Josh Atchley
CS 340
6/6/2017

Final Project

Outline

This database is a representation of a very rudimentary airport landing fee billing system. The details of the entities, their attributes, and their relationships will follow in the coming sections, but here is a summary of the miniworld that this database is attempting to represent.

The basic pieces of the system are Airports, Aircraft, and Operators. These represent the locations aircraft fly into, the aircraft themselves, and who owns the aircraft. Without this information, it is impossible to charge a landing fee.

There is also ancillary information to allow for more specificity in charging landing fees. For example, Aircraft have a model code, represented by the AircraftModel. This stores basic information about a particular model, such as its maximum landing and takeoff weights. This is important to know when charging landing fees, because most airports that charge a landing fee do so at a rate that is determined by the weight of the aircraft that landed there.

Finally, there are the fees themselves, which are added based on FeeRules for an Airport. The rules for an airport are represented in the AirportFeeRule table. An Airport can have multiple fees it charges based on different conditions, or charge different prices based on different aircraft weights. These fees are collected and added to create Invoice records, one Invoice per Operator and Airport for a given period. An Operator can receive multiple Invoices from multiple Airports over different days, if they operated their Aircraft at multiple Airports that charge a landing fee.

Database Outline

Here is a more detailed description of the database entities, their attributes, and their relationships, grouped by table name.

Airport
Represents airports where aircraft can operate at, to determine which billing fees to apply.

- Id – int not null primary key auto_increment
- AirportCode – varchar(255) unique not null
    - The ICAO (International Civil Aviation Organization) code representing the airport. In the contiguous United States, these codes all begin with the letter ‘K’ (e.g. KORD for Chicago O’Hare)
- FullName – varchar(255) unique not null
    - The full name for the airport (e.g. Truckee-Tahoe Airport)

Operator
Represents aircraft operators, so invoices can be sent to the right person.

- Id – int not null primary key auto_increment
- FirstName – varchar(255) not null
    - The operator's first name
- LastName – varchar(255) not null
    - The operator's last name
- UNIQUE KEY FullName (FirstName, LastName)
    - Unique combo key of the first and last name to prevent duplicates

AircraftModel
Represents the model of an aircraft (e.g. an Airbus 320) to determine its weight and other billable attributes.

- Id – int not null primary key auto_increment
- ModelCode – varchar(10) unique not null
    - The ICAO model code uniquely representing the aircraft (e.g. A320 for the Airbus 320)
- LandingWeight – int not null
    - The maximum possible weight in pounds of the aircraft model on landing. This is provided by the manufacturer and the FAA
- TakeoffWeight – int not null
    - The maximum possible weight in pounds of the aircraft model on takeoff. This is provided by the manufacturer and the FAA

Aircraft
Represents the individual aircraft that are being flown.

- Id – int not null primary key auto_increment
- TailNumber – varchar(10) unique not null
    - The unique registration number for the aircraft. Think of it as like a license plate number on a car.
- ModelId – int foreign key references Id of AircraftModel
    - A foreign key that ties back to an AircraftModel record. All aircraft have an aircraft model.
- OperatorId – int foreign key references Id of Operator
    - A foreign key that ties back to an Operator record. All aircraft have an owner, represented in this case by the Operator record.

Invoice
Represents the billable invoices for an operator at an airport.

- Id – int not null primary key auto_increment
- InvoiceDate – date not null
    - The date the invoice was created for billing fees.
- Amount – numeric(15,2) not null
    - The amount, in dollars, of the invoice.
- OperatorId – int foreign key references Id of Operator
    - A foreign key that ties back to an Operator record. An invoice must be tied to an Operator so that it can be paid.
- AirportId – int foreign key references Id of Airport
    - A foreign key that ties back to an Airport record. An Invoice must be associated with an Airport so that the fees are paid to the right place.

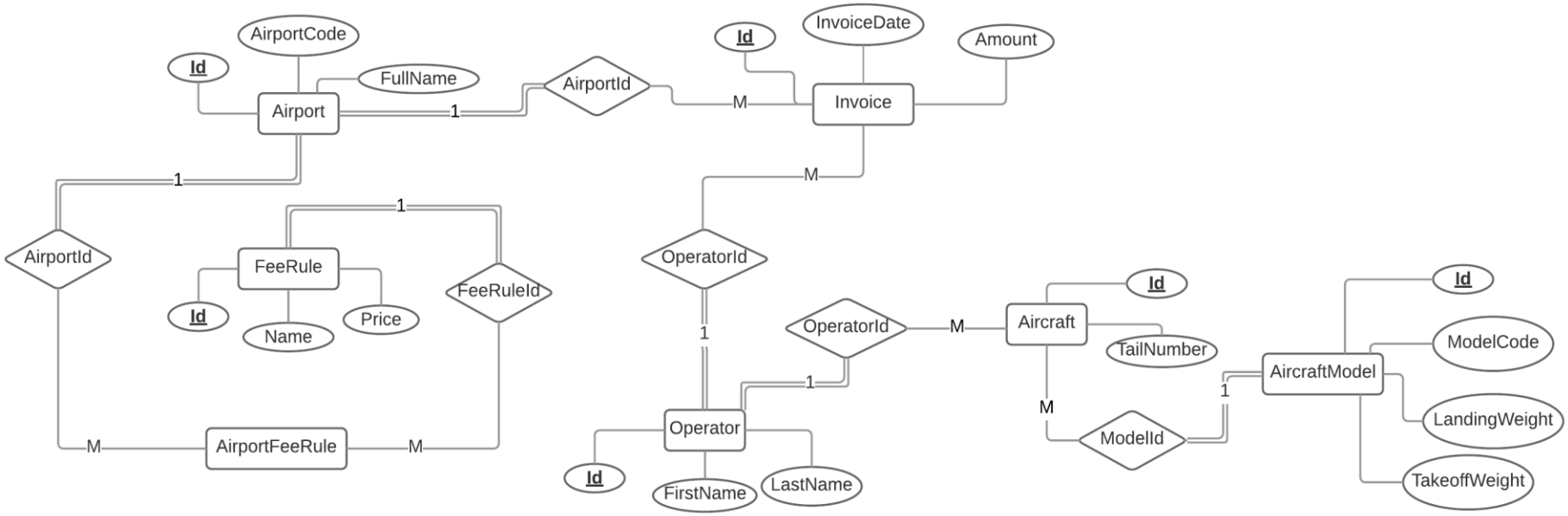
FeeRule
Represents the available billable fee rules of the system.

- Id – int not null primary key auto_increment
- Name – varchar(255) unique not null
    - The name of the fee rule, which is usually a short description of the rule (e.g. LandingWeightOver10000lbs).
- Price – numeric(15,2) not null
    - The amount in dollars that this fee rule is worth. For example, if an aircraft landed at an airport that used the fee rule with name LandingWeightOver10000lbs and it weighed more than 10,000 lbs, it would be charged the price for that activity.

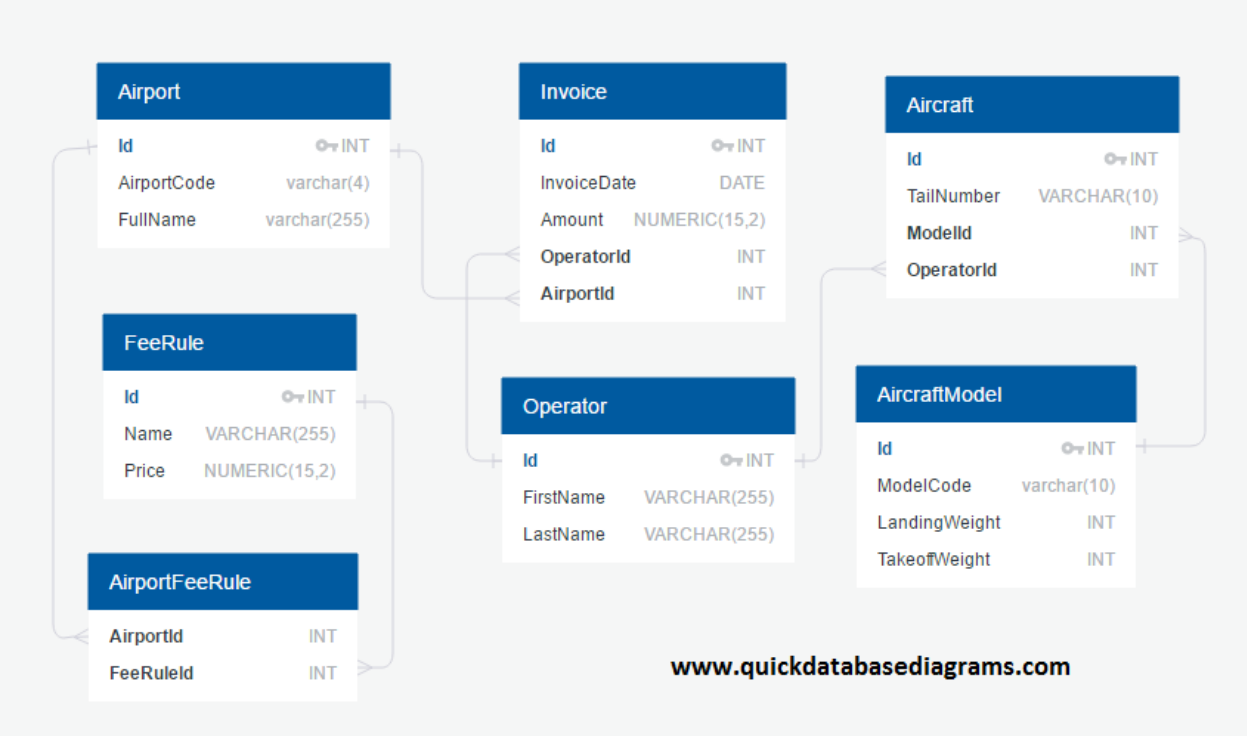
AirportFeeRule
A relationship table that maps airports to fee rules. An airport can have many fee rules, and fee rules can be used at many airports, but each row of this table represents a single pairing.

- AirportId – int foreign key references Id of Airport
    - A foreign key that ties back to an Airport record. Every relationship of this type must have one Airport.
- FeeRuleId – int foreign key references Id of FeeRule
    - A foreign key that ties back to a FeeRule record. Every relationship of this type must have one FeeRule.

Entity-Relationship Diagram



Schema



Data Definition Queries

```
-- CREATE TABLES
DROP TABLE IF EXISTS `Invoice`;
DROP TABLE IF EXISTS `Aircraft`;
DROP TABLE IF EXISTS `AircraftModel`;
DROP TABLE IF EXISTS `Operator`;
DROP TABLE IF EXISTS `AirportFeeRule`;
DROP TABLE IF EXISTS `FeeRule`;
DROP TABLE IF EXISTS `Airport`;

CREATE TABLE Airport (
  Id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
  AirportCode VARCHAR(4) UNIQUE NOT NULL,
  FullName VARCHAR(255) UNIQUE NOT NULL
);

CREATE TABLE Operator (
  Id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
  FirstName VARCHAR(255) NOT NULL,
  LastName VARCHAR(255) NOT NULL,
  UNIQUE KEY FullName (FirstName, LastName)
```

```
);

CREATE TABLE AircraftModel (
  Id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
  ModelCode VARCHAR(10) UNIQUE NOT NULL,
  LandingWeight INT NOT NULL,
  TakeoffWeight INT NOT NULL
);

CREATE TABLE Aircraft (
  Id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
  TailNumber VARCHAR(10) UNIQUE NOT NULL,
  ModelId INT,
  OperatorId INT,
  CONSTRAINT `fk_aircraft_aircraftmodel_id` FOREIGN KEY (ModelId) REFERENCES AircraftModel (Id),
  CONSTRAINT `fk_aircraft_operator_id` FOREIGN KEY (OperatorId) REFERENCES Operator (Id)
);

CREATE TABLE Invoice (
  Id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
  InvoiceDate DATE NOT NULL,
  Amount NUMERIC(15,2) NOT NULL,
  OperatorId INT,
  AirportId INT,
  CONSTRAINT `fk_invoice_operator_id` FOREIGN KEY (OperatorId) REFERENCES Operator (Id),
  CONSTRAINT `fk_invoice_airport_id` FOREIGN KEY (AirportId) REFERENCES Airport (Id),
  UNIQUE KEY OperatorAirportDate (OperatorId, AirportId, InvoiceDate)
);

CREATE TABLE FeeRule (
  Id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
  Name VARCHAR(255) UNIQUE NOT NULL,
  Price NUMERIC(15,2) NOT NULL
);

CREATE TABLE AirportFeeRule (
  AirportId INT,
  FeeRuleId INT,
  CONSTRAINT `fk_airportfeerule_airport_id` FOREIGN KEY (AirportId) references Airport (Id),
  CONSTRAINT `fk_airportfeerule_feerule_id` FOREIGN KEY (FeeRuleId) REFERENCES FeeRule (Id)
);
```

Data Manipulation Queries

Create New Aircraft

```
INSERT INTO
Aircraft
(TailNumber, ModelId, OperatorId)
VALUES
([TailNumberInput], [SelectedModelId], [SelectedOperatorId])
```

Create New Aircraft Model

```
INSERT INTO
AircraftModel
(ModelCode, LandingWeight, TakeoffWeight)
VALUES
([ModelCodeInput], [LandingWeightInput], [TakeoffWeightInput])
```

Create New Airport

```
INSERT INTO
Airport
(AirportCode, FullName)
VALUES
([AirportCodeInput], [FullNameInput])
```

Create New Invoice

```
INSERT INTO
Invoice
(InvoiceDate, Amount, OperatorId, AirportId)
VALUES
([InvoiceDateInput], [AmountInput], [SelectedOperatorId], [SelectedAirportId])
```

Create New Operator

```
INSERT INTO
Operator
(FirstName, LastName)
VALUES
([FirstNameInput], [LastNameInput])
```

Create New Fee Rule

```
INSERT INTO
FeeRule
(Name, Price)
VALUES
([NameInput], [PriceInput])
```

Create New Airport Fee Rule

```
INSERT INTO
AirportFeeRule
(AirportId, FeeRuleId)
VALUES
([SelectedAirportId], [SelectedFeeRuleId])
```

Update Aircraft Model Weights

```
UPDATE AircraftModel
SET
LandingWeight = [LandingWeightInput],
TakeoffWeight = [TakeoffWeightInput]
WHERE
Id = [SelectedAircraftModelId]
```

Delete Airport Fee Rule

```
DELETE
FROM
AirportFeeRule
WHERE AirportId = [SelectedAirportId]
AND FeeRuleId = [SelectedFeeRuleId]
```

Filter Airport Fee Rule By Airport

```
SELECT
Airport.AirportCode, FeeRule.Name
FROM
AirportFeeRule
INNER JOIN Airport
on AirportFeeRule.AirportId = Airport.Id
INNER JOIN FeeRule
on AirportFeeRule.FeeRuleId = FeeRule.Id
WHERE
AirportFeeRule.AirportId = [SelectedAirportId]
ORDER BY Airport.AirportCode, FeeRule.Name ASC
```