LMS Database Schema Documentation

Overview

This document describes the PostgreSQL database schema for the Learning Management System (LMS) application. The schema is designed to support a comprehensive educational platform with role-based access control, classroom management, content delivery, and progress tracking.

Schema Structure

The database is organized under the `lms` schema namespace and includes the following main components:

- **User Management**: Users, roles, and permissions
- **Educational Entities**: Students, parents, tutors
- **Academic Structure**: Programs, classrooms, class instances
- **Content Management**: Materials, assignments, and progress tracking
- **Attendance and Reporting**: Session attendance and progress analytics

Core Tables

1. Users Table (`lms.users`)

The central user table that stores basic information for all system users regardless of their role.

Key Fields:

- `user_id` (UUID, PK): Unique identifier
- `email` (VARCHAR, UNIQUE): User's email address (used for login)
- `password_hash` (VARCHAR): Encrypted password
- `first_name`, `last_name` (VARCHAR): User's full name
- `phone`, `address`, `city`, `state`, `country`: Contact information
- `is_active`, `is_verified` (BOOLEAN): Account status flags

Relationships:

- One-to-Many with `students`, `parents`, `tutors`
- Many-to-Many with `roles` through `user_roles`

2. Roles Table (`lms.roles`)

Defines the different roles available in the system.

Predefined Roles:

- Student
- Teacher
- Program Content Creator
- Onboarding Specialist
- Administrator

Key Fields:

- `role_id` (UUID, PK): Unique identifier
- `role_name` (VARCHAR, UNIQUE): Role name
- `role_description` (TEXT): Detailed description

3. User Roles Table (`lms.user_roles`)

Implements Role-Based Access Control (RBAC) through a many-to-many relationship between users and roles.

Key Fields:

- `user_id` (UUID, FK): Reference to users table
- `role_id` (UUID, FK): Reference to roles table
- `assigned_by` (UUID, FK): User who assigned the role
- `is_active` (BOOLEAN): Whether the role assignment is active

4. Students Table (`lms.students`)

Extends user information with student-specific data.

Key Fields:

- `student_id` (UUID, PK): Unique identifier
- `user_id` (UUID, FK): Reference to users table
- `student_number` (VARCHAR, UNIQUE): Student identification number
- `enrollment_date`, `graduation_date` (DATE): Academic timeline
- `grade_level` (VARCHAR): Current grade level
- `emergency_contact_*`: Emergency contact information
- `special_needs`, `learning_preferences` (TEXT): Educational accommodations

5. Parents Table (`lms.parents`)

Stores parent/guardian information with relationship mapping to students.

Key Fields:

- `parent_id` (UUID, PK): Unique identifier
- `user_id` (UUID, FK): Reference to users table
- `occupation`, `employer`: Professional information
- `preferred_contact_method`, `preferred_contact_time`: Communication preferences

Related Table:

- `parent_student_relationships`: Many-to-many mapping between parents and students

6. Tutors Table (`lms.tutors`)

Contains tutor/teacher specific information and qualifications.

Key Fields:

- `tutor_id` (UUID, PK): Unique identifier
- `user_id` (UUID, FK): Reference to users table
- `employee_id` (VARCHAR, UNIQUE): Employee identification
- `hire_date`, `termination_date` (DATE): Employment timeline
- `qualifications`, `specializations` (TEXT): Professional credentials
- `hourly_rate` (DECIMAL): Compensation information
- `max_students_per_class` (INTEGER): Teaching capacity

7. Programs Table (`lms.programs`)

Defines educational programs offered by the institution.

Key Fields:

- `program_id` (UUID, PK): Unique identifier
- `program_name`, `program_code` (VARCHAR): Program identification
- `description`, `objectives`, `prerequisites` (TEXT): Program details
- `duration_weeks` (INTEGER): Program length
- `difficulty_level` (VARCHAR): Beginner/Intermediate/Advanced/Expert
- `age_group_min`, `age_group_max` (INTEGER): Target age range
- `price`, `currency` (DECIMAL, VARCHAR): Pricing information

8. Classrooms Table (`lms.classrooms`)

Represents specific class instances of programs with scheduling information.

Key Fields:

- `classroom_id` (UUID, PK): Unique identifier
- `classroom_name`, `classroom_code` (VARCHAR): Class identification
- `program_id` (UUID, FK): Associated program
- `primary_tutor_id`, `secondary_tutor_id` (UUID, FK): Assigned tutors
- `start_date`, `end_date` (DATE): Class duration
- `schedule_days`, `start_time`, `end_time`: Schedule information
- `is_online` (BOOLEAN): Virtual/physical class indicator
- `capacity` (INTEGER): Maximum enrollment

Related Table:

- `classroom_enrollments`: Many-to-many mapping between classrooms and students

9. Class Instances Table (`lms.class_instances`)

Represents individual class sessions/meetings within a classroom.

Key Fields:

- `instance_id` (UUID, PK): Unique identifier
- `classroom_id` (UUID, FK): Parent classroom
- `instance_number` (INTEGER): Sequential session number
- `session_date`, `start_time`, `end_time`: Session timing

- `conducting_tutor_id` (UUID, FK): Session instructor
- `session_title`, `session_description`: Session details
- `status`: Scheduled/In Progress/Completed/Cancelled/Postponed

Related Table:

- `class_instance_attendance`: Student attendance tracking per session

10. Materials System

Material Types Table (`lms.material_types`)

Categorizes different types of learning materials.

Predefined Types:

- Learning Notes
- Practice Exercise
- Homework
- Quiz/Exam
- Video/Audio Lessons
- Presentations
- Interactive Content
- Lab Exercises
- Projects

Materials Table (`lms.materials`)

Stores learning resources and content.

Key Fields:

- `material_id` (UUID, PK): Unique identifier
- `material_type_id` (UUID, FK): Material category
- `title`, `description`, `content` (TEXT): Material details
- `file_url`, `file_size_bytes`, `file_mime_type`: File information
- `difficulty_level`, `estimated_completion_time`: Learning metrics
- `points_possible` (INTEGER): Assessment scoring
- `is_required`, `is_published` (BOOLEAN): Availability flags

Class Instance Materials (`lms.class_instance_materials`)

Links materials to specific class sessions.

Student Material Progress (`lms.student_material_progress`)

Tracks individual student progress on materials.

Progress Tracking:

- Status: Not Started/In Progress/Completed/Submitted/Graded/Overdue

- `progress_percentage` (0-100%)
- Time tracking: `time_spent_minutes`
- Scoring: `score`, `max_score`
- Timestamps: started, completed, submitted, graded

Database Features

1. UUID Primary Keys

All tables use UUID primary keys for better scalability and security.

2. Timestamp Tracking

Most tables include `created_at` and `updated_at` timestamps with automatic triggers for updates.

3. Soft Deletes

Uses boolean flags (`is_active`, `is_enrolled`, etc.) instead of hard deletes to maintain data integrity.

4. Data Validation

Includes CHECK constraints for data validation (e.g., date ranges, enum values, percentage ranges).

5. Comprehensive Indexing

Strategic indexes on frequently queried columns for optimal performance.

6. Useful Views

Pre-built views for common queries:

- `v_active_students`: Active students with user details
- `v_active_tutors`: Available tutors with qualifications
- `v_classroom_details`: Classroom information with enrollment counts
- `v_student_enrollments`: Student-classroom relationships
- `v_class_instance_summary`: Session attendance summaries
- `v_user_roles`: User role assignments

Security Considerations

1. **Password Security**: Passwords are stored as hashes, never in plain text
2. **Role-Based Access**: Comprehensive RBAC implementation
3. **Data Privacy**: Personal information is properly normalized and protected
4. **Audit Trail**: Tracking of who created/modified records where applicable

Scalability Features

1. **UUID Keys**: Better for distributed systems and prevents enumeration attacks
2. **Normalized Design**: Reduces data redundancy and maintains consistency
3. **Efficient Indexing**: Optimized for common query patterns
4. **JSON Support**: JSONB fields for flexible metadata storage

Installation Order

Execute the SQL files in the following order:

1. `01_create_schema.sql` - Create schema and extensions
2. `02_create_roles_table.sql` - Roles and permissions
3. `03_create_users_table.sql` - Base user table
4. `04_create_user_roles_table.sql` - RBAC implementation
5. `05_create_students_table.sql` - Student entities
6. `06_create_parents_table.sql` - Parent entities and relationships
7. `07_create_tutors_table.sql` - Tutor entities
8. `08_create_programs_table.sql` - Educational programs
9. `09_create_classrooms_table.sql` - Classrooms and enrollments
10. `10_create_class_instances_table.sql` - Class sessions and attendance
11. `11_create_materials_table.sql` - Learning materials and progress
12. `12_create_triggers.sql` - Automated timestamp updates
13. `13_create_views.sql` - Reporting and convenience views

Future Enhancements

Consider these potential additions:

1. **Notifications System**: User notifications and alerts
2. **Gradebook**: Comprehensive grading and assessment tools
3. **Calendar Integration**: Calendar events and scheduling
4. **Communication**: Messaging between users
5. **Payment Processing**: Billing and payment tracking
6. **Reports and Analytics**: Advanced reporting capabilities
7. **Mobile API Support**: REST API considerations
8. **File Management**: Document storage and versioning