Step (2)

1. **Identify Entities: Based on the provided information, we can identify the main entities that will be part of your library database. Some possible entities include:**

   - *Item*: Represents items in the library, such as print books, online books, magazines, scientific journals, CDs, records, etc.
   - *User*: Represents library users.
   - *Personnel*: Represents library personnel.
   - *HelpRequest*: Represents help requests submitted by library users.
   - *Event*: Represents library events, such as book clubs, book-related events, art shows, film screenings, etc.
   - *Room*: Represents library social rooms where events are held.

2. **Define Attributes: For each entity, list the relevant attributes. Here are some attributes for each entity:**

   - Item:
     - Item_ID (Primary Key): Unique identifier for the item.
     - Title: Title of the item.
     - Type: Type of the item (e.g., print book, online book, magazine, etc.).
     - Author/Artist: Name of the author or artist (for books, CDs, records, etc.).
     - Genre: Genre or category of the item.
     - Available_Copies: Number of available copies in the library.

   - Future_Items:
     - Item_ID(Primary Key): Unique identifier for the item.
     - Type: Type of the item (e.g., print book, online book, magazine, etc.).
     - Title:Title of the item.
     - Available_Date: Date of availability in the library.

   - Loan:
     - Loan_ID(Primary Key): Unique identifier for the loan record.
     - Item_ID(Foreign Key): Unique identifier for the item being borrowed, links to the Item entity.
     - User_ID(Foreign Key): Unique identifier for the user borrowing the item, links to the User entity.
     - Borrow_Date: Date when the item was borrowed.
     - Due_Date: Date when the item is due to be returned.
     - Return_Date: Date when the item was actually returned.
     - Fine: Amount of fine incurred for returning the item late, if any.

   - User:
     - User_ID (Primary Key): Unique identifier for library users.
     - Name: Name of the user.

    - Address: Address of the user.
    - Email: Email address of the user.
    - Total_Fines: Total amount of fines on the user's account.

- Personnel:
    - Personnel_ID (Primary Key): Unique identifier for library personnel.
    - Name: Name of the personnel.
    - Job_Title: Job title of the personnel.

- Help_Request:
    - Request_ID (Primary Key): Unique identifier for the help request.
    - Name: Name of the user submitting the help request.
    - Contact: Contact information for the user submitting the help request.
    - Question: The user's question or request for help.

- Event:
    - Event_ID (Primary Key): Unique identifier for the event.
    - Event_date: Date when the event is scheduled to take place.
    - Room_ID(Foreign Key): Unique identifier for the social room where the event is being held, links to the Room entity.
    - Title: Title of the event.
    - Description: Description of the event.
    - Audience: Specific audience for which the event is recommended.

- Room:
    - Room_ID (Primary Key): Unique identifier for the social room.
    - Location: Location of the social room in the library.

3. **Define Relationships: Next, establish relationships between the entities. Some relationships might include:**

- **Borrow Relationship (Between Person and Item):**
  Multiplicity: Many-to-Many (M:N)
  Explanation: Multiple persons can borrow multiple items from the library, and each person can borrow multiple items, and each item can be borrowed by multiple persons. This results in a many-to-many relationship. The relationship is represented by the "Loan" entity, which serves as an association table between "Person" and "Item."

- **Organize Relationship (Between Personnel and Event):**
  Multiplicity: One-to-Many (1:N)
  Explanation: One personnel can organize multiple events, but each event is organized by only one personnel. This results in a one-to-many relationship.

- **Attend Relationship (Between Person and Event):**

Multiplicity: Many-to-Many (M:N)
Explanation: Multiple persons can attend multiple events, and each person can attend multiple events, and each event can be attended by multiple persons. This results in a many-to-many relationship. The relationship is represented by the "Event_Attendees" entity, which serves as an association table between "Person" and "Event."

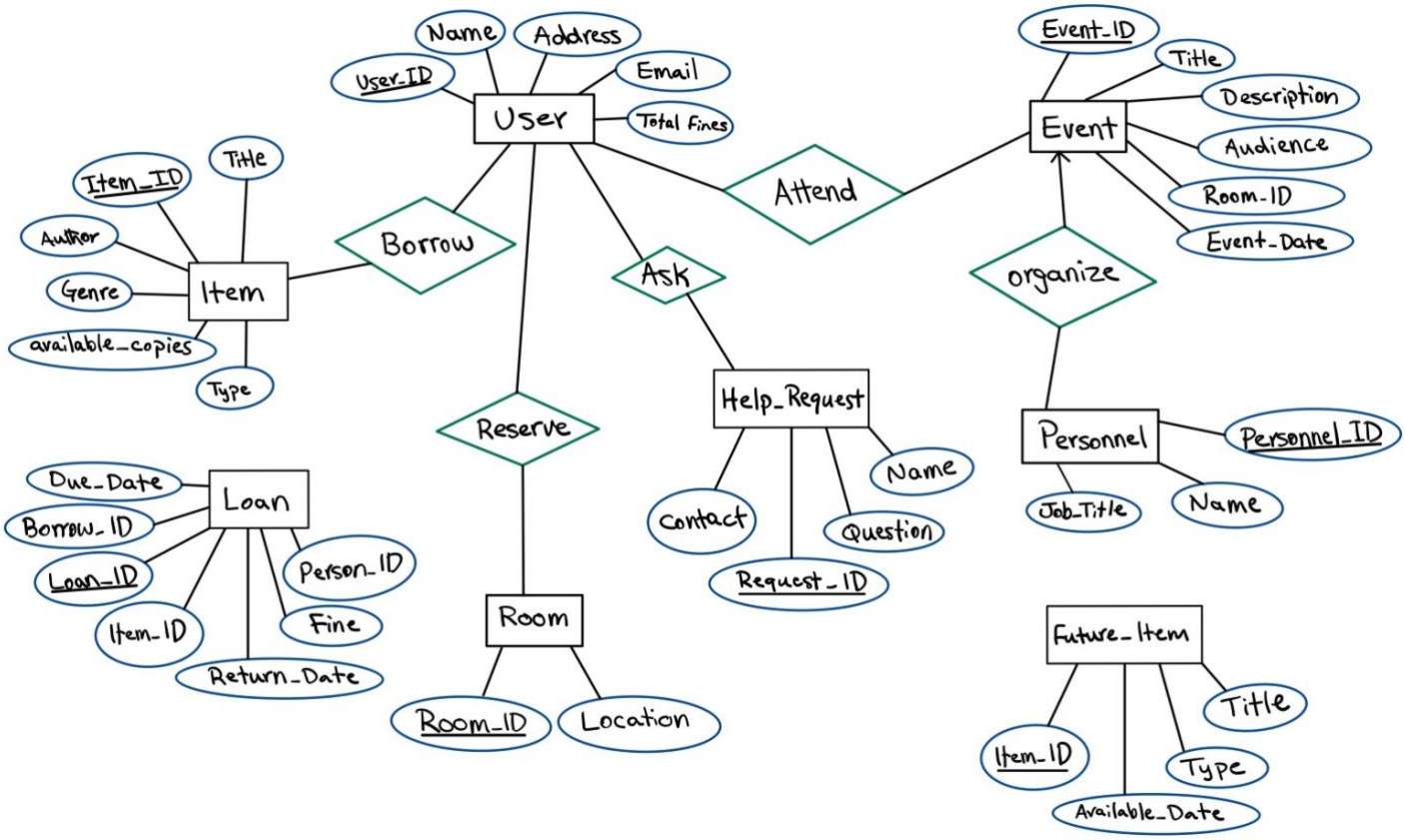
- **Ask Relationship (Between User and HelpRequest):**
  Multiplicity: One-to-Many (1:N)
  Explanation: One user can submit multiple help requests, but each help request is submitted by only one user. This results in a one-to-many relationship.

- **Reserve Relationship (Between User and Room):**
  Multiplicity: Many-to-Many (M:N)
  Explanation: Multiple users can reserve multiple rooms, and each user can reserve multiple rooms, and each room can be reserved by multiple users. This results in a many-to-many relationship. The relationship is represented by the "Reservation" entity, which serves as an association table between "User" and "Room."

4. **Key and Participation Constraints: Specify the primary key(s) for each entity and any participation constraints.**

- Item_ID is the primary key for the Item entity.
  - User_ID is the primary key for the User entity.
  - Personnel_ID is the primary key for the Personnel entity.
  - Request_ID is the primary key for the HelpRequest entity.
  - Event_ID is the primary key for the Event entity.
  - Room_ID is the primary key for the Room entity.

Step (3)


Name
Address
User_ID
Email
User
Total Fines
Item_ID
Title
Author
Genre
Item
available_copies
Type
Borrow
Attend
Event_ID
Title
Description
Event
Audience
Room_ID
Event_Date
Ask
organize
Reserve
Help_Request
Name
Contact
Question
Request_ID
Personnel
Personnel_ID
Job_Title
Name
Due_Date
Loan
Borrow_ID
Loan_ID
Person_ID
Item_ID
Fine
Return_Date
Room
Room_ID
Location
Future_Item
Title
Item_ID
Type
Available_Date

Step (4)

1. **Identify Functional Dependencies (FDs):**
    - Item Entity:
        - Item_ID -> Title, Type, Author/Artist, Genre, Available_Copies, Due_Date
    - User Entity:
        - User_ID -> Name, Address, Email, Total_Fines
    - Personnel Entity:
        - Personnel_ID -> Name, Job_Title
    - Event Entity:
        - Event_ID -> Title, Description, Audience, Event_date, Room_ID
    - Room Entity:
        - Room_ID -> Location
    - Loan Entity:
        - Loan_ID -> Item_ID, User_ID, BorrowDate, DueDate, ReturnDate, Fine
    - Future_Item Entity:
        - Item_ID -> Type, Title, Available_Date
    - Help_Request Entity:
        - Request_ID -> Name, Contact, Question

2. **Verify Primary Keys:**
    - All entities have a primary key attribute that uniquely identifies their records.

3. **Check for Partial Dependencies:**
    - There are no partial dependencies. All non-key attributes in each entity are fully functionally dependent on the primary key.

4. **Check for Transitive Dependencies:**
    - There are no transitive dependencies between non-key attributes. Each non-key attribute is directly and solely dependent on the primary key.

5. **Apply Boyce-Codd Normal Form (BCNF):**
    - All entities satisfy BCNF. Each entity has a primary key, and all non-key attributes are fully functionally dependent on the primary key. There are no partial dependencies or transitive dependencies.

6. **Review Relationships:**
    - The relationships “Borrow,” “Organize,” and “Attend,” “Reserve,” “Ask” are correctly established between entities.

7. **Verify Anomaly-Free Design:**

- Based on the analysis, the database design does not allow anomalies. It satisfies the requirements discussed in the project specifications and ensures data integrity and consistency.

Analysis Results: After analysis of the library database design, we can conclude that the design does not allow anomalies. All tables are in Boyce-Codd Normal Form (BCNF), meaning that they are free from any update, insertion, or deletion anomalies. Each entity's primary key uniquely identifies its records, and all other attributes are fully functionally dependent on the primary key.

The relationships between entities are well-defined, and the design maintains data integrity throughout the system. The library database application can be confidently developed based on this anomaly-free design.