Comprehensive normalized RDBMS structure for the SOS Astro Academy. Here are the key features:

**Database Structure Highlights:**

**Core Entity Tables (15 tables)**

* **Certification Levels** - Stores the 4 certification levels with fees and requirements
* **Curriculum Organization** - Months, Weeks, Topics, and Practical Exercises hierarchically linked
* **Assessments** - Different assessment types with scoring details

**Student Management (8 tables)**

* Student profiles, enrollments, progress tracking
* Assessment results and certifications awarded
* Complete student lifecycle from enrollment to certification

**Faculty & Mentorship (3 tables)**

* Faculty profiles with specializations
* Mentor-student assignments
* Session tracking for mentorship meetings

**Learning Resources (4 tables)**

* Resource types (videos, documents, etc.)
* Resource library with metadata
* Week-resource mappings for curriculum structure

**Financial Management (2 tables)**

* Course packages with pricing
* Payment tracking with installments

**Communication & Engagement (5 tables)**

* Live sessions with recordings
* Attendance tracking
* Discussion forum with topics and posts

**Quality Assurance (1 table)**

* Student feedback across multiple dimensions

**Views for Reporting**

* Active students summary with progress
* Student performance metrics
* Revenue analysis

**Key Design Principles:**

✅ **3NF Normalization** - Eliminates redundancy ✅ **Referential Integrity** - Foreign keys maintain data consistency ✅ **Indexing Strategy** - Optimized for common queries ✅ **Scalability** - Can handle thousands of students ✅ **Audit Trail** - Created/updated timestamps ✅ **Flexibility** - Easy to extend with new features

This structure supports all aspects of the academy from enrollment to certification and career tracking!

-- ============================================

-- SOS ASTRO ACADEMY DATABASE SCHEMA

-- Normalized RDBMS Structure with Module Prefixes

-- ============================================

-- MODULE PREFIXES:

-- CUR\_ = Curriculum Management

-- ASM\_ = Assessment Management

-- PKG\_ = Package & Pricing

-- FAC\_ = Faculty & Mentorship

-- STU\_ = Student Management

-- RES\_ = Learning Resources

-- PLT\_ = Platform Features

-- CAR\_ = Career Support

-- PAY\_ = Payment & Financial

-- COM\_ = Communication & Collaboration

-- QUA\_ = Quality Assurance

-- ============================================

-- CURRICULUM MANAGEMENT MODULE (CUR\_)

-- ============================================

-- Certification Levels

CREATE TABLE cur\_certification\_levels (

level\_id INT PRIMARY KEY AUTO\_INCREMENT,

level\_name VARCHAR(100) NOT NULL,

level\_type ENUM('Foundation', 'Practitioner', 'Professional', 'Master') NOT NULL,

duration\_months INT NOT NULL,

duration\_weeks INT NOT NULL,

duration\_days INT NOT NULL,

sequence\_order INT NOT NULL,

passing\_score\_percentage DECIMAL(5,2) NOT NULL,

fee\_usd DECIMAL(10,2) NOT NULL,

description TEXT,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

UNIQUE KEY uk\_level\_type (level\_type)

);

-- Months (Curriculum Organization)

CREATE TABLE cur\_months (

month\_id INT PRIMARY KEY AUTO\_INCREMENT,

level\_id INT NOT NULL,

month\_number INT NOT NULL,

month\_title VARCHAR(200) NOT NULL,

description TEXT,

FOREIGN KEY (level\_id) REFERENCES cur\_certification\_levels(level\_id) ON DELETE CASCADE,

UNIQUE KEY uk\_level\_month (level\_id, month\_number)

);

-- Weeks (Detailed Curriculum Breakdown)

CREATE TABLE cur\_weeks (

week\_id INT PRIMARY KEY AUTO\_INCREMENT,

month\_id INT NOT NULL,

week\_start INT NOT NULL,

week\_end INT NOT NULL,

week\_title VARCHAR(200) NOT NULL,

theory\_hours DECIMAL(5,2) DEFAULT 0,

practical\_hours DECIMAL(5,2) DEFAULT 0,

self\_study\_hours DECIMAL(5,2) DEFAULT 0,

description TEXT,

FOREIGN KEY (month\_id) REFERENCES cur\_months(month\_id) ON DELETE CASCADE,

INDEX idx\_month\_weeks (month\_id, week\_start)

);

-- Topics

CREATE TABLE cur\_topics (

topic\_id INT PRIMARY KEY AUTO\_INCREMENT,

week\_id INT NOT NULL,

topic\_title VARCHAR(300) NOT NULL,

topic\_order INT NOT NULL,

topic\_description TEXT,

is\_core\_topic BOOLEAN DEFAULT TRUE,

FOREIGN KEY (week\_id) REFERENCES cur\_weeks(week\_id) ON DELETE CASCADE,

INDEX idx\_week\_topics (week\_id, topic\_order)

);

-- Practical Exercises

CREATE TABLE cur\_practical\_exercises (

exercise\_id INT PRIMARY KEY AUTO\_INCREMENT,

week\_id INT NOT NULL,

exercise\_title VARCHAR(300) NOT NULL,

exercise\_description TEXT,

exercise\_order INT NOT NULL,

estimated\_duration\_hours DECIMAL(5,2),

difficulty\_level ENUM('Beginner', 'Intermediate', 'Advanced', 'Master') NOT NULL,

FOREIGN KEY (week\_id) REFERENCES cur\_weeks(week\_id) ON DELETE CASCADE,

INDEX idx\_week\_exercises (week\_id, exercise\_order)

);

-- ============================================

-- ASSESSMENT MANAGEMENT MODULE (ASM\_)

-- ============================================

-- Assessment Types

CREATE TABLE asm\_types (

assessment\_type\_id INT PRIMARY KEY AUTO\_INCREMENT,

type\_name VARCHAR(100) NOT NULL,

description TEXT,

UNIQUE KEY uk\_type\_name (type\_name)

);

-- Assessments

CREATE TABLE asm\_assessments (

assessment\_id INT PRIMARY KEY AUTO\_INCREMENT,

level\_id INT NOT NULL,

assessment\_type\_id INT NOT NULL,

assessment\_name VARCHAR(200) NOT NULL,

duration\_hours DECIMAL(5,2),

max\_marks INT,

quantity INT DEFAULT 1,

description TEXT,

FOREIGN KEY (level\_id) REFERENCES cur\_certification\_levels(level\_id) ON DELETE CASCADE,

FOREIGN KEY (assessment\_type\_id) REFERENCES asm\_types(assessment\_type\_id),

INDEX idx\_level\_assessments (level\_id)

);

-- ============================================

-- PACKAGE & PRICING MODULE (PKG\_)

-- ============================================

-- Course Packages

CREATE TABLE pkg\_course\_packages (

package\_id INT PRIMARY KEY AUTO\_INCREMENT,

package\_name VARCHAR(100) NOT NULL,

package\_type ENUM('Complete', 'Level-by-Level', 'Fast-Track') NOT NULL,

total\_fee\_usd DECIMAL(10,2) NOT NULL,

duration\_months INT NOT NULL,

hours\_per\_week INT,

savings\_usd DECIMAL(10,2) DEFAULT 0,

payment\_plan\_available BOOLEAN DEFAULT FALSE,

payment\_installments INT,

description TEXT,

is\_active BOOLEAN DEFAULT TRUE,

UNIQUE KEY uk\_package\_type (package\_type)

);

-- Package Inclusions

CREATE TABLE pkg\_inclusions (

inclusion\_id INT PRIMARY KEY AUTO\_INCREMENT,

package\_id INT NOT NULL,

inclusion\_item VARCHAR(300) NOT NULL,

inclusion\_order INT NOT NULL,

FOREIGN KEY (package\_id) REFERENCES pkg\_course\_packages(package\_id) ON DELETE CASCADE,

INDEX idx\_package\_inclusions (package\_id, inclusion\_order)

);

-- ============================================

-- FACULTY & MENTORSHIP MODULE (FAC\_)

-- ============================================

-- Faculty

CREATE TABLE fac\_faculty (

faculty\_id INT PRIMARY KEY AUTO\_INCREMENT,

first\_name VARCHAR(100) NOT NULL,

last\_name VARCHAR(100) NOT NULL,

email VARCHAR(255) UNIQUE NOT NULL,

phone VARCHAR(20),

years\_of\_experience INT,

specialization VARCHAR(200),

bio TEXT,

is\_lead\_faculty BOOLEAN DEFAULT FALSE,

is\_guest\_faculty BOOLEAN DEFAULT FALSE,

traditional\_lineage VARCHAR(200),

published\_works TEXT,

profile\_image\_url VARCHAR(500),

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP

);

-- Mentor Requirements

CREATE TABLE fac\_mentor\_requirements (

requirement\_id INT PRIMARY KEY AUTO\_INCREMENT,

level\_id INT NOT NULL,

mentor\_type VARCHAR(100) NOT NULL,

student\_ratio VARCHAR(20) NOT NULL,

description TEXT,

FOREIGN KEY (level\_id) REFERENCES cur\_certification\_levels(level\_id) ON DELETE CASCADE

);

-- Student Mentor Assignments

CREATE TABLE fac\_student\_mentors (

assignment\_id INT PRIMARY KEY AUTO\_INCREMENT,

student\_id INT NOT NULL,

faculty\_id INT NOT NULL,

level\_id INT NOT NULL,

assigned\_date DATE NOT NULL,

end\_date DATE,

is\_active BOOLEAN DEFAULT TRUE,

FOREIGN KEY (student\_id) REFERENCES stu\_students(student\_id) ON DELETE CASCADE,

FOREIGN KEY (faculty\_id) REFERENCES fac\_faculty(faculty\_id),

FOREIGN KEY (level\_id) REFERENCES cur\_certification\_levels(level\_id),

INDEX idx\_active\_assignments (is\_active),

INDEX idx\_student\_mentor (student\_id, faculty\_id)

);

-- Mentorship Sessions

CREATE TABLE fac\_mentorship\_sessions (

session\_id INT PRIMARY KEY AUTO\_INCREMENT,

assignment\_id INT NOT NULL,

session\_date DATETIME NOT NULL,

duration\_minutes INT,

session\_type ENUM('One-on-One', 'Group', 'Emergency', 'Project Review') NOT NULL,

topic\_discussed VARCHAR(300),

notes TEXT,

student\_feedback\_rating INT CHECK (student\_feedback\_rating BETWEEN 1 AND 5),

FOREIGN KEY (assignment\_id) REFERENCES fac\_student\_mentors(assignment\_id) ON DELETE CASCADE,

INDEX idx\_session\_date (session\_date)

);

-- ============================================

-- STUDENT MANAGEMENT MODULE (STU\_)

-- ============================================

-- Students

CREATE TABLE stu\_students (

student\_id INT PRIMARY KEY AUTO\_INCREMENT,

first\_name VARCHAR(100) NOT NULL,

last\_name VARCHAR(100) NOT NULL,

email VARCHAR(255) UNIQUE NOT NULL,

phone VARCHAR(20),

whatsapp VARCHAR(20),

country VARCHAR(100),

enrollment\_date DATE NOT NULL,

current\_level\_id INT,

sos\_astro\_account\_id VARCHAR(100),

is\_active BOOLEAN DEFAULT TRUE,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

FOREIGN KEY (current\_level\_id) REFERENCES cur\_certification\_levels(level\_id),

INDEX idx\_enrollment\_date (enrollment\_date),

INDEX idx\_active\_students (is\_active)

);

-- Student Enrollments

CREATE TABLE stu\_enrollments (

enrollment\_id INT PRIMARY KEY AUTO\_INCREMENT,

student\_id INT NOT NULL,

package\_id INT NOT NULL,

level\_id INT NOT NULL,

enrollment\_date DATE NOT NULL,

start\_date DATE,

expected\_completion\_date DATE,

actual\_completion\_date DATE,

status ENUM('Enrolled', 'In Progress', 'Completed', 'Dropped', 'On Hold') NOT NULL DEFAULT 'Enrolled',

payment\_status ENUM('Pending', 'Partial', 'Paid', 'Refunded') NOT NULL DEFAULT 'Pending',

FOREIGN KEY (student\_id) REFERENCES stu\_students(student\_id) ON DELETE CASCADE,

FOREIGN KEY (package\_id) REFERENCES pkg\_course\_packages(package\_id),

FOREIGN KEY (level\_id) REFERENCES cur\_certification\_levels(level\_id),

INDEX idx\_student\_status (student\_id, status),

INDEX idx\_enrollment\_date (enrollment\_date)

);

-- Student Progress

CREATE TABLE stu\_progress (

progress\_id INT PRIMARY KEY AUTO\_INCREMENT,

student\_id INT NOT NULL,

week\_id INT NOT NULL,

completion\_percentage DECIMAL(5,2) DEFAULT 0,

status ENUM('Not Started', 'In Progress', 'Completed', 'Skipped') DEFAULT 'Not Started',

started\_date DATE,

completed\_date DATE,

notes TEXT,

FOREIGN KEY (student\_id) REFERENCES stu\_students(student\_id) ON DELETE CASCADE,

FOREIGN KEY (week\_id) REFERENCES cur\_weeks(week\_id) ON DELETE CASCADE,

UNIQUE KEY uk\_student\_week (student\_id, week\_id),

INDEX idx\_student\_progress (student\_id, status)

);

-- Student Assessments

CREATE TABLE stu\_assessments (

student\_assessment\_id INT PRIMARY KEY AUTO\_INCREMENT,

student\_id INT NOT NULL,

assessment\_id INT NOT NULL,

attempt\_number INT DEFAULT 1,

assessment\_date DATE,

marks\_obtained DECIMAL(5,2),

max\_marks DECIMAL(5,2),

percentage DECIMAL(5,2),

passed BOOLEAN DEFAULT FALSE,

feedback TEXT,

evaluated\_by INT,

evaluated\_date DATE,

FOREIGN KEY (student\_id) REFERENCES stu\_students(student\_id) ON DELETE CASCADE,

FOREIGN KEY (assessment\_id) REFERENCES asm\_assessments(assessment\_id),

FOREIGN KEY (evaluated\_by) REFERENCES fac\_faculty(faculty\_id),

INDEX idx\_student\_assessments (student\_id, assessment\_id)

);

-- Certifications Awarded

CREATE TABLE stu\_certifications (

certification\_id INT PRIMARY KEY AUTO\_INCREMENT,

student\_id INT NOT NULL,

level\_id INT NOT NULL,

issue\_date DATE NOT NULL,

certificate\_number VARCHAR(100) UNIQUE NOT NULL,

blockchain\_hash VARCHAR(200),

verification\_url VARCHAR(500),

is\_active BOOLEAN DEFAULT TRUE,

revoked\_date DATE,

revocation\_reason TEXT,

FOREIGN KEY (student\_id) REFERENCES stu\_students(student\_id) ON DELETE CASCADE,

FOREIGN KEY (level\_id) REFERENCES cur\_certification\_levels(level\_id),

INDEX idx\_student\_certs (student\_id),

INDEX idx\_cert\_number (certificate\_number)

);

-- ============================================

-- LEARNING RESOURCES MODULE (RES\_)

-- ============================================

-- Resource Types

CREATE TABLE res\_types (

resource\_type\_id INT PRIMARY KEY AUTO\_INCREMENT,

type\_name VARCHAR(100) NOT NULL,

description TEXT,

UNIQUE KEY uk\_type\_name (type\_name)

);

-- Learning Resources

CREATE TABLE res\_resources (

resource\_id INT PRIMARY KEY AUTO\_INCREMENT,

resource\_type\_id INT NOT NULL,

resource\_title VARCHAR(300) NOT NULL,

description TEXT,

file\_url VARCHAR(500),

duration\_minutes INT,

is\_premium BOOLEAN DEFAULT FALSE,

created\_date DATE,

created\_by INT,

FOREIGN KEY (resource\_type\_id) REFERENCES res\_types(resource\_type\_id),

FOREIGN KEY (created\_by) REFERENCES fac\_faculty(faculty\_id),

INDEX idx\_resource\_type (resource\_type\_id)

);

-- Week Resources (Mapping)

CREATE TABLE res\_week\_mapping (

week\_resource\_id INT PRIMARY KEY AUTO\_INCREMENT,

week\_id INT NOT NULL,

resource\_id INT NOT NULL,

resource\_order INT NOT NULL,

is\_mandatory BOOLEAN DEFAULT TRUE,

FOREIGN KEY (week\_id) REFERENCES cur\_weeks(week\_id) ON DELETE CASCADE

-- ============================================

-- INDEXES FOR PERFORMANCE OPTIMIZATION

-- ============================================

CREATE INDEX idx\_students\_email ON students(email);

CREATE INDEX idx\_students\_active ON students(is\_active, current\_level\_id);

CREATE INDEX idx\_enrollments\_status ON student\_enrollments(status, level\_id);

CREATE INDEX idx\_progress\_completion ON student\_progress(student\_id, completion\_percentage);

CREATE INDEX idx\_certifications\_issue\_date ON certifications\_awarded(issue\_date);

CREATE INDEX idx\_payments\_status ON payments(payment\_status, payment\_date);

-- ============================================

-- VIEWS FOR COMMON QUERIES

-- ============================================

-- Active Students Summary

CREATE VIEW v\_active\_students\_summary AS

SELECT

s.student\_id,

CONCAT(s.first\_name, ' ', s.last\_name) AS student\_name,

s.email,

cl.level\_name,

se.status AS enrollment\_status,

se.start\_date,

se.expected\_completion\_date,

COUNT(DISTINCT sp.week\_id) AS weeks\_completed,

AVG(sp.completion\_percentage) AS avg\_completion\_percentage

FROM students s

JOIN student\_enrollments se ON s.student\_id = se.student\_id

JOIN certification\_levels cl ON se.level\_id = cl.level\_id

LEFT JOIN student\_progress sp ON s.student\_id = sp.student\_id AND sp.status = 'Completed'

WHERE s.is\_active = TRUE AND se.status = 'In Progress'

GROUP BY s.student\_id, s.first\_name, s.last\_name, s.email, cl.level\_name, se.status, se.start\_date, se.expected\_completion\_date;

-- Student Performance Summary

CREATE VIEW v\_student\_performance AS

SELECT

s.student\_id,

CONCAT(s.first\_name, ' ', s.last\_name) AS student\_name,

cl.level\_name,

COUNT(sa.student\_assessment\_id) AS total\_assessments,

AVG(sa.percentage) AS avg\_score,

SUM(CASE WHEN sa.passed = TRUE THEN 1 ELSE 0 END) AS passed\_assessments

FROM students s

JOIN student\_enrollments se ON s.student\_id = se.student\_id

JOIN certification\_levels cl ON se.level\_id = cl.level\_id

LEFT JOIN student\_assessments sa ON s.student\_id = sa.student\_id

GROUP BY s.student\_id, s.first\_name, s.last\_name, cl.level\_name;

-- Revenue Summary

CREATE VIEW v\_revenue\_summary AS

SELECT

DATE\_FORMAT(p.payment\_date, '%Y-%m') AS month\_year,

cp.package\_name,

COUNT(DISTINCT se.student\_id) AS students\_enrolled,

SUM(p.amount\_usd) AS total\_revenue

FROM payments p

JOIN student\_enrollments se ON p.enrollment\_id = se.enrollment\_id

JOIN course\_packages cp ON se.package\_id = cp.package\_id

WHERE p.payment\_status = 'Completed'

GROUP BY DATE\_FORMAT(p.payment\_date, '%Y-%m'), cp.package\_name;

-- ============================================

-- END OF SCHEMA

-- ============================================