**HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY**

**A picture containing icon

Description automatically generated**

**PROJECT REPORT**

DATABASE DESIGN AND IMPLEMENTATION

BK Food Shops

**GROUP 7**

**Đặng Ngọc Huy**

**Vũ Hồng Quang**

**Nguyễn Văn Tiến**

**Major: Global ICT – IT-E7 – K65**

**Specialization: UNDERGRADUATE**

**Supervisor:** Associate Professor Vũ Tuyết Trinh

Signature

**Department:** Computer Engineering

**School:** School of Information and Communications Technology

HANOI, 02/2023

# **Group members**

* Nguyễn Văn Tiến – 20205195
* Đặng Ngọc Huy – 20200270
* Vũ Hồng Quang - 20205192

# **Problem description**

1. **Introduction:**

In recent years, services in the food sector are increasingly developing, one of which can be mentioned is ordering food through e-commerce platforms such as Shopee Food, Gofood…. This helps commuters, students, who have little time during their lunch break or no time to cook, have quick and convenient meals. Therefore, a system containing information about the food, the food store, is necessary for both the buyer and the seller.

In this project, we want to build a database containing information about food shops in Bach Khoa area and surrounding areas with many shops, dishes, reviews of those stores and food ...

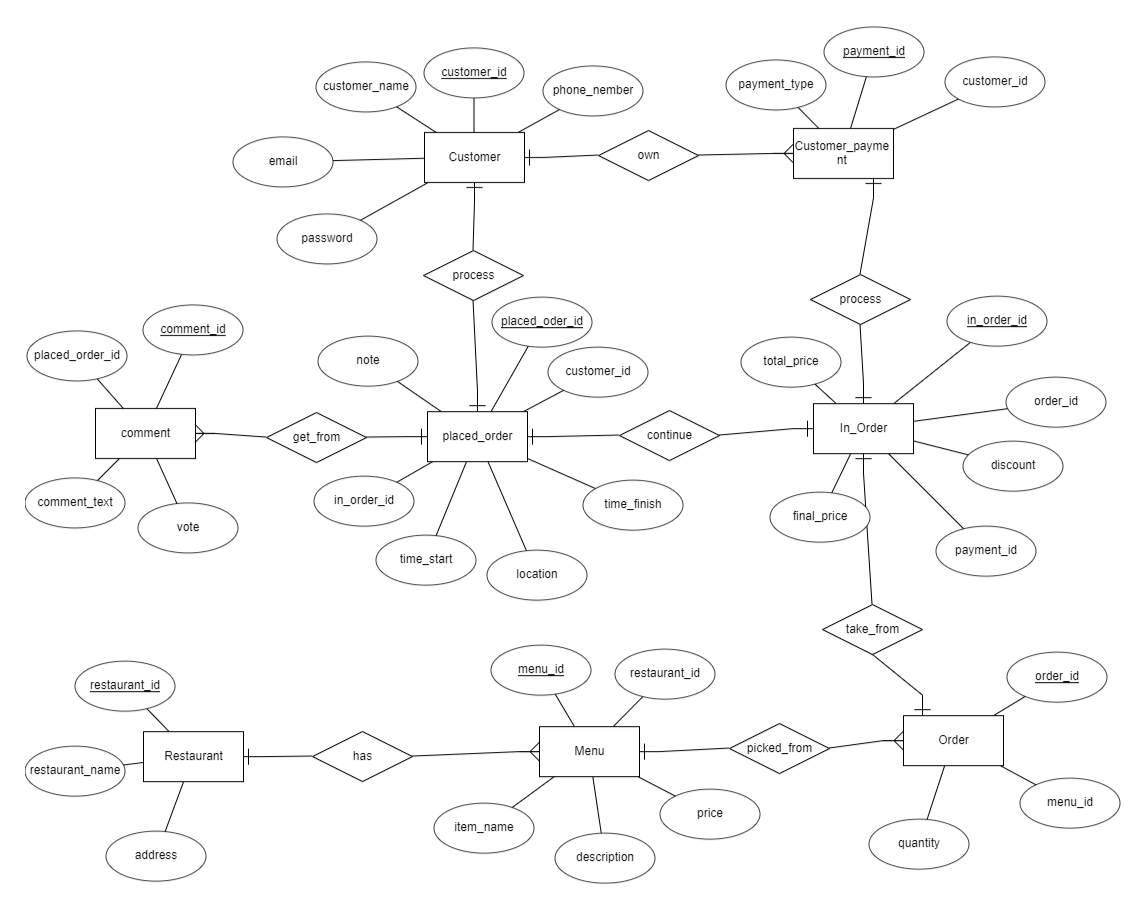
1. **Description**

The DATABASE DESIGN AND IMPLEMENTATION BK Food Shops is built on a SQL schema that consists of six tables: Customers, Customer\_payment, Restaurants, Menu, Orders, in\_order, placed\_order, and Comment. These tables store customer information, payment details, restaurant details, menu items, order details, placed order details, and customer feedback.

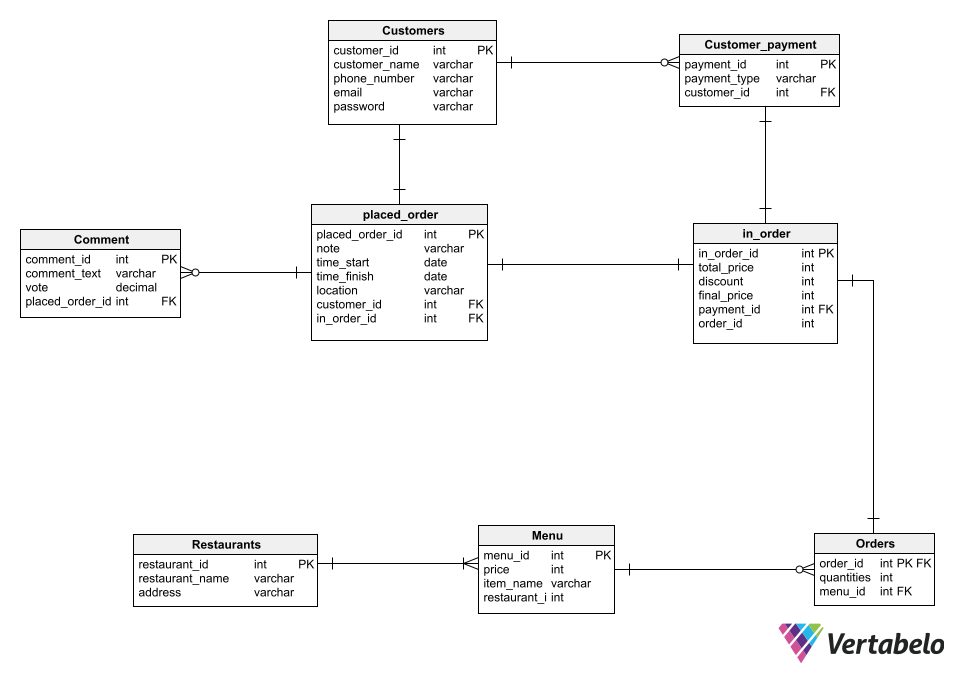
The database has the following features:

* + - Customer registration and login: Customers can register and login to the system to place their orders.
    - Menu browsing: Customers can browse the menu and select the items they want to order.
    - Order placement: Customers can place their orders and specify any notes or special requirements.
    - Payment processing: Customers can pay for their orders using various payment methods such as credit cards, debit cards, and mobile payments.
    - Order status tracking: Customers can track the status of their orders and receive updates on their order progress.
    - Customer feedback: Customers can leave feedback on their orders, which can help restaurant owners improve their services and products.
    - Real-time analytics: Restaurant owners can view real-time analytics on sales, order volume, and customer feedback.

1. **Overview database**
   * ERD:



* + Schema



* + Explain database:
    - Customers: This table stores information about the customers such as their customer ID, customer name, phone number, email, and password. The customer ID is the primary key of this table, and it is used as a foreign key in other tables to reference specific customers.
    - Customer\_payment: This table contains information about the payment methods used by customers. It includes payment ID, payment type, and customer ID. The payment ID is the primary key, and the customer ID is a foreign key that references the Customers table.
    - Restaurants: This table stores information about the restaurants, including the restaurant ID, restaurant name, and address. The restaurant ID is the primary key, and it is used as a foreign key in other tables to reference specific restaurants.
    - Menu: This table contains information about the menu items offered by the restaurants. It includes the menu ID, price, item name, and restaurant ID. The menu ID is the primary key, and the restaurant ID is a foreign key that references the Restaurants table.
    - Orders: This table stores information about the orders placed by customers. It includes the order ID, quantities, and menu ID. The order ID is the primary key, and the menu ID is a foreign key that references the Menu table.
    - in\_order: This table contains information about the orders placed by customers and the corresponding payment details. It includes the in\_order ID, total price, discount, final price, payment ID, and order ID. The in\_order ID is the primary key, and the payment ID and order ID are foreign keys that reference the Customer\_payment and Orders tables, respectively.
    - placed\_order: This table stores information about the orders that have been placed, including notes, start and finish times, location, customer ID, and in\_order ID. The placed\_order ID is the primary key, and the customer ID and in\_order ID are foreign keys that reference the Customers and in\_order tables, respectively.
    - Comment: This table contains information about the comments left by customers. It includes the comment ID, comment text, vote, and placed\_order ID. The comment ID is the primary key, and the placed\_order ID is a foreign key that references the placed\_order table.

1. **Data**

We collecting data of table ‘restaurant’ and ‘menu\_item’ in internet. About the customer info, we randomly generated. And the orders data, we created base on customers and restaurants info.

1. **Index**
2. Without index

Ảnh có chứa văn bản

Mô tả được tạo tự độngẢnh có chứa văn bản

Mô tả được tạo tự độngẢnh có chứa văn bản

Mô tả được tạo tự động

Ảnh có chứa văn bản

Mô tả được tạo tự động

1. Using BTREE

Ảnh có chứa văn bản

Mô tả được tạo tự động

Ảnh có chứa văn bản

Mô tả được tạo tự động

Ảnh có chứa văn bản

Mô tả được tạo tự động

Ảnh có chứa văn bản

Mô tả được tạo tự động

1. Using hash

Ảnh có chứa văn bản

Mô tả được tạo tự động

Ảnh có chứa văn bản

Mô tả được tạo tự độngẢnh có chứa văn bản

Mô tả được tạo tự độngẢnh có chứa văn bản

Mô tả được tạo tự động

- Comment:

* Experiment 1: Show all item which has price > 0

In all 3 cases, the DB executes the same query plan, there is no difference between using

index or not using index.

*Explain*: when filtering with small bounds, the DB still needs to traverse almost the entire

table, so using the index is not necessary.

* Experiment 2: Show all item which has price > 20000

Btree-index is effective in this case, it reduces the cost significantly. B-tree index helps the database not to scan the entire table to find the data, but just search starting at the root node, and search up to the branch and leaf, until all the data is found. query condition

* Experiment 3: Show all item which has price = 50000

Both Btree-index and hash index are used in this scenario and help improve

performance. But it seems hash index runs faster.

* Experiment 4: Show all item which has first three character is “Cơm”

Only Btree-index help improve performance, runs faster.

- Conclude:

* Should use Btree-index with operators =, <>, >, < provided that the comparison is with a large enough number.
* The optimizer can also use a B-tree index for queries involving the pattern matching operators LIKE and ~ if the pattern is a constant and is anchored to the beginning of the string.
* Hash-index is efficient when searching with the = operator, not for word math searching for a range of values like > or <.
* In other cases, the index should not be used, because the index even degrades the

performance when the database takes more time, space to save index and consider

whether the index is effective or not.

1. **Trigger/Function**
2. *customer\_trigger:*

The ‘customer\_payment\_trigger’ trigger creates a new row in the

‘Customer\_payment’ table when a new row is inserted into the ‘Customers’ table with a default payment method of "credit card", and deletes all payment methods associated with a customer when the corresponding row in the ‘Customers’ table is deleted.

Ảnh có chứa văn bản

Mô tả được tạo tự động

1. *menu\_trigger*

The ‘menu\_trigger trigger’ creates a new row in the ‘Menu’ table when a new row is inserted into the ‘Restaurants’ table with default values for price and item name, and deletes all menu items associated with a restaurant when the corresponding row in the ‘Restaurants’ table is deleted.

Ảnh có chứa văn bản

Mô tả được tạo tự động

1. *restaurant\_trigger*

The ‘restaurant\_trigger’ trigger creates a new row in the ‘Customer\_payment’ table when a new row is inserted into the ‘Restaurants’ table, and deletes the corresponding row from the ‘Customer\_payment’ table when the corresponding row in the ‘Restaurants’ table is deleted.

Ảnh có chứa văn bản

Mô tả được tạo tự động

1. *compute\_total\_price*

The ‘total\_price\_trigger’ trigger updates the ‘total\_price’ field in the ‘in\_order table’ with the total price of all items in an order when a new row is inserted into the ‘Orders’ table or when an existing row is updated.

Ảnh có chứa văn bản

Mô tả được tạo tự động

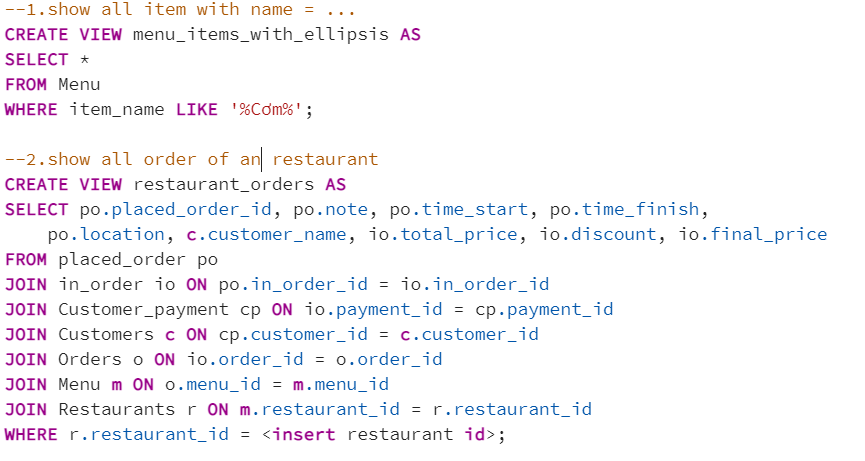
1. *compute\_final\_price*

The ‘final\_price\_trigger’ trigger calculates the final price of an order by subtracting the discount from the ‘total price’ and updates the ‘final\_price’ field in the ‘in\_order’ table when a new row is inserted into the ‘in\_order table’ or when an existing row is updated. This trigger executes before the insertion or update.

Ảnh có chứa văn bản

Mô tả được tạo tự động

1. **View**
   * The ‘menu\_items\_with\_ellipsis’ view shows all the item in ‘Menu’ table with a keyword user search
   * The ‘restaurant\_orders’ view shows all orders of a restaurant with a specific ‘restaurant\_id’.



# **User Interface:**

We can’t finish our interface, so in this report we won’t review it.

# **Lesson learned from this project**

* The design of database must follow the recommended order. When we first design the database, we made the mistake of jumping directly to the drawing of tables without careful plannings. As a result, we have to alter most of the table and the process is unnecessary complicated.
* The use of View allows shorter syntax and reduced execution time thanks to the feature of saving the query plan of VIEW.
* The use of FUNCTION will come in handy when we must execute one type of query many times.
* The use of index will help to reduce cost of query significantly in certain situations (as shown above)
* We make a very big mistake is time management. We at first do not make a clearly plan, so until now, our project is not complete!!

*This summary is the first trial of our team about the total database design and implementation. There will definitely be some mistakes or unsuitable features that need adjusting. Thus, we would really appreciate any of your comments to help us perfect our work! Thank you!*