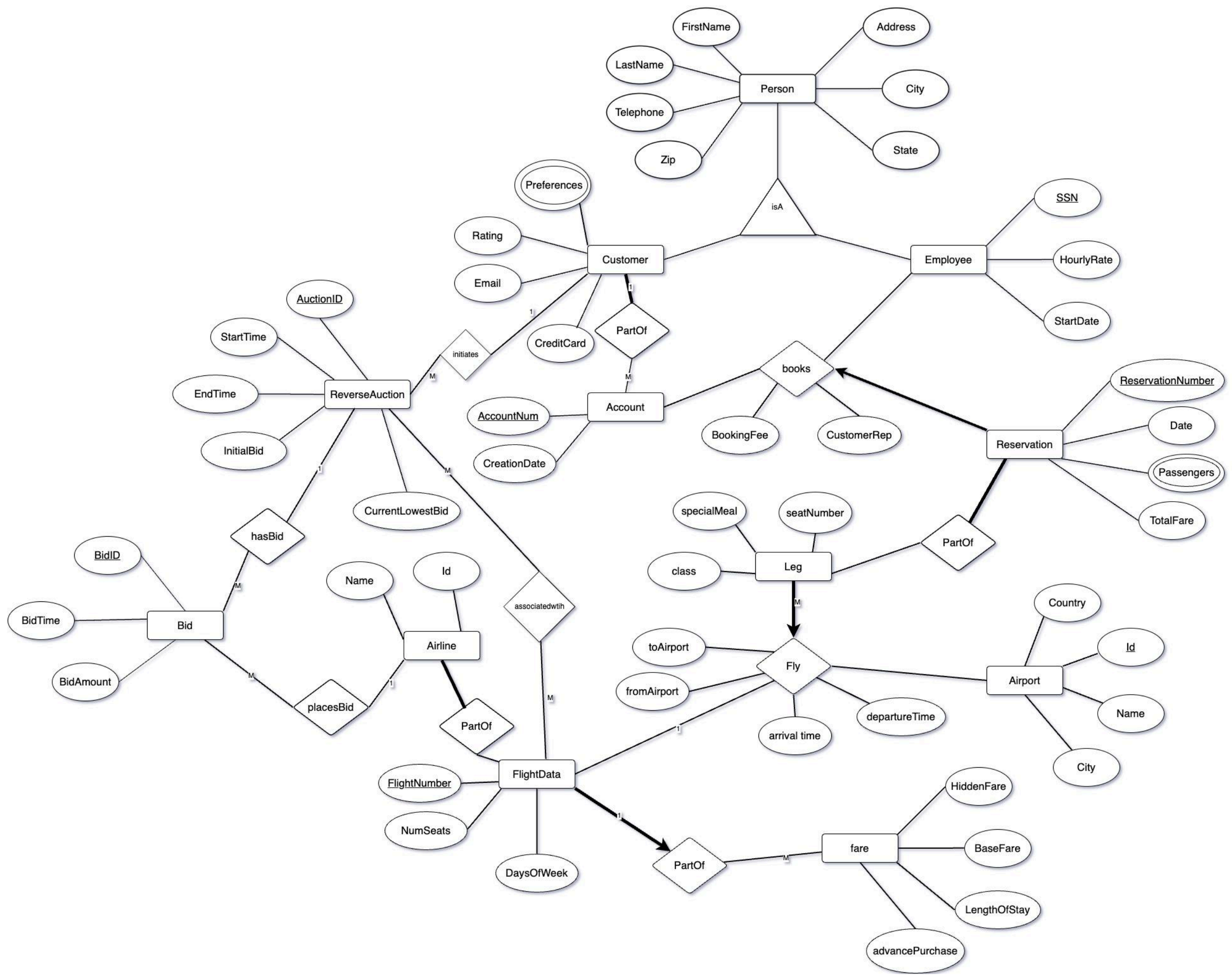
DATA MANAGEMENT
PROJECT ASSIGNMENT - 1

Team Name: SKYGauger

NAME	EMAIL	SBU ID
Atharv Raotole	atharv.raotole@stonybrook.edu	116480860
Parth Chavan	parth.chavan@stonybrook.edu	116558055
Dhananjay Sharma	dhananjay.sharma@stonybrook.edu	116740841

ENTITY RELATIONAL DIAGRAM (ER DIAGRAM)



RATIONALE OF ER DIAGRAM :

The Entity-Relationship (ER) diagram illustrates the structure and interactions between various components in the system. It includes primary entities, inheritance structures, and associative entities that establish relationships between core components. The following breakdown provides an in-depth understanding of each entity, its attributes, and how these entities are connected:

1. Person
    a. Attributes: FirstName, LastName, Address, City, State, Zip, Telephone
    b. Description: This entity captures the basic personal details of individuals, serving as a superclass for both Customer and Employee entities.

2. Customer *(inherits from Person)*
    a. Attributes: Rating, Email, Preferences
    b. Description: Extends the Person entity to include specific attributes related to customers, such as preferences and rating.

3. Employee *(inherits from Person)*
    a. Attributes: SSN, HourlyRate, StartDate
    b. Description: Represents employees, inheriting attributes from Person and adding employment-related attributes.

4. Account
    a. Attributes: AccountNum, CreationDate, CreditCard
    b. Description: Captures information about customer accounts, including account number, creation date, and associated credit card details.

5. Reservation
    a. Attributes: ReservationNumber, Date, Passengers, TotalFare
    b. Description: Represents reservations made by customers, recording the total fare and number of passengers.

6. FlightData
    a. Attributes: FlightNumber, NumSeats, DaysOfWeek
    b. Description: Stores information related to flights, including flight number, number of seats, and operating days.

7. Leg
    a. Attributes: Class, SeatNumber, SpecialMeal
    b. Description: Describes individual legs of a flight, including seat number and special meal preferences.

8. Airport
    a. Attributes: Id, Name, City, Country
    b. Description: Represents airports with unique identifiers, names, and location information.

9. Fare
    a. Attributes: BaseFare, HiddenFare, LengthOfStay, AdvancePurchase
    b. Description: Contains fare details for flights, such as base fare, hidden charges, and applicable conditions like advance purchase requirements.

10. ReverseAuction
    a. Attributes: AuctionID, StartTime, EndTime, InitialBid, CurrentLowestBid
    b. Description: Captures details of reverse auctions for booking fares, including starting and ending times, as well as the current lowest bid.

11. Bid
    a. Attributes: BidID, BidAmount, BidTime
    b. Description: Records individual bids made during reverse auctions, linking them to customers.

12. Fly (Associative Entity)
    a. Attributes: ArrivalTime, DepartureTime
    b. Description: Acts as an associative entity linking flights to airports, including flight arrival and departure times.

Relationships

The relationships within the diagram describe how entities interact with each other:

books : Links Employee to Reservation, representing employees managing customer reservations.

PartOf : Links Customer to Reservation, FlightData to Leg, and Fare to FlightData, representing hierarchical and associative relationships.

placesBid : Links Customer to ReverseAuction, depicting the bidding process by customers in reverse auctions.

RELATIONAL MODEL (LOGICAL DESIGN) :

Below is the description for the above framework of entity relationship diagram.

Airlines Table:

```
CREATE TABLE Airline (
   AirlineID  CHAR(2),
   AirlineName  VARCHAR(100)  NOT NULL,
   PRIMARY KEY  (AirlineId)
);
```

Airports Table:

```
CREATE TABLE Airport (
   AirportID  CHAR(3) ,
   AirportName  VARCHAR(100)  NOT NULL,
   City  VARCHAR(100),
   Country  VARCHAR(100),
   PRIMARY KEY  (AirportID)
);
```

Flights Table:

```
CREATE TABLE Flight (
   FlightNumber  VARCHAR(10),
   AirlineID  CHAR(2),
   NumberOfSeats  INTEGER,
   DaysOfWeek  VARCHAR(20),
   BaseFare  DECIMAL(10, 2),
   HiddenFare  DECIMAL(10, 2),
   AdvancePurchase  INTEGER,
   LengthOfStay  INTEGER,
   PRIMARY KEY  (FlightNumber, AirlineID),
   FOREIGN KEY  (AirlineID)  REFERENCES  Airline(AirlineID)  ON  DELETE  CASCADE
);
```

Customer Table:

```
CREATE TABLE Customer (
   AccountNumber  INTEGER  AUTO_INCREMENT,
  LastName  VARCHAR(100)  NOT  NULL,
  FirstName  VARCHAR(100)  NOT  NULL,
  Address  VARCHAR(255),
  City  VARCHAR(100),
  State  VARCHAR(50),
  ZipCode  VARCHAR(10),
  Telephone  VARCHAR(20),
  EmailAddress  VARCHAR(100) UNIQUE,
  AccountCreationDate  DATE,
  CreditCardNumber  VARCHAR(16),
  Rating  INTEGER,
  PRIMARY KEY  (AccountNumber)
);
```

As in the information document it is mentioned that preferences can take multiple values, so here we are using Normalization technique. This maintains high data integrity, query efficiency, and flexibility. Next two tables i.e. Preference table and CustomerPreferences table are to implement Normalization.

Preference Table:

```
CREATE  TABLE  Preference (
  PreferenceID  INTEGER  AUTO_INCREMENT,
  PreferenceName  VARCHAR(100)  NOT NULL  UNIQUE,
  PRIMARY KEY  (PreferenceID)
);
```

CustomerPreferences Table:

```
CREATE  TABLE  CustomerPreferences (
  AccountNumber  INTEGER,
  PreferenceID  INTEGER,
  PRIMARY KEY  (AccountNumber, PreferenceID),
      FOREIGN  KEY  (AccountNumber)  REFERENCES  Customer(AccountNumber)  ON  DELETE
CASCADE,
  FOREIGN KEY (PreferenceID) REFERENCES Preference(PreferenceID) ON DELETE CASCADE
);
```

Employee Table:

```
CREATE TABLE Employee (
   SSN  CHAR(9),
   LastName  VARCHAR(100)  NOT  NULL,
   FirstName  VARCHAR(100)  NOT  NULL,
   Address  VARCHAR(255),
   City  VARCHAR(100),
   State  VARCHAR(50),
   ZipCode  VARCHAR(10),
   Telephone  VARCHAR(20),
   StartDate  DATE,
   HourlyRate  DECIMAL(5, 2),
   PRIMARY KEY  (SSN)
);
```

Bid Table: (Reverse Auction)

```
CREATE TABLE Bid (
   BidId  INTEGER  AUTO_INCREMENT,
   AuctionID  INTEGER,
   AccountNumber  INTEGER,
   BidTime  DATETIME,
   BidAmount  DECIMAL(10, 2),
   PRIMARY KEY  (BidId),
  FOREIGN KEY (AccountNumber) REFERENCES  Customer(AccountNumber) ON DELETE
CASCADE
);
```

Reservations Table:

```
CREATE TABLE Reservation (
   ReservationNumber  INTEGER  AUTO_INCREMENT,
   ReservationDate  DATE  NOT NULL,
   TotalFare  DECIMAL(10, 2)  NOT NULL,
   BookingFee  DECIMAL(5, 2),
   FareRestrictions  VARCHAR(255),
   AccountNumber  INTEGER,
   CustomerRepresentative  VARCHAR(9),
   PRIMARY KEY  (ReservationNumber),
   FOREIGN KEY (AccountNumber) REFERENCES Customer(AccountNumber) ON DELETE
CASCADE,
   FOREIGN KEY (CustomerRepresentative) REFERENCES Employee(SSN) ON DELETE  SET NULL
);
```

Flight Legs Table:

```
CREATE  TABLE  Leg (
   LegID  INTEGER  AUTO_INCREMENT,
   FlightNumber  VARCHAR(10),
   FromAirport  CHAR(3),
   ToAirport  CHAR(3),
   DepartureTime  DATETIME,
   ArrivalTime  DATETIME,
   SeatNumber  VARCHAR(10),
   Class  VARCHAR(20),
   SpecialMeal  VARCHAR(100),
   PRIMARY KEY  (LegID),
  FOREIGN KEY (FlightNumber) REFERENCES Flight(FlightNumber) ON DELETE CASCADE,
   FOREIGN KEY (FromAirport) REFERENCES Airport(AirportID) ON DELETE CASCADE,
   FOREIGN KEY (ToAirport) REFERENCES Airport(AirportID) ON DELETE CASCADE,
   CHECK  (Class IN ('Economy', 'Business', 'First'))
);
```

Reservation Leg Table:

```
CREATE  TABLE  Reservation_Leg (
   ReservationNumber  INTEGER,
   LegID  INTEGER,
   PRIMARY KEY  (ReservationNumber, LegID),
   FOREIGN KEY  (ReservationNumber) REFERENCES Reservation(ReservationNumber) ON
DELETE CASCADE,
   FOREIGN KEY (LegID) REFERENCES Leg(LegID) ON DELETE CASCADE
);
```

Indexing *(to improve query performance)*:

```
CREATE INDEX index_flight_airlineid ON Flight (AirlineID);
CREATE INDEX index_flight_flightnumber_airlineid ON Flight (FlightNumber, AirlineID);

CREATE INDEX index_customer_accountnumber ON Customer (AccountNumber);

CREATE INDEX index_bid_accountnumber ON Bid (AccountNumber);

CREATE INDEX index_reservation_accountnumber ON Reservation (AccountNumber);
CREATE INDEX index_reservation_customerrepresentative ON Reservation (CustomerRepresentative);

CREATE INDEX index_reservationleg_reservationnumber ON Reservation_Leg (ReservationNumber);
CREATE INDEX index_reservationleg_legid ON Reservation_Leg (LegID);

CREATE INDEX index_leg_flightnumber ON Leg (FlightNumber);
CREATE INDEX index_leg_fromairport ON Leg (FromAirport);
CREATE INDEX index_leg_toairport ON Leg (ToAirport);
CREATE INDEX index_leg_from_to_airport ON Leg (FromAirport, ToAirport);
```

FUNCTIONAL DEPENDENCIES (FDS)

1. Person Entity:
   - PersonID → {FirstName, LastName, Address, City, State, Zip, Telephone}
2. Customer Entity:
   - CustomerID → PersonID
   - PersonID → {Rating, Email, Preferences}
3. Employee Entity:
   - EmployeeID → PersonID
   - PersonID → {SSN, HourlyRate, StartDate}
4. Account Entity:
   - AccountID → {AccountNum, CreationDate, CreditCard}
5. Reservation Entity:
   - ReservationID → {ReservationNumber, Date, Passengers, TotalFare, BookingFee, CustomerRep, CustomerID}
   - CustomerID → {ReservationID, ReservationNumber}
6. FlightData Entity:
   - FlightID → {FlightNumber, NumSeats, DaysOfWeek}
7. Leg Entity:
   - LegID → {Class, SeatNumber, SpecialMeal, FlightID}
   - FlightID → {LegID}
8. Airport Entity:
   - AirportID → {Name, City, Country}
9. Fare Entity:
   - FareID → {BaseFare, HiddenFare, LengthOfStay, AdvancePurchase, FlightID}
   - FlightID → {FareID}
10. ReverseAuction Entity:
    - AuctionID → {StartTime, EndTime, InitialBid, CurrentLowestBid, CustomerID}
    - CustomerID → {AuctionID}
11. Bid Entity:
    - BidID → {BidAmount, BidTime, AuctionID}
    - AuctionID → {BidID, CurrentLowestBid}
12. Fly Entity (Associative):
    - FlyID → {ArrivalTime, DepartureTime, FlightID, AirportID}
    - FlightID → {FlyID, ArrivalTime, DepartureTime}
    - AirportID → {FlyID}

INTEGRITY CONSTRAINTS :

Integrity constraints are essential rules defined in a database to maintain the accuracy and consistency of data. They ensure that data adheres to certain conditions and relationships. Below are the main types of integrity constraints, including referential integrity:

1. Entity Integrity:
Example: In the **Customer** table, **AccountNumber** serves as the primary key, ensuring every customer has a unique identifier.

2. Referential Integrity:
Example: In the Flight table, the **AirlineID** foreign key must reference an existing **AirlineID** in the **Airline** table. If an airline is deleted, any associated flights can also be deleted due to the ON DELETE CASCADE rule.

3. Domain Integrity:
Example: In the **Leg** table, the **Class** column has a CHECK constraint that restricts values to 'Economy', 'Business', or 'First'.

Summary of Referential Integrity:
Referential integrity ensures valid relationships between tables by preventing orphaned records, requiring foreign key values to match existing primary key records. It is enforced through cascading actions, determining how dependent records are affected when a referenced record is updated or deleted (e.g., ON DELETE CASCADE, ON UPDATE CASCADE, or SET NULL). This helps maintain data consistency and prevents errors from invalid relationships.

THE RATIONALE FOR RELATIONAL MODEL :

1. Entity Representation:
   Each entity in the ER diagram (e.g., `Person`, `Customer`, `Employee`, `Account`, `Reservation`, `FlightData`, etc.) is converted into a separate table in the relational model.

2. Primary Keys:
   Each table has a unique primary key to identify each record, ensuring that no two records are identical.

3. Foreign Keys and Relationships:
   Foreign keys are used to link related tables, such as linking `Customer` to `Account` and `Reservation` or linking `FlightData` to `Fare`. This enforces the relationships and ensures data consistency.

4. Inheritance:
   `Customer` and `Employee` inherit attributes from `Person` to avoid redundancy and manage shared details like names and addresses.

5. Associative Entities:
   Associative entities like `Fly` and `Bid` are used to handle many-to-many relationships, ensuring all interactions between entities (e.g., flights and airports, customers and auctions) are correctly tracked.

6. Integrity Constraints:
   Constraints like primary keys, foreign keys, and check constraints are added to enforce data validity and maintain referential integrity.

7. Normalization:
   The model is designed to reduce redundancy by organizing data into related tables, minimizing data duplication and making the database more efficient.

8. Handling Complex Relationships:
   Complex relationships such as `books`, `PartOf`, and `placesBid` are represented using foreign keys to maintain clear and accurate connections between entities.

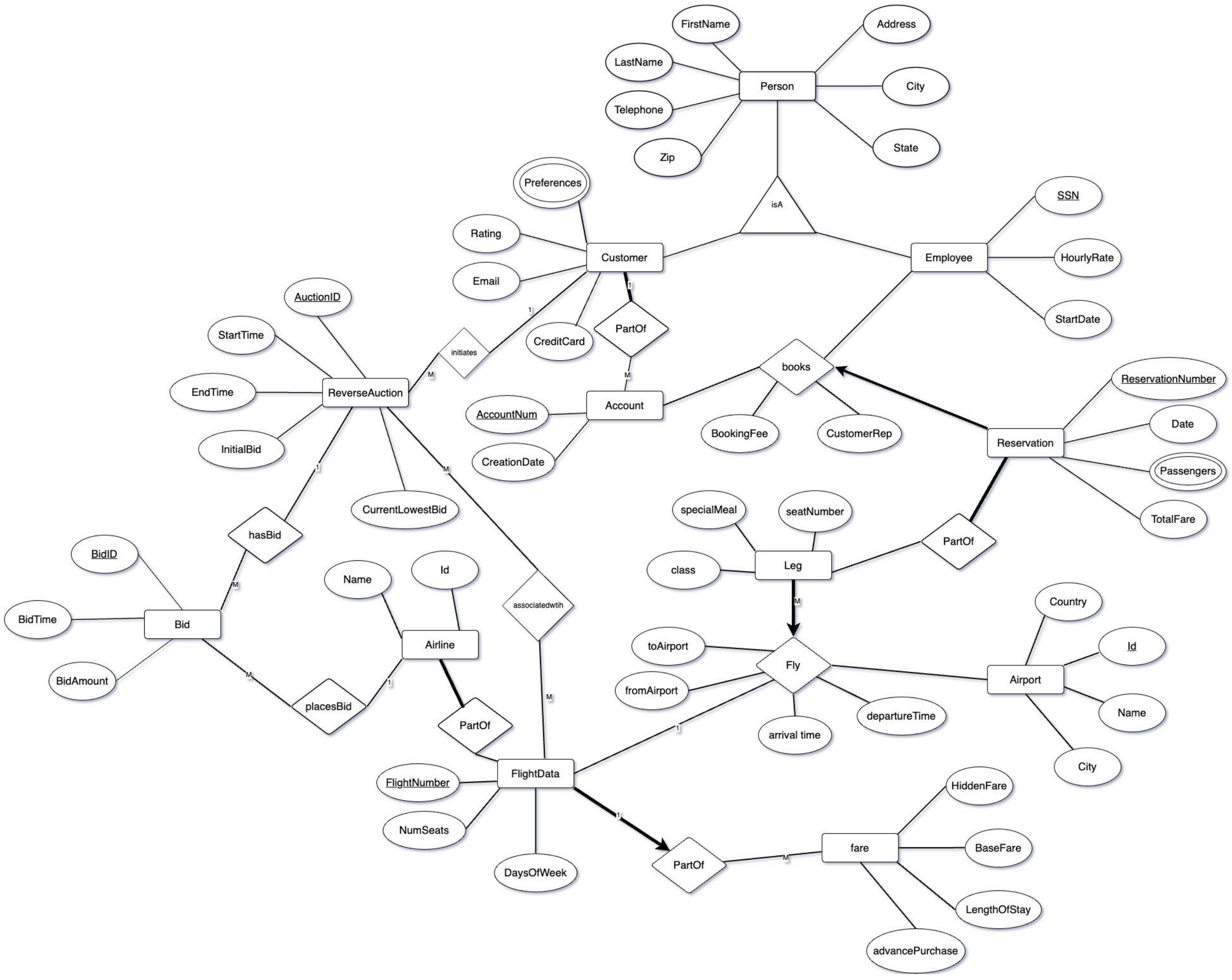
9. Well-handled many-to-many relationships with junction tables, ensuring that complex
   queries between entities remain efficient.

10. Partitioning and caching techniques that further optimize query performance,
    especially for high-demand queries.

CONCLUSION :

The SkyGauger.com online reservation system offers a user-friendly platform for booking airline reservations, integrating functionalities like flight search, customer preferences, seat reservations, and bid auctions. Its optimized database ensures efficient transactions and data integrity while accommodating growing demand. With advanced fare calculations and robust customer management, the system simplifies the booking process and enhances operational transparency for airlines.

ER DIAGRAM


FirstName
Address
LastName
Person
City
Telephone
Zip
State
Preferences
isA
SSN
Rating
Customer
Employee
HourlyRate
Email
AuctionID
StartDate
StartTime
initiates
CreditCard
PartOf
books
ReservationNumber
EndTime
ReverseAuction
AccountNum
Account
BookingFee
CustomerRep
Reservation
Date
InitialBid
CreationDate
Passengers
CurrentLowestBid
specialMeal
seatNumber
TotalFare
hasBid
PartOf
BidID
class
Leg
Name
Id
associatedwtih
Country
BidTime
Bid
Airline
toAirport
Id
Fly
BidAmount
placesBid
fromAirport
Airport
PartOf
arrival time
departureTime
Name
FlightNumber
FlightData
City
HiddenFare
NumSeats
fare
BaseFare
PartOf
DaysOfWeek
LengthOfStay
advancePurchase
1
M