University Management System – Data Model Documentation

~ By Parth Shah

1. Overview

The University Management System (UMS) data model manages core academic and research processes:

- Student enrollment in course **sections** (term-specific offerings)
- Teaching assignments (including **co-teaching**)
- Course **prerequisites**
- **Grades** recorded per enrollment
- **Research projects** with department ownership, professor leadership, and student participation

The design prioritizes integrity, clarity of relationships, and scalability for reporting and analytics.

---

2. Objectives & Scope

- Maintain master data for **Departments**, **Professors**, **Students**, and **Courses**
- Represent **Course Sections** by **term** and **year**; support **multiple instructors** per section
- Record **Student enrollments** at the section level, including status and dates
- Store **grade events** (e.g., midterm, final) tied to each enrollment
- Capture **Course prerequisites** (many-to-many, self-referential)
- Manage **Research Projects**: ownership by department, **professor leads (PIs)**, and student members

---

3. Entity Descriptions & Attributes

3.1 Departments

- **dept_id (PK)**

- **name (UNIQUE)**
- **head_prof_id (FK → Professors.prof_id)**
- **created_at, updated_at**
  Notes: Head should belong to the same department (business rule).

3.2 Professors

- **prof_id (PK)**
- **full_name**
- **email (UNIQUE, case-insensitive)**
- **rank** ∈ {Assistant, Associate, Full}
- **dept_id (FK → Departments.dept_id)**
- **created_at, updated_at**

3.3 Students

- **student_id (PK)**
- **full_name**
- **email (UNIQUE, case-insensitive)**
- **major**
- **gpa** (0.00–4.00)
- **advisor_prof_id (FK → Professors.prof_id)**
- **created_at, updated_at**

3.4 Courses

- **course_id (PK)**
- **course_code** (e.g., CS101, unique within department)
- **name, description**
- **credits** (>0, ≤10)
- **dept_id (FK → Departments.dept_id)**
- **created_at, updated_at**

3.5 Course_Sections

- **section_id (PK)**
- **course_id (FK → Courses.course_id)**
- **term** ∈ {Spring, Summer, Fall, Winter}
- **year**
- **section_code** (e.g., A / 001)
- **capacity** (optional)
- **start_date, end_date**
- **created_at, updated_at**

3.7 Enrollments

- **enrollment_id (PK)**
- **student_id (FK → Students.student_id)**
- **section_id (FK → Course_Sections.section_id)**
- **enrolled_at**
- **status** ∈ {ENROLLED, DROPPED, COMPLETED, WITHDRAWN}
- **UNIQUE(student_id, section_id)**

3.8 Grades

- **grade_id (PK)**
- **enrollment_id (FK → Enrollments.enrollment_id)**
- **grade_value** (A, A-, B+, Pass, etc.)
- **date_assigned**
- **remarks**

3.9 Research_Projects

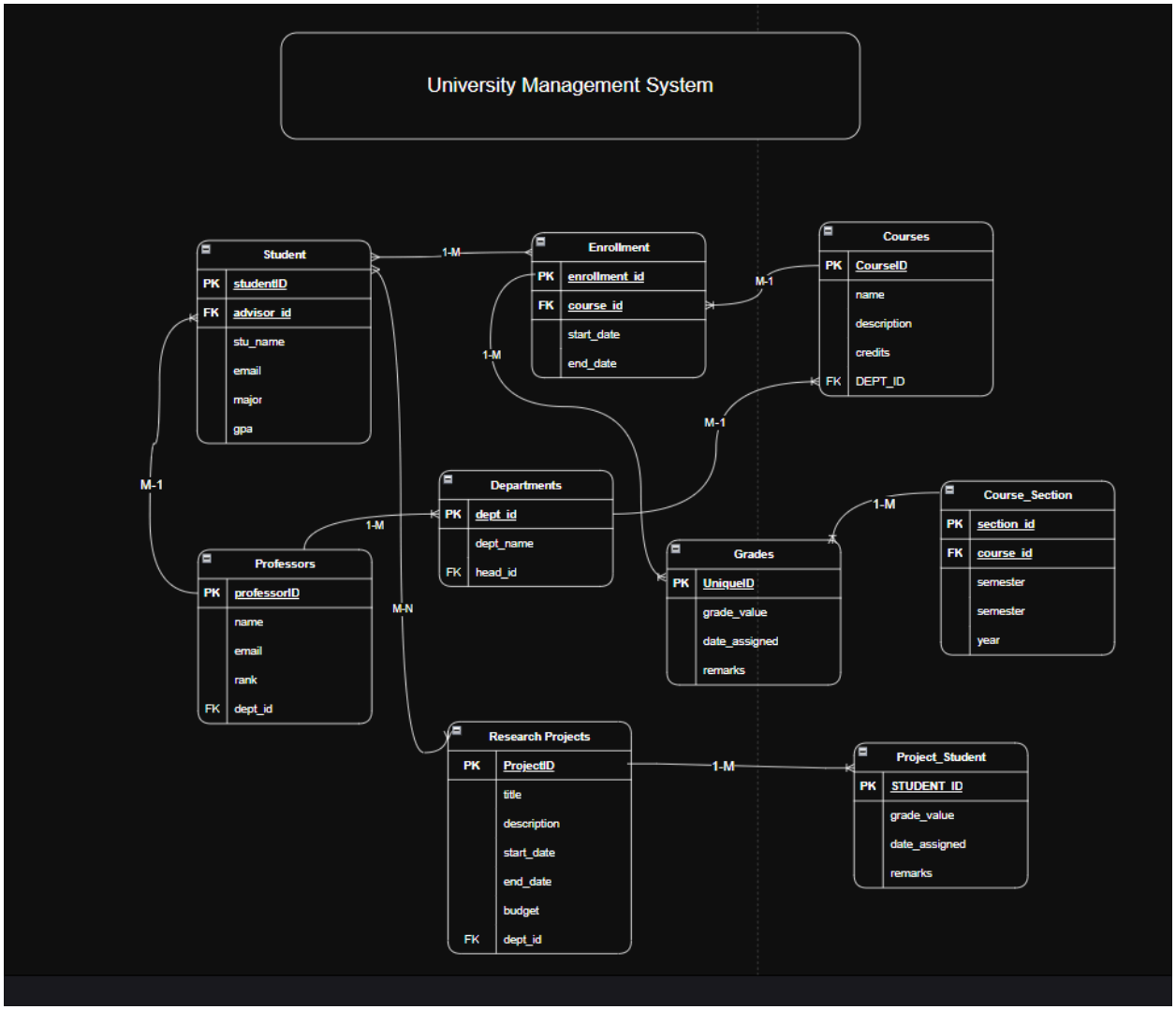
- **project_id (PK)**
- **dept_id (FK → Departments.dept_id)**
- **title, description**
- **start_date, end_date** (end ≥ start or NULL)
- **budget** (≥ 0)
- **created_at, updated_at**

---

5. Relationships & Cardinality

- **Professors (M) — (1) Departments**
  `professors.dept_id → departments.dept_id`
- **Courses (M) — (1) Departments**
  `courses.dept_id → departments.dept_id`
- **Students (M) — (1) Professors** (academic advisor)
  `students.advisor_prof_id → professors.prof_id`
- **Course_Sections (M) — (1) Courses**
  `course_sections.course_id → courses.course_id`
- **Section (M) — (1) Course_Sections**
  `section_instructors.section_id → course_sections.section_id`
- **Section — (1) Professors**
  `section_instructors.prof_id → professors.prof_id`

- **Enrollments (M) — (1) Students**
  `enrollments.student_id → students.student_id`
- **Enrollments (M) — (1) Course_Sections**
  `enrollments.section_id → course_sections.section_id`
- **Grades (M) — (1) Enrollments**
  `grades.enrollment_id → enrollments.enrollment_id`
- **Research_Projects (M) — (1) Departments**
  `research_projects.dept_id → departments.dept_id`



9. Non-Goals / Out of Scope (to the Current Version)

- Detailed academic policies (pre-requisite checking logic, program audits)
- Waitlists and auto-enroll logic

- Grading schemes & GPA automation
- Room scheduling and timetable optimization
- Audit trails (created_by / changed_by)

---

10. Future Enhancements

- **Waitlists** and seat reservation windows
- **Grade Schemes** and automatic **GPA** computation
- **Advisor load** monitoring and approval workflows
- Research **funding sources** and yearly budgets
- **Audit** metadata (created_by, changed_by, change_reason)

Library Management System - Data Model Documentation

1. Overview

The Library Management System (LMS) data model manages core library operations, including:

- Cataloging books and authors
- Tracking publishers and book editions
- Maintaining multiple physical copies of each book
- Managing members and their borrowing privileges
- Recording borrowing activities (loan, due date, fines, return condition)
- Handling reservations for books currently on loan
- Managing multiple library branches and distributing books across them

The design focuses on data integrity, clarity of relationships, and scalability for future enhancements such as inter-branch transfers and digital book support.

---

2. Objectives & Scope

- Maintain master data for **Books**, **Authors**, **Publishers**, **Branches**, **Members**
- Store multiple **Book Copies** per title
- Record **Borrowing transactions** including due dates and fines
- Allow **Reservations** for books that are unavailable
- Support **Library Branches** so copies exist across locations
- Ensure accurate tracking of **book availability and member activity**

---

3. Entity Descriptions & Attributes

3.1 Books

- **ISBN (PK)**
- title
- publication_year
- edition
- genre
- publisher_id (FK → Publishers.publisher_id)
- format (Hardcover, Paperback, eBook, Audio, etc.)

Notes: A book represents a title-level record, not an individual physical item.

3.2 Authors

- **author_id (PK)**
- name
- nationality

Notes: Authors can write multiple books. (In this simplified model, a book associates with one author.)

3.3 Publishers

- **publisher_id (PK)**
- name
- contact
- email

Notes: A publisher can publish many books.

3.4 Book_Copies

- **copyID (PK)**
- ISBN (FK → Books.ISBN)
- condition (New, Good, Fair, Poor)
- status (Available, On Loan, Reserved, Lost)

Notes: Each physical copy of a book receives its own ID and lifecycle state.

3.5 Branches

- **branchID (PK)**
- name
- location
- contact

Notes: Each library branch stores its own inventory of book copies.

3.6 Members

- **member_id (PK)**
- name
- contact_info
- membership_type (Regular, Premium)

Notes: Membership type may determine borrowing limits or loan durations.

3.7 Borrowings

- **borrowing_id (PK)**
- member_id (FK → Members.member_id)
- copy_id (FK → Book_Copies.copyID)
- borrow_date
- due_date
- return_date
- fines
- return_condition

Notes: Tracks each time a book copy is borrowed and returned.

3.8 Reservations

- **reservation_id (PK)**
- member_id (FK → Members.member_id)
- ISBN (FK → Books.ISBN)
- reserved_date
- expiration_date
- notification_status (Pending, Sent, Cancelled, Expired, Fulfilled)

Notes: Reservations occur at the *book title level*, not copy level.

---

4. Relationships & Cardinality

Books (1) — (M) Book_Copies

`book_copies.isbn → books.ISBN`

Each book can have many copies.

Publishers (1) — (M) Books

`books.publisher_id → publishers.publisher_id`

A publisher can publish many books.

Members (1) — (M) Borrowings

`borrowings.member_id → members.member_id`

A member may borrow many books.

Book_Copies (1) — (M) Borrowings

`borrowings.copy_id → book_copies.copyID`

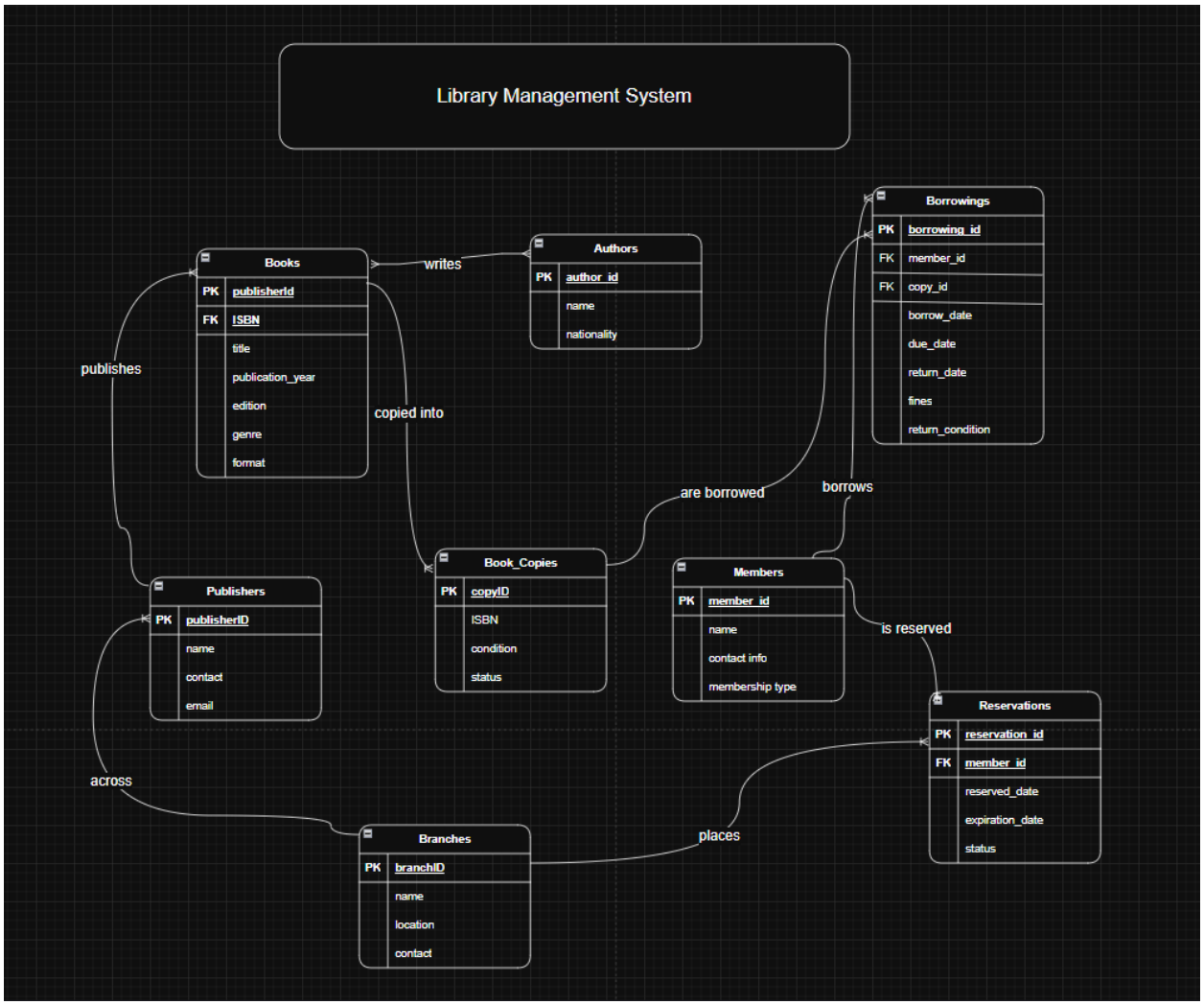
A copy may be borrowed many times.

Members (1) — (M) Reservations

`reservations.member_id → members.member_id`

Books (1) — (M) Reservations

`reservations.ISBN → books.ISBN`



---

5. Non-Goals / Out of Scope (Current Version)

- Tracking inter-branch book transfers
- Managing digital/eBook licensing
- Fine calculation rules (grace periods, rate per day)
- Automatic selection of next reservation holder
- Shelf location and physical cataloging systems
- Staff accounts and permissions
- Audit logs (created_by / updated_by)

---

6. Future Enhancements

- Add **inter-branch transfer logs** to track movement of copies
- Add **Book_Authors** table to support real-world many-to-many authorship
- Implement **loan rules** per membership type (Regular vs. Premium)
- Add **automatic fine calculation** and fine payment records
- Integrate **notifications system** for overdue items and reservation alerts
- Add **digital book support** for eBooks and audio files
- Add **branch inventory tracking** with shelf locations