COMP306

Database Management Systems

Term Project

Barış Şenkal

Maya Younes

Berk Kaan Kuğuoğlu

Spring 2016

1. Introduction

“FoodOrder” is an online food ordering platform that provides its users a portal to place their orders with the restaurant registered to the system. Similar to many other examples existing on the web, such as yemeksepeti.com, FoodOrder also has a number of restaurants registered to its system, as well as customers who are looking for placing their orders. As a user friendly food ordering portal, FoodOrder provides its users a display of menus, and delivering the order with the corresponding restaurant. Furthermore, not surprisingly, customers can fill their shopping cart with multiple numbers of food before they place their order. In this report, it is briefly summarized how the system, system capabilities, and functionalities are determined. In addition, our database design, back end and front end solutions are broadly discussed.

2. Design

2.1 ER Model

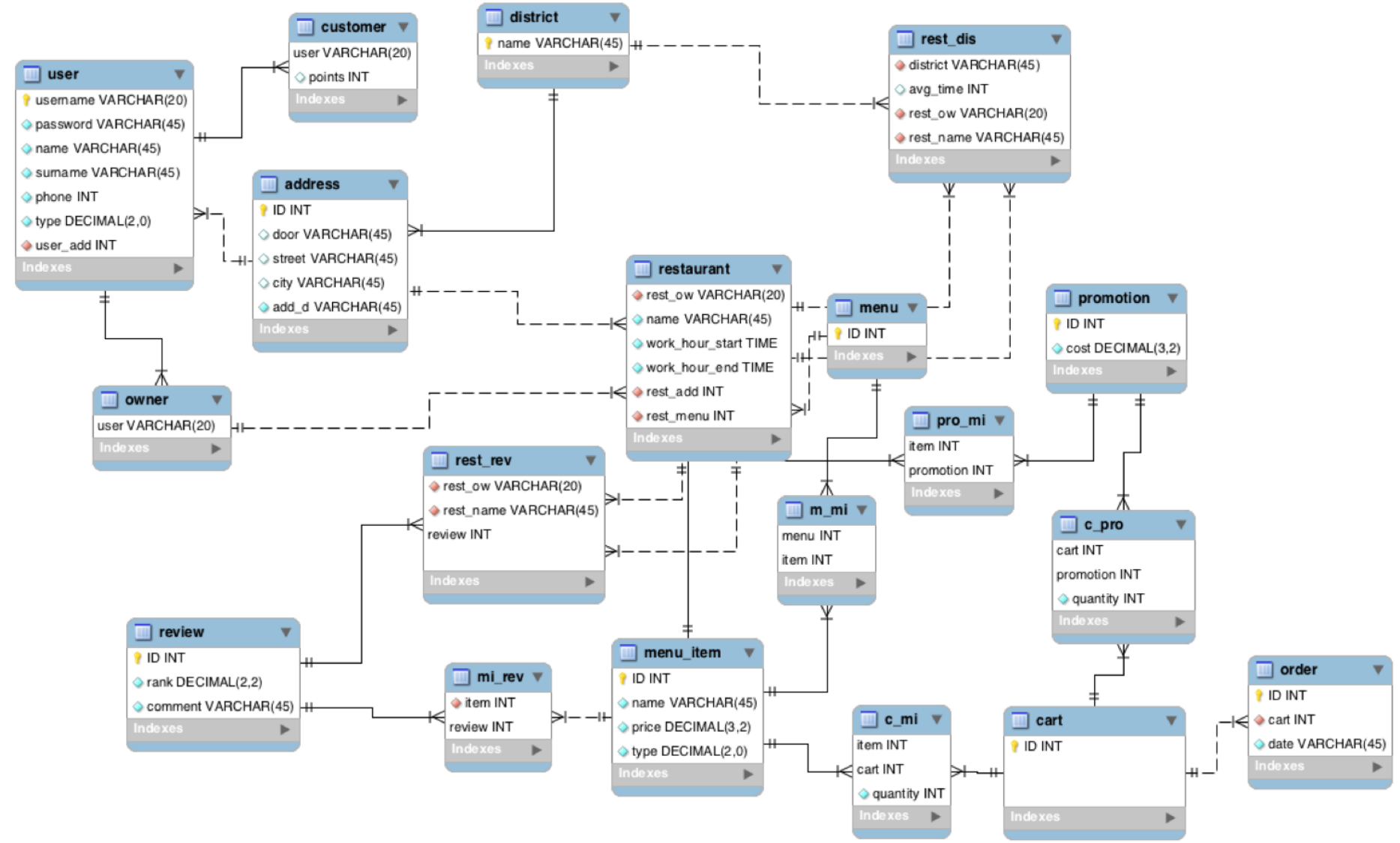
As can be seen in the figure (Figure 2.1), there are numerous relations that are smoothly connected to each other and holds the information for the corresponding fields of both restaurants and users. To start with, our most used and therefore crucial relations are simply as follows: *USER, CUSTOMER, OWNER, ADDRESS, DISTRICT, RESTAURANT, MENU, REVIEW, MENU, MENU_ITEM, PROMOTION, CART,* and *ORDER*. Each relation between these tables, e,g. one-to-one, one-to-many, and total participation, can be easily seen and analyzed from the figure. Moreover, the SQL statements for the creation of the table can be found in the Appendix (Appendix 5.1).

To start with, we will briefly discuss relations between objects and put an emphasis on the relations and entity sets that we considered them as hard to conceptualize and significant to discuss. ***Restaurant*** is a weak entity set that is identified by owner’s username and name attribute and therefore its primary keys are rest_ow and

name. We used Attribute Inheritance for ***owner*** and ***customer***, in this case, it extends to user relation. In order to make sure restaurant can have average delivery time for “regions”, we made ***district*** a set by itself and included it in address. Further, we combined rank and comment into ***review***, restaurant and menu_item have reviews, so to say.

Things have got a bit trickier while designing the ***menu*** relation and after long hours of brainstorm, we decided that restaurant chains (e.g. Domino’s, McDonnalds) can choose to make multiple restaurants use the same menu. Since It was not mentioned, menu is a set by itself. This created the need for 2 extra relations to connect restaurants and ***menu_items*** together. Alternative would be to have a menu relation that connects restaurants and menu_items directly.

Finally, ***c_pro*** makes sure promotions can be added to carts. carts now have menu_items (***c_mi***) and ***promotions***. Furthermore, ***order*** uses attribute inheritance as a way to store cart at the state the order was made. In doing so, database table creation state we can just refer to cart’s ID.


user
username VARCHAR(20)
password VARCHAR(45)
name VARCHAR(45)
surname VARCHAR(45)
phone INT
type DECIMAL(2,0)
user_add INT
Indexes
customer
user VARCHAR(20)
points INT
Indexes
district
name VARCHAR(45)
Indexes
rest_dis
district VARCHAR(45)
avg_time INT
rest_ow VARCHAR(20)
rest_name VARCHAR(45)
Indexes
address
ID INT
door VARCHAR(45)
street VARCHAR(45)
city VARCHAR(45)
add_d VARCHAR(45)
Indexes
restaurant
rest_ow VARCHAR(20)
name VARCHAR(45)
work_hour_start TIME
work_hour_end TIME
rest_add INT
rest_menu INT
Indexes
menu
ID INT
Indexes
promotion
ID INT
cost DECIMAL(3,2)
Indexes
owner
user VARCHAR(20)
Indexes
pro_mi
item INT
promotion INT
Indexes
rest_rev
rest_ow VARCHAR(20)
rest_name VARCHAR(45)
review INT
Indexes
m_mi
menu INT
item INT
Indexes
c_pro
cart INT
promotion INT
quantity INT
Indexes
review
ID INT
rank DECIMAL(2,2)
comment VARCHAR(45)
Indexes
mi_rev
item INT
review INT
Indexes
menu_item
ID INT
name VARCHAR(45)
price DECIMAL(3,2)
type DECIMAL(2,0)
Indexes
c_mi
item INT
cart INT
quantity INT
Indexes
cart
ID INT
Indexes
order
ID INT
cart INT
date VARCHAR(45)
Indexes

2.2Normalization Process

Database normalization is the process of arranging attributes and relations of a relational database in order to store the data in the most efficient way by minimizing data redundancy. Thus, it is composed of steps composed of organizing tables into less redundant relations without any information loss. The main objective here is to prevent our database model from repetitions and design a system that each operation can be done by using one table and delegate to smaller tables through defined primary keys.

In our project, one of our main objectives is to create a database model that serves its users efficiently and with the least data redundancy. Throughout the design phase of the project we also challenged ourselves to reach to this goal by normalizing our database at least at the 3NF level.

To start with, our design for the project is normalized by eliminating its redundancies and creating new relations to store attributes that are not dependent on the primary key. Further, each row in the user table is uniquely identifies one person as well as each column identifies exactly one attribute. The project also concerns about the cells in every table to be single valued and holds the same type of data. Consequently, our designed completed its first normalization form.

Although the steps we had already taken decrease the redundancy in our database, it was still incapable of organizing the data in a way that each attribute depends on only the primary key of the table. To illustrate, in our design, there are relations that are created to hold the data for other relations, e.g., “review” and “address”. In fact, review only keeps the attributes called “rank” and “comment” and identifies the rows with respect to it primary key called “ID”. Similarly, “address” relation also provides the data related to location information, namely “door”, “street”, and “city”. In doing so, we separated the attributes that are not dependent on the primary keys of the previous tables, respectively “restaurant” and “user”. Hence, we also used the requirements of the Second Normal Form(2NF) for data normalization by creating these tables and many others can be seen in our design.

After the 2NF step, our model was capable of organizing and providing the data that

are only related to each other, yet it lacked a solution for representing the attributes in the relations only by their candidate keys, but not by any non-prime attributes. Thus, in order to minimize the storage costs as well as improving database processing, we took a step further and applied Third Normal Form(3NF). Since there are some ordering processes contain different data, it was more of a requirement for our project to achieve rather than a decision. Then, we eliminated non-prime keys that have nothing to with the primary keys that are responsible for identifying every row in the relation, mostly by creating new relations. For instance, it can be noticed that the “avg_time” related to the district of the restaurant is not dependent on any primary key of neither “district” nor “restaurant” relations, however it is a field to be shown or calculated in the project. Hence, the fields called “total_cost” in the cart relation and “avg_time” are considered in the sense that they have to be calculated after each operation and can cause a repetition in our database. More importantly, they are independent from the primary keys in any relations, so they both have to be separated as a table or defined as a function in the code. Therefore, our goal to normalize our database as far as we could do reached to the 3NF with these two cases and many more can be seen in our design.

In order to take a step further, in other words, to minimize our database processes, the design makes sure that each determinant in relations must be a candidate key and it also makes it possible for our design to achieve Boyce-Codd Normal Form. Moreover, we also made sure that there are no multi-valued cells in our relations, a user that is a customer and also owns a restaurant at the same time so to say. The “user” relation hold a “type” field that prevent multi- valued cells in the relation. In doing so, the database model offered in our project is normalized to the Fourth Normal Form(4NF) by satisfying the minimal requirement for it.

In conclusion, all the normalization processes and our effort on normalizing our database model was aimed to design a database that has no redundancies, or as less as possible, and minimizing all the costs and complexities related to storage and database processes. Therefore, our project was normalized with the same motivation and goals defined through the discussion, however, there may be still some issues or data models in our database that cause the project. As we progress and dive into the

project further, we are dedicated to optimize all the resources and data structures we have now to design a database for our project, which is not yet to be completed or reached its final artifact.

3. Implementation

In the implementation, the group 6 has utilized all of the resources and skill sets that its members had and completed the project in a collaboration. In addition to the success in the teamwork, we used and enhanced our web development and database management skills throughout the project. In this section, the techniques we used and strategies we followed will be briefly discussed.

On the client side, we implemented and widely used the benefits of AngularJS while we have implemented our server side based on php. In order to store our tokens, we made use of our local storage. During the creation of the artifact and coding phase, git systems, via BitBucket, are used efficiently. In order to process a successful authentication, tokens are eventually sent and stored. Moreover, for security purposes, we stored passwords in hash.

In terms of functionality of the code, the terms type and job are defined and their corresponding functions are as follows. As it can be seen in the code, several jobs for each type are defined in a way that the program compiles the SQL queries according to the inputs given to the program. Further information about the code can be found in the additional files submitted with the project report.

Type 1	General Requests
Type 2	User Authentication/Creation
Type 3	Restaurant-related Processes
Type 4	Cart and User Information Editing

4. Results

As described in the project description, the idea of a portal that allows customers to order food online and to manage the requirements related to both client and server side is not a whole new concept, but has its several examples on the internet. Thus, the main goal was not to come up with totally new concept of a portal, in fact, we used the techniques and skills acquired both in the class and outside the class to create the expected portal in our design. So, we used and therefore improved all the skills and information learned from the course Database Management Systems (Comp 306).

Although it was our one of the main principles of the implementation to create a compact and integrated database, server side and client side systems, there is still an inevitable possibility of making some implementation mistakes, or bad design choices. However, in each part of our implementation, we aimed for the best results with the most optimal design choices we could do during the implementation, though, sometimes it turns out to be quite difficult to choose the best options for implementation.

In conclusion, throughout the project, Group 6 has experienced lots of challenges related to both implementation and design. In fact, they were comparatively harder to complete than the other parts of the project. Furthermore, the project has been quite knowledgeable for the project group since it prompts students to make use of all the information gained during the course, or maybe even more. In this report, the design and implementation processes of the project are briefly explained and provide an insight of the project in deeper details. For more information and related resources can be found in the Appendix chapter of the project report.

5. Appendix

5.1Table Statements

```
-- MySQL Script generated by MySQL Workbench
-- Sat Apr 23 17:07:15 2016
-- Model: New Model    Version: 1.0
-- MySQL Workbench Forward Engineering

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='TRADITIONAL,ALLOW_INVALID_DATES';

-- -----------------------------------------------------
-- Schema Group_6_db
-- -----------------------------------------------------

-- -----------------------------------------------------
-- Schema Group_6_db
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `Group_6_db` DEFAULT CHARACTER SET utf8 ;
USE `Group_6_db` ;

-- -----------------------------------------------------
-- Table `Group_6_db`.`district`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Group_6_db`.`district` (
  `name` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`name`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `Group_6_db`.`address`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Group_6_db`.`address` (
  `ID` INT NOT NULL,
  `door` VARCHAR(45) NULL,
  `street` VARCHAR(45) NULL,
  `city` VARCHAR(45) NULL,
  `add_d` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`ID`),
  INDEX `add_d_idx` (`add_d` ASC),
```

```
  CONSTRAINT `add_d`
    FOREIGN KEY (`add_d`)
    REFERENCES `Group_6_db`.`district` (`name`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `Group_6_db`.`user`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Group_6_db`.`user` (
  `username` VARCHAR(20) NOT NULL,
  `hash` CHAR(60) NOT NULL,
  `name` VARCHAR(45) NOT NULL,
  `surname` VARCHAR(45) NOT NULL,
  `phone` INT NOT NULL,
  `type` DECIMAL(2,0) NOT NULL,
  `user_add` INT NOT NULL,
  PRIMARY KEY (`username`),
  INDEX `user_add_idx` (`user_add` ASC),
  CONSTRAINT `user_add`
    FOREIGN KEY (`user_add`)
    REFERENCES `Group_6_db`.`address` (`ID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `Group_6_db`.`token`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Group_6_db`.`token` (
  `token` CHAR(40) NOT NULL,
  `username` VARCHAR(20) NOT NULL,
  PRIMARY KEY (`username`),
  INDEX `token_idx` (`token` ASC),
  CONSTRAINT `tokenusername`
    FOREIGN KEY (`username`)
    REFERENCES `Group_6_db`.`user` (`username`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `Group_6_db`.`customer`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Group_6_db`.`customer` (
  `user` VARCHAR(20) NOT NULL,
  `points` INT NULL,
  PRIMARY KEY (`user`),
  CONSTRAINT `username`
    FOREIGN KEY (`user`)
    REFERENCES `Group_6_db`.`user` (`username`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `Group_6_db`.`owner`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Group_6_db`.`owner` (
  `user` VARCHAR(20) NOT NULL,
  PRIMARY KEY (`user`),
  CONSTRAINT `ID`
    FOREIGN KEY (`user`)
    REFERENCES `Group_6_db`.`user` (`username`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `Group_6_db`.`menu`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Group_6_db`.`menu` (
  `ID` INT NOT NULL,
  PRIMARY KEY (`ID`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `Group_6_db`.`restaurant`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Group_6_db`.`restaurant` (
```

```
  `rest_ow` VARCHAR(20) NOT NULL,
  `name` VARCHAR(45) NOT NULL,
  `work_hour_start` TIME NOT NULL,
  `work_hour_end` TIME NOT NULL,
  `rest_add` INT NOT NULL,
  `rest_menu` INT NOT NULL,
  `payment` decimal(2,0),
  INDEX `name_idx` (`name` ASC),
  INDEX `user_idx` (`rest_ow` ASC),
  INDEX `ID_idx` (`rest_add` ASC),
  INDEX `ID_idx1` (`rest_menu` ASC),
  PRIMARY KEY (`rest_ow`, `name`),
  CONSTRAINT `user`
    FOREIGN KEY (`rest_ow`)
    REFERENCES `Group_6_db`.`owner` (`user`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `rest_add`
    FOREIGN KEY (`rest_add`)
    REFERENCES `Group_6_db`.`address` (`ID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `rest_menu`
    FOREIGN KEY (`rest_menu`)
    REFERENCES `Group_6_db`.`menu` (`ID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `Group_6_db`.`rest_dis`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Group_6_db`.`rest_dis` (
  `district` VARCHAR(45) NOT NULL,
  `rest_ow` VARCHAR(20) NOT NULL,
  `rest_name` VARCHAR(45) NOT NULL,
  `avg_minutes` DECIMAL(3,0),
  INDEX `name_idx` (`rest_name` ASC),
  INDEX `rest_ow_idx` (`rest_ow` ASC),
  CONSTRAINT `name`
    FOREIGN KEY (`district`)
    REFERENCES `Group_6_db`.`district` (`name`)
```

```
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `rest_name`
    FOREIGN KEY (`rest_name`)
    REFERENCES `Group_6_db`.`restaurant` (`name`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `rest_ow`
    FOREIGN KEY (`rest_ow`)
    REFERENCES `Group_6_db`.`restaurant` (`rest_ow`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `Group_6_db`.`menu_item`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Group_6_db`.`menu_item` (
  `ID` INT NOT NULL,
  `menu` INT NOT NULL,
  `name` VARCHAR(45) NOT NULL,
  `price` DECIMAL(3,2) NOT NULL,
  `type` DECIMAL(2,0) NOT NULL,
  `ordercount` INT,
  CONSTRAINT `mi_menu`
    FOREIGN KEY (`menu`)
    REFERENCES `Group_6_db`.`menu` (`ID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  PRIMARY KEY (`ID`,`menu`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `Group_6_db`.`review`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Group_6_db`.`review` (
  `ID` INT NOT NULL,
  `rank` DECIMAL(2,2) NOT NULL,
  `comment` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`ID`))
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `Group_6_db`.`rest_rev`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Group_6_db`.`rest_rev` (
  `rest_ow` VARCHAR(20) NOT NULL,
  `rest_name` VARCHAR(45) NOT NULL,
  `review` INT NOT NULL,
  `orderID` INT NOT NULL,
  INDEX `name_idx` (`rest_name` ASC),
  INDEX `rest_ow_idx` (`rest_ow` ASC),
  INDEX `orderID_idx` (`orderID` ASC),
  PRIMARY KEY (`orderID`, `rest_ow`, `rest_name`),
  CONSTRAINT `order_rest_rev`
    FOREIGN KEY (`orderID`)
    REFERENCES `Group_6_db`.`order` (`ID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `rest_name_rev`
    FOREIGN KEY (`rest_name`)
    REFERENCES `Group_6_db`.`restaurant` (`rest_ow`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `rest_ow_rev`
    FOREIGN KEY (`rest_ow`)
    REFERENCES `Group_6_db`.`restaurant` (`rest_ow`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `rest_review_ID`
    FOREIGN KEY (`review`)
    REFERENCES `Group_6_db`.`review` (`ID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `Group_6_db`.`rest_pro`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Group_6_db`.`rest_pro` (
  `rest_ow` VARCHAR(20) NOT NULL,
  `rest_name` VARCHAR(45) NOT NULL,
```

```
  `promotion` INT NOT NULL,
  INDEX `name_idx` (`rest_name` ASC),
  INDEX `rest_ow_idx` (`rest_ow` ASC),
  PRIMARY KEY (`promotion`, `rest_ow`, `rest_name`),
  CONSTRAINT `rest_name_pro`
    FOREIGN KEY (`rest_name`)
    REFERENCES `Group_6_db`.`restaurant` (`name`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `rest_ow_pro`
    FOREIGN KEY (`rest_ow`)
    REFERENCES `Group_6_db`.`restaurant` (`rest_ow`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `rest_pro_id`
    FOREIGN KEY (`promotion`)
    REFERENCES `Group_6_db`.`promotion` (`ID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `Group_6_db`.`rest_fav`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Group_6_db`.`rest_fav` (
  `rest_ow` VARCHAR(20) NOT NULL,
  `rest_name` VARCHAR(45) NOT NULL,
  `user` VARCHAR(20) NOT NULL,
  INDEX `name_idx` (`rest_name` ASC),
  INDEX `rest_ow_idx` (`rest_ow` ASC),
  PRIMARY KEY (`user`, `rest_ow`, `rest_name`),
  CONSTRAINT `rest_name_fav`
    FOREIGN KEY (`rest_name`)
    REFERENCES `Group_6_db`.`restaurant` (`name`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `rest_ow_fav`
    FOREIGN KEY (`rest_ow`)
    REFERENCES `Group_6_db`.`restaurant` (`rest_ow`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `rest_user_fav`
```

```
  FOREIGN KEY (`user`)
  REFERENCES `Group_6_db`.`user` (`username`)
  ON DELETE CASCADE
  ON UPDATE CASCADE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `Group_6_db`.`mi_rev`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Group_6_db`.`mi_rev` (
 `item` INT NOT NULL,
 `review` INT NOT NULL,
 PRIMARY KEY (`review`),
 INDEX `ID_idx` (`item` ASC),
 CONSTRAINT `mi_rev_item`
  FOREIGN KEY (`item`)
  REFERENCES `Group_6_db`.`menu_item` (`ID`)
  ON DELETE CASCADE
  ON UPDATE CASCADE,
 CONSTRAINT `mi_rev_rev`
  FOREIGN KEY (`review`)
  REFERENCES `Group_6_db`.`review` (`ID`)
  ON DELETE CASCADE
  ON UPDATE CASCADE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `Group_6_db`.`promotion`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Group_6_db`.`promotion` (
 `ID` INT NOT NULL,
 `cost` DECIMAL(3,2) NOT NULL,
 PRIMARY KEY (`ID`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `Group_6_db`.`cart`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Group_6_db`.`cart` (
 `ID` INT NOT NULL,
```

```
  PRIMARY KEY (`ID`))
ENGINE = InnoDB;


-- -- -----------------------------------------------------
-- -- Table `Group_6_db`.`m_mi`
-- -- -----------------------------------------------------
-- CREATE TABLE IF NOT EXISTS `Group_6_db`.`m_mi` (
--   `menu` INT NOT NULL,
--   `item` INT NOT NULL,
--   PRIMARY KEY (`menu`, `item`),
--   INDEX `ID_idx` (`item` ASC),
--   CONSTRAINT `m_mi_menu`
--     FOREIGN KEY (`menu`)
--     REFERENCES `Group_6_db`.`menu` (`ID`)
--     ON DELETE CASCADE
--     ON UPDATE CASCADE,
--   CONSTRAINT `m_mi_item`
--     FOREIGN KEY (`item`)
--     REFERENCES `Group_6_db`.`menu_item` (`ID`)
--     ON DELETE CASCADE
--     ON UPDATE CASCADE)
-- ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `Group_6_db`.`pro_mi`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Group_6_db`.`pro_mi` (
  `item` INT NOT NULL,
  `promotion` INT NOT NULL,
  PRIMARY KEY (`item`, `promotion`),
  INDEX `ID_idx` (`promotion` ASC),
  CONSTRAINT `pro_mi_pro`
    FOREIGN KEY (`promotion`)
    REFERENCES `Group_6_db`.`promotion` (`ID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `pro_mi_ID`
    FOREIGN KEY (`item`)
    REFERENCES `Group_6_db`.`menu_item` (`ID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
```

```
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `Group_6_db`.`c_mi`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Group_6_db`.`c_mi` (
  `item` INT NOT NULL,
  `cart` INT NOT NULL,
  `quantity` INT NOT NULL,
  PRIMARY KEY (`item`, `cart`),
  INDEX `ID_idx` (`cart` ASC),
  CONSTRAINT `c_mi_c`
    FOREIGN KEY (`cart`)
    REFERENCES `Group_6_db`.`cart` (`ID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `c_mi_mi`
    FOREIGN KEY (`item`)
    REFERENCES `Group_6_db`.`menu_item` (`ID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `Group_6_db`.`order`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Group_6_db`.`order` (
  `ID` INT NOT NULL,
  `cart` INT NOT NULL,
  `user` VARCHAR(45) NOT NULL,
  `date` VARCHAR(45) NOT NULL,
  `rest_ow` VARCHAR(20) NOT NULL,
  `rest_name` VARCHAR(45) NOT NULL,
  `status` INT NOT NULL,
  PRIMARY KEY (`ID`),
  INDEX `ID_idx` (`cart` ASC),
  CONSTRAINT `rest_name_order`
    FOREIGN KEY (`rest_name`)
    REFERENCES `Group_6_db`.`restaurant` (`rest_ow`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
```

```
  CONSTRAINT `rest_ow_order`
    FOREIGN KEY (`rest_ow`)
    REFERENCES `Group_6_db`.`restaurant` (`rest_ow`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `order_cart`
    FOREIGN KEY (`cart`)
    REFERENCES `Group_6_db`.`cart` (`ID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `order_user`
    FOREIGN KEY (`user`)
    REFERENCES `Group_6_db`.`user` (`username`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `Group_6_db`.`user_cart`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Group_6_db`.`user_cart` (
  `cart` INT NOT NULL,
  `user` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`user`),
  INDEX `ID_idx` (`cart` ASC),
  CONSTRAINT `uc_cart`
    FOREIGN KEY (`cart`)
    REFERENCES `Group_6_db`.`cart` (`ID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `uc_user`
    FOREIGN KEY (`user`)
    REFERENCES `Group_6_db`.`user` (`username`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `Group_6_db`.`c_pro`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `Group_6_db`.`c_pro` (
```

```
  `cart` INT NOT NULL,
  `promotion` INT NOT NULL,
  `quantity` INT NOT NULL,
  PRIMARY KEY (`cart`, `promotion`),
  INDEX `ID_idx` (`cart` ASC),
  INDEX `ID_idx1` (`promotion` ASC),
  CONSTRAINT `c_pro_cart`
    FOREIGN KEY (`cart`)
    REFERENCES `Group_6_db`.`cart` (`ID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `c_pro_pro`
    FOREIGN KEY (`promotion`)
    REFERENCES `Group_6_db`.`promotion` (`ID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;


SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```