Contents

[1.Introduction 3](#_Toc202133852)

[2.Database Requirements 3](#_Toc202133853)

[User requirements 3](#_Toc202133854)

[System Requirements 4](#_Toc202133855)

[Data requirements 4](#_Toc202133856)

[3.Database Design 5](#_Toc202133857)

[Conceptual Design 5](#_Toc202133858)

[Definition 5](#_Toc202133859)

[Objective 5](#_Toc202133860)

[Components 5](#_Toc202133861)

[My Conceptual Design 5](#_Toc202133862)

[Mapping and schema 6](#_Toc202133863)

[My attributed with their types 6](#_Toc202133864)

[Step1: Mapping of Regular (strong) Entity Types 7](#_Toc202133865)

[Step2: Mapping of Weak Entity Types 8](#_Toc202133866)

[Step3: Mapping of 1:1 Relation types 8](#_Toc202133867)

[Step4: Mapping of 1:M Relation types 8](#_Toc202133868)

[Step5: Mapping of M:N Relation types 8](#_Toc202133869)

[Final schema 8](#_Toc202133870)

[Logical design 9](#_Toc202133871)

[Goals 9](#_Toc202133872)

[Components 9](#_Toc202133873)

[My logical design 10](#_Toc202133874)

[Physical Design 11](#_Toc202133875)

[Goals 11](#_Toc202133876)

[Components 11](#_Toc202133877)

[My Physical Design 12](#_Toc202133878)

[Effectiveness of the design 13](#_Toc202133879)

[4.Normalization 14](#_Toc202133880)

[1st NF 14](#_Toc202133881)

[2nd NF 15](#_Toc202133882)

[3rd NF 16](#_Toc202133883)

[5.Physical Schema 17](#_Toc202133884)

[6.Database Development 18](#_Toc202133885)

[Database Overview 18](#_Toc202133886)

[Tables Overview 18](#_Toc202133887)

[Views Overview 22](#_Toc202133888)

[Procedures Overview 23](#_Toc202133889)

[Security 24](#_Toc202133890)

[User Interface 26](#_Toc202133891)

[Flowchart and data movement diagrams 26](#_Toc202133892)

[Interfaces Development 29](#_Toc202133893)

[7.Maintenance 30](#_Toc202133894)

[Database recovery & backups 30](#_Toc202133895)

[My Database recovery and backup 31](#_Toc202133896)

[Database maintenance in general 35](#_Toc202133897)

[8.Testing 36](#_Toc202133898)

[Data validation 36](#_Toc202133899)

[Output validation 40](#_Toc202133900)

[Security Validation 43](#_Toc202133901)

[GUI Validation 45](#_Toc202133902)

[Assess whether meaningful data has been extracted 49](#_Toc202133903)

[Relationships 49](#_Toc202133904)

[My views and procedures 50](#_Toc202133905)

[Effectiveness of testing 53](#_Toc202133906)

[References 54](#_Toc202133907)

# 1.Introduction

This report presents the technical documentation for the development of the donor database system for EDU Youth Foundation. It aims to manage donor information, events, and communication efficiently. The report describes the design, implementation, and functionality of the database, hence acting like a guide on how the system works in accomplishing the mission of the organization, which is quality education. The database has been designed to enhance organizational efficiency, facilitate access to data, and enable decision-making within the organization.

This report provides detailed information on how the database has been structured, what functionality it achieves, and how it meets the needs of the organization. This will include discussion on the system architecture, the tools and technologies adopted, the methodology followed in developing the system, and how certain features have been specifically implemented to address the organization's needs. This report serves as a reference not only to the current stakeholders but also for any future upgrade and scalability of the system. The intention of this report is to make the database system maximally usable and impactful by providing clear and structured insight into it.

# 2.Database Requirements

## User requirements

* Donor:

The donor will be able to:

**1. Administrator:**

* Complete access to all database functions.
* Manage donor records, including adding, updating, and deleting information.
* Manage staff member and their roles.
* Maintain system security by managing user roles and permissions.

**2. Employee:**

* View donor information but connect add new or remove records.
* Track donations and participation history.
* Generate donor-related reports to support strategic decision-making.
* Communicate with the donors.

**3. Event Manager**

* Schedule and update event details, such as location, date, participants.
* Allocate donations to specific events and manage event finances.

Record location engagement and generate event summary reports.

**4. Data Analyst**

* Retrieve information regarding donors, donations, and further analyze to forecast trend.
* Execute queries and generate insights for organizational growth.
* Evaluate event success using data visualization and statistical techniques.

## System Requirements

* Software Requirements:
* **Operating System:** The operating system should be compatible with Windows, macOS, and Linux.
* **Database:** The system should be built using a relational database management system such as MySQL
* Hardware Requirements:
* **Server**: The system needs a web server to host the application.
* **Storage:** The system should have enough free disk space to store information about donors, logs of communications, event files, and records of donations made. This depends on the base size of donors.
* **Backup:** The system should have a reliable backup system for database and file backups to ensure data integrity and recovery.

## Data requirements

* The system is designed to efficiently manage donor information, staff details, events, donations, and their interrelated connections. Each donor has a unique identification number (Donor ID) and their information includes their first and last name, nationality, social security number (SSN), registration date, email, multiple phone numbers, address ( country, city, and street name), and date of birth. If a donor does not provide their nationality, it defaults to “Jordanian.”
* Staff members are an essential part of the system =, and their details include a unique Staff ID, first and last name, salary, level (Administrator, employee), email, phone numbers, date of birth, and address. Like donors, staff members can have multiple phone numbers, which are recorded separately.
* Events are another core component of the system. Each event has a unique ID, a name, a scheduled date, a venue name, and an address consisting of city and street name. Donors can participate in multiple events, and each participation record tracks the donor’s ID, the event ID, and the date of participation.
* Donations are vital for supporting events and are tracked comprehensively. Each donation has a unique ID, a donor ID linking it to the donor who made the contribution, the amount donated, and the date of the donation. The amount for every donation must be greater than (100jd), and donations can be allocated to one or more events. Donation allocations are tracked using a dedicated record linking the donation ID to the event ID. This ensures transparency in how donations are utilized for different events.

# 3.Database Design

## Conceptual Design

### Definition

Conceptual design is the abstract representation of a system or database that captures the very essence of the data and their interrelationships. It provides explicit concentration on the understanding and determination of the business needs and their translation into an effective logical structure. The conceptual design does not concern itself with the details of technical implementation but, in fact, finds the entities, their attributes, and relationships in a manner that could easily be comprehended by the stakeholders.

### Objective

The aim of the conceptual design is to bridge the gap between business requirements and technical implementation. It aims to:

1. Clearly define the structure and scope of the data.
2. Identify key data elements and their relationships.
3. Ensure alignment with organizational objectives and user needs.
4. Provide a basis for subsequent logical and physical database designs.

### Components

The major components of a conceptual design include:

* **Entities:** Represent the major objects or concepts in the system. In my case, donors, events, communication, and donations.
* **Attributes:** Properties or characteristics of the entities. For example, donor name, event date.
* **Relationships:** Links between entities. For example, donors sponsoring events, events having attendees.

### My Conceptual Design

A diagram of a flowchart

Description automatically generated

The conceptual design above provides an overview of the entities and relationships in this database. For example, **Donor** and **Event** share an **M: N relationship**, indicating that one donor can attend multiple events, and any event can have multiple attendees. This is represented by the **attend** connection. The entities **Donor** and **Donation** have a **1:M relationship**, as one donor can make multiple donations, represented by the **make** connection. The **Donor** and **Staff** entities also have a **1:M relationship**, where one staff member can manage multiple donors, depicted by the **communicate** connection. Additionally, the entities **Donation** and **Event** share an **M:N relationship**, where one donation can contribute to multiple events, and an event can receive funds from multiple donations. These relationships ensure accurate representation of interactions and maintain data integrity in the system.

## Mapping and schema

### My attributed with their types

* **Donor** (Donor ID, ssn, Donor name, Registration date, Email, phone number, Adress, Birth date, nationality, Age)
* **Staff** (Staff ID, name, level, salary, Phone, Address, email, birth date, Age)
* **Event** (Event ID, Event name, Event date, Event location)
* **Communication** (communication ID, Type, Date)
* **Donation** (Donation ID, Amount, Date)

|  |  |  |
| --- | --- | --- |
| Attribute | Type | Description |
| Entity1: Donor | | |
| Donor ID (PK) | Simple, Single-valued | Unique identifier for the donor |
| SSN | Simple, Single-valued | Social Security Number of the donor |
| Donor name | Composite | Consists of first name, middle name, and last name |
| Registration date | Simple, Single-valued | Date the donor registered in the system |
| Email | Simple, Single-valued | Email address of the donor |
| Phone number | Multivalued | A donor may have multiple phone numbers |
| Address | Composite | Includes street, city, state, and zip code |
| Birth date | Simple, Single-valued | Date of birth of the donor |
| Age | Derived | Age of the donor |
| nationality | Simple, single-valued | Where the donor from |
| Entity2: Staff | | |
| Staff ID (PK) | Simple, Single-valued | Unique identifier for the staff member |
| Name | Composite | Consists of first name, middle name, and last name |
| Salary | Simple, Single-valued | Salary of the staff member |
| Level | Simple, Single-valued | Level or position of the staff member |
| Phone | Multivalued | A staff member may have multiple phone numbers |
| Address | Composite | Includes street, city, state, and zip code |
| Email | Simple, Single-valued | Email address of the staff member |
| Birth date | Simple, Single-valued | Date of birth of the staff member |
| Age | derived | Age of the staff |
| Entity3: Event | | |
| Event ID (PK) | Simple, Single-valued | Unique identifier for the event |
| Event name | Simple, Single-valued | Name of the event |
| Event date | Simple, Single-valued | Date when the event will take place |
| Event location | Composite | Includes venue name, city, and zip code |
| Entity 4: Donation | | |
| Donation ID ( Pk) | Simple, Single-valued | Unique identifier for the donation |
| Amount | Simple, Single-valued | Amount of the donation |
| Date | Simple, Single-valued | Date when the donation was made |

### Step1: Mapping of Regular (strong) Entity Types

In step 1, the strong entity should be mapped by resolving composite attributes and separating them into simpler attributes. Additionally, any multi-valued or complex attributes should be extracted and moved to a new, separate table. Finally I will delete the derived attributes.

* **Donor** (Donor ID, SSN, Nationality, Donor First Name, Donor Last Name, Registration Date, Email, Birth Date, country, city, street name)
* **Donors Phone** (Donor ID, Phone Number)
* **Donation** (Donation ID, Amount, Date)
* **Staff** (Staff ID, Staff First Name, Staff Last Name, Salary, Level, Email, Birth Date, country, city, street name)
* **Staff Phone** (Staff ID, Phone Number)
* **Event** (Event ID, Event Name, Event Date, Venue Name, City, street name)

### Step2: Mapping of Weak Entity Types

There is no weak entity in my scenario

### Step3: Mapping of 1:1 Relation types

In my scenario there is no 1:1 relation

### Step4: Mapping of 1:M Relation types

In this step we should make a link between tables by adding the PK of the (1) relation table to the many relations table as FK:

**Donor** (Donor ID, Staff ID, SSN, Donor First Name, Donor Last Name, Registration Date, Email, Birth Date, Country, City, Street Name)

**Donation** (Donation ID, Donor ID, Amount, Date)

### Step5: Mapping of M:N Relation types

In this step, we will represented as new table. And the primary key of this table will be combination of the primary keys of the two ends of the relation:

**Participation** (Donor ID, Event Id, participation date).

**Donation Allocation** (Event ID, Donation ID)

### Final schema

**Donor** (Donor ID, Staff ID, Nationality, SSN, Donor First Name, Donor Last Name, Registration Date, Email, Birth Date, Country, City, Street Name)

**Donors Phone** (Donor ID, Phone Number)

**Staff** (Staff ID, Staff First Name, Staff Last Name, Salary, Level, Email, Birth Date, Country, City, Street Name)

**Staff Phone** (Staff ID, Phone Number)

**Event** (Event ID, Event Name, Event Date, Venue Name, City, Street Name)

**Donation** (Donation ID, Donor ID, Amount, Date)

**Participation** (Donor ID, Event ID, Participation Date)

**Donation Allocation** (Event ID, Donation ID)

## Logical design

The logical design delineates how the conceptual database is organized into well-structured entities and relationships. The data should be stored in a normalized format, so redundancy is avoided and there is enhancement in consistency. It involves taking the identified entities, attributes, and their relationships from the conceptual model into relational tables. It will provide key elements such as primary keys to uniquely identify the records, foreign keys that link tables together, and associative tables that maintain many-to-many relationships.

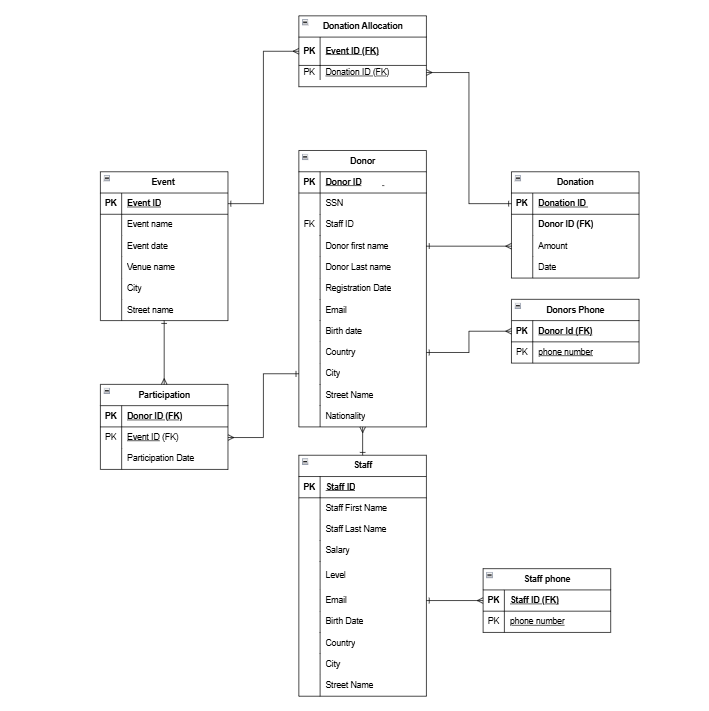
### Goals

The primary goal of logical design is to ensure the database structure accurately reflects the real-world business processes, relationships, and requirements while being easy to maintain and scalable. It aims to achieve data integrity, consistency, and clarity while preparing for physical implementation.

### Components

* Entities
* Attributes
* Relationships

### My logical design



Above is the logical design with all the attributes, keys, and the relationships that exist among the entities of the database. The relationship of Donor to Donors Phone would be 1:M-for one donor, there might be many phone numbers-while for Donor to Donation, the relationship would be 1:M too, since one donor can give multiple donations. The Participation entity resolves the many-to-many relationship between the entities Donor and Event; it contains a composite primary key, which includes two foreign keys: one coming from the Donor entity, namely, Donor ID, and another from the Event entity, namely, Event ID. This design ensures that all relationships are well-defined and that primary and foreign keys have been used to maintain data integrity and correctly represent the real-world links of these entities.

## Physical Design

Physical design is the process of creating a clear plan showing how a system will work in its technical setting based on the logical design. This will also involve decisions on how data will be stored, defining data types for each attribute, and setting rules such as primary keys, foreign keys, unique constraints, and validation rules that will ensure the data is accurate and consistent. It also involves setting up the system to satisfy the needs of performance, reliability, and scalability, in mind of the underlying hardware and software environments. (Banister)

### Goals

The main goal of physical design is to make sure the system works correctly, is reliable, and can be scaled up within the hardware and software chosen. Within the given budget and resource constraints, this involves system performance optimization, data security, and usability.

### Components

* Entities
* Attributes
* Relationships
* Data types
* Constrains

### My Physical Design

A diagram of a program

Description automatically generated with medium confidence

## Effectiveness of the design

**Selection of Entities and Attributes**

My database consists of four main entities: **Donor**, **Staff**, **Event**, and **Donation**. The entity **Donor** includes all the important attributes needed for identifying and communicating with a donor. For example, the **Donor** entity has a Donor ID, Nationality, SSN, First Name, Last Name, Registration Date, Email, Birth Date, Country, City, and Street Name. The **Staff** entity is an example of staff people who will manage the information about donors and communicate with them. It has attributes such as Staff ID, First Name, Last Name, Salary, Level, Email, Birth Date, Country, City, and Street Name. The **Event** entity is used to track events that donors can attend and donate. Attributes in this entity provide information on each event. Lastly, the **Donation** entity tracks the donations made by donors, and the amount and date of donation. These entities ensure that the system manages donors, staff, events, and donations appropriately.

**Establishment of Relationships**

The relationships in this design accurately reflect real-world interactions. The **1:M relationship** between **Donor** and **Donation** illustrates that a donor can make multiple donations, allowing for detailed tracking of donation history. The **M:N relationship** between **Donor** and **Events**, implemented via a **Participation table**, captures the realistic scenario where a donor can attend multiple events and events can have multiple attendees. Similarly, the **M:N relationship** between **Events** and **Donations**, represented through a **Donation Allocation table**, highlights that a single donation can be distributed across multiple events, and an event can receive funds from various donations. Additionally, the **1:M relationship** between **Staff** and **Donor** ensures that each staff member can effectively communicate with and manage multiple donors, facilitating a streamlined approach to donor management. This design ensures data integrity, accurate representation of interactions, and ease of access, without requiring any **1:1 relationship**, as no entity pairs demand exclusivity.

**Conceptual Design and Its Intended Audience**

The conceptual design was created to give a clear and high-level representation of the system's entities and their relationships. It helps simplify complex ideas and ensures everyone understands the system's goals. The target audience includes project managers, developers, and other stakeholders who need a shared understanding to guide the project effectively.

**Schema Creation and Its Importance**

The schema was created to ensure the rational structure of data. It acts as an interface between the conceptual design and the physical implementation and provides a structure for the storing, accessing, and interrelation of data. Providing primary keys like Donor ID and Event ID, the schema enforces consistency and integrity of data via foreign key constraints that maintain the relationships. with well-defined parameters of the schema, integration and expandability happen seamlessly.

**Logical Design Specifications**

The logical design included specific details such as normalization of attributes and interrelations among the tables, and the inclusion of foreign keys. For instance, the separation of Donor Phone into a standalone table handled multi-valued attributes thus upholding the principles of a database. This design phase was very critical in executing complex queries and ensuring the reliability of the database when carrying out processes such as donor reporting and event tracking.

**Physical Design Enhancements**

The physical design specified data types for each attribute and incorporated indexing for performance optimization. Attributes like Donor ID were defined as integers to allow efficient storage and retrieval, while text fields like Email were optimized for length and format. Backup and recovery mechanisms were also included to ensure data security and resilience, addressing the organization’s need for reliability and continuity.

**Conclusion**

The integration of these processes—right from entity selection to normalization—ensured that the database design met all the requirements of the organization in a comprehensive manner. The design, by considering both the user requirements, including monitoring donations and coordinating events, and the system requirements, including data integrity and scalability, lays a proper foundation for the EDU Youth Foundation.

# 4.Normalization

### 1st NF

The First Normal Form ensures that the database schema adheres to the following principles:

1. Each cell in a table should contain a single, atomic value.
2. Each record in the table must be unique, with no duplicate rows.
3. Values stored in a column should belong to the same domain, ensuring data consistency.
4. All columns in a relation should have unique names to avoid ambiguity.

In this step, composite attributes are decomposed into atomic attributes, multi-valued attributes are extracted into separate relations, and any violations of 1NF are resolved. The table below identifies the violations in the schema and provides solutions for achieving 1NF compliance:

|  |  |  |  |
| --- | --- | --- | --- |
| Relations | Attributes | Violation description | Solution – Relations |
| Donor (Donor ID, SSN, Staff ID, Nationality, Donor Name, Registration Date, Email, Birth Date, Donor Adress, Donor phone) | 1)Adress  2)Donor phone | 1. **Multivalued attribute**: Donors may have multiple phone numbers.  2. **Composite attributes**: The Name and Address fields are not atomic | **Donor** (Donor ID, SSN, Staff ID, Nationality, Donor First Name, Donor Last Name, Registration Date, Email, Birth Date, Country, City, Street Name)  **Donor Phone** (Donor ID, Phone Number) |
| Event (Event ID, Event Name, Event Date, Venue Name, Adress) | 1) Adress | 1. **Composite attribute:** The Adress fields are not atomic | **Event** (Event ID, Event Name, Event Date, Venue Name, City, Street Name) |

### 2nd NF

The Second Normal Form (2NF) builds upon the foundation of the First Normal Form (1NF) and addresses partial dependencies. For a relation to be in 2NF, it must first satisfy all 1NF requirements. Additionally, all non-key attributes must be fully functionally dependent on the entire primary key, ensuring that there are no partial dependencies. This means that every non-key attribute should depend solely on the whole primary key and not on a subset of it. The table below identifies any violations of 2NF in the schema and details the steps taken to resolve them, ensuring a higher level of normalization and minimizing redundancy:

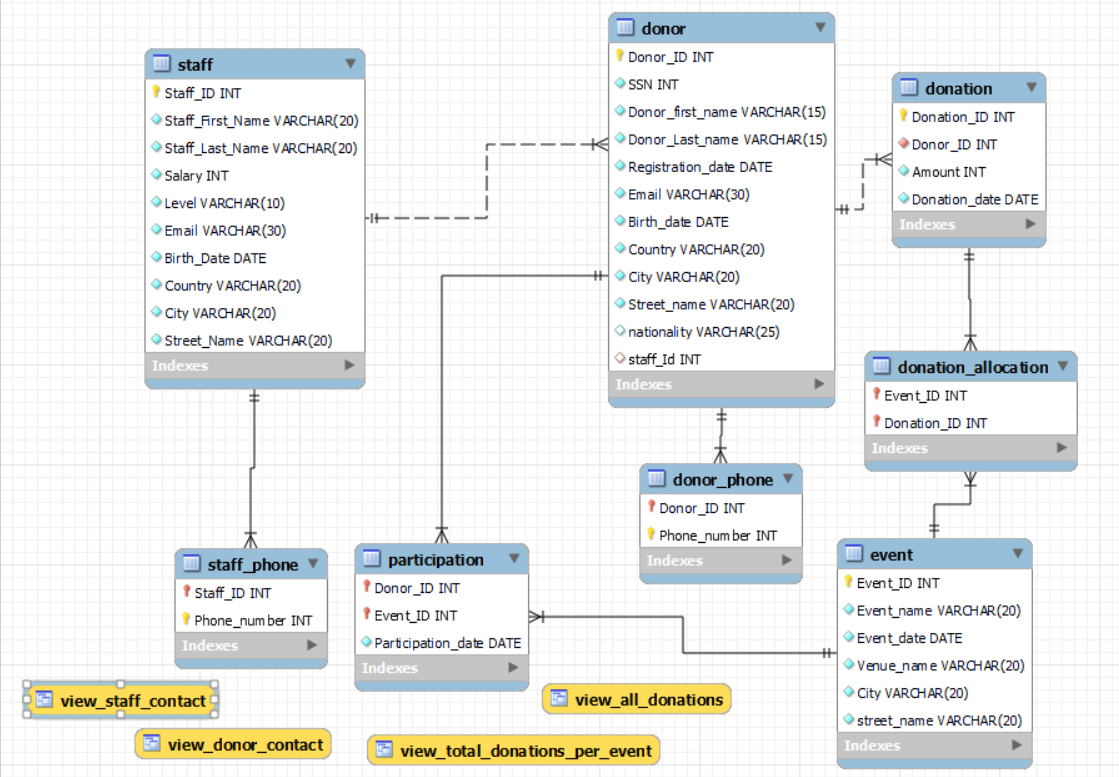
|  |  |  |  |
| --- | --- | --- | --- |
| Relations | FDs | Violation description | Solution – Relations |
| Donor (Donor ID, Donation ID, Donation amount, donation date, Staff ID, Nationality, SSN, Donor First Name, Donor Last Name, Registration Date, Email, Birth Date, Country, City, Street Name) | **Donation ID🡪**(Donor ID, Donation amount, donation date)  **Donor ID 🡪(** Staff ID, Nationality, SSN, Donor First Name, Donor Last Name, Registration Date, Email, Birth Date, Country, City, Street Name**)** | The non-prime attributes  (Donation ID, Donation amount, and donation date)  depend on a part of the key  (Donation ID), and the non-prime  attributes (Staff ID, Nationality, Donor First Name, Donor Last Name, Registration Date, Email, Birth Date, Country, City, Street Name, SSN) depend  on a part of the pk which is  (Donor ID)  So, we have partial  dependencies | **Donor** (Donor ID, Staff ID, Nationality, SSN, Donor First Name, Donor Last Name, Registration Date, Email, Birth Date, Country, City, Street Name)  **Donation** (Donation ID, Donor ID, Donation amount, donation date) |
| Event (Event ID, Donation ID, Event Name, Event Date, Venue Name, City, Street Name, Donation Amount, Donation Date, Donor ID) | **Event ID 🡪 (**Event ID, Event Name, Event Date, Venue Name, City, Street Name**)**  **Donation 🡪(** Donation ID, Donor ID, Amount, Date) | The non-prime attributes  (Event Name, Event Date, Event ID**,** Event Name, Event Date, Venue Name, City, And Street Name)  depend on a part of the key  (Event ID), and the non-prime  attributes ( Donor ID, Amount, Date) depend  on a part of the pk which is  (Donation ID)  So, we have partial  dependencies | **Event** (Event ID, Event Name, Event Date, Venue Name, City, Street Name)  **Donation** (Donation ID, Donor ID, Amount, Date) |

### 3rd NF

3NF further extends the definition of 2NF by eliminating transitive dependencies. For a relation to be in 3NF, it needs to be in 2NF, and any dependency of a non-key attribute is to occur only with candidate keys, never with other non-prime attributes. That is to say, all the attributes of the table have to depend upon the primary key alone for ensuring data integrity and avoiding data redundancy. The table below evaluates each relation for 3NF compliance, identifies any transitive dependencies, and provides solutions to restructure the schema accordingly:

|  |  |  |  |
| --- | --- | --- | --- |
| Relations | FDs | Violation description | Solution – Relations |
| Donor (Donor ID, Donation ID, Donation amount, donation date, Staff ID, Nationality, SSN, Donor First Name, Donor Last Name, Registration Date, Email, Birth Date, Country, City, Street Name) | Donor ID 🡪 (Donation ID)  Donation ID 🡪  (Donation amount, donation date)  Donor ID 🡪  (Donation amount, donation date) | The PK (Donor ID) can determine  any attribute in the table  including Donation ID. At the  same time, the Donation ID  determine Donation amount and  Donation date. As  Donation is not a candidate  here, we have a transitive  dependency. | **Donor (**Donor ID, Staff ID, Nationality, SSN, Donor First Name, Donor Last Name, Registration Date, Email, Birth Date, Country, City, Street Name**)**  **Donation** (Donation ID, Donor ID, Amount, Date) |
| Event (Event ID, Event Name, Event Date, Venue Name, City, Street Name, Donation ID, Donor ID, Amount, Date) | Event ID 🡪  (Donation ID)  Donation ID🡪  (**Donor ID, Amount, Date**)  Donor ID🡪  (**Donor ID, Amount, Date**)  ) | The PK (ID) can determine  any attribute in the table including course\_ID. At the same time, the course\_ID  determine courseName and  course\_Description. As  course\_ID is not a candidate  here, we have a transitive  dependency. | **Event** (Event ID, Event Name, Event Date, Venue Name, City, Street Name)  **Donation** (Donation ID, Donor ID, Amount, Date)  **Participation** (Donor ID, Event ID, Participation Date) |

# 5.Physical Schema



# 6.Database Development

## Database Overview

### Tables Overview

|  |  |  |
| --- | --- | --- |
| **Table** | **Name** | **Description** |
|  | Donor | This table is responsible for storing the donor’s information. The this attributes: (id, ssn, first\_name, last\_name, registration\_date, email, birth\_date, country, city, street\_name, nationality).   * **id**: The id attribute is the primary key of the donor table; it is used to search between different donors. This attribute is an integer and has a NOT NULL constraint. . * **ssn**: This attribute stores the Social Security Number of the donor. It is unique and has a NOT NULL constraint to ensure that the value is always entered. * **first\_name**: Stores the first name of the donor. This varchar attribute can hold up to 15 characters and has a NOT NULL constraint. * **last\_name**: Stores the last name of the donor. This varchar attribute can hold up to 15 characters and has a NOT NULL constraint. * **registration\_date**: Stores the date when the donor registered. This attribute is of type DATE and has a NOT NULL constraint. * **email**: Stores the email address of the donor. This unique attribute is a candidate key and can hold up to 30 characters. * **birth\_date**: Stores the birth date of the donor. This attribute is of type DATE and has a NOT NULL constraint. * **country**: Stores the country of the donor. This varchar attribute has a limit of 20 characters. And I has no null constraint * **city**: Stores the city of the donor. This varchar attribute has a limit of 20 characters . And its has no null constraint. * **street\_name**: Stores the street name of the donor’s address. This varchar attribute can hold up to 20 characters and has a NOT NULL constraint. * **nationality**: Stores the nationality of the donor. This varchar attribute has a limit of 20 characters, with a default value of Jordanian if not specified. |
|  | Staff | This table is responsible for storing staff information. The table has the following attributes: (Staff\_ID, Staff\_First\_Name, Staff\_Last\_Name, Salary, Level, Email, Birth\_Date, Country, City, Street\_Name).   * **Staff\_ID**: The Staff\_ID attribute is the primary key of the Staff table. It is an integer with a NOT NULL constraint. * **Staff\_First\_Name**: Stores the staff member's first name. This varchar(20) attribute has a NOT NULL constraint. * **Staff\_Last\_Name**: Stores the staff member's last name. This varchar(20) attribute has a NOT NULL constraint. * **Salary**: Stores the staff member's salary. This int attribute has a NOT NULL constraint. * **Level**: Stores the position level of the staff member (e.g., admin, employee). This varchar(10) attribute has a NOT NULL constraint with a check constraint restricting values to admin or employee. * **Email**: Stores the staff member's email address. This varchar(30) attribute has a NOT NULL constraint. * **Birth\_Date**: Stores the staff member's date of birth. This DATE attribute has a NOT NULL constraint. * **Country**: Stores the staff member's country. This varchar(20) attribute has a NOT NULL constraint. * **City**: Stores the staff member's city. This varchar(20) attribute has a NOT NULL constraint. * **Street\_Name**: Stores the staff member's street name. This varchar(20) attribute has a NOT NULL constraint. |
|  | Event | This table is responsible for storing event details. The table has the following attributes: (Event\_ID, Event\_name, Event\_date, Venue\_name, City, Street\_name).   * **Event\_ID**: The Event\_ID attribute is the primary key of the Event table. It is an integer, auto-incremented, and has a NOT NULL constraint. * **Event\_name**: Stores the name of the event. This varchar(20) attribute has a NOT NULL constraint. * **Event\_date**: Stores the date of the event. This DATE attribute has a NOT NULL constraint. * **Venue\_name**: Stores the name of the event venue. This varchar(20) attribute has a NOT NULL constraint. * **City**: Stores the city where the event is held. This varchar(20) attribute has a NOT NULL constraint. * **Street\_name**: Stores the street name of the event venue. This varchar(20) attribute has a NOT NULL constraint. |
|  | Donation | This table is responsible for storing donation records. The table has the following attributes: (Donation\_ID, Donor\_ID, Amount, Donation\_date).   * **Donation\_ID**: The Donation\_ID attribute is the primary key of the Donation table. It is an integer with a NOT NULL constraint. * **Donor\_ID**: A foreign key linking to the Donor table. This integer attribute has a NOT NULL constraint and cascading updates and deletes. * **Amount**: Stores the donation amount. This int attribute has a NOT NULL constraint and a check constraint ensuring it is greater than 100. * **Donation\_date**: Stores the date of the donation. This DATE attribute has a NOT NULL constraint. |
|  | Donation Allocation | This table tracks donor participation in events. The table has the following attributes: (Donor\_ID, Event\_ID, Participation\_date).   * **Donor\_ID**: A foreign key linking to the Donor table. This integer attribute has a NOT NULL constraint and cascading updates and deletes. * **Event\_ID**: A foreign key linking to the Event table. This integer attribute has a NOT NULL constraint and cascading updates and deletes. * **Participation\_date**: Stores the date of participation. This DATE attribute has a NOT NULL constraint. * **Primary Key**: The composite key consists of Donor\_ID and Event\_ID. |
|  | Donor Phone | This table is responsible for storing the donor's phone numbers. The table has the following attributes: (Donor\_ID, Phone\_number).   * **Donor\_ID**: A foreign key linking to the Donor table. This integer attribute has a NOT NULL constraint and cascading updates and deletes. * **Phone\_number**: Stores the donor's phone number. This int attribute is part of the composite primary key and is unique for each donor. |
|  | Donation Allocation | This table is responsible for mapping donations to events. The table has the following attributes: (Event\_ID, Donation\_ID).   * **Event\_ID**: A foreign key linking to the Event table. This integer attribute has a NOT NULL constraint and cascading updates and deletes. * **Donation\_ID**: A foreign key linking to the Donation table. This integer attribute has a NOT NULL constraint and cascading updates and deletes. * **Primary Key**: The composite key consists of Event\_ID and Donation\_ID. |
|  | Staff Phone | This table is responsible for storing the staff's phone numbers. The table has the following attributes: (Staff\_ID, Phone\_number).   * **Staff\_ID**: A foreign key linking to the Staff table. This integer attribute has a NOT NULL constraint and cascading updates and deletes. * **Phone\_number**: Stores the staff member's phone number. This int attribute is part of the composite primary key and is unique for each staff member. |

### Views Overview

|  |  |  |
| --- | --- | --- |
| **View** | **Name** | **Description** |
|  | View\_All\_Donations | This view summarizes all donations, combining donation details (ID, amount, and date), donor information (first name, last name, and email), and associated event details (name and date). It eases querying cross-table relationships. |
|  | View\_Donor\_Contact | This view shows all the contact information for donors. It includes their ID, first name, last name, email, city, country, and phone numbers. It is a view of the Donor table joined with the Donor\_phone table to make it easier to find donor communication details. |
|  | View\_Staff\_Contact | This view will show the contact details of all staff members, including ID, first name, last name, email, city, country, and phone numbers. It joins the Staff and Staff\_phone tables, so the staff contact details are in one easy-to-access location. |
|  | View\_Total\_Donations\_Per\_Event | This view will show the contact details of all staff members, including ID, first name, last name, email, city, country, and phone numbers. It joins the Staff and Staff\_phone tables, so the staff contact details are in one easy-to-access location. |

### Procedures Overview

|  |  |  |
| --- | --- | --- |
| **Procedure** | **Name** | **Description** |
|  | Get\_Total\_Donations\_By\_Country | This procedure sums up the total donations collected from donors in each country. It groups the donations by the donor's country, summing up the donation amounts and therefore providing insights into regional contributions. |
|  | Get\_Largest\_Donors | This procedure is used to find donors who have donated more than a given amount. It takes a minimum amount as an input parameter and then returns a list of donors along with their total donations that are more than the given amount. |
|  | Get\_Donors\_participate | This procedur takes an event ID and returns a list of all donors who have attended the event; it also includes their contact info email and phone number for use in communications or follow-up after the event. |
|  | Get\_number\_of\_donation\_by\_country | This procedure counts the number of donations donors from each country have made. It first organizes the data by country and then counts the total number of donation records to expose donation activity in various regions. |

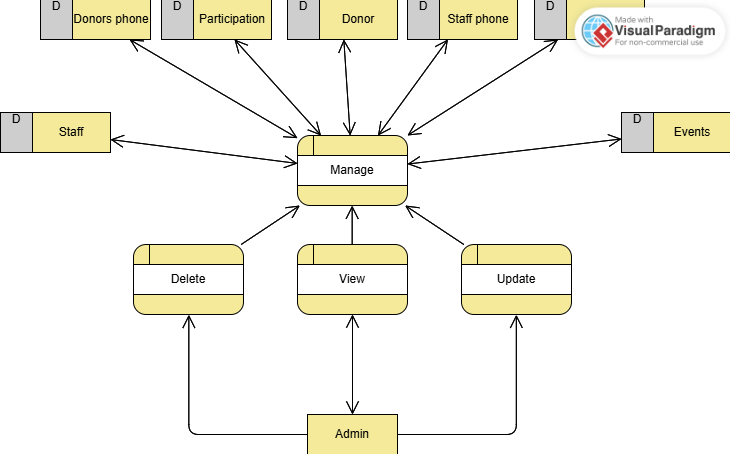
## Security

|  |  |  |  |
| --- | --- | --- | --- |
| **User name** | **Privilege Command** | **Description** | **Screenshot** |
| Employee | grant Select on donor\_tracking\_database.donor to Employee; | This command grants Employee role read only access to the donor table. Employees can view donor information but cannot modify, insert, or delete any data in the table. |  |
| Administrator | grant Select, Insert, Update, Delete on donor\_tracking\_database.\* to Adminstrator; | This command provides the Administrator role with full access to all tables in the database. Administrators can read, add, update, and delete data from any table, allowing them to manage the database comprehensively. |  |
| EventManager | GRANT SELECT, INSERT, UPDATE ON donor\_tracking\_database.Event TO EventManager;  GRANT SELECT ON donor\_tracking\_database.Donation\_Allocation TO EventManager;  GRANT SELECT ON donor\_tracking\_database.Participation TO EventManager; | These commands provide the EventManager role access to manage events and view associated data. They can add, update, and view event details and view data related to donation allocations and participation in events. |  |
| DataAnalyst | GRANT SELECT, EXECUTE ON donor\_tracking\_database.\* TO DataAnalyst; | This command grants the DataAnalyst role read only access to all tables and permission to execute stored procedures. This allows analysts to retrieve and analyze data without modifying it or affecting database integrity |  |

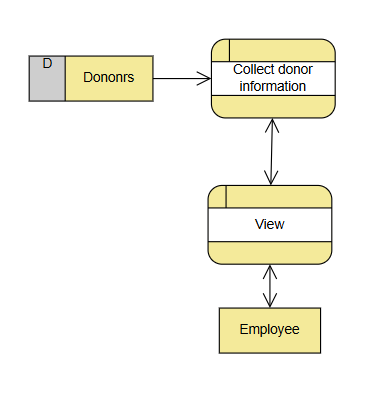
## User Interface

### Flowchart and data movement diagrams

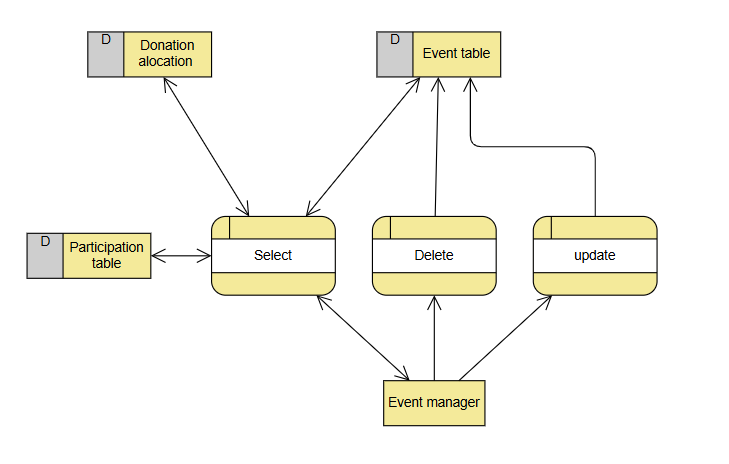
**Amin DFD:**



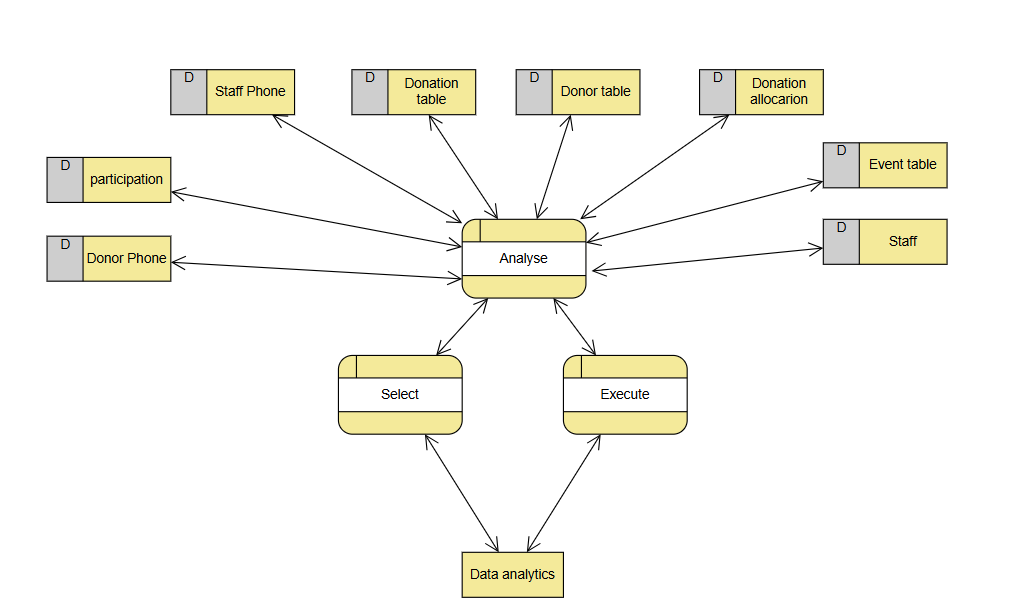
Employee:



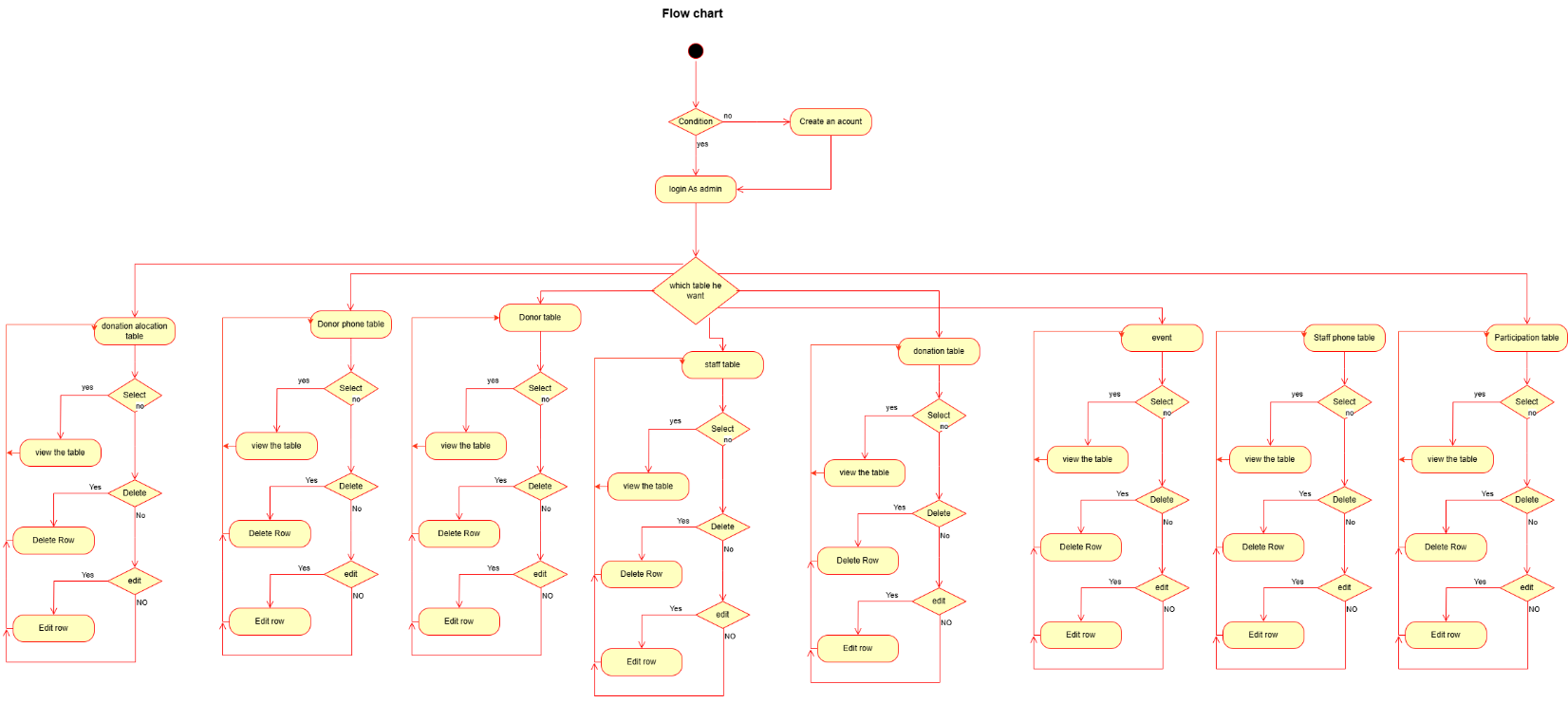
Event manager:



Data Analytics:



**Admin Flowchart:**



### Interfaces Development

|  |  |  |  |
| --- | --- | --- | --- |
| **Page ID** | **Title** | **Description** | **Screenshot** |
|  | Home Page | This is the main page that opens, from this interface you have to put your database port and its name and the username who want to log in and his password |  |
|  | Select table page | From this page the user can see the tables that have access to them and choose which one he want |  |
|  | Data base table | From this page the user can see the attribute of the chosen table and edit or delete or add on them if he has privilege to do it |  |
|  | Add page | From this page the user can add the values to add new row |  |

# 7.Maintenance

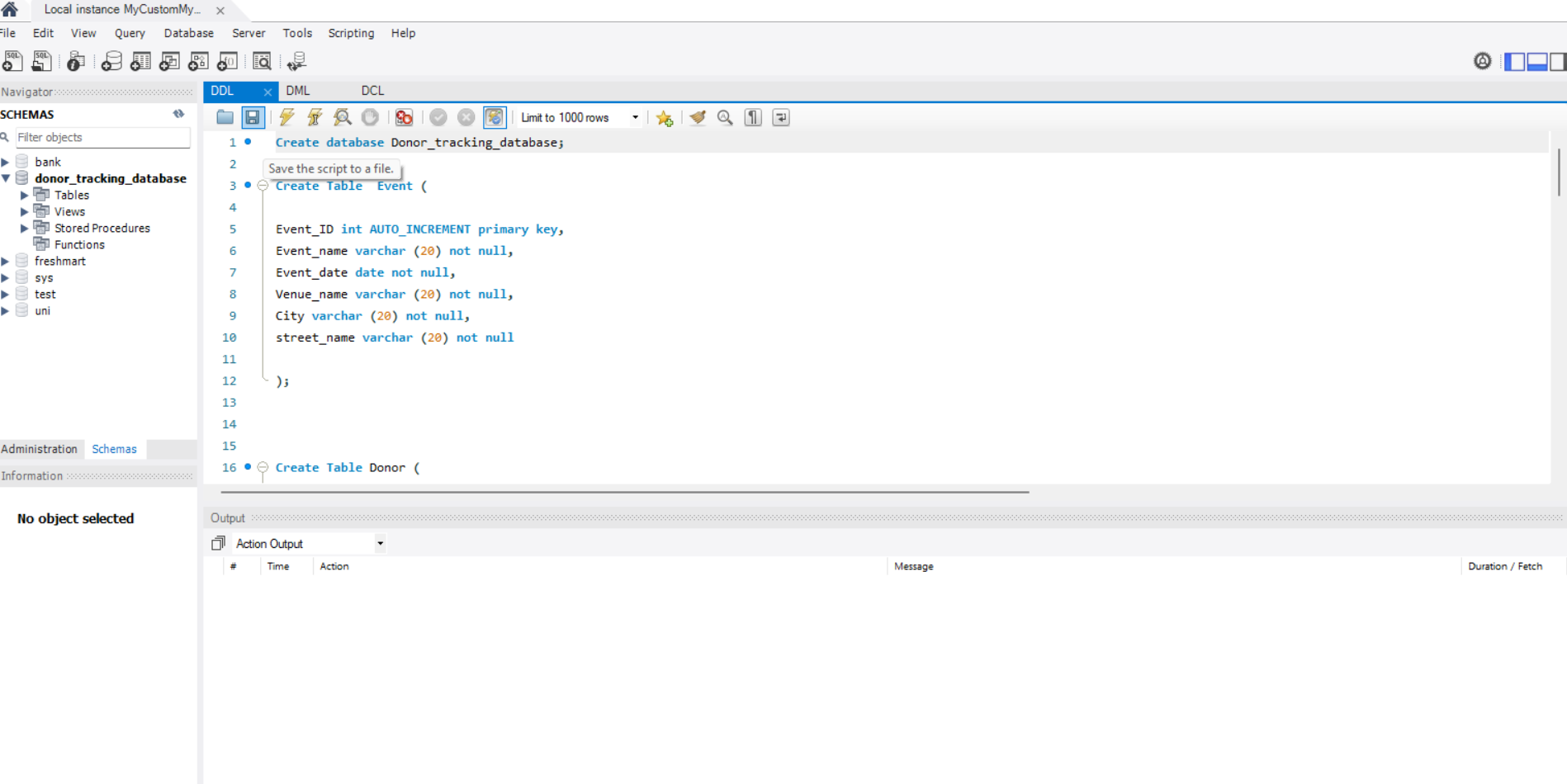
## Database recovery & backups

Data recovery is a very important thing for keeping an organization stable and reliable by ensuring valuable information is still accessible even in the face of unexpected problems. In today's world, which depends so much on data, organizations rely on digital assets to run their operations, make smart choices, and provide services to customers. Data loss can be caused by various reasons, such as accidental deletions, hardware failures, software problems, cyberattacks, and natural disasters. In case businesses are not prepared to recover data, they may suffer from serious issues like losing money, downtime in operations, loss of customer trust, and potential legal issues. Good data recovery solutions help organizations quickly restore lost or damaged data, reducing downtime and keeping the business running. This becomes very important in areas such as healthcare, finance, and retail, where accurate and timely data is key to daily operations. In addition, rules often require organizations to keep data secure and recoverable to meet industry standards and avoid legal issues. A strong data recovery plan includes regular backups, testing how to recover data, and using advanced security measures to stop data loss from happening. Investing in good data recovery systems secures an organization's important information and makes it stronger against possible risks and unknowns. A clear recovery plan will let businesses keep running smoothly, protect their reputation, and build enduring trust with stakeholders and customers. Ultimately, data recovery is not just a technical necessity but a strategic investment that contributes to an organization's sustainability and growth in an increasingly digital landscape.

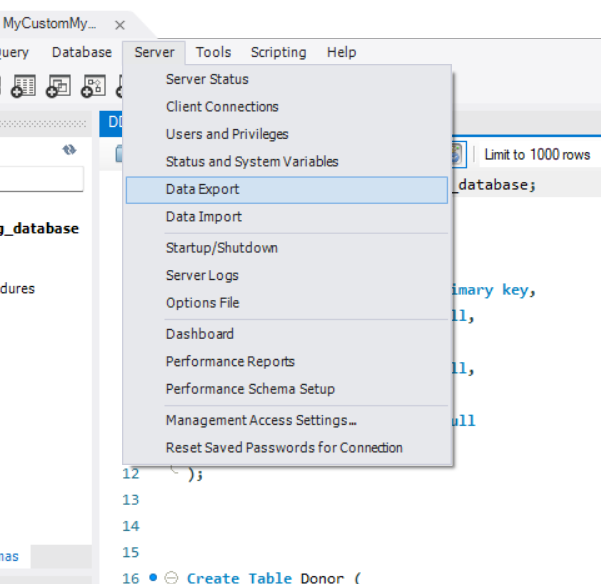
## My Database recovery and backup

**First export:**

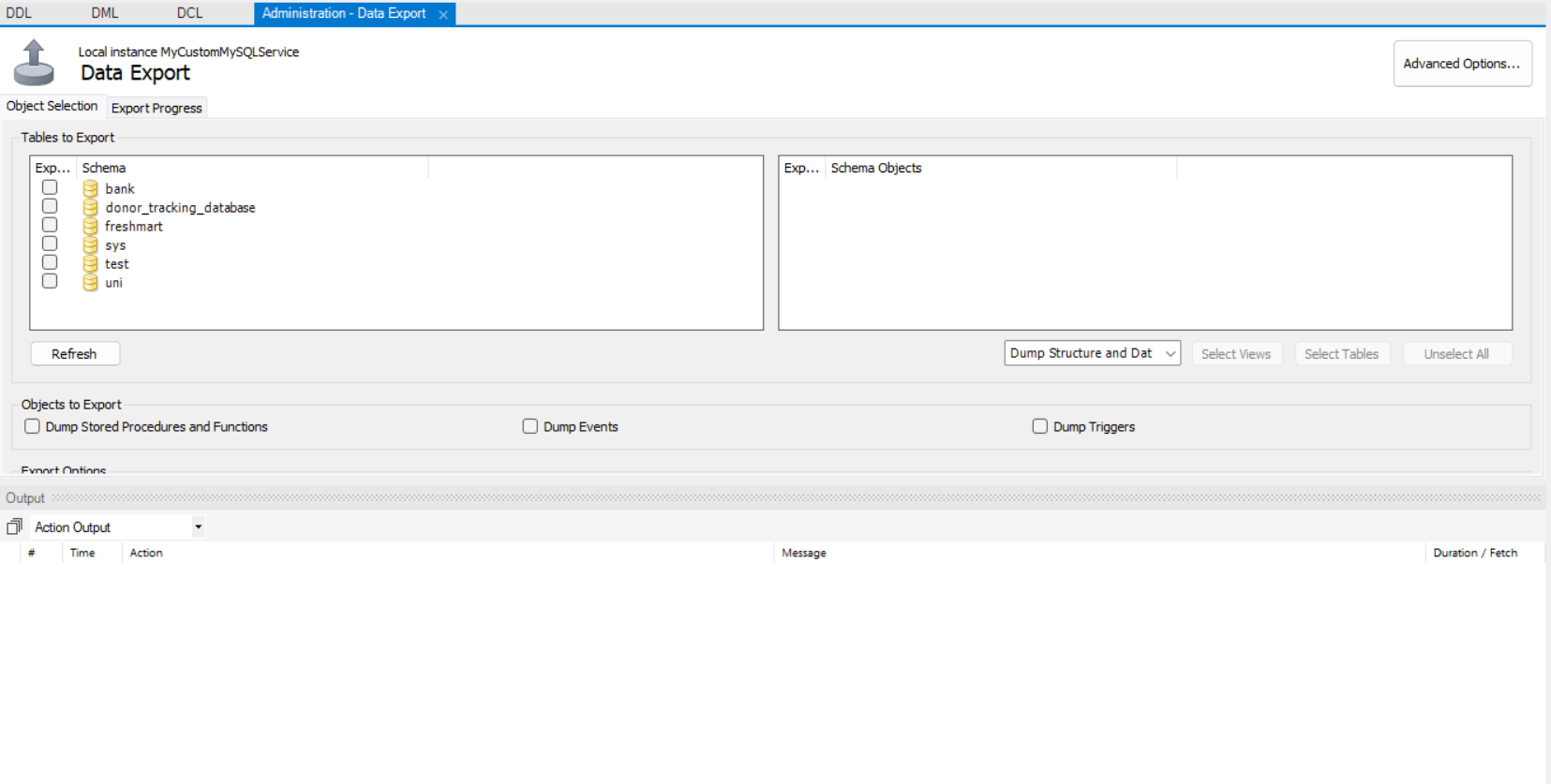
First, we go to the connection and choose Server.



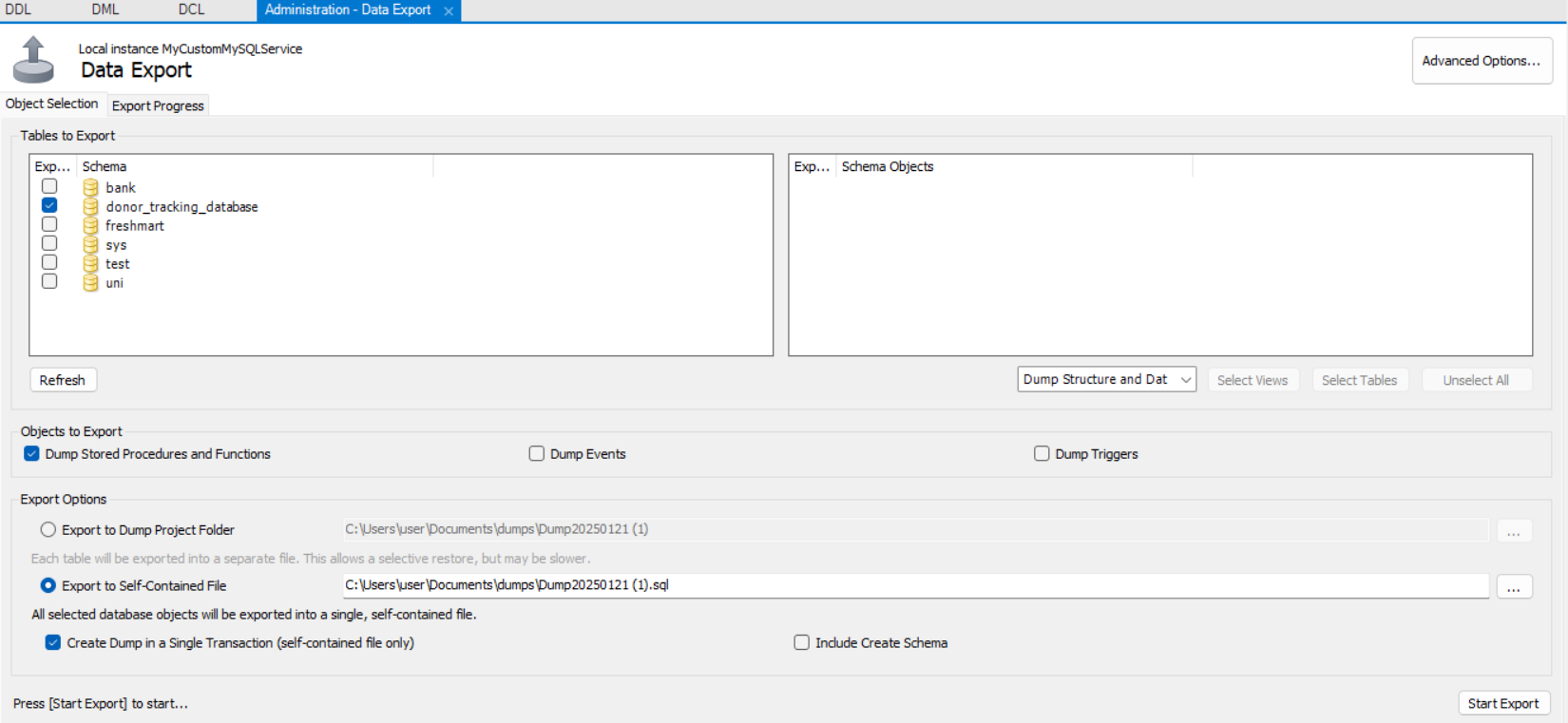
After that we go to Data export.



Then we choose the database we want to export

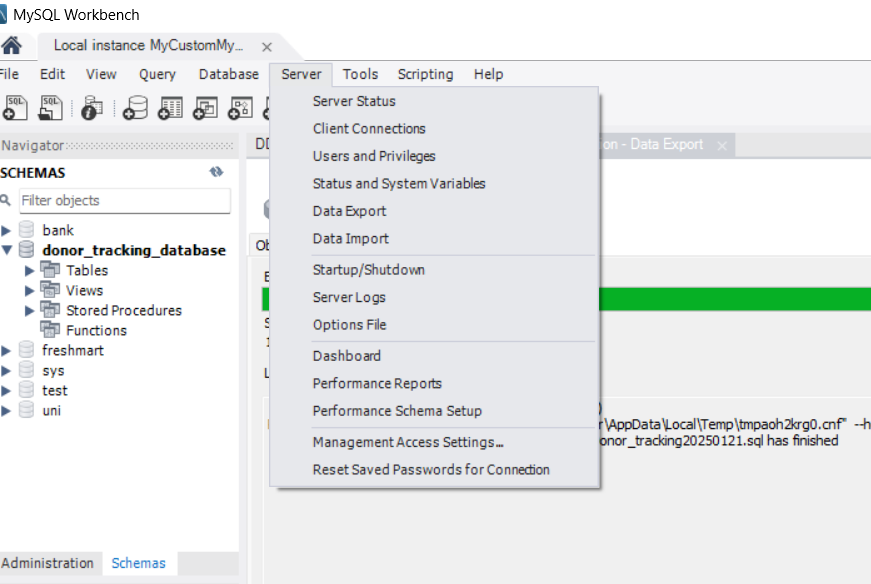


Make sure that I stored the database with the data and I stored the procedure and export them as one file



**Second Data import:**

First go to server then import database

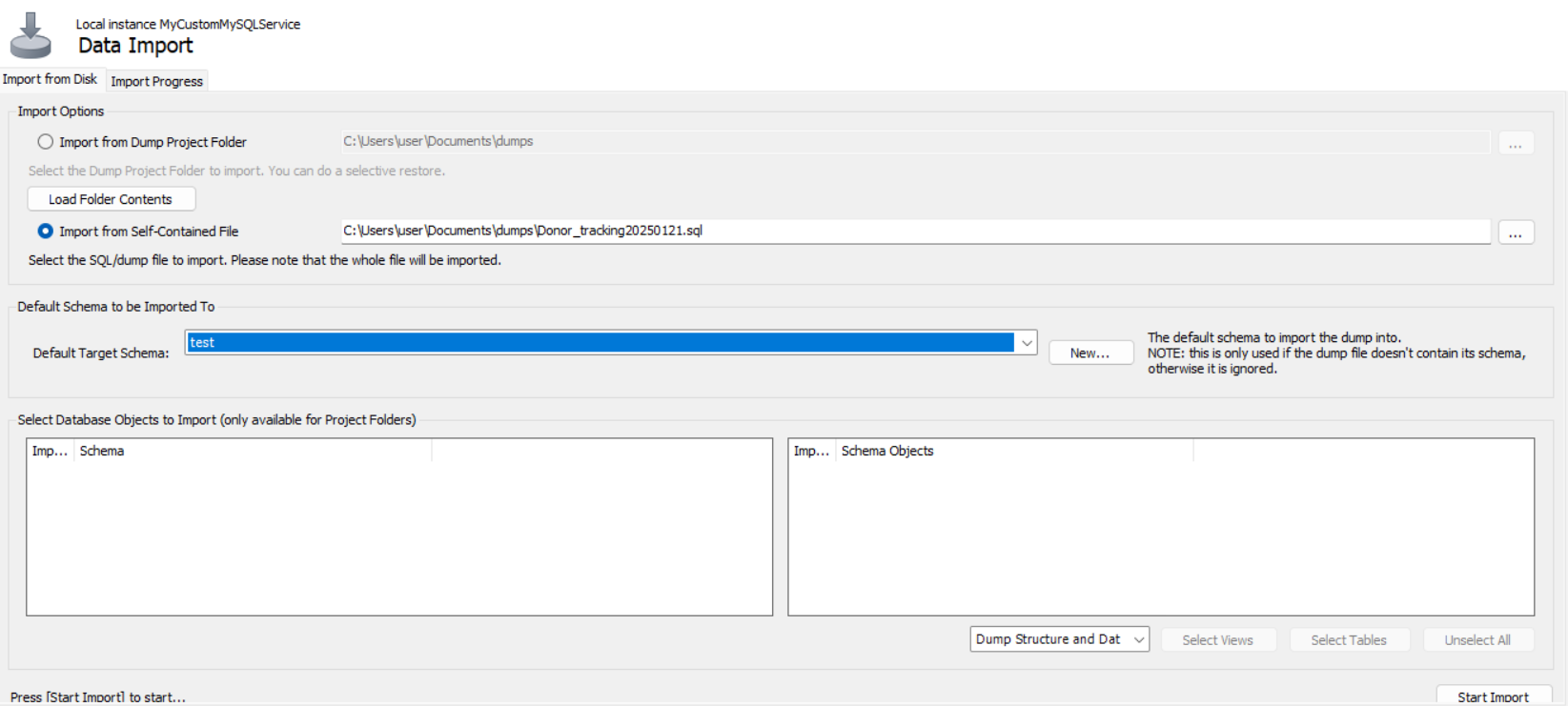


Then this page will appear

A screenshot of a computer

Description automatically generated

Choose the database you want to import and choose the schema you want to import in it or create a new one



The click on start import and like this we import the database successfully.

## Database maintenance in general

Database maintenance is the critical practice that ensures reliability, performance, and security of the data assets in an organization. Activities in the maintenance window can prevent issues that would have otherwise led to data loss, system downtime, or a situation where data integrity may be compromised. Proactive management and optimization of databases by organizations ensure the quality of their operations and their data.

**Importance of Database Maintenance**

The most important aspect of database maintenance is that it can sustain optimum performance and data integrity. Databases, over time, tend to get fragmented, accumulate redundant data, or have old statistics, which could result in a decline in performance. Performing regular maintenance resolves these issues and ensures that the databases continue to work efficiently and meet the needs of the organization. This, in turn, helps in the identification and mitigation of security vulnerabilities to protect sensitive information from unauthorized access or breaches through consistent maintenance practices.

**Database Maintenance Methods**

Effective database maintenance includes some important methods:

* Regular Backups: Regular backups are very critical in the event of hardware failures, accidental deletions, or cyberattacks for the recovery of data. A good backup strategy is one that includes full, differential, and incremental backups for flexible restoration options. (Seis)
* Index Optimization: Indexes provide fast access to data; however, they tend to fragment over time, resulting in poor query performance. Periodic rebuilding or reorganizing of indexes keeps the queries performing optimally and maintains the overall efficiency of the database. (Dimitrov)
* Updating Statistics: The database uses statistical data to come up with an optimal query execution plan. Updating such statistics regularly ensures that the database optimizer is equipped with the actual distribution of data, hence better performance. (Dimitrov)
* Data Integrity Checks: Consistency and integrity checks of the data on a regular basis help in identifying and correcting any anomalies or corruption in the database. Constraints and validation rules ensure accuracy and reliability of the data. (Seis)
* Disk Space Management: It involves monitoring and managing disk space utilization to prevent issues related to storage shortages that may cause system crashes or slow down database operations. Regular cleanups of unused data and archiving of old records are some of the practices carried out to ensure efficient use of disk space. (Seis)
* Apply Updates and Patches: The DBMS and associated software should be updated with the latest patches and updates in order to address security and performance concerns. The timely application of patches released by vendors addresses the known vulnerabilities and bugs. (Collins)
* Performance Monitoring and Tuning: The application of continuous monitoring of database performance metrics enables the identification of any bottlenecks or resource-intensive queries. Tuning such aspects ensures that resources are utilized efficiently to maintain optimal response times. (Collins).

# 8.Testing

## Data validation

|  |  |  |  |
| --- | --- | --- | --- |
| **Number** | **Type** | **Description** | **Screenshot** |
|  | All cases of PK | **Uniqueness:**  -In this test I will duplicate primary key in the donation table which is the donation ID  -the expected output is that an error will arise refusing to add a duplicate Primary key |  |
| **Not Null:** In this test I will try to add null value to the primary key of the donation table which is Donation ID  -Expected Output : give me error because the primary key cant be null |  |
|  | All cases of FK | **Non-existed PK:**  -In this test I will try to add a new row in the staff phone table and give a non-existed primary key for the staff Id which is 19  -the expected output is error appear because there is no staff with ID 18 |  |
| **On update cascade:**  -In this test I will test the validity of the on update cascade, between the staff phone table and staff table by update the staff ID  -The expected output is update the staff Id on both table because this is what update on cascade do | And after updating it to 14: |
| **on delete restrict**  -I will try to delete a staff that have id as foreign key in another table  -The Expected output is error |  |
|  | Unique | - On this test will duplicate a unique attribute in the donor table which is SSN  - the expected output is to give me error because this constraint not allow to duplicate the values | After I make a new row and give it same SSN of first donor: |
|  | Default | -In this test I will try to add a new donor in the donor table and not give value in Nationality  -The expected value to give it Jordanian as a default value | There is no nationality value let’s see how it appear on the table: |
|  | Not null | -In this test I will try to put a null value in attribute that has not null constraint like salary in staff table  -The expected output to give me error that there is null value |  |
|  | Check | -In this test in the donation table the donation amount attribute which have check constraint to check if the value more than 100 , so I will add 12 as value  - give me error |  |

## Output validation

|  |  |  |  |
| --- | --- | --- | --- |
| **Number** | **Query Description** | **Screenshot (query + result)** | **Result validation** |
|  | This query help us to see the donors and there donation information without have to open both tables |  | This is the 2 tables before use join:    when we compare them we find they have the same result: |
|  | This Query will give me all the donors who give donation more than 1000 its help in case we want to see best donors |  | I will see the donator id and see if he donate above 1000:    Lets check on the donation table:  700 + 500 = 1200 > 1000 |
|  | -This query lets us know how many employees in the employee table have the same rank.  -Every company must keep a record of its employees and the level the employees are in because it is very important to know how many employees are on their respective levels. If they are being managed correctly or not. |  | Let’s see the employee table to see how many members with same level: |
|  | -This Query update the phone number of the employee with id 2  - this query is important if some of the employees buy new phone number and we need to change t |  | This is the table before the update:    And this after the update: |

## Security Validation

|  |  |  |  |
| --- | --- | --- | --- |
| **Number** | **Username** | **Description of privilege/no privilege** | **Screenshot (query + result)** |
|  | Admin | This privilege is to delete a row in event donation table. I gave this privilege to admin because I wrote in the user requirements that the admin can delete whatever he want. |  |
|  | Admin | This privilege give the admin ability to add new row I will add new row in donor table and see if will allow me to do it . and I gave this privilege to admin because he must have an ability to manage the tables and new values in it |  |
|  | Employee | Employee can select the donor table from and see the values |  |
|  | Event manager | Event manager has privilege to update in the event table because his responsibility to manage the events so I will test if I can update on event table using Event manager connection |  |
|  | Employee | Employee don’t have privilege to delete in the database |  |
|  | Data analytic | I didn’t give the data analytic privilege to update on any table so will test if he can insert new value in the donor table |  |
|  | Event manager | The event manager can just select the event , Donation allocation, and participation tables so I will test it |  |
|  | Employee | No user should have privilege to drop database because it will cause a huge threat |  |

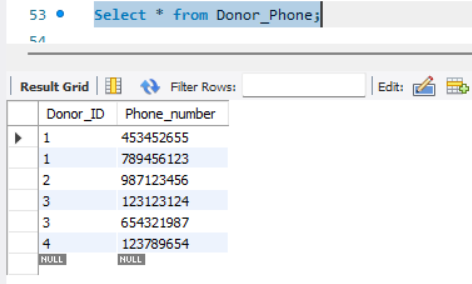
## GUI Validation

|  |  |  |
| --- | --- | --- |
| **Number** | **Description** | **Screenshot** |
|  | To test the validation the output values from view and test if we can see the information and I will compare between GUI and Query from the database to see if the output from them is the same. | GUI:      We can see that the output from the database and the GUI is completely the same. |
|  | Now I will test if adding new row in GUI will effect on the database  Now we add a new row and fill the info  After saving the new row, we can see that it has been added successfully to the staff members table  We can confirm that the new row has been added after checking on the database directly. | Adding a new row and save it:    New added row:    Checking from the database: |
|  | I will edit a row in the Donor table through the graphical user interface (GUI), I will modify the email value  After making the change, we observed that the updated value was successfully reflected in the GUI. | Changing value:      Edited row:    Database: |
|  | I'll delete a row from the donor table via the GUI to check if it eill be deleted in the in the database | Deleting a row from GUI:    Row deleted from database: |

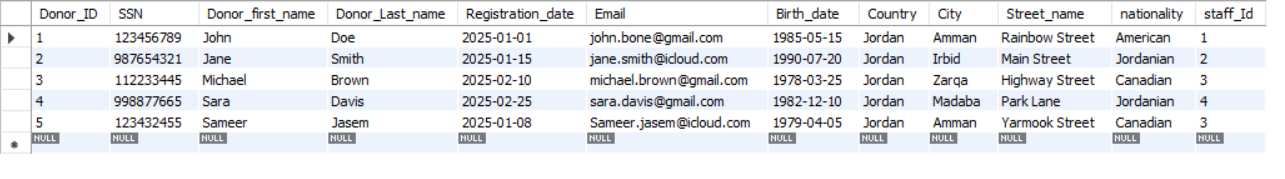
## Assess whether meaningful data has been extracted

### Relationships

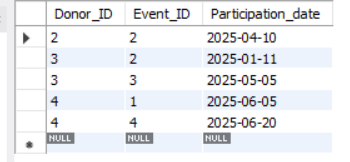
In my schema, some entities have **multi-value** attributes, which stored in separate table. As example the donor can have multi phone numbers so instead of storing multi-valued on one column I created donor phone table. See the donors with id 1 and 3 have 2 phone numbers:



Next, I have a 1:M relationship between donor and staff each staff responsible for multiple donors. Lets see example:



Now lets test see the M:N relationship table in my database, I made a third table for my M:m relationship because case like donor can participate in multiple events, and an event can have multiple donors so I managed this by participation table. Lets see example:



### My views and procedures

**Donation information view**: This view shows donor details and their contact information, like email, city, country, and phone numbers. It helps with communication and engaging individuals.

**Who Benefits:**

* **Administrators** to track donations and event funding.
* **Data Analysts** for financial analysis and donor trends.
* **Event Managers** to understand how events attract donations.

**Example:** The admin uses the Donation Information View to check that John Doe from Amman donated 500 JOD on March 1, 2025, and his contribution was allocated to the Charity Gala event. This helps the admin track fundraising success and donor engagement.

**Donor contact information:** This view shows donor details and their contact information, like email, city, country, and phone numbers. It helps with communication and engaging individuals.

**Who Benefits:**

* **Administrators:** can contact donors to sustain relationships.
* **Event Managers:** can invite donors to events and fundraisers.
* **Employees:** can help with donor queries.

**Example:** The event manager uses the Donor Contact Information View to retrieve the email and phone number of Sara Davis from Madaba, so they can send her an invitation for the upcoming Blood Donation Camp event on May 5, 2025.

**Staff Contact Information:** This view provide the ability to view all contact details of the staff members, making internal communication and more management effectiveness.

**Who Benefits:**

* **Administrators:** managing staff assignments.

**Example:** The admin uses the Staff Contact Information View to find the phone number of Clara Wilson, an admin in Zarqa, in order to discuss upcoming donor outreach efforts.

**Total donations per event view:** This view sums donation amounts and displays the total contributions per event. It assists in seeing what events brought in the most money.

**Who Benefits:**

* **Administrators:** to evaluate event performance.
* **Data Analysts:** can view financial contributions by event.
* **Event Managers**: can try to improve poorly performing events.

**Example:** The data analyst uses the Total Donations Per Event View to see that the Charity Gala on March 15, 2025, raised a total of 1,200 JOD, helping the organization determine which events attract the most donations.

**Total Donations by country Procedure**: This process aggregates the total donations by country. It allows us to see donor trends by geographical area and focus our outreach activities.

**Who Benefits:**

* **Data Analysts:** are working on understanding donations by region.
* **Administrators:** help make fundraising campaigns more focused.

**Example:** The admin runs the Total Donations by Country Procedure and discovers that Jordan has received 2,850 JOD in total donations, indicating that Amman and Zarqa are key donor regions.

**Identifying Major Donors Procedure:** This process collects the donors who have given over a specified amount, making it easy to identify those high-value donors.

**Who Benefits:**

* **Administrators:**so, they can identify and reward their biggest donors.
* **Data Analysts**: so, they can see donation trends over time**.**

**Example:** The admin uses the Identifying Major Donors Procedure to find that Sameer Jasem has donated a total of 900 JOD, making him a top donor. The admin can now send him a special thank-you email and a VIP invitation to future events.

**Donors Participating in an Event procedure:** This procedure retrieves all donors who participated in a specific event, along with their contact details. It simplifies post-event communication and donor engagement.

**Who Benefits:**

* **Event Managers:** to follow up with attendees.
* **Administrators:** to analyze donor involvement in different events.

**Example:** The event manager uses the Donors Participating in an Event Procedure to see that Jane Smith and Michael Brown attended the Food Drive event on April 10, 2025. The event manager then sends them thank-you emails and invites them to future events.

**Procedure: Number of Donations by Country:** This procedure returns the total count of donations from each country, which helps organizations measure donor participation levels in different regions.

**Who Benefits:**

* **Data Analysts:** to understand donor distribution.
* **Administrators:** to assess the success of international fundraising efforts.

**Example**: The data analyst runs the Number of Donations by Country Procedure and sees that Jordan had 5 donations, while Canada had only 2 donations. This information helps them recommend a stronger fundraising campaign in Canada.

Also In my database different users have the appropriate access levels while maintaining security:

* **Employee**: Can view donor details but cannot modify any records.
* **Administrator**: Has full control over the database, including inserting, updating, and deleting records.
* **Event Manager**: Can manage event-related data, including updating event details and viewing donor participation.
* **Data Analyst**: Can execute stored procedures and analyze donation patterns.

## Effectiveness of testing

**Significance of the Testing Process**

Testing is a critical step in developing a database. It renders the system efficient, secure, and accurate. A well-tested database reduces data inconsistencies, unauthorized access, and slow query processing. In donor tracking database, testing is necessary to ensure donations, donor information, staff information, events, and participation records are accurately stored, retrieved, and updated..

**Testing Steps and Their Importance**

To ensure my database performed well, I employed a straightforward testing methodology. I did unit testing, testing views, and stored procedures in isolation to find out if they produced the appropriate results. I tested View\_Total\_Donations\_Per\_Event as an example to ensure it accurately produced donation amounts by event. I was able to identify syntax errors, join errors, and lack of conditions within SQL queries using this.

Then, I did functional testing to ensure that all four user roles—Administrator, Employee, Event Manager, and Data Analyst—had the appropriate degree of access rights. I verified situations such as whether an Employee should be able to read donor information but not delete or update it, while an Administrator should be able to carry out add, update, and delete operations in complete command. This ensured that role-based access control.

Integration testing was crucial. I verified foreign key relationships to ensure consistency in data between tables. For instance, when dropping a donation record, I ensured that the associated entry in Donation\_Allocation was dropped as well. This activity prevented orphaned records and provided referential integrity in the database.

To verify for security vulnerabilities, I performed security testing by attempting to perform things that were not permitted, such as attempting to delete donor records when logged in as an Employee. Since Employees only had read access to donor information, the system properly prevented the deletion attempt, demonstrating that the security rules were being enforced. This was crucial to prevent errors or malicious behavior with the data.

Through this extensive testing, I discovered and resolved some serious issues. One such serious issue was with data integrity violations; when a donation was deleted, its corresponding entry in Donation\_Allocation was not automatically removed, resulting in inconsistencies. I fixed this using ON DELETE CASCADE, so dependent records were deleted automatically. There was a second issue of poor access control, where Employees had unnecessary DELETE privileges on the Donor table. I resolved this by revoking their DELETE privileges, with deletion privileges reserved for Administrators alone.

Optimizing query performance was a second significant change on testing. The Get\_Largest\_Donors stored procedure was not properly grouping donations, and this was resulting in incorrect donor contribution amounts. I discovered that the issue was a missing GROUP BY clause, and once I corrected it, the procedure was reporting the donor rankings accurately. The View\_Donor\_Contact query was also producing duplicate donor records for donors with more than one phone number. We accomplished that by modifying the JOIN condition so that every donor was listed once with their contact information.

# References

Banister, Rick. “Physical Database Design | Sesame Software.” *Sesame Software*, 2019, sesamesoftware.com/white-papers/physical-database-design/.

Collins, Emily. “A Comprehensive Handbook for Regular Database Maintenance -.” *The 9th Door*, 12 Oct. 2023, the9thdoor.com/guide-to-routine-database-maintenance/?utm\_source=chatgpt.com. Accessed 21 Jan. 2025.

Dimitrov, Miroslav. “Tips and Tricks for SQL Server Database Maintenance Optimization.” *SQL Shack - Articles about Database Auditing, Server Performance, Data Recovery, and More*, 11 Jan. 2016, www.sqlshack.com/tips-and-tricks-for-sql-server-database-maintenance-optimization/?utm\_source=chatgpt.com. Accessed 21 Jan. 2025.

Nayak, Shruti. “Database Backup and Recovery Techniques - Shruti Nayak - Medium.” *Medium*, Medium, 2 May 2023, [medium.com/@shruti.nayak20/database-backup-and-recovery-techniques-c541faf3444a](mailto:medium.com/@shruti.nayak20/database-backup-and-recovery-techniques-c541faf3444a).

NetApp. “What Is Backup and Recovery? - Why It’s Important | NetApp.” *Www.netapp.com*, 2023, [www.netapp.com/cyber-resilience/data-protection/data-backup-recovery/what-is-backup-recovery/](http://www.netapp.com/cyber-resilience/data-protection/data-backup-recovery/what-is-backup-recovery/).

Seis, David. “Essential Database Maintenance Tasks Every DBA Should Know.” *SQL Server Consulting - Straight Path Solutions*, 23 Aug. 2023, straightpathsql.com/archives/2023/08/essential-database-maintenance-tasks-every-dba-should-know/?utm\_source=chatgpt.com. Accessed 21 Jan. 2025.

Veritas. “Data Backup & Recovery: The Essential Guide for Businesses | Veritas.” *Www.veritas.com*, 2023, [www.veritas.com/information-center/data-backup-and-recovery](http://www.veritas.com/information-center/data-backup-and-recovery).

“What Is Backup and Recovery?” *Cohesity*, [www.cohesity.com/glossary/backup-and-recovery/](http://www.cohesity.com/glossary/backup-and-recovery/).