Library System List of Relational Schemas

Employees – A

Fields:

- Employees (Entity)
- Employee_Name – string
- Employee_ID (Primary Key) - int
- Email_Address - string
- {Phone_Number} – numerous values accepted, int
- Address (Composite Attribute with components):
    - Street, City, State, Zip

Relationships & Cardinality:

- Employees – Members
    - Relationship: Checks Out
        - Many employees can check out many members
        - Many members can be checked out by many employees
    - Cardinality: Many-to-many

Schemas:

Employees
Employee_Name
Employee_ID (Primary Key)
Email_Address
Street
City
State
Zip

Employee Phone Numbers
Employee_ID (Foreign key references Employee(Employee_ID)
Phone_Number

Books – A

Fields:

- Books (Entity)
- Book_name - string
- BookID (Primary Key) - int
- ISBN - int
- Author - string

Relationships & Cardinality:

- Books – Members
    - Relationship: Borrows
        - A member can borrow many books.
        - A book can only be borrowed by one member at a time.
    - Cardinality: one-to-many

Schemas:

Books
Book_Name
BookID (Primary Key)
ISBN
Author

Movies - S

Fields:

- Movies (Entity)
- Movie_Title - string
- MovieID (Primary Key) – int
- Director – string
- Duration - int
- Release Date – date
- Genre – string

Relationships & Cardinality:

- Movies – Members

    - Relationship: Borrows
        - A member can borrow many Movies
        - A movie can only be borrowed by one member at a time
    - Cardinality: one-to-many

Schemas:

Movies
Movie_Title
MovieID (Primary Key)
Director
Duration
Release Date
Genre

Members – V

Fields:

- name – string
- memberID – int (Primary Key)
- email
- {phone number} – numerous values accepted

Relationships & Cardinality:

- Members – Library Cards:
    - Relationship: has
        - A member has to have a library card
        - A library card
    - Cardinality: one-to-one
- Members – Books – Movies:
    - Relationship: can borrow
        - A member can borrow numerous books or movies
        - A book or movie can only be borrowed by one member at a time
    - Ternary Relationship
    - Cardinality: one-to-many
- Employees – Members
    - Relationship: Checks Out
        - Many employees can check out many members
        - Many members can be checked out by many employees

Schemas:

Member
Name
memberID (PK)
email
street
city
state
zip

Member Phone Numbers
memberID (foreign key references Member(memberID)
phoneNumber

Library Cards – V

Fields:

- cardID (primary key)
- creationDate
- expirationDate
- fees

Relationships & Cardinality:

- Members – Library Cards:
    - Relationship: has
        - A member has to have a library card
        - A library card
    - Cardinality: one-to-one

Schemas:

Library Card
cardID (primary key)
memberID (foreign key references Member(memberID)
creationDate
expirationDate
fees

Borrowed Movies - S

Fields:

- Due_date (attribute) – date
- Return_date (attribute) – date
- MovieID (Primary Key) (Foreign Key from Movies) – int
- memberID (Foreign Key from Member) – int

Relationships & Cardinality:

- Member – Borrows – Movie
  - Relationship: borrow
    - A member can borrow many movies over time
    - A movie can be borrowed by many members over time
  - Cardinality: many-to-many (M:N)

Schemas:

Borrowed Movies
MovieID (Primary Key) (Foreign Key references Movies(MovieID)
memberID (Foreign Key references Member(memberID)
Due_date
Return_date

Borrowed Books - J

Fields:

-borrowed_book_id

-due _date

-return_date

-memberId(refers to members.membersID)

-bookId(refers to books.bookID)

Relationships & Cardinality:

-it has a many to many relationship with books

-it has a many to many relationship with members

Schemas:

Borrowed books
orrowed_book_id
memberid(foreign key references Member(memberID)
bookID (foreign key references Books(bookID)
-due_date
-return_date

Check Outs - J

Fields:

-check_outs_id

-checkout _date

-memberId(refers to members.membersID)

-employeeId(refers to employee.employee_ID)

Relationships & Cardinality:

-it has a many to many relationship with employees

-it has a many to many relationship with members

Schemas:

Checkouts
Check_outs_id
memberID (foreign key references Member(memberID)
employeeID (foreign key references employees(emplployee_ID)

checkout_date