ASSUMPTIONS (Entities)

User:

- Each user has a unique UserID.
- Each user needs credentials (username/password) for authentication.
- Each user has an email for communication.
- Each user has a preferred HomeCurrency for reporting and currency conversion.

User is an entity because the user is central to the application. The user is not a piece of information that is stored as part of a trip or expense like an attribute. They have their own profile, which is why it is an entity.

Trip:

- Each trip has a unique TripID.
- Each trip is created and owned by one user (UserID)
- Each trip has a name, start/end dates, and a description (optional).

Trips are also a central element of the application. Expenses and bookings are both related to Trips, so we store it as an entity. We cannot store Trip as attributes for other entities because that would require a lot of attributes.

Booking:

- Each booking has a unique BookingID.
- Each booking is associated with a specific TripID. (each trip has a booking)
- Each booking has a type, date, cost, and currency.

Bookings represent expenses with extra attributes than the Expense entity. They would need to be on a trip, so we make it a separate entity.

TripUser:

- A user can be associated with multiple trips they don't “own” (shared trips).
- A trip can be shared with multiple users.
- Each user can have a specific permission level for each shared trip (edit/view access).

We need this as an entity in between User and Trip to enforce the many to many relationship between the two through allowing collaborative editing on the trip.

Expense:

- Each expense has a unique ExpenseID.
- Each expense is associated with a specific TripID. (each trip has an expense)

- Each expense has a date, amount, currency, category, and description.

Expenses are the primary reason for the application, since it is an expense tracker. Because they require many attributes, we store it as an entity. Again, storing Expense's attributes on other entities requires a lot of attributes.

Currency:

- Each exchange rate is unique for a given date and currency pair.
- The exchange rate is obtained from an external API.

We store all currencies separately, so we create an entity just for currency.

ASSUMPTIONS/JUSTIFICATIONS (Relationships)

User to Trip: Many to Many (resolved as two One to Many relationships with TripUser)

- Each user can create multiple trips and a trip can be shared by multiple users.
- Since users can participate in multiple trips, there can be multiple users in TripUser.
- Multiple users per trip just means they can collaborate on planning the trip.

Trip to Booking: One to Many

- Each trip can have multiple bookings associated with it (two flights, bus ride, hotel shuttle, etc)
- Since there are going to be multiple expenses, there can be multiple bookings as well.

Trip to Expense: One to Many

- Each trip can have multiple expenses associated with it.
- Many expenses are going to be made (food, accommodation, etc)

Booking is a weak entity, weak relation to Expense (not sure yet)?

- We can connect Booking to Expense by adding a Foreign Key relationship in Booking called ExpenseID which references Expense's ExpenseID. This means that there is an optional link between the two. Booking is still its own entity so that it could possibly be canceled.

RELATIONAL SCHEMA

User(UserID:INT [PK], Username:VARCHAR(255), Password:VARCHAR(255), Email:VARCHAR(255), HomeCurrency:VARCHAR(3))

TripUser(TripID:INT [PK][FK to Trip.TripID], UserID:INT [PK][FK to User.UserID], PermissionLevel:VARCHAR(50))

Trip(TripID:INT [PK], UserID:INT [FK to User.UserID], TripName:VARCHAR(255), StartDate:DATE, EndDate:DATE, Description:TEXT)

Booking(BookingID: INT [PK], TripID: INT [FK to Trip.TripID], BookingType: VARCHAR(255), Provider: VARCHAR(255), ConfirmationNumber: VARCHAR(255), BookingDate: DATE, Cost: DECIMAL(10,2), Currency: VARCHAR(3))

Expense(ExpenseID:INT [PK], TripID:INT [FK to Trip.TripID], Date:DATE, Amount:DECIMAL(10,2), Currency:VARCHAR(3), CategoryName:VARCHAR(255), Description:TEXT)

Currency(Date:DATE [PK], BaseCurrency:VARCHAR(3) [PK], TargetCurrency:VARCHAR(3) [PK], Rate:DECIMAL(10,6))

NORMALIZATION

We normalized our database to adhere to 3NF (third normal form). Our schema is already normalized considering the FDs.

The functional dependencies in the schema adhere to (3NF) because each non-key attribute is fully functionally dependent on the primary key of its respective table, and there are no transitive dependencies that exist. In each table, attributes are determined only by the primary key, meaning that there are no dependencies on other non-key attributes. In the User table, Username, Password, Email, and HomeCurrency all depend only on UserID, which prevents partial or transitive dependencies for the table. Similarly, in the Booking and Expense tables, attributes Cost, Currency, and Amount are directly determined by their respective primary keys (BookingID and ExpenseID). The TripUser table, which uses a composite primary key (TripID, UserID), ensures that PermissionLevel is fully dependent on both attributes and does not introduce unnecessary dependencies. The Currency table follows 3NF because the exchange rate is uniquely determined by the composite key (Date, BaseCurrency, TargetCurrency). Since there are no cases where a non-key attribute depends on another non-key attribute, the schema adheres to 3NF.

- User(UserID, Username, Password, Email, HomeCurrency)
    - FDs: {Date, BaseCurrency, TargetCurrency} → Rate
- Trip(TripID, UserID, TripName, StartDate, EndDate, Description)
    - FDs: TripID → UserID, TripName, StartDate, EndDate, Description
- TripUser(TripID, UserID, PermissionLevel)
    - FDs: {TripID, UserID} → PermissionLevel
- Booking(BookingID, TripID, BookingType, BookingDate, Cost, Currency)
    - FDs: BookingID → TripID, BookingType, BookingDate, Cost, Currency

- Expense(ExpenseID, TripID, Date, Amount, Currency, CategoryName, Description)
    - FDs: ExpenseID → TripID, Date, Amount, Currency, CategoryName, Description
- Currency(Date, BaseCurrency, TargetCurrency, Rate)
    - FDs: {Date, BaseCurrency, TargetCurrency} → Rate

ENTITIES AND ATTRIBUTES

User:

UserID (PK): Unique identifier for each user.

Username: User's login name.

Password: User's password (hashed)

Email: User's email address.

HomeCurrency: User's preferred currency.

Trip:

TripID (PK): Unique identifier for each trip.

UserID (FK): Foreign key referencing User.UserID, indicating the trip's owner.

TripName: Name of the trip.

StartDate: Trip's start date.

EndDate: Trip's end date.

Description: Description of the trip.

Booking:

BookingID (PK): Unique identifier for each booking.

TripID (FK): Foreign key referencing Trip.TripID, indicating the trip the booking belongs to.

BookingType: Type of booking (e.g., flight, hotel).

BookingDate: Date the booking was made.

Cost: Cost of the booking.

Currency: Currency of the booking.

Expense:

ExpenseID (PK): Unique identifier for each expense.

TripID (FK): Foreign key referencing Trip.TripID, indicating the trip the expense belongs to.

Date: Date the expense was incurred.

Amount: Amount of the expense.

Currency: Currency of the expense.

CategoryName: Category of the expense (e.g., food, accommodation).

Description: Description of the expense.

Currency:

Date (PK): Date of the exchange rate.

BaseCurrency: Base currency in the exchange rate.

TargetCurrency: Target currency in the exchange rate.

Rate: Exchange rate.

TripUser:

TripID (PK, FK): Foreign key referencing Trip.TripID. Part of the composite primary key.

UserID (PK, FK): Foreign key referencing User.UserID. Part of the composite primary key.

PermissionLevel: Level of access the user has to the trip.