数据库系统课程设计报告

姓名	赵雅萱	班级	物联网 23
学号	220701618	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
杨嘉雯	220201409	大数据

1、个人和团队

课程设计概述

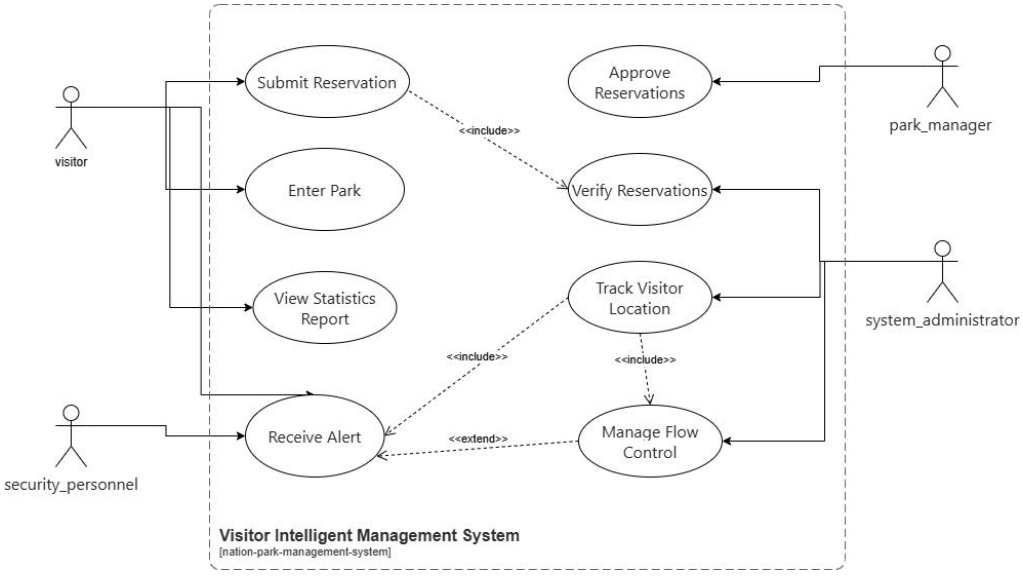
（要求：总结课程设计的业务内容-200 字-300 字-1.5 倍行距-5 号宋体）

游客智能管理业务线以“预约—审批—核验—入园—实时追踪—客流管理—统计”为核心流程，串联游客、公园管理、安保等环节，实现全流程智能化管控。

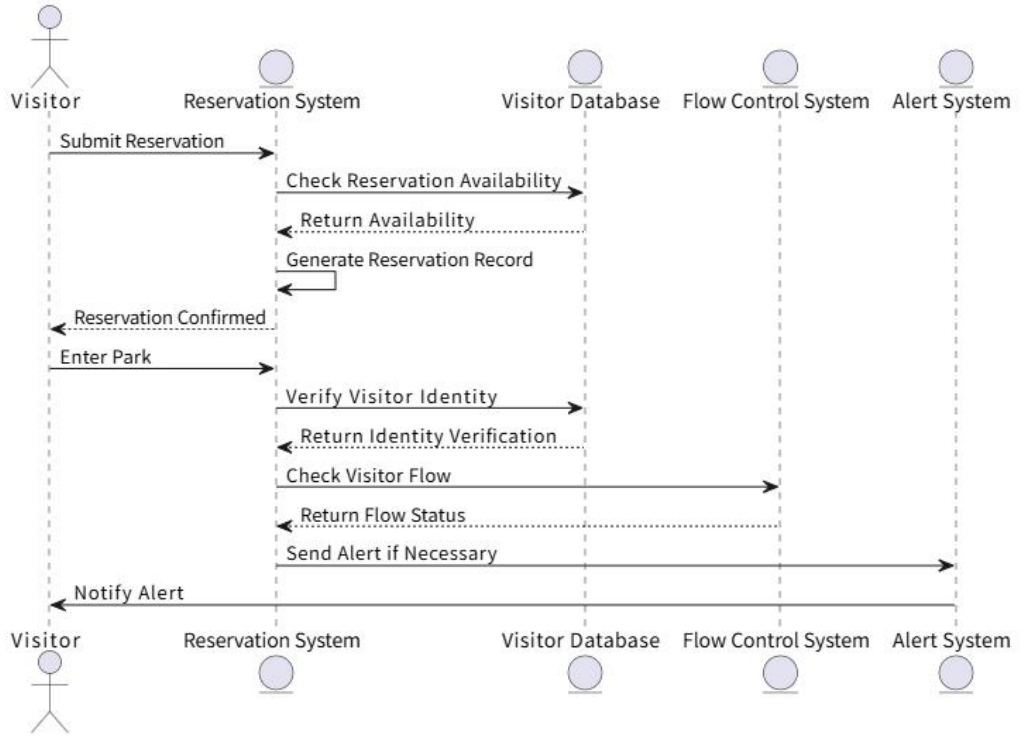
核心参与者为游客（visitor）、公园管理员（park_manager）、安保人员（security_personnel）与系统管理员（system_administrator）。系统包含核心对象：Reservation（预约记录）、Visitor（游客信息）、ParkArea（园区区域）、VisitorLocation（实时位置）、StatisticsReport（统计报表）、Alert（告警信息）、FlowControlPolicy（客流策略）。

游客执行用例“Submit Reservation”提交预约申请，系统自动校验后置状态为“pending”；公园管理员执行用例“Approve Reservation”审批更新为“confirmed”；游客入园时系统执行用例“Verify Reservation”核验预约并记录时间；系统实时追踪游客位置；安保人员基于客流执行用例“Manage Flow Control”并触发用例“Receive Alert”；系统管理员执行用例“Generate Statistics Report”生成客流统计报表。结构层面，Reservation 关联 Visitor 与 ParkArea，VisitorLocation 关联 Visitor 与 ParkArea，FlowControlPolicy 关联 ParkArea，StatisticsReport 聚合数据支撑“游客—预约—区域—行为”的跨实体查询分析。

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）** 本人负责游客智能管理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，并且统一检查字段命名与类型格式和整合全局数据字典（Issue #10）。之后完成了 Visitor、Reservation、VisitorTrajectory、FlowControl 等表的 DDL 设计与说明（Issue #28）。实现游客智能管理模块编写测试数据、复杂 SQL，并完成 DAO 持久层代码与单元测试（Issue #43）。后期又完成视图、索引、存储过程及触发器设计（Issue #51），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 #63），并且协助完成小组报告整合 + 答辩 PPT 制作与讨论（Issue #55）。
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，我与张万泉同学一起审查全局 UML 图的设计（Issue #18），确定实体关系以及结构划分的正确性。参与了三次小组会议，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。技术协作方面，在合并全局数据字典的过程中（Issue #10），与 李雪彤、蔡邵涵、杨嘉雯同学就数据字典的结构划分、实体关系进行讨论；同时在问题讨论中，参与相关 Issue 的解决（Issue #18、#19）。最后，我协助杨嘉雯同学完成了小组报告整合 + 答辩 PPT 制作与讨论（Issue #55）。
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）** 用例图：



鲁棒图：



游客智能管理业务线围绕“游客预约、入园核验、实时追踪与客流管控”构建完整业务流程，用例图明确了游客、公园管理员、安保人员及系统管理员四类参与者的职责分工。游客通过 Submit Reservation 用例提交预约申请，系统自动包含 Verify Reservations 用例，对预约信息进行完整性校验并生成待审批记录；公园管理员执行 Approve Reservations 用例完成审批，更新预约状态为“已确认”；游客在 Enter Park 时触发 Verify Reservations 核验身份与预约有效性，并联动 Track Visitor Location 实时采集位置数据；系统根据区域实时人流执行 Manage Flow Control，

	当客流超阈值时，该用例扩展触发 Receive Alert，通知安保人员干预；系统管理员可执行 View Statistics Report，基于预约、入园和轨迹数据生成客流统计报表。 鲁棒图从对象交互层面细化了上述流程的实现机制，清晰划分了 Boundary（如 Reservation System)、Controller(如 Flow Control System)与 Entity(如 Visitor Database）的职责边界。重点体现了预约记录在“pending → confirmed → completed”状态间的流转，以及游客轨迹数据在进入园区后持续更新的过程。同时，Manage Flow Control 作为核心控制逻辑，协调 Flow Control System 与 Alert System，实现动态限流与告警联动，保障园区运行安全与秩序。
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）** 游客智能管理业务线的非功能需求聚焦于安全性与完整性两大核心维度。 安全性需求方面，系统需实现多角色权限隔离，游客仅可访问自身预约信息，公园管理员仅能审批管辖区域预约，安保人员可查看客流告警但无修改权限，系统管理员掌握全局配置权限；敏感数据（游客身份证、手机号）采用 AES 加密存储，传输层启用 HTTPS；所有操作（预约提交、审批、客流管控）记录审计日志，包含操作人、时间、内容，便于追溯；DAO 层采用参数化查询（参考 base_dao.py 的 execute 方法），避免 SQL 注入风险。 完整性需求 涵盖多层约束：实体完整性要求各表主键非空且唯一，如 Visitor 的 visitor_id、Reservation 的 reservation_id 均设为主键；参照完整性通过外键约束实现（参考 90_constraints.sql），如 Reservation 的 visitor_id 关联 User(user_id)，配置 ON UPDATE CASCADE、ON DELETE SET NULL，保证主键更新 / 删除时外键同步调整；域完整性限定字段取值，如预约状态仅允许 pending/confirmed/cancelled/completed，客流阈值为正整数，手机号为 11 位字符；业务规则完整性要求预约时间不早于当前时间、同一游客同一时段不可重复预约同一区域，入园核验需匹配预约信息与身份信息，避免无效数据录入。

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）** 游客管理业务的局部 E-R 图围绕 7 个核心实体构建，各实体属性与关系均满足 BCNF 范式（无部分 / 传递依赖）。核心实体及属性：①Visitor（游客）：visitor_id（主键）、name、id_card、phone、register_time；②Reservation（预约记录）：reservation_id（主键）、visitor_id（外键）、park_area_id（外键）、apply_time、status、approve_time、approver_id；③ParkArea（园区区域）：park_area_id（主键）、area_name、max_capacity、current_capacity；④VisitorLocation（游客位置）：location_id（主键）、visitor_id（外键）、park_area_id（外键）、location_time、longitude、latitude；⑤FlowControlPolicy（客流策略）：policy_id（主键）、park_area_id（外键）、threshold、control_strategy、effective_time；⑥Alert（告警信息）：alert_id（主键）、policy_id（外键）、alert_time、alert_level、processed_status；⑦User（系统用户）：user_id（主键）、role、name、permission。实体间关系：Visitor 与 Reservation 为一对多（1 个游客可提交多条预约，1 条预约归属 1 个游客）；ParkArea 与 Reservation、FlowControlPolicy 均为一对多（1 个区域对应多条预约 / 多条客流策略）；Visitor 与 VisitorLocation 为一对多（1 个游客对应多条位置记录）；FlowControlPolicy 与 Alert 为一对多（1 个策略触发多条告警）。E-R 图清晰体现 “游客 - 预约 - 区域 - 客流 - 告警” 的业务链路，为后续关系模式转换提供核心依据。
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）** 本次课程设计围绕游客智能管理业务线完成全流程设计与开发，核心涵盖逻辑结构、物理结构、SQL 代码、索引与视图设计四大模块。 逻辑结构设计：基于 E-R 图完成关系模式转换，确保所有实体满足 BCNF 范式：定义 Visitor、Reservation 等核心表的关系模式，明确主键、外键及字段约束，如 Visitor 表中 id_card 设为唯一约束，Reservation 表中 status 限定为枚举类型（pending/confirmed 等）；梳理实体间参照关系，如 Reservation 的 visitor_id 关联 User 表，保证数据逻辑一致性，避免冗余与异常更新。 物理结构设计：适配 MySQL 数据库特性：选用 InnoDB 存储引擎（支持事务与外键）；优化字段类型，如 visitor_id 用 CHAR (20)（固定长度提升查询效率）、经纬度用 DECIMAL (10,6)（高精度定位）、客流阈值用 INT（符合业务取值）；配置外键约束规则（ON UPDATE CASCADE、ON DELETE SET NULL），参考 90_constraints.sql 实现，保证主键变更时外键同步调整，避免数据丢失。 业务相关 SQL 代码：覆盖 DDL、DML、复杂查询与 DAO 层实现：①DDL 编写核心表创建语句，如 Visitor 表包含 visitor_id、name 等字段，Reservation 表关联游客与区域；②DML 实现参数化 CRUD，，编写 VisitorDAO 的 insert/update/delete 方法，避免 SQL 注入；③复杂查询采用 CTE 预过滤，如异常客流查询：通过 CTE 筛选超容区域，再关联预约与游客信息；④单元测试编写预约状态更新、异常 ID 操作等测试用例，覆盖正常 / 异常场景。 索引和视图设计：提升查询效率：①索引设计：为 Reservation 的 park_area_id+status 创建复合索引（高频查询区域已确认预约），为 Visitor 的 id_card 创建唯一索引；②视图设计：创建 v_visitor_reservation_detail 视图，整合游客、预约、区域信息，简化业务端查询，避免重复 JOIN 操作。 开发过程中严格遵循团队 Issue 规范（参考 CONTRIBUTING.md），每个模块对应唯

	一 task Issue（如 #28 完成 DDL 设计、#43 实现 DAO 层与单元测试），确保开发内容与答辩分工一致，同时通过代码审查保证逻辑与物理设计的合理性。
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）** 本次游客管理业务线开发全程基于 GitHub 完成工程管理，严格遵循团队协作规范。核心管理方式：①任务管理：通过 Issue 划分任务，每个任务对应唯一 task Issue（如 #10 整合数据字典、#28 设计 DDL、#43 开发 DAO 层），明确完成标准、截止时间，Issue 为任务执行的唯一依据；②版本控制：创建 feature/visitor-management 分支开发个人模块，避免直接修改主分支，完成后提交 PR，由组长张万泉审查后合并，保证主分支代码稳定；③协作沟通：通过 Issue 评论区讨论技术问题（如 #18 审查 UML 图、#19 优化外键约束），PR 关联对应 Issue 便于追溯；④文档管理：将数据库设计文档、SQL 代码、测试用例按仓库目录结构（sql/DDL、src/test 等）提交，保证版本一致；⑤进度跟踪：通过 GitHub Projects 看板可视化任务状态（待办 / 进行中 / 已完成），定期更新 Issue 进度，确保个人任务按计划推进。
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）** 游客管理系统开发与运行阶段存在多类潜在风险，需针对性制定控制手段：①数据安全风险：游客身份证、手机号等敏感信息泄露，或预约记录、位置数据丢失。控制手段：敏感数据加密存储（AES 算法）、传输层启用 HTTPS；定期执行全量 + 增量备份，备份文件异地存储；设置外键约束保证数据参照完整性，避免删除主键导致关联数据失效。②功能风险：预约状态更新异常（如重复审批）、高并发下查询性能低。控制手段：编写单元测试覆盖核心场景（如过期预约自动取消、异常 ID 更新返回 False），模拟高并发压力测试；优化索引设计（如复合索引 idx_reservation_area_status），提升查询效率；DAO 层采用参数化查询，避免 SQL 注入。③操作风险：误删预约记录、权限滥用（非授权人员修改客流策略）。控制手段：实现 RBAC 权限控制，按角色（游客 / 管理员 / 安保）分配操作权限；所有增删改操作记录审计日志，包含操作人、时间、内容；设置数据修改二次确认机制（如删除预约需确认）；定期审查权限配置，回收闲置账号权限。④运维风险：触发器 / 存储过程执行异常。控制手段：监控触发器执行日志，定期测试存储过程逻辑，避免异常触发告警或数据统计错误。

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）** 基于存储过程与触发器设计，实现游客管理业务的高效运维与性能优化：①存储过程优化运维效率：编写 sp_statistics_flow 存储过程，自动统计指定时段各区域入园人数、预约数、客流峰值，替代人工编写重复 SQL，每日通过定时任务执行生成报表；编写 sp_update_expired_reservation 存储过程，凌晨自动将过期未入园的预约状态更新为 cancelled，无需人工干预，降低运维成本。②触发器保障数据实时性与业务规则：设计 tr_flow_alert 触发器，当区域客流超阈值时自动插入告警记录，实时通知安保人员；设计 tr_update_area_capacity 触发器，游客位置更新时同步调整区域当前客流，保证数据实时性；触发器无需人工调用，实现业务规则的自动化执行。③运维优化手段：定期分析 SQL 执行计划，删除低效索引、新增高频查询索引；对 VisitorLocation 大表按 location_time 分区，提升历史位置数据查询速度；清理超过 3 个月的位置记录，减少数据量；监控 Alert 表的未处理告警，确保客流异常及时响应；定期测试存储过程与触发器逻辑，修复执行异常问题；通过参数化查询与索引优化，降低数据库 CPU 与 IO 消耗，提升系统整体响应速度。