**Entities and Attributes:**

* **Donor:**  
  Attributes: DonorID (Primary Key), Name, ContactInfo, DonationHistory, Preferences.
* **Donation:**  
  Attributes: DonationID (Primary Key), Amount, Date, PaymentMethod, DonorID (Foreign Key).
* **Campaign:**  
  Attributes: CampaignID (Primary Key), Goal, FundsRaised, StartDate, EndDate.
* **Event:**  
  Attributes: EventID (Primary Key), Name, Date, Location, CampaignID (Foreign Key).
* **Receipt:**  
  Attributes: ReceiptID (Primary Key), DonationID (Foreign Key), Date, Amount.
* **Admin:**  
  Attributes: AdminID (Primary Key), Name, Role, Permissions.

**Relationships and Cardinalities:**

* **Donor → Donation:** One-to-Many (One donor can make multiple donations).
* **Donation → Campaign:** Many-to-One (Each donation supports one campaign).
* **Campaign → Event:** One-to-Many (One campaign can involve multiple events).
* **Donation → Receipt:** One-to-One (Each donation has a unique receipt).
* **Admin → All Entities:** One-to-Many (Admins manage data for all entities).

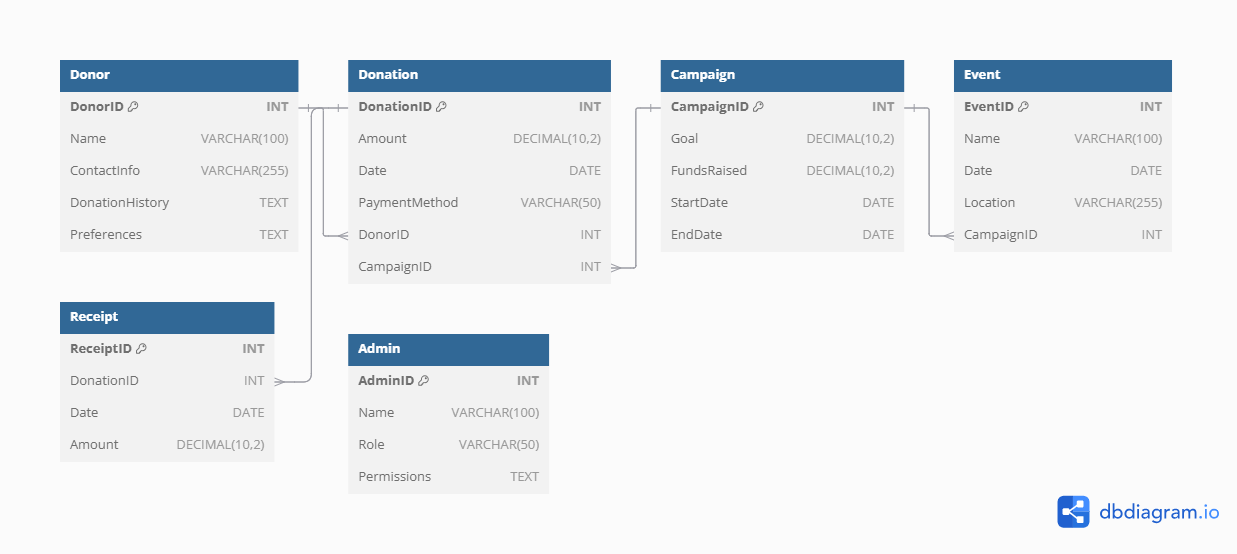
|  |  |  |  |
| --- | --- | --- | --- |
| Table Name | Fields | Primary Key (PK) | Foreign Key (FK) |
| Donor | DonorID, Name, ContactInfo, DonationHistory, Preferences | DonorID | None |
| Donation | DonationID, Amount, Date, PaymentMethod, DonorID | DonationID | DonorID → Donor(DonorID) |
| Campaign | CampaignID, Goal, FundsRaised, StartDate, EndDate | CampaignID | None |
| Event | EventID, Name, Date, Location, CampaignID | EventID | CampaignID → Campaign(CampaignID) |
| Receipt | ReceiptID, DonationID, Date, Amount | ReceiptID | CampaignID → Campaign(CampaignID) |
| Admin | AdminID, Name, Role, Permissions | AdminID | None |

* **Donor Table**:  
  Stores details about donors, uniquely identified by DonorID.
* **Donation Table**:  
  Captures details of donations, with each donation linked to a donor using DonorID as a foreign key.
* **Campaign Table**:  
  Tracks fundraising campaigns, uniquely identified by CampaignID.
* **Event Table**:  
  Logs details of events related to campaigns, with CampaignID as a foreign key linking events to their respective campaigns.
* **Receipt Table**:  
  Manages tax receipts for donations, with DonationID as a foreign key to link each receipt to a specific donation.
* **Admin Table**:  
  Stores details of administrators managing the system, uniquely identified by AdminID.

**Challenges Faces:**

* **Ensuring Data Integrity**: Maintaining data accuracy and consistency across tables.
* **Handling Many-to-Many Relationships**: Managing complex relationships that require intermediary tables.
* **Normalization vs. Performance**: Balancing between reducing redundancy and ensuring fast query performance.
* **Defining Clear Relationships**: Determining and establishing the correct type of relationships (one-to-one, one-to-many, many-to-many) between entities.
* **Handling Redundancy and Data Duplication**: Preventing unnecessary duplication of data across tables.
* **Choosing Appropriate Data Types**: Selecting the correct data types for fields to ensure efficient storage and retrieval.
* **Handling Null Values**: Deciding when to allow or disallow null values in fields.

**Relational Model**



**Brief Report: Design Decisions and Relational Model**

**Introduction**

The Charity Management System is designed to support various operations of a charity organization, including tracking donations, managing fundraising campaigns, handling events, and generating receipts. The goal of the system is to automate and streamline the operations, ensuring accurate and efficient management of resources. The relational data model (RDM) for this system is designed to capture entities like **Donors**, **Donations**, **Campaigns**, **Events**, **Receipts**, and **Admins**, and the relationships between them.

This report outlines the design decisions made, explains how the relational model supports the system requirements, and discusses challenges and trade-offs encountered during the design process.

**1. Design Decisions**

The design of the database involves breaking down the charity management system into entities and their relationships. The following are key design decisions made to structure the system:

* **Entities**: The entities are selected based on the system's functional requirements:
  + **Donor**: Stores information about individuals contributing to the charity. Key attributes include DonorID, Name, ContactInfo, DonationHistory, and Preferences.
  + **Donation**: Tracks donation details, including the amount, date, payment method, and links to a specific donor and campaign.
  + **Campaign**: Represents the fundraising campaigns organized by the charity. Each campaign has a goal, funds raised, and start and end dates.
  + **Event**: Tracks events associated with campaigns, including event name, date, and location.
  + **Receipt**: Captures the details of the receipt generated for each donation, including the amount and the donation ID.
  + **Admin**: Represents the staff managing the system, with attributes like AdminID, Name, Role, and Permissions.
* **Attributes and Data Types**:
  + Fields such as **Name**, **ContactInfo**, and **DonationHistory** are given VARCHAR types to store textual data.
  + Monetary values like **Amount** and **Goal** are given a DECIMAL type with a precision of 10 digits and 2 decimal places.
  + **Date** is stored as DATE to keep track of donation, event, and campaign timings.
* **Primary and Foreign Keys**:
  + Each table has a primary key (e.g., DonorID, DonationID, CampaignID) to uniquely identify records.
  + Foreign keys are used to establish relationships between tables, ensuring referential integrity. For instance:
    - **Donation.DonorID** references **Donor.DonorID** to link donations to donors.
    - **Donation.CampaignID** references **Campaign.CampaignID** to associate each donation with a campaign.
    - **Event.CampaignID** references **Campaign.CampaignID** to tie events to specific campaigns.
    - **Receipt.DonationID** references **Donation.DonationID** to link receipts to donations.

**2. Supporting System Requirements**

The relational model has been designed to meet the core functional requirements of the charity management system:

* **Efficient Donation Tracking**: The model ensures that every donation is associated with a donor and a campaign. The **Donation** table captures essential information such as amount, payment method, and date. By linking the **Donor** and **Campaign** tables via foreign keys, it is easy to track who donated and to which campaign.
* **Campaign and Event Management**: The **Campaign** and **Event** tables enable tracking of each campaign's progress and the events tied to them. The **Event** table's connection to the **Campaign** table allows for reporting on campaign-related events and their details.
* **Receipt Generation**: Each donation generates a unique receipt, which is stored in the **Receipt** table. This ensures the charity can provide donors with official documentation for their donations.
* **Admin Access Control**: The **Admin** table stores the roles and permissions of the staff managing the system. This table is essential for managing who has access to different parts of the system, ensuring security and proper authorization.

**3. Challenges and Considerations**

Several challenges and considerations were encountered during the design process:

* **Many-to-Many Relationships**: While the relationships between **Donor** and **Donation** (one-to-many) and **Donation** and **Receipt** (one-to-one) are straightforward, certain relationships like **Donor-to-Event** or **Event-to-Campaign** could potentially become many-to-many. However, these relationships were not required in this specific design. If future requirements demand such relationships, intermediary tables (e.g., **DonorEvent** for many-to-many between donors and events) could be added.
* **Data Redundancy**: One consideration was minimizing redundancy in donation records. By linking donations to campaigns and donors through foreign keys, we ensured that no redundant data is stored. The use of **DonationHistory** in the **Donor** table captures the cumulative record of donations, preventing multiple entries for the same donor.
* **Dynamic Campaign Goals**: Campaign goals and funds raised can change over time, requiring updates. The relational model accounts for this by storing campaign data in the **Campaign** table, allowing for easy updates of the campaign's goal and funds raised without affecting other parts of the system.
* **Receipt Uniqueness**: The challenge of ensuring unique receipt generation for each donation was addressed by creating a one-to-one relationship between **Donation** and **Receipt**, guaranteeing that each donation has one unique receipt.

**4. Trade-offs**

* **Normalization vs. Performance**: In some cases, normalization was prioritized to reduce redundancy and maintain data integrity. However, this could impact performance in larger systems due to the number of joins needed between tables. For example, retrieving a donor's complete donation history involves multiple joins between the **Donor** and **Donation** tables.
* **Future Scalability**: The current design ensures that the system can scale by adding new campaigns, events, and donations without major structural changes. However, adding more complex features like multi-currency donations or larger event management will require further modifications.

**Conclusion**

The relational data model designed for the charity management system supports the functional requirements of tracking donations, managing campaigns and events, and generating receipts. By defining clear relationships and using appropriate data types, the system ensures data integrity and efficiency. The challenges encountered, such as minimizing data redundancy and ensuring unique receipt generation, were addressed through careful relational design decisions. Although some trade-offs regarding performance and scalability were made, the current design provides a robust foundation for managing charity operations effectively.