Part B Design Normalized Database

Author: Xujia Qin

Date: May 30th, 2025

1. For the relation represented by all of the columns in the CSV file, define all functional dependencies and list them.

- flight.number + date → airline, aircraft, dep.airport
    - The flight details (airline, aircraft type, and departure airport) depend on the combination of flight number and date.

- iid → date, flight.number, incident.type, severity, delay.mins, num.injuries, reported.by
    - Each incident ID uniquely identifies all other incident-related attributes.

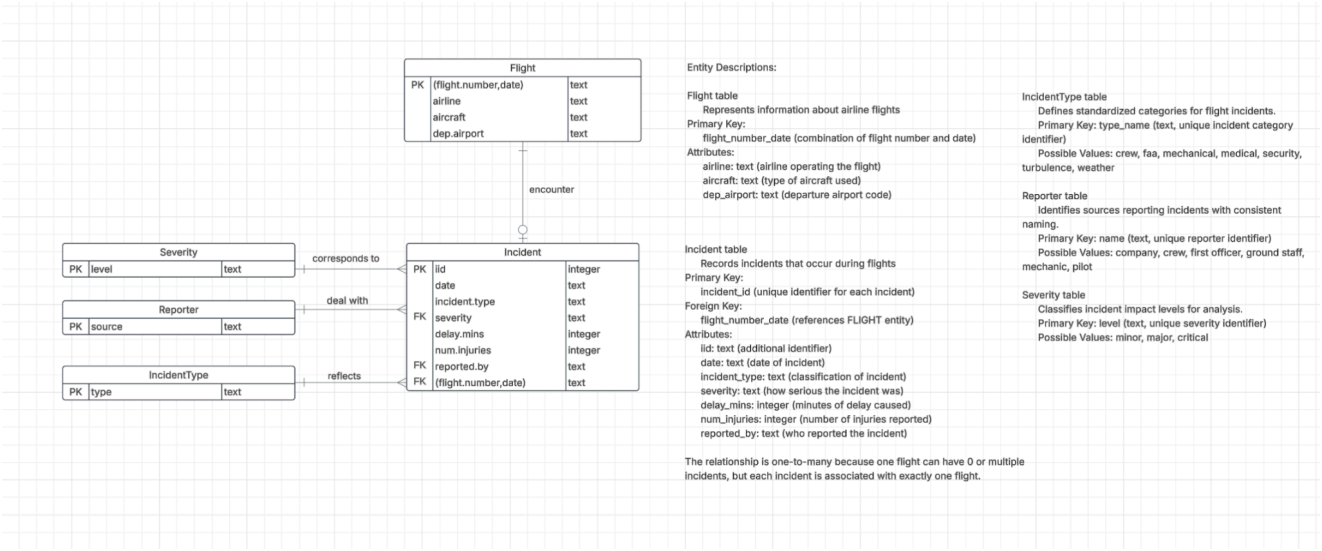
2. Using the functional dependencies and the rules of normalization, decompose the relational from the CSV into several relations that all satisfy 3NF; give the relations reasonable names.
    - No partial dependencies:
        - For the flight table, all non-prime attributes depend on the full composite PK, not just flight_number or date alone. Example: If flight_number → airline (partial dependency), it would violate 2NF. But here, the airline depends on both flight_number and date (the same flight number can be used by different airlines on different dates).

        - For the incident table, since the PK is atomic (iid), partial dependencies cannot exist.

    - No transitive dependencies:
        - For the flight table, all non-prime attributes depend directly on the PK, not on each other. Example: If airline → aircraft (e.g., "Delta only uses Boeing 737"), this would be a transitive dependency (violating 3NF). But in reality, an airline can use multiple aircraft types, so no such dependency exists.

        - For the incident table, all attributes depend only on iid, not on each other. Example: If incident_type → severity (e.g., " Mechanical only causes minor"), this would be a transitive dependency (violating 3NF). But mechanical incidents can cause multiple severity types, so no such dependency exists.

3. For the relations resulting from the normalization, create an ERD in the IE (Crow's Feet) notation. Add all attributes, attribute name, primary and foreign keys, data types, and entity descriptions. You may use any modeling tool of your choosing, e.g., LucidChart or Mermaid. Embed the ERD into your document. Save the document containing your database design as a PDF.



Entity Descriptions:

Flight table
Represents information about airline flights
Primary Key:
flight_number_date (combination of flight number and date)
Attributes:
airline: text (airline operating the flight)
aircraft: text (type of aircraft used)
dep_airport: text (departure airport code)

Incident table
Records incidents that occur during flights
Primary Key:
incident_id (unique identifier for each incident)
Foreign Key:
flight_number_date (references FLIGHT entity)
Attributes:
iid: text (additional identifier)
date: text (date of incident)
incident_type: text (classification of incident)
severity: text (how serious the incident was)
delay_mins: integer (minutes of delay caused)
num_injuries: integer (number of injuries reported)
reported_by: text (who reported the incident)

The relationship is one-to-many because one flight can have 0 or multiple incidents, but each incident is associated with exactly one flight.

IncidentType table
Defines standardized categories for flight incidents.
Primary Key: type_name (text, unique incident category identifier)
Possible Values: crew, faa, mechanical, medical, security, turbulence, weather

Reporter table
Identifies sources reporting incidents with consistent naming.
Primary Key: name (text, unique reporter identifier)
Possible Values: company, crew, first officer, ground staff, mechanic, pilot

Severity table
Classifies incident impact levels for analysis.
Primary Key: level (text, unique severity identifier)
Possible Values: minor, major, critical