Triggers

Grégoire Détrez

February 24, 2016

Exercice 1

Booking flights

Domain Description

We extend the shema from last week with the following relations to handle bookings:

```
AvailableFlights(_flight_, _date_, numberOfFreeSeats, price)
  flight -> Flights.code
Bookings(_reference_, flight, date, passenger, price)
  (flight, date) -> AvailableFlights.(code, date)
```

Where `Bookings.price` is the price that was paid for a particular ticket and `AvailableFlights.price` is the current price of the flight, which is different.

Question 1

Create a view that lists booking references, passengers, flight codes, dates, and departure and destination cities.

Question 2

Create a trigger on the view defined in question 1. This trigger takes care of booking a new passenger to a flight. It is fired by an insertion of a passenger and a flight code to the view, for instance, "book Annie Adams for AF666". Its effect should be the following:

1. if the number of free seats on AF666 is positive, decrement it by one; the booking is successful
2. if there are no free seats, the booking fails
3. if the booking is successful, add Annie Adams and AF666 to Bookings, with the price given in AvailableFlights when booking her; also add a booking reference which is the maximum of the previous references (for all flights) plus one
4. if the booking is successful, increment the price by 50 SEK for the next passenger (thus the fuller the flight, the more you pay)

Refactoring

The airline decides to upgrade its database to keep track of its fleet. This means adding a table to list the available planes and updating the `AvailableFlights` in the following way:

```
Planes(_regnr_, capacity)
AvailableFlights(_flight_, _date_, numberOfFreeSeats, price, plane)
  (flight, date) -> Flights.(code, date)
    plane -> Planes.regnr
```

Question 3

Your job is to create a trigger that automatically update `numberOfSeats` when the company changes plane for a flight.

Exercice 2

At the restaurant

Domain Description

In this exercise, we are creating the following database for a restaurant:

```
Tables(_number_, seats)
Bookings(_name_, _time_, nbpeople, table)
  table -> Tables.number
```

For the sake of simplicity, we assume that the database only needs to hold bookings for the current day and that bookings are always done on the hour (i.e. `time` is an integer between 0 and 23). Finally, tables are always booked for two hours. This means that if table 1 is booked at 19.00, it can't be booked at 20.00 but it can be booked again at 21.00.

Question 1

Write a view that lists the times at which tables are blocked by a booking. In the example above, where table 1 is booked at 19.00, the view should contain the following rows:

table	time
1	18
1	19
1	20

Question 2

Write a trigger on the table `Bookings` that automatically assign a table to new rows if none is specified. The assigned table should respect the following rules:

- it should be free for the duration of the booking.
- it should be big enough for the number of people in the party
- it should be the smallest possible table to accommodate this number of people

Exercice 3

Wiki

Domain Description

In this exercise, we are creating a database for a wiki. A wiki is a website that *allows collaborative modification of its content and structure directly from the web browser* (Wikipedia).
To make collaborative edition easier, we needs to keep an history of the modifications of each page. To achieve this, we will use a simple model that simply keeps each version of each page as a separate row:

```
PageRevision(_name_, _date_, author, text)
```

Question 1

To make it easier to access the wiki, creat a view `Page(name, last_author, text)` that shows only the latest version of each page.

Question 2

Create a trigger on your newly created view so that when a user tries to update a given page, a new revision is created instead.

Sometimes, pages on the wiki needs to be completely deleted (for instance, if a page contains sensitive information or copyrighted content). In that case, we want to remove all revisions of the page from the database but we still want to remember that the page has existed but has been deleted. To this end, we add the following table to our database:

```
DeleteLog(_pagename_, _date_)
```

Question 3

Write a trigger on the `Page` view such that when a page is deleted:

- all its revisions are removed from the database
- the deletion is recorded in the `DeleteLog`.