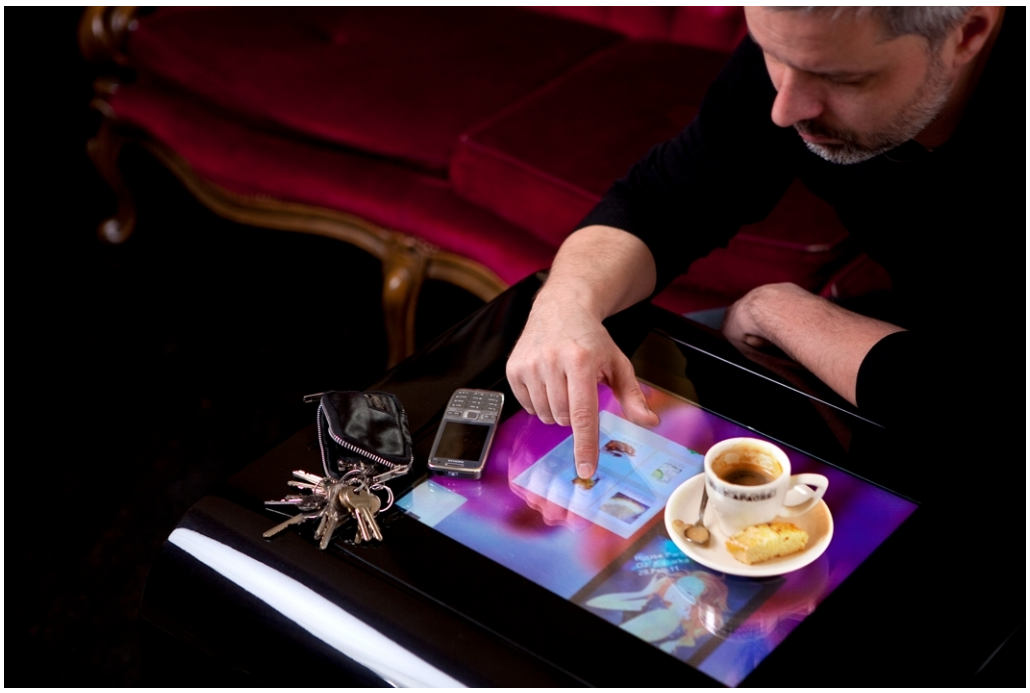
Universite Catholique de Louvain

Databases

Mission 4 : Database Programming
The Automated Cafe



Auteurs :
Jonathan Powell (61331200)
Jérôme Lemaire (69601000)

Cours :
LINGI2172

Groupe :
Query builder 20

Titulaire :
Bernard Lambeau

Assistants :
Antoine Cailliau
Quentin De Coninck

INFO21
10 mai 2016

Introduction

As part of the course LINGI2172, we realized different approach to organize the database of the Automated Café application. In this report you will find a small summarize of the work done for this project. In the main directory of the project there is a `Readme.md` file, don't forget to have a look to know how to run the different parts of the project. There is also different scenarios files that can be ran to test our models. The project was divided in three different steps and we will discuss in this report the realization done for each of these parts.

Structure of database

The structure of the database stays the same for each step. The Figure 1 below shows the different relvars.

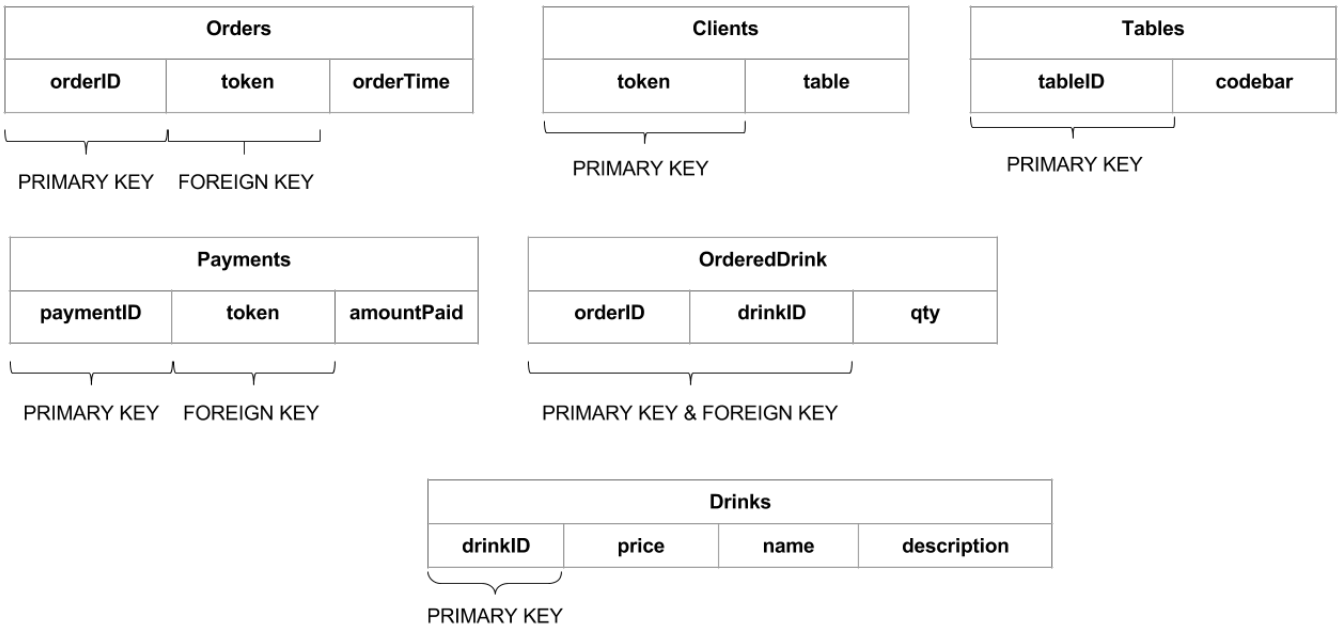


FIGURE 1 – Relations : Structure of the database

In the relation `Clients` each token must unique, this is why we use it as the primary key. Each single primary key are incremented automatically as well for the step 1 and step 2 that for the step 3. The relvar `Clients`, `Orders`, `Payments` and `OrderedDrink`, as shown on the Figure 1, also have foreign keys. We also added constraint on the column `qty`, `price` and `amountPaid` since they represent quantity and money they must be equals or greater than 0.

Step 1

The files in this step are in the directory `database`. The most interesting file to analyze is the file `procedures.sql`. The language used is `PL/pgSQL`. It contains the definition of type, storage procedure, view and trigger. description of the four major procedural functions :

AcquireTable allows to add a customer in the `Clients` table if the table is free. To check if the table is free to insert before a customer in the table `Clients`, we implemented a trigger on `Clients`. If the table is not free, we display an error message. A table is free if the number of the customer sitting in this table is equals to the number of the payment in this table.

OrderDrinks simulates the drinks order. A drink order is represented by the type `orderList`. `OrderList` represent a tuple (drinkid, quantity of drink). The `OrderDrinks` procedure insert the order in the table `Orders` and `OrderedDrink`.

IssueTicket is call when the customer will ask for the bill. We implemented the view `bill` which contains all the drinks ordered but not yet paid for all the table occupied by a client. The function `IssueTicket` returns the total amount of the orders and a list of `ordernameList` for the customer. The `ordernameList` type represent the tuple (drinkname, quantity).

PayTable is call when the customer will pay and free his table. First, we check if the token is valid and the table is occupied with the function `is_valid_token`. Note that this function is also use in the `OrderDrinks` and

`IssueTicket` procedures. Finally before updating the database, we check that the payment is greater than the amount due.

Step 2

The files in this step is in the directory `call-level-api`. We implement the call-level API with the library `Psycopg2` in `Python`. It calls the procedures implemented in the step 1.

Step 3

We use Django as higher-level approach to database connectivity, more precisely we are going through a framework named Django-Query-Builder [1]. Django-Query-Biulder is a Python library that allows us to construct and execute SQL instruction in query builder mode. This part of the project are in the directory `query-builder`. The database design and connectivity is implemented thanks to the files `models.py` and `settings.py` in the `TheAutomatedCafe` application as Django requires. The `manage.py` files tkaes care of the administrative tasks, see further information here

The file `step3.py` in the `query-builder` directory is the core implementation of the task and regroups the different functionalities asked. Given that we use the database as a pure storage, the queries in SQL are really simple and most of the work is done in Python. Note that each function raise an Error when the `PRE` condition are not respected. Moreover, there is also the function `ticketToString` that return the bill to a well-formated string.

Sources :

[1] Image, Automaton Cafe, `http://www.draqie.com/images/restaurant.jpg`

1. Django-Query-Builder version 0.10.0 : `https://pypi.python.org/pypi/django-query-builder`