Rest API Design

Oliver Holder, Nik Dijkema, Tai Ting (Group 18)

February 2019

1 Introduction

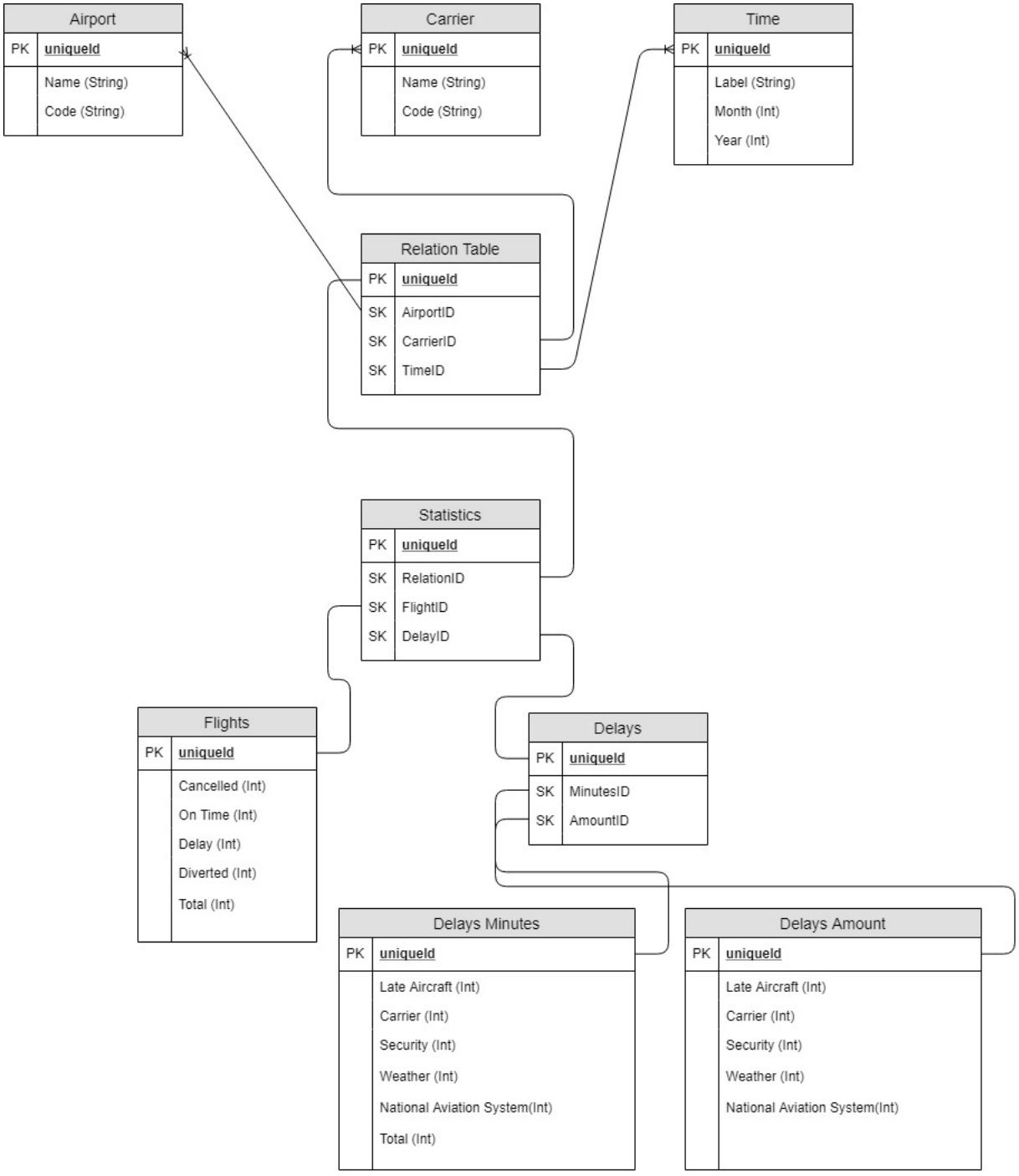
1.1 Framework choice

(Reasons for choosing python and flask etc.. here). – Python (Bootstrap, Flask). Python flask allows for character escaping. – SQLAlchemy for storage.

1.2 Data storage options

In this project, the initial data we receive is in JSON and CSV format. Either of these formats can be directly parsed and loaded into memory as lists in python. This way of processing data works well for read only data, however, there will be PUSH requests in the API, meaning the data will be subject to change. This causes problems for data loaded in memory, saving the data as text for every PUSH request is very inefficient. Therefore SQL will be used as the main storage method. Because the API will be made in python, the SQL handling will be performed by SQLAlchamy.

2 Data model


Airport
PK uniqueId
Name (String)
Code (String)
Carrier
PK uniqueId
Name (String)
Code (String)
Time
PK uniqueId
Label (String)
Month (Int)
Year (Int)
Relation Table
PK uniqueId
SK AirportID
SK CarrierID
SK TimeID
Statistics
PK uniqueId
SK RelationID
SK FlightID
SK DelayID
Flights
PK uniqueId
Cancelled (Int)
On Time (Int)
Delay (Int)
Diverted (Int)
Total (Int)
Delays
PK uniqueId
SK MinutesID
SK AmountID
Delays Minutes
PK uniqueId
Late Aircraft (Int)
Carrier (Int)
Security (Int)
Weather (Int)
National Aviation System(Int)
Total (Int)
Delays Amount
PK uniqueId
Late Aircraft (Int)
Carrier (Int)
Security (Int)
Weather (Int)
National Aviation System(Int)

The objects and their relations must be defined before an API can be designed. Visible Objects (with URI):

–Airport
–Carrier
    –Statistics
        –Delays
            –Amount
            –Minutes
        –Flights

2.1 URIs

Airport: url/airports/< *code* >
Carrier: url/carriers/< *code* >
Statistics: url/carriers/< *code* >/statistics
Delays(amount): url/carriers/< *code* >/delays/count
Delays(minutes): url/carriers/< *code* >/delays/minutes
Flights: url/carriers/< *code* >/flights

3 End Points

3.1 Airports

3.1.1 /airports (1)

Type: GET
Return: List of airport URIs

3.1.2 /airports/< *code* > (3)

Type: GET
return: Airport name + List of carrier URIs related to that airport

3.2 Carriers

3.2.1 /carriers (2)

Type: GET
Return: List of carrier URIs

3.2.2 /carriers/< *code* >

Type: GET
Return: Statistics URI + Carrier Name (string).

3.3 Statistics

3.3.1 /carriers/< *code* >/statistics (4)

TYPE: GET
Return: Statistics URI for given carrier returned as delay and flight resources.
(- query options: month, airport)

3.3.2 /carriers/< *code* >/statistics/flights (5)

TYPE: GET
Return: Json Flights information list.
(- query options: month, airport)

3.3.3 /carriers/< *code* >/statistics/delays/minutes (6)

TYPE: GET
Return: Json (minute) delays information list.
(- query options: $delay_type$, month, airport)

3.3.4 /carriers/< *code* >/statistics/delays/minutes/averages

(- query options: *delay_type*, airport1, airport2, carrier)

4 URL GET METHOD LIST

- /airports
- /airports/< *code* >
- /carriers
- /carriers/< *code* >
- /carriers/< *code* >/statistics
- /carriers/< *code* >/statistics/flights
- /carriers/< *code* >/statistics/delays/minutes
- /carriers/< *code* >/statistics/delays/minutes/verages

5 Verification and security

Security for viewing data is unnecessary. (what about push?)
Using HTTPS with a self generated certificate.