Software Laboratory Management System

Database Schema Design

Presented by:

Group No. : 19

1. ALOK RANJAN(2022CSB091)
2. RANVEER KUMAR(2022CSB092)
3. KARAN KUMAR(2022CSB093)
4. RAJESH KUMAR(2022CSB094)

Table of Contents

Project Overview

Problem

The Software Laboratory Management System facilitates students, lab-in-charge, faculty members, and lab assistants in managing lab resources, scheduling, issue reporting, and monitoring.

Assignment

Scope

Data modelling and designing database schema for the respective modules following requirements analysis from Assignment-5.

System Requirements - Overview

Functional Requirements

- User Management
- Lab Scheduling & Reservations
- Equipment & Resource Management
- Issue Reporting & Maintenance
- Monitoring & Reports

Non-Functional Requirements

- Performance: Support 100+ concurrent users
- Security: Role-based access control
- Usability: User-friendly interfaces
- Reliability: 99.9% system availability

Entity-Relationship Overview

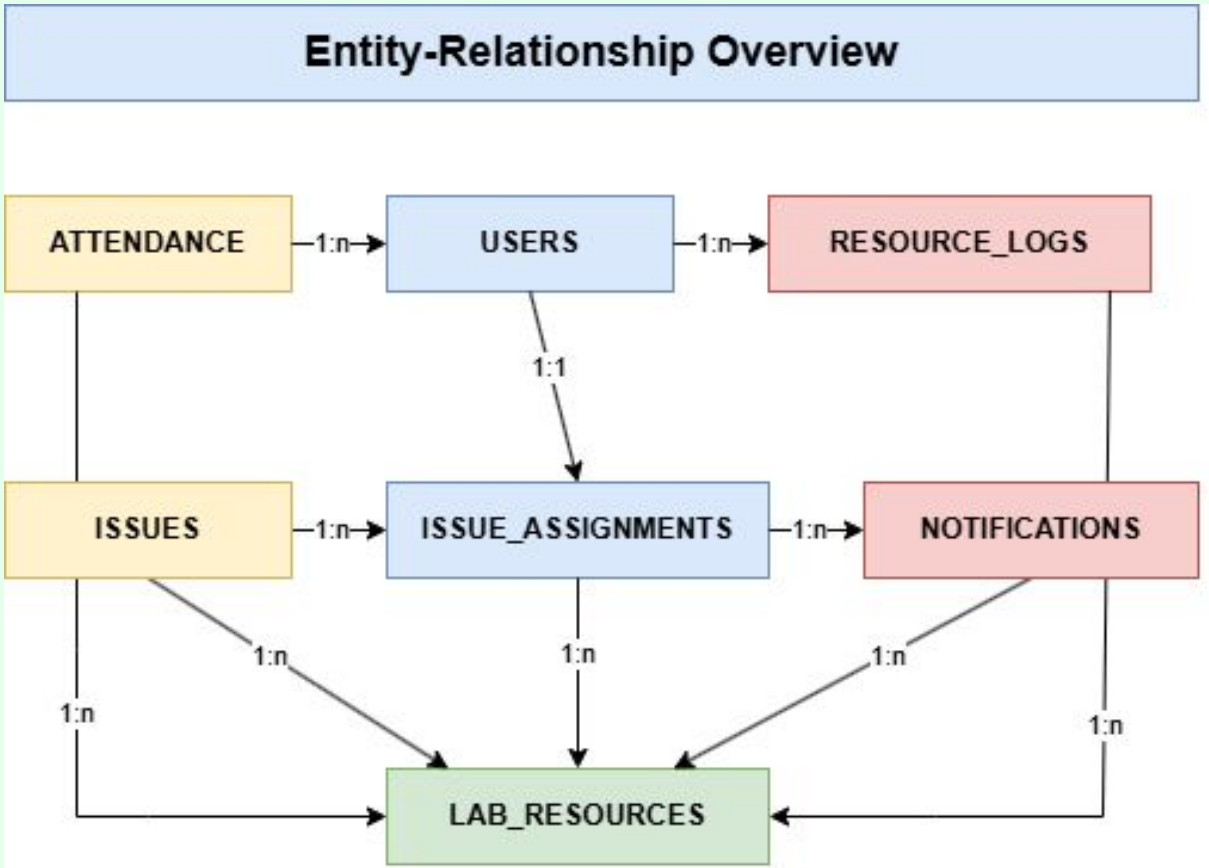

Figure: Entity-Relationship Diagram of Software Laboratory Management System

Color-coded by functional domain

Arrows indicate relationship direction and cardinality

Figure: Entity-Relationship Diagram of Software Laboratory Management System

Complete Database Schema

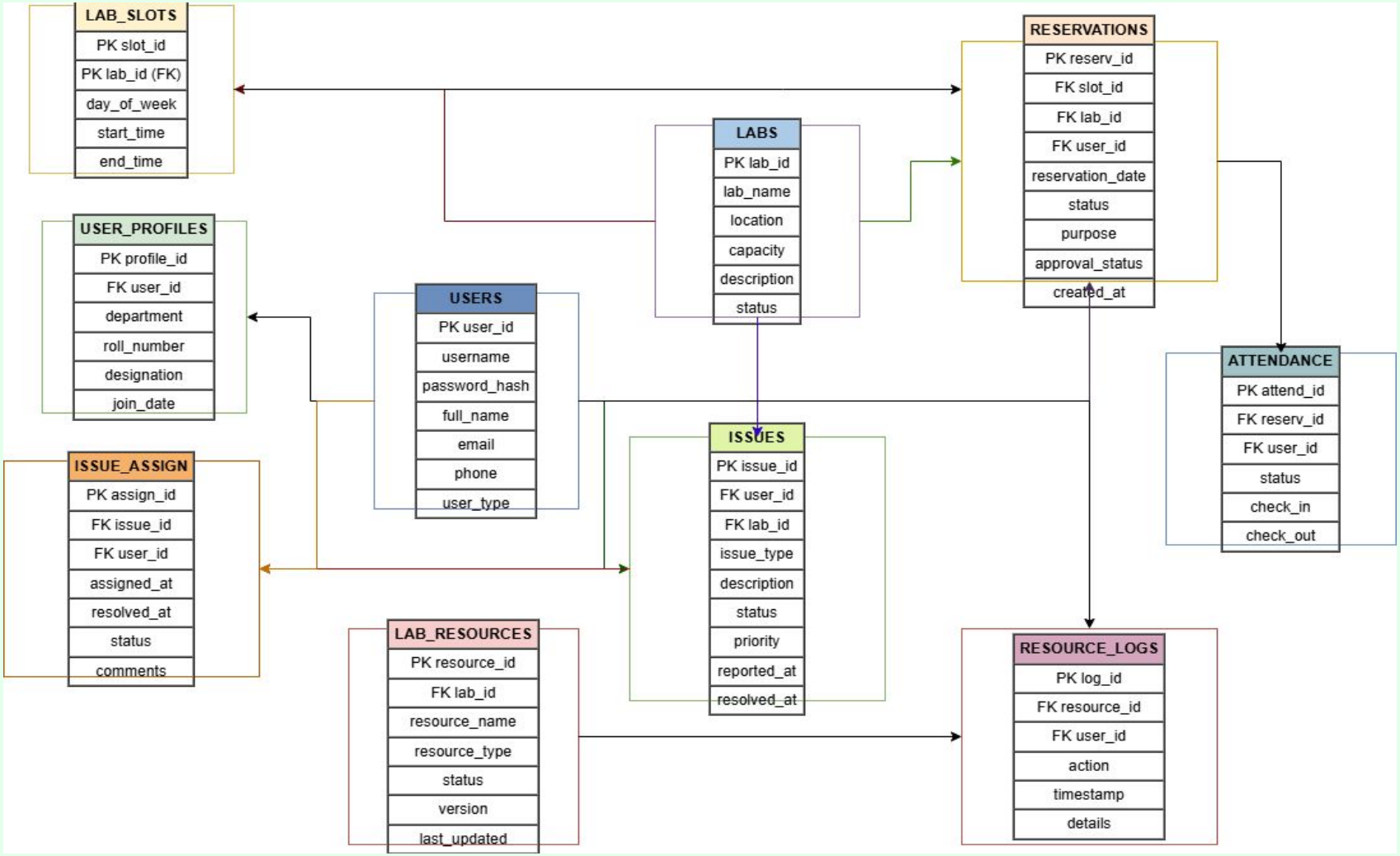

Figure: Complete Entity-Relationship Diagram with All Tables and Relationships

Core Tables - User Management

USERS

- user id (PK)
- username
- password hash
- full name email
- phone user
- type is
- active
-

USER_PROFILES

- profile id (PK)
- user id (FK)
- department
- roll number (for students)
- designation (for staff) join
- date

Detailed

Design Rationale

Centralized user information with role-specific profiles Security through password hashing and account status tracking Support for different user types with minimal data redundancy

Core Tables - Lab Management

LABS

- lab_id (PK)
- lab_name
- location
- capacity
- description
- status

LAB_SLOTS

- slot_id (PK)
- lab_id (FK)
- day_of_week
- start_time end time

LAB_RESOURCES

- resource_id (PK)
- lab_id (FK)
- resource_name
- resource_type
- status version

Design

Design Rationale

Hierarchical structure: Labs → Slots → Resources Enables fine-grained scheduling and resource tracking Status tracking for maintenance planning

Core Tables - Scheduling & Attendance

RESERVATIONS

- reservation id (PK)
- slot id (FK)
- lab id (FK)
- user id (FK)
- reservation date
- status
- purpose approval
- status approved by (FK)

ATTENDANCE

- attendance id (PK)
- reservation id (FK)
- user id (FK)
- status
- check in
- check out

Design

Design Rationale

Approval workflow for lab bookings

Tracking attendance for each reservation

Core Tables - Issue Management

ISSUES

- issue id (PK)
- user id (FK) lab id (FK)
- resource id (FK)
- issue type
- description status
- priority
- reported at
- resolved at

ISSUE_ASSIGNMENTS

- assignment id (PK)
- issue id (FK) assigned to (FK) assigned by (FK) assigned at
- status resolution
- notes resolved at

Design Rationale

Core Tables - Logging & Notifications

RESOURCE_LOGS

- log id (PK)
- resource id (FK)
- user id (FK) action
- timestamp details

NOTIFICATIONS

- notification id (PK)
- user id (FK)
- title message
- is read
- created at

Design Rationale

1. Comprehensive audit trails for resource
2. usage Event-based notification system
3. Support for system-generated alerts

ri

SQL Schema Example

```
-- Example table creation CREATE
TABLE Labs (
        lab_id INTEGER PRIMARY          KEY AUTOINCREMENT,
        lab_name     VARCHAR (100)   NOT   NULL ,
        location     VARCHAR (100)   NOT   NULL ,
        capacity     INTEGER NOT NULL,
        description TEXT,
        status ENUM ('active', 'maintenance', 'inactive') DEFAULT 'active',
        created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
        last_updated TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

SQL Schema Example

```
-- Example trigger to prevent double bookings CREATE TRIGGER
prevent_double_booking
BEFORE INSERT ON Reservations FOR
EACH ROW
BEGIN
    SELECT CASE
        WHEN EXISTS (
            SELECT 1 FROM Reservations WHERE
            slot_id = NEW.slot_id AND lab_id =
            NEW.lab_id
            AND reservation_date = NEW.
                reservation_date
            AND status IN ('approved','pending')
        )
        THEN RAISE(ABORT, 'This lab slot is already booked')
    END;
```

Database Optimization Techniques

Indexing Strategy

Primary and foreign key indexes for all tables

Additional indexes on frequently queried columns:

- username in USERS
- reservation date in RESERVATIONS
- status in ISSUES
- resource type in LAB RESOURCES

Composite indexes for combined search conditions

Views for Common Queries

1. Available Lab Slots - For checking
2. availability Pending Approvals - For lab
3. in-charge workflow Unresolved Issues -
4. For maintenance monitoring Lab Usage

Database Automation with Triggers

Data Integrity Triggers

- Prevent double booking of lab slots
- Validate reservation times against lab schedules
- Ensure proper issue status transitions
- Maintain data consistency across tables

Notification Triggers

- Auto-notify on reservation approval/rejection
- Alert users when their reported issues are resolved
- Notify lab assistants on new issue assignments
- Resource status change notifications

Benefits

1. Automated workflow with minimal manual intervention
2. Consistent application of business rules

Role-Based Access Implementation

Students

- View lab schedules
- Book available slots
- Report issues
- View their attendanc

Faculty

- Reserve labs
- Mark attendance
- Report issues
- View resourc

Lab Assistants

- Manage resources
- Resolve issues
- Update inventory
- student s

Lab-in-Charge

- Approve bookings
- Manage schedules
- Assign tasks
- Generate

Database Security Implementation

1. Role determination from user type field
2. View-based access control for complex
3. permissions Stored procedures with role
4. Validation Audit logging of all critical operations

Key Features & Benefits

Features

- Comprehensive user role management
- Flexible lab scheduling system Detailed resource tracking
- Complete issue management workflow
- Integrated notification system Audit logging capabilities

Benefits

- Efficient resource utilization
- Streamlined administrative workflows
- Improved issue resolution times
- Enhanced transparency and accountability
- Data-driven decision making
- Scalable architecture

Summary

Database Schema

A comprehensive database design for Software Laboratory Management System with:

1. 11 core tables with well-defined relationships
2. Carefully designed foreign key constraints for data integrity
3. Strategic indexing for performance optimization
4. Automated triggers for business rule enforcement
5. Views for simplified access to common data sets

Next Steps

1. Implement database schema in chosen RDBMS
2. Develop application layer with appropriate access control
3. Create UI components for different user roles
4. Implement reporting and analytics functionality

Thank You