数据库系统课程设计报告

姓名	李雪彤	班级	物联网 23 班
学号	231002616	指导教师	崔晓晖
组名	1		

团队其他成员（可自行增加行数，第一为组长）

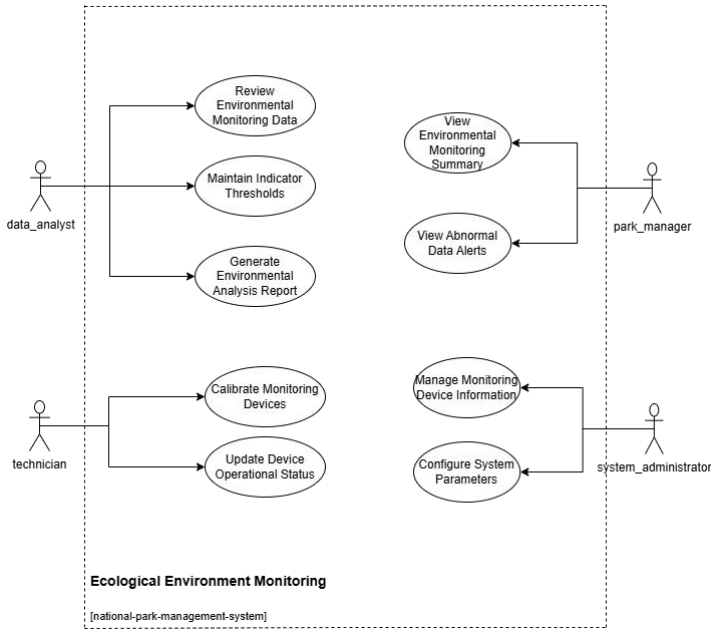
姓名	学号	班级
张万泉	230101524	物联网 23
李雪彤	231002616	物联网 23
蔡邵涵	231002605	物联网 23
赵雅萱	220701618	物联网 23
杨嘉雯	220201409	大数据 23-2

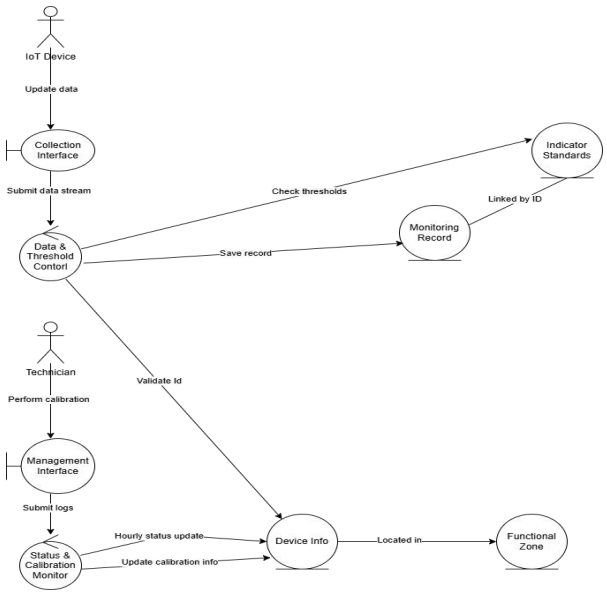
1、个人和团队

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）** 本次课程设计围绕国家公园生态环境监测业务，构建面向实际应用的数据库系统解决方案。生态环境监测业务线聚焦空气质量、水质、土壤湿度等关键生态指标，通过物联网监测设备实现环境数据的自动采集、存储、分析与预警。系统在数据库层面完成了监测指标、监测设备、环境监测数据及区域信息等核心实体的统一建模，支持多源数据的规范化管理与高效查询。通过阈值对比与异常判定机制，实现对生态环境异常状态的自动识别与预警，为公园管理人员和数据分析师提供可靠的数据支撑。整个设计过程覆盖需求分析、概念结构设计、逻辑结构设计、物理结构设计及 SQL 实现，体现了数据库系统在复杂生态管理场景中的工程应用价值。
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）** 在本次课程设计中，我主要负责生态环境监测业务线的数据库设计与实现工作。具体任务包括：完成该业务线的 UML 类图、用例图与鲁棒图设计，明确业务对象及其交互关系；编制完整的数据字典，规范实体属性、数据类型与约束；完成逻辑结构与物理结构设计，编写对应的 DDL 语句；构造测试数据并验证多表复杂查询的正确性与性能；设计并实现业务相关的视图、索引、存储过程与触发器；完成持久层代码及测试，并撰写安全与权限控制相关文档。各项任务均按计划完成，数据库结构合理、约束完整，能够满足生态环境监测业务的实际需求。

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）** 在本次课程设计中，我主要负责数据库结构规范性校验与全局集成协作工作。在各业务线完成初步设计后，基于全局数据字典与 UML 类图，对子系统表结构进行第三范式检查，统一字段命名、数据类型及主外键设计，避免跨业务线的数据冗余与结构冲突。在集成阶段，我协助梳理并补充业务线表与全局公共表之间的外键关联，统一命名与级联策略，并整理形成独立约束脚本。同时参与最终数据库集成脚本的汇总与验证，确保可一次性部署。此外，我还撰写了系统安全与风险管理报告，为数据库稳定运行与后续扩展提供保障。

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）** 在本次课程设计中，我基于用例图与鲁棒图完成了生态环境监测业务线的业务需求建模与数据库支撑分析。  用例图中，系统需支撑四类角色的核心操作：数据分析师的环境监测数据查看、监测指标阈值维护及环境分析报告生成；公园管理者的监测数据汇总与异常告警信息查看；技术人员的监测设备校准与运行状态更新；系统管理员的监测设备基础信息管理及系统参数配置。



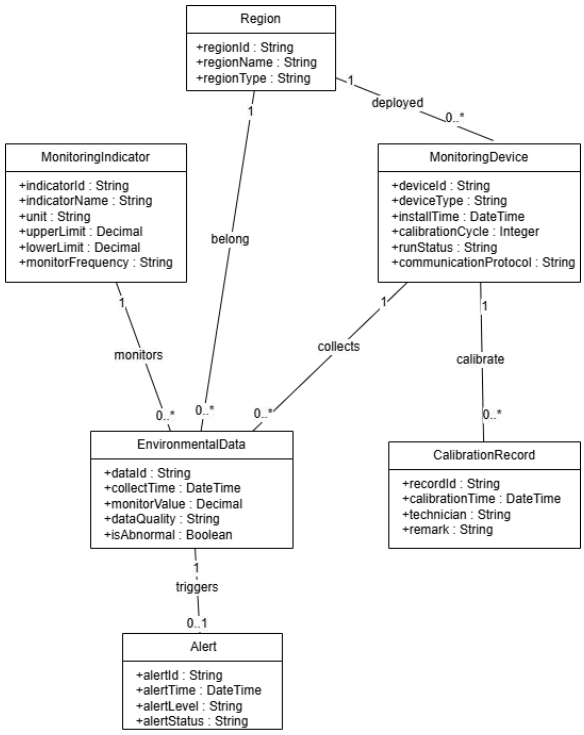
在此基础上，鲁棒图进一步刻画了业务数据的流转逻辑：物联网监测设备通过数据采集接口向系统传输环境数据流，由数据与阈值控制模块完成阈值校验并生成监测记录；技术人员通过管理接口提交设备校准与维护日志，状态监测模块同步更新设备运行状态与校准信息，并关联设备信息、功能区域及指标标准等数据实体，实现生态环境监测全流程的数据库支撑。

业务的非功能需求

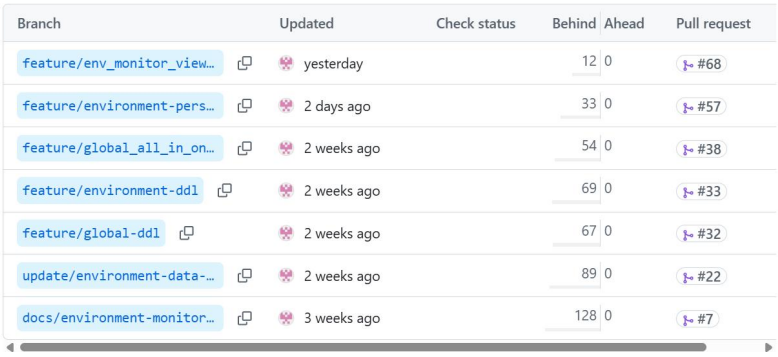
（要求：总结个人在数据库系统课程设计所负责业务的安全性、完整性需求-300 字-500 字-1.5 倍行距-5 号宋体）

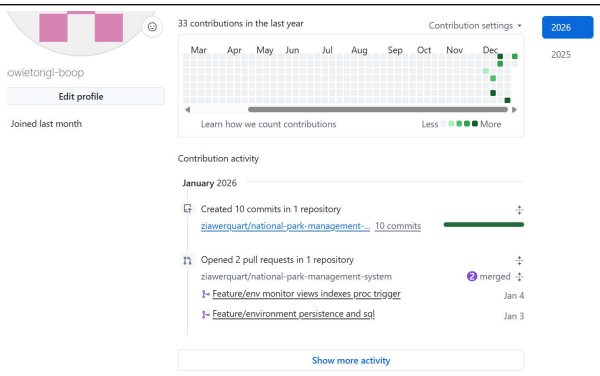
在本次数据库系统课程设计中，针对生态环境监测业务线数据来源多、更新频繁、业务关联复杂的特点，重点从数据库层面设计并落实安全性与完整性保障机制。在安全性方面，系统采用基于角色的访问控制策略（RBAC），根据数据分析师、公园管理人员、技术人员及系统管理员等不同角色划分数据访问权限，通过视图对敏感字段进行隔离，确保各角色仅能访问其业务所需的数据范围。同时，对关键业务操作接口设置权限校验，防止越权访问和非法数据修改。

在完整性方面，围绕监测指标、监测设备、功能区域与环境监测记录等核心实体，建立严格的主键与外键约束，确保监测数据与指标标准、设备信息之间的引用一致性。通过检查约束与触发器实现监测值阈值校验与异常标记，防止不符合业务规则的数据进入系统。对设备校准与状态更新过程，采用事务机制保证多表更新的原子性，避免因异常中断导致的数据不一致。上述安全性与完整性设计共同保障了生态环境监测业务数据的可靠性、可追溯性与系统运行的稳定性。

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）  在本次数据库系统课程设计中，我围绕生态环境监测业务线完成了局部 UML 类图设计，用于刻画该业务中核心数据对象及其关系，并作为数据库结构设计的重要依据。类图以监测指标、功能区域、监测设备和环境监测数据为核心，覆盖环境数据采集、设备管理、阈值判定与异常预警等关键业务需求。 通过 MonitoringIndicator 类定义监测指标的计量单位、上下限阈值及监测频率，满足对不同生态指标进行标准化管理和阈值校验的需求；Region 类用于描述国家公园功能分区，为设备部署与数据归属提供空间维度支撑。在设备与数据管理方面，MonitoringDevice 类记录设备类型、运行状态、通信协议及校准周期，支撑设备运行监控与维护管理；EnvironmentalData 类用于存储监测设备采集的实时环境数据，并通过与指标和区域的关联，实现监测值溯源与统计分析。针对设备维护与数据可靠性需求，引入 CalibrationRecord 类记录设备校准过程，保障监测数据的准确性。同时，通过 Alert 类描述异常预警信息，实现当监测数据超出指标阈值时的业务联动。 类之间的一对多关系清晰反映了实际业务逻辑，为后续数据库表结构设计、主外键约束、触发器与视图实现提供了明确的数据模型基础，确保生态环境监测业务在数据库层面的完整性、一致性与可扩展性。

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）** 在本次数据库系统课程设计中，我主要负责生态环境监测业务线的数据库设计与实现工作，涵盖逻辑结构设计、物理结构设计、核心业务 SQL 编写、索引与视图设计等内容，并与小组整体数据库架构保持一致。该业务线以环境监测指标、监测设备、监测数据、设备校准及异常告警为核心，服务于数据分析师、公园管理人员、技术人员及系统管理员等多类角色。 在逻辑结构设计阶段，我基于需求分析与 UML 类图，将生态环境监测业务抽象为 MonitoringIndicator、EnvironmentalData、MonitoringDevice、CalibrationRecord 与 Alert 等核心关系模式。各表之间通过明确的一对多关系构建数据链路：监测指标与环境数据之间建立“一对多”关系，监测设备与环境数据、校准记录之间建立“一对多”关系，环境监测数据与异常告警之间建立“一对零或一”关系，同时通过 Region 表引入空间维度，实现监测数据的区域归属管理。上述关系模式均满足第三范式要求，避免了冗余数据存储。 在物理结构设计方面，我根据业务访问特征合理选取数据类型与约束条件，使用 InnoDB 存储引擎以支持事务与外键约束。通过主键与外键约束，确保环境监测数据必须依附于合法的监测指标、设备及区域，保障数据引用完整性。同时，针对监测频率、数据质量、运行状态等枚举型字段，统一使用 ENUM 类型以规范取值范围，减少非法数据写入风险。 在业务相关 SQL 实现方面，我围绕生态环境监测的典型业务场景设计并验证了多条复杂查询语句，包括异常环境数据与告警信息联合查询、按区域与指标统计异常发生次数、获取设备最新监测数据以及检测校准超期设备等。针对同一业务需求，分别采用 JOIN、子查询、CTE 以及窗口函数等不同实现方式，并结合 EXPLAIN ANALYZE 对执行计划进行对比分析，为后续性能优化提供依据。 在索引设计方面，我结合高频查询条件，在 EnvironmentalData 表上设计了基于“区域 + 时间”和“指标 + 时间”的联合索引，用于提升历史数据回溯、趋势分析与实时监控场景下的查询效率；在 MonitoringDevice 表上设计了“区域 + 运行状态”索引，以支持设备运维与状态筛选；在 Alert 表上设计了“告警时间 + 告警等级”索引，便于管理人员进行告警分析与应急决策。

	在视图设计方面，我围绕不同角色的数据需求，设计了实时监测视图、指标统计分析视图、设备运行状态视图及异常告警管理视图。各视图通过多表关联，对底层复杂结构进行封装，既提升了查询效率与可读性，又遵循最小权限原则，为后续系统安全控制与业务扩展提供了良好基础。整体设计内容与本人在答辩中的任务分工和技术说明保持一致，完整支撑了生态环境监测业务线的数据库实现需求。
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）** 在本次数据库系统课程设计中，小组项目基于 GitHub 进行统一的版本管理与协作开发，整体仓库采用分层目录结构，将需求分析文档、数据库脚本、程序代码及测试结果进行明确拆分，提升了项目的可维护性与协作效率。我个人负责的核心内容集中在生态环境监测业务线，对应的数据库脚本与代码均以 20_environment 作为统一前缀，主要分布于 sql/DDL/20_environment.sql、sql /DML/ 20_environment_seed.sql、sql/VIEW、sql/INDEX、sql/PROC、sql/TRIGGER 以及 src/dao/environment 等目录下，形成了从表结构设计、约束与索引定义，到视图、存储过程、触发器，再到持久层封装与测试的完整实现链路。  在工程管理过程中，所有涉及数据库结构调整、跨表外键补充、复杂 SQL 查询优化以及存储过程、触发器编写的工作，均通过独立分支完成，并以 Pull Request 的形式合并至主分支，确保关键数据库变更具备良好的可审查性与可回滚性。提交过程中严格遵循“单一职责”原则，每一次提交均对应明确的业务功能或 SQL 文件，避免多任务混合提交带来的维护风险。



在开发阶段，我结合单元测试对生态环境监测业务的 DAO 层进行验证，测试代码统一放置于 tests/environment 目录下，通过测试手段保障核心 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，实现了个人负责业务从设计、实现到优化的全过程追踪，为系统稳定迭代与最终课程答辩提供了可靠的工程支撑。

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）

在本次数据库系统课程设计中，围绕本人负责的生态环境监测业务线，在系统实现与运行过程中重点识别并分析了多类潜在风险，并针对不同风险类型设计了相应的控制与缓解手段，以保障系统的安全性、稳定性与可扩展性。

首先，在数据一致性与完整性风险方面，生态环境监测业务涉及监测指标、设备、区域、采集数据、告警信息等多类核心实体，表之间关联关系复杂，若缺乏约束容易产生脏数据或孤立记录。对此，在数据库层面通过主键、外键约束以及合理的级联策略来保证跨表数据一致性，并结合枚举类型与取值范围约束，降低非法数据写入的风险。

其次，在系统性能与并发访问风险方面，环境监测数据具有采集频率高、数据量持续增长的特点，若查询设计不合理，容易在统计分析和告警查询场景中引发性能瓶颈。针对该问题，通过对高频访问字段（如区域、指标、采集时间、告警时间等）建立复合索引，并采用视图与窗口函数、CTE 等方式优化复杂查询逻辑，减少重复扫描与不必要的 JOIN 操作，从而提升系统在高并发场景下的响应能力。

最后，在安全与误操作风险方面，通过角色区分（如运维人员、管理人员、技术人员）和 DAO 层封装，限制直接操作底层数据表的范围，降低人为误操作对核心数据的影响。结合版本控制工具对数据库脚本变更进行审查与回滚管理，进一步提升系统在实际运行与迭代过程中的风险可控性。

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）** 在本人负责的生态环境监测业务中，针对环境数据采集频繁、异常判定逻辑固定且对实时性要求较高的特点，综合运用存储过程与触发器对系统运维与性能优化进行了针对性设计，从而在数据库层面实现业务规则的自动化与稳定化。 在运维层面，存储过程 sp_env_auto_alert 被用于封装环境监测数据的异常判定与告警生成逻辑。该过程通过统一读取监测指标的上下限，并与实时采集的监测值进行比较，实现异常识别规则的集中管理。相较于将异常判断逻辑分散在应用层，该方式有效降低了业务代码复杂度，同时确保不同系统模块在告警生成规则上的一致性。在实际运维中，当监测指标阈值发生调整时，仅需维护指标表数据即可，无需修改应用程序或多处 SQL，从而显著降低了系统调整成本和运维风险。 在性能与优化方面，存储过程内部通过对告警记录进行存在性校验，避免了重复告警的产生，减少了告警表的冗余数据规模，有利于后续告警统计与查询性能的稳定。此外，将异常标识字段直接维护在环境数据表中，使异常数据筛选能够通过索引快速完成，为实时监控视图和统计分析提供了良好的查询基础。 在数据规范与一致性控制方面，触发器 trg_envdata_fill_time 对环境监测数据的采集时间进行自动填充，避免因上游设备或接口异常导致关键时间字段缺失的问题。该触发器将基础数据修正逻辑前移至数据库层，保证所有进入系统的数据在结构和语义上的完整性，减少人工补救和异常排查的工作量。 总体来看，通过将核心业务规则下沉至存储过程与触发器中，实现了环境监测业务在异常识别、告警生成与数据规范化方面的自动化运维，不仅提升了系统的运行稳定性，也为后续业务扩展和性能优化提供了可持续的数据库层支撑。