName	字符型	对植物信息描述
分类类型: classifyType	字符型	信息科属种类型描述
父节点编号: classifyID_parent	字符型	为实现树状存储 而加入的数据项
地区分类编号: AreaID	字符型	作为外键与分类表连接
地区编号: disAreaID	字符型	唯一的地区标识
分布区域: area	字符型	植物的分布区域表示
省: province	字符型	植物集中分布的省
市: city	字符型	植物集中分布的市
修改信息编号: changeID	数字型	修改信息表主码

数据

库系统课程设计报告

修改人员: createPerson	字符型	修改信息的人员名称
创建日期: innovationTime	日期时间型	创建植物信息的时间
修改日期: updateTime	日期时间型	修改植物信息的时间
生长环境: Envir	字符型	植物的生长环境

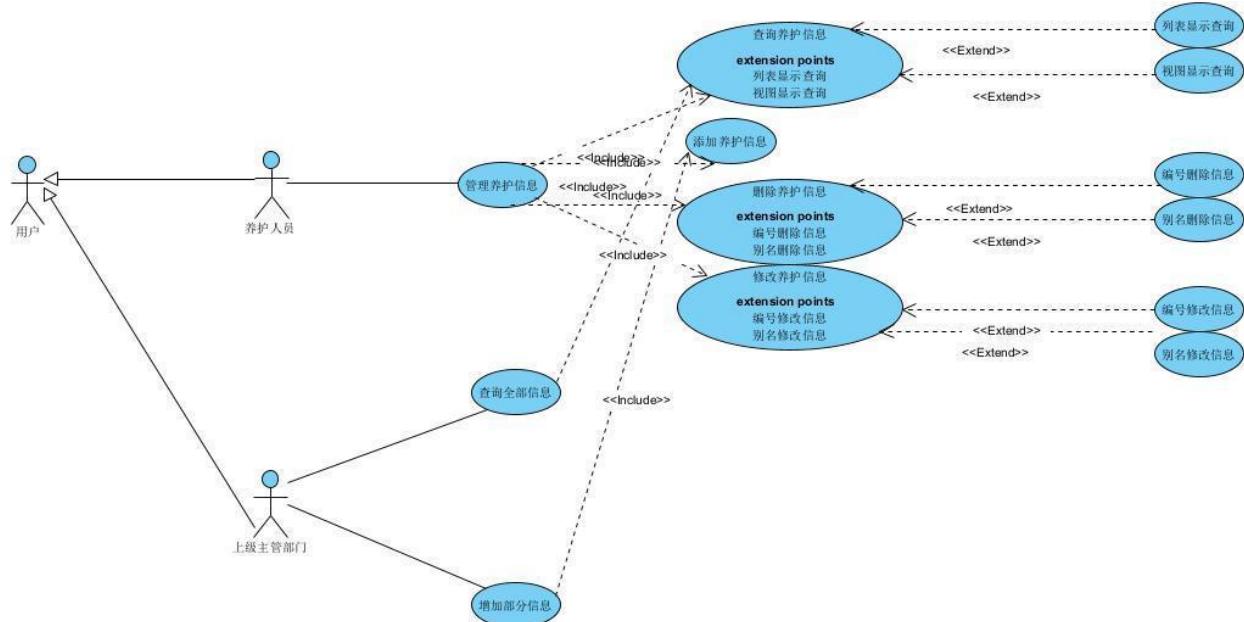
4. 安全性控制与完整性分析

对于安全性分析方面，园林植物分类系统需要进行身份验证与授权，比如其他人员如养护人员等无法进入该系统去查询分类信息，上级主管部门可以查询信息，而系统管理员可以对分类信息进行一系列的操作包括增删改查；在系统管理员进行数据的插入的时候，设计了触发器，如果该数据在树状存储结构中已经被上级引用或者在下属部分还有相关的数据，那么就会报出无法进行操作的错误，这保证了系统之间关系的完整性，确保数据之间的联系不被破坏。

对于植物分类信息的每个实体，即表格都对应着正确的数据类型比如各种 ID 对应数字型数据等，再因为不同的人员可能会对同一种植物进行操作，所以需要保证各表之间拥有数据一致性。

（四）园林植物养护管理

1. 用例图



园林植物养护管理可以被养护人员和上级主管部门进行访问。养护人员可以对于园林植物养护管理任何一个数据表进行增删改查操作。上级主管部门只能对于园林植物养护管理中的养护表进行查询和增加，上级主管部门通过这种方式进行养护任务的发布。

2. 鲁棒图

园林植物养护管理可以被养护人员和上级主管部门进行访问。养护人员通过养护子系统对于园林植物养护管理进行操作，可以对于园林植物养护管理的任何一个数据表进行增删改查操作。上级主管部门通过养护子系统对于园林植物养护管理进行操作，可以对于园林植物养护管理中的养护表进行查询和增加，上级主管部门通过这种方式进行养护任务的发布。其中，养护人员和上级主管部门对于园林植物养护管理的权限和操作不同，需要使用园林植物养护管理中两个不同的养护子系统进行权限和操作分割。

3. 数据项

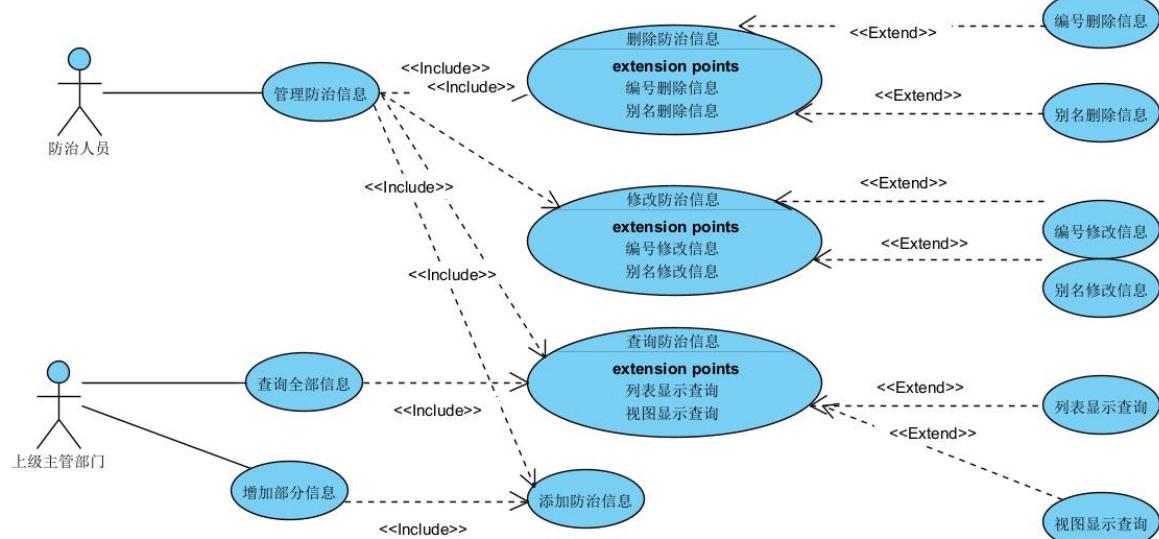
数据名称	数据类型	数据简要描述
任务编号: taskID	整数型	任务表主码
任务名称: taskName	字符型	任务的内容表示
任务描述: taskDes	字符型	任务的细节描述
养护编号: mainID	整数型	养护表主码
养护时间: mainTime	日期时间型	养护的发生时间
养护地点: mainLoc	字符型	养护的发生地点
养护人员: mainPer	字符型	养护的执行人
养护对象: id	字符型	养护表外码基本信息表主码

4. 安全性控制与完整性分析

在园林植物养护管理中，使用任务编号作为任务表主码，使用养护编号作为养护表主码，从而保证实体完整性。任务和养护是一对多的关系，一个养护只能进行一个任务，一个任务能够被多个养护完成。在任务养护表中，将任务编号和养护编号作为联合主码，并且分别设置任务编号和养护编号作为外码。为了保证参照完整性，在对于任务表和养护表进行增加、删除和更改的时候，需要考虑任务养护表中存在的数据。

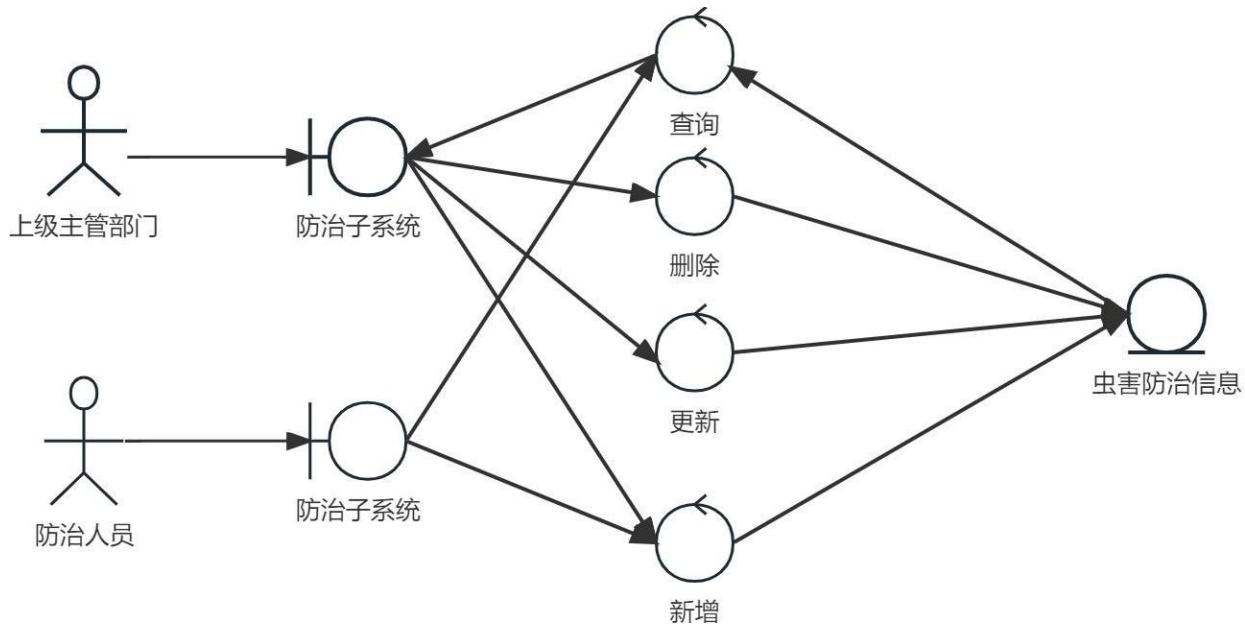
（五）园林植物病虫害防治管理

1. 用例图



该部分主要有防治人员和上级主管部门两个参与者。两种参与者均可访问病虫害防治管理子系统。其中防治人员可以管理防治信息，实现药剂表、病虫表、防治表和虫害防治表信息的增加、删除、更新和查询，上级主管部门可以查询全部信息例如药剂表信息、病虫信息，增加部分信息例如增加防治任务信息。

2. 鲁棒图



上级主管部门通过进入防治子系统实现查询、删除、更新和新增虫害防治信息功能。防治人员通过进入防治子系统实现虫害防治信息的查询和新增任务功能。

3. 数据项

数据名称	数据类型	数据简要描述
药剂编号: mediID	整数型	药剂表主码
药剂名称: mediName	字符型	药剂的中文名称
药剂用量: mediDos	整数型	一次药剂使用剂量

数据

库系统课程设计报告

作用期限: dataAct	日期型	药剂作用时间
病虫编号: pestID	整数型	虫害表主码
病虫名称: pestName	字符型	病虫中文名称
防治编号: conoID	整数型	防治表主码
防治时间: conTime	时间	防治发生时间
防治地点: conLoc	字符型	防治发生地点
防治人员: conPer	字符型	进行防治的工作人员
防治状态: status	字符型	植物防治状态
防治植物编号: id	整数型	防治表外码基本信息表主码

4. 安全性控制与完整性分析

安全性分析:

实施权限控制策略，根据不同角色的功能需求为其分配相应权限，例如上级主管部门在进行病虫害防治任务管理时，只可进行增加任务（D_Control）的操作，但实际并不能操作药剂表、病虫表和虫害防治表，只可查看，不可修改编辑。除此之外，能进入该数据也有严格角色限制。

同时病虫害防治管理中，触发器的设计实现了审计监控、安全信息生成等功能，比如在管理员修改虫害信息时，如果修改数据成功则生成特定情景语句，保证操作的正确性安全性。

完整性分析:

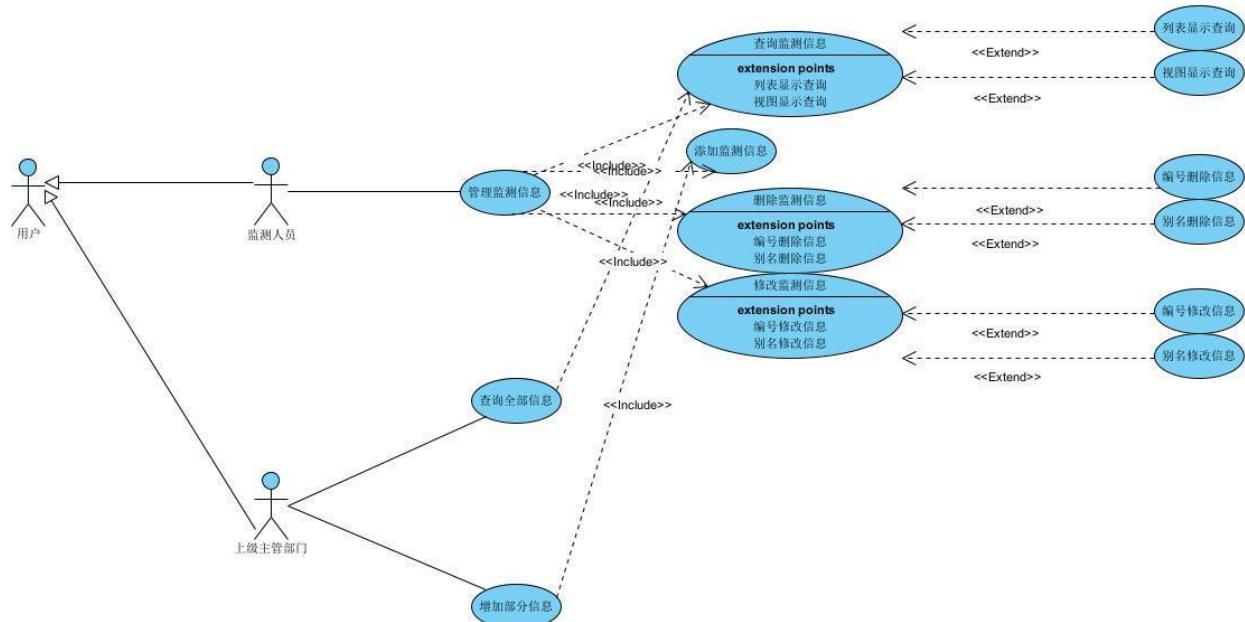
遵循实体完整性约束：病虫害管理中的药剂表、虫害表和防治表均有唯一标识符即主键，mediID、pestID 和 conID，属性值唯一且不为空。

遵循参照完整性约束：病虫害管理中的三个表通过加入表主键属性实现与其他系统中业务线实现联动。例如 D_Control 表中加入基本信息表中属性 id 实现与基本信息任务线互动。同时确保了各表之间关系的一致性。要求插入、更新或删除信息时，必须遵循外键和主键之间的关系。

遵循用户自定义完整性约束：例如防治表 D_Control 中属性 status，创建 DDL 语句时，通过语句 `status VARCHAR(255) CHECK (status IN ('已完成', '进行中', '未完成'))` 限制其取值只可为已完成、进行中或未完成。

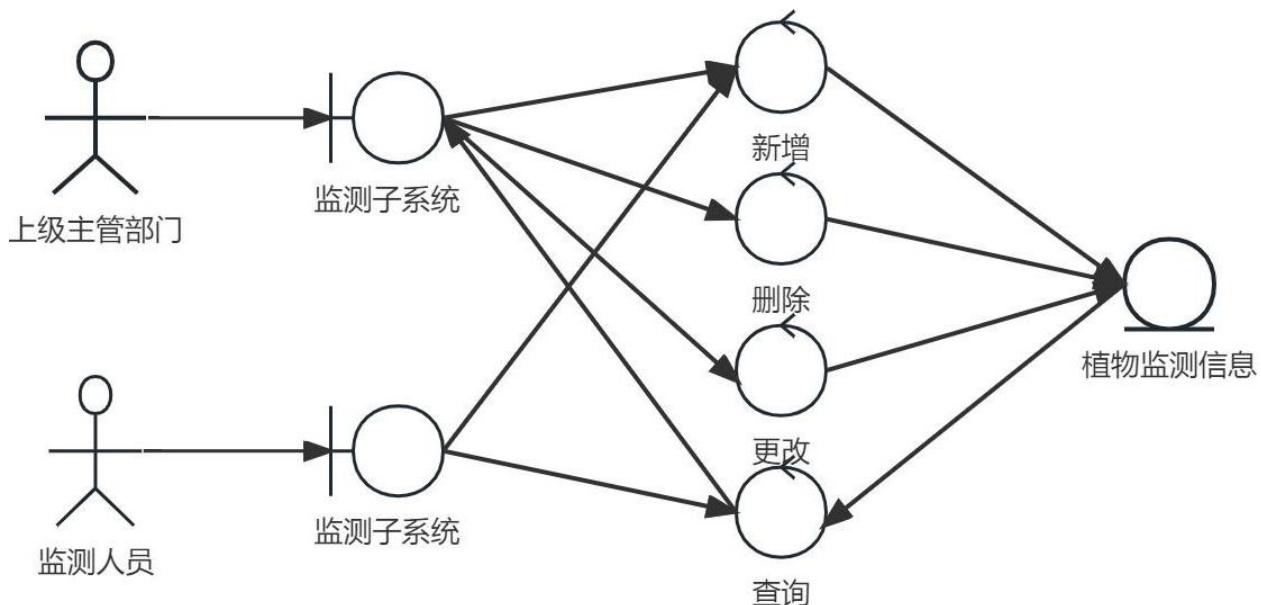
（六）园林植物监测管理

1. 用例图



园林植物监测管理可以被监测人员和上级主管部门进行访问。监测人员可以对于园林植物监测管理任何一个数据表进行增删改查操作。上级主管部门只能对于园林植物监测管理中的监测表进行查询和增加，上级主管部门通过这种方式进行监测任务的发布。

2. 鲁棒图



园林植物监测管理可以被监测人员和上级主管部门进行访问。监测人员通过监测子系统对于园林植物监测管理进行操作，可以对于园林植物监测管理的任何一个数据表进行增删改查操作。上级主管部门通过监测子系统对于园林植物监测管理进行操作，可以对于园林植物监测管理中的监测表进行查询和增加，上级主管部门通过这种方式进行监测任务的发布。其中，监测人员和上级主管部门对于园林植物监测管理的权限和操作不同，需要使用园林植物监测管理中两个不同的监测子系统进行权限和操作分割。

3. 数据项

数据名称	数据类型	数据简要描述
设备编号: equipNum	整数型	设备表主码

数据

库系统课程设计报告

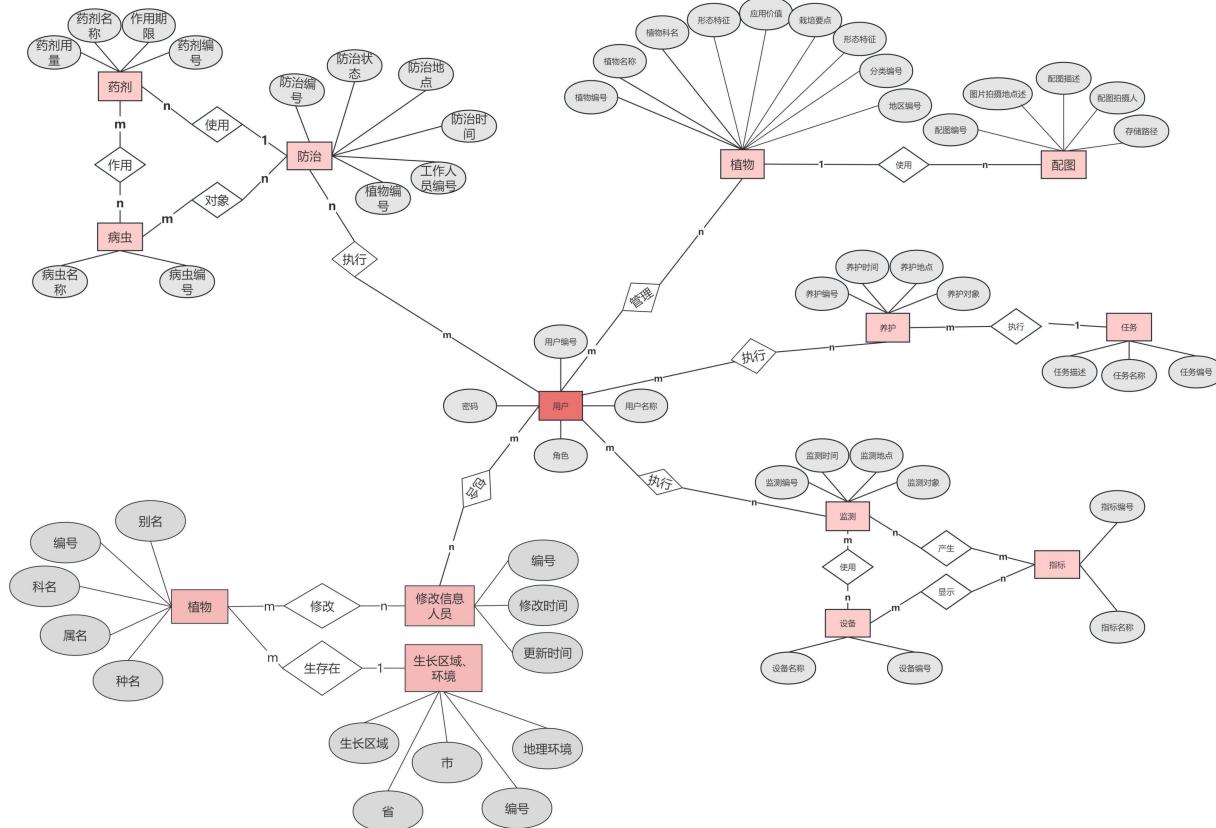
设备名称: equipName	字符型	设备的内容表示
指标编号: indexNum	整数型	指标表主码
指标名称: indexName	字符型	指标的内容表示
监测编号: detecNum	整数型	监测表主码
监测时间: detecTime	日期时间型	监测的发生时间
监测地点: detecLoc	字符型	监测的发生地点
监测人员: detecPer	字符型	监测的执行人
监测对象: id	字符型	监测表外码基本信息表主码

4. 安全性控制与完整性分析

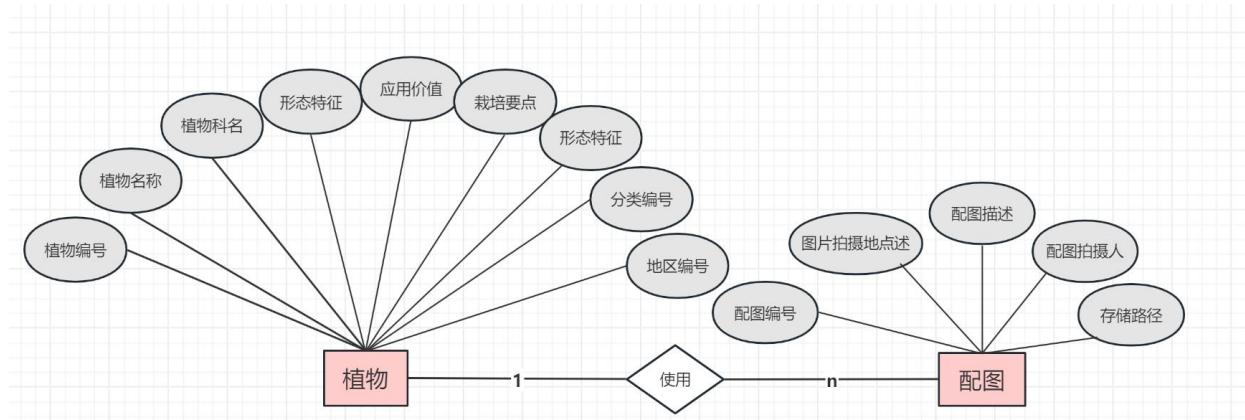
在园林植物监测管理中，使用设备编号作为设备表主码，使用指标编号作为指标表主码，使用监测编号作为监测表主码，从而保证实体完整性。设备和指标是多对多的关系，一个设备能够显示多个指标，一个指标能够来源多个设备；设备和监测是多对多的关系，一个设备能够进行多个监测，一个监测能够使用多个设备；指标和监测是多对多的关系，一个指标能够属于多个监测，一个监测能够产生多个指标。在设备指标监测表中，将设备编号、指标编号和监测标号作为联合主码，并且分别设置设备编号、指标编号和监测标号作为外码。为了保证参照完整性，在对于任务表和养护表进行增加、删除和更改的时候，需要考虑设备指标监测表中存在的数据。

三、系统概念结构设计（观测点 2.4）

全局 ER 图

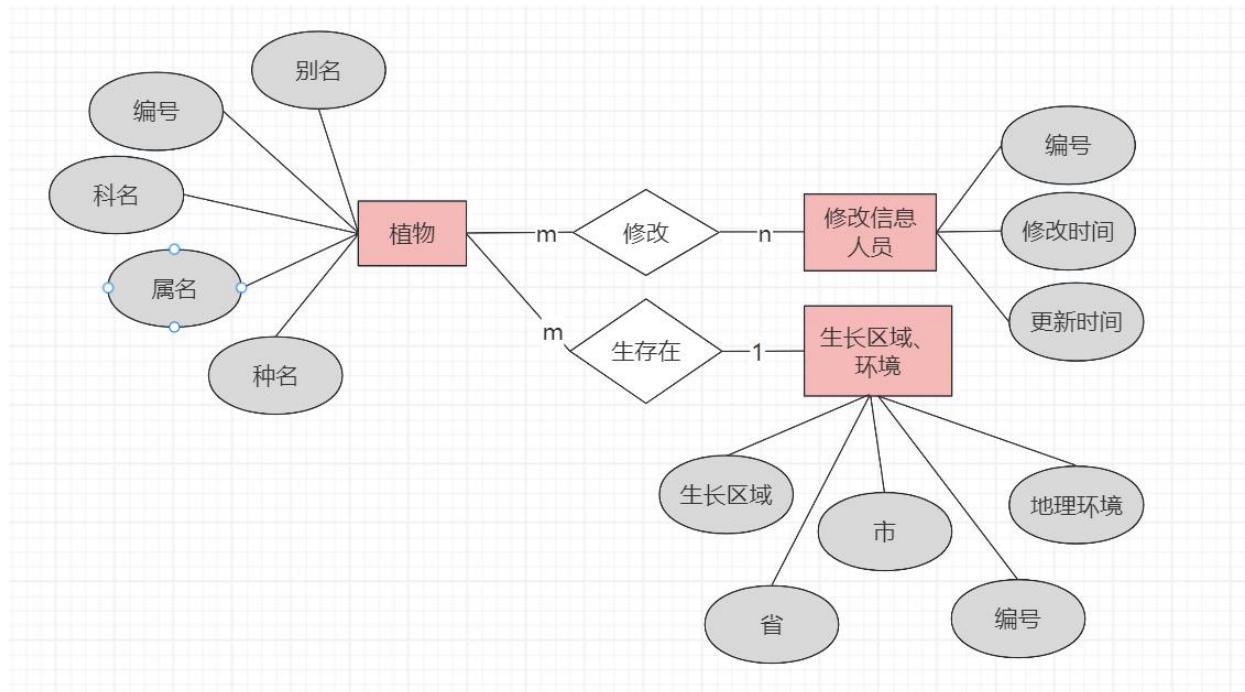


(一) 园林植物基本信息管理



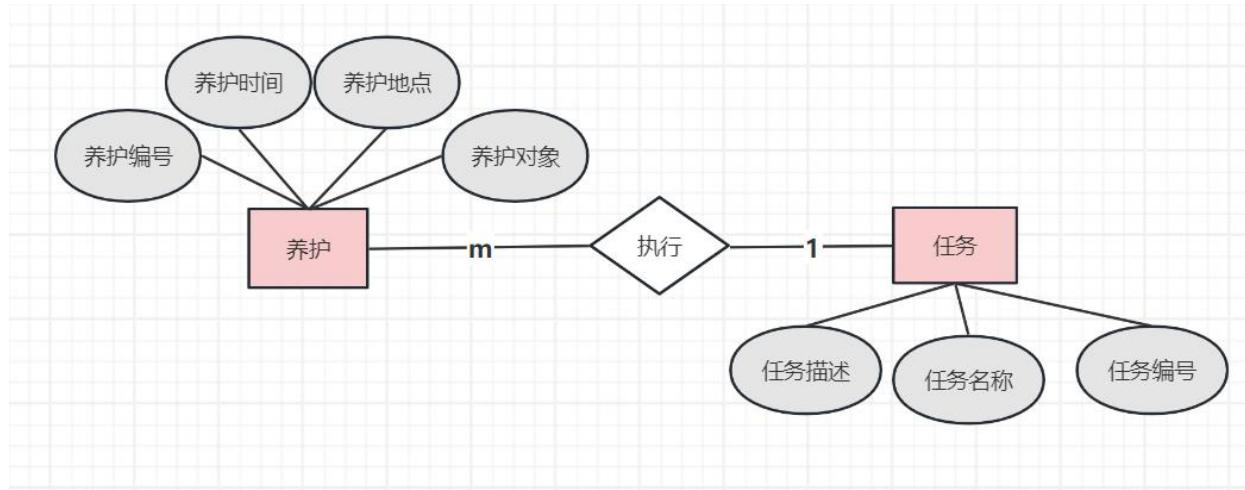
在植物基本信息管理中有两个实体，分别是植物和图片，植物实体里包括植物的编号、名称、形态特征、应用价值、栽培要点等基本信息；图片实体里包括图片编号、配图拍摄地点、配图描述、配图拍摄人、存储路径的信息。这两个实体是一对多的关系。

(二) 园林植物分类管理



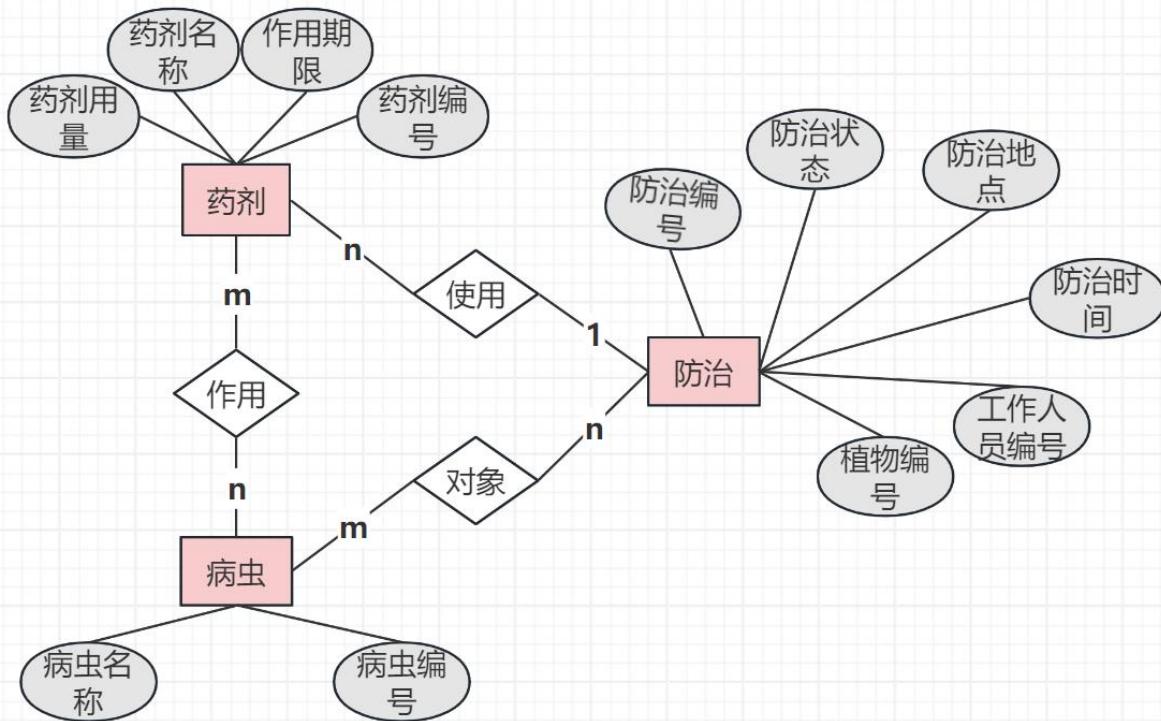
在园林植物分类信息中有三个实体，分别是植物的细分类，植物的生长区域与环境，修改人员。植物的细分类有科、属、种以及植物的编号和别名，植物的生长区域与环境有分布区域、环境气候、省、市以及植物区域编号，修改人员有编号、修改时间、更新时间和人员姓名。这三个实体是多对多的关系。

(三) 园林植物养护管理



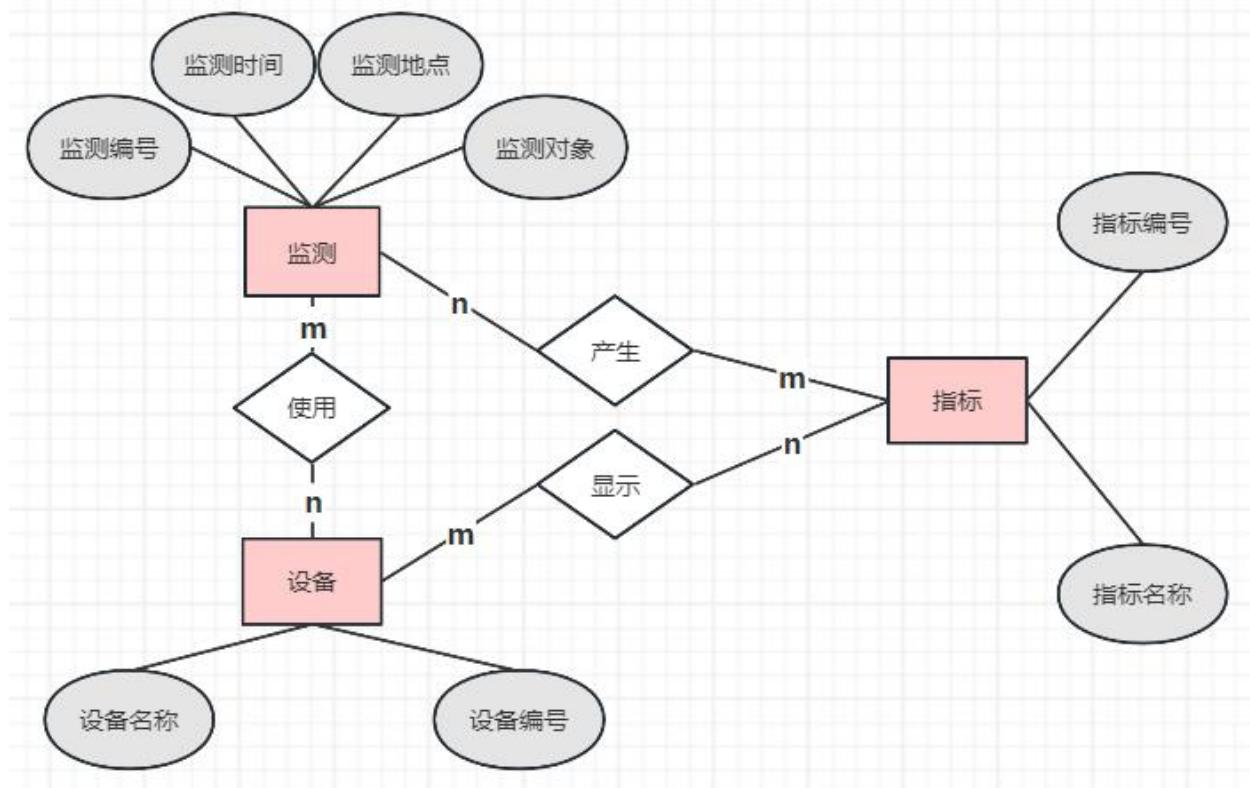
在园林植物养护管理中有两个实体，分别是任务和养护。任务包含养护任务的任务编号、任务名称和任务描述。养护包含植物养护的养护编号、养护时间、养护地点、养护人员和养护对象。任务和养护是一对多的关系，一个养护只能进行一个任务，一个任务能够被多个养护完成。

(四) 园林植物病虫害防治管理



在园林植物病虫害防治管理中有三个实体，分别是药剂、病虫和防治。药剂实体包含属性为药剂用量、药剂名称、作用期限和药剂编号，防治实体包含属性有防治编号、植物编号、防治地点、防治地点、防治状态、防治时间和工作人员编号，病虫实体包含病虫名称、病虫编号属性。三个实体中除防治与药剂是一对多关系外，其他实体间均是两两对应关系。

(五) 园林植物监测管理



在园林植物监测管理中有三个实体，分别是设备、指标和监测。设备包含监测设备的设备编号和设备名称，指标包含监测指标的指标编号和指标名称，监测包含植物监测的监测编号、监测时间、监测地点、监测人员和监测对象。这三个实体中两两实体是多对多关系。设备和指标是多对多的关系，一个设备能够显示多个指标，一个指标能够来源多个设备；设备和监测是多对对多的关系，一个设备能够进行多个监测，一个监测能够使用多个设备；指标和监测是多对对多的关系，一个指标能够属于多个监测，一个监测能够产生多个指标。

四、系统的逻辑结构设计（观测点 3.4）

(一) 园林植物基本信息管理

1.1 植物基本信息表

（植物编号： id， 别名： altername， 科名： familyname， 形态特征： feature， 应用价值：
value， 栽培要点： skill， 分类编号： classifyID ， 分布地区编号： disAreaID ）

主码： 植物编号 id

外码： 分类编号： classifyID ， 分布地区编号： disAreaID

1.2 图片表

（图片编号： photoId， 配图拍摄地点： photoPlace， 配图描述： photoDes， 配图拍摄人：
photoPer 存储路径： photoPath）

主码： 图片编号 photoId

1.3 配图表

（图片编号： photoId， 植物编号： plantId）

主码： 图片编号： photoId， 植物编号： plantId

外码： 图片编号： photoId， 植物编号： plantId

第一范式要求所有属性不可再分。第二范式在第一范式的基础上消除非主属性对主码的部分函数依赖。第三范式在第二范式的基础上消除了非主属性对主码的传递函数依赖。

在园林植物基本信息管理中，每一个属性均为不可再分。每个非主属性不具有对于主码的部分函数依赖。每个非主属性都完全依赖主键，不具有于主码的传递函数依赖。因此均属于 3NF 范式。

(二) 园林植物分类管理

2.1 植物分类表

（分类植物编号： classifyID 科名： familyName 属名： genusName 种名： speciesName
别名： alterName）

主码： 分类植物编号： classifyID

2.2 环境表

（分类植物编号： classifyID 生长环境： Envir 分布区域： area）

主码： 分类植物编号： classifyID 分布区域： area

数据

库系统课程设计报告

外码：分类植物编号： classifyID 分布区域： area

2.3 分布区域表

（分布区域编号： disAreaID 分类植物编号： classifyID 别名： alterName 分布区域： area
省： province 市： city ）

主码：分布区域编号： disAreaID

2.4 修改信息表

（修改编号： changeID 别名 alterName 创建人员： createPerson 创新时间：
innovationTime 更新时间： updateTime ）

主码：修改编号： changeID

第一范式要求所有属性不可再分。第二范式在第一范式的基础上消除非主属性对主码的部分函数依赖。第三范式在第二范式的基础上消除了非主属性对主码的传递函数依赖。

在园林植物分类管理中，每一个实体属性均为不可。每个非主属性不具有对于主码的部分函数依赖。每个非主属性都完全依赖主键，不具有于主码的传递函数依赖。因此均属于 3NF 范式。

(三) 园林植物养护管理

3.1 任务表

（任务编号： taskID， 任务名称： taskName， 任务描述： taskDes ）

主码：任务编号： taskID

3.2 养护表

（养护编号： mainID， 养护时间： mainTime， 养护地点： mainLoc， 养护人员：
mainPer， 养护对象： id ）

主码： 养护编号： mainID

外码： 养护对象： id

3.3 任务养护表

（任务编号： taskID， 养护编号： mainID ）

主码： 任务编号： taskID 和养护编号： mainID 的联合主码

外码： 任务编号： taskID， 养护编号： mainID

第一范式要求所有属性不可再分。第二范式在第一范式的基础上消除非主属性对主码的部分函数依赖。第三范式在第二范式的基础上消除了非主属性对主码的传递函数依赖。

数据

库系统课程设计报告

在园林植物养护管理中，每一个实体属性均为不可。每个非主属性不具有对于主码的部分函数依赖。每个非主属性都完全依赖主键，不具有于主码的传递函数依赖。因此均属于 3NF 范式。

(三) 园林植物病虫害防治管理

药剂表

(药剂编号： mediID,药剂名称： mediName,药剂用量： mediDos,作用期限： dataAct)

主码： 药剂编号： mediID

病虫害表

(病虫害编号 pestID,病虫害名称： pestName)

主码： 病虫害编号： pestID

防治表

(防治编号： conID,防治时间： conTime,防治地点 conLoc,防治人员 conPer,防治对象 id)

主码： 防治编号： conID

外码： 防治对象 id

虫害防治表

(防治编号： conID,药剂编号： mediID, 病虫害编号： pestID)

联合主码： 防治编号： conID,药剂编号： mediID, 病虫害编号： pestID

外码： 防治编号： conID,药剂编号： mediID, 病虫害编号： pestID

第一范式要求所有属性不可再分。第二范式在第一范式的基础上消除非主属性对主码的部分函数依赖。第三范式在第二范式的基础上消除了非主属性对主码的传递函数依赖。

在园林植物病虫害防治管理中，每一个实体属性均为不可。每个非主属性不具有对于主码的部分函数依赖。每个非主属性都完全依赖主键，不具有于主码的传递函数依赖。因此均属于 3NF 范式。

(五)园林植物监测管理

5.1 设备表

(设备编号： equipNum, 设备名称： equipName)

主码： 设备编号： equipNum

5.2 指标表

(指标编号： indexNum, 指标名称： indexName)

数据

库系统课程设计报告

主码：指标编号：indexNum

5.3 监测表

（监测编号：detecNum，监测时间：detecTime，监测地点：detecLoc，监测人员：
detecPer，监测对象：id）

主码：监测编号：detecNum

外码：监测对象：id

5.4 设备指标监测表（设备编号：equipNum，指标编号：indexNum，监测编号： detecNum）

主码：设备编号：equipNum、指标编号：indexNum 和监测编号：detecNum 的联合主码

外码：设备编号：equipNum，指标编号：indexNum，监测编号：detecNum

第一范式要求所有属性不可再分。第二范式在第一范式的基础上消除非主属性对主码的部分函数依赖。第三范式在第二范式的基础上消除了非主属性对主码的传递函数依赖。

在园林植物监测管理中，每一个实体属性均为不可。每个非主属性不具有对于主码的部分函数依赖。每个非主属性都完全依赖主键，不具有于主码的传递函数依赖。因此均属于 3NF 范式。

五、系统物理结构设计、持久层设计和关键业务代码（观测点 3.4）

软件名称：SQL Server

软件版本号：SQL Server2019

硬件环境：Windows11，SSMS18

（一）登录系统

用户表

创建用户表的 DDL 语句

```
CREATE TABLE usr (
    no INT PRIMARY KEY,
    username VARCHAR(255) NOT NULL,
    password VARCHAR(255) NOT NULL,
    role VARCHAR(255) NOT NULL
);
```

（二）园林植物基本信息管理

1. 数据表 DDL 语句

1.1 植物基本信息表

```
CREATE TABLE A_Plant (
    id CHAR(4) NOT NULL,
    altername VARCHAR(255) NOT NULL,
    familyname VARCHAR(255) NOT NULL,
    feature VARCHAR(255) DEFAULT NULL,
    value VARCHAR(255) DEFAULT NULL,
    skill VARCHAR(255) DEFAULT NULL,
    classifyID INT NOT NULL,
    disAreaID VARCHAR(255) NOT NULL,
    PRIMARY KEY (id),
    FOREIGN KEY (classifyID) REFERENCES B_Classification (classifyID),
    FOREIGN KEY (disAreaID) REFERENCES B_Area (disAreaID)
);
```

1.2 图片表

```
CREATE TABLE A_Picture (
```

```
    photoId CHAR(4) NOT NULL,  
    photoPlace VARCHAR(255) DEFAULT NULL,  
    photoDes VARCHAR(255) DEFAULT NULL,  
    photoPer VARCHAR(255) DEFAULT NULL,  
    photoPath VARCHAR(255) DEFAULT NULL,  
    PRIMARY KEY (photoId)  
);
```

1.3 植物配图表

```
CREATE TABLE A_Plant_Picture (  
    photoId CHAR(4) NOT NULL,  
    plantId CHAR(4) NOT NULL,  
    PRIMARY KEY (photoId, plantId),  
    FOREIGN KEY (photoId) REFERENCES A_Picture (photoId),  
    FOREIGN KEY (plantId) REFERENCES A_Plant (id)  
);
```

2. 索引 DDL 语句

```
CREATE INDEX idx_picture_person ON A_Picture (photoPer);  
CREATE INDEX idx_Plant_Kind ON A_Plant (classifyID);  
CREATE INDEX idx_Plant_Place ON A_Plant (disAreaID);  
CREATE INDEX idx_PlantPicture_photo ON A_Plant_Picture (photoId);  
CREATE INDEX idx_PlantPicture_plant ON A_Plant_Picture (plantId);
```

3. 视图 DDL 语句

3.1 植物配图视图

```
CREATE VIEW PlantPictureView AS
```

```
SELECT  
    p.id,  
    p.altername,  
    pic.photoId,  
    pic.photoPlace,  
    pic.photoPer,  
    pic.photoPath
```

```
FROM
```

```
    A_Plant p
```

```
JOIN
```

数据

库系统课程设计报告

```
A_Plant_Picture pp ON p.id = pp.plantId
```

JOIN

```
A_Picture pic ON pp.photoId = pic.photoId;
```

查询视图

```
SELECT * FROM PlantPictureView;
```

3.2 查询属于蔷薇科的植物信息

```
CREATE VIEW 蔷薇科_View AS
```

SELECT

```
p.id,  
p.altername,  
c1.alterName AS 科,  
c2.alterName AS 属
```

FROM

```
A_Plant p
```

JOIN

```
B_Classification c1 ON p.classifyID = c1.classifyID
```

JOIN

```
B_Classification c2 ON c1.classifyID = c2.classifyID_parent
```

WHERE

```
c1.alterName = '蔷薇科';
```

查询视图

```
SELECT * FROM 蔷薇科_View;
```

3.3 查询图片拍摄人员为小四的植物及图片相关信息

```
CREATE VIEW PhotoView AS
```

SELECT

```
p.altername,  
pic.photoPlace,  
pic.photoPer,  
pic.photoPath
```

FROM

```
A_Plant p
```

JOIN

```
A_Plant_Picture pp ON p.id = pp.plantId
```

JOIN

数据

库系统课程设计报告

```
A_Picture pic ON pp.photoId = pic.photoId  
WHERE
```

```
pic.photoPer = '小四';
```

查询视图

```
SELECT * FROM PhotoView;
```

3.4 查询植物应用价值中包含观赏的植物信息

```
CREATE VIEW PlantValue_View AS
```

```
SELECT
```

```
p.id,  
p.altername,  
p.feature,  
p.value,  
p.skill
```

```
FROM
```

```
A_Plant p
```

```
WHERE
```

```
p.value LIKE '%观赏%';
```

查询视图

```
SELECT * FROM PlantValue_View;
```

3.5 查询分布地区在江苏省的植物

```
CREATE VIEW Area_江苏省 AS
```

```
SELECT
```

```
p.id,  
p.altername,  
a.area,  
a.province,  
a.city
```

```
FROM
```

```
A_Plant p
```

```
JOIN
```

```
B_Area a ON p.disAreaID = a.disAreaID
```

```
WHERE
```

```
a.province = '江苏省';
```

查询视图

数据

库系统课程设计报告

```
SELECT * FROM Area_江苏省;
```

3.6 创建视图查询各科植物的总数

```
CREATE VIEW PlantCountByKind AS
```

```
SELECT
```

```
    B_Classification.classifyID AS kind_id,
```

```
    B_Classification.alterName AS kind_name,
```

```
    COUNT(A_Plant.id) AS plant_count
```

```
FROM
```

```
    B_Classification
```

```
LEFT JOIN
```

```
    A_Plant ON B_Classification.classifyID = A_Plant.classifyID
```

```
WHERE
```

```
    B_Classification.classifyType = '科'
```

```
GROUP BY
```

```
    B_Classification.classifyID,
```

```
    B_Classification.alterName;
```

4. DAO 基础实现代码

登录实体

数据

库系统课程设计报告

```
package Entity;
public class Login {
    private String no;
    private String username;
    private String password;
    private String role;
    public String getNo() { return no; }
    public void setNo(String no) { this.no = no; }
    public String getUsername() { return username; }
    public void setUsername(String username) { this.username = username; }
    public String getPassword() { return password; }
    public void setPassword(String password) { this.password = password; }
    public String getRole() { return role; }
    public void setRole(String role) { this.role = role; }
    @Override
    public String toString() {
        return "LoginDetail{" +
            "no='" + no + '\'' +
            ", username='" + username + '\'' +
            ", password='" + role + '\'' +
            ", role='" + no + '\'' +
            '}';
    }
}
```

登录 DAO

```
package Dao;
import Entity.Login;
public interface LoginDao {
    Login getLoginByUsername(String username); // 根据username查询Login
}
```

登录 DAOImpl

数据

库系统课程设计报告

```
package Dao.Impl;
import Dao.DaoBase;
import Dao.LoginDao;
import Entity.Login;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
public class LoginDaoImpl extends DaoBase implements LoginDao {
    /**
     * 根据用户输入的username查询role
     *
     * @param username 用户输入的username
     * @return LoginDetail对象，包含了查询到的no和role
     */
    @Override
    public Login getLoginByUsername(String username) {
        Connection con = null;
        try {
            con = getConnection();
            String sql1 = "SELECT username, role, no,password FROM usr WHERE username=?;";
            PreparedStatement psmt = con.prepareStatement(sql1);
            psmt.setString(parameterIndex: 1, username);
            ResultSet rs = psmt.executeQuery();
            Login loginDetail = new Login();
            if (rs.next()) {
                loginDetail.setUsername(username);
                loginDetail.setNo(rs.getString(columnLabel: "no"));
                loginDetail.setRole(rs.getString(columnLabel: "role"));
                loginDetail.setPassword(rs.getString(columnLabel: "password"));
            } else {
                System.out.println("未查询到该用户名的信息");
            }
            // 如果能查到不止一条数据
            if (rs.next() == true) {
                System.out.println("查询错误");
                System.exit(status: 0);
            }
            psmt.close();
            return loginDetail;
        } catch (Exception e) {
            e.printStackTrace();
            return null;
        } finally {
            try {
                con.close();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }
}
```

植物实体

数据

库系统课程设计报告

```
package Entity;
public class Plant {
    public String id;
    public String alternname;      //植物名称
    public String familyname;     //植物科名
    public String feature;        //植物特征
    public String value;          //植物应用价值
    public String skill;          //栽培要点
    public int classifyID;        //分类编号
    public String disAreaID;      //地区编号
    public String getId() { return id; }
    public void setId(String id) {
        this.id = id;
    }
    public String getAlternname() { return alternname; }
    public void setAlternname(String alternname) { this.alternname = alternname; }
    public String getFamilyname() { return familyname; }
    public void setFamilyname(String familyname) { this.familyname = familyname; }
    public String getFeature() { return feature; }
    public void setFeature(String feature) { this.feature = feature; }
    public String getValue() { return value; }
    public void setValue(String value) { this.value = value; }
    public String getSkill() { return skill; }
    public void setSkill(String skill) { this.skill = skill; }
    public int getClassifyID() { return classifyID; }
    public void setClassifyID(int classifyID) { this.classifyID = classifyID; }
    public String getDisAreaID() { return disAreaID; }
    public void setDisAreaID(String disAreaID) { this.disAreaID = disAreaID; }
}
```

植物 DAO

```
package Dao;
import Entity.Plant;
import java.util.List;
public interface PlantDao {
    boolean addPlant(Plant plant);           //添加植物信息
    boolean deletePlant(Plant plant);         //删除植物信息
    void updatePlant(Plant plant, String[] fieldsToUpdate, String newValue); //修改植物信息
    List<Plant> findPlant(Plant plant);       //查看植物基本信息
}
```

数据

库系统课程设计报告

```
package Dao.Impl;
import ...
public class PlantDaoImpl extends DaoBase implements PlantDao {
    @Override
    /*PreparedStatement 是 Java JDBC 中的一个接口，用于表示预编译的 SQL 语句的对象。
    它是 java.sql.PreparedStatement 接口的实例。通过使用 PreparedStatement,
    可以在执行 SQL 语句之前预先编译它，提高执行效率，并且可以通过占位符（?）来动态设置参数，避免 SQL 注入攻击
    */
    public boolean addPlant(Plant plant) {
        boolean check = false;
        String sql = "INSERT INTO A_Plant VALUES(?,?,?,?,?,?)";
        Connection connection = getConnection();
        try {
            PreparedStatement preparedStatement = connection.prepareStatement(sql);
            preparedStatement.setString(parameterIndex: 1, plant.getId());
            preparedStatement.setString(parameterIndex: 2, plant.getAltername());
            preparedStatement.setString(parameterIndex: 3, plant.getFamilyname());
            preparedStatement.setString(parameterIndex: 4, plant.getFeature());
            preparedStatement.setString(parameterIndex: 5, plant.getValue());
            preparedStatement.setString(parameterIndex: 6, plant.getSkill());
            preparedStatement.setInt(parameterIndex: 7, plant.getClassifyID());
            preparedStatement.setString(parameterIndex: 8, plant.getDisAreaID());
            try {
                int i = preparedStatement.executeUpdate();
                System.out.println("插入影响的行数: " + i);
                if (i > 0) {
                    check = true;
                }
            }
            } catch (SQLException e) {
                e.printStackTrace();
                System.out.println("插入失败，错误信息: " + e.getMessage());
            }
            preparedStatement.close();
        }
        catch (Exception e) {
            System.out.println("插入失败，可能存在相同活动序号");
        } finally {
            // 在适当的地方关闭 PreparedStatement 和 Connection
            try {
                if (connection != null) {
                    connection.close();
                }
            } catch (Exception e) {
                e.printStackTrace();
            }
            // 关闭 Connection 等其他资源
        }
        return check;
    }
    @Override
    public boolean deletePlant(Plant plant) {
        boolean check = false;
        String sql = "delete from A_Plant where id = ?";
        Connection connection = getConnection();
        try {
            PreparedStatement preparedStatement = connection.prepareStatement(sql);
            preparedStatement.setString(parameterIndex: 1, plant.getId());
            int i = preparedStatement.executeUpdate();
```

数据

库系统课程设计报告

```
        if (i > 0) {
            check = true;
            System.out.println("Delete successful!");
        }
        preparedStatement.close();
    } catch (Exception e){
        System.out.println("插入失败，可能存在相同活动序号");
    } finally{
        // 在适当的地方关闭 PreparedStatement 和 Connection
        try {
            if (connection != null) {
                connection.close();
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        // 关闭 Connection 等其他资源
    }
    return check;
}
@Override
public void updatePlant(Plant plant, String[] fieldsToUpdate, String newValue) {
    StringBuilder setClause = new StringBuilder("SET ");
    for (String field : fieldsToUpdate) {
        setClause.append(field).append("=?,");
    }
    setClause.deleteCharAt( index: setClause.length() - 1); // 移除最后一个逗号
}
```

数据

库系统课程设计报告

```
// 构建 SQL 语句
String sql = "UPDATE A_Plant " + setClause + " WHERE id = ?";
Connection connection = getConnection();
PreparedStatement preparedStatement = null;
try {
    preparedStatement = connection.prepareStatement(sql);
    int paramIndex = 1;
    for (String field : fieldsToUpdate) {
        switch (field.trim()) {
            case "altername":
                preparedStatement.setString(paramIndex++, newValue);
                break;
            case "familyname":
                preparedStatement.setString(paramIndex++, newValue);
                break;
            case "feature":
                preparedStatement.setString(paramIndex++, newValue);
                break;
            case "value":
                preparedStatement.setString(paramIndex++, newValue);
                break;
            case "skill":
                preparedStatement.setString(paramIndex++, newValue);
                break;
            case "classifyID":
                preparedStatement.setString(paramIndex++, newValue);
                break;
            case "disAreaID":
                preparedStatement.setString(paramIndex++, newValue);
        }
    }
}
```

数据

库系统课程设计报告

```
        break;
    default:
        System.out.println("Invalid field: " + field);
        break;
    }
}
preparedStatement.setString(paramIndex, plant.getId());
preparedStatement.executeUpdate();
preparedStatement.close();
connection.close();
System.out.println("Update successful.");
} catch (SQLException e) {
    throw new RuntimeException(e);
}
}

@Override
public List<Plant> findPlant(Plant plant) {
    ArrayList<Plant> arrayList = new ArrayList<>();
    String sql = "select p.*\n" + //, pic.*"
        "FROM A_Plant p\n" +
        "JOIN A_Plant_Picture pp ON p.id = pp.plantId\n" +
        "JOIN A_Picture pic ON pp.photoId = pic.photoId\n" +
        "where id = ?;";
    Connection connection = getConnection();
    PreparedStatement preparedStatement = null;
    try {
        preparedStatement = connection.prepareStatement(sql);
        preparedStatement.setString(parameterIndex: 1, plant.getId());
```

数据

库系统课程设计报告

```
ResultSet resultSet = preparedStatement.executeQuery();
while(resultSet.next()){
    plant.setId(resultSet.getString( columnLabel: "id"));
    plant.setAlternename(resultSet.getString( columnLabel: "alternename"));
    plant.setFamilyname(resultSet.getString( columnLabel: "familyname"));
    plant.setFeature(resultSet.getString( columnLabel: "feature"));
    plant.setValue(resultSet.getString( columnLabel: "value"));
    plant.setSkill(resultSet.getString( columnLabel: "skill"));
    plant.setClassifyID(resultSet.getInt( columnLabel: "classifyID"));
    plant.setDisAreaID(resultSet.getString( columnLabel: "disAreaID"));
    arrayList.add(plant);
}
System.out.println("Find successful!");
preparedStatement.close();
connection.close();
} catch (SQLException e) {
    throw new RuntimeException(e);
}
return arrayList;
}
```

图片实体

数据

库系统课程设计报告

```
package Entity;
public class Picture {
    public String photoId;          //图片编号
    public String photoPlace;        //配图拍摄地点
    public String photoDes;          //配图描述
    public String photoPer;          //配图拍摄人
    public String photoPath;         //图片路径
    public String altername;         //植物名称
    public String getphotoId() { return photoId; }
    public void setphotoId(String photoId) {this.photoId = photoId;}
    public String getphotoPlace() { return photoPlace; }
    public void setphotoPlace(String photoPlace) { this.photoPlace = photoPlace; }
    public String getphotoDes() { return photoDes; }
    public void setphotoDes(String photoDes) { this.photoDes = photoDes; }
    public String getphotoPer() { return photoPer; }
    public void setphotoPer(String photoPer) { this.photoPer = photoPer; }
    public String getphotoPath() { return photoPath; }
    public void setphotoPath(String photoPath) { this.photoPath = photoPath; }
    public String getAltername() { return photoPath; }
    public void setAltername(String photoPath) { this.photoPath = photoPath; }
    @Override
    public String toString() {
        return "植物图片{" +
            "图片id='"+photoId+'\'' +
            ", 配图拍摄地点='"+photoPlace+'\'' +
            ", 配图描述='"+photoDes+'\'' +
            ", 拍摄人员='"+photoPer+'\'' +
            ", 图片路径='"+photoPath+'\'' +
            ", 植物名称='"+altername+'\'' +
    }
}
```

图片 DAO

```
package Dao;
import Entity.Plant;
import java.util.List;
public interface PlantDao {
    boolean addPlant(Plant plant);           //添加植物信息
    boolean deletePlant(Plant plant);         //删除植物信息
    void updatePlant(Plant plant, String[] fieldsToUpdate, String newValue); //修改植物信息
    List<Plant> findPlant(Plant plant);      //查看植物基本信息
}
```

图片 DAOImpl

数据

库系统课程设计报告

```
package Dao.Impl;
import ...
public class PictureDaoImpl extends DaoBase implements PictureDao {
    @Override
    public void addPicture(Picture picture) {
        Connection connection = getConnection();
        try {
            String sql1 = "INSERT INTO A_Picture (alternename,photoId, photoDes, photoPlace, photoPer, photoPath) VALUES(?,?,?,?,?,?)";
            //插入 A_Picture 表的数据
            PreparedStatement statement1 = connection.prepareStatement(sql1);
            statement1.setString( parameterIndex: 1, picture.getAlternename());
            statement1.setString( parameterIndex: 2, picture.getphotoId());
            statement1.setString( parameterIndex: 3, picture.getphotoDes());
            statement1.setString( parameterIndex: 4, picture.getphotoPlace());
            statement1.setString( parameterIndex: 5, picture.getphotoPer());
            statement1.executeUpdate();
            statement1.close();
            /*String sql2 = "DECLARE @newPictureId INT;\n" +
             "SET @newPictureId = SCOPE_IDENTITY();";*/
            String sql2 = "SELECT SCOPE_IDENTITY();";
            Statement statement2 = connection.createStatement();
            ResultSet resultSet = statement2.executeQuery(sql2);
            String newPictureId = null;
            if (resultSet.next()) {
                newPictureId = resultSet.getString( columnIndex: 1);
            }
            statement2.close();
        }

        String sql3 = "INSERT INTO A_Plant_Picture (plantId, photoId)\n" +
            "SELECT p.id, @newPictureId\n" +
            "FROM A_Plant p\n" +
            "WHERE p.alternename = (SELECT alternename FROM A_Picture WHERE photoId = @newPictureId);";
        //插入 Plant_Picture 表的关联关系
        PreparedStatement statement3 = connection.prepareStatement(sql3);
        statement3.setString( parameterIndex: 1, newPictureId);
        statement3.setString( parameterIndex: 2, picture.getphotoId());
        statement3.executeUpdate();
        statement3.close();

        System.out.println("Add successful!");
        connection.close();
    }catch (SQLException e) {
        throw new RuntimeException(e);
    }
}

@Override
public void deletePicture(Picture picture, String plantNameToDelete, String pictureIdToDelete) {
    Connection connection = getConnection();
    try {
        // 删除图片植物关联表数据
        String sql1 = "DELETE FROM A_Plant_Picture WHERE photoId = ?;" ;
        //插入 A_Picture 表的数据
        PreparedStatement statement1 = connection.prepareStatement(sql1);
        statement1.setString( parameterIndex: 1, pictureIdToDelete);
        statement1.executeUpdate();
        statement1.close();
    }
}
```

数据

库系统课程设计报告

```
// 删除图片表数据
String sql2 = "DELETE FROM A_Picture WHERE altername = ? AND photoId = ?";
PreparedStatement statement2 = connection.prepareStatement(sql2);
statement2.setString( parameterIndex: 1, plantNameToDelete);
statement2.setString( parameterIndex: 2, pictureIdToDelete);
statement2.executeUpdate();
statement2.close();
System.out.println("Delete successful!");
connection.close();

} catch (SQLException e) {
    throw new RuntimeException(e);
}

}

@Override
public void updatePicture(Picture picture, String[] fieldsToUpdate, String newValue) {
    StringBuilder setClause = new StringBuilder("SET ");
    for (String field : fieldsToUpdate) {
        setClause.append(field).append("=?,");
    }
    setClause.deleteCharAt( index: setClause.length() - 1); // 移除最后一个逗号

    // 构建 SQL 语句
    String sql = "UPDATE A_Picture " + setClause + " WHERE id=?";
    Connection connection = getConnection();
    PreparedStatement preparedStatement = null;
    try {
        preparedStatement = connection.prepareStatement(sql);
        int paramIndex = 1;

        for (String field : fieldsToUpdate) {
            switch (field.trim()) {
                case "photoPlace":
                    preparedStatement.setString(paramIndex++, newValue);
                    break;
                case "photoDes":
                    preparedStatement.setString(paramIndex++, newValue);
                    break;
                case "photoPer":
                    preparedStatement.setString(paramIndex++, newValue);
                    break;
                case "photoPath":
                    preparedStatement.setString(paramIndex++, newValue);
                    break;
                default:
                    System.out.println("Invalid field: " + field);
                    break;
            }
        }
        preparedStatement.setString(paramIndex, picture.getphotoId());
        preparedStatement.executeUpdate();
        preparedStatement.close();
        connection.close();
        System.out.println("Update successful.");
    } catch (SQLException e) {
        throw new RuntimeException(e);
    }
}
```

```

@Override
public void findPicture(Picture picture) {
    String sql = " SELECT p.id,p.altername," +
        "pic.photoId,pic.photoPlace,pic.photoPer,pic.photoPath\n" +
        "FROM A_Plant p\n" +
        "JOIN A_Plant_Picture pp ON p.id = pp.plantId\n" +
        "JOIN A_Picture pic ON pp.photoId = pic.photoId;" ;
    Connection connection = getConnection();
    PreparedStatement preparedStatement = null;
    try {
        preparedStatement = connection.prepareStatement(sql);
        preparedStatement.setString( parameterIndex: 1, picture.getphotoId());
        preparedStatement.executeUpdate();
        System.out.println("Find successful!");
        preparedStatement.close();
        connection.close();
    } catch (SQLException e) {
        throw new RuntimeException(e);
    }
}
}

```

(三)园林植物分类管理

1. 数据表 DDL 语句

1. 数据表 DDL 语句

1.1 植物分类表

```

CREATE TABLE B_Classification (
    classifyID INT PRIMARY KEY,
    alterName VARCHAR(255),
    classifyType VARCHAR(50),
    classifyID_parent INT,
    FOREIGN KEY (classifyID_parent) REFERENCES B_Classification(classifyID)
);

```

1.2 分布区域表

```

CREATE TABLE B_Area (
    areaID INT PRIMARY KEY,
    disAreaID VARCHAR(255),
    alterName VARCHAR(255),
    area nchar(10),
    province VARCHAR(255),

```

数据

库系统课程设计报告

```
    city VARCHAR(255),  
    FOREIGN KEY (areaID) REFERENCES B_Classification(classifyID)  
);
```

1.3 修改信息表

```
CREATE TABLE B_Change (  
    changeID INT PRIMARY KEY,  
    alterName VARCHAR(255),  
    createPerson VARCHAR(255),  
    innovationTime DATETIME,  
    updateTime DATETIME,  
);
```

1.4 生长环境表

```
CREATE TABLE B_Envir(  
    envirID INT PRIMARY KEY,  
    classifyID INT,  
    envir VARCHAR(255),  
    area nchar(10),  
    FOREIGN KEY (classifyID) REFERENCES B_Classification(classifyID)  
);
```

2. 索引 DDL 语句

2.1 植物的别名索引

```
CREATE INDEX idx_classifyID ON B_Classification (alterName);
```

2.2 植物的地区编号索引

```
CREATE INDEX idx_disAreaID ON B_Area (disAreaID);
```

2.3 植物的修改编号索引

```
CREATE INDEX idx_changeID ON B_Change (changeID);
```

2.4 人员名称索引

```
CREATE INDEX idx_createPerson ON B_Change (createPerson);
```

2.5 生长环境索引

```
CREATE INDEX idx_envir ON B_Envir (Envir);
```

3. 视图 DDL 语句

数据

库系统课程设计报告

3.1 查询某植物的所有分类信息

```
CREATE VIEW PlantInfo AS
SELECT  B_Classification.classifyID,
        B_Classification.alterName,
        B_Classification.classifyType,
        B_Classification.classifyID_parent,
        B_Area.disAreaID,
        B_Area.area,
        B_Envir.Envir
FROM B_Classification
JOIN B_Area ON B_Classification.classifyID = B_Area.disAreaID
JOIN B_Envir ON B_Classification.classifyID = B_Envir.classifyID;
```

3.2 查询植物的分布地区与生长环境

```
CREATE VIEW ALLinfor AS
SELECT B_Classification.classifyID,
        B_Classification.alterName,
        B_Classification.classifyType,
        B_Envir.Envir,
        B_Envir.area
FROM B_Classification
JOIN B_Envir ON B_Classification.classifyID = B_Envir.classifyID;
```

3.3 查找属于温带季风气候丘陵这个环境的植物以及其信息

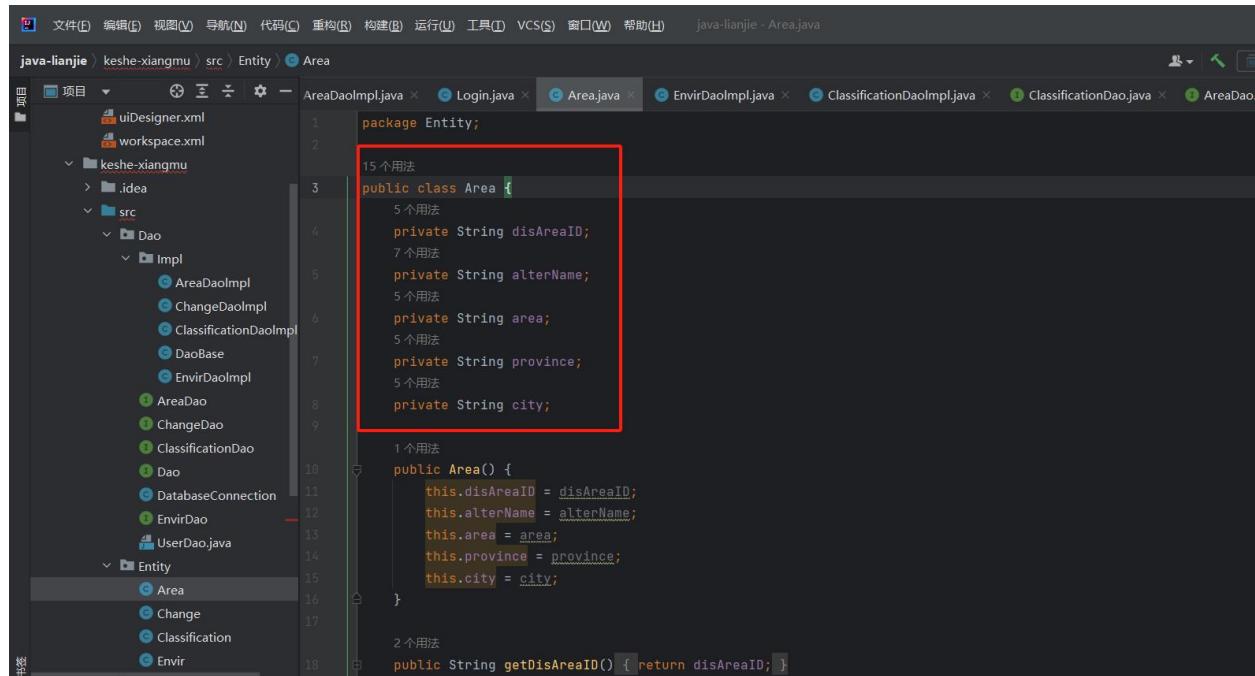
```
CREATE VIEW findInfor AS
SELECT  B_Change.alterName,
        B_Area.area,
        B_Change.createPerson AS Creator,
        B_Change.innovationTime AS CreationTime
FROM B_Change
JOIN B_Area ON B_Change.changeID = B_Area.disAreaID
JOIN B_Envir ON B_Area.area = B_Envir.area
WHERE B_Envir.Envir = '温带季风气候丘陵';
```

4.DAO 基础实现代码

DAO 实体： Area

数据

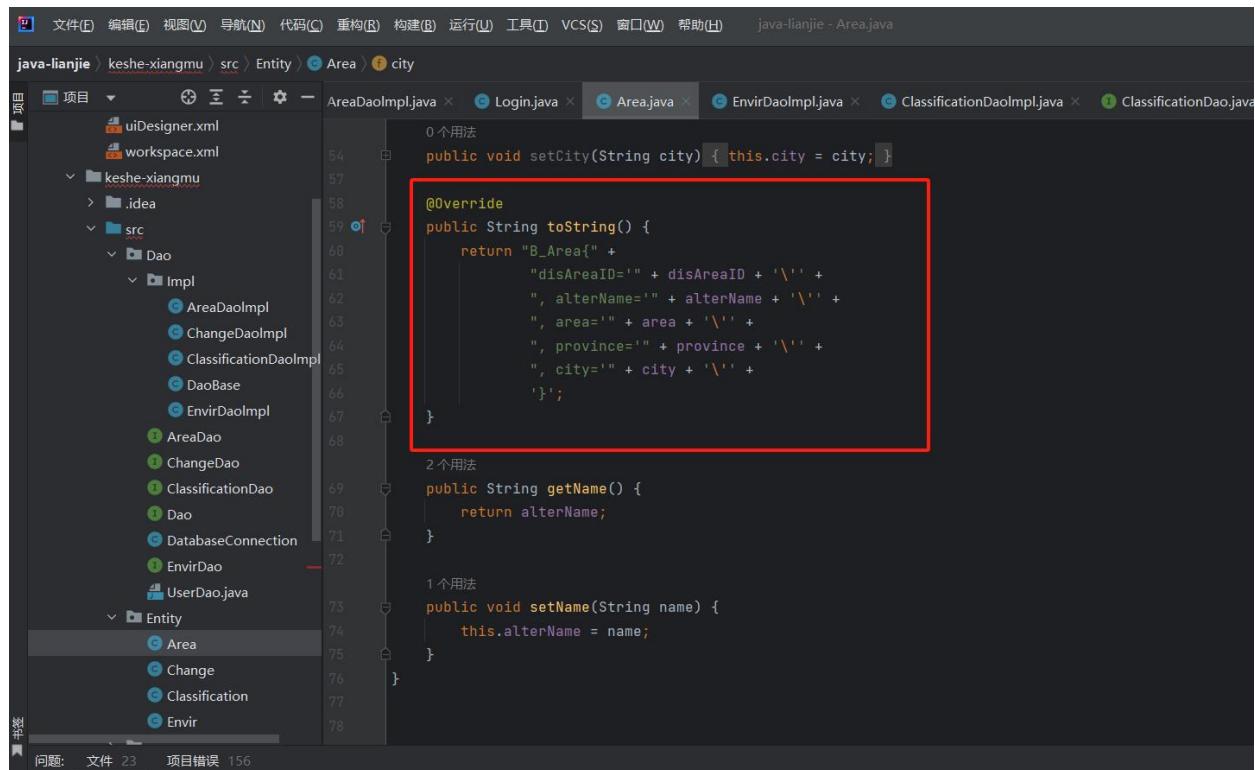
库系统课程设计报告



java-lianjie > keshe-xiangmu > src > Entity > Area

```
package Entity;
public class Area {
    private String disAreaID;
    private String alterName;
    private String area;
    private String province;
    private String city;
}
public Area() {
    this.disAreaID = disAreaID;
    this.alterName = alterName;
    this.area = area;
    this.province = province;
    this.city = city;
}
public String getDisAreaID() { return disAreaID; }
```

The code block shows the definition of the Area class with its fields and constructor. A red box highlights the constructor and the getDisAreaID method.



java-lianjie > keshe-xiangmu > src > Entity > Area > f: city

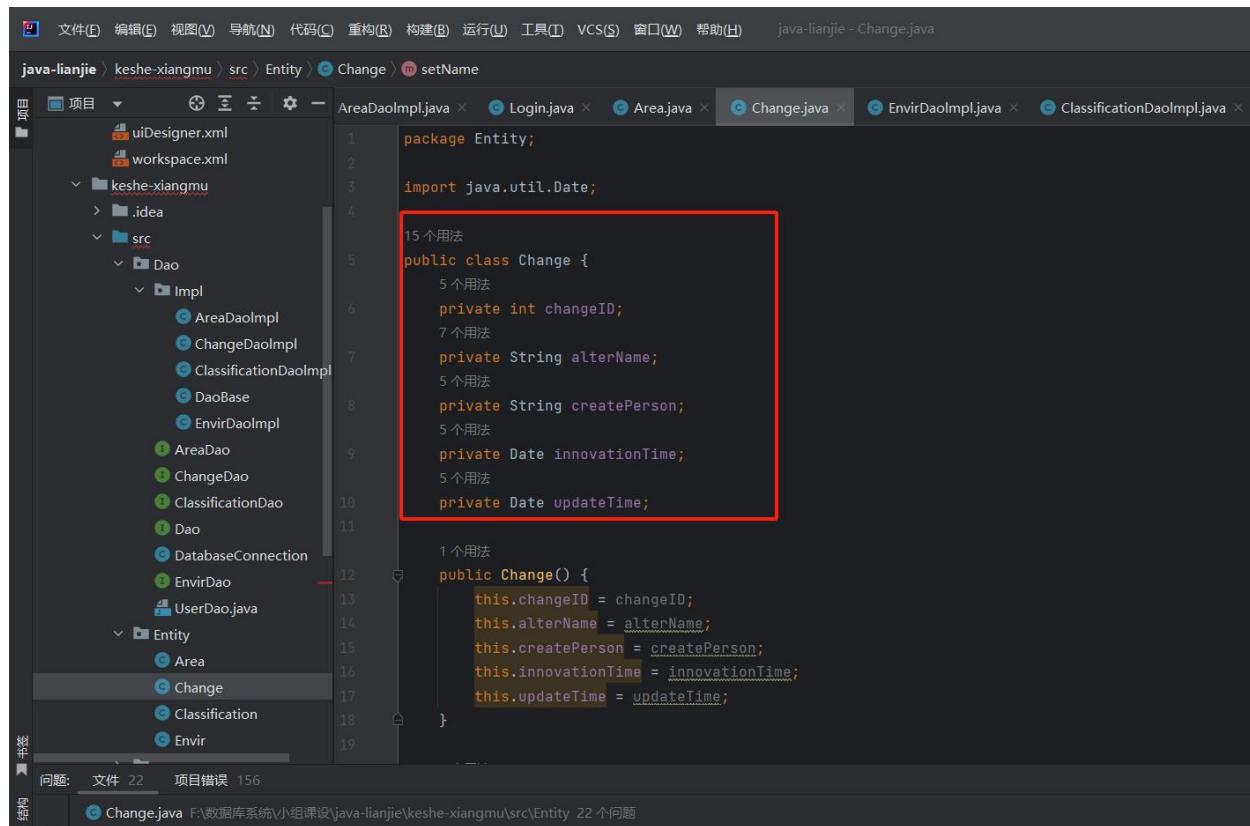
```
public void setCity(String city) { this.city = city; }
@Override
public String toString() {
    return "B_Area{" +
        "disAreaID=''" + disAreaID + '\'' +
        ", alterName=''" + alterName + '\'' +
        ", area=''" + area + '\'' +
        ", province=''" + province + '\'' +
        ", city=''" + city + '\'';
}
public String getName() {
    return alterName;
}
public void setName(String name) {
    this.alterName = name;
}
```

The code block shows the Area class with annotations (@Override) and methods (setCity, toString, getName, setName). A red box highlights the toString method.

DAO 实体: Change

数据

库系统课程设计报告

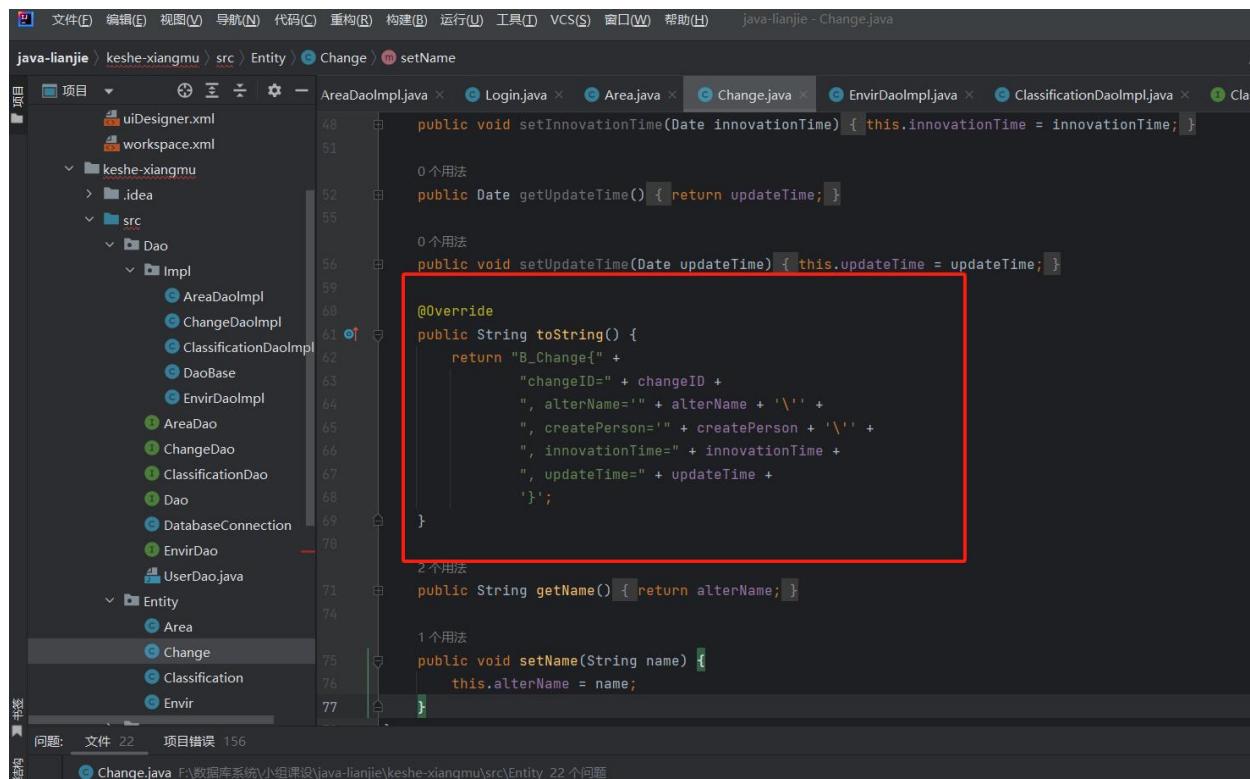


```
package Entity;

import java.util.Date;

15个用法
public class Change {
    5个用法
    private int changeID;
    7个用法
    private String alterName;
    5个用法
    private String createPerson;
    5个用法
    private Date innovationTime;
    5个用法
    private Date updateTime;
}

1个用法
public Change() {
    this.changeID = changeID;
    this.alterName = alterName;
    this.createPerson = createPerson;
    this.innovationTime = innovationTime;
    this.updateTime = updateTime;
}
```



```
public void setInnovationTime(Date innovationTime) { this.innovationTime = innovationTime; }

0个用法
public Date getUpdateTime() { return updateTime; }

0个用法
public void setUpdateTime(Date updateTime) { this.updateTime = updateTime; }

@Override
public String toString() {
    return "B_Change{" +
        "changeID=" + changeID +
        ", alterName='" + alterName + '\'' +
        ", createPerson='" + createPerson + '\'' +
        ", innovationTime=" + innovationTime +
        ", updateTime=" + updateTime +
        '}';
}

2个用法
public String getName() { return alterName; }

1个用法
public void setName(String name) {
    this.alterName = name;
}
```

DAO 实体: Change

数据

库系统课程设计报告

The screenshot shows the IntelliJ IDEA interface with the Classification.java file open. The code defines a Classification class with private fields classifyID, alterName, classifyType, and classifyID_parent. It includes a constructor, a toString method, and a getName method. The IDE's code analysis tool highlights various parts of the code with annotations like '11 个用法' (11 uses) and '4 个用法' (4 uses).

```
package Entity;
public class Classification {
    private int classifyID;
    private String alterName;
    private String classifyType;
    private int classifyID_parent;
    public Classification() {}
    public Classification(int classifyID, String alterName, String classifyType, int classifyID_parent) {
        this.classifyID = classifyID;
        this.alterName = alterName;
        this.classifyType = classifyType;
        this.classifyID_parent = classifyID_parent;
    }
    public String toString() {
        return "Classification{" +
            "classifyID=" + classifyID +
            ", alterName='" + alterName + '\'' +
            ", classifyType='" + classifyType + '\'' +
            ", classifyID_parent=" + classifyID_parent +
            '}';
    }
    public String getName() {
        return alterName;
    }
    public void setName(String name) {
        this.alterName = name;
    }
}
```

This screenshot shows the same Classification.java file, but the setName method is highlighted with a red border. The code for setName is as follows:

```
public void setName(String name) {
    this.alterName = name;
}
```

DAO 实体: Classification

数据

库系统课程设计报告

```
AreaDaopl.java × Login.java × Area.java × Change.java × Classification.java × EnvDaopl.java × ClassificationDao.java ×  
3  public class Classification {  
4      4个用法  
5      private int classifyID;  
6      6个用法  
7      private String alterName;  
8      4个用法  
9      private String classifyType;  
10     4个用法  
11     private int classifyID_parent;  
12  
13     1个用法  
14     public Classification() {  
15     }  
16  
17     0个用法  
18     public Classification(int classifyID, String alterName, String classifyType, int classifyID_parent) {  
19         this.classifyID = classifyID;  
20         this.alterName = alterName;  
21         this.classifyType = classifyType;  
22         this.classifyID_parent = classifyID_parent;  
23     }  
24  
25     2个用法  
26     public int getClassifyID() {  
27         return classifyID;  
28     }  
29
```

```
AreaDaopl.java × Login.java × Area.java × Change.java × Classification.java × EnvDaopl.java × ClassificationDao.java ×  
39     public void setClassifyType(String classifyType) {  
40         this.classifyType = classifyType;  
41     }  
42  
43     0个用法  
44     public int getClassifyID_parent() {  
45         return classifyID_parent;  
46     }  
47  
48     0个用法  
49     public void setClassifyID_parent(int classifyID_parent) {  
50         this.classifyID_parent = classifyID_parent;  
51     }  
52     public String toString() {  
53         return "Classification{" +  
54             "classifyID=" + classifyID +  
55             ", alterName='" + alterName + '\'' +  
56             ", classifyType='" + classifyType + '\'' +  
57             ", classifyID_parent=" + classifyID_parent +  
58             '}';  
59     }  
60  
61     2个用法  
62     public String getName() {  
63         return alterName;  
64     }  
65
```

连接:

数据

库系统课程设计报告

```
package Dao;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

2个用法
public class DatabaseConnection {
    1个用法
    private static final String DB_URL = "jdbc:sqlserver://localhost:1433;databaseName=PLANT";
    1个用法
    private static final String USER = "zpf";
    1个用法
    private static final String PASSWORD = "123456";
    9个用法
    public static Connection getConnection() throws SQLException {
        return DriverManager.getConnection(DB_URL, USER, PASSWORD);
    }
}
```

实体的实现

```
package Dao;

import Entity.Area;

3个用法 1个实现
public interface AreaDao {
    0个用法 1个实现
    void addArea(Area area);
    0个用法 1个实现
    void deleteArea(String disAreaID);
    0个用法 1个实现
    void updateArea(Area area);
    0个用法 1个实现
    Area findArea(String disAreaID);
}
```

数据

库系统课程设计报告

```
package Dao;

import Entity.Change;

3个用法 1个实现
public interface ChangeDao {
    0个用法 1个实现
    void addChange(Change change);
    0个用法 1个实现
    💡 void deleteChange(int changeID);
    0个用法 1个实现
    void updateChange(Change change);
    0个用法 1个实现
    Change findChange(int changeID);
}
```

```
1 package Dao;
2
3 import Entity.Envir;
4 0个用法
5 public interface EnvirDao {
    0个用法
    💡 void addEnvir(Envir envir);
    0个用法
    void updateEnvir(Envir envir);
    0个用法
    void deleteEnvir(int classifyID);
    0个用法
    Envir findEnvir(int classifyID);
}
10
```

Impl 文件：

AreaDaolmpl

```
1 package DaoImpl;
2
3 import Entity.Area;
4
5 import java.sql.Connection;
6 import java.sql.PreparedStatement;
7 import java.sql.ResultSet;
8 import java.sql.SQLException;
9
10 import static Dao.DatabaseConnection.getConnection;
11
12 0个用法
13 public class AreaDaolmpl {
14     0个用法
15     @ public void insertArea(Area area) {
16         Connection con = null;
17         try {
18             con = getConnection();
19             String sql = "INSERT INTO areas (disAreaID, name) VALUES (?, ?)";
20             PreparedStatement psmt = con.prepareStatement(sql);
21             psmt.setString( parameterIndex: 1, area.getDisAreaID());
22             psmt.setString( parameterIndex: 2, area.getName());
23             int rowsAffected = psmt.executeUpdate();
24             if (rowsAffected > 0) {
25                 System.out.println("insertArea成功");
26             } else {
27             }
28         } catch (SQLException e) {
29             e.printStackTrace();
30         } finally {
31             if (con != null) {
32                 try {
33                     con.close();
34                 } catch (SQLException e) {
35                     e.printStackTrace();
36                 }
37             }
38         }
39     }
40 }
```

数据

库系统课程设计报告

```
3             System.out.println("insertArea成功");
4         } else {
5             System.out.println("insertArea失败");
6         }
7         psmt.close();
8     } catch (SQLException e) {
9         e.printStackTrace();
10    } finally {
11        try {
12            if (con != null) {
13                con.close();
14            }
15        } catch (SQLException e) {
16            e.printStackTrace();
17        }
18    }
19 }

0个用法
20
21 public Area findArea(String disAreaID) {
22     Connection con = null;
23     Area area = null;
24     try {
25         con = getConnection();
26         String sql = "SELECT * FROM areas WHERE disAreaID = ?";
27         PreparedStatement psmt = con.prepareStatement(sql);
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70 }
```

```
46     String sql = "SELECT * FROM areas WHERE disAreaID = ?";
47     PreparedStatement psmt = con.prepareStatement(sql);
48     psmt.setString( parameterIndex: 1, disAreaID);
49     ResultSet rs = psmt.executeQuery();
50     if (rs.next()) {
51         area = new Area();
52         area.setDisAreaID(rs.getString( columnLabel: "disAreaID"));
53         area.setName(rs.getString( columnLabel: "name"));
54     }
55     rs.close();
56     psmt.close();
57 } catch (SQLException e) {
58     e.printStackTrace();
59 } finally {
60     try {
61         if (con != null) {
62             con.close();
63         }
64     } catch (SQLException e) {
65         e.printStackTrace();
66     }
67 }
68 return area;
69 }
70 }

0个用法
```

数据

库系统课程设计报告

```
0个用法
public void updateArea(Area area) {
    Connection con = null;
    try {
        con = getConnection();
        String sql = "UPDATE areas SET name = ? WHERE disAreaID = ?";
        PreparedStatement psmt = con.prepareStatement(sql);
        psmt.setString(parameterIndex: 1, area.getName());
        psmt.setString(parameterIndex: 2, area.getDisAreaID());
        int rowsAffected = psmt.executeUpdate();
        if (rowsAffected > 0) {
            System.out.println("updateArea成功");
        } else {
            System.out.println("updateArea失败");
        }
        psmt.close();
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        try {
            if (con != null) {
                con.close();
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

ChangeDaolmpl

数据

库系统课程设计报告

```
1 package Dao.Impl;
2
3 import java.sql.Connection;
4 import java.sql.PreparedStatement;
5 import java.sql.ResultSet;
6 import java.sql.SQLException;
7
8 import static Dao.DatabaseConnection.getConnection;
9 import Entity.Change;
10
11 1个用法
12 public class ChangeDaoimpl {
13     0个用法
14     @
15     public void addChange(Change change) {
16         Connection con = null;
17         try {
18             con = getConnection();
19             String sql = "INSERT INTO changes (changeID, name) VALUES (?, ?)";
20             PreparedStatement psmt = con.prepareStatement(sql);
21             psmt.setInt( parameterIndex: 1,
22                         change.getChangeID());
23             psmt.setString( parameterIndex: 2, change.getName());
24             int rowsAffected = psmt.executeUpdate();
25             if (rowsAffected > 0) {
26                 System.out.println("addChange成功");
27             } else {
28                 System.out.println("addChange失败");
29             }
30         } catch (SQLException e) {
31             e.printStackTrace();
32         } finally {
33             try {
34                 if (con != null) {
35                     con.close();
36                 }
37             } catch (SQLException e) {
38                 e.printStackTrace();
39             }
40         }
41     }
42     public void deleteChange(int changeID) {
43         Connection con = null;
44         try {
45             con = getConnection();
46             String sql = "DELETE FROM changes WHERE changeID = ?";
47             PreparedStatement psmt = con.prepareStatement(sql);
48             psmt.setInt( parameterIndex: 1, changeID);
49             int rowsAffected = psmt.executeUpdate();
50             if (rowsAffected > 0) {
51                 System.out.println("deleteChange成功");
52             } else {
53                 System.out.println("deleteChange失败");
54             }
55             psmt.close();
56         } catch (SQLException e) {
57             e.printStackTrace();
58         } finally {
59             try {
60                 if (con != null) {
61                     con.close();
62                 }
63             } catch (SQLException e) {
64                 e.printStackTrace();
65             }
66         }
67     }
}
```

```
public void updateChange(Change change) {  
    Connection con = null;  
    try {  
        con = getConnection();  
        String sql = "UPDATE changes SET name = ? WHERE changeID = ?";  
        PreparedStatement psmt = con.prepareStatement(sql);  
        psmt.setString(parameterIndex: 1, change.getName());  
        psmt.setInt(parameterIndex: 2, change.getChangeID());  
        int rowsAffected = psmt.executeUpdate();  
        if (rowsAffected > 0) {  
            System.out.println("updateChange成功");  
        } else {  
            System.out.println("updateChange失败");  
        }  
        psmt.close();  
    } catch (SQLException e) {  
        e.printStackTrace();  
    } finally {  
        try {  
            if (con != null) {  
                con.close();  
            }  
        } catch (SQLException e) {  
            e.printStackTrace();  
        }  
    }  
}
```

(四) 园林植物养护管理

1. 数据表 DDL 语句

1.1 任务表

```
CREATE TABLE C_Task (  
    taskID INT PRIMARY KEY,  
    taskName VARCHAR(255),  
    taskDes VARCHAR(255)  
);
```

1.2 养护表

```
CREATE TABLE C_Maintenance (  
    mainID INT PRIMARY KEY,  
    mainTime DATE,  
    mainLoc VARCHAR(255),  
    mainPer VARCHAR(255),  
    plantId CHAR(4) NOT NULL,
```

数据

库系统课程设计报告

```
status VARCHAR(255) CHECK (status IN ('已完成', '进行中', '未完成')),  
FOREIGN KEY (plantId) REFERENCES A_Plant (id)  
);
```

1.3 任务养护表

```
CREATE TABLE C_TaskMaintenance (  
mainID INT,  
taskID INT,  
PRIMARY KEY (mainID, taskID),  
FOREIGN KEY (mainID) REFERENCES C_Maintenance (mainID),  
FOREIGN KEY (taskID) REFERENCES C_Task (taskID)  
);
```

2.索引 DDL 语句

2.1 任务编号索引

```
CREATE INDEX idx_taskID ON C_Task (taskID);
```

2.2 任务名称索引

```
CREATE INDEX idx_taskName ON C_Task (taskName);
```

2.3 养护编号索引

```
CREATE INDEX idx_maintenanceID ON C_Maintenance (mainID);
```

2.4 养护名称索引

```
CREATE INDEX idx_maintenancePer ON C_Maintenance (mainPer);
```

2.5 养护状态索引

```
CREATE INDEX idx_maintenancesstatus ON C_Maintenance (status);
```

3.视图 DDL 语句

3.1 人物任务视图：使用于工具间保证人物可以对应任务找到对应工具

```
CREATE VIEW C_PersonTask AS
```

```
SELECT
```

```
m.mainPer,  
t.taskName
```

```
FROM C_Task t
```

```
LEFT JOIN C_TaskMaintenance tm ON t.taskID = tm.taskID
```

```
LEFT JOIN C_Maintenance m ON tm.mainID = m.mainID;
```

数据

库系统课程设计报告

3.2 时间地点人物视图：使用于办公室保证人物可以对应时间找到对应地点

```
CREATE VIEW C_TimePersonLocation AS
```

```
SELECT
```

```
    m.mainTime,
```

```
    m.mainPer,
```

```
    m.mainLoc
```

```
FROM C_Maintenance m;
```

3.3 人物植物任务视图：使用于园林地点保证人物可以对应任务找到对应植物

```
CREATE VIEW C_PersonTaskPlant AS
```

```
SELECT
```

```
    m.mainPer,
```

```
    t.taskName,
```

```
    p.altername
```

```
FROM C_Task t
```

```
LEFT JOIN C_TaskMaintenance tm ON t.taskID = tm.taskID
```

```
LEFT JOIN C_Maintenance m ON tm.mainID = m.mainID
```

```
LEFT JOIN A_Plant p ON m.plantId = p.id;
```

4.DAO 基础实现代码

Entity:

数据

库系统课程设计报告

```
1 package project.entity; ▲ 12 ▲ 5 ✅ 12 ⌂ ⌃
2
3 20 个用法
4 public class C_Maintenance {
5     5 个用法
6         private int mainID;
7             5 个用法
8                 private String mainTime;
9                     5 个用法
10                    private String mainLoc;
11                        6 个用法
12                            private String mainPer;
13                                4 个用法
14                                    private String plantId;
15                                        5 个用法
16                                            private String status;
17
18    1 个用法
19        public C_Maintenance() {
20            this.mainID = mainID;
21            this.mainTime = mainTime;
22            this.mainLoc = mainLoc;
23            this.mainPer = mainPer;
24            this.plantId = plantId;
25            this.status = status;
26        }
27
28    3 个用法
29        public int getmainID() { return mainID; }
30
31    1 个用法
32        public void setmainID(int mainNum) { this.mainID = mainNum; }
```

数据

库系统课程设计报告

```
2 个用法  
28     public String getmainTime() { return mainTime; }  
31  
1 个用法  
32     public void setmainTime(String mainTime) { this.mainTime = mainTime; }  
35  
2 个用法  
36     public String getmainLoc() { return mainLoc; }  
39  
1 个用法  
40     public void setmainLoc(String mainLoc) { this.mainLoc = mainLoc; }  
43  
2 个用法  
44     public String getmainPer() { return mainPer; }  
47  
1 个用法  
48     public void setmainPer(String mainPer) { this.mainPer = mainPer; }  
51  
2 个用法  
52     public String getplantId() { return plantId; }  
55  
1 个用法  
56     public void setplantId(String plantId) { this.plantId = plantId; }  
59  
2 个用法  
60     public String getStatus() {  
61         return status;  
62     }  
63  
1 个用法  
64     public void setStatus(String status) { this.status = status; }  
67  
68     @Override  
69     public String toString() {  
70         return "Main{" +  
71             "mainTime=" + mainTime + ", "  
72             "mainLoc=" + mainLoc + ", "  
73             "mainPer=" + mainPer + ", "  
74             "plantId=" + plantId + ", "  
75             "status=" + status + '}';  
76     }  
77  
78 }
```

数据

库系统课程设计报告

```
68     @Override
69     public String toString() {
70         return "C_Maintenance{" +
71             "mainNum=" + mainID +
72             ", mainTime='" + mainTime + '\'' +
73             ", mainLoc='" + mainLoc + '\'' +
74             ", mainPer='" + mainPer + '\'' +
75             ", plantId='" + plantId + '\'' +
76             ", status='" + status + '\'' +
77             '}';
78     }
79 }
80 }
```

数据

库系统课程设计报告

```
1 package project.entity; ▲ 6 ▲ 2 ✘ 6 ^ v
2
3     20 个用法
4
5     public class C_Task {
6         5 个用法
7             private int taskID;
8             5 个用法
9                 private String taskName;
10                5 个用法
11                    private String taskDes;
12
13                1 个用法
14                    public C_Task() {
15                        this.taskID = taskID;
16                        this.taskName = taskName;
17                        this.taskDes = taskDes;
18
19                    }
20
21                3 个用法
22                    public int gettaskID() { return taskID; }
23
24                1 个用法
25                    public void settaskID(int taskID) { this.taskID = taskID; }
26
27                2 个用法
28                    public String gettaskName() { return taskName; }
29
30                1 个用法
31                    public void settaskName(String taskName) { this.taskName = taskName; }
32
33                2 个用法
34                    public String gettaskDes() { return taskDes; }
35
36
37
38
39
39
```

数据

库系统课程设计报告

```
1 1个用法
34     public void settaskDes(String taskDes) { this.taskDes = taskDes; }
37
38     @Override
39     public String toString() {
40         return "C_Task{" +
41             "taskID=" + taskID +
42             ", taskName='" + taskName + '\'' +
43             ", taskDes='" + taskDes + '\'' +
44             '}';
45     }
46 }
47 |
```

Dao:

```
1 package project.dao.C_Maintenance;
2
3 import project.entity.C_Maintenance;
4
5 import java.util.List;
6
7 4个用法 1个实现
8 ①↓ public interface C_Maintenance_Dao {
9     1个实现
10    void addInfo(C_Maintenance maintenance);
11    1个实现
12    void deleteInfo(int mainID);
13    1个实现
14    void updateInfo(C_Maintenance maintenance);
15    1个实现
16    List<C_Maintenance> findInfo(int mainID);
17 }
```

数据

库系统课程设计报告

```
1 package project.dao.C_Task;
2 import project.entity.C_Task;
3
4 import java.util.List;
5
6 4个用法 1个实现
7 ①↓ public interface C_Task_Dao {
8     1个实现
9     void addInfo(C_Task task);
10    1个实现
11    void deleteInfo(int taskID);
12    1个实现
13    void updateInfo(C_Task task);
14    1个实现
15    List<C_Task> findInfo(int taskID);
16 }
```

Service:

```
1 package project.service.C_Maintenance;
2 import project.entity.C_Maintenance;
3
4 1个用法 1个实现
5 ①↓ public interface C_Maintenance_Service {
6     1个实现
7     void addInfo(C_Maintenance maintenance);
8     1个实现
9     void deleteInfo(int mainID);
10    1个实现
11    void updateInfo(C_Maintenance maintenance);
12    1个实现
13    C_Maintenance findInfo(int mainID);
14 }
```

```
1 package project.service.C_Task;
2
3 import project.entity.C_Task;
4
5 1个用法 1个实现
6 ①↓ public interface C_Task_Service {
7     1个实现
8     void addInfo(C_Task task);
9     1个实现
10    void deleteInfo(int taskID);
11    1个实现
12    void updateInfo(C_Task task);
13    1个实现
14    C_Task findInfo(int taskID);
15 }
```

Impl:

```
1 package project.dao.C_Maintenance;
2 import project.entity.C_Maintenance;
3
4 import java.sql.Connection;
5 import java.sql.PreparedStatement;
6 import java.sql.ResultSet;
7 import java.sql.SQLException;
8 import java.util.ArrayList;
9 import java.util.List;
10
11 0个用法
12
13 public class C_Maintenance_Impl implements C_Maintenance_Dao {
14
15     1个用法
16     private C_Maintenance maintenance;
17
18     @Override
19     public void addInfo(C_Maintenance maintenance){
20         String sql = "INSERT INTO maintenance VALUES(?,?,?,?,?,?)";
21         Connection connection = maintenance.getConnection();
22         PreparedStatement preparedStatement = null;
23         try {
24             preparedStatement = connection.prepareStatement(sql);
25             preparedStatement.setInt( parameterIndex: 1, maintenance.getmainID());
26             preparedStatement.setString( parameterIndex: 2, maintenance.getmainTime());
27             preparedStatement.setString( parameterIndex: 3, maintenance.getmainLoc());
28             preparedStatement.setString( parameterIndex: 4, maintenance.getmainPer());
29             preparedStatement.setString( parameterIndex: 5, maintenance.getplantId());
30             preparedStatement.setString( parameterIndex: 6, maintenance.getStatus());
31             preparedStatement.executeUpdate();
32             preparedStatement.close();
33         } catch (SQLException e) {
34             throw new RuntimeException(e);
35         }
36     }
37 }
```

数据

库系统课程设计报告

```
30             } catch (SQLException e) {
31                 throw new RuntimeException(e);
32             }
33         }
34     }
35
36     @Override
37     public void deleteInfo(int mainID){
38         String SQL = "delete from maintenance where mainID= ?";
39         Connection connection = getConnection();
40         PreparedStatement preparedStatement = null;
41         try {
42             preparedStatement = connection.prepareStatement(SQL);
43             preparedStatement.setInt( parameterIndex: 1, maintenance.getmainID());
44             preparedStatement.executeUpdate();
45             preparedStatement.close();
46             connection.close();
47         } catch (SQLException e) {
48             throw new RuntimeException(e);
49         }
50     }
51
52     @Override
53     public void updateInfo(C_Maintenance maintenance){
54         String sql = "UPDATE maintenance SET mainID=?,mainTime=?,mainLoc=?,mainPer=?,plantId=?,status=?";
55         Connection connection = getConnection();
56         PreparedStatement preparedStatement = null;
57         try {
58             preparedStatement = connection.prepareStatement(sql);
59             preparedStatement.setInt( parameterIndex: 1, maintenance.getmainID());
60             preparedStatement.setString( parameterIndex: 2, maintenance.getmainTime());
61             preparedStatement.setString( parameterIndex: 3, maintenance.getmainLoc());
62             preparedStatement.setString( parameterIndex: 4, maintenance.getmainPer());
63             preparedStatement.setString( parameterIndex: 5, maintenance.getplantId());
```

数据

库系统课程设计报告

```
63     preparedStatement.setString( parameterIndex: 6, maintenance.getStatus());
64     preparedStatement.executeUpdate();
65     preparedStatement.close();
66 } catch (SQLException e) {
67     throw new RuntimeException(e);
68 }
69 }
70
71 @Override
72 public List<C_Maintenance> findInfo(int mainID) {
73     ArrayList<C_Maintenance> arrayList = new ArrayList<>();
74     String SQL = "";
75     try {
76         Connection connection = getConnection();
77         PreparedStatement preparedStatement = null;
78         SQL = "select * from C_Maintenance where mainID=?";
79         preparedStatement = connection.prepareStatement(SQL);
80         preparedStatement.setInt( parameterIndex: 1,mainID);
81
82         ResultSet resultSet = preparedStatement.executeQuery();
83         while(resultSet.next()){
84             C_Maintenance C_Maintenance = new C_Maintenance();
85             C_Maintenance.setmainID(resultSet.getInt( columnLabel: "mainID"));
86             C_Maintenance.setmainTime(resultSet.getString( columnLabel: "mainTime"));
87             C_Maintenance.setmainLoc(resultSet.getString( columnLabel: "mainLoc"));
88             C_Maintenance.setmainPer(resultSet.getString( columnLabel: "mainPer"));
89             C_Maintenance.setplantId(resultSet.getString( columnLabel: "plantId"));
90             C_Maintenance.setStatus(resultSet.getString( columnLabel: "status"));
91             arrayList.add(C_Maintenance);
92         }
93         preparedStatement.close();
94         connection.close();
95     } catch (SQLException e) {
96         throw new RuntimeException(e);
97     }
98     return arrayList;
99 }
100 }
```

数据

库系统课程设计报告

```
1 package project.dao.C_Task ;
2 import project.entity.C_Task ;
3
4 import java.sql.Connection;
5 import java.sql.PreparedStatement;
6 import java.sql.ResultSet;
7 import java.sql.SQLException;
8 import java.util.ArrayList;
9 import java.util.List;
10
11 0个用法
12
13 public class C_Task_Impl implements C_Task_Dao {
14
15     1个用法
16     private C_Task task ;
17
18     @Override
19     public void addInfo(C_Task task ){
20         String sql = "INSERT INTO task VALUES(?,?)";
21         Connection connection = task.getConnection();
22         PreparedStatement preparedStatement = null;
23         try {
24             preparedStatement = connection.prepareStatement(sql);
25             preparedStatement.setInt( parameterIndex: 1, task.gettaskID());
26             preparedStatement.setString( parameterIndex: 2, task.gettaskName());
27             preparedStatement.setString( parameterIndex: 3, task.gettaskDes());
28             preparedStatement.executeUpdate();
29             preparedStatement.close();
30         } catch (SQLException e) {
31             throw new RuntimeException(e);
32         }
33     }
34 }
```

数据

库系统课程设计报告

```
32     @Override
33     public void deleteInfo(int taskID){
34         String SQL = "delete from task where taskID= ?";
35         Connection connection = getConnection();
36         PreparedStatement preparedStatement = null;
37         try {
38             preparedStatement = connection.prepareStatement(SQL);
39             preparedStatement.setInt( parameterIndex: 1, task .gettaskID());
40             preparedStatement.executeUpdate();
41             preparedStatement.close();
42             connection.close();
43         } catch (SQLException e) {
44             throw new RuntimeException(e);
45         }
46     }
47
48     @Override
49     public void updateInfo(C_Task  task){
50         String sql = "UPDATE task SET taskID=? ,taskName=?";
51         Connection connection = getConnection();
52         PreparedStatement preparedStatement = null;
53         try {
54             preparedStatement = connection.prepareStatement(sql);
55             preparedStatement.setInt( parameterIndex: 1, task.gettaskID());
56             preparedStatement.setString( parameterIndex: 2, task.gettaskName());
57             preparedStatement.setString( parameterIndex: 3, task.gettaskDes());
58             preparedStatement.executeUpdate();
59             preparedStatement.close();
60         } catch (SQLException e) {
61             throw new RuntimeException(e);
62         }
63     }
```

数据

库系统课程设计报告

```
65     @Override
66     public List<C_Task> findInfo(int taskID) {
67         ArrayList<C_Task> arrayList = new ArrayList<>();
68         String SQL="";
69         try {
70             Connection connection = getConnection();
71             PreparedStatement preparedStatement = null;
72             SQL = "select * from C_Task where taskID=?";
73             preparedStatement = connection.prepareStatement(SQL);
74             preparedStatement.setInt(parameterIndex: 1,taskID);
75
76             ResultSet resultSet = preparedStatement.executeQuery();
77             while(resultSet.next()){
78                 C_Task C_Task = new C_Task();
79                 C_Task.settaskID(resultSet.getInt(columnLabel: "taskID"));
80                 C_Task.settaskName(resultSet.getString(columnLabel: "taskName"));
81                 C_Task.settaskDes(resultSet.getString(columnLabel: "taskDes"));
82                 arrayList.add(C_Task );
83             }
84             preparedStatement.close();
85             connection.close();
86         } catch (SQLException e) {
87             throw new RuntimeException(e);
88         }
89         return arrayList;
90     }
91 }
```

(五)园林植物病虫害防治管理

1.数据表 DDL 语句

1.1 药剂表

```
CREATE TABLE D_Medicine (
    mediID INT PRIMARY KEY,
    mediName VARCHAR(255),
    mediDos INT,
    dataAct DATE
);
```

1.2 病虫害表

```
CREATE TABLE D_Pest (
    pestID INT PRIMARY KEY,
    pestName VARCHAR(255)
);
```

1.3 防治表

数据

库系统课程设计报告

```
CREATE TABLE D_Control (
    conID INT PRIMARY KEY,
    conTime DATE,
    conLoc VARCHAR(255),
    conPer VARCHAR(255),
    status VARCHAR(255) CHECK (status IN ('已完成', '进行中', '未完成')),
    id char(4) NOT NULL,
    FOREIGN KEY (id) REFERENCES A_Plant (id)
);
```

1.4 药剂虫害防治表

```
CREATE TABLE D_MediPestCon (
    conID INT,
    pestID INT,
    mediID INT,
    FOREIGN KEY (conID) REFERENCES D_Control(conID),
    FOREIGN KEY (pestID) REFERENCES D_Pest(pestID),
    FOREIGN KEY (mediID) REFERENCES D_Medicine(mediID),
    PRIMARY KEY (conID, pestID, mediID)
);
```

2. 索引 DDL 语句

2.1 药剂编号索引

```
CREATE INDEX idx_medicineID ON D_Medicine (mediID);
```

2.2 植物 id, 防治时间排序索引

```
CREATE INDEX idx_plantId_conTime ON D_Control(id, conTime);
```

2.3 防治编号索引

```
CREATE INDEX idx_controlID ON D_Control (conID);
```

2.4 药剂名称索引

```
CREATE INDEX idx_mediName ON D_Medicine (mediName);
```

2.5 病虫索引

```
CREATE INDEX idx_pestName ON D_Pest (pestName);
```

3. 视图 DDL 语句

数据

库系统课程设计报告

3.1 查询某一地点聚集病虫视图

```
CREATE VIEW D_LocationPestInfo AS
SELECT
    D_Control.conLoc,
    D_Pest.pestName
FROM
    D_Control
JOIN
    D_MediPestCon ON D_Control.conID = D_MediPestCon.conID
JOIN
    D_Pest ON D_MediPestCon.pestID = D_Pest.pestID
WHERE
    D_Control.conLoc IS NOT NULL
ORDER BY conLoc
OFFSET 0 ROWS
FETCH NEXT 100 ROWS ONLY;
```

3.2 系统人员查询某一防治人员近期使用药物信息

```
CREATE VIEW D_MedicationUsage AS
SELECT
    D_Control.conPer,
    D_Medicine.mediName
FROM
    D_Control
JOIN
    D_MediPestCon ON D_Control.conID = D_MediPestCon.conID
JOIN
    D_Medicine ON D_MediPestCon.mediID = D_Medicine.mediID
ORDER BY conPer
OFFSET 0 ROWS
FETCH NEXT 100 ROWS ONLY;
```

3.3 药剂使用详细视图

```
CREATE VIEW D_AllMediPestCon AS
SELECT
    D_MediPestCon.conID,
```

数据

库系统课程设计报告

```
D_MediPestCon.pestID,  
D_MediPestCon.mediID,  
D_Control.conTime,  
D_Control.conLoc,  
D_Control.conPer,  
D_Medicine.mediName,  
D_Pest.pestName  
  
FROM  
    D_MediPestCon  
  
JOIN  
    D_Control ON D_MediPestCon.conID = D_Control.conID  
  
JOIN  
    D_Medicine ON D_MediPestCon.mediID = D_Medicine.mediID  
  
JOIN  
    D_Pest ON D_MediPestCon.pestID = D_Pest.pestID;
```

3.4 具体植物就病信息视图

```
CREATE VIEW PlantControlView AS  
SELECT A_Plant.id AS PlantID, A_Plant.altername, D_Control.conID, D_Control.conTime,  
D_Control.conLoc, D_Control.conPer, D_Control.status  
FROM A_Plant  
LEFT JOIN D_Control ON A_Plant.id = D_Control.id;
```

5. DAO 基础实现代码

实体类：

数据

库系统课程设计报告

```
package project.entity; ▲ 12

16 个用法
public class D_Control {
    3 个用法
    private int conID;
    3 个用法
    private String conTime;
    3 个用法
    private String conLoc;
    3 个用法
    private String conPer;
    3 个用法
    private String status;
    3 个用法
    private String id;
    // 构造函数
    0 个用法
    public D_Control(int conID, String conTime, String conLoc, String conPer, String status, String id) {
        this.conID = conID;
        this.conTime = conTime;
        this.conLoc = conLoc;
        this.conPer = conPer;
        this.status = status;
        this.id = id;
    }
    // 获取conID属性的getter方法
```

数据

库系统课程设计报告

```
0个用法
public int getConID() { return conID; }

// 设置conID属性的setter方法
0个用法
public void setConID(int conID) { this.conID = conID; }

// 获取conTime属性的getter方法
0个用法
public String getConTime() { return conTime; }

// 设置conTime属性的setter方法
0个用法
public void setConTime(String conTime) { this.conTime = conTime; }

// 获取conLoc属性的getter方法
0个用法
public String getConLoc() { return conLoc; }

// 设置conLoc属性的setter方法
0个用法
public void setConLoc(String conLoc) { this.conLoc = conLoc; }

// 获取conPer属性的getter方法
0个用法
public String getConPer() { return conPer; }
```

```
package project.entity;

import java.sql.Date;

16个用法
public class D_Medicine {
    4个用法
    private int mediID;
    4个用法
    private String mediName;
    4个用法
    private int mediDosage;
    4个用法
    private String dataAct;

    0个用法
    public D_Medicine(){

    }

    // 构造函数
    0个用法
    public D_Medicine(int mediID, String mediName, int mediDosage, Date dataAct) {
        this.mediID = mediID;
        this.mediName = mediName;
        this.mediDosage = mediDosage;
        this.dataAct = String.valueOf(dataAct);
    }
}
```

数据

库系统课程设计报告

```
package project.entity;

1 个用法
public class D_Pest {
    3 个用法
    private int pestID;
    3 个用法
    private String pestName;

    // 构造函数
    0 个用法
    public D_Pest(int pestID, String pestName) {
        this.pestID = pestID;
        this.pestName = pestName;
    }

    // 获取pestID属性的getter方法
    0 个用法
    public int getPestID() { return pestID; }

    // 设置pestID属性的setter方法
    0 个用法
    public void setPestID(int pestID) { this.pestID = pestID; }

    // 获取pestName属性的getter方法
    0 个用法
    public String getPestName() { return pestName; }
}
```

dao 接口

```
package project.dao.D_Control;
import project.entity.D_Control;
4 个用法 1 个实现
public interface D_ControlDao {
    1 个实现
    void addInfo(D_Control control);
    1 个实现
    void deleteInfo(int conID);
    1 个实现
    void updateInfo(D_Control control);
    1 个实现
    D_Control findInfo(int conID);
}

package project.dao.D_Medicine;
import project.entity.D_Medicine;

4 个用法 1 个实现
public interface D_MedicineDao {
    1 个实现
    void addInfo(D_Medicine medicine);
    1 个实现
    void deleteInfo(int mediID);
    1 个实现
    void updateInfo(D_Medicine medicine);
    1 个实现
    D_Medicine findInfo(int mediID);
}
```

数据

库系统课程设计报告

```
package project.dao.D_Pest;

import project.entity.D_Pest;

4个用法 1个实现
public interface D_PestDao {
    1个实现
    void addInfo(D_Pest pest);
    1个实现
    void deleteInfo(int pestID);
    1个实现
    void updateInfo(D_Pest pest);
    1个实现
    D_Pest findInfo(int pestID);
}
```

dao 实现类:

数据

库系统课程设计报告

```
package project.dao.D_Control;
import project.entity.D_Control;

0个用法
public class D_ControlDaoImpl implements project.dao.D_Control.D_ControlDao{
    public void addInfo(D_Control control) {
        // 实现添加操作信息的逻辑
        System.out.println("添加操作信息: " + control);
    }
    public void deleteInfo(int conlID) {
        // 实现删除操作信息的逻辑
        System.out.println("删除操作信息编号为: " + conlID);
    }
    public void updateInfo(D_Control control) {
        // 实现更新操作信息的逻辑
        System.out.println("更新操作信息: " + control);
    }
    public D_Control findInfo(int conlID) {
        // 实现查找操作信息的逻辑
        System.out.println("查找操作信息编号为: " + conlID);
        return null; // 根据实际情况返回查找到的操作信息
    }
}

package project.dao.D_Medicine;
import project.entity.D_Medicine;

0个用法
public class D_MedicineDaoImpl implements project.dao.D_Medicine.D_MedicineDao{
    public void addInfo(D_Medicine medicine) {
        // 实现添加药物信息的逻辑
        System.out.println("添加药物信息: " + medicine);
    }
    public void deleteInfo(int mediID) {
        // 实现删除药物信息的逻辑
        System.out.println("删除药物信息编号为: " + mediID);
    }
    public void updateInfo(D_Medicine medicine) {
        // 实现更新药物信息的逻辑
        System.out.println("更新病虫信息: " + medicine);
    }
    public D_Medicine findInfo(int mediID) {
        // 实现查找药物信息的逻辑
        System.out.println("查找药物信息编号为: " + mediID);
        return null; // 根据实际情况返回查找到的药物信息
    }
}
```

数据

库系统课程设计报告

```
package project.dao.D_Pest;
import project.entity.D_Pest;
0个用法
public class D_PestDaoImpl implements project.dao.D_Pest.D_PestDao {

    public void addInfo(D_Pest pest) {
        // 实现添加病虫信息的逻辑
        System.out.println("添加病虫信息: " + pest);
    }
    public void deleteInfo(int pestID) {
        // 实现删除病虫信息的逻辑
        System.out.println("删除病虫信息编号为: " + pestID);
    }
    public void updateInfo(D_Pest pest) {
        // 实现更新病虫信息的逻辑
        System.out.println("更新病虫信息: " + pest);
    }
    public D_Pest findInfo(int pestID) {
        // 实现查找病虫信息的逻辑
        System.out.println("查找病虫信息编号为: " + pestID);
        return null; // 根据实际情况返回查找到的病虫信息
    }
}
```

service 接口:

```
package project.service.D_Control;

import project.entity.D_Control;

1个用法 1个实现
public interface D_Control_Service {
    1个实现
    void addInfo(D_Control control);
    1个实现
    void deleteInfo(int conID);
    1个实现
    void updateInfo(D_Control control);
    1个实现
    D_Control findInfo(int conID);
}
```

```
package project.service.D_Medicine;
import project.entity.D_Medicine;

1个用法 1个实现
public interface D_Medicine_Service {
    1个实现
    void addInfo(D_Medicine medicine);
    1个实现
    void deleteInfo(int mediID);
    1个实现
    void updateInfo(D_Medicine medicine);
    1个实现
    D_Medicine findInfo(int mediID);
}
```

数据

库系统课程设计报告

```
package project.service.D_Pest;
import project.entity.D_Pest;

1个用法 1个实现
public interface D_Pest_Service {
    1个实现
    void addInfo(D_Pest pest);
    1个实现
    void deleteInfo(int pestID);
    1个实现
    void updateInfo(D_Pest pest);
    1个实现
    D_Pest findInfo(int pestID);
}
```

(六)园林植物监测管理

1. 数据表 DDL 语句

1.1 设备表

```
CREATE TABLE E_Equipment (
    equipNum INT PRIMARY KEY,
    equipName VARCHAR(255)
);
```

1.2 指标表

```
CREATE TABLE E_Indicator(
    indexNum INT PRIMARY KEY,
    indexName VARCHAR(255)
);
```

1.3 监测表

```
CREATE TABLE E_Detection(
    detecNum INT PRIMARY KEY,
    detecTime DATE,
    detecLoc VARCHAR(255),
    detecPer VARCHAR(255),
    status VARCHAR(255) CHECK (status IN ('已完成', '进行中', '未完成')),
    plantId CHAR(4) NOT NULL,
    FOREIGN KEY (plantId) REFERENCES A_Plant (id)
);
```

1.4 设备指标监测表

```
CREATE TABLE E_EquipDetecIndica(
    equipNum INT,
```

数据

库系统课程设计报告

```
indexNum INT,  
detecNum INT,  
PRIMARY KEY (equipNum, indexNum, detecNum),  
FOREIGN KEY (equipNum) REFERENCES E_Equipment (equipNum),  
FOREIGN KEY (indexNum) REFERENCES E_Indicator (indexNum),  
FOREIGN KEY (detecNum) REFERENCES E_Detection (detecNum)  
);
```

2.索引 DDL 语句

2.1 监测编号索引

```
CREATE INDEX idx_detectionNum ON E_Detection (detecNum);
```

2.2 监测人物索引

```
CREATE INDEX idx_detectionPer ON E_Detection (detecPer);
```

2.3 监测状态索引

```
CREATE INDEX idx_detectionstatus ON E_Detection (status);
```

2.4 设备编号索引

```
CREATE INDEX idx_equimentNum ON E_Equipment (equipNum);
```

2.5 指标编号索引

```
CREATE INDEX idx_indicatorNum ON E_Indicator (indexNum);
```

3.视图 DDL 语句

3.1 时间地点人物视图：使用于办公室保证人物可以对应时间找到对应地点

```
CREATE VIEW E_TimePersonLocation AS
```

```
SELECT
```

```
    d.detecTime,  
    d.detecPer,  
    d.detecLoc
```

```
FROM E_Detection d;
```

3.2 植物设备指标视图：使用于园林地点保证植物可以对应设备找到对应指标

```
CREATE VIEW E_PlantEquipmentIndicator AS
```

```
SELECT
```

```
    p.alterName,  
    e.equipName,  
    i.indexName
```

数据

库系统课程设计报告

```
FROM E_EquipDetecIndica edi  
JOIN E_Equipment e ON edi.equipNum = e.equipNum  
JOIN E_Indicator i ON edi.indexNum = i.indexNum  
JOIN E_Detection d ON edi.detecNum = d.detecNum  
JOIN A_Plant p ON d.plantId = p.id;
```

3.3 人物设备指标视图：使用于工作间保证人物可以对应设备找到对应指标

```
CREATE VIEW E_PersonEquipmentIndicator AS
```

```
SELECT
```

```
    d.detecPer,  
    e.equipName,  
    i.indexName  
FROM E_EquipDetecIndica edi  
JOIN E_Equipment e ON edi.equipNum = e.equipNum  
JOIN E_Indicator i ON edi.indexNum = i.indexNum  
JOIN E_Detection d ON edi.detecNum = d.detecNum;
```

数据

库系统课程设计报告

4.DAO 基础实现代码

```
1 package project.entity; 28 ^ v
2
3 2个用法
4 public class E_Detection {
5     4个用法
6         private int detecNum;
7         4个用法
8         private String detecTime;
9         4个用法
10        private String detecLoc;
11        5个用法
12        private String detecPer;
13        3个用法
14        private String plantId;
15        4个用法
16        private String status;
17
18    1个用法
19    public E_Detection(int detecNum, String detecTime, String detecLoc, String de
20        this.detecNum = detecNum;
21        this.detecTime = detecTime;
22        this.detecLoc = detecLoc;
23        this.detecPer = detecPer;
24        this.plantId = plantId;
25        this.status = status;
26    }
27
28    3个用法
29    public int getdetecNum() { return detecNum; }
30
31    1个用法
32    public void setdetecNum(int detecNum) { this.detecNum = detecNum; }
```

数据

库系统课程设计报告

```
2 个用法  
28     public String getdetectTime() { return detectTime; }  
31  
1 个用法  
32     public void setdetectTime(String detectTime) { this.detectTime = detectTime; }  
35  
2 个用法  
36     public String getdetectLoc() { return detectLoc; }  
39  
1 个用法  
40     public void setdetectLoc(String detectLoc) { this.detectLoc = detectLoc; }  
43  
2 个用法  
44     public String getdetectPer() { return detectPer; }  
47  
1 个用法  
48     public void setdetectPer(String detectPer) { this.detectPer = detectPer; }  
51  
2 个用法  
52     public String getplantId() { return plantId; }  
55  
1 个用法  
56     public void setplantId(String plantId) { this.plantId = plantId; }  
59  
2 个用法  
60     public String getStatus() { return status; }  
63  
1 个用法  
64     public void setStatus(String status) { this.status = status; }  
67  
68     @Override  
69     public String toString() {  
70         return "E_Detection{" +
```

数据

库系统课程设计报告

```
68     @Override
69     public String toString() {
70         return "E_Detection{" +
71                 "detecNum=" + detecNum +
72                 ", detecTime='" + detectTime + '\'' +
73                 ", detecLoc='" + detectLoc + '\'' +
74                 ", detecPer='" + detectPer + '\'' +
75                 ", plantId='" + plantId + '\'' +
76                 ", status='" + status + '\'' +
77                 '}';
78     }
79 }
80
```

数据

库系统课程设计报告

```
1 package project.entity; ▲ 4 ▲ 1 ✅ 4 ⌂ ⌄
2
3     20个用法
4
5     public class E_Indicator {
6         5个用法
7         private int indexNum;
8         5个用法
9         private String indexName;
10
11         1个用法
12         public E_Indicator() {
13             this.indexNum = indexNum;
14             this.indexName = indexName;
15         }
16
17         3个用法
18         public int getIndexNum() { return indexNum; }
19
20         1个用法
21         public void setIndexNum(int indexNum) {
22             this.indexNum = indexNum;
23         }
24
25         2个用法
26         public String getIndexName() { return indexName; }
27
28         1个用法
29         public void setIndexName(String indexName) {
30             this.indexName = indexName;
31         }
32
33         @Override
34         public String toString() {
35             @Override
36             public String toString() {
37                 return "E_Indicator{" +
38                     "indexNum=" + indexNum +
39                     ", indexName='" + indexName + '\'' +
40                     '}';
41         }
42     }
43 }
```

数据

库系统课程设计报告

```
1 package project.entity; ⚠ 4 ⚠ 1 ✅ 4 ⌂ ⌃
2
3     20个用法
4
5         public class E_Equipment {
6             5个用法
7                 private int equipNum;
8                 5个用法
9                 private String equipName;
10
11                 1个用法
12             public E_Equipment() {
13                 this.equipNum = equipNum;
14                 this.equipName = equipName;
15             }
16
17                 3个用法
18             public int getequipNum() { return equipNum; }
19
20                 1个用法
21             public void setequipNum(int equipNum) { this.equipNum = equipNum; }
22
23                 2个用法
24             public String getequipName() { return equipName; }
25
26                 1个用法
27             public void setequipName(String equipName) { this.equipName = equipName; }
28
29             @Override
30             public String toString() {
31                 return "E_Equipment{" +
32                     "equipNum=" + equipNum +
33                     ", equipName='" + equipName + '\'' +
34                     '}';
35
36         }
```

数据

库系统课程设计报告

Dao:

```
1 package project.dao.E_Detection;           ✘ 2 ⌂ ⌄
2 import project.entity.E_Detection;
4 个用法 1个实现
3 ⌄ public interface E_Detection.Dao {
    1个实现
4 ⌄     void addInfo(E_Detection Detection);
    1个实现
5 ⌄     void deleteInfo(int detecNum);
    1个实现
6 ⌄     void updateInfo(E_Detection Detection);
    1个实现
7 ⌄     E_Detection findInfo(int detecNum);
8 }
```

```
1 package project.dao.E_Equipment;           ✓
2 import project.entity.E_Equipment;
3
4 import java.util.List;
5
4 个用法 1个实现
6 ⌄ public interface E_Equipment.Dao {
    1个实现
7 ⌄     void addInfo(E_Equipment Equipment);
    1个实现
8 ⌄     void deleteInfo(int equipNum);
    1个实现
9 ⌄     void updateInfo(E_Equipment Equipment);
    1个实现
10 ⌄    List<E_Equipment> findInfo(int equipNum);
11 }
```

```
1 package project.dao.E_Indicator;           ✓
2 import project.entity.E_Indicator;
3
4 import java.util.List;
5
4 个用法 1个实现
6 ⌄ public interface E_Indicator.Dao {
    1个实现
7 ⌄     void addInfo(E_Indicator Indicator);
    1个实现
8 ⌄     void deleteInfo(int indexNum);
    1个实现
9 ⌄     void updateInfo(E_Indicator Indicator);
    1个实现
10 ⌄    List<E_Indicator> findInfo(int indexNum);
11 }
```

Service:

```
1 package project.service.E_Detection;           ✓ 2 ⌂ ⌂
2 import project.entity.E_Detection;
3
4 ①↓ 1个用法 1个实现
5 ①↓ public interface E_Detection_Service {
6    1个实现
7    ①↓ void addInfo(E_Detection detection);
8    1个实现
9    ①↓ void deleteInfo(int detecNum);
10   1个实现
11   ①↓ void updateInfo(E_Detection detection);
12   1个实现
13   ①↓ E_Detection findInfo(int detecNum);
14 }
```



```
1 package project.service.E_Equipment;           ✓
2
3 import project.entity.E_Equipment;
4
5 ①↓ 1个用法 1个实现
6 ①↓ public interface E_Equipment_Service {
7    1个实现
8    ①↓ void addInfo(E_Equipment equipment);
9    1个实现
10   ①↓ void deleteInfo(int equipNum);
11   1个实现
12   ①↓ void updateInfo(E_Equipment equipment);
13   1个实现
14   ①↓ E_Equipment findInfo(int equipNum);
15 }
```

数据

库系统课程设计报告

```
1 package project.service.E_Indicator;
2
3 import project.entity.E_Indicator;
4
5 1个用法 1个实现
6 ①↓ public interface E_Indicator_Service {
7     1个实现
8     void addInfo(E_Indicator indicator);
9     1个实现
10    void deleteInfo(int indexNum);
11    1个实现
12    void updateInfo(E_Indicator indicator);
13    1个实现
14    E_Indicator findInfo(int indexNum);
15 }
```

Impl:

数据

库系统课程设计报告

```
1 package project.dao.E_Detection;
2 import project.entity.E_Detection;
3
4 import java.sql.Connection;
5 import java.sql.PreparedStatement;
6 import java.sql.ResultSet;
7 import java.sql.SQLException;
8 import java.util.ArrayList;
9 import java.util.List;
10
11     0个用法
12
13 public class E_Detection_Impl implements E_Detection_Dao {
14
15     1个用法
16     private E_Detection detection;
17
18     @Override
19     public void addInfo(E_Detection detection) {
20         String sql = "INSERT INTO detection VALUES(?, ?, ?, ?, ?, ?)";
21         Connection connection = detection.getConnection();
22         PreparedStatement preparedStatement = null;
23         try {
24             preparedStatement = connection.prepareStatement(sql);
25             preparedStatement.setInt(parameterIndex: 1, detection.getdetecNum());
26             preparedStatement.setString(parameterIndex: 2, detection.getdetectTime());
27             preparedStatement.setString(parameterIndex: 3, detection.getdetecLoc());
28             preparedStatement.setString(parameterIndex: 4, detection.getdetecPer());
29             preparedStatement.setString(parameterIndex: 5, detection.getplantId());
30             preparedStatement.setString(parameterIndex: 6, detection.getStatus());
31             preparedStatement.executeUpdate();
32             preparedStatement.close();
33         } catch (SQLException e) {
34             throw new RuntimeException(e);
35         }
36     }
37 }
```

数据

库系统课程设计报告

```
30     } catch (SQLException e) {
31         throw new RuntimeException(e);
32     }
33 }
34
35 @Override
36 public void deleteInfo(int detecNum){
37     String SQL = "delete from detection where detecNum= ?";
38     Connection connection = getConnection();
39     PreparedStatement preparedStatement = null;
40     try {
41         preparedStatement = connection.prepareStatement(SQL);
42         preparedStatement.setInt( parameterIndex: 1, detection.getdetecNum());
43         preparedStatement.executeUpdate();
44         preparedStatement.close();
45         connection.close();
46     } catch (SQLException e) {
47         throw new RuntimeException(e);
48     }
49 }
50
51 @Override
52 public void updateInfo(E_Detection detection){
53     String sql = "UPDATE detection SET detecNum=? ,detecTime=? ,detecLoc=? ,detecPer=? ,plantId=? ,status=?";
54     Connection connection = getConnection();
55     PreparedStatement preparedStatement = null;
56     try {
57         preparedStatement = connection.prepareStatement(sql);
58         preparedStatement.setInt( parameterIndex: 1, detection.getdetecNum());
59         preparedStatement.setString( parameterIndex: 2, detection.getdetecTime());
60         preparedStatement.setString( parameterIndex: 3, detection.getdetecLoc());
61         preparedStatement.setString( parameterIndex: 4, detection.getdetecPer());
62         preparedStatement.setString( parameterIndex: 5, detection.getplantId());
```

数据

库系统课程设计报告

```
63     preparedStatement.setString( parameterIndex: 6, detection.getstatus());
64     preparedStatement.executeUpdate();
65     preparedStatement.close();
66 } catch (SQLException e) {
67     throw new RuntimeException(e);
68 }
69 }

70

71 @Override
72 public List<E_Detection> findInfo(int detecNum) {
73     ArrayList<E_Detection> arrayList = new ArrayList<>();
74     String SQL = "";
75     try {
76         Connection connection = getConnection();
77         PreparedStatement preparedStatement = null;
78         SQL = "select * from E_Detection where detecNum=?";
79         preparedStatement = connection.prepareStatement(SQL);
80         preparedStatement.setInt( parameterIndex: 1, detecNum);

81

82         ResultSet resultSet = preparedStatement.executeQuery();
83         while(resultSet.next()){
84             E_Detection E_Detection = new E_Detection();
85             E_Detection.setdetecNum(resultSet.getInt( columnLabel: "detecNum"));
86             E_Detection.setdetecTime(resultSet.getString( columnLabel: "detecTime"));
87             E_Detection.setdetecLoc(resultSet.getString( columnLabel: "detecLoc"));
88             E_Detection.setdetecPer(resultSet.getString( columnLabel: "detecPer"));
89             E_Detection.setplantId(resultSet.getString( columnLabel: "plantNum"));
90             E_Detection.setstatus(resultSet.getString( columnLabel: "status"));
91             arrayList.add(E_Detection);
92         }
93         preparedStatement.close();
94         connection.close();

95     } catch (SQLException e) {
96         throw new RuntimeException(e);
97     }
98     return arrayList;
99 }
100 }
```

数据

库系统课程设计报告

```
1 package project.dao.E_Equipment ;
2 import project.entity.E_Equipment ;
3
4 import java.sql.Connection;
5 import java.sql.PreparedStatement;
6 import java.sql.ResultSet;
7 import java.sql.SQLException;
8 import java.util.ArrayList;
9 import java.util.List;
10
11 0个用法
12
13 public class E_Equipment_Impl implements E_Equipment_Dao {
14
15     1个用法
16     private E_Equipment equipment ;
17
18     @Override
19     public void addInfo(E_Equipment equipment ){
20         String sql = "INSERT INTO equipment VALUES(?,?)";
21         Connection connection = equipment.getConnection();
22         PreparedStatement preparedStatement = null;
23         try {
24             preparedStatement = connection.prepareStatement(sql);
25             preparedStatement.setInt( parameterIndex: 1, equipment .getequipNum());
26             preparedStatement.setString( parameterIndex: 2, equipment .getequipName());
27             preparedStatement.executeUpdate();
28             preparedStatement.close();
29         } catch (SQLException e) {
30             throw new RuntimeException(e);
31         }
32     }
33 }
```

数据

库系统课程设计报告

```
31     @Override
32     public void deleteInfo(int equipNum){
33         String SQL = "delete from equipment where equipNum= ?";
34         Connection connection = getConnection();
35         PreparedStatement preparedStatement = null;
36         try {
37             preparedStatement = connection.prepareStatement(SQL);
38             preparedStatement.setInt( parameterIndex: 1, equipment .getequipNum());
39             preparedStatement.executeUpdate();
40             preparedStatement.close();
41             connection.close();
42         } catch (SQLException e) {
43             throw new RuntimeException(e);
44         }
45     }
46
47     @Override
48     public void updateInfo(E_Equipment  equipment ){
49         String sql = "UPDATE equipment  SET equipNum=?,equipName=?";
50         Connection connection = getConnection();
51         PreparedStatement preparedStatement = null;
52         try {
53             preparedStatement = connection.prepareStatement(sql);
54             preparedStatement.setInt( parameterIndex: 1, equipment .getequipNum());
55             preparedStatement.setString( parameterIndex: 2, equipment .getequipName());
56             preparedStatement.executeUpdate();
57             preparedStatement.close();
58         } catch (SQLException e) {
59             throw new RuntimeException(e);
60         }
61     }
}
```

数据

库系统课程设计报告

```
63     @Override
64     public List<E_Equipment> findInfo(int equipNum) {
65         ArrayList<E_Equipment> arrayList = new ArrayList<>();
66         String SQL = "";
67         try {
68             Connection connection = getConnection();
69             PreparedStatement preparedStatement = null;
70             SQL = "select * from E_Equipment where equipNum=?";
71             preparedStatement = connection.prepareStatement(SQL);
72             preparedStatement.setInt(parameterIndex: 1, equipNum);
73
74             ResultSet resultSet = preparedStatement.executeQuery();
75             while(resultSet.next()){
76                 E_Equipment E_Equipment = new E_Equipment ();
77                 E_Equipment .setequipNum(resultSet.getInt(columnLabel: "equipNum"));
78                 E_Equipment .setequipName(resultSet.getString(columnLabel: "equipName"));
79                 arrayList.add(E_Equipment );
80             }
81             preparedStatement.close();
82             connection.close();
83         } catch (SQLException e) {
84             throw new RuntimeException(e);
85         }
86         return arrayList;
87     }
88 }
```

数据

库系统课程设计报告

```
1 package project.dao.E_Indicator;
2 import project.entity.E_Indicator;
3
4 import java.sql.Connection;
5 import java.sql.PreparedStatement;
6 import java.sql.ResultSet;
7 import java.sql.SQLException;
8 import java.util.ArrayList;
9 import java.util.List;
10
11 0个用法
12
13 public class E_Indicator_Impl implements E_Indicator_Dao {
14
15     1个用法
16     private E_Indicator indicator;
17
18     @Override
19     public void addInfo(E_Indicator indicator){
20         String sql = "INSERT INTO indicator VALUES(?,?)";
21         Connection connection = indicator.getConnection();
22         PreparedStatement preparedStatement = null;
23         try {
24             preparedStatement = connection.prepareStatement(sql);
25             preparedStatement.setInt( parameterIndex: 1, indicator.getIndexNum());
26             preparedStatement.setString( parameterIndex: 2, indicator.getIndexName());
27             preparedStatement.executeUpdate();
28             preparedStatement.close();
29         } catch (SQLException e) {
30             throw new RuntimeException(e);
31         }
32     }
33 }
```

数据

库系统课程设计报告

```
31     @Override
32     public void deleteInfo(int indexNum){
33         String SQL = "delete from indicator where indexNum= ?";
34         Connection connection = getConnection();
35         PreparedStatement preparedStatement = null;
36         try {
37             preparedStatement = connection.prepareStatement(SQL);
38             preparedStatement.setInt( parameterIndex: 1, indicator.getIndexNum());
39             preparedStatement.executeUpdate();
40             preparedStatement.close();
41             connection.close();
42         } catch (SQLException e) {
43             throw new RuntimeException(e);
44         }
45     }
46
47     @Override
48     public void updateInfo(E_Indicator indicator){
49         String sql = "UPDATE indicator SET indexNum=?,indexName=?";
50         Connection connection = getConnection();
51         PreparedStatement preparedStatement = null;
52         try {
53             preparedStatement = connection.prepareStatement(sql);
54             preparedStatement.setInt( parameterIndex: 1, indicator.getIndexNum());
55             preparedStatement.setString( parameterIndex: 2, indicator.getIndexName());
56             preparedStatement.executeUpdate();
57             preparedStatement.close();
58         } catch (SQLException e) {
59             throw new RuntimeException(e);
60         }
61     }
}
```

数据

库系统课程设计报告

```
63
64     @Override
65     public List<E_Indicator> findInfo(int indexNum) {
66         ArrayList<E_Indicator> arrayList = new ArrayList<>();
67         String SQL = "";
68         try {
69             Connection connection = getConnection();
70             PreparedStatement preparedStatement = null;
71             SQL = "select * from E_Indicator where indexNum=?";
72             preparedStatement = connection.prepareStatement(SQL);
73             preparedStatement.setInt(parameterIndex: 1, indexNum);
74
75             ResultSet resultSet = preparedStatement.executeQuery();
76             while(resultSet.next()){
77                 E_Indicator E_Indicator = new E_Indicator();
78                 E_Indicator.setindexNum(resultSet.getInt(columnLabel: "indexNum"));
79                 E_Indicator.setindexName(resultSet.getString(columnLabel: "indexName"));
80                 arrayList.add(E_Indicator);
81             }
82             preparedStatement.close();
83             connection.close();
84         } catch (SQLException e) {
85             throw new RuntimeException(e);
86         }
87         return arrayList;
88     }
```

六、项目运维管理和优化（观测点 11.1）

(一) 管理工具和实施过程

小组使用 github 平台的仓库进行项目托管，每位同学按照小组设计的推送要求将自己部分的数据库架构和数据有序推送到仓库中，全部上传完成后，进行数据库脚本的整合拷贝副本至本地文件进行进一步的测试工作，修改后推送给仓库，再进行进一步数据库架构和数据修改。

(1) 下载 Github Desktop 软件

优点：代替命令行操作，提供图形界面操作更加简单便捷



(2) 创建仓库，添加协作成员，由组长创建仓库，并向其它小组成员发出协作邀请，其他成员需通过协作邀请

A screenshot of the GitHub Desktop application interface showing repository settings. It lists three collaborators: '1zpf' (Collaborator), 'wuhula' (Collaborator), and 'yangyuan' (Collaborator). Each entry includes a checkbox, a remove button, and a 'Remove' link. Below the list, there's a section for organizations with a 'Create an organization' button. A note says: 'Get team access controls and discussions for your contributors in an organization.' A 'NEW' badge indicates that 'Private repos and unlimited members are free.'

(3) 协作成员添加完成后，小组成员登录自己的 Github Desktop 可发现自己的页面中已存在 database 仓库

数据

库系统课程设计报告

Home
Issues
Pull requests
Projects
Discussions
Codespaces

Explore
Marketplace

Repositories

wyn725/database
yan'an1006/Database-Project

(4) 推送即可将修改后的代码进行上传

Merge remote-tracking branch 'origin/master'

yangyuan -> 26d9f49 → +117 -11

4 changed files

src\Dao\Plant\PlantDaoImpl.java → src\Dao\Impl\PlantDaoImpl.java

```
@@ -1,13 +1,13 @@  
- package Dao.Plant;  
+ package Dao.Impl;  
+ import Dao.DaoBase;  
+ import Dao.PlantDao;  
+ import Entity.Plant;  
+ import java.sql.Connection;  
+ import java.sql.PreparedStatement;  
+ import java.sql.SQLException;  
- import java.util.List;  
-  
- public class PlantDaoImpl extends DaoBase implements Dao.plant.PlantDao {  
+ public class PlantDaoImpl extends DaoBase implements PlantDao {  
    @Override  
    /*PreparedStatement 是 Java JDBC 中的一个接口，用于表示预编译的 SQL 语句的对象。  
     * 它是 java.sql.PreparedStatement 接口的实例。通过使用 PreparedStatement,  
     * 可以在执行查询或更新语句时自动处理参数。*/  
    public List<Plant> findPlant(String id, int i) {  
        return null;  
    }  
    public void findPlant(Plant plant) {  
        String sql = "select A_Plant.*, A_Picture.* " +  
                    "FROM A_Plant p\n" +  
                    "JOIN A_Plant_Picture pp ON p.id = pp.plantId\n" +  
                    "JOIN A_Picture pic ON pp.photoId = pic.photoId" +  
                    "where id= ?;"  
        Connection connection = getConnection();  
        PreparedStatement preparedStatement = null;
```

No releases published
Create a new release

Packages

No packages published
Publish your first package

Contributors 4

yangyuan-ada yangyuan
yan'an1006
1zpf
wyn725 wyn

数据

库系统课程设计报告

(5) WPS 共享文件

ER.pom	今天 17:39:14
数据库系统课程设计报告模板2023 (小组) .docx	4人正在编辑 4.65MB
视图及触发器构建.docx	2人正在编辑 17.65KB
小组课设-可编辑版.docx	3人正在编辑 2.57MB
鲁棒图.pom	今天 15:45:18
用例.pom	今天 12:27:09
数据库系统属性命名大全.docx	今天 01:06:53 16.48KB
1.docx	今天 00:02:36 13.12KB
数据库系统课程设计报告模板2023 (个人) .docx	昨天 15:47:45 14.72KB
数据表数据填充.docx	2人正在编辑 20.79KB
数据库系统用例图整理.docx	2023-12-08 13.03KB

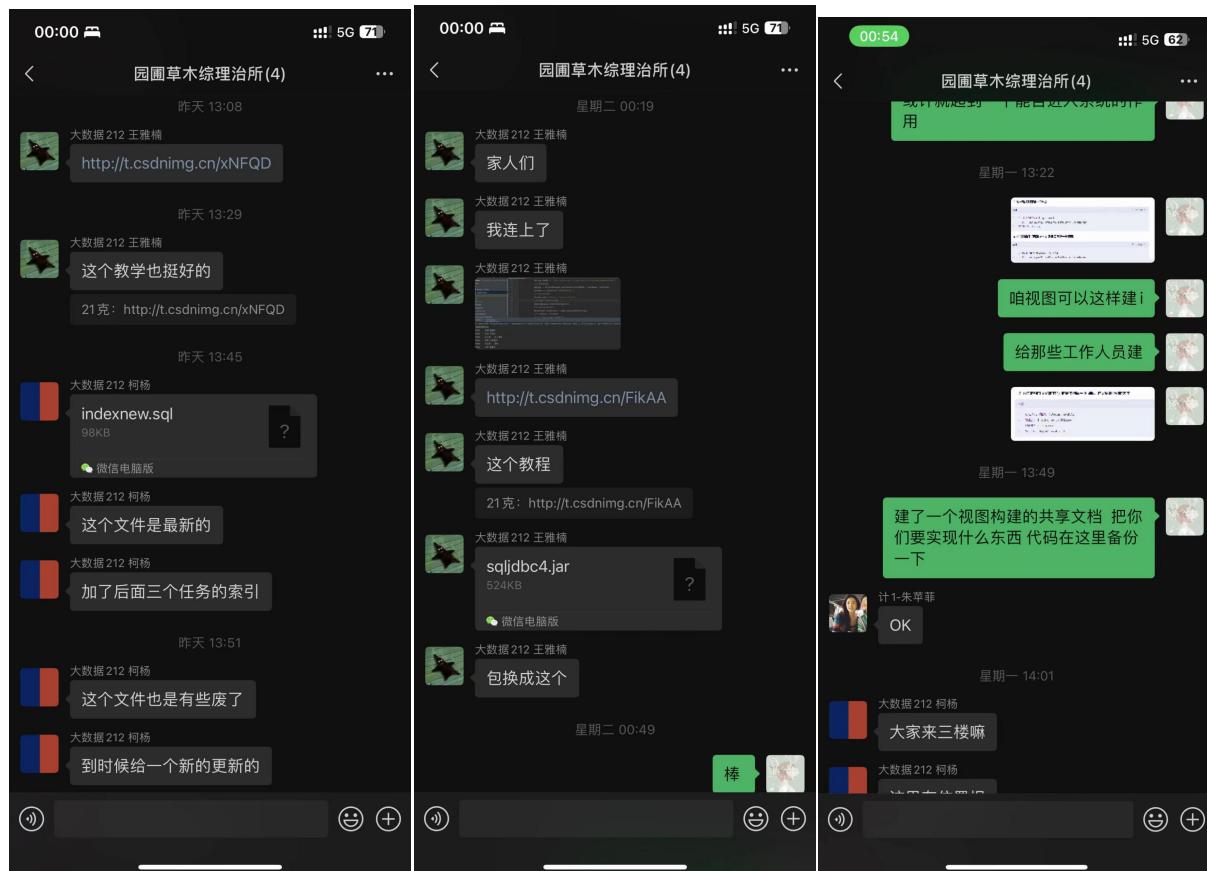
(6) 线上线下结合

线下集中讨论



数据

库系统课程设计报告



(二) 数据优化管理

数据

库系统课程设计报告

1. 视图：

1.1 植物信息和植物分类交互

视图查询各科植物的总数

```
CREATE VIEW PlantCountByKind AS
```

```
SELECT
```

```
    B_Classification.classifyID AS kind_id,  
    B_Classification.alterName AS kind_name,  
    COUNT(A_Plant.id) AS plant_count
```

```
FROM
```

```
    B_Classification
```

```
LEFT JOIN
```

```
    A_Plant ON B_Classification.classifyID = A_Plant.classifyID
```

```
WHERE
```

```
    B_Classification.classifyType = '科'
```

```
GROUP BY
```

```
    B_Classification.classifyID,  
    B_Classification.alterName;
```

1.2 植物信息和养护管理交互

人物植物任务视图：使用于园林地点保证人物可以对应任务找到对应植物

```
CREATE VIEW C_PersonTaskPlant AS
```

```
SELECT
```

```
    m.mainPer,  
    t.taskName,  
    p.alternname
```

```
FROM C_Task t
```

```
LEFT JOIN C_TaskMaintenance tm ON t.taskID = tm.taskID
```

```
LEFT JOIN C_Maintenance m ON tm.mainID = m.mainID
```

```
LEFT JOIN A_Plant p ON m.plantId = p.id;
```

1.3 植物信息和防治管理交互

植物防治视图：使用于园林地点保证人物可以对应防治找到对应植物

```
CREATE VIEW PlantControlView AS
```

```
SELECT A_Plant.id AS PlantID, A_Plant.alternname, D_Control.conID, D_Control.conTime,
```

```
    D_Control.conLoc, D_Control.conPer, D_Control.status
```

```
FROM A_Plant
```

```
LEFT JOIN D_Control ON A_Plant.id = D_Control.id;
```

数据

库系统课程设计报告

1.4 植物信息和监测管理交互

植物设备指标视图：使用于园林地点保证植物可以对应设备找到对应指标

```
CREATE VIEW E_PlantEquipmentIndicator AS
```

```
SELECT
```

```
p.alterName,  
e.equipName,  
i.indexName  
FROM E_EquipDetecIndica edi  
JOIN E_Equipment e ON edi.equipNum = e.equipNum  
JOIN E_Indicator i ON edi.indexNum = i.indexNum  
JOIN E_Detection d ON edi.detecNum = d.detecNum  
JOIN A_Plant p ON d.plantId = p.id;
```

2. 索引

2.1 植物信息和植物分类交互

```
CREATE INDEX idx_Plant_Kind ON A_Plant (classifyID);
```

对于植物的分类编号进行检索，对于 classifyID 进行排序或分组操作

2.2 物信息和养护管理交互

```
CREATE INDEX idx_maintenancePlant ON C_Maintenance (plantId);
```

对于植物的植物编号检索，使用 plantId 连接植物信息表

2.3 植物信息和防治管理交互

```
CREATE INDEX idx_controlPlant ON D_Control (plantId);
```

对于植物的植物编号检索，使用 plantId 连接植物信息表

2.4 植物信息和监测管理交互

```
CREATE INDEX idx_detectionPlant ON E_Detection (plantId);
```

对于植物的植物编号检索，使用 plantId 连接植物信息表

(三)数据库高级编程

1.园林植物基本信息管理

1.1 存储过程

实现方案：

```
CREATE PROCEDURE GetPlantDetailsById
```

```
    @plantId VARCHAR(50) -- 输入参数：植物编号
```

```
AS
```

```
BEGIN
```

```
    SELECT
```

数据

库系统课程设计报告

```
p.altername AS PlantName,  
c1.alterName AS FamilyName,  
c2.alterName AS GenusName,  
a.province,  
a.city  
FROM  
    A_Plant p  
JOIN  
    B_Classification c1 ON p.classifyID = c1.classifyID  
LEFT JOIN  
    B_Classification c2 ON c1.classifyID = c2.classifyID_parent  
LEFT JOIN  
    B_Area a ON p.disAreaID = a.disAreaID  
WHERE  
    p.id = @plantId;  
END;
```

适用范围：

适用于进行植物的名称、科属种和对应的分布地区联合查询，将植物基本信息表、植物分类表、分布地区表联动

1.2 触发器

实现方案：

```
CREATE TRIGGER PreventDeletePlantTrigger  
ON A_Plant  
INSTEAD OF DELETE  
AS  
BEGIN  
    -- 存储未完成的任务列表  
    DECLARE @UnfinishedTasks NVARCHAR(MAX) = '';  
    -- 检查养护表中是否存在未完成的任务  
    IF EXISTS (SELECT 1 FROM C_Maintenance WHERE status = '未完成' OR status = '进行  
    中')  
        BEGIN  
            SET @UnfinishedTasks = '养护任务';  
        END
```

数据

库系统课程设计报告

```
-- 检查防治表中是否存在未完成的任务
IF EXISTS (SELECT 1 FROM D_Control WHERE status = '未完成' OR status = '进行中')
BEGIN
    IF @UnfinishedTasks <> "
        SET @UnfinishedTasks = @UnfinishedTasks + ', 防治任务';
    ELSE
        SET @UnfinishedTasks = '防治任务';
END

-- 检查监测表中是否存在未完成的任务
IF EXISTS (SELECT 1 FROM E_Detection WHERE status = '未完成' OR status = '进行中')
BEGIN
    IF @UnfinishedTasks <> "
        SET @UnfinishedTasks = @UnfinishedTasks + ', 监测任务';
    ELSE
        SET @UnfinishedTasks = '监测任务';
END

-- 如果存在未完成的任务，则抛出错误提示
IF @UnfinishedTasks <> "
BEGIN
    DECLARE @ErrorMessage NVARCHAR(MAX) = '无法删除植物，存在未完成的 ' +
    @UnfinishedTasks + '! ';
    RAISERROR(@ErrorMessage, 16, 1);
    ROLLBACK TRANSACTION;
    RETURN;
END

-- 如果没有未完成的任务，则可以执行植物删除操作
DELETE FROM A_Plant WHERE id IN (SELECT id FROM deleted);
END;
```

适用范围：

当删除植物信息时，若当前植物存在养护、防治、监测其中任一任务时，禁止删除相关数据

数据

库系统课程设计报告

2.1 存储过程

实现方案：

```
CREATE PROCEDURE usp_DeleteB_Classification
    @classifyID INT
AS
BEGIN
    SET NOCOUNT ON;

    -- 检查是否存在引用该数据的记录
    IF EXISTS (SELECT 1 FROM B_Classification WHERE classifyID_parent = @classifyID)
    BEGIN
        RAISERROR('无法删除，存在引用该数据的记录。', 16, 1);
        RETURN;
    END;
    -- 执行删除操作
    DELETE FROM B_Classification
    WHERE classifyID = @classifyID;
    SELECT '删除成功' AS Message;
END;
```

2.2 触发器

实现方案：

```
CREATE TRIGGER trg_B_Classification_Delete
ON B_Classification
INSTEAD OF DELETE
AS
BEGIN
    -- 检查是否存在引用该数据的记录
    IF EXISTS (SELECT 1 FROM B_Classification WHERE classifyID_parent IN (SELECT
    classifyID FROM deleted))
    BEGIN
        RAISERROR('无法删除，存在引用该数据的记录。', 16, 1);
        ROLLBACK TRANSACTION; -- 取消删除操作
        RETURN;
    END;

```

```
-- 执行删除操作  
DELETE FROM B_Classification  
WHERE classifyID IN (SELECT classifyID FROM deleted);
```

```
COMMIT TRANSACTION; -- 提交事务
```

```
END;
```

适用范围：

该触发器是实现的是，在分类表中的树状存储结构中，如果要删除数据，那么就要满足被删除的这条数据不可以被其他记录所引用，不可以有子节点数据，否则就不可以执行删除操作，保证数据的完整性。

3.园林植物养护管理

3.1 存储过程

实现方案：

```
CREATE PROCEDURE proc_C_TaskMaintenance_Select  
AS  
BEGIN  
    SELECT *  
    FROM [C_TaskMaintenance]  
    WHERE mainID = 2010;  
END
```

```
EXEC proc_C_TaskMaintenance_Select
```

适用范围：

在数据库中查找表中 mainID 为 2010 的记录并返回所有相关数据。整个数据表中按照顺序进行实现，查询表中 mainID 为 2010 的记录为最后一条，从而确保整个数据表的完整性。

3.2 触发器

实现方案：

```
CREATE TRIGGER trg_C_TaskMaintenance_Update  
ON C_TaskMaintenance  
AFTER UPDATE  
AS  
BEGIN
```

数据

库系统课程设计报告

```
IF UPDATE(taskID) OR UPDATE(mainID)
BEGIN
    PRINT 'C_TaskMaintenance 信息已出现修改。';
END
END;
```

适用范围：

数据表中任意数据项进行修改，触发器给出相关提示信息。数据表为整个任务中的关系连接表，数据表出现修改时给出相关提示信息，有助于数据表相关安全性维护。

4.园林植物病虫害防治管理

4.1 存储过程

```
CREATE PROCEDURE select_S
AS
BEGIN
```

```
    SELECT conPer FROM D_Control WHERE status='未完成';
END
```

-- 执行存储过程

```
EXECUTE select_S;
```

适用范围：便于上级主管部门通过防治（任务）表，查询具体工作人员的执行情况，上级主管部门查到防治任务“未完成”时采取相应措施提醒防治人员进行工作。通过使用该存储过程，可以在数据库端执行复杂的操作，从而简化客户端应用程序的复杂性。同时只允许通过存储过程进行数据操作，从而提高数据的安全性。

4.2 触发器

```
CREATE TRIGGER trg_D_MediPestCon_Update
ON D_MediPestCon
AFTER UPDATE
AS
BEGIN
IF UPDATE(conID) OR UPDATE(pestID) OR UPDATE(mediID)
BEGIN
    PRINT 'D_MediPestCon 修改成功。';
END ELSE BEGIN PRINT '没有字段被更新。';
END; END;
```

数据

库系统课程设计报告

适用场景：触发器的功能是检查在更新操作中哪些字段被更改了。如果 conID、pestID 或 mediID 中的任何一个字段被更新，它将打印出"D_MediPestCon 修改成功。"; 如果所有字段都没有被更新，它将打印出"没有字段被更新。"

5.园林植物监测管理

5.1 存储过程

实现方案：

```
CREATE PROCEDURE proc_E_EquipDetecIndica_Select
AS
BEGIN
    SELECT *
    FROM [E_EquipDetecIndica]
    WHERE detecNum = 3010;
END
```

```
EXEC proc_E_EquipDetecIndica_Select
```

适用范围：

在数据库中查找表中 detecNum 为 3010 的记录并返回所有相关数据。整个数据表中按照顺序进行实现，查询表中 detecNum 为 3010 的记录为最后一条，从而确保整个数据表的完整性。

5.2 触发器

实现方案：

```
CREATE TRIGGER trg_E_EquipDetecIndica_Update
ON E_EquipDetecIndica
AFTER UPDATE
AS
BEGIN
    IF UPDATE(equipNum) OR UPDATE(indexNum) OR UPDATE(detecNum)
    BEGIN
        PRINT 'E_EquipDetecIndica 信息已出现修改。';
    END
END;
```

适用范围：

数据表中任意数据项进行修改，触发器给出相关提示信息。数据表为整个任务中的关系连接表，数据表出现修改时给出相关提示信息，有助于数据表相关安全性维护。

实现方案：

数据

库系统课程设计报告

```
CREATE TRIGGER trg_E_EquipDetecIndica_Abnormal
ON E_EquipDetecIndica
AFTER UPDATE
AS
BEGIN
    IF UPDATE(indexNum)
    BEGIN
        IF EXISTS (SELECT 1 FROM inserted WHERE indexNum IN (2004,
2005))
        BEGIN
            PRINT 'E_EquipDetecIndica 信息出现异常。';
        END
    END
END;
```

适用范围：

数据表中指标信息出现异常，触发器给出相关提示信息。

(四)数据安全和数据备份

1. 备份工作

为防止服务器出现问题导致数据库数据丢失，小组在初步完成数据表的设计以及相关数据的填充后每隔一天就会进行数据库脚本的导出备份工作，如果数据库遭到破坏，将选用最后一次导出的脚本实现数据库的恢复。

与此同时，在 GitHub 中导入了数据库结构以及数据，让小组成员的数据可以达到共享与协同编写，在 Java 中 DAO 的开发中也进行了每一部分文件的备份，以防止环境问题导致的代码被破坏。

数据

库系统课程设计报告

名称	修改日期	类型	大小
📁 secondtimenew	2023/12/24 22:24	文件夹	
📄 D_AllMediPestCon.sql	2023/12/25 19:36	Microsoft SQL Serv...	1 KB
📄 D_LocationPestInfo.sql	2023/12/25 14:46	Microsoft SQL Serv...	1 KB
📄 D_MedicationUsage.sql	2023/12/25 14:42	Microsoft SQL Serv...	1 KB
📄 F1527.sql	2023/12/26 15:27	Microsoft SQL Serv...	102 KB
📄 indexfinal.sql	2023/12/26 14:05	Microsoft SQL Serv...	103 KB
📄 new.sql	2023/12/25 19:45	Microsoft SQL Serv...	87 KB
📄 plant_final.sql	2023/12/25 11:49	Microsoft SQL Serv...	64 KB
📄 plantnew.sql	2023/12/24 23:58	Microsoft SQL Serv...	25 KB
📄 PlantTotalData.sql	2023/12/25 15:52	Microsoft SQL Serv...	67 KB
📄 script(2).sql	2023/12/25 9:50	Microsoft SQL Serv...	21 KB
📄 script.sql	2023/12/24 17:02	Microsoft SQL Serv...	28 KB
📄 SQLQuery小项目.sql	2023/12/25 12:47	Microsoft SQL Serv...	6 KB
📄 第二次新.sql	2023/12/25 0:28	Microsoft SQL Serv...	56 KB

2. 避免 SQL 注入

避免 sql 注入，在 Dao 层编写过程中，对于需要传入参数的 SQL 语句，均使用了 PrepareStatement，而不是用 Statement 进行对应的字符串拼接，避免用户恶意输入实现 Sql 注入造成数据库信息损失。

3. 输出信息控制

对于 Sql 语句的执行过程，可能会出现对应的报错，对于开发人员来说，报错有助于检查数据库结构是否符合规定，但这可能会成为攻击者探查后端数据库内部结构很好途径，因此在 Dao、Service 编写时，只输出了对应的一般信息，不会展示 sql 错误的具体信息。

(五)关键问题和解决方法

关键问题 1：配图图片在数据库中的存储，系统管理员删除植物信息的时候系统需要考虑当前植物是否存在养护、防治、监测任务。

解决方法 1：配图图片在数据库中用图片路径存储，采用触发器进行解决。

关键问题 2：在进行脚本传输文件时，发现 sql server 数据库存在“记忆”，即在进行多次脚本导入导出数据时，电脑有自己的记忆，尽管导入新的一版脚本时，出现的数据仍是导入之前的。所以在进行脚本更新时，要及时抽查表格数据，查看导入是否

成功。

解决方法 2：小组成员还发现重新建立角色再进行导入脚本操作会避免这种现象。

关键问题 3：项目进度顺序发生错误，小组在进行初期数据库概念结构构建时就先绘制了 ER 图，没有经过后期表结构连接测试，在进行真正 sql 语句操作时发现问题，是表属性连接出错，所以需要全部重新再来，损耗大量时间。

解决方法 3：在进行真正的 er 图绘制，之前一定要确保各实体和实体属性的正确性，先画草图再经过各种健壮性测试再绘制终版 ER 图。

七、附件内容

- 1- 协同文件
- 2- 答辩 ppt
- 3- 讲解录屏
- 4- 各任务 ER 图+用例图
- 5- UML 类图
- 6- 全局 ER 图
- 7- 各任务鲁棒图
- 8- 系统用例图
- 9- 代码
- 10- plant-database.sql
- 11- 答辩反馈与解决方案
- 12- DDL 语句汇总