American Red Cross Database Simulation Project

Katherine Curro & Pranati Mita

I. Introduction

The American Red Cross is a non-profit organization that helps people all around the world. From disaster recovery to blood drives, the Red Cross plays an instrumental part in prevention and recovery from emergencies everywhere. The Red Cross is funded by monetary donations. In order to organize their donations the Red Cross has to use a database. Our expert and contact person, Kim, works as a data analyst for the American Red Cross and provided us with information on the database system used to track donations, which she uses daily.

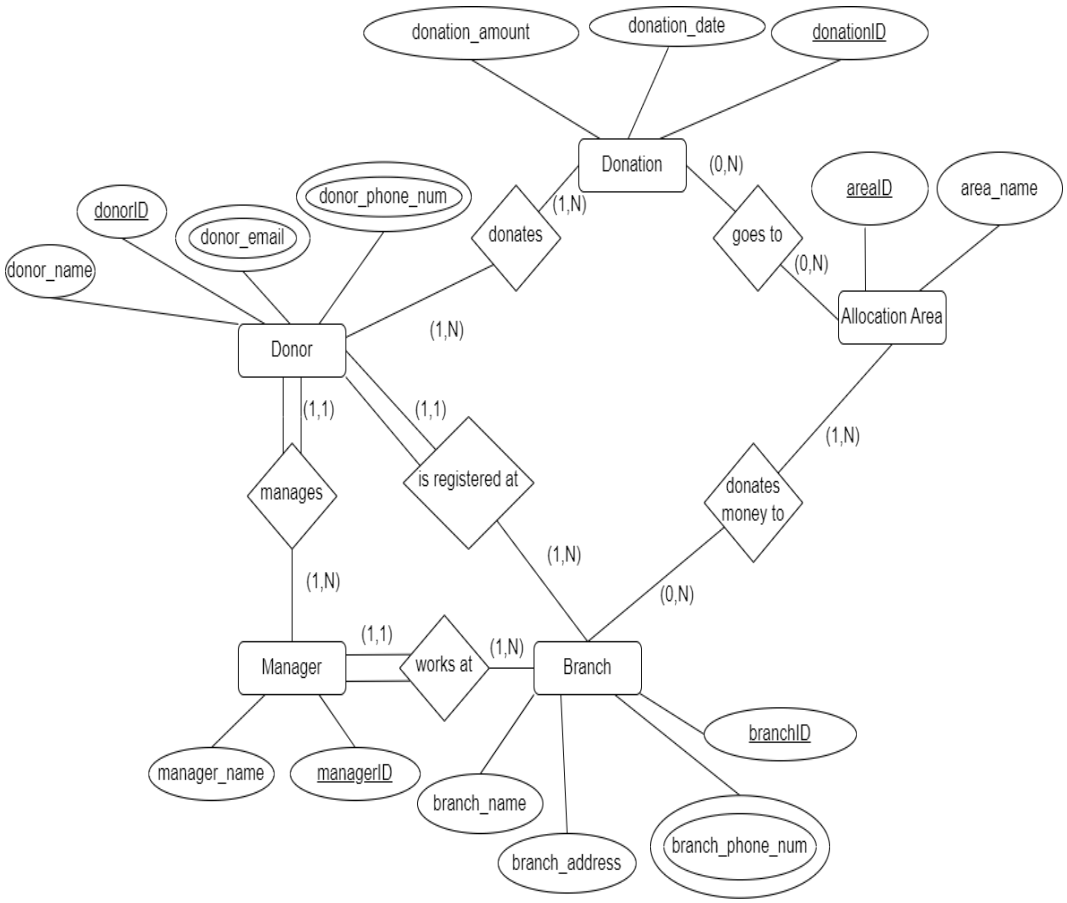
II. Description of the Organization

The Donor Database stores relevant information regarding the American Red Cross' desire to track donations made across the world. The Donor Database contains data about the donors available, managers of the donors, the branches which each donor is registered at, the monetary donation made by each donor, and the allocation area where each donation goes to:

- Donors are recorded to account for funds for the Red Cross. Each donor possesses a unique identification number, a name, a phone number, and an email. Furthermore, each donor can make multiple donations and each donation may come from multiple donors. Each donor is also registered at only one branch, while each branch registers at least one donor.
- Each branch has a unique identification number, a phone number, an address, and name. Every branch also has at least one manager, however, each manager will only work at one branch.
- The role of the manager is to ensure that each of the donors's donations gets processed and to ensure their branch is running smoothly. Every manager has a unique identification number and a name. Each manager manages at least one donor, while each donor can only have one manager to manage them.

- The donations made by donors are also tracked to organize the funds of the Red Cross. Every donation has a unique identification number, a date, and an amount. Each donation can go to multiple allocation areas and each allocation area can have multiple donations.
- The allocation areas are designed to gain a better understanding of where/what funds will be put towards. Every allocation area has a unique area identification number and an area name. Each allocation area must receive money from at least one branch, while each branch may have money donated from multiple allocation areas.

III. ER Diagram



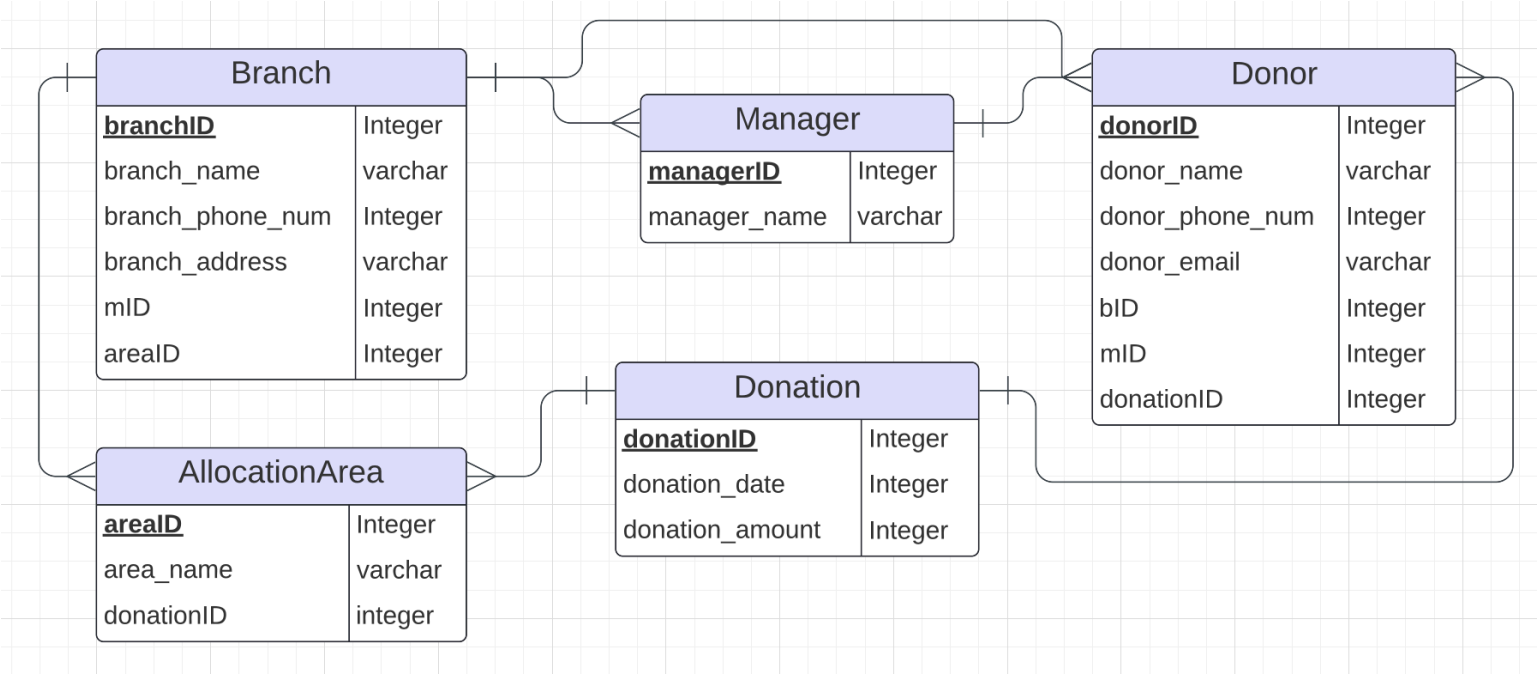
The figure above displays the ER diagram of the Donor Database.

IV. ER Diagram Uncaptured Constraints

In this section, we include both constraint and justification for said constraint.

1. Donation.donation_amount must be greater than 0
    a. Negative donations are nonsensical
2. Donation.donation_amount must be of double datatype
    a. Some donations may include fractions of dollars, cents, and the database must support this
3. Donation.donation_amount should be in USD
    a. Since these locations are based in the US, they should all be in a consistent currency
4. Donation.donationID, Allocation Area.areaID, Branch.branchID, Manager.managerID, Donor.donorID must all be unique and nonnull
    a. These are primary keys that differentiate instances, so they should be unique and nonnull
5. Donation.donation_date must be a valid date following a consistent mm/dd/yyyy format
    a. Consistency across the database is important, and invalid dates are nonsensical
6. Branch.branch_phone_num and Donor.donor_phone_num must be valid numbers with 10 digits.\
    a. Phone numbers without 10 digits do not work
7. Donor.donor_email must be a valid email with an '@emailserver.domainextension'
    a. Emails without a server and domain extension do not work
8. Manager.manager_name, Donor.donor_name should have an attached first and last name
    a. For good record keeping, the database should require a first and last name, for future reference to this person

V. Relational Schema



The figure above displays the relational schema of the Donor Database.

VI. Relational Schema with Referential Integrity

Branch(<u>branchID</u>**,** branch_name, branch_phone_num, branch_address, mID)

foreign key(**mID**) references donor(**managerID**)

Manager(<u>managerID</u>**,** manager_name)

Donor(<u>donorID</u>, donor_name, donor_phone_num, donor_email, bID, mID)

foreign key(**bID**) references branch(**branchID**)

foreign key(**mID**) references donor(**managerID**)

Donation(<u>donationID</u>, donation_date, donation_amount)

AllocationArea(<u>areaID</u>, area_name)

donates(<u>donorID, donationID</u>)

foreign key(**donorID**) references Donor(**donorID**)

foreign key(**donationID**) references Donation(**donationID**)

donates_money_to(<u>branchID, areaID</u>)

foreign key(**branchID**) references Branch(**branchID**)

foreign key(**areaID**) references AllocationArea(**areaID**)

goes_to(<u>donationID, areaID</u>)

foreign key(**donationID**) references Donation(**donationID**)

foreign key(**areaID**) references AllocationArea(**areaID**)

VII. Relational Table Details

The relational schema given in Section 5.1 was mapped into the following tables in the Red Cross Donor Database. Primary keys have been underlined.

Table Name	Attribute	Description
Branch	<u>branchID</u>	Unique branch ID number
	branch_phone_num	Branch phone numbers
	branch_name	Branch name
	brnach_address	Branch address
Manager	<u>managerID</u>	Unique manager ID number
	manager_name	Manager name
Donor	<u>donorID</u>	Unique donor ID number
	donor_name	Donor name
	donor_phone_num	Donor phone numbers
	donor_email	Donor email address
Donation	<u>donationID</u>	Unique donation ID
	donation_date	Date donation was given
	donation_amount	Amount donated
AllocationArea	<u>areaID</u>	Unique allocation area ID number
	area_name	Allocation area name

VIII. Brief Descriptions Of Queries

Query Name	Description	Output	Relations Accessed
DonorSum	List the donors from highest to lowest total donations	donorID, donation_amt	Donor Donation donates
AreaDonationAmount	Find the donation amounts of each allocation area.	areaID, area_name, SUM(D.donation_amount)	AllocationArea, Donation, goes_to
AllocationAreaDonations	List the name of each allocation area and the number of donations each area has in order of the highest to lowest number of donations.	area_name, donationsPerArea	Donation goes_to AllocationArea
DonorsNotDonating	Find the donors who have not yet donated and provide their branch and email, as well as the amount their area has collected in total, sorted by name.	donor_name, donor_email, branch_name, SUM(donation_amount) as total_donations_for_area	Donor Branch Donation
AvgBranchDonations	Retrieves the average donation made to each branch, sorted by donation amount ascending	branchID, AvgBranchDonation	Branch AllocationArea Donation

IX. Implementation Of Queries

Query (DonorSum):

```
SELECT D.donorID, SUM(D.donation_amt)
FROM Donation D, Donor O
WHERE D. donorID = O.donorID
GROUP BY D.donorID
ORDER BY SUM(D.donation_amt) DESC
```

Output:

	donorid character varying (50)	sum bigint
1	11112	265
2	11111	170
3	11115	150
4	11114	50
5	11113	25
6	11116	10

Query (AreaDonationAmount):

```
SELECT A.areaID, A.area_name, SUM(D.donation_amt) AS total_donation_amt
FROM AllocationArea A, Donation D
WHERE A.areaID = D.areaID
GROUP BY A.areaID, A.area_name
ORDER BY total_donation_amt DESC
```

Output:

	area_id [PK] character varying (50)	area_name character varying (255)	total_donation_amt bigint
1	A5	Area E	300
2	A3	Area C	145
3	A2	Area B	125
4	A4	Area D	50

Query (AllocationAreaDonations):

```
SELECT A.area_name, Count(D.donationID) AS donationsPerArea
FROM AllocationArea A, Donation D
WHERE D.areaID = A.areaID
GROUP BY A.areaID
ORDER BY donationsPerArea DESC
```

Output:

	area_name character varying (255)	donationsperarea bigint
1	Area B	3
2	Area E	2
3	Area C	2
4	Area D	1
5	Area A	1

Query (DonorsNotDonating):

```
SELECT O.donorID, O.donor_email, B.branchID, SUM(D.donation_amt) as
total_donation_for_area
FROM Donor O, Branch B, Donation D
WHERE B.branchID = O.branchID AND B.areaID = D.areaID
AND O.donorID NOT IN (SELECT donorID FROM Donation)
GROUP BY B.branchID, O.donorID, O.donor_email
ORDER BY O.donor_name;
```

Output:

	donor_name character varying (50)	donor_email character varying (50)	branch_name character varying (50)	total_donation_for_area bigint
1	Jane Doe	janedoe@gmail.com	Las Vegas	125
2	John Doe	jdoe@outlook.com	Philadelphia	145

Query (AvgBranchDonations):

```
SELECT B.branchID, B.branch_name, avg(D.donation_amt) as AvgBranchDonation
FROM AllocationArea A
JOIN Donation D
ON A.areaID = D.areaID
JOIN Branch B
ON A.areaID = B.areaID
GROUP BY B.branchID
ORDER BY AvgBranchDonation desc
```

Output:

	branchid [PK] integer	branch_name character varying (50)	avgbranchdonation numeric
1	10006	Washington D.C	150.0000000000000000
2	10003	New York City	72.5000000000000000
3	10004	Philadelphia	72.5000000000000000
4	10001	Los Angeles	50.0000000000000000
5	10005	Fort Wayne	50.0000000000000000
6	10002	Las Vegas	41.6666666666666667

X. DML, DDL, SQL Statements

```
CREATE TABLE AllocationArea(areaID varchar(50) PRIMARY KEY, area_name
varchar(255));

INSERT INTO AllocationArea (areaID, area_name)
VALUES
('A1', 'Area A'),
('A2', 'Area B'),
('A3', 'Area C'),
('A4', 'Area D'),
('A5', 'Area E');

CREATE TABLE Manager(managerID varchar(50) PRIMARY KEY, manager_name
varchar(50), );

INSERT INTO Manager(managerID, manager_name)
VALUES
('M1', 'Eliza Cohen'),
('M2', 'Trina Vega'),
('M3', 'Neha Singh'),
('M4', 'Owen Cline'),
('M5', 'Layla Dyson'),
('M6', 'Carson Smith');

CREATE TABLE Branch(branchID varchar(50) PRIMARY KEY, branch_name varchar(50),
branch_address varchar(50), branch_phone_num varchar(50), areaID varchar(50), FOREIGN
KEY (areaID) REFERENCES AllocationArea(areaID), managerID varchar(50), FOREIGN
KEY (managerID) REFERENCES Manager(managerID));

INSERT INTO Branch(branchID, branch_name, branch_address, branch_phone_num, areaID ,
managerID)
VALUES
('10001', 'Los Angeles', '3438 San Marino Street' , '4246567731', 'A1', 'M1'),
('10002', 'Las Vegas', '10700 Space Odyssey Avenue' ,  '7023368547','A2', 'M2'),
('10003', 'New York City', '326 Theatre Drive', '2124962270','A3', 'M3'),
('10004', 'Philadelphia', '801 North Bambrey Street', '21564859012','A3', 'M4'),
('10005', 'Fort Wayne', '914 West Gump Road', '2606371994','A4', 'M5'),
('10006', 'Washington D.C', '110 Maryland Avenue', '2026541576','A5', 'M6');
```

```
CREATE TABLE Donor(donor_name varchar(50), donorID varchar(50) PRIMARY KEY,
donor_email varchar(50), donor_phone_num varchar(50), branchID varchar(50), FOREIGN
KEY (branchID) REFERENCES Branch(branchID), managerID varchar(50), FOREIGN KEY
(managerID) REFERENCES Manager(managerID));

insert into Donor(donor_name, donorID, donor_email, donor_phone_num, branchID,
managerID)
values
('Todd Smith', '11111', 'tsmithy@hotmail.com', '7089895446', '10001', 'M1'),
('Elena Baker','11112', 'bakersman@gmail.com', '7089387495', '10002', 'M2'),
('Riley MacBeth', '11113', 'whereartthou@yahoo.com', '2698499847', '10003', 'M2'),
('John Macafee', '11114', 'macafeematters@gmail.com', '7080394495', '10004', 'M3'),
('Jack Ripper', '11115', 'beware@outlook.com', '7084549983', '10005', 'M4'),
('Ginny George', '11116', 'ggeorge@gmail.com', '5159389948', '10006', 'M5'),
('John Doe', '11117', 'jdoe@outlook.com', '7083455555', '10004', 'M5'),
('Jane Doe', '11118', 'janedoe@gmail.com', '5159037485', '10002', 'M6');

CREATE TABLE Donation(donationID varchar(50) PRIMARY KEY, donation_date date,
donation_amt integer, area_ID varchar(50), donorID varchar(50), areaID varchar(50), FOREIGN
KEY (areaID) REFERENCES AllocationArea(areaID), FOREIGN KEY (donorID)
REFERENCES Donor(donorID));

INSERT INTO Donation(donationID, donation_date, donation_amt, areaID, donorID)
VALUES
('D1', to_date( '2023/10/20', 'YYYY/MM/DD'), 50, 'A1',  '11111'),
('D2', to_date( '2023/10/21', 'YYYY/MM/DD'), 100, 'A2',  '11112'),
('D3', to_date( '2023/10/20', 'YYYY/MM/DD'), 25, 'A3',  '11113'),
('D4', to_date( '2023/10/22', 'YYYY/MM/DD'), 50, 'A4',  '11114'),
('D5', to_date( '2023/10/23', 'YYYY/MM/DD'), 150, 'A5',  '11115'),
('D6', to_date( '2023/10/24', 'YYYY/MM/DD'), 15, 'A2',  '11112'),
('D7', to_date( '2023/10/23', 'YYYY/MM/DD'), 10, 'A2',  '11116'),
('D8', to_date( '2023/10/20', 'YYYY/MM/DD'), 150, 'A5',  '11112'),
('D9', to_date( '2023/10/19', 'YYYY/MM/DD'), 120, 'A3',  '11111');

CREATE TABLE goes_to(donationID varchar(50) references Donation(donationID) on update
cascade, areaID varchar(50) references AllocationArea(areaID) on update cascade);

ALTER TABLE goes_to
ADD CONSTRAINT PK_donationarea
PRIMARY KEY (donationID, areaID);
```

```
INSERT INTO goes_to(donationID,areaID)
VALUES
('D1','A1'),
('D2','A2'),
('D3','A3'),
('D4', 'A4'),
('D5', 'A5'),
('D6', 'A2'),
('D7', 'A2'),
('D8', 'A5'),
('D9', 'A3');

CREATE TABLE donates_money_to(branchID varchar(50) references Branch(branchID) on
update cascade, areaID varchar(50) references AllocationArea(areaID) on update cascade);

INSERT INTO donates_money_to(branchID, areaID)
VALUES
('10001', 'A1'),
('10002', 'A2'),
('10003', 'A3'),
('10004', 'A4'),
('10005', 'A5'),
('10006', 'A2');

ALTER TABLE donates_money_to
ADD CONSTRAINT PK_brancharea
PRIMARY KEY (branchID, areaID);

CREATE TABLE donates(donorID varchar(50) REFERENCES Donor(donorID) on update
cascade, donationID varchar(50) REFERENCES Donation(donationID) on update cascade);

INSERT INTO donates(donorID, donationID)
VALUES
('11111', 'D1'),
('11112', 'D2'),
('11113', 'D3'),
('11114', 'D4'),
('11115', 'D5'),
('11116', 'D6'),
```

```
('11117', 'D7'),
('11118', 'D8');

ALTER TABLE donates
ADD CONSTRAINT PK_donordonation
PRIMARY KEY (donorID, donationID);
```

XI. ORM Implementation

```
from typing import List
from typing import Optional
from sqlalchemy import ForeignKey
from sqlalchemy import String, Integer
from sqlalchemy.orm import DeclarativeBase
from sqlalchemy.orm import Mapped
from sqlalchemy.orm import mapped_column
from sqlalchemy.orm import relationship
from sqlalchemy import create_engine
from sqlalchemy.orm import Session
from sqlalchemy import select

#DB Connection:
engine = create_engine("postgresql+psycopg2://postgres:(832354)Tm!!@localhost/postgres")

class Base(DeclarativeBase):
	pass

#Branch -

class Branch(Base):
	__tablename__ = "Branch"
```

```
    branchID: Mapped[int] = mapped_column(Integer, primary_key=True)
    branch_name: Mapped[str] = mapped_column(String(50))
    branch_phone_num: Mapped[str] = mapped_column(String(50))
    branch_address: Mapped[str] = mapped_column(String(50))

    allocationarea: Mapped[List["AllocationArea"]] = relationship(back_populates="branch",
cascade="all, delete-orphan")
    donor: Mapped[List["Donor"]] = relationship(back_populates="branch", cascade="all,
delete-orphan")
    manager: Mapped[List["Manager"]] = relationship(back_populates="branch",
cascade="all, delete-orphan")

#Donor -

class Donor(Base):
    __tablename__ = "Donor"

    donorID: Mapped[int] = mapped_column(String(50), primary_key=True)
    donor_name: Mapped[str] = mapped_column(String(50), nullable=True)
    donor_email: Mapped[str] = mapped_column(String(50), nullable=True)
    donor_phone_num: Mapped[str] = mapped_column(String(50), nullable=True)

    branchID: Mapped[int] = mapped_column(Integer, ForeignKey("Branch.branchID"))
    branch: Mapped["Branch"] = relationship(back_populates="donor")

    managerID: Mapped[str] = mapped_column(String(50),
ForeignKey("Manager.managerID"), nullable=True)
    manager: Mapped["Manager"] = relationship(back_populates="donor")

    donation: Mapped[List["Donation"]] = relationship(back_populates="donor",
cascade="all, delete-orphan")
```

```
#AllocationArea

class AllocationArea(Base):
	__tablename__ = "AllocationArea"

	areaID: Mapped[str] = mapped_column(String(50), primary_key=True)
	area_name: Mapped[str] = mapped_column(String(255), nullable=True)

	branchID: Mapped[int] = mapped_column(Integer, ForeignKey("Branch.branchID"))
	branch: Mapped["Branch"] = relationship(back_populates="allocationarea")

	donation: Mapped[List["Donation"]] = relationship(back_populates="allocationarea",
cascade="all, delete-orphan")

#Manager -

class Manager(Base):
	__tablename__ = "Manager"

	managerID: Mapped[str] = mapped_column(String(50), primary_key=True)
	manager_name: Mapped[str] = mapped_column(String(50), nullable=True)

	branchID: Mapped[int] = mapped_column(Integer, ForeignKey("Branch.branchID"))
	branch: Mapped["Branch"] = relationship(back_populates = "manager")

	donor: Mapped[List["Donor"]] = relationship(back_populates="manager", cascade="all,
delete-orphan")

#Donation -
```

```
class Donation(Base):
	__tablename__ = "Donation"

	donationID: Mapped[str] = mapped_column(String(50), primary_key=True)
	donation_date: Mapped[str] = mapped_column(String(50), nullable=True)
	donation_amt: Mapped[int] = mapped_column(Integer, nullable=True)

	areaID: Mapped[str] =
mapped_column(String(50),ForeignKey("AllocationArea.areaID"), nullable=True)
	allocationarea: Mapped["AllocationArea"] = relationship(back_populates="donation")
	donorID: Mapped[str] = mapped_column(String(50),ForeignKey("Donor.donorID"),
nullable=True)
	donor: Mapped["Donor"] = relationship(back_populates="donation")

Base.metadata.create_all(engine)

##INSERT DATA

#Branch --
with Session(engine) as session:
	one = Branch(
		branchID= 10001,
		branch_name= "Los Angeles",
		branch_phone_num= "4246567731",
		branch_address= "3438 San Marino Street",
		allocationarea= [AllocationArea(areaID = "A1")],
		donor= [Donor(donorID = "11111")],
		manager= [Manager(managerID = "M1")]
	)
	two = Branch(
		branchID = 10002,
```

```
        branch_name = "Las Vegas",
        branch_phone_num = "7023368547",
        branch_address = "10700 Space Odyssey Avenue",
        allocationarea = [AllocationArea(areaID = "A2")],
        donor= [Donor(donorID = "11112")],
        manager = [Manager(managerID = "M2")]
)
three = Branch(
        branchID = 10003,
        branch_name = "New York City",
        branch_phone_num = "2124962270",
        branch_address = "326 Theatre Drive",
        allocationarea = [AllocationArea(areaID = "A3")],
        donor= [Donor(donorID = "11113")],
        manager = [Manager(managerID = "M3")]
)
four = Branch(
        branchID = 10004,
        branch_name = "Philadelphia",
        branch_phone_num = "21564859012",
        branch_address = "801 North Bambrey Street",
        allocationarea = [AllocationArea(areaID = "A4")],
        donor= [Donor(donorID = "11114"), Donor(donorID = "11118")],
        manager = [Manager(managerID = "M4")]
)
five = Branch(
        branchID = 10005,
        branch_name = "Fort Wayne",
        branch_phone_num = "2606371994",
        branch_address = "914 West Gump Road",
        allocationarea = [AllocationArea(areaID = "A5")],
```

```
            donor= [Donor(donorID = "11115")],
            manager = [Manager(managerID = "M5")]
        )
        six = Branch(
            branchID = 10006,
            branch_name = "Washington D.C.",
            branch_phone_num = "2026541576",
            branch_address = "110 Maryland Avenue",
            allocationarea = [AllocationArea(areaID = "A6")],
                #can list more than one allocation area since this is the one branch to many
AA relation
            donor= [Donor(donorID = "11116"), Donor(donorID = "11117")],
            manager = [Manager(managerID = "M6")]
        )
        session.add_all([one, two, three, four, five, six])
        session.commit()

#AllocationArea --

with Session(engine) as session:
        stmt = select(AllocationArea).where(AllocationArea.areaID == "A1")
        A1 = session.scalars(stmt).one()
        A1.area_name = "Area 1"

        stmt = select(AllocationArea).where(AllocationArea.areaID == "A2")
        A2 = session.scalars(stmt).one()
        A2.area_name = "Area 2"

        stmt = select(AllocationArea).where(AllocationArea.areaID == "A3")
        A3 = session.scalars(stmt).one()
        A3.area_name = "Area 3"
```

```
    stmt = select(AllocationArea).where(AllocationArea.areaID == "A4")
    A4 = session.scalars(stmt).one()
    A4.area_name = "Area 4"

    stmt = select(AllocationArea).where(AllocationArea.areaID == "A5")
    A5 = session.scalars(stmt).one()
    A5.area_name = "Area 5"

    stmt = select(AllocationArea).where(AllocationArea.areaID == "A6")
    A6 = session.scalars(stmt).one()
    A6.area_name = "Area 6"
    session.commit()

#Manager --

with Session(engine) as session:
    stmt = select(Manager).where(Manager.managerID == "M1")
    M1 = session.scalars(stmt).one()
    M1.manager_name = "Eliza Cohen"

    stmt = select(Manager).where(Manager.managerID == "M2")
    M2 = session.scalars(stmt).one()
    M2.manager_name = "Trina Vega"

    stmt = select(Manager).where(Manager.managerID == "M3")
    M3 = session.scalars(stmt).one()
    M3.manager_name = "Neha Singh"

    stmt = select(Manager).where(Manager.managerID == "M4")
    M4 = session.scalars(stmt).one()
```

```
        M4.manager_name = "Owen Cline"

        stmt = select(Manager).where(Manager.managerID == "M5")
        M5 = session.scalars(stmt).one()
        M5.manager_name = "Layla Dyson"

        stmt = select(Manager).where(Manager.managerID == "M6")
        M6 = session.scalars(stmt).one()
        M6.manager_name = "Carson Smith"
        session.commit()

#Donor --

with Session(engine) as session:
        stmt = select(Donor).where(Donor.donorID == "11111")
        D1 = session.scalars(stmt).one()
        D1.donor_name = "Todd Smith"
        D1.donor_email = "tsmithy@hotmail.com"
        D1.donor_phone_num = "7089895446"
        D1.managerID = "M1"

        stmt = select(Donor).where(Donor.donorID == "11112")
        D2 = session.scalars(stmt).one()
        D2.donor_name = "Elena Baker"
        D2.donor_email = "bakersman@gmail.com"
        D2.donor_phone_num = "7089387495"
        D2.managerID = "M2"

        stmt = select(Donor).where(Donor.donorID == "11113")
        D3 = session.scalars(stmt).one()
        D3.donor_name = "Riley MacBeth"
```

```
D3.donor_email = "Wherearttthou@yahoo.com"
D3.donor_phone_num = "2698499847"
D3.managerID = "M2"

stmt = select(Donor).where(Donor.donorID == "11114")
D4 = session.scalars(stmt).one()
D4.donor_name = "John Macafee"
D4.donor_email = "macafeematters@gmail.com"
D4.donor_phone_num = "7080394495"
D4.managerID = "M3"

stmt = select(Donor).where(Donor.donorID == "11115")
D5 = session.scalars(stmt).one()
D5.donor_name = "Jack Ripper"
D5.donor_email = "beware@outlook.com"
D5.donor_phone_num = "7084549983"
D5.managerID = "M4"

stmt = select(Donor).where(Donor.donorID == "11116")
D6 = session.scalars(stmt).one()
D6.donor_name = "Ginny George"
D6.donor_email = "ggeorge@gmail.com"
D6.donor_phone_num = "5159389948"
D6.managerID = "M5"

stmt = select(Donor).where(Donor.donorID == "11117")
D7 = session.scalars(stmt).one()
D7.donor_name = "John Doe"
D7.donor_email = "jdoe@outlook.com"
D7.donor_phone_num = "7083455555"
D7.managerID = "M5"
```

```
        stmt = select(Donor).where(Donor.donorID == "11118")
        D8 = session.scalars(stmt).one()
        D8.donor_name = "Jane Doe"
        D8.donor_email = "janedoe@gmail.com"
        D8.donor_phone_num = "5159037485"
        D8.managerID = "M6"
        session.commit()

#Donation --

with Session(engine) as session:
        D1 = Donation(
                donationID = "D1",
                donation_date= "2023/10/20",
                donation_amt = 50,
        )
        D2 = Donation(
                donationID = "D2",
                donation_date= "2023/10/21",
                donation_amt = 100,
        )
        D3 = Donation(
                donationID = "D3",
                donation_date= "2023/10/20",
                donation_amt = 25,
        )
        D4 = Donation(
                donationID = "D4",
                donation_date= "2023/10/22",
                donation_amt = 50,
```

```
)
D5 = Donation(
        donationID = "D5",
        donation_date= "2023/10/23",
        donation_amt = 150,
)
D6  = Donation(
        donationID = "D6",
        donation_date= "2023/10/24",
        donation_amt = 15,
)
D7 = Donation(
        donationID = "D7",
        donation_date= "2023/10/23",
        donation_amt = 10,
)
D8 = Donation(
        donationID = "D8",
        donation_date= "2023/10/20",
        donation_amt = 150,
)
D9 = Donation(
        donationID = "D9",
        donation_date= "2023/10/19",
        donation_amt = 120,
        #areaID = [AllocationArea(areaID = "A3")],
        #donorID = [Donor(donorID = "11111")]
)
session.add_all([D1, D2, D3, D4, D5, D6, D7, D8, D9])

stmt = select(Donation).where(Donation.donationID == "D1")
```

```
D1 = session.scalars(stmt).one()
D1.areaID = "A1"
D1.donorID = "11111"

stmt = select(Donation).where(Donation.donationID == "D2")
D2 = session.scalars(stmt).one()
D2.areaID = "A2"
D2.donorID = "11112"

stmt = select(Donation).where(Donation.donationID == "D3")
D3 = session.scalars(stmt).one()
D3.areaID = "A3"
D3.donorID = "11113"

stmt = select(Donation).where(Donation.donationID == "D4")
D4 = session.scalars(stmt).one()
D4.areaID = "A4"
D4.donorID = "11114"

stmt = select(Donation).where(Donation.donationID == "D5")
D5 = session.scalars(stmt).one()
D5.areaID = "A5"
D5.donorID = "11115"

stmt = select(Donation).where(Donation.donationID == "D6")
D6 = session.scalars(stmt).one()
D6.areaID = "A2"
D6.donorID = "11112"

stmt = select(Donation).where(Donation.donationID == "D7")
D7 = session.scalars(stmt).one()
```

```
D7.areaID = "A2"
D7.donorID = "11116"

stmt = select(Donation).where(Donation.donationID == "D8")
D8 = session.scalars(stmt).one()
D8.areaID = "A5"
D8.donorID = "11112"

stmt = select(Donation).where(Donation.donationID == "D9")
D9 = session.scalars(stmt).one()
D9.areaID = "A3"
D9.donorID = "11111"

session.commit()
```

#Queries

Query (BranchDonations)

```
session = Session(engine)
branch = session.query(Branch)
for B in branch:
    stmt = (
        select(AllocationArea)
        .where(AllocationArea.branchID == B.branchID))
    AA = session.scalars(stmt).one()
    branch_AA = str(AA.areaID)
    stmt2 = (
    select(Donation.donation_amt)
    .where(Donation.areaID == branch_AA)
    )
    DD = session.scalars(stmt2).all()
    total = sum(DD)
    num = len(DD)
    print(str(B.branch_name) + " Donation Total: $" + str(total) + " with "+ str(num) + "
donation(s)")
    if num != 0:
       avg = total / num
        print(str(B.branch_name) + " Average Donation: $" + str(avg))
    else:
        print(str(B.branch_name) + " Average Donation: $0")
```

```
Los Angeles Donation Total: $50 with 1 donation(s)
Los Angeles Average Donation: $50.0
Las Vegas Donation Total: $125 with 3 donation(s)
Las Vegas Average Donation: $41.666666666666664
New York City Donation Total: $145 with 2 donation(s)
New York City Average Donation: $72.5
Philadelphia Donation Total: $50 with 1 donation(s)
Philadelphia Average Donation: $50.0
Fort Wayne Donation Total: $300 with 2 donation(s)
Fort Wayne Average Donation: $150.0
Washington D.C. Donation Total: $0 with 0 donation(s)
Washington D.C. Average Donation: $0
```

Query (DonorsPerAllocationArea)

```
# Create a session
session = Session (engine)
# Construct a query to get allocation areas with donor counts, ordered by donor count descending
stmt =  session.query (
        AllocationArea.areaID,
        AllocationArea.area_name,
        func.count (Donation.donorID).label ('donor_count'))
# Query to get a list of all allocation areas in descending order of donors count
   .outerjoin (Donation, Donation.area_ID == AllocationArea.areaID)
   .group_by (AllocationArea.areaID, AllocationArea.area_name)
   .order_by (func.count (Donation.donorID).desc ())
   .all ()
 # Retrieve all results
)
# Printing the list of allocation areas with the most donors in descending order
print("The Allocation Areas ranked by the highest number of donors in descending order are as
follows")
for area in stmt:
   print(f"Area ID: {area.areaID}, Area Name: {area.area_name}, Donor Count:
{area.donor_count}")
```

```
jazmincamacho@Jazmins-MacBook-Air Documents % python3 project.py
The Allocation Areas ranked by the highest number of donors in descending order are as follows:
Area ID: A2, Area Name: Area 2, Donor Count: 3
Area ID: A5, Area Name: Area 5, Donor Count: 2
Area ID: A3, Area Name: Area 3, Donor Count: 2
Area ID: A4, Area Name: Area 4, Donor Count: 1
Area ID: A1, Area Name: Area 1, Donor Count: 1
Area ID: A6, Area Name: Area 6, Donor Count: 0
```

Query (AllDonors)

```
session = Session(engine)
donors = session.query(Donor)
print("There are "+str(donors.count())+" donors.")
for donor in donors:
    print("Name: ")
    print(donor.donor_name)
    print("ID: ")
    print(donor.donorID)
    print("Email: ")
    print(donor.donor_email)
    print()
```

Output:

```
C:\Users\taylo\Desktop\ORM>python finalORMtablesandsomedata.py
There are 8 donors.
Name:
Todd Smith
ID:
11111
Email:
tsmithy@hotmail.com

Name:
Elena Baker
ID:
11112
Email:
bakersman@gmail.com

Name:
Riley MacBeth
ID:
11113
Email:
Wherearttthou@yahoo.com
```

```
Name:
John Macafee
ID:
11114
Email:
macafeematters@gmail.com

Name:
Jack Ripper
ID:
11115
Email:
beware@outlook.com

Name:
Ginny George
ID:
11116
Email:
ggeorge@gmail.com

Name:
John Doe
ID:
11117
Email:
jdoe@outlook.com
```

```
Name:
Jane Doe
ID:
11118
Email:
janedoe@gmail.com
```

Query (DonationInfo)

```
session = Session(engine)
donation = session.query(Donation)
print("These are the current donations in the database:")
for dono in donation:
        print("Donation ID: ", dono.donationID, "Donation Amount: ", dono.donation_amt,
"Donation Date: ", dono.donation_date, "Donation Area", dono.areaID)
```

Output:

```
(base) calicurro@MacBook-Pro-5 ~ % python3 untitled10.py
These are the current donations in the database:
Donation ID:  D1 Donation Amount:  50 Donation Date:  2023/10/20 Donation Area A1
Donation ID:  D2 Donation Amount:  100 Donation Date:  2023/10/21 Donation Area A2
Donation ID:  D3 Donation Amount:  25 Donation Date:  2023/10/20 Donation Area A3
Donation ID:  D4 Donation Amount:  50 Donation Date:  2023/10/22 Donation Area A4
Donation ID:  D5 Donation Amount:  150 Donation Date:  2023/10/23 Donation Area A5
Donation ID:  D6 Donation Amount:  15 Donation Date:  2023/10/24 Donation Area A2
Donation ID:  D7 Donation Amount:  10 Donation Date:  2023/10/23 Donation Area A2
Donation ID:  D8 Donation Amount:  150 Donation Date:  2023/10/20 Donation Area A5
Donation ID:  D9 Donation Amount:  120 Donation Date:  2023/10/19 Donation Area A3
(base) calicurro@MacBook-Pro-5 ~ %
```

Query (ManagerOfDonor)

```
# (Using the Select.join() method) Find the name and ID of the managers for donors "Jack Ripper" and "Ginny George"

session = Session(engine)

stmt1 = (
        select(Manager)
        .join(Manager.donor)
        .where(Donor.donor_name =="Jack Ripper"))
stmt2 = (
        select(Manager)
        .join(Manager.donor)
        .where(Donor.donor_name =="Ginny George"))
ManagerDonorJack = session.scalars(stmt1).one()
ManagerDonorGinny = session.scalars(stmt2).one()

print("The manager of Jack Ripper is " + ManagerDonorJack.manager_name + "with manager ID: " + ManagerDonorJack.managerID)
print("The manager of Ginny George: " + ManagerDonorGinny.manager_name+ "with manager ID: " + ManagerDonorGinny.managerID)
```

```
C:\Users\Pranati Mitta>python "C:\Users\Pranati Mitta\OneDrive - Loyola University Chicago\
Sophmore Sem 1\Comp 353\In class exxercises\sqlAlchemyORM.py"
The manager of Jack Ripper is Owen Clinewith manager ID: M4
The manager of Ginny George: Layla Dysonwith manager ID: M5
```

XII. ORM Queries and Brief Descriptions with Implementation

Query Name	Description	Output	Relations Accessed
AllDonors	A query summing the number of donors in the Donor Table and then listing their attributes	Total Number of Donors Name, ID, and Email	Donor
DonationInfo	This query reports each donation and it's relevant attributes	Donation ID, Donation Amount, Donation Date, Donation Area	Donation
ManagerOfDonor	A query which uses the Select.join() method to report the names and IDs of the managers for donors named "Jack Ripper" and "Ginny George".	Manager Name, Manager ID	Manager
BranchDonations	A query that reports the total donation amount for each branch, as well as the number of donations, and the average amount of each donation.	Branch Donation Total, Average Branch Donation, Total Number of Donations by Branch	Branch, AllocationArea, Donation
DonorsPerAllocationArea	A query that lists donors per Allocation Area.	Area ID, Area Name, Donor Count	Donors, AllocationArea