Programming Project: Krusty Kookies

Johannes Jansson, F11
tfy11jja@student.lu.se

Victor Miller, F11
tfy11vmi@student.lu.se

Introduction

The fictive company Krusty Kookies has recently expanded into Sweden, and needs a computerized system for production and delivery of cookies. The system needs to handle everything: storage of ingredients, lists of recipes, tracking of pallets, placed orders and deliveries. This project is a part of the LTH course EDA216 – Database technology, and our task is to implement the parts of this system related to production. We will also create a screen that simulated production, so that the system can be tested without the presence of an actual factory.

Requirements

Even though the requirements were a bit diffuse (as intended) we are fairly confident that we have met all of them.

Outline of the System

The system is constructed in a way much similar to the system in lab 4. The database manager MySQL is used to manage the database, located on `puccini.cs.lth.se`. The creation and population of the database is performed by sourcing the file **tables.sql**, the creation is described in listing 2.

The factory interface is a web page written in PHP. The php tool PDO was used to establish a connection to the database. The PHP class **database.inc.php** handles everything related to the SQL database, and the PHP class **pallet.inc.php** is used to transmit data from the database class to the webpage. The connection is established just as in the lab. Updates and queries are made using perpared statements:

```
$stmt = $this->conn->prepare($query);
$stmt->execute($param);
```

to protect the database from SQL injections. The newly created methods for this assignment are:

- **producePallet($date, $time, $name)** simulates the production of a pallet.
- **getPallet($barcode)** returns a php pallet object with information from the corresponding pallet tuple in Pallets.
- **blockIntervall($product, $date, $startTime, $endTime)** finds all pallets meeting the criteria, blocks them and returns them.
- **generalSearch** handles all searching requested by the user.

There are also methods for getting all locations, production dates, barcodes and products, used for generating dropdown menus in the user interface. The class **database.inc.php** is quite well commented, have a look there for more details.

The pallet class only consists of variables, a constructor and getters.

The interface consists of X views. tk explain them

E/R Diagram

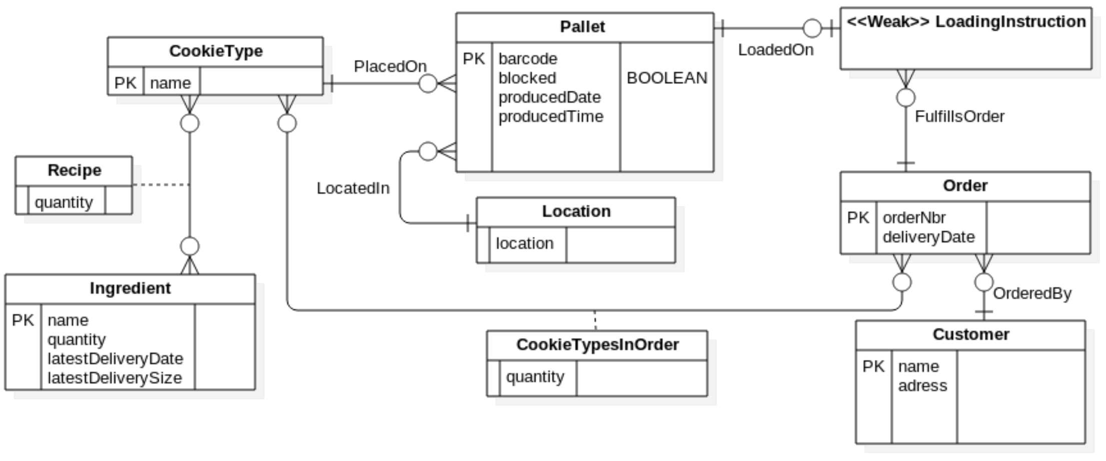

Figure 1: E/R–diagram for the system

Relations

The following relations were the basis for creating the database:

Listing 1: Relations.txt

```
CookieTypes(_name_);
(Primary key: name)

Ingredients(_name_, quantity, latestDeliveryDate,
    latestDeliverySize);
(Primary key: name)

Customers(_name_,address);
(Primary key: name)
```

```
Orders(_orderNbr_,deliveryDate,~customerName~);
(Primary key: orderNbr. Foreign key: customerName)

Locations(_location_);
(Primary key: location)

Pallets(_barcode_,~location~,blocked,producedDate,producedTime,
    ~cookieName~);
(Primary key: barcode. Foreign keys: location, cookieName)

Recipes(_~cookieName~,~ingredientName~_,quantity);
(Primary keys: cookieName, ingredientName.
    Foreign keys: cookieName, ingredientName)

CookieTypesInOrders(_~cookieName~,~orderNbr~_,quantity);
(Primary keys: cookieName, orderNbr.
    Foreign keys: cookieName, orderNbr)

LoadingInstructions(~orderNbr~,_~barcode~_);
(Primary key: barcode. Foreign keys: orderNbr, barcode)
```

SQL Statements

The following SQL statements were used to create the database:

Listing 2: tables.sql

```
-- Delete the old tables
set foreign_key_checks = 0;
drop table if exists Pallets;
drop table if exists Locations;
drop table if exists CookieTypes;
drop table if exists Recipes;
drop table if exists Ingredients;
drop table if exists CookieTypesInOrders;
drop table if exists LoadingInstructions;
drop table if exists Orders;
drop table if exists Customers;
set foreign_key_checks = 1;

-- Create the new tables
create table CookieTypes (
  name            varchar(64),
  primary key (name)
);

create table Ingredients (
  name                varchar(64),
  quantity            integer check (quantity >= 0),
  latestDeliveryDate  date,
  latestDeliverySize  integer,
  primary key (name)
);

create table Customers (
  name    varchar(128),
  address varchar(256),
  primary key (name)
);

create table Orders (
  orderNbr      integer auto_increment,
  deliveryDate  date,
  customerName  varchar(128),
  primary key (orderNbr),
  foreign key (customerName) references Customers(name)
);

create table Locations (
  location     varchar(32),
  primary key (location)
);

create table Pallets (
  barcode       integer auto_increment,
  location      varchar(32) default 'Freezer',
  blocked       boolean default 0,
  producedDate  date,
  producedTime  time,
  cookieName    varchar(64),
  primary key (barcode),
  foreign key (cookieName) references CookieTypes(name),
  foreign key (location) references Locations(location)
);

create table Recipes (
  cookieName       varchar(64),
  ingredientName  varchar(64),
  quantity        integer check (quantity >= 0),
  primary key (cookieName, ingredientName),
  foreign key (cookieName) references CookieTypes(name),
  foreign key (ingredientName) references Ingredients(name)
);

create table CookieTypesInOrders (
  cookieName  varchar(64),
  orderNbr    integer,
  quantity    integer check (quantity >= 0),
  primary key (cookieName, orderNbr),
  foreign key (cookieName) references CookieTypes(name),
  foreign key (orderNbr) references Orders(orderNbr)
);

create table LoadingInstructions (
  orderNbr  integer,
  barcode   integer,
  primary key (barcode),
  foreign key (orderNbr) references Orders(orderNbr),
  foreign key (barcode) references Pallets(barcode)
);
```

User's manual

We consider the system self–explanatory enough not to require an user manual. The system has, in fact, been tested on a med–student with great success. The only things worth pointing out are that:

1. This thing
2. This thing
3. And this thing