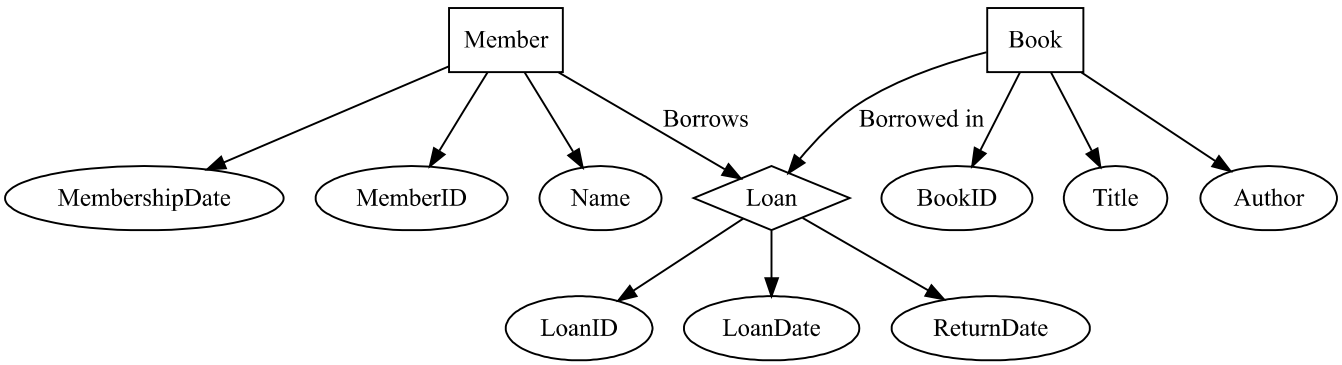
Advanced Databases - HW2 - Entity Relation Models

AUTHORS
Mohith Peddineni (V01106924) - peddinenim@vcu.edu
Nagaraj Islavath (V01108258) - islavathn@vcu.edu
Tejeshwar Reddy Renthem (V01113196) - renthemt@vcu.edu
Sai Vinay Potluri (V01106912) - potluris@vcu.edu
Github_Repo_Link

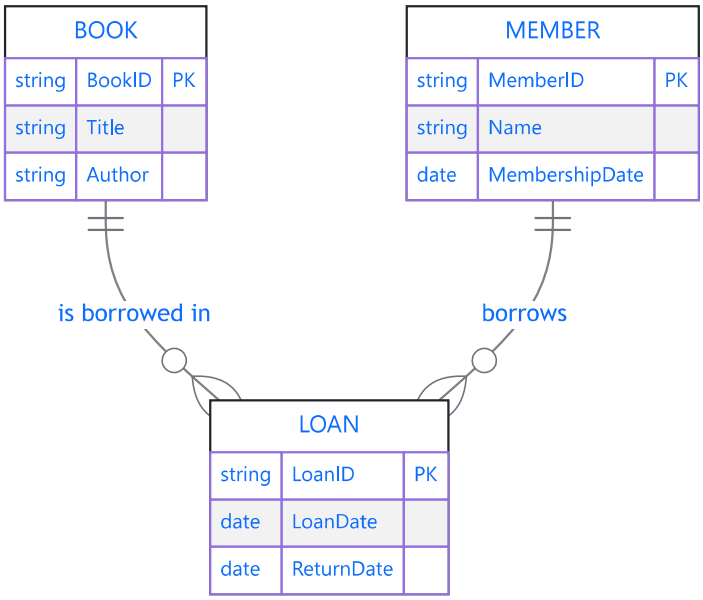
1. Library Management System

Picture a library that needs to manage books, members, and loans. Each book has a book ID, title, and author. Members have a member ID, name, and membership date. Loans have a loan ID, loan date, and return date. A member can borrow multiple books, and each book can be borrowed by multiple members over time. Each loan is associated with a single member borrowing a single book.

Graphviz (Chen Notation)



Mermaid (Crow's Foot Notation)



A Library Management System (LMS) is a structured database-driven solution designed to handle books, members, and loan transactions efficiently. The system models entities such as **Book**, **Member**, and **Loan** to track borrowing activities and enforce constraints such as due dates and member eligibility.

Entities and Relationships

- **Book**: Stores details about books in the library.
- **Member**: Stores details about library members.
- **Loan**: Represents the borrowing of a book by a member.

Relation Set Schema

Book

```
Book(BookID, Title, Author)
```

- `BookID (PK)`: Unique identifier for each book.
- `Title`: Name of the book.
- `Author`: Author of the book.

Member

```
Member(MemberID, Name, MembershipDate)
```

- `MemberID (PK)`: Unique identifier for each member.
- `Name`: Name of the member.
- `MembershipDate`: Date the member joined the library.

Loan

```
Loan(LoanID, LoanDate, ReturnDate, MemberID [FK], BookID [FK])
```

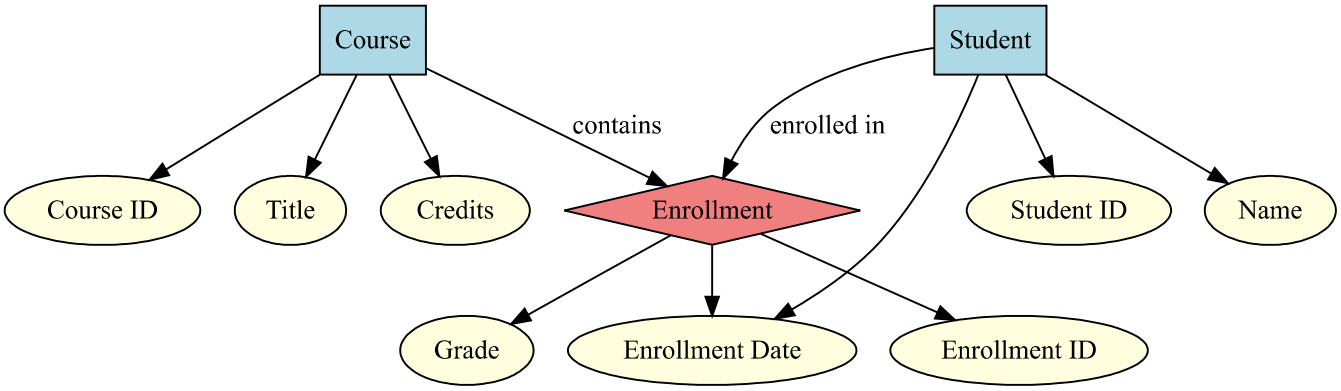
- `LoanID (PK)`: Unique identifier for each loan.
- `LoanDate`: Date the book was borrowed.
- `ReturnDate`: Date the book is expected to be returned.
- `MemberID (FK)`: The member who borrowed the book (references Member).
- `BookID (FK)`: The book being borrowed (references Book).

Each loan entry connects a member with a book, allowing multiple books to be borrowed by a member over time.

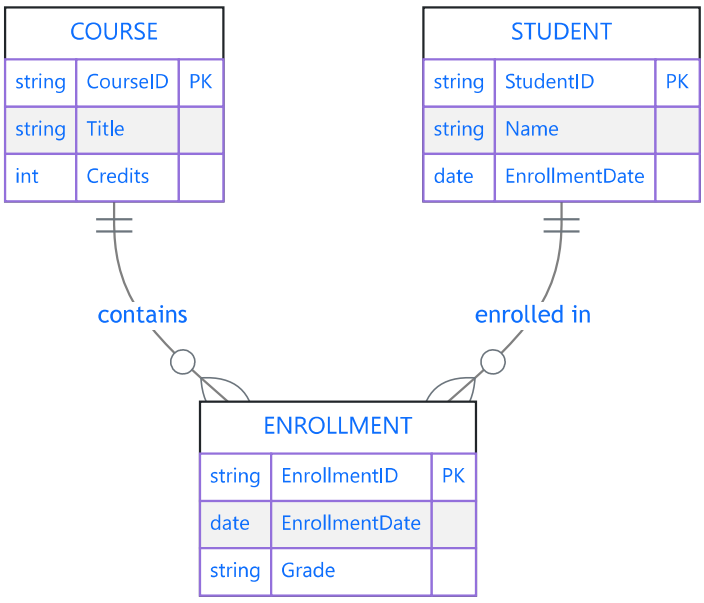
2. School Course Enrollment System

Imagine a school that needs to manage courses, students, and enrollments. Each course has a course ID, title, and credits. Students have a student ID, name, and enrollment date. Enrollments have an enrollment ID, enrollment date, and grade. A student can enroll in multiple courses, and each course can have multiple students. Each enrollment is associated with a single student enrolling in a single course.

Graphviz (Chen Notation)



Mermaid (Crow's Foot Notation)



School Course Enrollment System

A School Course Enrollment System is designed to manage student-course relationships in an academic institution. The core database model consists of **Student**, **Course**, and **Enrollment** to track student enrollments, manage grades, and schedule courses efficiently while maintaining data integrity through foreign key constraints.

Entities and Relationships

- **Course**: Represents courses offered by the school.
- **Student**: Represents students enrolled in the school.
- **Enrollment**: Tracks which students have enrolled in which courses.

Relation Set Schema

Course

```
Course(CourseID, Title, Credits)
```

- `CourseID (PK)`: Unique identifier for each course.
- `Title`: Name of the course.
- `Credits`: Number of credits awarded for completing the course.

Student

```
Student(StudentID, Name, EnrollmentDate)
```

- `StudentID (PK)`: Unique identifier for each student.
- `Name`: Name of the student.
- `EnrollmentDate`: Date the student enrolled in the school.

Enrollment

```
Enrollment(EnrollmentID, EnrollmentDate, Grade, StudentID [FK], CourseID [FK])
```

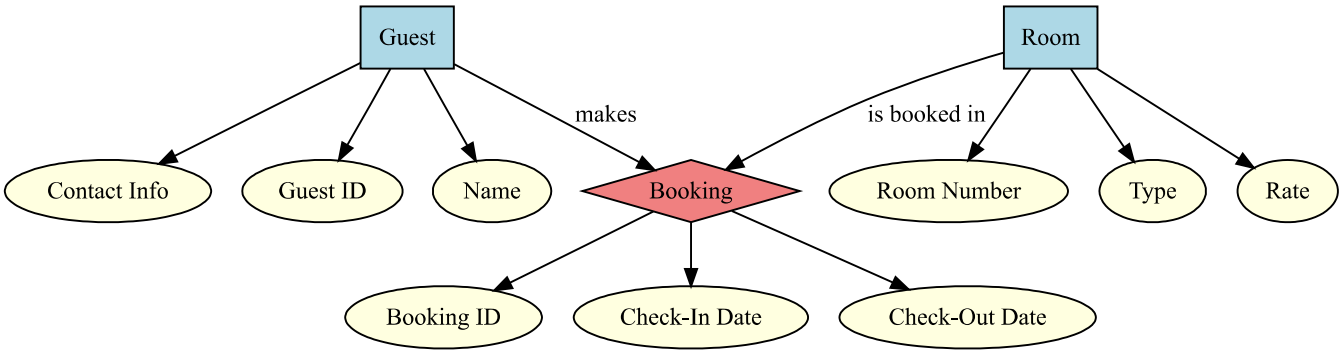
- `EnrollmentID (PK)`: Unique identifier for each enrollment record.
- `EnrollmentDate`: Date the student enrolled in the course.
- `Grade`: Grade received in the course.
- `StudentID (FK)`: The student who enrolled (references Student).
- `CourseID (FK)`: The course being enrolled in (references Course).

Each student can enroll in multiple courses, and each course can have multiple students. The **Enrollment** entity serves as a many-to-many bridge between students and courses.

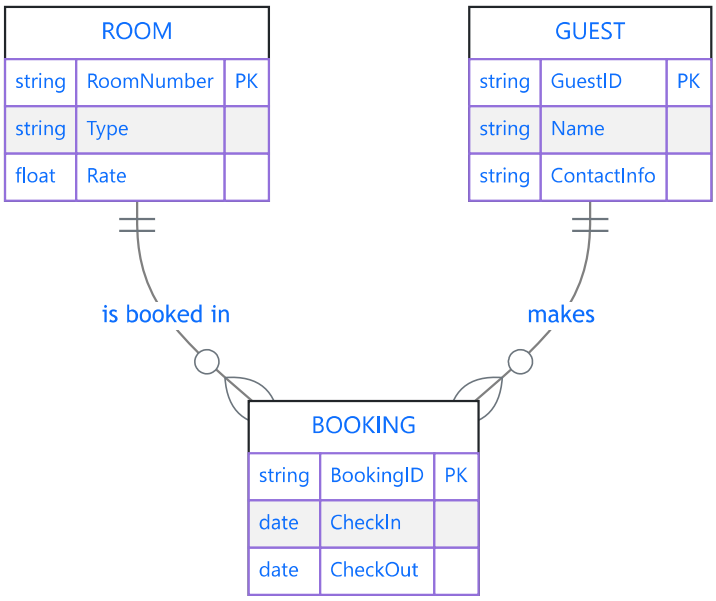
3. Hotel Booking System

Consider a hotel that wants to manage rooms, guests, and bookings. Each room has a room number, type, and rate. Guests have a guest ID, name, and contact information. Bookings have a booking ID, check-in date, and check-out date. A guest can make multiple bookings, and each room can be booked multiple times over different dates. Each booking is associated with one guest and one room.

Graphviz (Chen Notation)



Mermaid (Crow’s Foot Notation)



Hotel Booking System

A Hotel Booking System is a relational database application designed to track room reservations and guest management. The key entities include **Room**, **Guest**, and **Booking** to ensure optimized room allocation, prevent double bookings, and enhance guest experience through data-driven availability tracking.

Entities and Relationships

- **Room**: Represents the rooms available in the hotel.
- **Guest**: Represents guests who stay at the hotel.
- **Booking**: Tracks room reservations by guests.

Relation Set Schema

Room

```
Room(RoomNumber, Type, Rate)
```

- `RoomNumber (PK)`: Unique number assigned to each room.
- `Type`: Type of room (e.g., single, double, suite).
- `Rate`: Cost per night for the room.

Guest

```
Guest(GuestID, Name, ContactInfo)
```

- `GuestID (PK)`: Unique identifier for each guest.
- `Name`: Name of the guest.
- `ContactInfo`: Contact details of the guest.

Booking

```
Booking(BookingID, CheckIn, CheckOut, GuestID [FK], RoomNumber [FK])
```

- `BookingID (PK)`: Unique identifier for each booking.
- `CheckIn`: Check-in date for the guest.
- `CheckOut`: Check-out date for the guest.
- `GuestID (FK)`: The guest who made the booking (references Guest).
- `RoomNumber (FK)`: The room booked (references Room).

Each guest can book multiple rooms over time, and each room can be booked multiple times by different guests. The **Booking** entity manages these relationships efficiently.