`

Flight Reservation System Documentation

CIS 280 Project: A flight reservation database system.

Department_Team ID:

General_21

Team Members:

محمد هشام صلاح عبدالحميد

Seat Num.: 20201701245 Section Num.: 26

مازن محسن إبراهيم سعد

Seat Num.: 20201701229 Section Num.: 22

عمر ياسر حامد رمزي عبدالعزيز

Seat Num.: 20201701195 Section Num.: 19

سلمى ايمن محمد السيد العسال

Seat Num.: 20201700345 Section Num.: 13

يحيى إدريس مختار صديق

Seat Num.: 20201700982 Section Num.: 33

يوسف محمود مدبولي مصطفى

Seat Num.: 20201701037 Section Num.: 35

T.A.: Nagwa Moustafa *Dr.: Huda Amin*

`

Table of Contents

`

System Description & Mini-World Assumptions

The Flight Reservation System acts as the middleman between different airlines in the world and the end-user (agent). It assists the agent in the ticket reservation process.

First, we need to create an Airline database. We store each airline's name, location, and give it a unique ID. An agent *reserves* the ticket which *indicates* the flight. Every agent may *reserve* one or many tickets. Every ticket is *issued* by a single airline. Agents *purchase* tickets by different payment choices.

We store every agent's name, age, e-mail, phone number, and their unique ID and passport number. Each ticket contains the class type, seat number, and its unique ID. Every payment process contains information about the amount paid, its date and a unique ID. We also keep track of where each flight departs from and arrives at, its transit info, date and arrival/departure time.

Each flight is *flown* by an aircraft which *belongs* to an airline. Each flight *arrives* at a specific airport. Each aircraft *belongs* to only one airline, but each airline may have many aircrafts. We store each aircraft's model, capacity and its unique ID.

`

Proper Naming of Schema Constructs

The choice of names for entity types, attributes, relationship types, and (particularly) roles is not always straightforward. One should choose names that convey, as much as possible, the meanings attached to the different constructs in the schema. We chose to use singular names for entity types, rather than plural ones, because the entity type name applies to each individual entity belonging to that entity type. In our ER diagram, we used the convention that entity type and relationship type names are in uppercase letters, attribute names are capitalized, and role names are in lowercase letters.

As a general practice, given a narrative description of the database requirements, the nouns appearing in the narrative tend to give rise to entity type names, and the verbs tend to indicate names of relationship types. Attribute names generally arise from additional nouns that describe the nouns corresponding to entity types.

Another naming consideration involves choosing relationship names to make the ER diagram of the schema readable from left to right and from top to bottom. We have generally followed this guideline.

`

Entities & Attributes

TICKET

Ticket_ID

Class_Type

Seat_Num

INDICATES

Flight_Num

RESERVED_BY

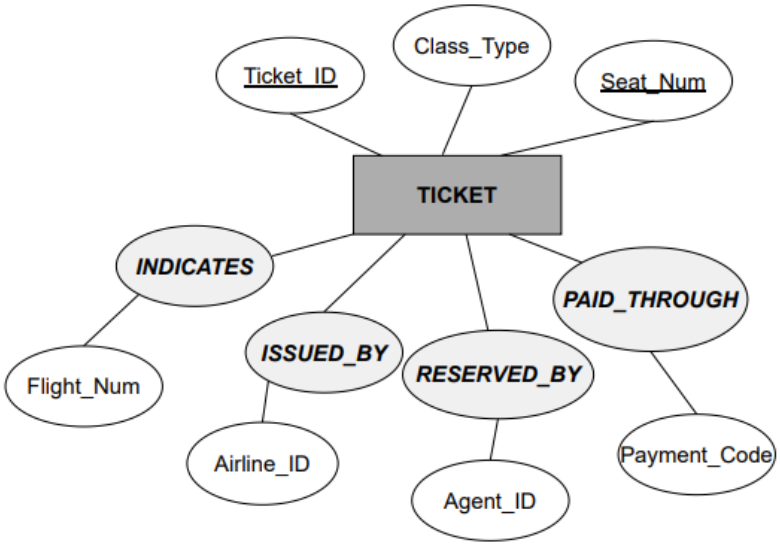
Agent_ID

ISSUED_BY

Airline_ID

PAID_THROUGH

Payment_Code



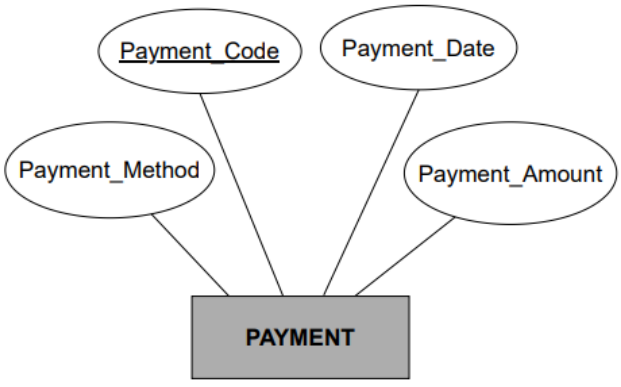
PAYMENT

Payment_Code

Payment_Method

Payment_Amount

Payment_Date



`

FLIGHT

Flight_Num

Flight_Source

Flight_Destination

Departure_Time

Arrival_Time

Estimated_Time

Flight_Date

Transit (Multivalue)

- Transit_Location
- Transit_Time

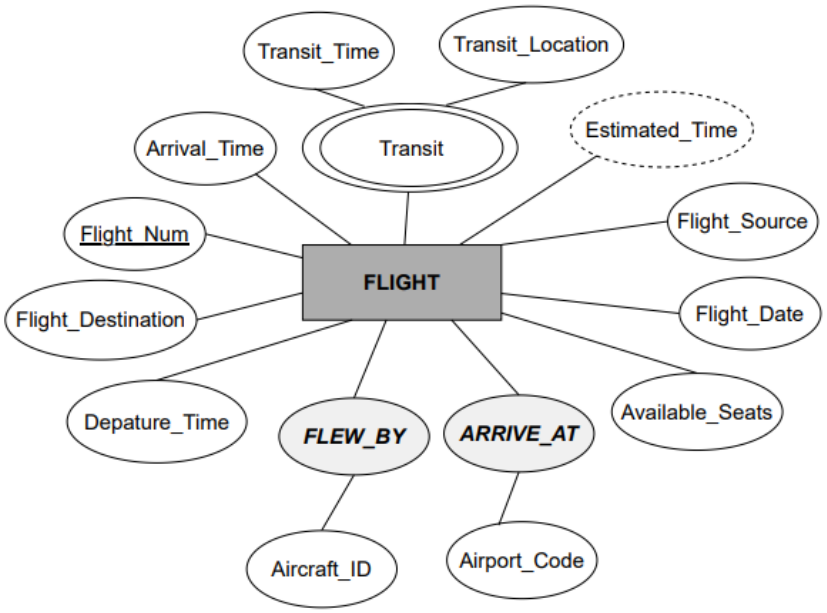
Available_Seats

FLEW_BY

- Aircraft_ID

ARRIVE_AT

- Airport_Code



AIRCRAFT

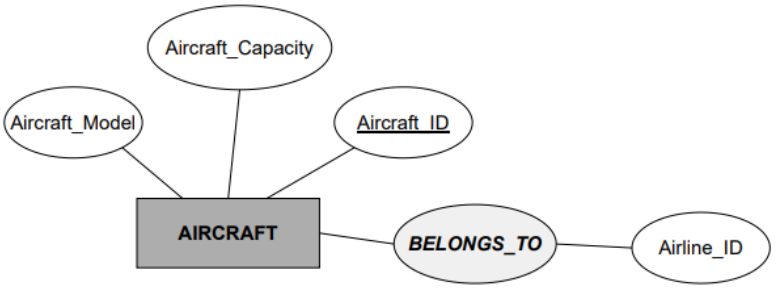
Aircraft_ID

Aircraft_Model

Aircraft_Capacity

BELONGS_TO

- Airline_ID



`

AGENT

Agent_ID

Passport_Num

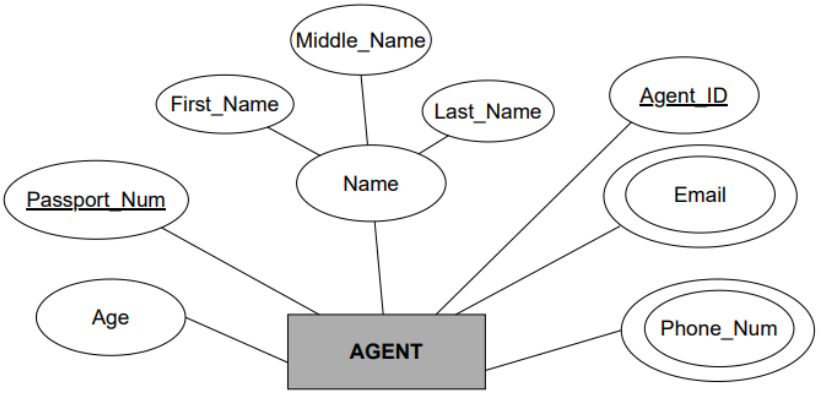
Name

- First_Name
- Middle_Name
- Last_Name

Age

Phone_Num (Multivalue)

Email (Multivalue)

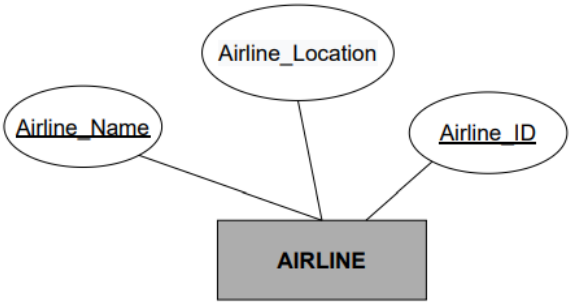


AIRLINE

Airline_ID

Airline_Name

Airline_Location

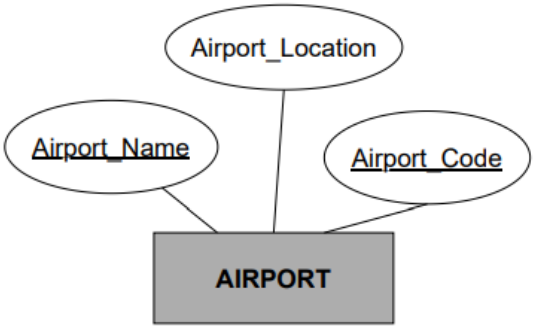


AIRPORT

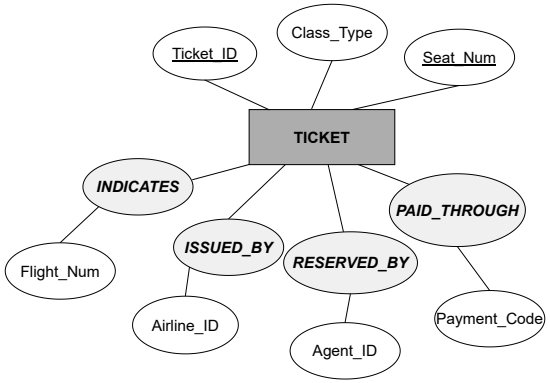
Airport_Code

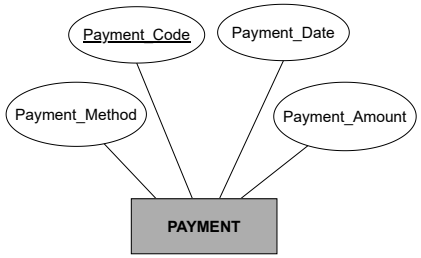
Airport_Name

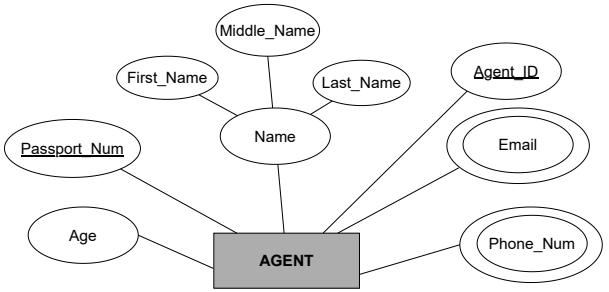
Airport_Location

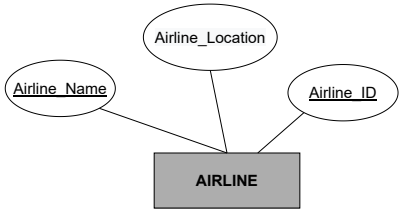


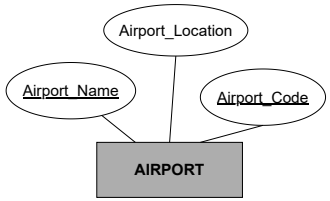
Initial Design of Entity Types

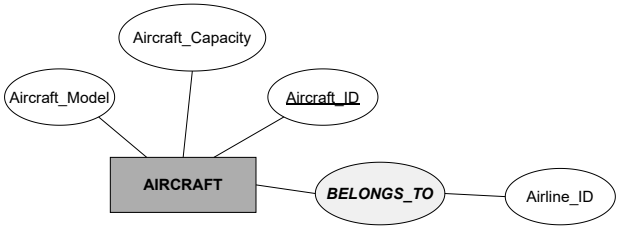


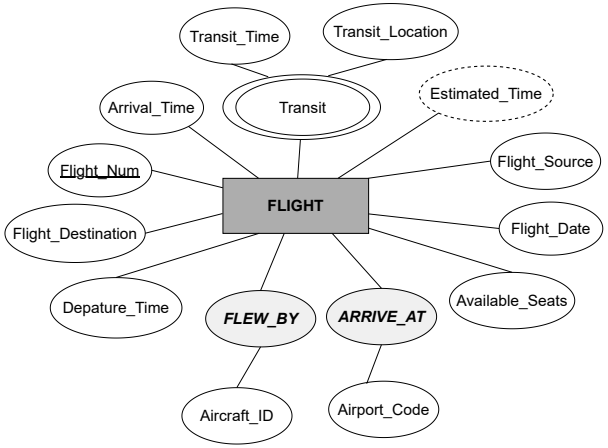












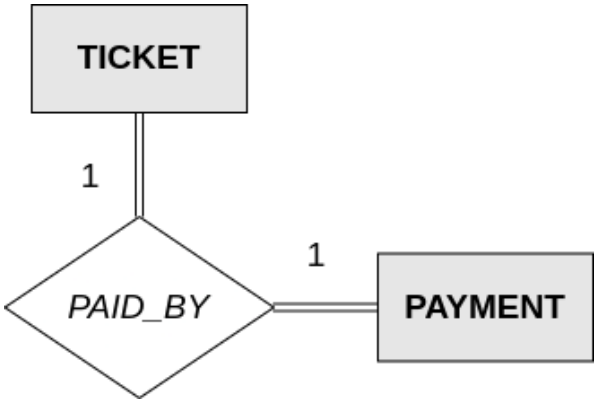
`

Entities' Relationships & Cardinality Ratio Calculations

One-to-One Relationships:

- *TICKET-PAYMENT:*
  Each Ticket is paid through one Payment, but each Payment can pay only one Ticket.

TICKET	PAYMENT
1	1
1	1
1	1

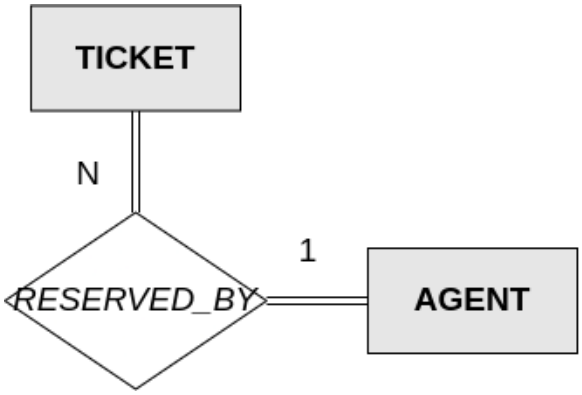


`

One-to-Many Relationships:

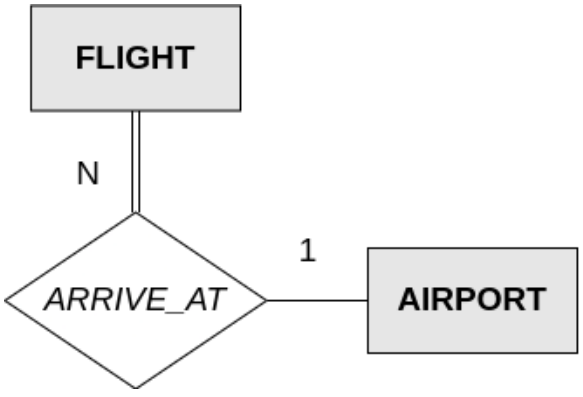
- *TICKET-AGENT:*
  Each Ticket is reserved by one Agent, but each Agent can reserve many Tickets.

TICKET	AGENT
1	1
N	1
N	1



- *FLIGHT-AIRPORT:*
  Each Flight arrives at one Airport, but each Airport can contain many Flights that arrive within.

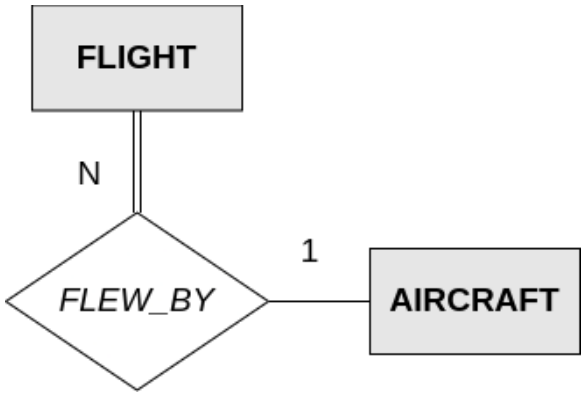
FLIGHT	AIRPORT
1	1
N	1
N	1



`

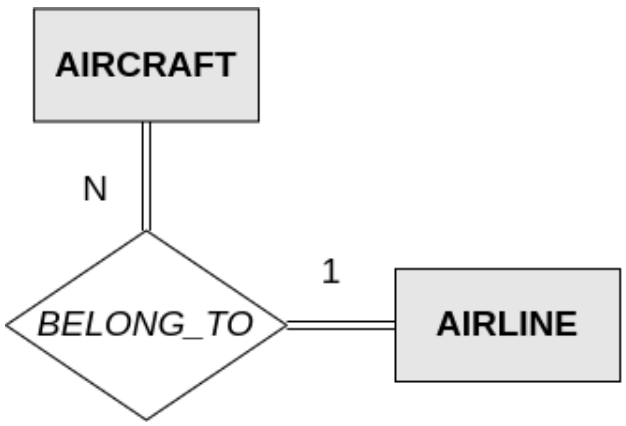
- *FLIGHT-AIRCRAFT:*
  Each Flight is flown by one Aircraft, but each Aircraft can fly many Flights.

FLIGHT	AIRCRAFT
1	1
N	1
N	1



- *AIRCRAFT-AIRLINE:*
  Each Aircraft belongs to one Airline, but each Airline owns many Aircrafts.

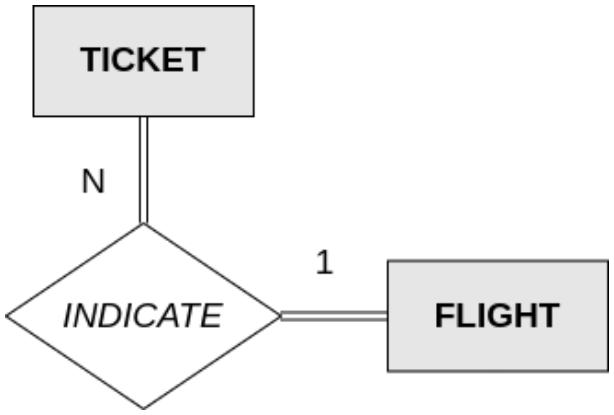
AIRCRAFT	AIRLINE
1	1
N	1
N	1



`

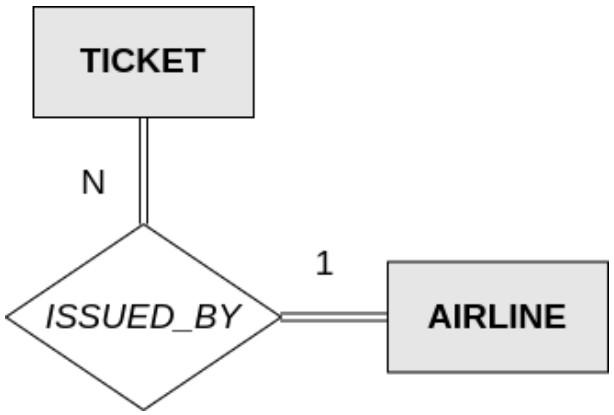
- *TICKET-FLIGHT:*
  Each Ticket indicates one Flight, but each Flight requires many Tickets.

TICKET	FLIGHT
1	1
N	1
N	1

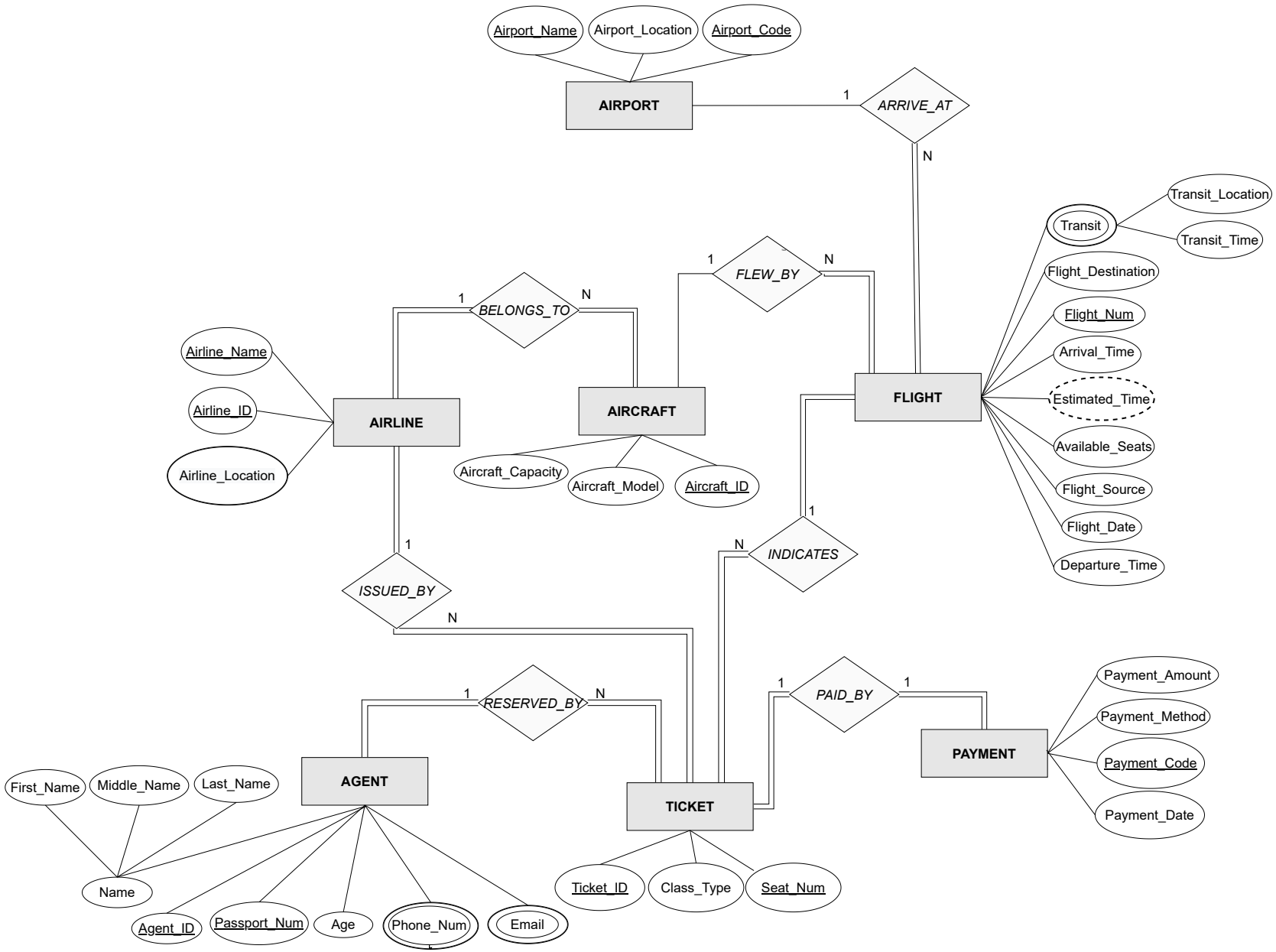


- *TICKET-AIRLINE:*
  Each Ticket is issued by one Airline, but each Airline can issue many Tickets.

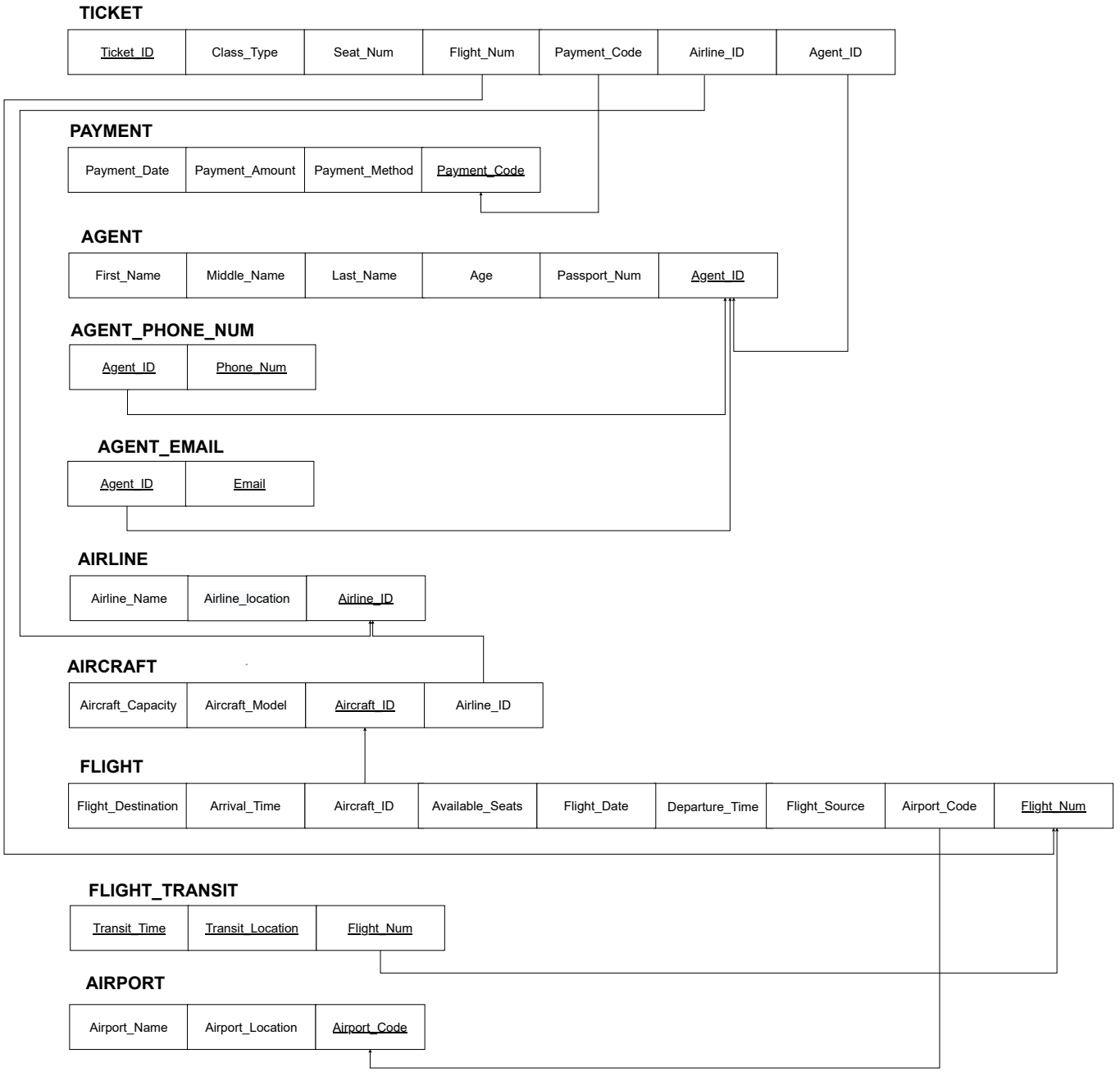
TICKET	AIRLINE
1	1
N	1
N	1



Refined Entity-Relationship Diagram (ERD)



System Schema



`

Normalization

Database normalization is a technique for creating database tables with suitable columns and keys by decomposing a large table into smaller logical units. The process also considers the demands of the environment in which the database resides.

Normalization is an iterative process. Commonly, normalizing a database occurs through a series of tests. Each subsequent step decomposes tables into more manageable information, making the overall database logical and easier to work with.

Step 1: First Normal Form (1NF)
First normal form (1NF) is considered to be part of the formal definition of a relation in the basic flat relational model. The relation schema is in the first normal form (1NF) as it disallows the existence of composite or multivalued attributes.

Step 2: Second Normal Form (2NF)
The relation schema is in the second normal form (2NF) as it is in the first normal form (1NF) and every non-prime attribute in the relation is fully functionally dependent on the primary key.

`

Step 3: Third Normal Form (3NF)
The relation schema is in the third normal form (3NF) as it is in the second normal form (2NF) and no non-prime attribute in the relation is transitively dependent on the primary key.

Step 4: Boyce-Codd Normal Form (BCNF)
The relation schema is in the Boyce-Codd normal form (BCNF) as it is in the third normal form (3NF) and whenever a function dependency X -> A holds in the relation, then X is a superkey of the relation.

Conclusion
The relation schema is in the Boyce-Codd normal form (BCNF) as it disallows the existence of composite or multivalued attributes, every non-prime attribute in the relation is fully functionally dependent on the primary key, no non-prime attribute in the relation is transitively dependent on the primary key, and whenever a function dependency X -> A holds in the relation, then X is a superkey of the relation.