《数据库系统原理》大作业

系统设计报告

题目名称：餐厅订餐系统

学号及姓名：18373502 杨周启柯

18373513 朱辰啸

18373515 单乾

2020年 12月 13日

组内同学承担任务说明

|  |  |  |
| --- | --- | --- |
|  | 负责同学 | 备注 |
| 子任务1：  系统功能设计与数据库设计 | 朱辰啸 |  |
| 子任务2：  系统服务器端开发 | 杨周启柯 |  |
| 子任务3：  系统客户端开发 | 单乾 |  |

1. **需求分析**
2. 需求描述

在这次大作业的设计过程中，我们计划实现一个餐厅订餐系统，其内部存储了如食品、套餐、订单等和顾客点餐的相关信息，以及部分独属于餐厅本身的信息，如员工、餐厅地址等。使用这个系统，顾客可以根据餐厅提供的菜品随时下单，也可以对每次的订单进行反馈；而数据库管理员则可以通过增删查改等操作完成对整个系统的维护。

我们对整个餐厅订餐系统进行如下描述：

系统内包括若干餐厅，每个餐厅有若干员工在职。系统中有若干顾客账号，每个顾客账号可以有多个取餐信息。每个顾客账号都可以提交多个订单，每个订单都需要唯一的对应一个餐厅，订单内容可以有食品或套餐，其中每个套餐都由若干食品组成。在提交订单后，顾客账户可以对某个订单进行评价反馈。

根据以上描述，我们可以明确，本系统的主要设计目标是给顾客提供一个高效的下单系统，同时给餐厅提供数据库管理系统。进而，我们可以相应的进行功能设计，主要功能如下：

（1）用户管理：整个系统存在管理员、顾客两种用户，管理员的用户信息在系统初始化时已经存储在数据库中，而顾客用户则需要进行注册。

（2）数据管理：餐厅、员工、食品、套餐等系统“静态”信息的管理全部由管理员进行实现，而订单、评价反馈、取餐信息等顾客相关信息则由顾客用户“动态”的进行增删改等操作。

（3）数据查询：对于数据库管理员，系统中的所有数据均可以进行查询，而对于顾客，除了系统中的“静态”信息外，只有属于自己用户的信息才可以查询。

根据以上功能设计，本系统的管理员可以查询系统内的全部数据并对所有“静态”数据进行增删改，而顾客用户则可以查询系统内的“静态”数据以及属于本用户的全部数据，同时可以对属于本用户的数据进行增删改操作。

1. 数据流图

根据以上需求描述及功能设计，我们可以设计得到相应的数据流图。该数据流图可以表达数据在系统内的逻辑流向以及逻辑变换过程，从而表达系统的逻辑功能。

本系统的数据流图如下：

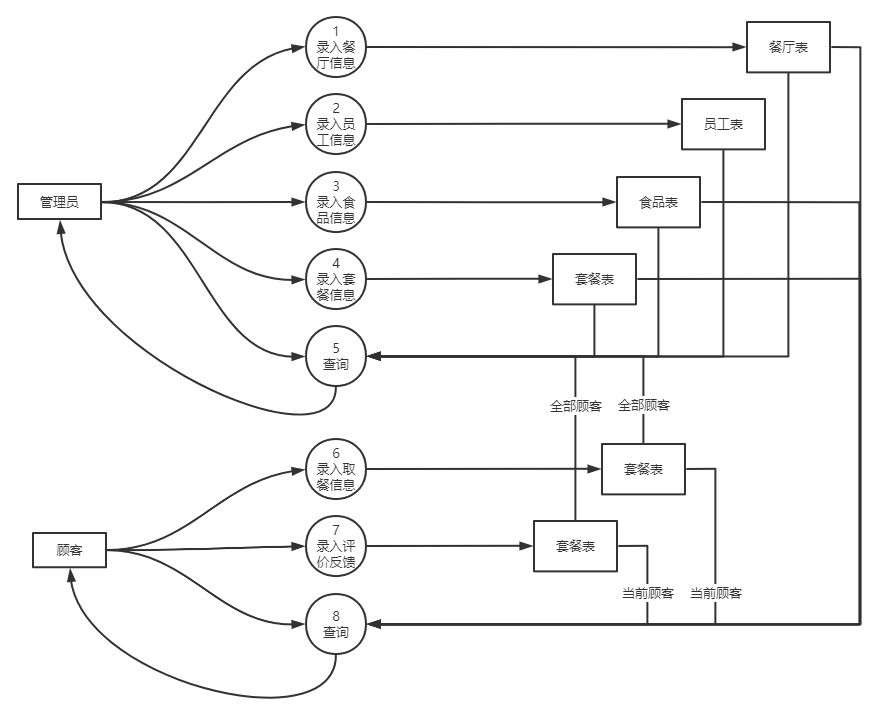


图1 数据流图

1. 数据元素表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 数据项名 | 别名 | 数据类型 | 宽度 | 含义 |
| 取餐信息的编号 | id\_ai | INT | 4 | 唯一标识取餐信息 |
| 取餐人姓名 | name | VARCHAR | 45 | 取餐人姓名 |
| 取餐人电话 | tele | VARCHAR | 45 | 取餐人电话 |
| 管理员的编号 | id\_administrator | INT | 4 | 唯一标识管理员账号 |
| 管理员账户名 | name\_administrator | VARCHAR | 45 | 管理员账户名 |
| 管理员账户密码 | password\_administrator | VARCHAR | 45 | 管理员账户密码 |
| 顾客的编号 | id\_customer | INT | 4 | 唯一标识顾客 |
| 顾客账户名 | username\_customer | VARCHAR | 45 | 顾客账户名 |
| 顾客账户密码 | password\_customer | VARCHAR | 45 | 顾客账户密码 |
| 员工的编号 | id\_employee | INT | 4 | 唯一标识员工 |
| 员工姓名 | name\_employee | VARCHAR | 45 | 员工姓名 |
| 员工年龄 | age\_employeer | VARCHAR | 45 | 员工年龄 |
| 员工职位 | occupation\_employee | VARCHAR | 45 | 员工职位 |
| 员工电话 | tele\_employee | VARCHAR | 45 | 员工电话 |
| 入职时间 | in\_date | VARCHAR | 45 | 入职时间 |
| 员工职位 | occupation | VARCHAR | 45 | 员工职位 |
| 离职时间 | leave\_date | VARCHAR | 45 | 离职时间 |
| 反馈信息的编号 | id\_feedback | INT | 4 | 唯一标识反馈信息 |
| 反馈时间 | time\_feedback | VARCHAR | 45 | 反馈时间 |
| 反馈内容 | content\_feedback | VARCHAR | 200 | 反馈内容 |
| 食物的编号 | id\_food | INT | 4 | 唯一标识食物 |
| 食物的名字 | name\_food | VARCHAR | 45 | 食物的名字 |
| 食物的价格 | price\_food | FLOAT | 4 | 食物的价格 |
| 食物的数量 | num\_food | INT | 4 | 食物的数量 |
| 工作名称 | name\_job | VARCHAR | 45 | 工作名称 |
| 月工资 | salary\_job | VARCHAR | 45 | 月工资 |
| 订单编号 | id\_order | INT | 4 | 唯一标识订单 |
| 订单总价格 | tot\_price | FLOAT | 4 | 订单总价格 |
| 订餐时间 | time\_order | VARCHAR | 45 | 订餐时间 |
| 餐厅编号 | id\_restaurant | INT | 4 | 唯一标识餐厅 |
| 餐厅地址 | loc\_restaurant | VARCHAR | 45 | 餐厅地址 |
| 餐厅容纳人数 | accom\_restaurant | VARCHAR | 45 | 餐厅容纳人数 |
| 餐厅电话 | tele\_restaurant | VARCHAR | 45 | 餐厅电话 |
| 套餐的编号 | id\_set\_meal | INT | 4 | 唯一标识套餐 |
| 套餐名字 | name\_set\_meal | VARCHAR(45) | 45 | 套餐名字 |
| 套餐价格 | price\_set\_meal | FLOAT | 4 | 套餐价格 |
| 套餐的数量 | num\_set\_meal | INT | 4 | 套餐的数量 |
| 套餐的编号 | id\_set\_meal | INT | 4 | 套餐的编号 |
| 食物的编号 | id\_food | INT | 4 | 食物的编号 |
| 食物的数量 | num | INT | 4 | 食物的数量 |

1. **数据库概念模式设计**
2. 系统初步E-R图

根据以上设计内容以及数据元素表的定义，我们共计设置了8个实体。进一步地，我们可以为每个实体构造实体属性图，结果如下：

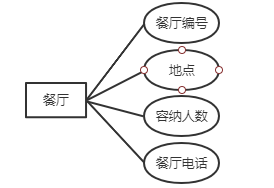


图2 餐厅实体属性图

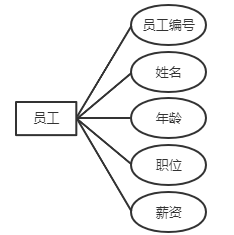


图3 员工实体属性图

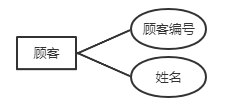


图4 顾客实体属性图

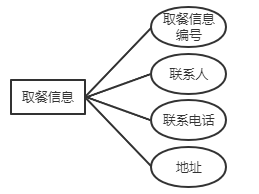


图5 取餐信息实体属性图

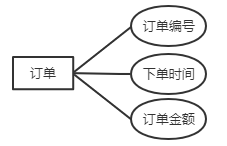


图6 订单实体属性图

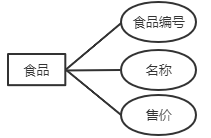


图7 食品实体属性图

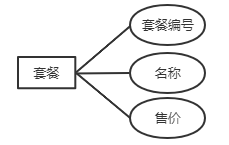


图8 套餐实体属性图

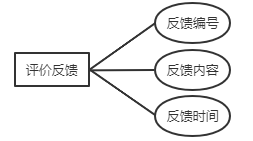


图9 评价反馈实体属性图

得到以上实体属性图后，我们根据实体的实际含义又设计了9个实体间联系，并将实体和联系整合到一张图中，最终得到了如下系统初步E-R图（图中已略去实体属性）：

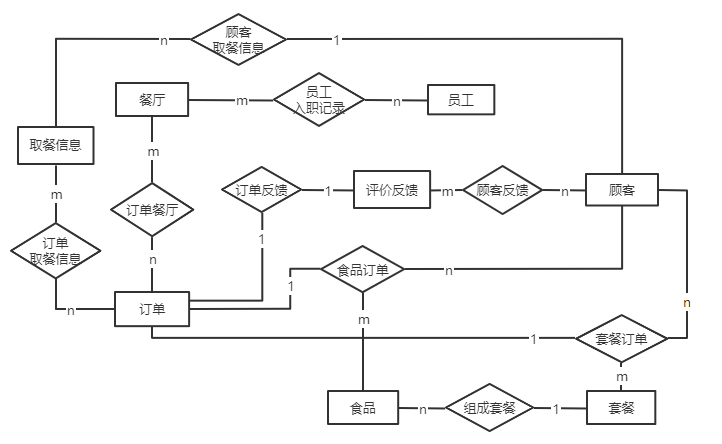


图10 数据库系统初步E-R图

1. 系统基本E-R图

对于上述系统初步E-R图，我们可以进行进一步分析。

根据报告中第一部分的需求分析，每个顾客针对子集已完成的特定订单，可以进行评价反馈，这样的关系表示在初步E-R图中表示为了两个联系，即一个顾客提交反馈的联系和一个评价反馈和订单之间一一对应的联系。我们不难发现，这样的设计事实上存在一定的冗余，我们可以将这两个联系进行合并，并以反馈编号作为新联系的主码。

在进行上述简化后，我们可以得到如下基本E-R图：

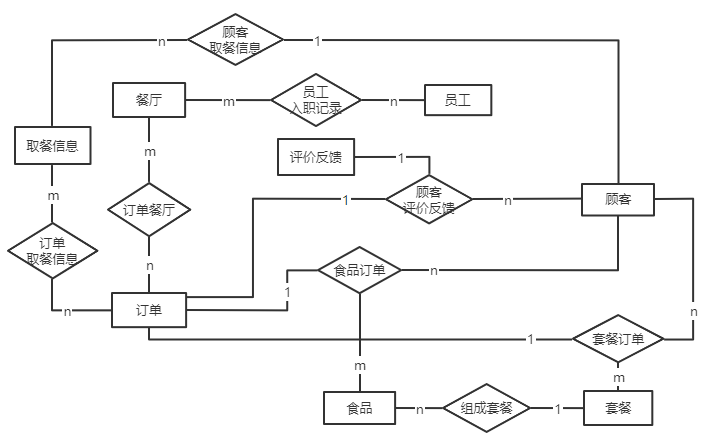


图11 数据库系统基本E-R图

至此，我们就得到了最终的系统基本E-R图，完成了餐厅订餐系统的概念模式设计。

1. **数据库逻辑模式设计**
2. 数据库关系模式

根据E-R图向关系模式的转换规则，我们可以将上文中得到的系统基本E-R图转化为如下关系模式：

（1）实体

餐厅（餐厅编号，地点，容纳人数，餐厅电话）

员工（员工编号，姓名，年龄，职位，薪资）

顾客（顾客编号，姓名）

订单（订单编号，下单时间，订单金额）

食品（食品编号，名称，售价）

套餐（套餐编号，名称，售价）

评价反馈（反馈编号，反馈内容，反馈时间）

取餐信息（取餐信息编号，联系人，联系电话，地址）

（2）实体间联系

员工入职记录（员工编号，餐厅编号，入职时间，职位，离职时间）

顾客取餐信息（顾客编号，取餐信息编号）

订单取餐信息（订单编号，取餐信息编号）

订单餐厅（餐厅编号，订单编号）

食品订单（订单编号，食品编号）

套餐订单（订单编号，套餐编号）

顾客评价反馈（反馈编号，订单编号，顾客编号）

组成套餐（套餐编号，食品编号，食品数量）

注：关系模式中含有下划线的属性组为主码。

1. 关系模式范式等级的判定与规范化
2. 函数依赖分析

餐厅表：

候选码：餐厅编号，地点

函数依赖：餐厅编号→容纳人数，餐厅编号→地点，餐厅编号→餐厅电话，地点→餐厅编号

员工表：

候选码：员工编号

函数依赖：员工编号→姓名，员工编号→年龄，员工编号→职位，员工编号→薪资，职位→薪资

顾客表：

候选码：顾客编号

函数依赖：顾客编号→姓名

食品表：

候选码：食品编号，名称

函数依赖：食品编号→名称，食品编号→售价，食品编号→数量，名称→食品编号

套餐表：

候选码：套餐编号，名称

函数依赖：套餐编号→名称，套餐编号→售价，名称→套餐编号

评价反馈表：

候选码：反馈编号

函数依赖：反馈编号→反馈内容，反馈编号→反馈时间

取餐信息表：

候选码：取餐信息编号

函数依赖：取餐信息编号→联系人，取餐信息编号→联系电话，取餐信息编号→地址

订单表：

候选码：订单编号

函数依赖：订单编号→下单时间，订单编号→订单金额

员工入职记录表：

候选码：（员工编号，餐厅编号，入职时间）

函数依赖：（员工编号，餐厅编号，入职时间）→ 职位，（员工编号，餐厅编号，入职时间）→ 离职时间

顾客取餐信息表：

候选码：（顾客编号，取餐信息编号）

函数依赖：无

订单取餐信息表：

候选码：（订单编号，取餐信息编号）

函数依赖：无

订单餐厅表：

候选码：（订单编号，餐厅编号）

函数依赖：无

食品订单表：

候选码：（订单编号，食品编号）

函数依赖：无

套餐订单表：

候选码：（套餐编号，食品编号）

函数依赖：无

顾客评价反馈表：

候选码：（反馈编号，订单编号，顾客编号）

函数依赖：无

组成套餐表：

候选码：（套餐编号，食品编号）

函数依赖：（套餐编号，食品编号）→ 食品数量

1. 范式等级判定与规范化

根据上述函数依赖关系，我们不难发现，所有关系模式中的每个非主属性都完全依赖于码，因此以上关系模式属于2NF。

对于员工表，我们发现它的码“员工编号”到属性“薪资”的函数依赖关系为传递依赖（存在函数依赖：“员工编号→职位”和“职位→薪资”），因此上述关系模式不属于3NF。

为了将关系模式规范到3NF，我们使用投影分解的方法将员工表分解为如下两个表：

员工（员工编号，姓名，年龄，职位）

职位表（职位，薪资）

经过进一步函数依赖分析，我们可以发现，规范化后的关系模式达到了3NF的标准。

1. 数据库设计优化

为了提高存储效率和存储空间的利用率，我们可以对关系模式进行水平或垂直分解。

水平分解，即将关系的元组分为若干子集合，定义每个子集合为一个子关系，以提高系统效率。

垂直分解，即将关系模式的属性分解为若干子集合，进而形成若干子关系模式。垂直分解的原则为将经常在一起使用的属性从关系模式中分解出来，形成一个子关系模式。同时，垂直分解的过程必须确保无损连接性和函数依赖。

在本系统中，我们的主要使用对象为顾客和餐厅管理人员。

对于顾客用户，主要操作包括下单、评价反馈、添加取餐信息和查询餐厅信息，其中下单为最高频的操作。在下单的过程中，顾客用户会经常查询食品和套餐的信息，而对于这两类信息，当前关系模式已经分别用两个基本表进行了存储，且基本表内无冗余属性。

对于餐厅管理人员，我们可以将其主要操作分为系统维护和信息查询。在系统维护的过程中，管理员进行增删改操作，联系实际，我们可以认为这类操作的频率较低。而对于信息查询操作，管理员拥有查询系统内所有信息的权限，联系存储信息的实际含义，我们可以认为管理员会较高频的查询评价反馈信息，而对其他信息的查询频率较为均衡。而对于评价反馈信息，本餐厅订餐系统将其存储在了一个基本表中，内容包括反馈编号、内容和反馈时间，内容并无冗余。

因此，结合关系模式的优化目的以及本系统的实际情况，我们认为当前关系模式无需进行进一步优化调整。