“Flights Booking System” Relational Schema and Normalization

Course: Data Management I, Academic Year: 2023 / 2024

Supervised by: Pr. Karima ECHIHABI

*Safae HAJJOUT, Kawtar LABZAE, Othmane AZOUBI, Mounia BADDOU*

*Mohammed VI Polytechnic University, College of Computing, School of Computer Science*

Relations and Schemas *(Pre-Normalization)*

## **Relations:**

User, Passenger, Reservation, Ticket, FidelityCard, Flight, Airplane,

Checks.

## **Schemas:**

**User (**uemail: *String*, ufirstName: *String*, ulastName: *String*, ubirthDate: *Date*, passwordHash: *String***)**

**Passenger (** passportID: *String*, cin: *String*, pbirthDate: *Date*, phoneNumber: *String*, pfirstName: *String*, plastName: *String*, fctype: *String*, fcid: *String***)**

*FK fctype FROM FidelityCard*

**Reservation (**rid: *String*, rDate: *DateTime*, confirmDate: *DateTime*, uemail: *String***)**

*FK uemail FROM User*

**Ticket (**tid: *String*, seatNumber: *Integer*, class: *String*, price: *Float*, passportID: *String*, rid: *String*, fid: *String,* fctype: *String***)**

*FK passportID FROM Passenger, FK rid FROM Reservation, FK fid FROM Flight, FK fctype FROM FidelityCard*

**FidelityCard (**fctype: *String*, reduction: *Integer***)**

**Flight (** fid: *String*, arrivalTime: *DateTime*, departureTime: *DateTime*, destination: *String*, departure: *String*, registrationNumber: *Integer***)**

*FK registrationNumber FROM Airplane*

**Airplane (**registrationNumber: *Integer*, economySeats: *Integer*, premiumEconomySeats: *Integer*, businessClassSeats : *Integer*, firstClassSeats: *Integer*, maxweight: *Integer*, model: *String*, airline: *String***)**

**Checks (**uemail: *String*, fid: *String***)**

*FK uemail FROM User, FK fid FROM Flight*

Functional Dependencies

When working on our functional dependencies, it was clear that our set of functional dependencies coincides with the minimal cover itself, therefore we made the following table to the reader’s convenience, presenting simultaneously our set of FD’s and the minimal cover

## **Minimal cover:**

| **Reservation** | **rid** —> rDate, confirmDate, uemail |
| --- | --- |
| **User** | **uemail** —> ufirstName, ulastName, ubirthDate, passwordHash |
| **Passenger** | **passportID** —> pbirthDate, passportID, phoneNumber, pfirstName, plastName, fctype, fcid |
| **phoneNumber** —> pbirthDate, passportID, phoneNumber, pfirstName, plastName, fctype, fcid |
| **cin** —> pbirthDate, passportID, phoneNumber, pfirstName, plastName, fctype, fcid |
| **fcid** —> fctype |
| **Ticket** | **tid** —> seatNumber, class, price, passportID, rid, fid, fctype |
| **(fid, passportID)** —> seatNumber, class, price, passportID, rid, fid, fctype |
| **(fid, seatNumber)** —> seatNumber, class, price, passportID, rid, fid, fctype |
| **FidelityCard** | **fctype** —> reduction |
| **Flight** | **fid** —> arrivalTime, departureTime, destination, departure, registrationNumber |
| **Airplane** | **registrationNumber** —> economySeats, premiumEconomySeats, businessClassSeats, firstClassSeats, model, maxWeight, airline |
| **model** —> economySeats, premiumEconomySeats, businessClassSeats, firstClassSeats, maxweight |
| **Checks** | **(email, fid)** —> email, fid |

## 

## **Normalization:**

**User:**

From the dependencies above, all attributes are deduced from “email” the primary key of the User schema. This verifies the conditions to BCNF.

**Passenger:**

We may deduce from the calculated FD’s of the Passenger entity three candidate keys**:** “phoneNumber”, “cin” and “passportID”. However, this functional dependency breaks the BCNF rule:

**fcid** —> type

Thus the need for a table division, resulting in two new schemas:

* **Passenger (**passportID: *String*, cin: String, pbirthDate: *Date*, phoneNumber: *String*, pfirstName: *String*, plastName: *String*, fcid: *String***)**
* **PassengerCard (**fcid: *String*, fctype: *String***)**

Both tables are now in BCNF.

**Reservation:**

As we can see from the Functional Dependencies above, all of the attributes are deduced from and only from “rid”, which is the primary key. Thus the Reservation schema in BCNF.

**Ticket:**

Based on the Functional Dependencies above, there are three possible candidate keys for the Ticket entity: “tid”, “fid, cin” and “fid, seatNumber”. thus it is in BCNF. The primary key will remain “tid” as it is the simplest one.

**FidelityCard:**

From the functional dependencies, we can easily conclude that FidelityCard is in BCNF. Its primary key is “fctype”.

**Flight:**

Since the only functional dependency that represents the schema is the one where the primary key “flightID” gives all of the other attributes, we conclude that Flight is in BCNF.

**Airplane:**

From the calculated FD’s, ”registrationNumber” is a primary key. But there is a dependency that breaks the BCNF rule:

**model** → (economySeats, premiumEconomySeats, businessClassSeats, firstClassSeats, maxWeight)

Which leads to the creation of two tables that abide by BCNF:

* **AirplaneModel (**model: *String*, economySeats: *Integer*, premiumEconomySeats: *Integer*, businessClassSeats : *Integer*, firstClassSeats: *Integer*, maxWeight: *Integer***)**
* **Airplane (**registrationNumber: *Integer*, model: *String*, airline: *String***)**

Both tables are now in BCNF.

**Checks:**

Since “Checks” only contains foreign keys to the schemas “User” and “Flight”, the primary key of this table is the combination (email, fid), thus verifying the conditions of BCNF.

Relations and Schemas *(Post-Normalization)*

## **Relations:**

User, Passenger, PassengerCard, Reservation, Ticket, FidelityCard, Flight, Airplane, AirplaneModel, Checks.

## **Schemas:**

**User (**uemail: *String*, ufirstName: *String*, ulastName: *String*, ubirthDate: *Date*, passwordHash: *String***)**

**Passenger (**passportID: *String*, cin: *String*, pbirthDate: *Date*,phoneNumber: *String*, pfirstName: *String*, plastName: *String*, fcid: *String***)**

**PassengerCard (**fcid: *String*, fctype: *String***)**

*FK fcid FROM Passenger, FK fctype FROM FidelityCard*

**Reservation (**rid: *String*, rDate: *DateTime*, confirmDate: *DateTime*, uemail: *String***)**

*FK uemail FROM User*

**Ticket (**tid: *String*, seatNumber: *Integer*, class: *String*, price: *Float*, passportID: *String*, rid: *String*, fid: *String,* fctype: *String***)**

*FK passportID FROM Passenger, FK rid FROM Reservation, FK fid FROM Flight, FK fctype FROM FidelityCard*

**FidelityCard (**fctype: *String*, reduction: *Integer***)**

**Flight (** fid: *String*, arrivalTime: *DateTime*, departureTime: *DateTime*, destination: *String*, departure: *String*, registrationNumber: *Integer***)**

*FK registrationNumber FROM Airplane*

**AirplaneModel (**model: *String*, economySeats: *Integer*, premiumEconomySeats: *Integer*, businessClassSeats : *Integer*, firstClassSeats: *Integer*, maxWeight: *Integer***)**

**Airplane (**registrationNumber: *Integer*, model: *String*, airline: *String***)**

*FK model FROM AirplaneModel*

**Checks (**uemail: *String*, fid: *String***)**

*FK uemail FROM User, FK fid FROM Flight*