**Design decisions:**

Airline, airport, and aircraft reflect the need to represent these different objects in the enterprise. I have included a small set of attributes based on the requirements (and also based on my own knowledge of the data). The distance between two airports is kept in the "dist" relationship set. The fact that an airline has an airport as a hub is in the "hub" relationship set. Ownership of aircraft by airlines is in the "fleet" relationship set (see below for more detail on this). I have gone easy on constraints here, so that an airline does not necessarily have to have a hub, an airport does not have to be a hub, we do not need to know the distance between every pair of airports, and aircraft and airlines can exist without being related to airlines or aircraft (respectively).

Flights are stored in an entity set with one-to-many, total participation relationships with two airports, an aircraft, and an airline. This reflects the requirement that each flight be scheduled by a single airline on a single aircraft from one airport to another. The flight entity set could’ve alternatively been modeled as a relationship set between airlines, airports, aircraft, and fares; I chose to model it as an entity set because that prevents some redundancy: since there can be several different fares on each flight, modeling flights as relationship sets would require that all airports, airlines, and aircraft involved in the flight be duplicated in the data for each separate fare.

I chose not to model particular, identifiable aircraft owned by an airline. The “fleet” relationship set can only identify a relationship between an aircraft model/variant and an airline, but it does not have the power to model particular aircraft of a given model (because the “aircraft” entity set does not have...
unique identifiers for individual aircraft). Instead, I know that a flight will use a particular model, but of all the instances of that model that exist, I do not know which particular one will be used. I felt this was acceptable as I trust the airlines to be able to manage their own fleets.

Fares are modeled as entity sets in which entities are fares defined for particular flights. Fares are linked to flights in a one-to-many relationship, reflecting that a flight can have multiple fare groups, but any given fare group can only can have seats on at most one flight. The groups of seats available for a fare group on a flight is represented by two attributes on the relationship set: how many rows there are and how many seats in each row. This allows for some flexibility in defining fares. Airlines sell fares for a price, which is an attribute on the “sells” relationship set.

The actual, physical seats on planes are stored in a separate entity set. Each seat on each unique aircraft has a record in an instance of this entity set (e.g. seat 1A on a Boeing 777 and seat 1A on a Lockheed Martin 1-1011-L are separate records). The “inGroup” relationship set connects particular seats to fare code groups.

Customers buy an aggregated entity consisting of a fare code and (possibly) a seat. Every fare has to be represented in the “inGroup” relationship set, but fares do not have to have a seat (there can be a NULL value for seat). A customer can buy a fare without a seat and let the airline assign them a seat, or they can give a seat preference along with the fare purchase.

Credit cards are stored in a separate entity set so that customers can use more than one without redundancy. Travelers associated with an account are represented as a weak entity set with the customer as identifying owner: they cease to exist when the customer account is deleted. The “fflyer” relationship set keeps track of frequent flyer miles customers may have with any of the airlines.

Different aircraft offer different amenities (such as food service, AC or DC power, or infant care facilities). These are stored as entities in an entity set and related to aircraft with a relationship set.

Different aircraft have different cabins (Economy, Business, First), represented with a total-participation many-to-many relationship set. Each fare group goes with exactly one cabin.

Customer preferences are not stored at all. I assume these are only used at the application level to help define queries.