CS387: Database and Information Systems Lab

Project : Restaurant Management System
Project Design

Ayush Jangir (190050025)
D Chandrasekhara S S Hethahavya (190050031)
Koustav Sen (190050062)
Vishwanth K (190050131)

Department of Computer Science and Engineering
Indian Institute of Technology Bombay
2021-2022

Contents

1. Logical Schema

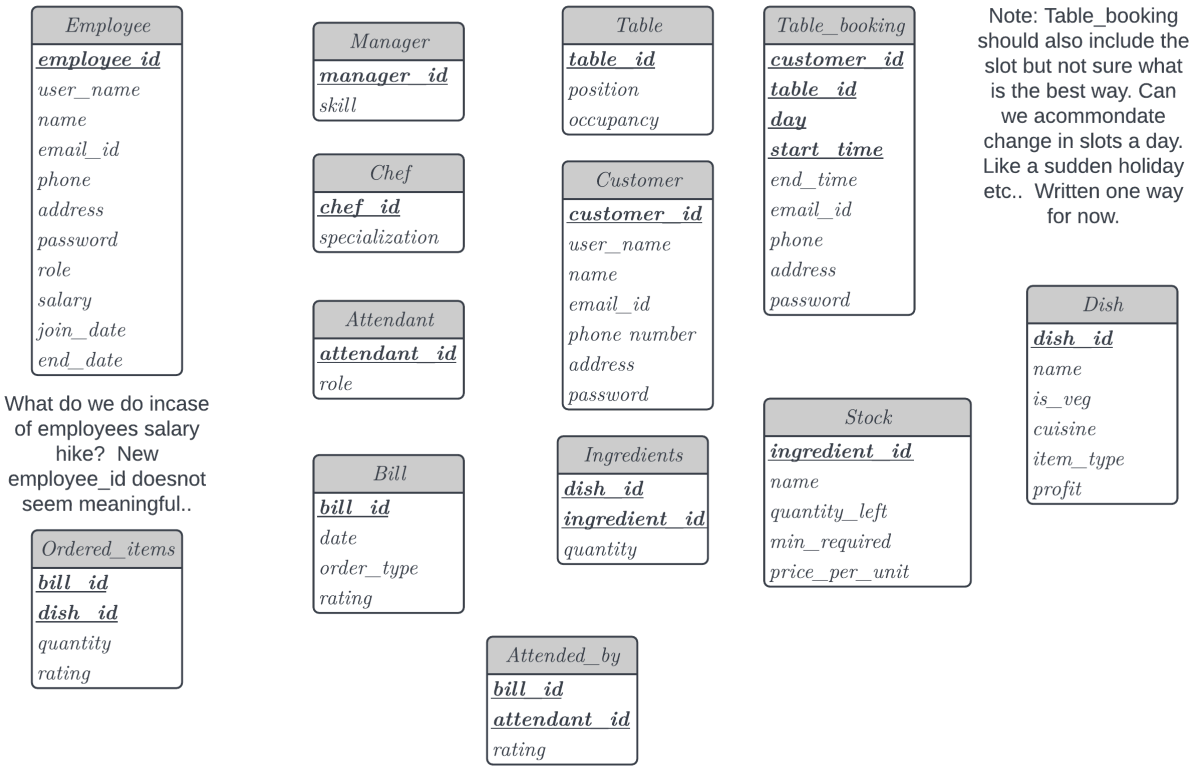

Figure 1.1: Logical Schema

2. Integrity Constraints and explanations

2.1 Employee

- `employee_id` Primary Key
- `user_name` Unique Constraint

2.2 Manager

- `manager_id` Primary Key
- `manager_id` references Employee

2.3 Attendant

- `attendant_id` Primary Key
- `attendant_id` references Employee

2.4 Chef

- `chef_id` Primary Key
- `chef_id` references Employee

2.5 Table

- `table_id` Primary Key

2.6 Bill

- `bill_id` Primary Key
- `rating` field can be `NULL` indicating that the customer has not yet given the rating for the **hotel service**.

2.7 Ingredient

- `ingredient_id Primary Key`

2.8 Dish

- `dish_id Primary Key`

2.9 Customer

- `customer_id Primary Key`
- `Unique phone number`
- `Unique email_id`
- `Unique user_name`

2.10 Ingredients

- `(dish_id, ingredient_id) Primary Key`
- `dish_id references Dish`
- `ingredient_id references Ingredient`
- `quantity` field is greater than zero. Discuss if such things are really necessary?

2.11 Table_Booking

- `(customer_id, table_id, date, start_time) Primary Key`
- `customer_id references Customer`
- `table_id references Table`

2.12 Ordered_Items

- `(bill_id, dish_id) Primary Key`
- `bill_id references Bill`
- `dish_id references dish`
- `rating` can be `NULL`. `NULL` here means that the customer has not given the rating for the **dish** yet.

2.13 Attended_by

- `(bill_id, attendant_id) Primary Key`
- `bill_id references Bill`
- `attendant_id references Attendant`
- `rating` can be `NULL`. `NULL` here means that the customer has not given the rating for the **attendant** yet.

3. Materialised views and explanations

3.1 Menu

- This view calculates the costs of all dishes using the cost of ingredients and the profit percentage on that dish.
- Schema for the view:

Menu
Dish_id
Dish_name
Cuisine
is_veg
item_type

- We need this as a materialised view because Menu is the single most important thing for any decent restaurant. So the number of requests that request for Menu will be very high. Hence by materialising it we can process the menu queries faster.

3.2 Past Orders

- This view collects all past orders of a given customer and displays them
- Schema for the view:

Orders
Bill_id
Dish_id
Dish_name
Quantity
Rating

- This view is important for the customer to give his reviews for his/her orders

4. Forms and Transactions for use cases

4.1 Register Customer

- Input :

user name
email id
name
phone number
address
password

- System validates integrity constraints and upon success registers a new customer.

4.2 Login

- Input :

user name
password

- System validates if any user with given username exists in the appropriate role table. If success logs in the home page for the specific account.

4.3 Table Booking

- Input :

date
time
members

- System validates if any table with given occupancy is free for given date and time. If success reserves the table for the customer.

4.4 Take out

- Input :

dish id
quantity

- System creates a new order with given dishes for the customer.

4.5 Register Employee

- Input :

user name
email id
name
phone number
address
password
salary
role

- System validates integrity constraints and upon success registers a new employee with the given role.

4.6 Assign waiter to table

- Input :

table id
attendant id
customer id
date
time

- System checks if a table is booked by the customer for the given date and time. System also confirms if a attendant is not assigned to other table for that time stamp. If success then assigns attendant to table.

4.7 Create new Bill

- Input :

dish id
quantity
customer id
attendant id
order type
date

- System creates a new bill for the customer

5. SQL Files

5.1 `DDL.sql`

5.2 `InsertData.py`

These files can be viewed in the github repo click here

6. SQL for Analysis and Views

6.1 top-3-items-by-type:

with dish_qt(dish_id, net_qt, dish_type, dish_name) as (select dish.dish_id, sum(quantity), item_type, dish_name from ordered_items, dish where ordered_items.dish_id = dish.dish_id group by(dish.dish_id)) select dish_name, dish_type from (select dish_id, dish_name, dish_type, RANK() OVER(PARTITION BY dish_type ORDER BY net_qt DESC) as dish_rank from dish_qt) as temp where temp.dish_rank <= 3 ORDER BY dish_type, dish_rank ASC

6.2 Best-day-of week by number of orders:

select day_of_week, count(*) as num_orders from (select to_char(bill_time, 'dy') as day_of_week from bill) as temp group by day_of_week ORDER BY num_orders ASC LIMIT 1;

6.3 menu :

select dish.dish_id, dish.dish_name, sum(ing.quantity * stock.price_per_unit) * (1 + dish.profit_percentage/100.0) as dish_price from ingredients as ing, stock, dish where ing.stock_id = stock.stock_id AND dish.dish_id = ing.dish_id group by(dish.dish_id) ORDER BY dish_price DESC

6.4 monthly-sales

with bill_details(bill_id, net_bill, year, month_txt, month_num) as (select bill.bill_id as bill_id, SUM(menu.dish_price * ordered_items.quantity) as net_bill, extract(year from bill.bill_time) as year, to_char(bill.bill_time, 'month') as month_txt, extract(month from bill.bill_time) as month_num from ordered_items, menu, bill where ordered_items.dish_id = menu.dish_id and ordered_items.bill_id = bill.bill_id group by bill.bill_id) select year, month_txt, sum(net_bill) as total_sales from bill_details group by(year, month_num, month_txt) ORDER BY year, month_num ASC