University Examination Management System – Database Schema Documentation

Current date reference: February 05, 2026
Version: 1.0 (based on provided CREATE TABLE statements)
Purpose: This document explains the full database schema, table relationships, data flow, and practical usage/maintenance guidelines for both non-technical users (exam committee, admin staff) and technical users (developers, DB admins).

1. Overview – What the System Does

This database supports **complete exam lifecycle** in a university (especially Computer Science / Technology programs):

- Define exams (subjects, dates, times, target years/programs)
- Manage rooms (capacity, type, seating layout)
- Assign exams to rooms + mix student groups from different years/programs
- Assign individual students to exact seats
- Assign teachers/invigilators to supervise specific rooms
- Track students and teachers (who they are, their status/workload)

Goal: Organized, fair, space-efficient exams with minimal conflicts and clear records.

2. The 7 Core Tables

Table Name	Real-world name	Main job (simple words)	Key fields summary
`student`	Student registry	Who are all the students? Year, major, retake?	student_id, student_number (unique), name, year_level, major, sem, specialization, retake
`teacher`	Teacher / invigilator list	Who can supervise? Rank, department, current duties count	teacher_id, name, rank (⚠ unique – probably mistake), department, total_periods_assigned
`room`	Room catalog	Which rooms exist? Size, type, chair layout	room_id, room_number (unique), capacity, room_type, rows, cols, is_available
`exam`	Exam timetable	Which subject/exam happens when? For whom?	exam_id, subject_code, exam_name, exam_date, session, semester, academic_year, year_level, program, specialization, start_time, end_time, day_of_week
`exam_room`	Exam-to-room assignment + group mixing	Which exam in which room? Which years/programs share?	exam_room_id, exam_id, room_id, assigned_capacity, year_level_primary/secondary, sem_primary/secondary, program_primary/secondary, students_primary/secondary
`seating_assignment`	Student seat map	Which student sits in which exact chair?	seating_id, exam_room_id, student_id, seat_number, row_number, column_number
`teacher_assignment`	Invigilator duty roster	Which teacher watches which room (and when)?	assignment_id, exam_room_id, teacher_id, role, shift_start, shift_end

3. How the Tables Connect (Relationships)

student ←——— seating_assignment ———→ exam_room ———→ exam teacher ←——— teacher_assignment ———┘ │ room ←————————————————————————————┘ ↑ └——— exam_room also links to room text- **Master data** (rarely change): `student`, `teacher`, `room`

- **Core planning**: `exam` → `exam_room`
- **Detailed execution**: `seating_assignment` + `teacher_assignment`

4. Step-by-Step Lifecycle – How Data Moves Through the System

Phase 0 – Setup (done once per semester/year)
Add/update students, teachers, rooms.

Phase 1 – Create exam timetable
Exam committee enters all exams into `exam` table
Example: exam_id=7 → "Programming in C++", 2026-10-01, Morning, First Year CST

Phase 2 – **Allocate exams to rooms & mix groups**
Decide rooms + combine different years (per policy: mix years when possible)
→ Insert into `exam_room` using primary + secondary fields
Example: exam 7 in room A-101 with 18 First Year CST (primary) + 11 First Year CT (secondary)

Phase 3 – **Assign individual student seats**
Generate seating plan (manual or script)
→ Insert into `seating_assignment` (one row per student per exam)
Example: student 1789 → row 3, column 12 in that exam_room

Phase 4 – **Assign invigilators/teachers**
Choose supervisors (balance workload via `total_periods_assigned`)
→ Insert into `teacher_assignment` (1–3 per exam_room)
Example: U Myo Myo as Chief Invigilator 08:30–12:45

Phase 5 – **Exam day & after**
Use reports: room usage, student locations, teacher duties, attendance.

5. Important Business Rules & Policies (from discussion)

- **Room sharing policy**: Different years should share rooms when groups are small (e.g. 15 Year 1 + 12 Year 2) → use primary/secondary fields often
- Secondary group is **optional** but **encouraged** for optimization
- Max assigned_capacity per room usually ~36 (hard-coded check)
- Teachers can supervise multiple sessions → track with `total_periods_assigned`
- No two teachers can have same rank (UNIQUE constraint) → **probably a mistake**, remove it

6. Common Reports / Queries (examples)

- **Where is student X sitting for exam Y?**
  Join `seating_assignment` → `exam_room` → `room`
- **Who supervises room Z?**
  Join `teacher_assignment` → `teacher`
- **Room usage on a date**
  Join `exam` → `exam_room` → `room`
- **Teacher workload**
  COUNT(*) from `teacher_assignment` + current `total_periods_assigned`

7. Maintenance & Improvement Suggestions

Non-technical:

- Enter master data (students/teachers/rooms) **early**
- Freeze seating & teacher assignments close to exam date
- Archive old semesters instead of deleting
- Use reports to balance teacher duties

Technical:

- Remove `UNIQUE` on `teacher.rank` (many teachers have same rank)
- Add `NOT NULL` on important FKs (exam_id, room_id, etc.)
- Add unique/exclusion constraint on room bookings (prevent overlap)
- Create views for workload, mixed-year rooms, room utilization
- Consider trigger to auto-update `teacher.total_periods_assigned`
- Possible new tables: exam_status, building, invigilator_log, student_enrollment

Prepared by Grok – feel free to extend with actual data examples or ER diagram.

Last updated: February 2026