数据库系统课程设计报告

姓名	张万泉	班级	物联网 23
学号	230101524	指导教师	崔晓辉
组名	1		

团队其他成员（可自行增加行数，第一为组长）

姓名	学号	班级
张万泉	230101524	物联网 23
李雪彤	231002616	物联网 23
蔡邵涵	231002605	物联网 23
赵雅萱	220701618	物联网 23
杨嘉雯	220201409	大数据

1、个人和团队

课程设计概述

（要求：总结课程设计的业务内容-200 字-300 字-1.5 倍行距-5 号宋体）

生物多样性监测业务线的用例以“采集—核实—审核—统计”为主流程，核心参与者为生态监测员（ecological_monitor）与数据分析师（data_analyst）。

系统包含三类核心对象：Species（物种）、Habitat（栖息地）、MonitoringRecord（监测记录）。

生态监测员执行用例 Upload Monitoring Record：选择目标 Species，绑定 MonitoringDevice(device_id)、记录人 recorder_id，填写 monitoring_time 与地理坐标（longitude/latitude），并上传监测内容（image_path/count_number/behavior_description）。提交后系统执行用例 Verify Monitoring Record Integrity，对必填字段与关联完整性进行校验，将记录状态置为 status='pending'。生态监测员可执行 Recheck Monitoring Record，对待核实记录二次确认并更新为 status='rechecked'。数据分析师执行 Valid Monitoring Record，对记录进行最终审核并置为 status='valid'，随后基于有效记录执行 Generate Monitoring Statistics Report 输出统计报表。结构层面，MonitoringRecord 关联 Species、MonitoringDevice 与

	User(recorder)；Habitat 作为空间与生态载体，与 Species 存在多物种关联（主要物种关系可通过关联结构表达），用于支撑“物种—栖息地—监测记录”的跨实体查询与分析。
个人任务情况	**（要求：总结个人的任务分工、每一项任务的重点工作、任务的预期达成情况-150-200 字-1.5 倍行距-5 号宋体）** 本人负责 GitHub 仓库管理及生物多样性监测业务线的核心设计与实现。前期完成用例图、鲁棒图及局部 UML 设计，并整合全局 UML（Issue #17）。随后完成 Species、Habitat、MonitoringRecord 等表的 DDL 设计与说明（Issue #26）。实现阶段编写测试数据、复杂 SQL，并完成 DAO 持久层代码与单元测试（Issue #39）。后期完成视图、索引、存储过程及触发器设计（Issue #49），并参与安全与备份方案讨论（Issue #54）。当前个人报告整理与提交工作按计划推进（Issue #61），已完成任务均达预期目标。
团队协作情况	**（要求：总结个人在团队中的协作情况、与哪些任务或者人员之间产生协作、协作的主要内容等-150-200 字-1.5 倍行距-5 号宋体）** 在团队协作中，我全程管理 GitHub 仓库，制定协作规范与目录结构，编写 README 与相关说明文档，并通过 Issue ，分支与 PR 流程进行审查。组织三次小组会议，分别围绕任务分工、数据库安全与备份策略（Issue #54）、以及小组报告与答辩 PPT 的组织方式展开讨论，形成可执行结论。技术协作方面，在合并全局 UML 类图过程中（Issue #17），与 C、D、E 就表结构划分、实体关系及字段设计进行讨论；同时在数据字典与结构问题讨论中，参与相关 Issue 的解决（如 #18、#19）。
2、问题分析	

业务需求分析

（要求：使用用例图和鲁棒图总结个人在数据库系统课程设计所负责的业务需求-200-300 字-1.5 倍行距-5 号宋体）

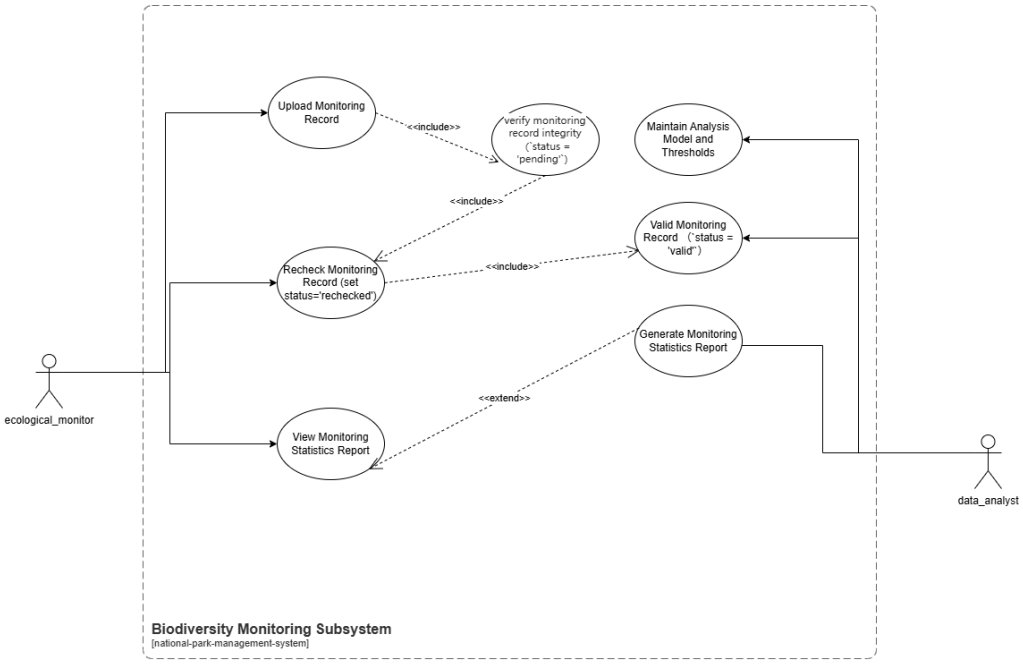

Figure 1 用例图

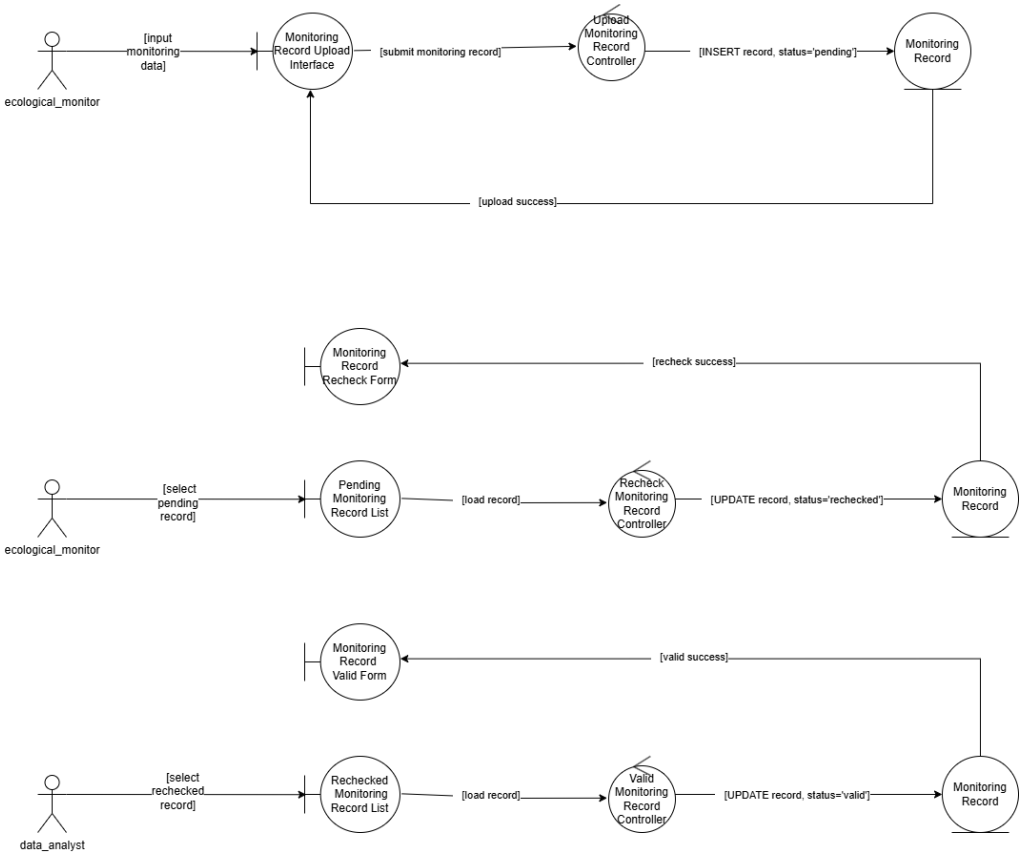

Figure 2 鲁棒图

	生物多样性监测业务线围绕“监测数据的采集、核实、审核与分析”构建完整业务流程，用例图明确了生态监测员、数据分析师与系统三类参与者的职责分工。生态监测员通过 **Upload Monitoring Record** 用例上传物种监测数据，系统在该过程中自动包含 **Verify Monitoring Record Integrity**，对数据完整性进行校验并将记录状态初始化为 pending。随后，生态监测员可对待核实数据执行 **Recheck Monitoring Record** 用例，对监测内容进行补充或修正，记录状态更新为 rechecked。数据分析师基于已复查数据执行 **Valid Monitoring Record** 用例，从专业角度完成终审，将数据确认为 valid，并进一步通过 **Generate Monitoring Statistics Report** 生成统计分析结果。 鲁棒图从对象交互层面细化了上述用例的实现过程，清晰划分了 Boundary、Controller 与 Entity 的职责边界，重点体现了监测记录在 pending → rechecked → valid 各状态之间的转换。
业务的非功能需求	**（要求：总结个人在数据库系统课程设计所负责业务的安全性、完整性需求-300 字-500 字-1.5 倍行距-5 号宋体）** 在生物多样性监测业务线中，系统的安全性与数据完整性需求主要围绕“防止越权访问、保证数据状态一致性、防止脏数据进入统计分析”展开。在数据库层面，我通过视图与触发器对核心业务逻辑进行约束与固化，降低对应用层的依赖，提升整体可靠性。 在完整性控制方面，通过在 MonitoringRecord 表上设计 **BEFORE INSERT 触发器**，统一维护监测记录的初始状态与关键字段校验逻辑。所有新插入的监测记录均被强制设置为待核实状态，并在入库阶段校验监测时间、经纬度等关键字段的合法性，避免缺失或明显异常的数据直接进入后续复查与统计流程 在安全与访问控制方面，通过为生物多样性业务线设计多类只读或受限 **VIEW**，实现“按角色暴露最小数据集”的原则。例如面向数据分析师的待核实记录视图，统一联表封装物种、设备与区域信息，避免直接开放底层明细表；面向管理与分析场景的统计视图仅提供聚合结果，不暴露原始敏感字段。

	此外，在 Issue #54 的安全与备份策略讨论中，明确了 RBAC 权限模型、登录安全策略及“每日增量、每周全量”的备份方案。
业务的局部概念结构设计	**（要求：总结个人在数据库系统课程设计所负责业务的局部 E-R 图，需求-300 字-500 字-1.5 倍行距-5 号宋体）** 围绕“物种—栖息地—监测记录”三类核心业务对象展开，重点支撑监测数据的采集、复查、审核及后续统计分析需求。概念结构设计以 Species（物种）、Habitat（栖息地）、MonitoringRecord（监测记录） 为核心实体，并通过合理的实体关系表达生态监测场景下的数据关联与业务约束。 其中，Species 实体用于描述物种的基础与分类信息，包含分类层级、保护级别、生存习性与分布特征，满足物种档案管理与保护等级统计需求；Habitat 实体用于描述栖息地的生态类型、面积、核心保护范围及环境适宜性评分，为分析物种分布与环境关系提供结构支撑。针对“一个栖息地包含多个主要物种、一个物种可能出现在多个栖息地”的需求，引入 HabitatPrimarySpecies 作为关联实体，实现多对多关系的规范化建模，并避免数据冗余。 MonitoringRecord 实体用于刻画一次具体的监测行为，是业务流转的核心载体。该实体通过外键关联 Species、MonitoringDevice 与 User，完整记录监测时间、空间位置、监测方式及监测内容，并通过状态字段刻画监测数据在 pending → rechecked → valid 各阶段的业务流转。该状态设计与用例图、鲁棒图中的职责划分保持一致，保证数据处理过程可追溯、可控制。
3、设计、开发解决方案	

课程设计工作详述	**（要求：总结个人说涉及业务的逻辑结构设计、物理结构设计、业务相关的 SQL 代码、索引和视图设计，需与答辩时候的分工、内容一致-800-1000 字-5 号宋体）** 本人在课程设计中负责生物多样性监测业务线的数据库设计与实现，工作内容覆盖从概念结构向逻辑结构的转换、物理表结构落地，以及面向业务查询与运维需求的 SQL、索引与视图设计。 在逻辑结构设计阶段，基于用例图、鲁棒图与局部 UML 类图，将核心实体转换为关系模式。业务核心表包括 Species、Habitat、MonitoringRecord 以及用于处理多对多关系的 HabitatPrimarySpecies。其中，Species 表以 species_id 为主键，完整描述物种分类层级与保护级别；Habitat 表以 habitat_id 为主键，描述生态类型、面积与环境适宜性；MonitoringRecord 作为业务流转核心表，以 record_id 为主键，关联物种、设备与记录人，并通过 status 字段刻画监测数据在 pending / rechecked / valid 各阶段的状态变化。所有关系模式均满足第三范式，避免将设备、用户或物种的属性冗余存储在监测记录中，仅通过外键关联，保证数据一致性。 在物理结构设计阶段，基于 MySQL 8.x 实现具体 DDL。字段类型统一采用数据字典规范，如经纬度使用 DECIMAL(10,6)，时间字段使用 DATETIME，状态与枚举字段使用 ENUM 或受控字符串。主键均采用业务主键而非自增 ID，便于与外部系统或设备数据对接。外键约束在同业务文件内显式声明，对跨业务或全局表的关联在逻辑层保持一致但不强制物理约束，以降低初始化与集成复杂度。 围绕核心业务查询，编写了多条多表关联 SQL，并结合实际访问模式设计索引。在 MonitoringRecord 表上重点设计了 (status, monitoring_time)、(species_id, monitoring_time)、(device_id, monitoring_time) 等复合索引，用于支撑“待核实记录列表”“按物种统计近 30 天监测情况”“按设备追踪数据质量”等高频场景。索引设计遵循“按 WHERE + ORDER BY 联合出现顺序建立”的原则，避免单列低选择性索引带来的性能浪费。 在视图设计方面，针对不同角色的数据访问需求，设计了多类业务视图以替代直接访问底层表。例如，面向数据分析师的待核实记录视图，将监测记录一次性联表到

	物种、设备、区域与记录人信息，减少重复 SQL 编写；面向管理层的统计视图，按区域与时间窗口聚合监测记录数量与状态分布，仅暴露汇总结果，不暴露原始明细字段。所有视图均遵循最小权限原则，既降低误操作风险，也为后续 RBAC 授权提供基础。 整体来看，该业务线的数据库设计以“状态可控、关系清晰、查询可优化”为目标，将业务流程中的关键约束前移到数据库层实现，为后续持久层代码、统计分析与系统运维提供了稳定的数据基础。
4、项目管理	

工程管理	（要求：总结个人涉及业务使用 GITHUB 等工具管理情况-300 字左右--1.5 倍行距-5 号宋体） 本课程设计中，我负责生物多样性监测业务线的数据库设计与实现，并通过 GitHub 对相关技术成果进行系统化管理。项目仓库采用分层目录结构，将需求分析、数据库脚本、程序代码与测试结果进行明确拆分。我负责的核心内容集中在 sql/DDL/10_biodiversity.sql、sql/DML/10_biodiversity_seed.sql、sql/VIEW、sql/INDEX、sql/PROC、sql/TRIGGER 以及 src/dao/biodiversity 等目录下，形成从表结构设计到业务逻辑封装的完整链路。 在工程管理过程中，所有涉及数据库结构调整、复杂 SQL 优化及存储过程、触发器编写的工作，均通过 Git 分支独立完成，并以 Pull Request 的方式合并至主分支，确保关键数据库变更具备可审查性与可回滚性。每一次提交均保持“单一职责”，对应明确的 SQL 文件或功能模块，避免混合提交导致的维护困难。 在开发阶段，我结合单元测试对生物多样性模块的 DAO 层进行验证，测试代码统一放置于 tests/biodiversity 目录，通过自动化测试保证 CRUD 操作与数据库设计的一致性。借助 GitHub 的提交记录与 PR 讨论过程，实现了数据库设计、实现与优化过程的全过程追踪，为项目的稳定迭代和最终答辩提供了可靠的工程依据。
风险和安全管理	（要求：总结实现系统的潜在风险管理及控制手段-不少于 300 字-1.5 倍行距-5 号宋体） 在系统实现过程中，结合安全专题会议讨论内容，重点识别并控制了数据库层面与业务流程中的主要风险，形成了可落地的风险管理与安全控制方案。 首先针对**越权访问风险**，系统采用基于角色的访问控制（RBAC）模型，对生态监测员、数据分析师、执法人员、科研人员等角色进行权限划分。权限粒度不直接作用于基础表，而是以“可访问视图 + 可写表范围”为核心控制手段。通过为不同角色设计专用视图，仅暴露其业务所需字段和记录范围，避免直接访问敏感表或无关数据，从源头降低误操作与越权查询风险。 其次针对**数据泄露与误用风险**，对敏感字段进行明确界定，如监测影像路径、执法证据位置、人员身份标识等，禁止在通用查询或统计视图中直接暴露，仅在必要业

	务场景下通过受控视图访问。同时，业务查询与持久层实现统一采用参数化 SQL，禁止字符串拼接，降低 SQL 注入风险。 在**账号与登录安全**方面，会议明确采用加密方式存储密码，并引入登录失败次数限制与账户临时锁定机制，防止暴力破解。同时设置会话超时控制，避免长时间未操作导致的会话劫持风险。 最后，在**数据可靠性与灾难恢复风险**方面，制定了“每日增量备份、每周全量备份”的备份策略，明确备份命名规则、存储路径及恢复流程，确保在误删除、系统故障或数据损坏场景下能够快速恢复业务数据。 通过以上措施，将安全控制落实到数据库结构、访问方式与运维流程中，形成了与业务紧密结合、具备可操作性的风险管理体系。
运维和优化管理	**（要求：结合存储过程和触发器，总结所涉及业务的运维和优化该方法-不少于 300 字-1.5 倍行距-5 号宋体）** 在运维自动化方面，设计并实现了用于监测数据入库的存储过程 sp_submit_monitoring_record。该过程将原本分散在应用层的多步操作（参数合法性校验、外键存在性检查、记录编号生成、数据插入）封装为一次数据库调用，确保所有新监测记录在写入时均满足基本业务约束，并统一从 status='to_verify' 状态进入后续核实流程。通过存储过程集中处理写入逻辑，可减少应用端重复代码，在批量导入、设备自动上报等场景下显著降低运维成本，同时便于后期统一修改业务规则。 在数据质量与一致性控制方面，通过在 MonitoringRecord 表上定义 BEFORE INSERT 触发器，对关键字段进行自动校验与维护。触发器在数据入库阶段强制设置初始状态，并对监测时间、经纬度范围等字段进行合法性检查，避免明显异常或缺失数据进入数据库。一旦校验失败，触发器直接阻断插入操作，从源头防止脏数据扩散至统计分析与业务视图中。

	从优化角度看，存储过程与触发器的引入，使数据库能够承担一部分业务规则执行职责，减少了应用层与数据库之间的交互次数，提高写入路径的整体效率。同时，这种设计也增强了系统的可维护性：当业务规则调整时，只需修改数据库对象即可生效，无需同步修改多个应用模块。整体上，该方法在保证数据质量的同时，提升了系统运维效率与业务执行的一致性。