RELATIONSHIP SCHEMA

Airline (airlineID, revenue)

Location (locationID)

Airport (airportID, locationID[FK1], airport_name, city, state, country)
FK1: locationD → Location(locationID)

Airplane (tail_num, airlineID[FK4], seat_cap, speed, locationID[FK5], maintained, model, variant, type)
FK2: airlineID → Airline(airlineID)
FK3: locationID → Location(locationID)

Route (routeID, legs)

Unused_legs (legID, distance, departure_airport[FK2], arrival_airport[FK3])
FK4: departure_airport → Airport(airportID)
FK5: arrival_airport → Airport(airportID)

Flight (flightID, (tail_num, support_airlineID)[FK7], progress, airplane_status, next_time, cost, routeID[FK8])
FK7: (tail_num, support_airlineID) → Airplane(tail_num, airlineID)
FK8: routeID → Route(routeID)

Pilot (personID, first_name, last_name, taxID, experience, licenses, flightID[FK9], locationD[FK10])
FK9: locactionID → Location(locID)
FK10: flightID → Flight(flightID)

Passenger (personID, first_name, last_name, funds, miles, vacation, locationID[FK11])
FK11: locactionID → Location(locactionID)

UPDATE/DELETE BEHAVIORS

FK1: locationID → Location(locationID)

On update cascade: When the locationID in Location is updated, it should cascade to Airport to ensure consistency. This is important because airports are assigned to specific locations.

On delete restrict: deleting a location shouldn't be allowed when reference by an airport because an airport cannot exist without a location.

FK2: airlineID → Airline(airlineID)

On update cascade: If the airlineID in Airline is updated, it should cascade to Airplane to ensure consistency. This is important because airplanes are owned by specific airlines.

On delete restrict: Airplanes cannot exist without a valid airline so having a restriction ensures this does not happen.

FK3: locationID → Location(locationID)

On update cascade: When the locationID in Location is updated, it should cascade to Airplane to ensure consistency. This is important because airplanes are assigned to specific locations.

On delete restrict: Airplanes cannot exist without a valid location so having this restriction ensures this does not happen.

FK4: departure_airport → Airport(airportID)

On update cascade: If the airportID is updated, it should cascade to Leg to ensure consistency because legs are defined by their departure and arrival airports.

On delete restrict: A leg should not be left without a valid departure airport, and restricting the deletion for departure_airport ensures this.

FK5: arrival_airport → Airport(airportID)

On update cascade: If the airport is updated, it should cascade to Leg to ensure consistency because legs are defined by their departure and arrival airports.

On delete restrict: A leg cannot be left without a valid arrival airport and restricting the deletion for arrival_airport ensures this.

FK7: (tail_num, support_airlineID) → Airplane(tail_num, airlineID)

On update cascade: If the tail_num or airlineID in Airplane is updated, it should cascade to Flight to ensure consistency. This is important because flights are supported by airplanes.

On delete cascade: When an airplane is deleted, we want all the corresponding rows in Flight to also be deleted so that we ensure no flight references a nonexistent airplane.

FK8: routeID → Route(routeID)

On update cascade: When the routeID in Route is updated, it should cascade to Flight to ensure consistency. This ensures that flights are assigned to specific routes.

On delete restrict: Flights cannot exist without a valid route, so restricting the delete for routeID prevents that.

FK9: locationID → Location(locationID)

On update cascade: When the locationID in Location is updated, it should cascade to Pilot to ensure consistency. This is important because pilots are assigned to specific locations.

On delete restrict: Pilots cannot exist without a valid location, so having this restriction prevents that.

FK10: flightID → Flight(flightID)

On update cascade: If a flightID in Flight is updated, it should cascade to Pilot to ensure consistency. This is important because pilots are assigned to specific flights.

On delete restrict: Pilots cannot be assigned to flights that do not exist and restricting this ensures this does not happen.

FK11: locationID → Location(locationID)

On update cascade: If a flightID in Flight is updated, should cascade to Passenger to ensure consistency. This is important because passengers are assigned to specific locations.

On delete restrict: Passengers cannot exist without a valid location, and this restriction ensures this does not happen.