INST 733 Database Design—Menu Management Database

Priyanka Baskaran and Kuan-Yu Chen

Introduction

The goal of the project was to create a database that would provide a platform for both managers and customers. Managers can track placed orders and manage menus of restaurants. Customers would be able to search menus for specific restaurants, place orders, mark items on menu as favorite items, and give ratings to specific items of restaurants.

Motivation

The reason we tried to build this database is that though most restaurants have point-of-sale systems that help restaurants to manage orders, the menus data stored in the POS system is separate from the database what customers can search from online. As a result, customers may encounter several problems when searching for menus of specific restaurants, such as no menu data available online, incorrect items or prices, etc. Therefore, we hope restaurants can use the same database as the one customers are be able to search online, so that customers can always have the most updated menus if needed.

Update of the Proposal

The goal of the final database is the same as we mentioned in the proposal. The only two different parts from the proposal are the entities we have and the scope of the sample data, which will be elaborated in the later parts of the report.

Logical Design

The original planed entities in proposal were as follows:

1. Restaurant
    - restaurantID
    - name
    - menuID
    - addressID
2. Cuisine
    - cuisineID
    - name

- price

3. Order
   - orderID
   - cuisineID
   - quantity
   - date
   - customerID
4. Customer
   - customerID
   - password
   - last_name
   - first_name
   - phone_number
   - addressID
5. Address
   - addressID
   - street
   - city
   - zipcode
6. PendingOrder
   - orderID
7. Menu
   - menuID
   - cuisineID
8. CompletedOrder
   - orderID
   - finishedTime
9. FavoriteFood
   - customerID
   - cusineID
10. CreditcardInfo
    - customerID
    - digit

- expire_date
- cvc_no

11. Photo
    - restaurantID
    - url
12. Rating
    - rating
    - cuisineID
    - customerID
    - date
13. Manager

- managerID
- password
- first_name
- last_name
- phone_number
- restaurantID

In our final ERD below(Fig.1), we added more attributes to each entity and deleted the “Address” entity because the address is not the focus of this database and we do not want to track the history of address. Also, we deleted the “PendingOrder” and “ CompletedOrder” ,and included these two attributes in the “Orders” entity with data type Boolean, which we thought it would be enough to track the order status. The “Photo” entity was deleted and included in the “Items” entity because we wanted to simplify the “Items” entity so that every item can have only one official photo. We spent more time on the relationships between items and menus since this is the focus of our database. Originally, we were thinking that having the “Item” and the “Item_on_menu” and the “Restaurants” entities would be enough to store a menu for each restaurant. However, after discussing with the professor, we realized that to create a “Menu” entity that stores the start date and the end date of a specific menu can help the database to track the menu history of certain restaurants since restaurants may have different menus over time.

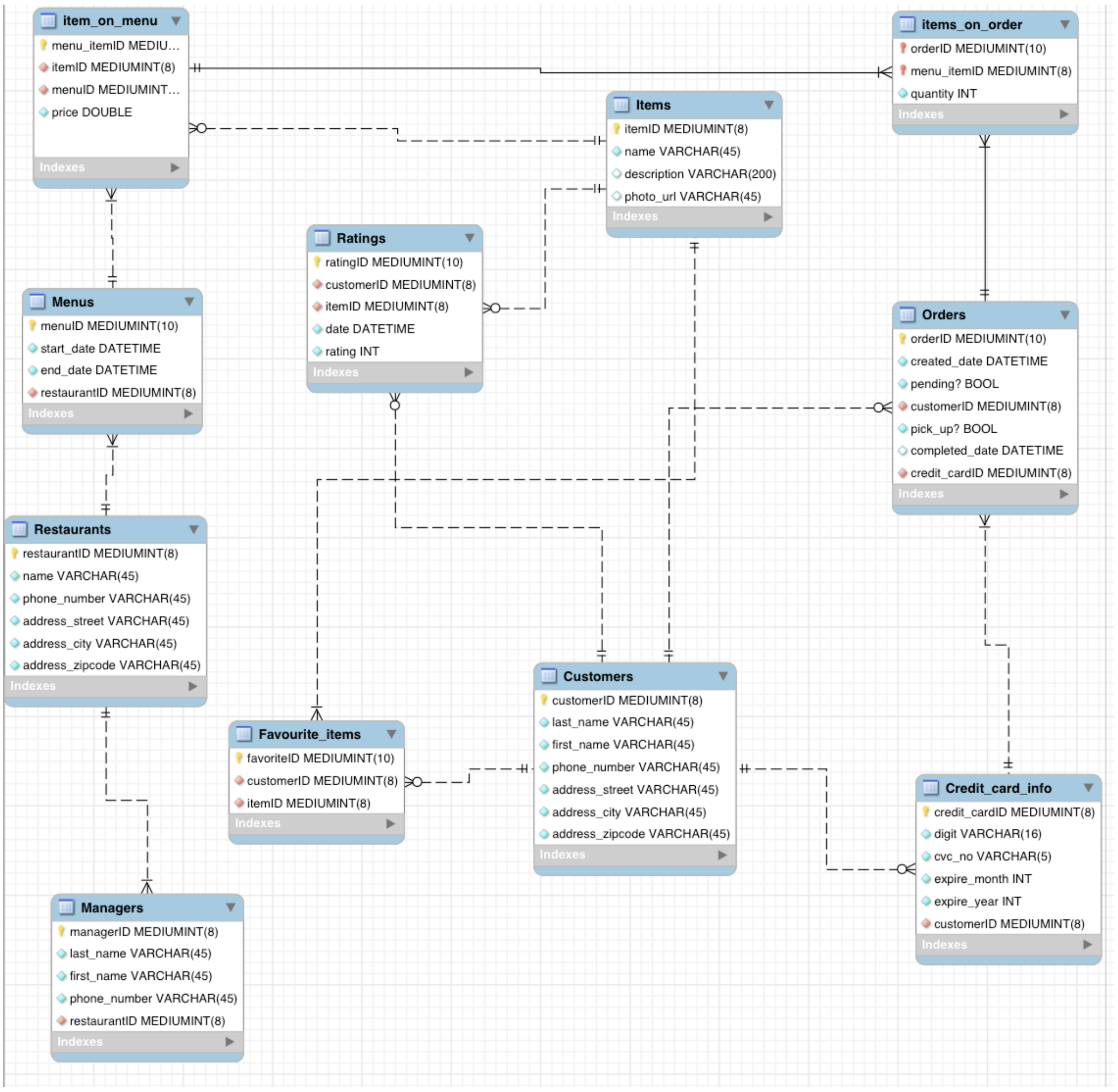

Fig .1 Final ERD(forward)

In our first version of ERD(Fig. 2), we related the "Items_on_order" entity to "Items" entity instead of "Item_on_menu" entity. However, we found a problem that this design seems making the "Items" entity useless and when a customer place an order, he/she should choose items from menus not directly from items. Therefore, we changed the relationship between "Items" and "Items_on_order", which may make more sense because customers should order items on menus that may have different prices on different menus. We forward engineered several time though we did not encounter problems in the process of forward engineering. The only difference we have between the pre and post forward engineer is that the two attributes that are Boolean data type in the ERD automatically changed to data type "tinyint(1)" in physical design, but we later found that it is the default setting of workbench.

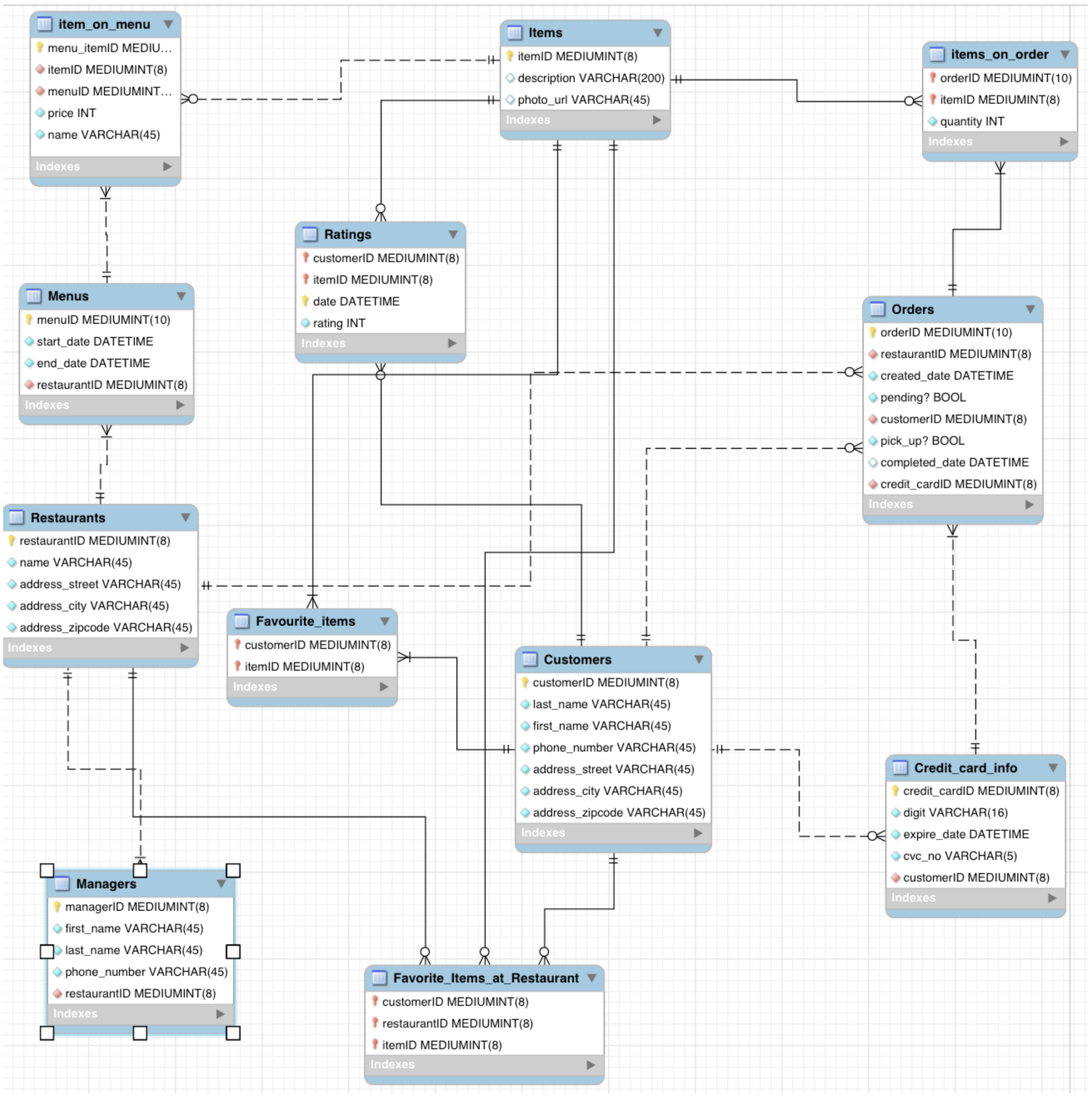

Fig. 2 First version of ERD

Physical Design

We did not encounter any problem in this phase after we finalized the ERD. However, between the logical design and physical phase, a question came to our mind. What about the expendability of the database? For example, we do not have ingredient information of items, but what if we want to include it in the future? After discussing with the professor, we learned that we may want to consider all the possible attributes when designing the database. Otherwise, we will need to redesign it from the beginning. However, we chose to not include ingredient information in our database in order to simplified it.

Sample Data and CRUD

While trying to enter sample data, we followed the normal procedure of leaving the primary key values blank so that it could be auto incremented by the server. However, to our surprise, the primary key column wasn't accepting an integer value and as a result, the unique ids could not be auto incremented. However, this problem only happened on one machine and the other machine with same version of workbench did not have the same problem. Eventually we figures out while entering data, we mentioned the unique ids as null instead of leaving it blank and that solved the issue.

We changed the scope of the sample data from what we originally proposed because we found that the physical menus we got have too many items(most of them have over 30 items). Therefore, we tried to include 5 items form each restaurants. We made up some customers and all orders, ratings, favorite items, managers, and credit card info are made up.

The lesson we learned from this phase is that when making up the sample data we need to be very careful especially for tables that have foreign keys. For example, when we made up sample data for "Items_on_order" table, despite making up the sample data we need to be careful that certain items may no be sold in certain restaurants. This made the sample data making up difficult because in our database design, it is not very intuitive to relate items to the restaurants.

Query Requirement Table

View name	Req. A	Req. B	Req. C	Req. D	Req. E
all_menus	x			x	
cheapest_soup		x			x
customer1_order_history	x	x	x	x	
favorite_as_taiwanese_hamburger	x	x		x	
find_pending_order	x	x	x	x	
hottest_item_restaurant22	x	x	x	x	
what_on_order5	x	x	x	x	
where_buy_soup		x			x
recommendation	x	x		x	

Queries explanation

1. all_menus: Search for all the menus we have in the database
2. cheapest_soup: Search for the cheapest soup sold in all menus.
3. customer1_order_history: Search for the order history of customer 1.
4. favorite_as_taiwanese_hamberger: Search for customers who saved Taiwanese Hamberger as their favorite.
5. find_pending_order: Search for orders that are pending.
6. hottest_item_restaurant22: Search for the items that are ordered most in restaurant 22.
7. what_on_order5: Search for the items on order5.
8. where_buy_soup: Search for which restaurants that sell soups.
9. recommendation: Search items rated over 3 by customers.

Future Improvement

When we were inserting the sample data, we found that our database design is too simplified that cannot appropriately store all the possible menus in the real world. In the real world, different restaurants have different kinds of menus, different categories of items, different ordering method(not only one item one price). Therefore, because our database focuses on storing menu information of all restaurants, we may need to add more tables to better fit the all the possible menus. Also, we want to include more information regarding each item, such as ingredients, which may help customers to understand what items should avoid eating or help restaurants to track the method to cook certain items.