Sinh viên thực hiện:

19126053-Dương Quang Minh

19126044-Lê Trung Hiếu

19126054-Nguyễn Gia Minh

18126020- Trần Bảo Khánh

GV phụ trách: TS THÁI LÊ VINH

HỒ THỊ HOÀNG VY

Đồ án/bài tập môn học  - CƠ SỞ DỮ LIỆU NÂNG CAO

HỌC KỲ III – NĂM HỌC 2021-2022

**môn cơ sở dữ liỆu nâng cao**

**BỘ MÔN HỆ THỐNG THÔNG TIN – KHOA CÔNG NGHỆ THÔNG TIN**

**ĐẠI HỌC KHOA HỌC TỰ NHIÊN TP.HCM, ĐẠI HỌC QUỐC GIA TP.HCM**

**BẢNG THÔNG TIN CHI TIẾT NHÓM**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Mã nhóm:** | **06** | | | |
| **Tên nhóm:** |  | | | |
| **Số lượng:** | **4** | | | |
| **MSSV** | **Họ tên** | **Email** | **Điện thoại** | **Hình ảnh** |
| 18126044 | Lê Trung Hiếu | 19126044@student.hcmus.edu.vn |  |  |
| 18126053 | Dương Quang Minh | 19126053@student.hcmus.edu.vn |  |  |
| 18126054 | Nguyễn Gia Minh | 19126054@student.hcmus.edu.vn |  |  |
| 18126020 | Trần Bảo Khánh | 18126006@student.hcmus.edu.vn |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bảng phân công & đánh giá hoàn thành công việc** | | | |  |
| **Công việc thực hiện** | **Người thực hiện** | **Mức độ hoàn thành** | **Đánh giá của nhóm** |  |
| Đặc tả đồ án |  | 100% | 9/10 |  |
| Quy trình nghiệp vụ |  | 100% | 10/10 |  |
| Toàn vẹn dữ liệu |  | 100% | 10/10 |  |
| Yêu cầu chức năng |  | 100% | 10/10 |  |
| Xác định yêu cầu dữ liệu |  | 100% | 10/10 |  |
| Layout |  | 100% | 9/10 |  |
| Xác định chỉ mục |  | 100% | 10/10 |  |
| Cài đặt chỉ mục |  | 100% | 10/10 |  |
| Phân tích |  | 100% | 10/10 |  |
| Kiểm chứng hiệu quả của chỉ mục |  | 100% | 10/10 |  |
| Partition |  | 100% | 10/10 |  |
| Chức năng | | | |  |
| Trang admin   * Xem doanh thu * Xem danh sách user trong hệ thống, tìm kiếm theo tên * Xem danh sách, thông tin, tìm kiếm các nhân viên. |  | 100% | 10/10 |  |
| Chức năng của nhân viên:   * Xem thông tin của khách hàng và chỉnh sửa * Thêm thông tin, tạo tour mới vào hệ thống |  | 100% | 10/10 |  |
| Giỏ Hàng:   * Ước tính giá tour, thay đổi số lượng hoặc xóa * Chức năng thanh toán đơn hàng. |  | 100% | 10/10 |  |
| Trang chủ:   * Xem danh sách tour trên trang chủ * Hiển thị theo danh sách , tìm kiếm theo tên tour * Đăng nhấp , đăng ký |  | 100% | 10/10 |  |
| User:   * Xem thông tin cá nhân, thay đổi thông tin cá nhân * -Xem thông tin các tour, xem danh sách các tour đã đặt, lịch sử đặt tour |  | 100% | 10/10 |  |

**YÊU CẦU ĐỒ ÁN- BÀI TẬP**

|  |  |
| --- | --- |
| **Loại bài tập** | **🗹 Lý thuyết Thực hành Đồ án 🗹 Bài tập** |
| **Ngày bắt đầu** |  |
| **Ngày kết thúc** |  |

Contents

**[Yêu cầu của Đồ án/Bài tập](#_Toc80858561)** [4](#_Toc80858561)

**[1.](#_Toc80858562)****[Mô Tả:](#_Toc80858562)** [4](#_Toc80858562)

[a.Quy trình nghiệp vụ 4](#_Toc80858563)

*[1)Quy trình cho khách hàng:](#_Toc80858564)* [4](#_Toc80858564)

*[2)Quy trình cho nhân viên (Bao gồm cả admin lẫn nhân viên thông thường):](#_Toc80858565)* [4](#_Toc80858565)

*[3)Quy trình thanh toán:](#_Toc80858567)* [4](#_Toc80858567)

b.Danh sách chức năng

**[2.](#_Toc80858569)****[Thiết kế dữ liệu quan niệm](#_Toc80858569)** [4](#_Toc80858569)

[a. Đặc tả hệ thống 4](#_Toc80858570)

[b. Ràng buộc dữ liệu 6](#_Toc80858571)

**[3. Thiết kế dữ liệu logic](#_Toc80858572)** [11](#_Toc80858572)

[a. Mô hình ER 11](#_Toc80858573)

[b. Mô hình quan hệ 12](#_Toc80858574)

[c. Chuẩn hóa dữ liệu: 13](#_Toc80858575)

**[4. Thiết kế vật lý (thông tin tần suất, cài chỉ mục, phân tích, kiểm chứng hiệu quả chỉ mục…)](#_Toc80858576)** [16](#_Toc80858576)

[a. Thông tin tần suất 16](#_Toc80858577)

[b. Chỉ mục 16](#_Toc80858578)

[c. Phân tích 18](#_Toc80858579)

[d. Kiểm chứng hiệu quả chỉ mục 19](#_Toc80858580)

[e. Trigger và Procedure 36](#_Toc80858581)

**[5. Xác định yêu cầu chức năng](#_Toc80858582)**[: 42](#_Toc80858582)

**[6.](#_Toc80858583)****[Layout của giao diện:](#_Toc80858583)** [45](#_Toc80858583)

**[7.](#_Toc80858584)****[Partiton](#_Toc80858584)** [56](#_Toc80858584)

**[I.](#_Toc80858585)****[Partition:](#_Toc80858585)** [56](#_Toc80858585)

**[II.](#_Toc80858586)****[Lợi ích của việc sử dụng Partition:](#_Toc80858586)** [57](#_Toc80858586)

**[III.](#_Toc80858587)****[Các kiểu phân chia dữ liệu:](#_Toc80858587)** [57](#_Toc80858587)

**[IV.](#_Toc80858588)****[Phân chia dữ liệu bảng Order (Hoá đơn) theo chiều ngang](#_Toc80858588)** [57](#_Toc80858588)

**Mô tả**

**A/Quy trình nghiệp vụ**

1. *Quy trình cho người mua:*

* Đăng nhập tài khoản.
* Tìm kiếm, tham khảo thông tin tour, dịch vụ, khuyến mại.
* Người Mua đưa ra quyết định đặt tour bằng cách click vào “Đặt hàng”.
* Người Mua lựa chọn cách thức giao dịch tuy theo chính sách bên bán.
* Đơn hàng của Người Mua sẽ được chuyển thông tin đến bên bán.
* Người Mua nhận được thông tin xác nhận về đơn hàng
* Người Mua chỉnh sửa đơn hàng.

1. *Quy trình cho bên cung cấp tour:*

* Đăng nhập tài khoản
* Tạo tài khoản (chỉ dành cho nhân viên có quyền admin)
* Tìm kiếm, xem thông tin, cập nhật, tạo các tour
* Quản lý đơn hàng
* Sửa bảng trạng thái của tour
* Thống kê thông tin
* Gửi thông báo đến người mua
  1. *Quy trình thanh toán:*

**B/Danh sách yêu cầu chức năng**

* Đăng nhập/Đăng ký user
* Tìm lại password
* Hiển thị sản phẩm
* Thêm sản phẩm vào giỏ hàng
* Thêm/xóa/ cập nhật thông tin sản phẩm
* Order sản phẩm
* Chỉnh sửa thông tin user
* Sắp xếp hiển thị sản phẩm theo loại sản phẩm

1. Yêu cầu phi chức năng

* Toàn vẹn dữ liệu
* Khả năng tương tác
* Khả năng sử dụng
* Khả năng quản lí
* Khả năng bảo trì

**Thiết kế dữ liệu quan niệm**

**A/.Đặc tả hệ thống**

Công ty MHMK mở một trang buôn bán, quản lý các tour du lịch. Công ty cần xây dựng 1 trang web cho phép công ty và các cá nhân thực hiện việc mua bán các tour.

Sau đây là kết quả của việc phân tích yêu cầu ứng dụng:

* Trang MHMKTour bán đủ loại tour với các tour do công ty tự bán và quản lý.
* Một **nhân viên**  có: *mã nhân viên, tên tài khoản, tên nhân viên*, với nhân viên không phải là admin, thì có thêm thông tin về Admin quản lý nhân viên đó. Với nhân viên có cấp bậc là admin có quyền tạo các tài khoản nhân viên và quy định cấp bậc cho họ. Với các nhân viên còn lại có quyền thay đổi, cập nhật, tạo các tour.
* Cần lưu trữ **thông tin khách hàng**, gồm : *mã khách hàng, tên khách hàng, số điện thoại,địa chỉ, email .*Mỗi khách hàng chỉ được sử dụng 1 số điện thoại duy nhất để đăng ký.
* Cần lưu lại thông tin về tất cả các tour mà công ty MHMK cung cấp gồm: *Mã tour, ngày bắt đầu, ngày kết thúc, mô tả, nơi bắt đầu, nơi tham quan, phương tiện di chuyển, giá thành*. Mỗi tour có một nơi bắt đầu tour gồm : *mã địa điểm, tên nơi bắt đầu*. Ngoài ra, các tour còn có nơi tham quan gồm: *mã địa điểm, tên nơi tham quan*. .
* Khi khách hàng chọn tour, thì sản phẩm sẽ được đưa vào **giỏ hàng** của khách hàng đó, có: Mã đơn, Mã khách hàng và Tổng tiền, bên cạnh đó, các các thông tin chi tiết của đơn hàng sẽ được lưu riêng biệt bao gồm: Mã tour, số lượng người lớn, số lượng trẻ em. Ngay sau khi khách hàng đặt mua thì trạng thái của đơn hàng sẽ được chuyển thành *chưa thanh toán.* Sau khi nhận được thông tin đơn hàng, nhân viên sẽ căn cứ vào số điện thoại của khách hàng được lưu trên thông tin thanh toán sẽ gọi để trao đổi thêm thông tin với khách hàng và thông tin tới khách hàng về việc thanh toán. Khi xác nhận thông tin thanh toán của khách hàng, nhân viên sẽ chuyển trạng thái của đơn hàng thành *đã thanh toán,* Tuy nhiên, nếu như khách hàng hủy hoặc quá thời hạn giữ đơn hàng mà khách hàng chưa thanh toán, đơn sẽ được hủy, nhưng để đảm bảo việc phục hồi đơn khi cần thiết, đơn hàng sẽ không được xóa hoàn toàn mà sẽ được đưa ra khỏi các tiến trình xử lý.
* MHMKTour cần lưu lại các thông tin giao dịch vào **nhật kí giao dịch** và lưu trữ theo: *mã giao dịch, mã đơn hàng, ngày lập giao dịch.* Tuy nhiên, chỉ các đơn hàng có trạng thái là *đã thanh toán* thì mới được đưa vào nhật ký giao dịch.

**B/.Ràng buộc dữ liệu**

* **Admin:**

|  |  |  |
| --- | --- | --- |
| **Mô tả** |  | |
| **Dữ liệu** | **Kiểu dữ liệu** | **Ràng buộc** |
| IdAdmin | int | PRIMARY Key |
| Username | nvarchar(50) | NOT NULL |
| Password | nvarchar(50) | NOT NULL |

* **Staff:**

|  |  |  |
| --- | --- | --- |
| **Mô tả** |  | |
| **Dữ liệu** | **Kiểu dữ liệu** | **Ràng buộc** |
| IdStaff | int | PRIMARY Key |
| IdAdmin | int | FOREIGN KEY |
| Fullname | nvarchar(50) | NOT NULL |
| Username | nvarchar(50) | NOT NULL |
| Password | nvarchar(50) | NOT NULL |

* **Client:**

|  |  |  |
| --- | --- | --- |
| **Mô tả** |  | |
| **Dữ liệu** | **Kiểu dữ liệu** | **Ràng buộc** |
| IdClient | int | PRIMARY Key |
| Username | nvarchar(32) | NOT NULL |
| Password | nvarchar(32) | NOT NULL |
| Fullname | nvarchar(10) | NOT NULL |
| Phone | nvarchar(10) | UNIQUE |
| Address | nvarchar(50) |  |
| Email | nvarchar(50) |  |

* **Tour:**

|  |  |  |
| --- | --- | --- |
| **Mô tả** |  | |
| **Dữ liệu** | **Kiểu dữ liệu** | **Ràng buộc** |
| IdTour | int | PRIMARY Key |
| TourName | nvarchar(50) | UNIQUE |
| DayStart | Date | NOT NULL |
| DayEnd | Date | NOT NULL |
| Description | nvarchar(10) | NOT NULL |
| LocationStart | nvarchar(10) | FOREIGN KEY |
| Vehicle | nvarchar(50) | NOT NULL |
| Cost | float | NOT NULL |

* **OrderTour:**

|  |  |  |
| --- | --- | --- |
| **Mô tả** |  | |
| **Dữ liệu** | **Kiểu dữ liệu** | **Ràng buộc** |
| IdOrder | int | PRIMARY Key |
| IdClient | int | FOREIGN Key |
| Total | float | NOT NULL |

* **OrderDetail:**

|  |  |  |
| --- | --- | --- |
| **Mô tả** |  | |
| **Dữ liệu** | **Kiểu dữ liệu** | **Ràng buộc** |
| id | **int** | PRIMARY KEY |
| IdOrder | int | FOREIGN KEY |
| IdTour |  | FOREIGN KEY |
| Adults | int | NOT NULL |
| Childrens | int | NOT NULL |

* **LocationVisit:**

|  |  |  |
| --- | --- | --- |
| **Mô tả** |  | |
| **Dữ liệu** | **Kiểu dữ liệu** | **Ràng buộc** |
| id | int | PRIMARY Key |
| VisitPlace | nvarchar | NOT NULL |
| idTour | char | FOREIGNKEY |

* **OrderStatus:**

|  |  |  |
| --- | --- | --- |
| **Mô tả** |  | |
| **Dữ liệu** | **Kiểu dữ liệu** | **Ràng buộc** |
| Id | int | PRIMARY KEY |
| IdOrder | int | FOREIGN KEY |
| Status | char | NOT NULL |

* **Trash:**

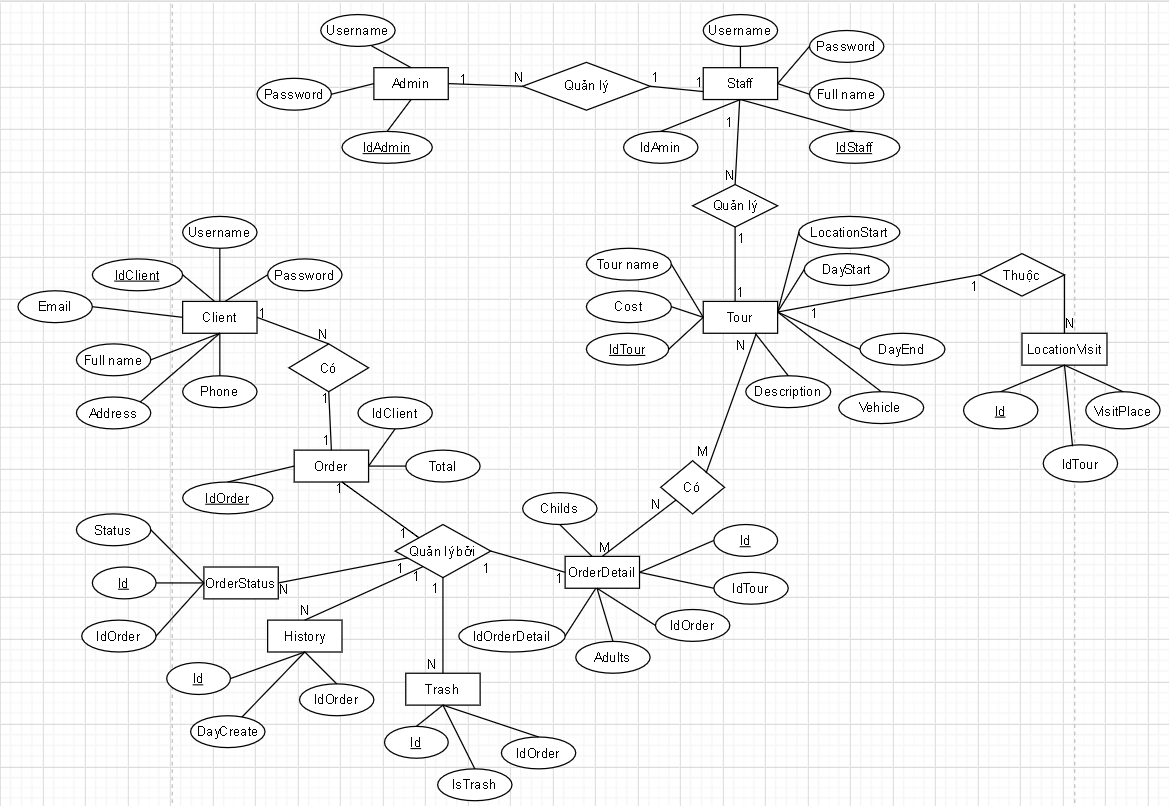
|  |  |  |
| --- | --- | --- |
| **Mô tả** |  | |
| **Dữ liệu** | **Kiểu dữ liệu** | **Ràng buộc** |
| Id | int | PRIMARY KEY |
| IdOrder | int | FOREIGN KEY |
| IsTrash | boolean | NOT NULL |

* **History:**

|  |  |  |
| --- | --- | --- |
| **Mô tả** |  |  |
| **Dữ liệu** | **Kiểu dữ liệu** | **Ràng buộc** |
| Id | int | PRIMARY KEY |
| IdOrder | int | FOREIGN KEY |
| CreateDate | DateTime | NOT NULL |

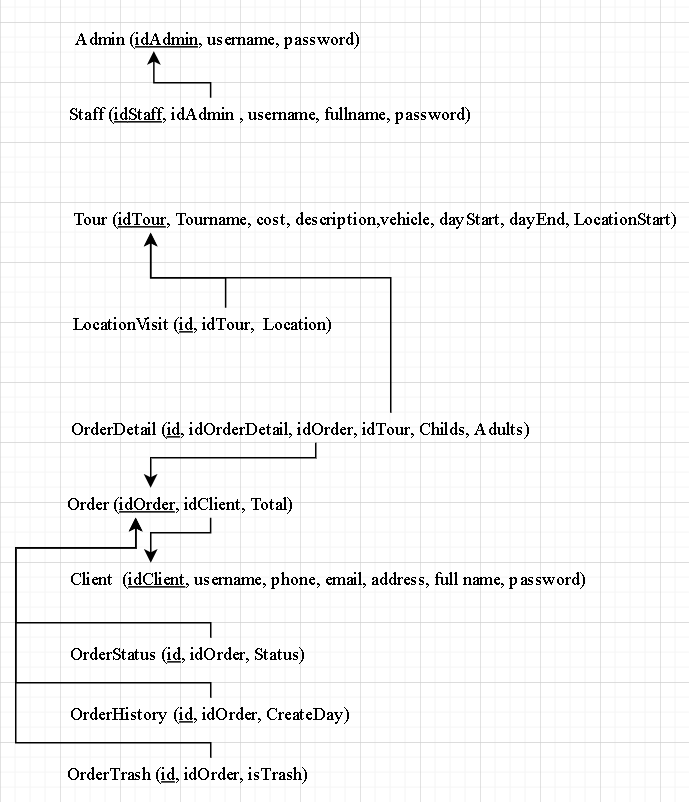
**Thiết kế dữ liệu logic**

1.Mô hình ER



Mô hình ER được xây dựng dựa trên cách thức vết dầu loang, với trung tâm là Tour.

1. Mô hình quan hệ



3.Chuẩn hóa dữ liệu:

* Client

Client  (idClient, username, phone, email, address, full name, password) không thuộc dạng chuẩn.

Trong bảng này có một trường là thuộc tính đa trị : address

* + Bảng mới sau khi nâng chuẩn sẽ là

Client (idClient, username, phone, email, full name, password)

Address (id, idClient, number, street, wards, district, city )

* + Bảng Customer sẽ ở dạng chuẩn 3NF :
    - Là 2NF
    - Không có phụ thuộc hàm bắc cầu
* Tour

Tour (idTour, Tourname, cost, description,vehicle, dayStart, dayEnd, LocationStart)

Trong bảng này có hai trường là thuộc tính đa trị : Vehicle, LocationStart

* + Bảng mới sau khi nâng chuẩn sẽ là :

Tour (idTour, Tourname, cost, description,IdVehicle, dayStart, dayEnd)

LocationStart(id, idTour, Location)

Vehicle(id, idTour, Vehicle, Capacity)

* + Bảng Customer sẽ ở dạng chuẩn 3NF :
    - Là 2NF
    - Không có phụ thuộc hàm bắc cầu
* Admin

Admin (idAdmin, username, password)

* + Bảng Admin sẽ ở dạng chuẩn 3NF:
    - Là 2NF
    - Không có phụ thuộc hàm bắc cầu
* Staff

Staff (idStaff, username, fullname, password)

* + Bảng Staff sẽ ở dạng chuẩn 3NF:
    - Là 2NF
    - Không có phụ thuộc hàm bắc cầu
* LocationVisit

LocationVisit (id, idTour, Location)

* + Bảng Location sẽ ở dạng chuẩn 3NF:
    - Là 2NF
    - Không có phụ thuộc hàm bắc cầu
* Order

Order (idOrder, idClient, Total)

* + Bảng Order sẽ ở dạng chuẩn 3NF:
    - Là 2NF
    - Không có phụ thuộc hàm bắc cầu
* OrderDetail

OrderDetail (id, idOrderDetail, idOrder, idTour, Childs, Adults)

* + Bảng OrderDetail sẽ ở dạng chuẩn 3NF:
    - Là 2NF
    - Không có phụ thuộc hàm bắc cầu
* OrderStatus

OrderStatus (id, idOrder, Status)

* + Bảng OrderStatus sẽ ở dạng chuẩn 3NF:
    - Là 2NF
    - Không có phụ thuộc hàm bắc cầu
* OrderHistory

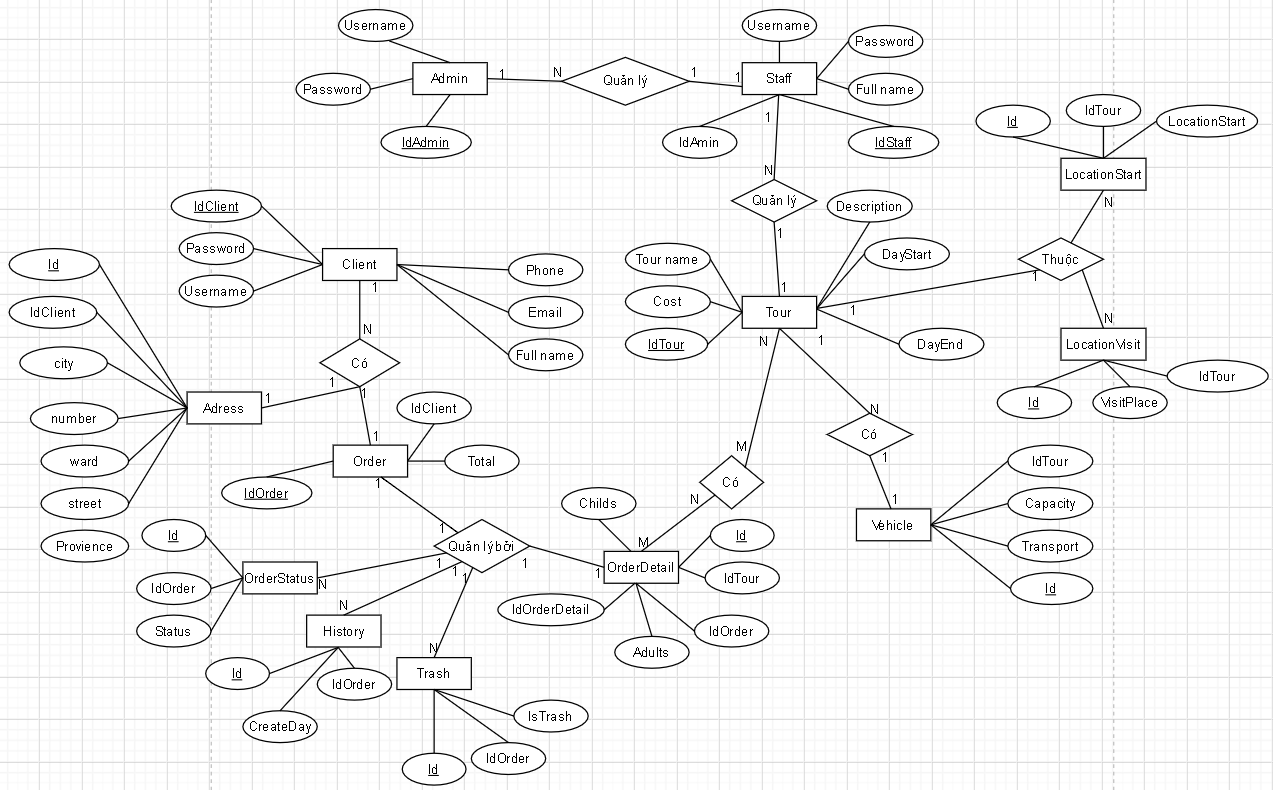
OrderHistory (id, idOrder, CreateDay)

* + Bảng Admin sẽ ở dạng chuẩn 3NF:
    - Là 2NF
    - Không có phụ thuộc hàm bắc cầu
* OrderTrash

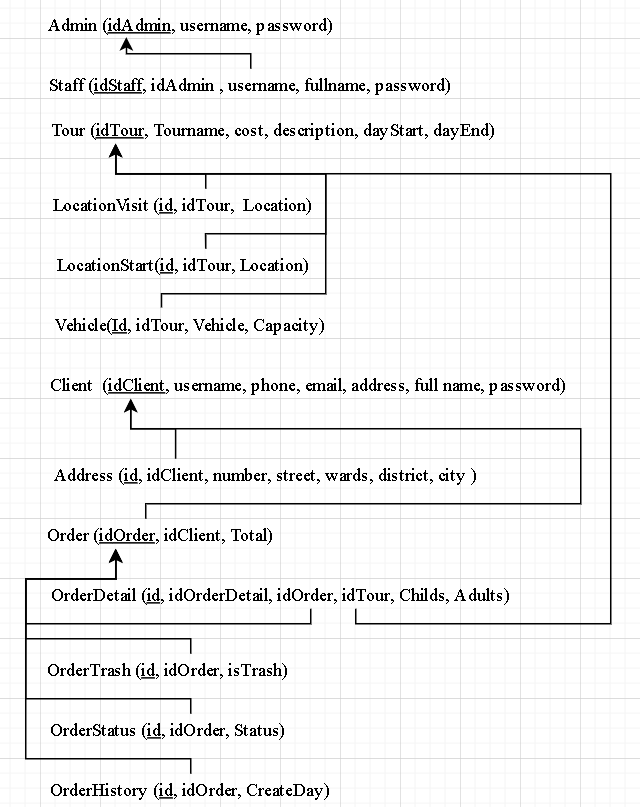
OrderTrash (id, idOrder, isTrash)

* + Bảng Admin sẽ ở dạng chuẩn 3NF:
    - Là 2NF
    - Không có phụ thuộc hàm bắc cầu

1. ERD, lược đồ quan hệ và miêu tả các trường dữ liệu sau khi được chuẩn hóa.
2. Mô hình ERD



1. Lược đồ quan hệ



1. Mô tả các trường dữ liệu bổ sung

* **Tour:**

|  |  |  |
| --- | --- | --- |
| **Mô tả** |  | |
| **Dữ liệu** | **Kiểu dữ liệu** | **Ràng buộc** |
| IdTour | int | PRIMARY Key |
| TourName | nvarchar(50) | UNIQUE |
| DayStart | Date | NOT NULL |
| DayEnd | Date | NOT NULL |
| Description | nvarchar(10) | NOT NULL |
| Cost | float | NOT NULL |

* **LocationStart:**

|  |  |  |
| --- | --- | --- |
| **Mô tả** |  | |
| **Dữ liệu** | **Kiểu dữ liệu** | **Ràng buộc** |
| IdLocation | int | PRIMARY Key |
| Provience | nvarchar | NOT NULL |
| Tour | char | FOREIGNKEY |

* **Vehicle:**

|  |  |  |
| --- | --- | --- |
| **Mô tả** |  | |
| **Dữ liệu** | **Kiểu dữ liệu** | **Ràng buộc** |
| Id | char | PRIMARY Key |
| Vehicle | char | NOT NULL |
| Capacity | int | NOT NULL |
| Tour | char | FOREIGNKEY |

* **Client:**

|  |  |  |
| --- | --- | --- |
| **Mô tả** |  | |
| **Dữ liệu** | **Kiểu dữ liệu** | **Ràng buộc** |
| IdClient | int | PRIMARY Key |
| Username | nvarchar(32) | UNIQUE |
| Password | nvarchar(32) | NOT NULL |
| Fullname | nvarchar(10) | NOT NULL |
| Phone | nvarchar(10) | UNIQUE |
| Email | nvarchar(50) |  |

* **Address**

|  |  |  |
| --- | --- | --- |
| **Mô tả** |  | |
| **Dữ liệu** | **Kiểu dữ liệu** | **Ràng buộc** |
| id | int | PRIMARY Key |
| username | int | FOREIGNKEY |
| number | int | NOT NULL |
| street | Nvarchar(32) | NOTNULL |
| ward | Nvarchar(32) | NOTNULL |
| city | Nvarchar(32) | NOTNULL |
| provience | Nvarchar(32) | NOTNULL |

**Thiết kế vật lý**

**(thông tin tần suất, cài chỉ mục, phân tích, kiểm chứng hiệu quả chỉ mục…)**

1. Thông tin tần suất

|  |  |  |
| --- | --- | --- |
| **Chức năng** | **Mục tiêu** | **Tuần suất (lượt truy vấn)** |
| Tìm kiếm thông tin khách hàng | Tìm theo tên hoặc số điện thoại của khách hàng | 1M+/ ngày |
| Tìm kiếm thông tin nhân viên (Staff) | Tìm theo tên hoặc tên tài khoản | 10K/ngày |
| Tìm kiếm thông tin nhân viên (Admin) | Tìm theo tên tài khoản | 10K/ngày |
| Tìm kiếm thông tin tour | Tìm theo tên tour hoặc tìm theo những nơi tham quan | 10M+/ ngày |
| Hiển thị thông tin giỏ hàng | Hiển thị giỏ hàng theo mã khách hàng | 500k/ ngày |
| Hiển thị đơn hàng | Hiển thị theo mã khách hàng hoặc ngày tạo order | 500k/ngày |
| Hiển thị lịch sử giao dịch của KH | Tìm theo mã khách hàng và ngày tạo log | 500K/ ngày |
| Hiển thị trạng thái của các đơn hàng | Tìm theo tình trạng đơn hàng | 500K/ ngày |
| Hiển thị doanh thu mà ncc thu được | Tìm kiếm theo ngày tạo và mã Order | 500K/ ngày |
| Hiển thị mức chi mà ncc phải chi | Tìm kiếm theo mã Order, ngày tạo, thông tin, trạng thái tour | 500K/ ngày |

1. Chỉ mục

-- Index cho Admin

CREATE INDEX TimKiemAdmin ON Admin(Username ASC)

-- Index cho nhân viên

CREATE NONCLUSTERED INDEX TimKiemNhanVien ON Staff(

Fullname ASC,

Username ASC)

--Index cho khách hàng

CREATE NONCLUSTERED INDEX TimKiemKhachHang ON Client

( Fullname ASC,

Username ASC,

Phone ASC,

Email ASC)

-- Index cho danh sách Tour

CREATE NONCLUSTERED INDEX TimKiemTour ON Tour (TourName ASC, DayStart ASC, DayEnd ASC)

-- Index cho nơi bắt đầu

CREATE INDEX TimKiemLocationStart\_byLocation ON LocationStart (Province ASC)

CREATE INDEX TimKiemLocationStart\_byTour ON LocationStart (Tour ASC)

-- Index cho nơi tham quan

CREATE INDEX TimKiemLocationVisit\_byLocation ON LocationVisit (VisitPlace ASC)

CREATE INDEX TimKiemLocationVisit\_byTour ON LocationVisit (Tour ASC)

-- Index cho phương tiện

CREATE INDEX TimKiemPhuongTien ON Vehicle (Transport)

-- Index cho đơn hàng

CREATE NONCLUSTERED INDEX TimKiemOrder ON OrderTour(IdClient ASC, createDate ASC)

-- Index cho trang thái đơn hàng

CREATE NONCLUSTERED INDEX TimKiemOrderStatus ON OrderStatus (IdOrder ASC, OrderStatus ASC)

1. Phân tích
   * + Tìm kiếm thông tin nhân viên theo tên hoặc username: do tần suất tìm kiếm nhân viên để thực hiện công việc quản lý, nên ta cài index để cải thiện performance cho hệ thống
     + Tìm kiếm thông tin khách hàng theo tên, username hoặc số điện thoại: do tần suất tìm kiếm khách hàng để tạo order hoặc xem thông tin là cao, nên ta cài index để cải thiện performance cho hệ thống.
     + Tìm kiếm tour là chức năng có lượt truy vấn nhiều nhất trên hệ thống, vì vậy phải cải thiện performance cho việc truy vấn liên quan đến sản phẩm
     + Bên cạnh việc tìm kiếm tour dựa trên TourName, người dùng còn có thể tìm kiếm dựa trên phương tiện, nơi bắt đầu, nơi tham quan vì thế ta cài index ở những bảng Vehicle, LocationStart, LocationVisit để cải thiện performance cho việc truy vấn liên quan đến thông tin tour
     + Hiển thị đơn hàng: hiển thị theo id khách hàng hoặc ngày tạo order dùng để kiểm soát thông tin order của khách trong việc thanh toán hoặc kiểm tra doanh thu, lịch sử giao dịch của khách hàng.
     + Các bảng kiểm tra lịch sử giao dịch được admin và nhà cung cấp xem lại hàng ngày/tháng/năm để kiểm soát doanh thu và cải thiện việc bán hàng. Vì vậy cần cài đặt index ở các bảng này để việc truy suất dữ liệu được nhanh hơn (do dữ liệu lưu trữ rất lớn).
2. Kiểm chứng hiệu quả chỉ mục

Kích thước dữ liệu dùng kể kiểm chứng hiệu quả của chỉ mục

1. Admin : 1.000 dòng dữ liệu
2. Nhân viên (Staff): 150.000 nhân viên

3. Khách hàng : 500.000 khách hàng

4. Tour : 1.000.000 sản phẩm

5. Vehicle: 1.500.000 phương tiện

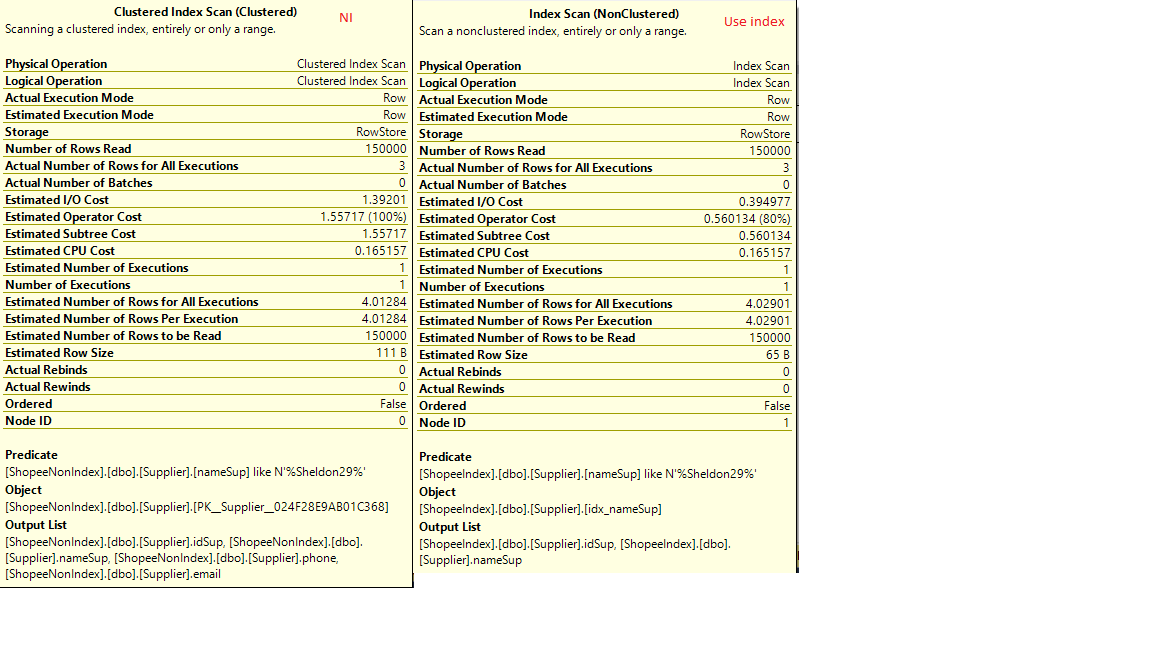
6. LocationStart: 1.000.000 vị trí

7. LocationVisit: 4.000.000 vị trí

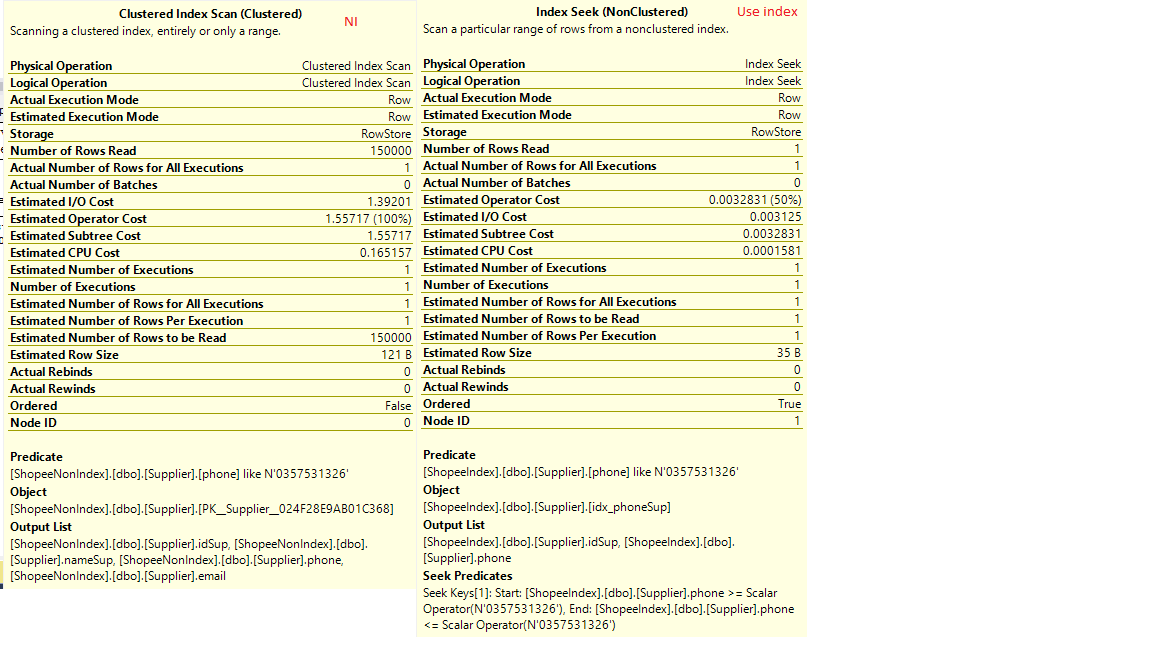
8. Order: 800.000 đơn hàng

9. OrderStatus: 800.000 đơn hàng

* Nhà cung cấp

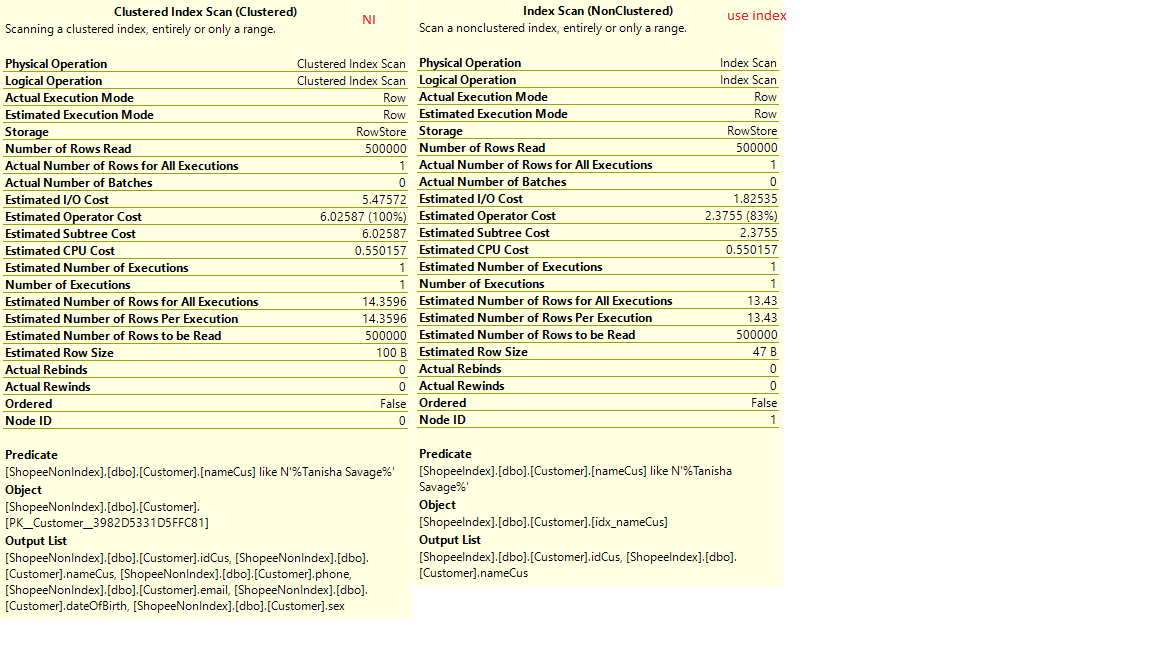
select \* from Supplier where nameSup like '%Sheldon29%'

select \* from Supplier where phone like '0357531326'

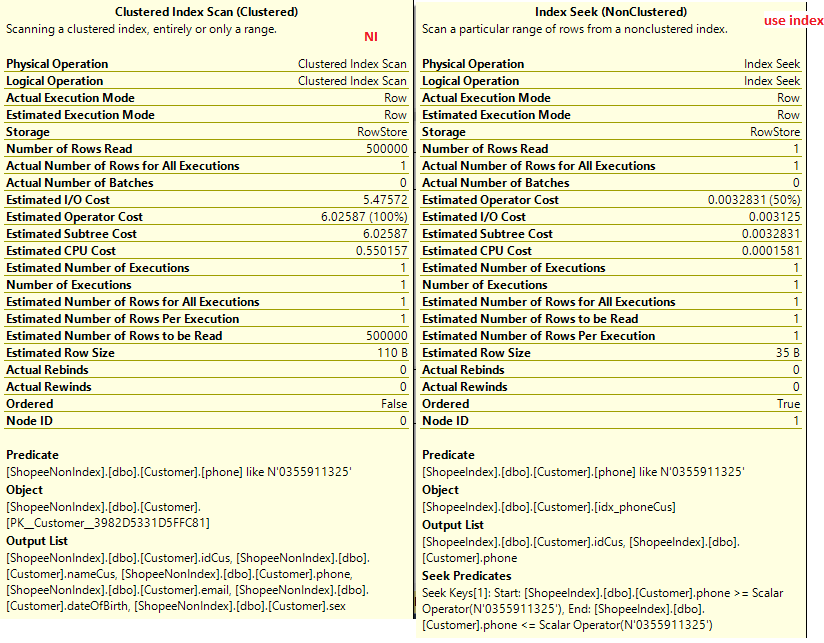


* Khách hàng

select \* from Customer where nameCus like '%Tanisha Savage%'

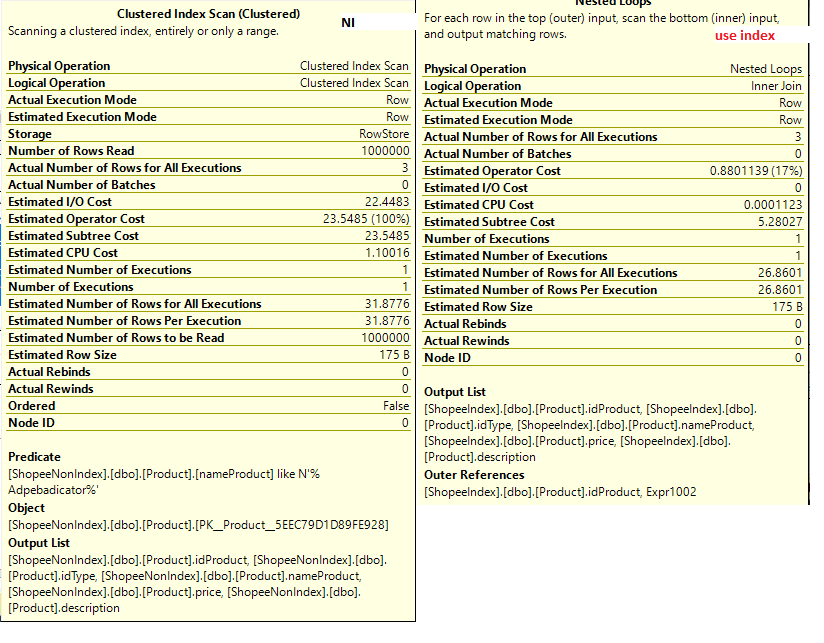


select \* from Customer where phone like '0355911325'



* Sản phẩm:

select \* from Product where nameProduct like '%Adpebadicator%'

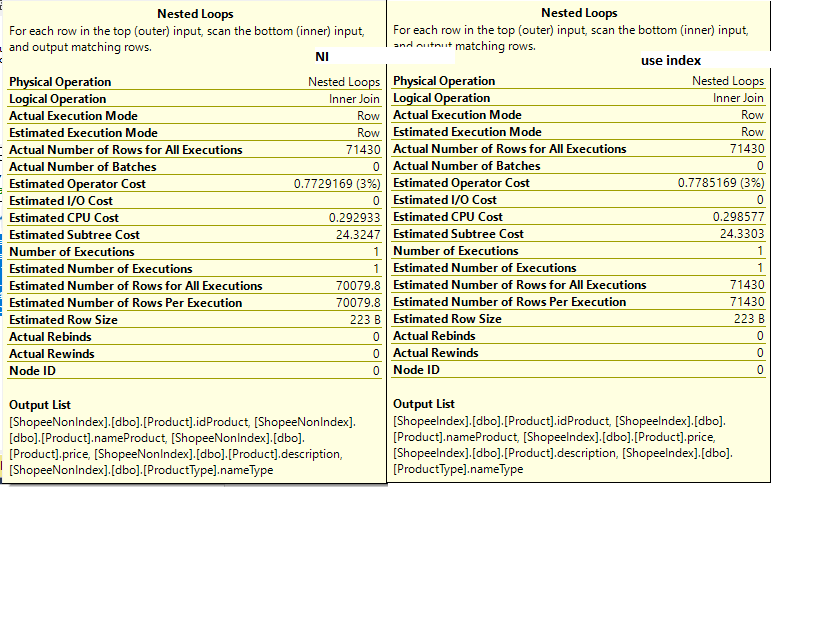


select pt.nameType,p.idProduct, p.nameProduct,p.price,p.description from Product p

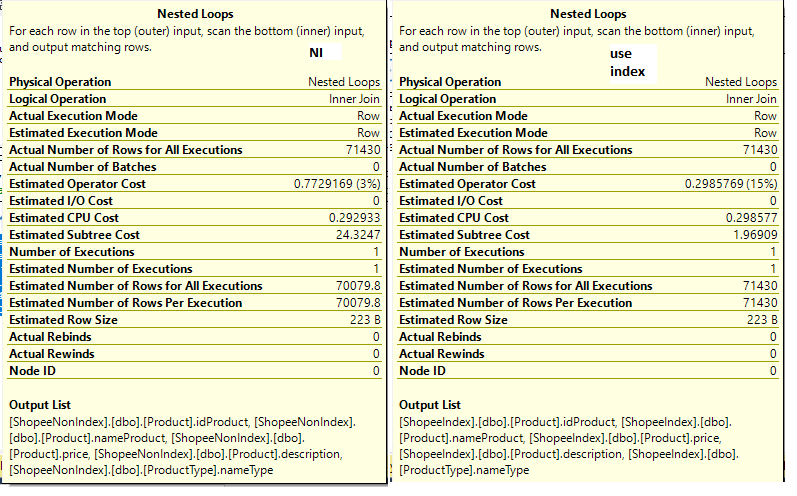
inner join ProductType pt

on p.idType = pt.idType

where p.idType = 7



Index recommend



* Kiểm tra sự thay đổi về giá sản phẩm

select s.idSup,s.nameSup,p.nameProduct,pl.old\_price,pl.new\_price,pl.date\_update

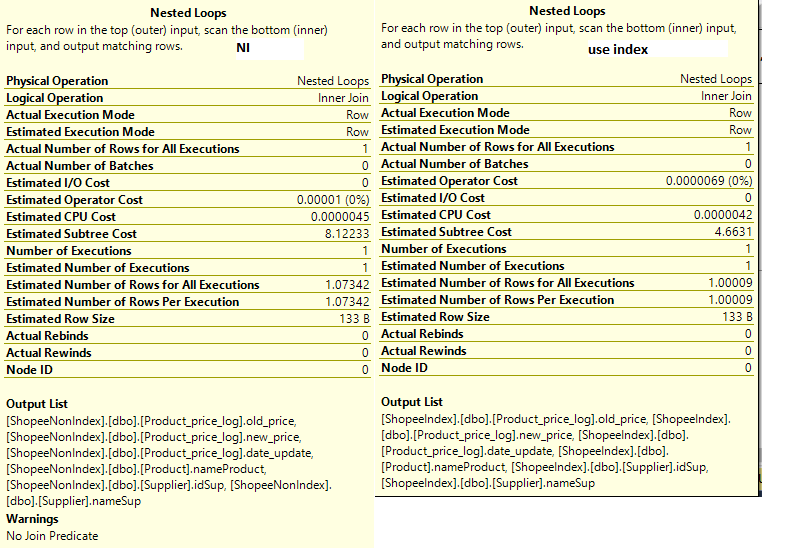
from Product\_price\_log pl

inner join Product p on p.idProduct = pl.idProduct

inner join Provide on Provide.idProduct = pl.idProduct

inner join Supplier s on s.idSup = Provide.idSup

where pl.idProduct = 620075



* Hiển thị đánh giá sản phẩm

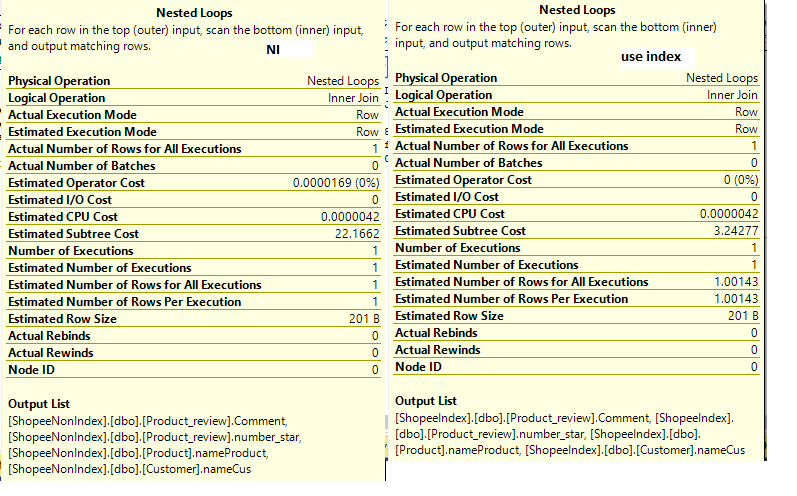
select p.nameProduct,c.nameCus, pr.Comment,pr.number\_star

from Product\_review pr

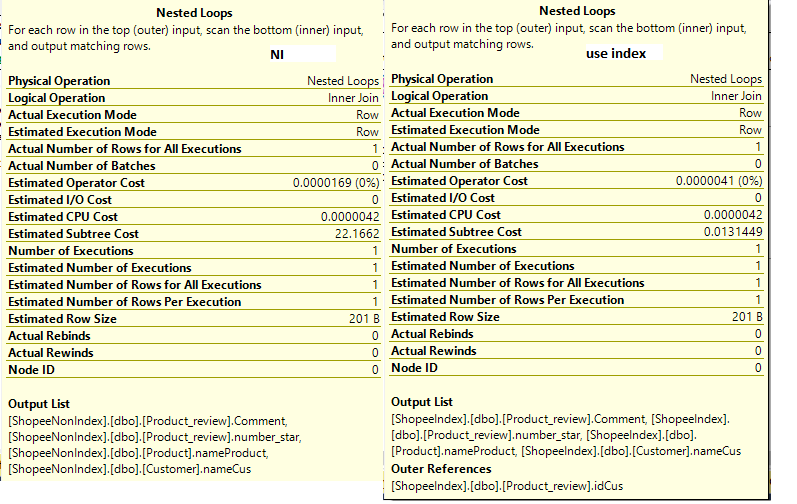
inner join Product p on p.idProduct = pr.idProduct

inner join Customer c on c.idCus = pr.idCus

where pr.idProduct = 164146



Recommend index



* Hiển thị thông tin giỏ hàng

select cus.nameCus, p.nameProduct, ca.amount from Cart ca

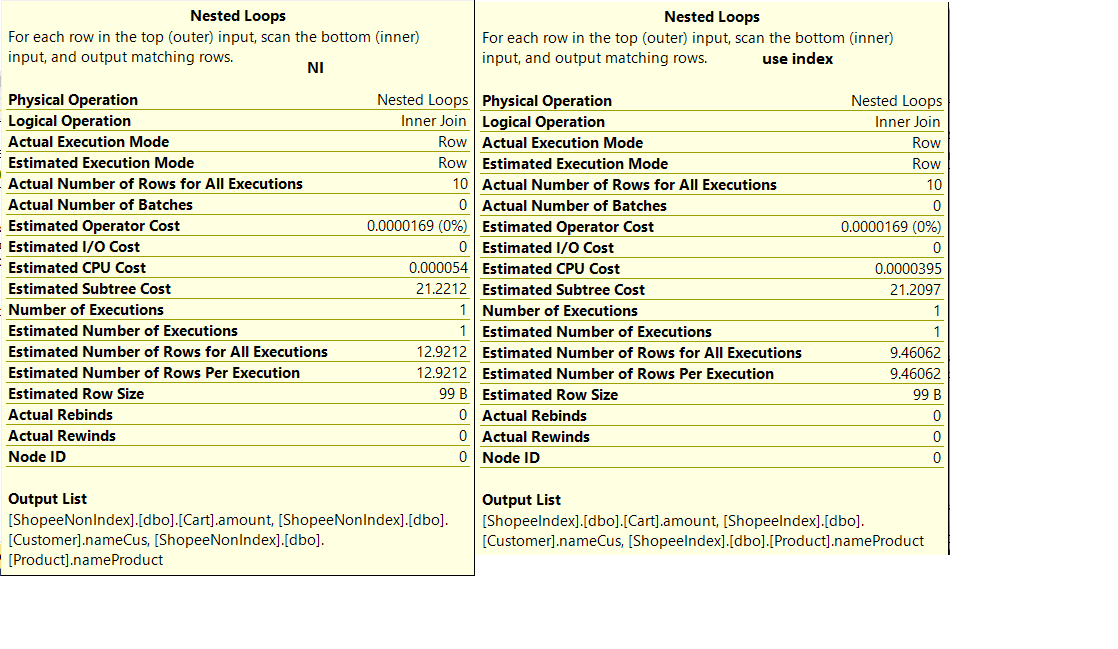
inner join Customer cus

on ca.idCus = cus.idCus

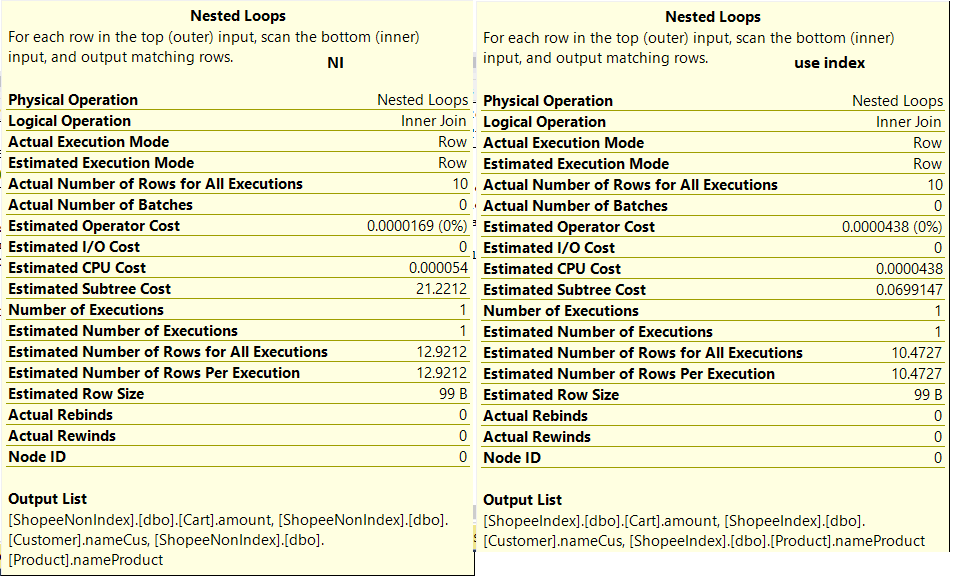
inner join Product p

on p.idProduct = ca.idProduct

where cus.idCus = 840



Recommend index:



* Hiển thị đơn hàng

select ot.idOrder, ot.date, cus.nameCus, p.nameProduct,

ot.delivery\_address,sc.name\_SCompany,sf.shipping\_fee ,

ot.total\_money + shipping\_fee as Total

from Order\_Table ot

inner join Customer cus

on cus.idCus = ot.idCus

inner join Shipping\_Fee sf

on sf.idOrder = ot.idOrder

inner join Cart ca

on ca.idOrder = ot.idOrder

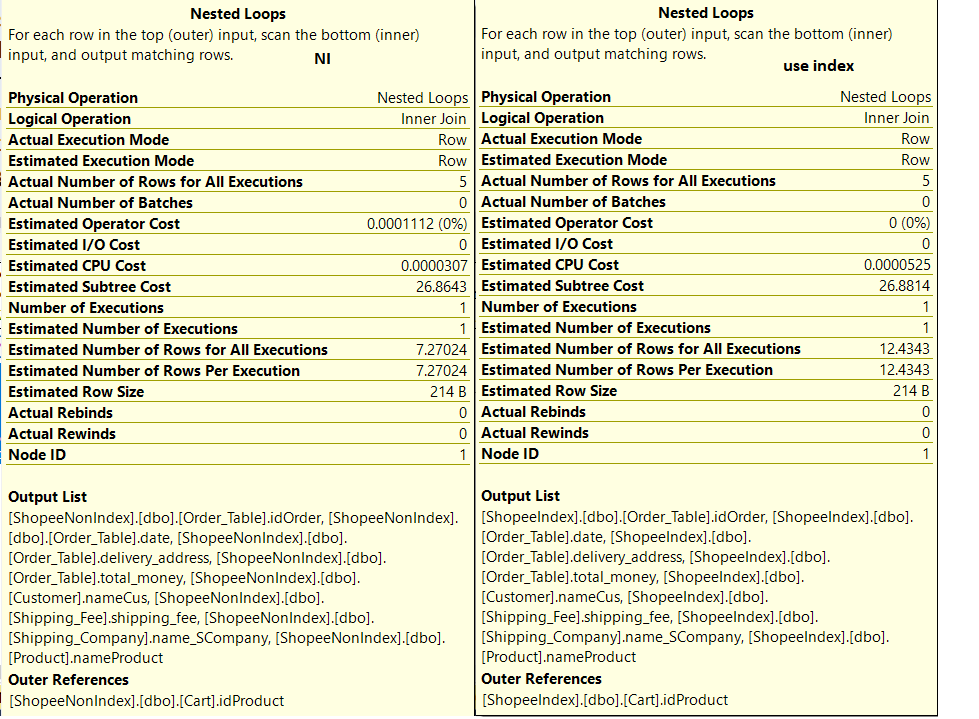
inner join Shipping\_Company sc

on sc.idSCompany = sf.idSCompany

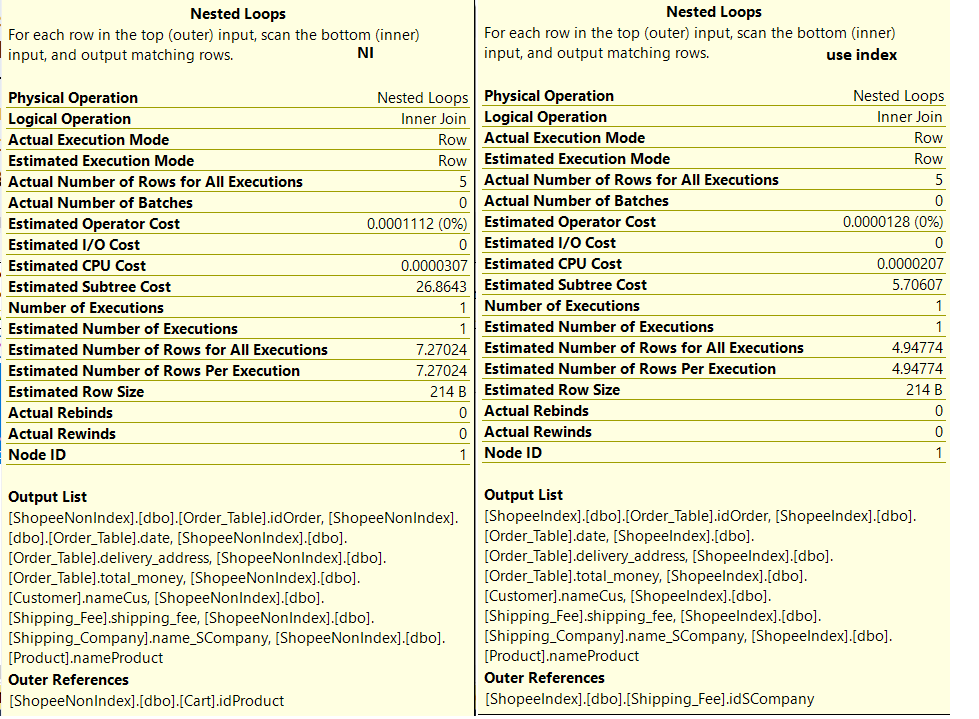
inner join Product p

on p.idProduct = ca.idProduct

where ot.idOrder = 264639



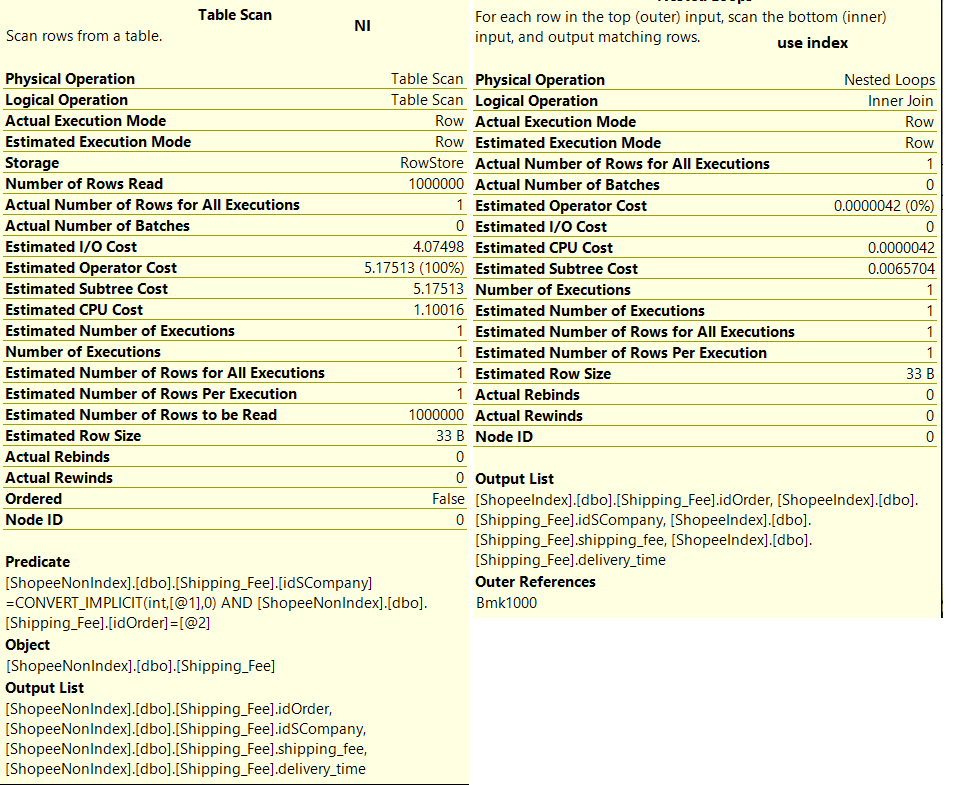
Recommend index



* Phí vận chuyển của đơn hàng

select \* from Shipping\_Fee

where idSCompany = 3 and idOrder = 897913



* Hiển thị lịch sử giao dịch của NCC

select p.nameProduct, sl.date,

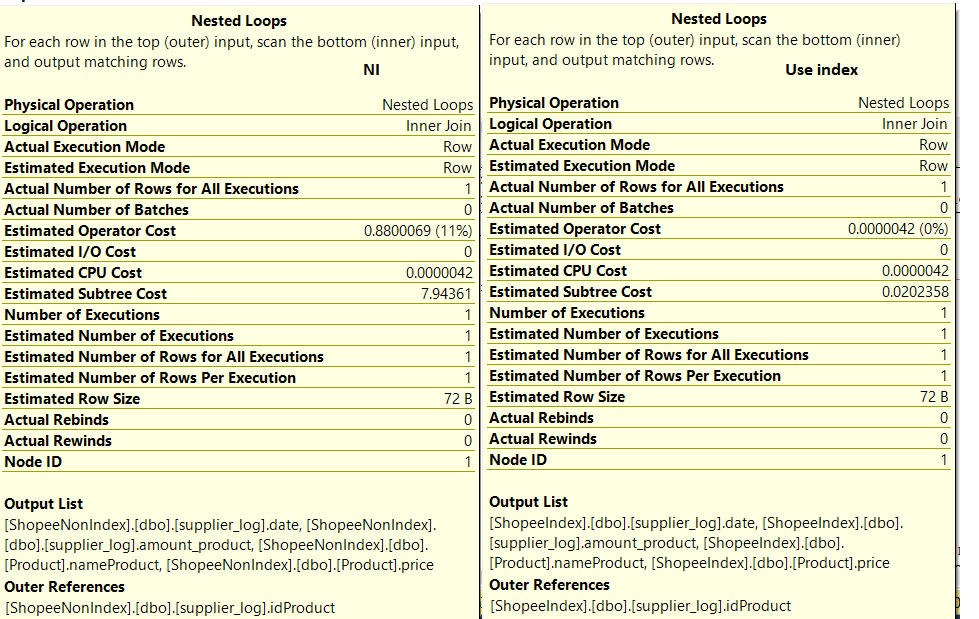
p.price \* amount\_product as revenue

from supplier\_log sl

inner join Product p

on p.idProduct = sl.idProduct

where date = '2020-01-21' and sl.idSup = 1440



* Hiển thị lịch sử giao dịch của khách hàng

select \* from Customer\_log cl

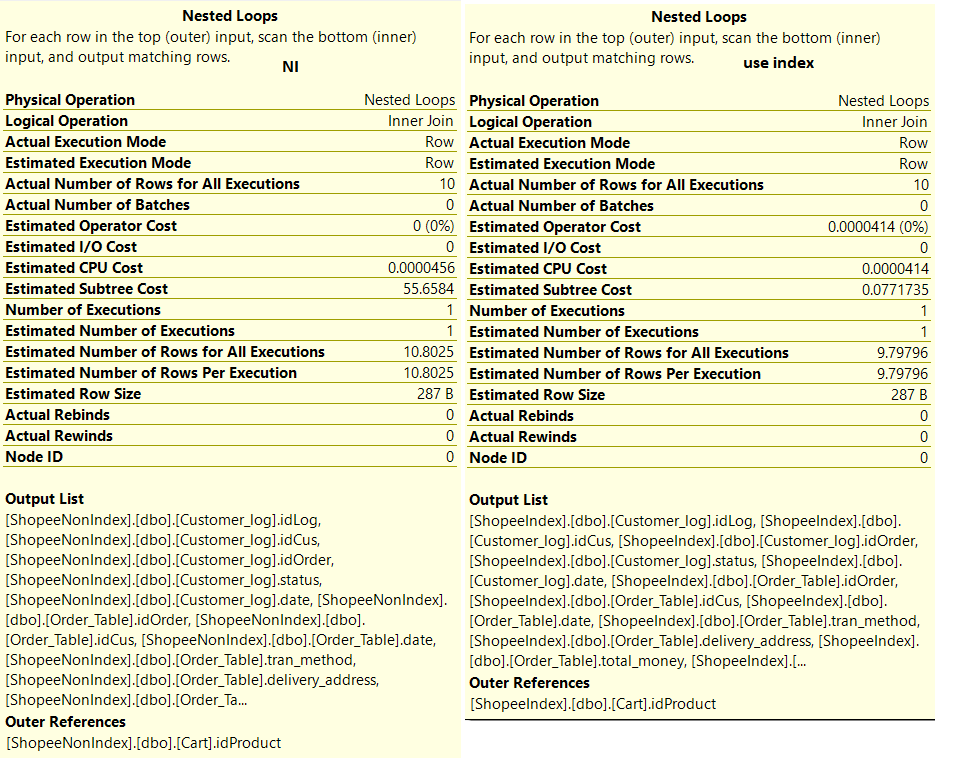
inner join Order\_Table ot

on cl.idOrder = ot.idOrder

inner join Cart ca on ca.idOrder = ot.idOrder

inner join Product p on p.idProduct = ca.idProduct

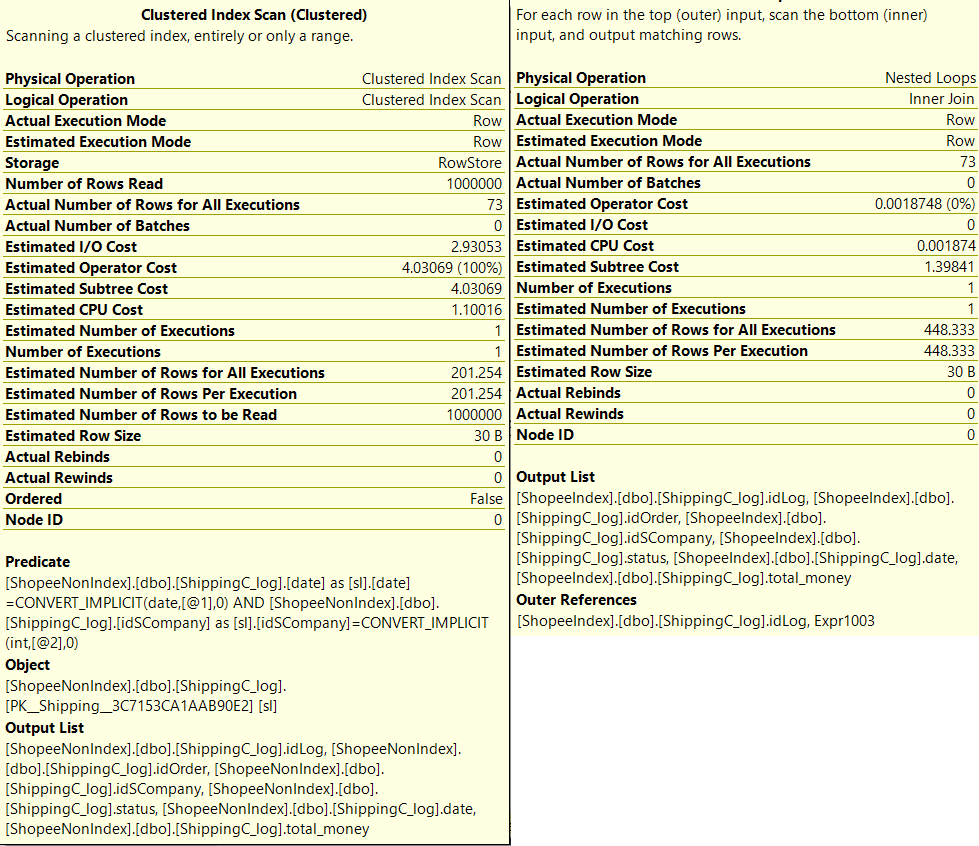
where cl.idCus = 282916



* Hiển thị lịch sử giao dịch với nhà vận chuyển

select \* from ShippingC\_log sl

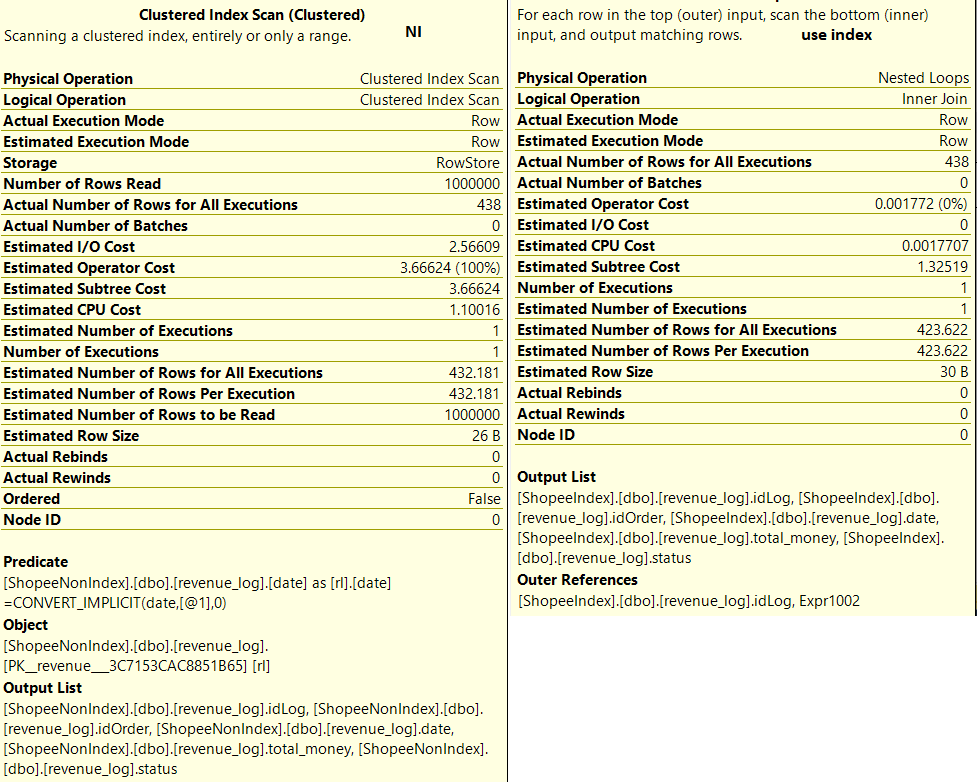
where sl.date = '2015-11-03' and sl.idSCompany = 2



* Hiển thị doanh thu mà shopee thu được

select \* from revenue\_log rl

where rl.date = '2020-10-21'



***Đánh giá tổng:***

* Sử dụng index cho kết quả truy vấn tốt hơn, nhanh hơn so với khi không dùng.
* Cải thiện tốc độ khi phải kết nhiều bảng với nhau
* Khi sử dụng index thì các chi phí bỏ ra để truy vấn kết quả đã được giảm đi 1 nửa (sub tree, )
* Những chỉ số khác như I/O, Operator, CPU Cost được giảm đi gần 9 lần (ví dụ ở bảng Hiển thị order của khách hàng: Operator cost được giảm đến 8.6 lần khi cài đặt index)
* Vậy việc sử dụng index là cần thiết trong việc vận hành hiệu quả ứng dụng, hệ thống, cải thiện tốc độ cũng như hiệu suất truy xuất dữ liệu

1. Trigger và Procedure

* Trigger

create trigger [dbo].[insert\_update\_cart]

ON [dbo].[Cart]

for INSERT, UPDATE

AS

BEGIN

declare @idProduct int

declare @idOrder int

declare @amount int

declare @idCart int

select @idProduct = ProductList.idProduct from inserted ProductList

print @idProduct

select @idOrder = ProductList.idOrder from inserted ProductList

select @idCart = ProductList.idCart from inserted ProductList

select @amount = ProductList.amount from inserted ProductList

declare @priceProduct int

select @priceProduct = price from [dbo].[Product] where [dbo].[Product].idProduct = @idProduct

declare @idSup int

select @idSup = idSup from [dbo].Provide where idProduct = @idProduct

if @priceProduct = 0

BEGIN

print 'Product id dont existed in the database'

ROLLBACK TRAN

END

if @amount = 0

BEGIN

print 'System dont accept amount = 0'

ROLLBACK TRAN

END

update [dbo].[Cart]

set amount = @amount

where [dbo].[Cart].[idCart] = @idCart and idProduct = @idProduct;

if exists (select idOrder from [dbo].Order\_Table where idOrder = @idOrder)

BEGIN

update Order\_Table set total\_money =

(

select Sum(CAST(c.amount\*p.price AS BIGINT)) from Cart c

inner join Product p on c.idProduct = p.idProduct

where Order\_Table.idOrder = c.idOrder

)

update [dbo].Order\_Table

set date = GETDATE()

where idOrder = @idOrder

END

END

go

create trigger [dbo].[delete\_order]

ON [dbo].[Order\_Table]

for DELETE

AS

BEGIN

declare @idOrder int

select @idOrder = OrderList.idOrder from deleted OrderList

if @idOrder = 0

BEGIN

print 'Order id cannot = 0'

ROLLBACK TRAN

END

delete from [dbo].[Cart] where idOrder = @idOrder;

END

go

create trigger [dbo].[insert\_update\_order]

ON [dbo].[Order\_Table]

FOR INSERT, UPDATE

AS

BEGIN

declare @date date

select @date = OrderList.date from inserted OrderList

declare @idOrder int

select @idOrder = OrderList.idOrder from inserted OrderList

declare @total\_money int

select @total\_money = total\_money from Order\_Table where idOrder = @idOrder

declare @idCus int

select @idCus = OrderList.idCus from inserted OrderList

declare @shippingFee int

select @shippingFee = shipping\_fee from Shipping\_Fee where idOrder = @idOrder

declare @tran\_method nvarchar(10)

select @tran\_method = OrderList.tran\_method from inserted OrderList

print @total\_money

execute insertUpdateCustomer\_log @idCus , @idOrder ,1 , @total\_money , @date

-- ShippingC\_log

IF @total\_money != 0

BEGIN

print @total\_money

execute insertUpdateShippingC\_log @idOrder , @date , 1 , @shippingFee

execute insertUpdateRevenue\_log @total\_money , @shippingFee , @idOrder , @idCus , @date , @tran\_method

END

END

go

create trigger [dbo].[delete\_product]

on [dbo].[Product]

for delete

as

begin

declare @idProduct int

select @idProduct = ProductList.idProduct from deleted ProductList

delete from Cart where idProduct = @idProduct

declare @idCart1 int

select @idCart1 = idCart from Cart where idProduct = @idProduct

declare @temp int

declare @idOrder int

select @idOrder = idOrder from Cart where idCart = @idCart1

select @temp = total\_money from Order\_Table where idOrder = @idOrder

declare @productPrice int

select @productPrice = price from Product

declare @temp2 int

set @temp2 = @temp - @productPrice

update Order\_Table

set total\_money = @temp2

where idOrder = @idOrder

delete from Product\_price\_log where idProduct = @idProduct

delete from Product\_review where idProduct = @idProduct

end

go

create trigger [dbo].[update\_product]

on [dbo].[Product]

for UPDATE

AS

BEGIN

declare @idProduct int

declare @newPrice int

select @idProduct = ProductList.idProduct from inserted ProductList

if @idProduct = 0

BEGIN

print 'Product id cannot = 0'

ROLLBACK TRAN

END

declare @oldPrice int

execute @oldPrice = getOldPrice @idProduct

print @oldPrice

select @newPrice = ProductList.price from inserted ProductList

EXEC procedure\_on\_updateProduct\_price\_log @newPrice1 = @newPrice, @oldPrice1 = @oldPrice, @idProduct1 = @idProduct

update Cart

set idProduct = @idProduct

where idProduct = @idProduct

END

* Procedure

create PROCEDURE [dbo].[getLatestIdLog]

AS

BEGIN

IF not exists (select max(idLog)+1 from Product\_price\_log)

BEGIN

return 1

END

ELSE

BEGIN

return (select max(idLog)+1 from Product\_price\_log)

END

END

go

create PROCEDURE [dbo].[getLatestIdLogCustomer\_log]

AS

BEGIN

return (select max(idLog)+1 from Customer\_log)

END

go

create PROCEDURE [dbo].[getLatestIdLogRevenue\_log]

AS

BEGIN

return (select max(idLog)+1 from revenue\_log)

END

go

create PROCEDURE [dbo].[getLatestIdLogShippingC\_log]

AS

BEGIN

return (select max(idLog)+1 from ShippingC\_log)

END

go

create PROCEDURE [dbo].[getLatestIdLogSupplier\_log]

AS

BEGIN

IF not exists (select max(idLog)+1 from supplier\_log)

BEGIN

return 1

END

ELSE

BEGIN

return (select max(idLog)+1 from supplier\_log)

END

END

go

create PROCEDURE [dbo].[getOldPrice] @idProduct1 int

AS

BEGIN

if not exists (select new\_price from Product\_price\_log where idProduct = @idProduct1)

begin

return 0

end

else

begin

return (select new\_price from Product\_price\_log where idProduct = @idProduct1)

end

END

go

create PROCEDURE [dbo].[insertUpdateCustomer\_log] @idCus int, @idOrder int,@status int, @date date

AS

BEGIN

IF not exists (select idLog from Customer\_log where idCus = @idCus)

BEGIN

declare @idLogCus int

execute @idLogCus = getLatestIdLogCustomer\_log

IF @idLogCus = 0

BEGIN

set @idLogCus = 1

END

insert into Customer\_log(idCus, idLog, idOrder, status, date)

values (@idCus, @idLogCus, @idOrder,@status, @date)

END

ELSE

BEGIN

declare @idLogCus2 int

select @idLogCus2 = idLog from Customer\_log where idOrder = @idOrder and idCus = @idCus

update Customer\_log

set date = @date, status = @status

where idLog = @idLogCus2

END

END

go

create PROCEDURE [dbo].[insertUpdateRevenue\_log] @total\_money int, @shippingFee int, @idOrder int, @date date, @status int

AS

BEGIN

print @total\_money

declare @temp int

set @temp = 0.02\*@shippingFee +0.02\*(@total\_money - @shippingFee)

IF not exists (select idLog from revenue\_log where idOrder = @idOrder)

BEGIN

declare @idLog int

execute @idLog = getLatestIdLogRevenue\_log

IF @idLog = 0

BEGIN

set @idLog = 1

END

insert into revenue\_log(idLog, idOrder, date, status ,total\_money)

values (@idLog, @idOrder, @date, @status ,@temp )

END

ELSE

BEGIN

declare @idLog2 int

select @idLog2 = idLog from revenue\_log where idOrder = @idOrder

update revenue\_log

set date = @date, total\_money = @temp, status = @status

where idLog = @idLog2

END

END

go

create PROCEDURE [dbo].[insertUpdateShippingC\_log] @idOrder int, @date date, @status int, @shippingFee int

AS

BEGIN

IF not exists (select idLog from ShippingC\_log where idOrder = @idOrder)

BEGIN

declare @idLogShippingC\_log int

execute @idLogShippingC\_log = getLatestIdLogShippingC\_log

IF @idLogShippingC\_log = 0

BEGIN

set @idLogShippingC\_log = 1

END

declare @idSCompany int

select @idSCompany = idSCompany from Shipping\_Fee where idOrder = @idOrder

insert into ShippingC\_log(idLog, idOrder, idSCompany, status, date, total\_money)

values (@idLogShippingC\_log, @idOrder,@idSCompany ,@status, @date, 0.98\*@shippingFee)

END

ELSE

BEGIN

declare @idLogShippingC\_log2 int

select @idLogShippingC\_log2 = idLog from ShippingC\_log where idOrder = @idOrder

update ShippingC\_log

set date = @date, status = @status, total\_money =0.98\*@shippingFee

where idLog = @idLogShippingC\_log2

END

END

go

create PROCEDURE [dbo].[procedure\_on\_updateProduct\_price\_log] @newPrice1 int, @oldPrice1 int, @idProduct1 int

AS

BEGIN

IF not exists (SELECT idLog FROM Product\_price\_log WHERE idProduct = @idProduct1)

BEGIN

declare @tempIdLog int

execute @tempIdLog = getLatestIdLog;

IF @tempIdLog = 0

BEGIN

set @tempIdLog = 1

END

insert into Product\_price\_log(idProduct, idLog, new\_price, date\_update, old\_price)

values(@idProduct1,@tempIdLog ,@newPrice1, GETDATE(), @oldPrice1)

END

else

begin

update Product\_price\_log

set new\_price = @newPrice1, old\_price = @oldPrice1, date\_update = GETDATE()

where idProduct = @idProduct1

END

END

1. Xác định yêu cầu chức năng:

Chức năng:

1. Trang chủ:

* Xem danh sách sản phẩm trên trang chủ.
* Hiển thị sản phẩm theo danh sách, tìm kiếm sản phẩm theo tên.
* Đăng nhập + đăng ký.
* Trang thông tin cá nhân:
* Xem thông tin cá nhân + địa chỉ giao hàng + đơn hàng vừa đặt, thay đổi thông tin cá nhân.
* Thay đổi địa chỉ giao hàng.
* Kiểm tra order theo mã order và email
* Xem danh sách tất cả order đã và đang đặt.

1. Admin:

* Xem doanh thu tháng này/ doanh thu tổng của ShopePhake.
* Xem danh sách user có trong hệ thống (có thể thay đổi chức năng của người dùng ), tìm kiếm người dùng theo tên
* Xem các nhà cung cấp hiện có trong hệ thống, tìm kiếm nhà cung cấp theo tên.
* Xem danh sách các đối tác giao dịch có trong hệ thống, tìm kiếm đối tác theo tên.
* Xem danh sách các đơn vị vận chuyển, tìm kiếm đơn vị vận chuyển theo tên.

1. Nhà Cung cấp:

* Xem thông tin của nhà cung cấp, thay đổi thông tin.
* Thêm sản phẩm vào hệ thống.
* Cập nhật giá.
* Xem danh sách đơn hàng của nhà cung cấp và tùy chỉnh trạng thái đơn.
* Xem doanh thu theo ngày/ tháng/ năm, xem doanh thu từ ngày này sang ngày khác.

1. Giỏ hàng

* Ước tính giá ship, thay đổi số lượng hoặc xóa cả giỏ hàng hoặc xóa 1 mặt hàng.
  + Chức năng thanh toán đơn.
* Chọn đơn vị vận chuyển, thời gian, hình thức thanh toán, xong tiến hành thanh toán đơn hàng.

|  |  |  |
| --- | --- | --- |
|  | Đối tượng sử dụng | Mô tả |
| Chức năng tạo đơn hàng mới, lấy dữ liệu từ đơn vị vận chuyển. Chỉnh sửa đơn hàng nếu chưa thực hiện giao hàng. | Khách hàng | Khách hàng chọn sản phẩm cần mua, tạo mới đơn, chọn đơn vị vận chuyển phù hợp, xác nhận đơn hàng và tiền hành thanh toán( tiền mặt/ tiền điện tử). Có thể thay đổi số lượng hoặc thêm sản phẩm khác tồn tại trong shop. |
| Chức năng đặt hàng chung | Khách hàng | Khi tiến hành đặt hàng, nếu giỏ hàng chưa đủ chỉ tiêu để nhận khuyến mãi (VD: Đơn hàng > 500.000VNĐ để nhận mã giảm 50%), thì khách hàng có thể dùng code của đơn hàng (hệ thống sẽ tự tạo) để đưa cho bạn bè mình đặt thêm. Khi vận chuyển thì đơn sẽ được tách ra và giao về địa chỉ của khách hàng và bạn bè của khách. |
| Chức năng trang trí giao diện của trang bán | Chủ shop | Chủ shop có thể tùy chỉnh giao diện shop của mình cho bắt mắt, thu hút khách hàng. Các giao diện này có thể được trang trí theo template của hệ thống hoặc chủ shop kéo thả các mục tùy sao cho hợp lý (VD: mục tên shop, đánh giá nên để đầu trang, màu sắc sao cho đẹp, sản phẩm nên để giữa trang cho khách hàng dễ dàng quan sát). |
| Chức năng theo dõi doanh thu bán hàng, số lượng và tình trang đơn hàng đang giao | Chủ shop | Chủ shop có thể xem thống kê doanh thu của tháng này/ năm/ tổng doanh thu. Sản phẩm bán chạy nhất, sản phẩm bán ít nhất để dễ dàng tùy chỉnh số lượng bán trong tương lai. Xem được tình trạng của đơn hàng đang giao (giao chậm, đang giao, đã giao, đã hủy). |
| Chức năng theo dõi các thành viên trong hệ thống | Admin( nhà quản lý) | Admin có thể xem được các tài khoản đang có trong hệ thống, có thể là khách hàng, chủ shop và các nhà quản lý khác. |
| Chức năng theo dõi doanh thu/ số lượng tồn kho/ đánh giá của từng shop | Admin (nhà quản lý) | Admin có thể xem được tình trạng hoạt động của shop, số lượng đánh giá tiêu cực, tích cực của shop. Nếu nhận quá nhiều đánh giá tiêu cực sẽ bị cảnh cáo, và có thể cho đóng shop nếu còn tiếp diễn. |
| Chức năng theo dõi đơn hàng | Admin (nhà quản lý) | Admin có thể quản lý các đơn hàng đang giao của từng shop để theo dõi liệu đơn vị vận chuyển A có giao đúng thời hạn không, chủ shop có đưa hàng đúng thời hạn cho đơn vị vận chuyển không. Thống kê số lần giao hàng trễ của đơn vị vận chuyển/ shop để cảnh cáo. |
| Chức năng quản lý đơn vị vận chuyển | Admin ( nhà quản lý) | Admin có thể quản lý các đơn vị vận chuyển đã có hợp đồng với công ty. Các thông tin quản lý như: Tên đơn vị, giá trên 1km, điều kiện nhận khuyến mãi theo từng đơn vị vận chuyển, thời gian trung bình giao 1 đơn tùy theo vị trí địa lý. |

* Chức năng đăng nhập: phân quyền người dùng( khách hàng, chủ shop, admin).
* Quy trình: khách hàng đăng nhập có thể xem danh sách các shop đang hoạt động (sản phẩm bán chạy nhất/đang khuyến mãi ở trang chủ). “Khách hàng” khi đăng ký bán sản phẩm thì trở thành “Chủ shop”. Tình huống: khách hàng => đăng ký bán sản phẩm => hệ thống xác nhận => chủ shop.
* Chủ shop đăng ký tạo 1 gian hàng online => thêm/xóa/sửa/cập nhật dữ liệu của sản phẩm trên gian hàng.
* Admin khi đăng nhập vào trang ví dụ : /wp-admin thì sẽ chuyển hướng vào trang quản lý người dùng, quản lý các chủ shop (doanh thu, doanh số, sao đánh giá, …).

KHÁCH HÀNG

* Chức năng tạo đơn, lấy dữ liệu từ các đơn vị vận chuyển ( những thông tin cần lấy: tên đơn vị, giá ship hàng theo km tính từ vị trí của khách hàng đến kho của công ty, khuyến mãi nếu có), xác nhận đơn, hoàn tất đơn. Chức năng đánh giá sản phẩm sau khi hoàn tất đơn. Chức năng chỉnh sửa đơn hàng nếu đơn hàng chưa được vận chuyển. Chức năng gửi email quảng cáo/khuyến mãi (chỉ với những người dùng đăng ký).
* **Chức năng có thể có:** đặt hàng chung với bạn bè thông qua mã liên kết giỏ hàng. Tình huống: người A đặt xong sản phẩm tuy nhiên vẫn chưa đủ chỉ tiêu để nhận khuyến mãi, lúc này người A có thể lấy mã liên kết giỏ hàng cho bạn mình đặt, sau khi hoàn tất đặt hàng thì sẽ có 2 đơn vị vận chuyển (giá ưu đãi) thực hiện chuyển phát.

CHỦ SHOP:

* + Chủ shop ngoài thêm xóa sửa cập nhật còn có: thiết kế giao diện của gian hàng**(TƯƠNG TỰ CHỨC NĂNG TRANG TRÍ SHOP CỦA SHOPEE).**

VÍ DỤ: https://banhang.shopee.vn/edu/article/851 .

* + Chủ shop có thể thống kê doanh thu/ số lượng đơn hàng.

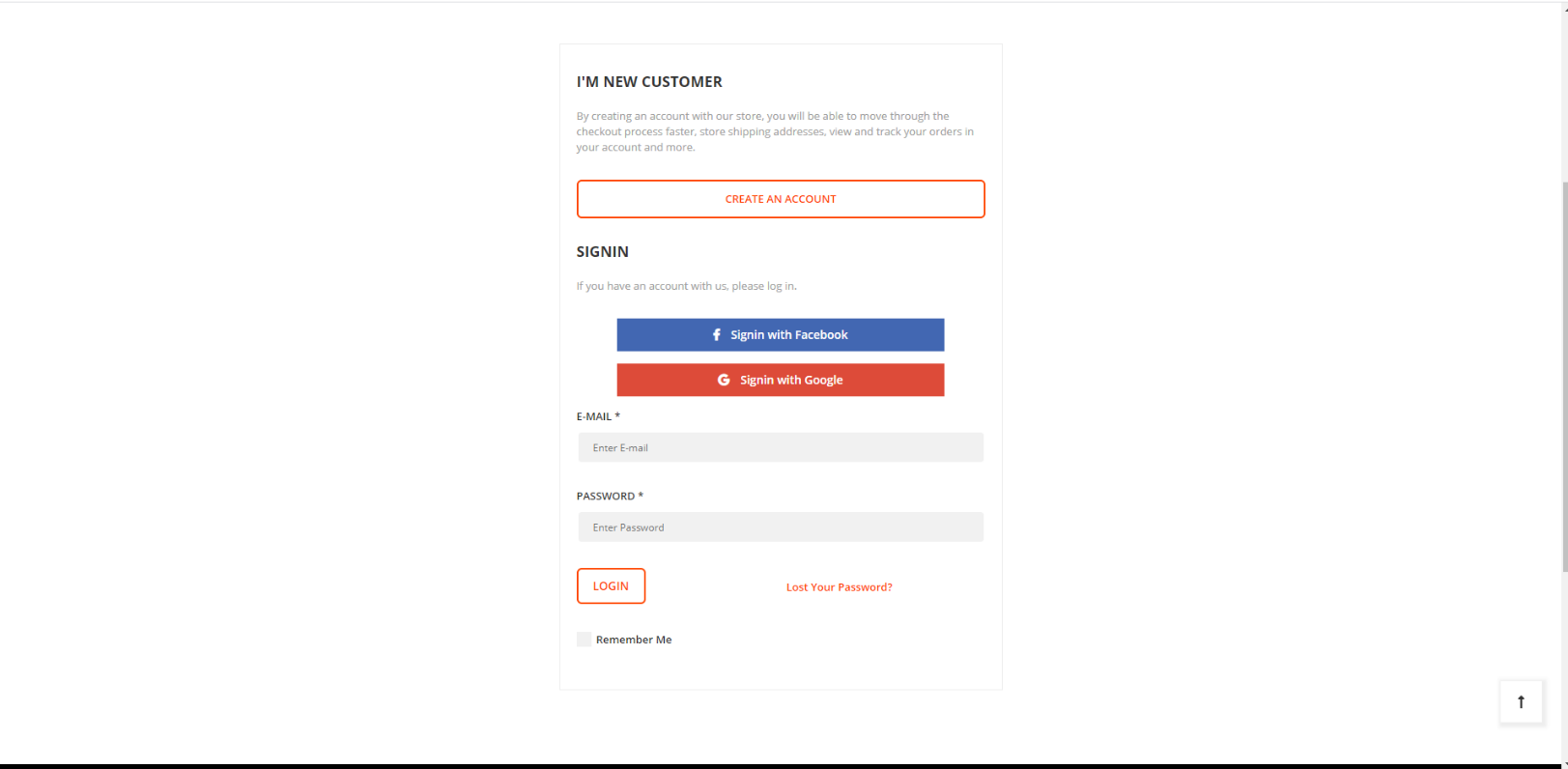


ADMIN ( NHÀ QUẢN LÝ )

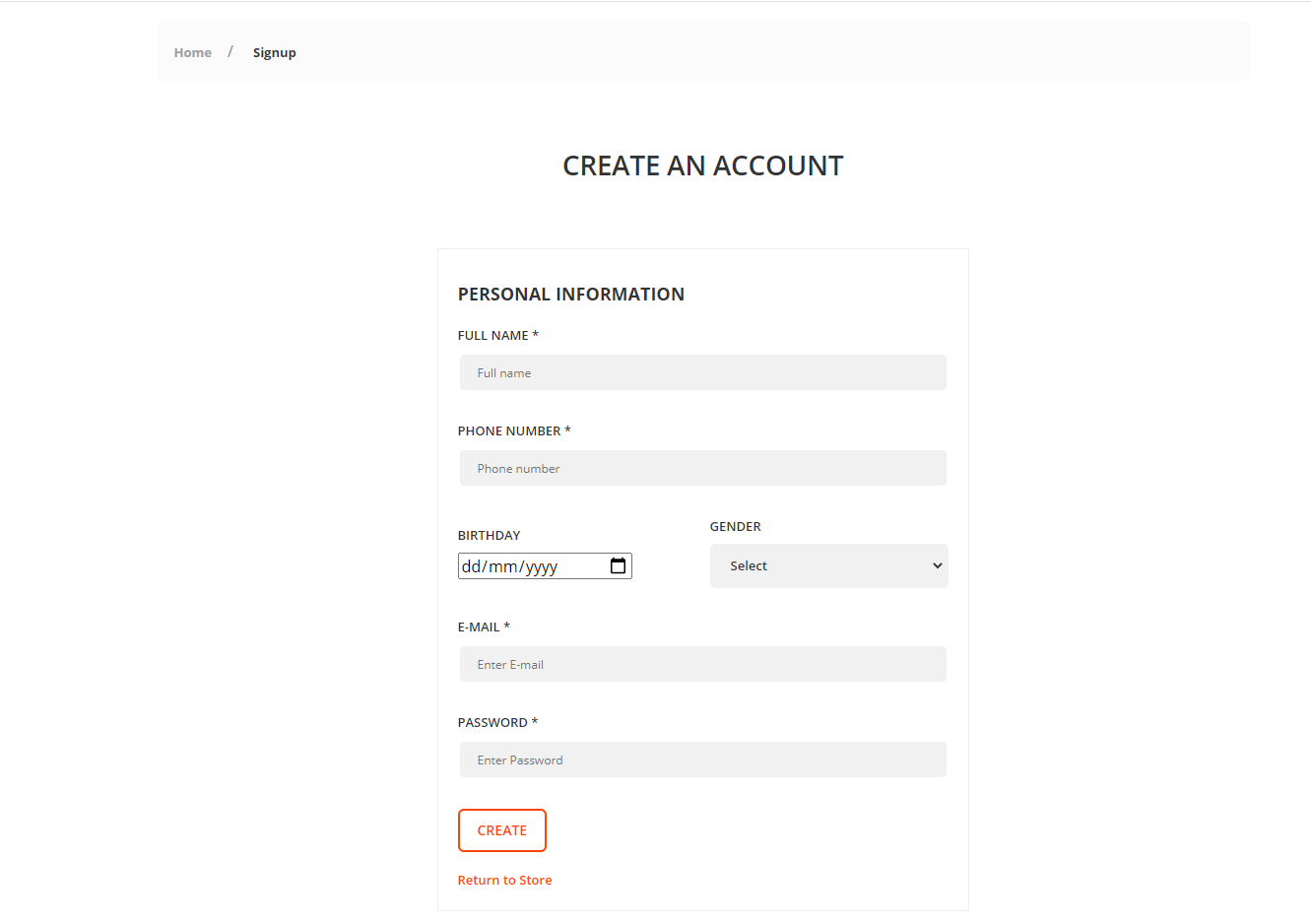
* Theo dõi số thành viên trong hệ thống - có thể là các quản lý, các khách hàng, các chủ shop- (tên tài khoản, email, vai trò, bài đăng/đánh giá, lần cuối đăng nhập)
* Chức năng theo dõi doanh thu/số lượng tồn của các shop, số lượng đánh giá tích cực/ tiêu cực. Nếu quá nhiều đánh giá tiêu cực/ tố cáo thì shop sẽ bị ngưng hoạt động.
* Theo dõi tình trạng đơn hàng đơn hàng đã giao, chưa giao, còn chờ giao, giao trễ của shop.
* Quản lý đơn vị vận chuyển (tên đơn vị, số tiền trên 1 km, điều kiện nhận khuyến mãi, thời gian trung bình giao 1 đơn tùy theo vị trí địa lý)

1. **Layout của giao diện:**

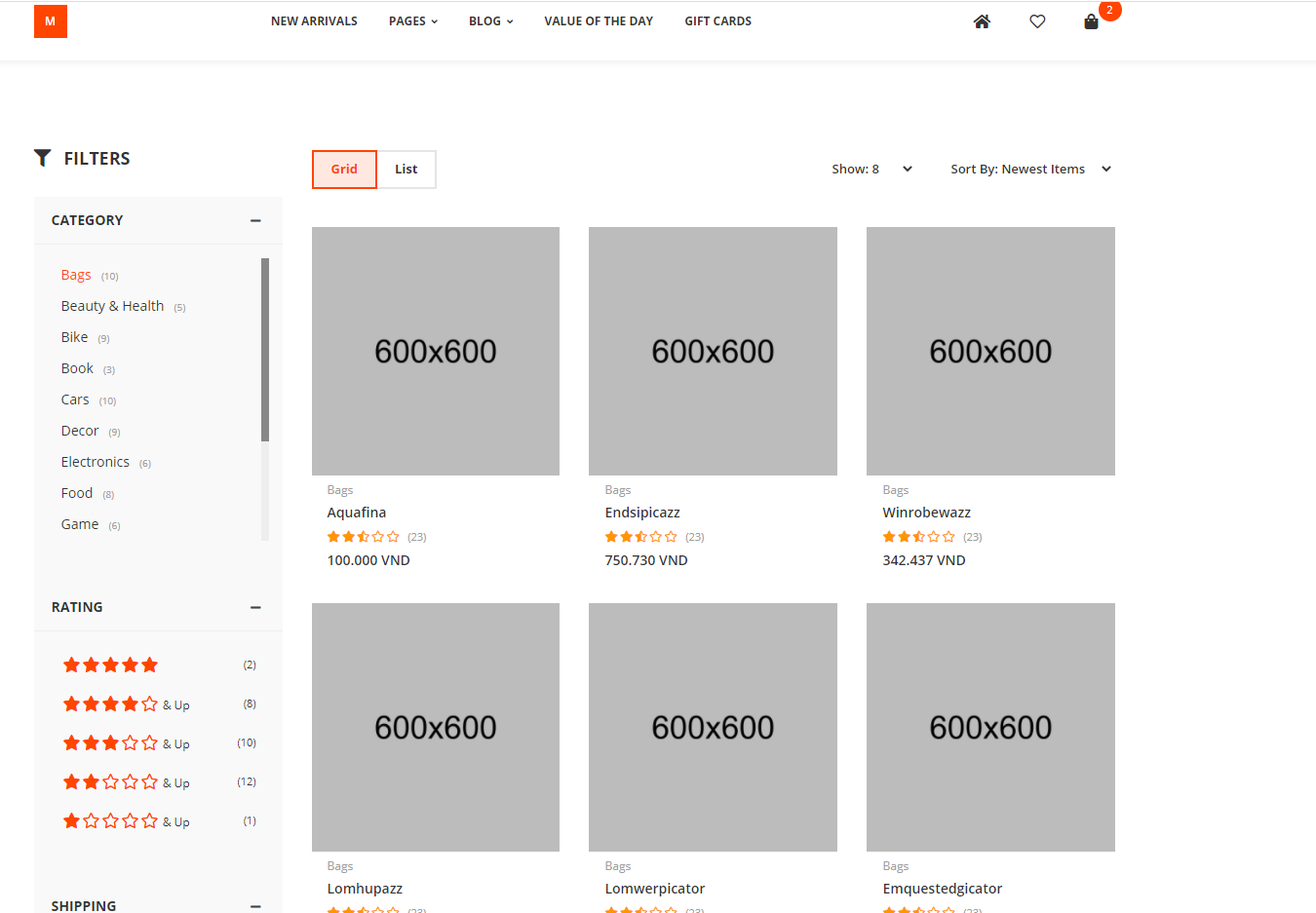
**Đăng nhập:**



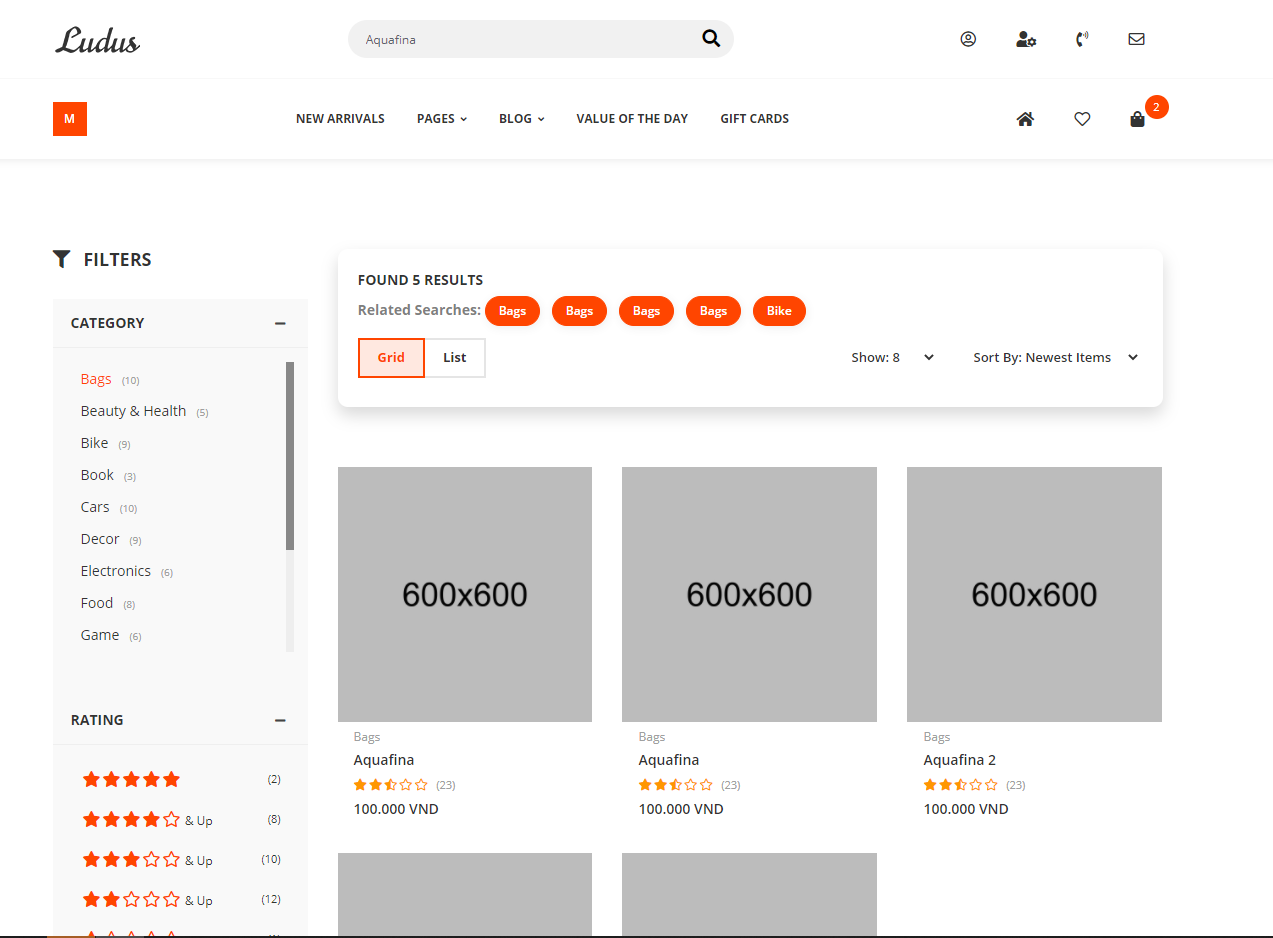
**Đăng ký:**



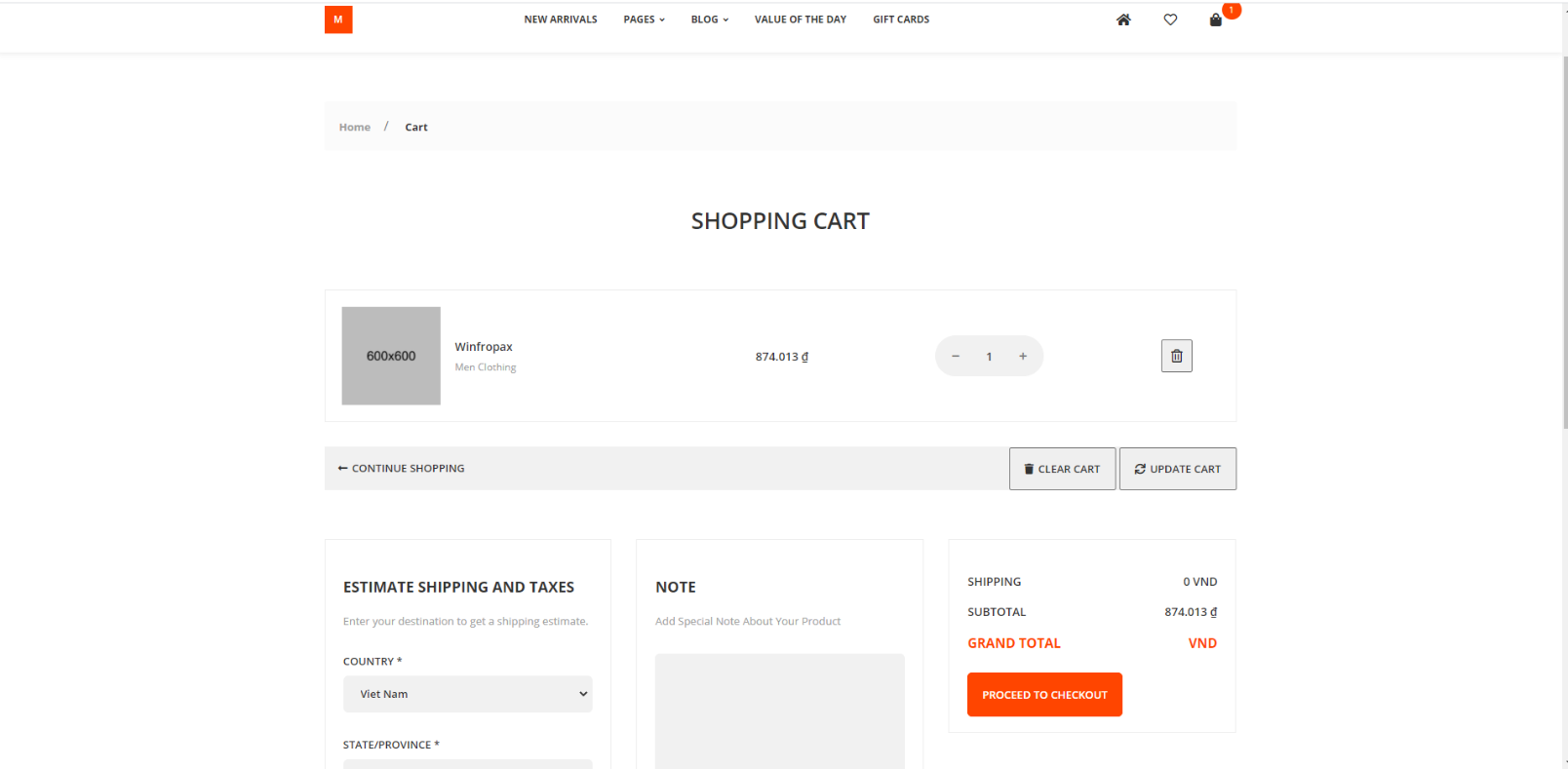
**Xem sản phẩm theo loại**



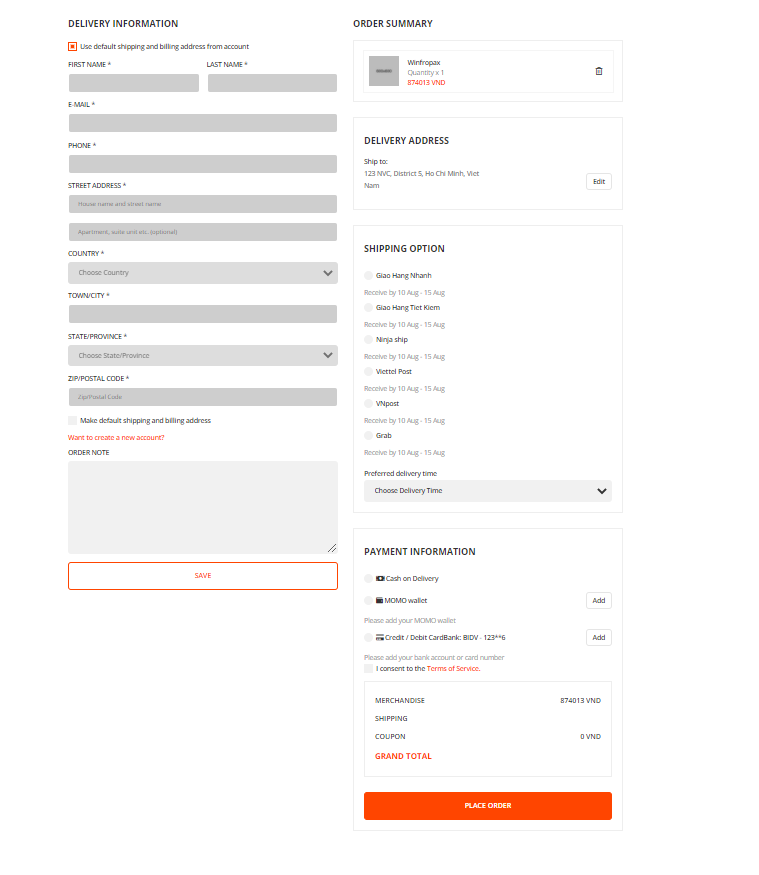
**Tìm kiếm sản phẩm(Aquafina):**



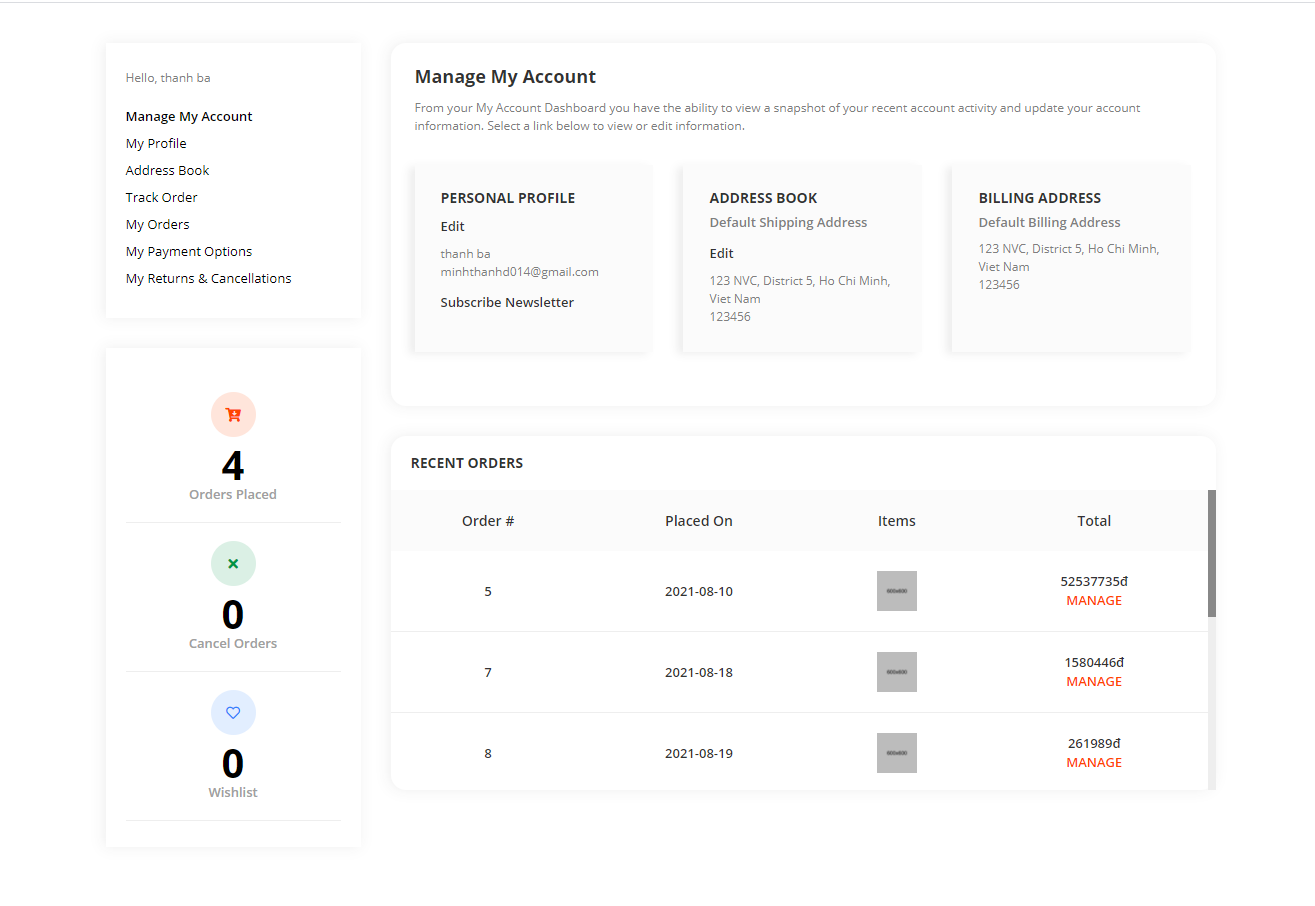
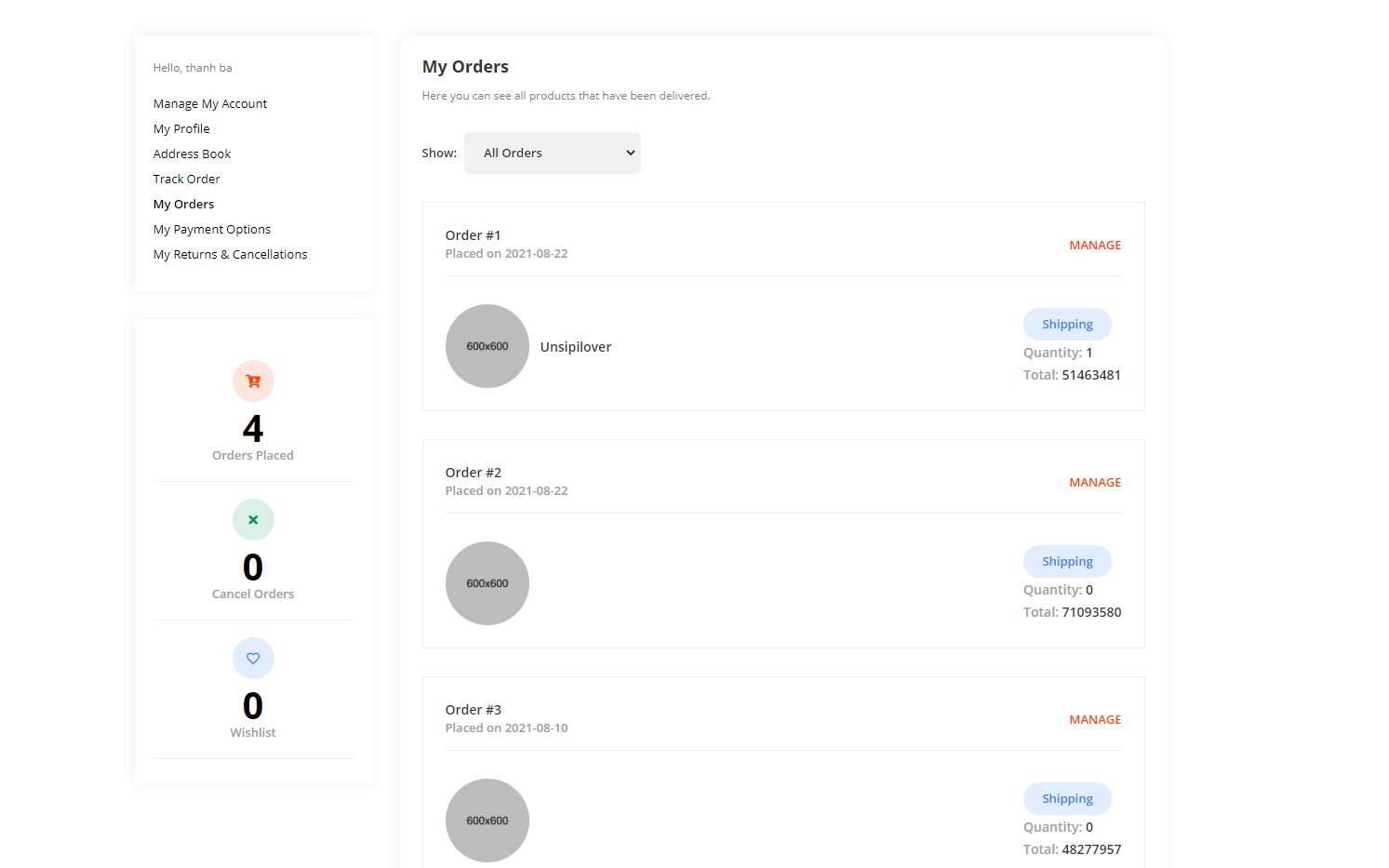
**Giỏ hàng:**



**Thanh toán:**

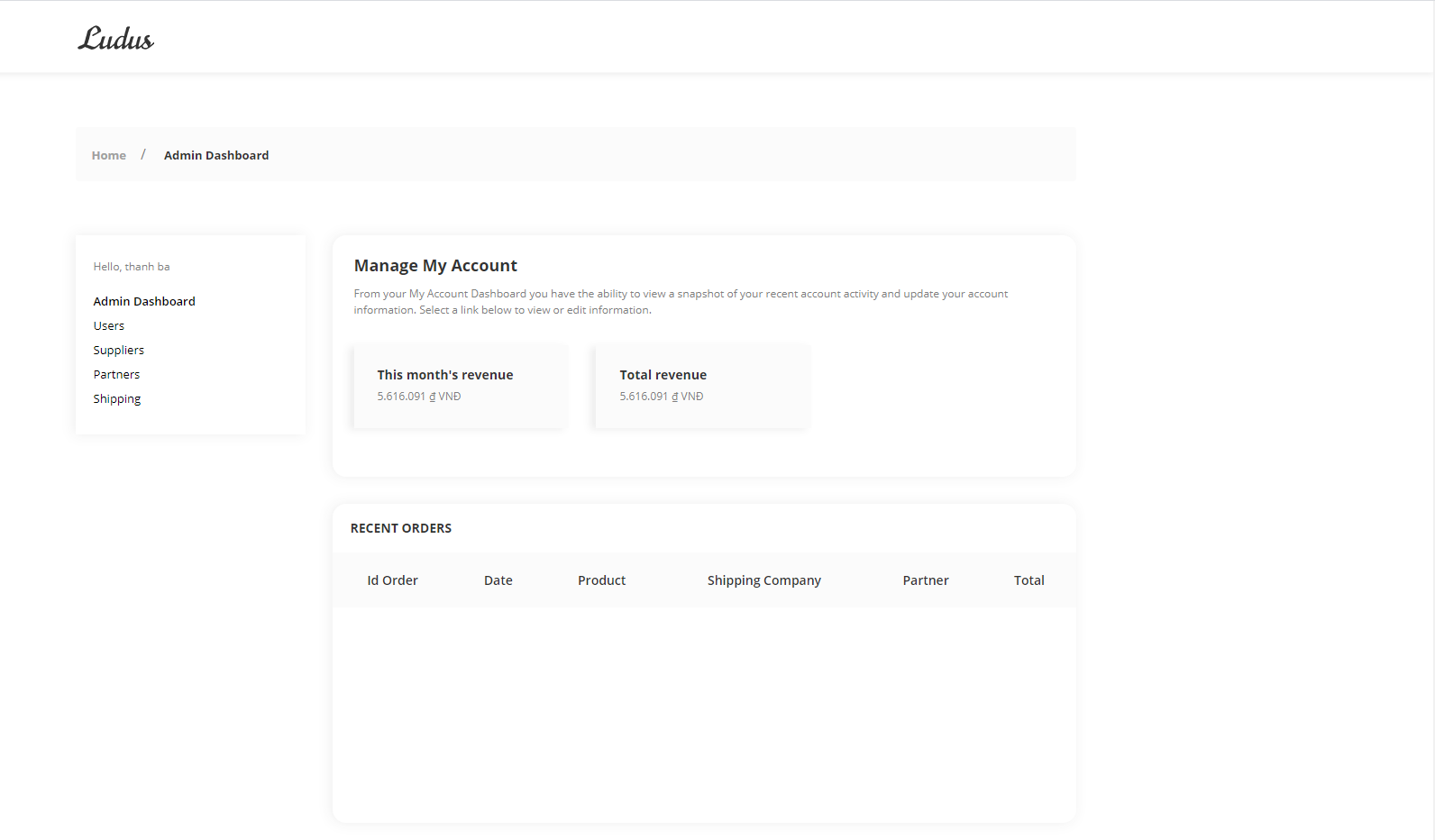


**Trang cá nhân:**

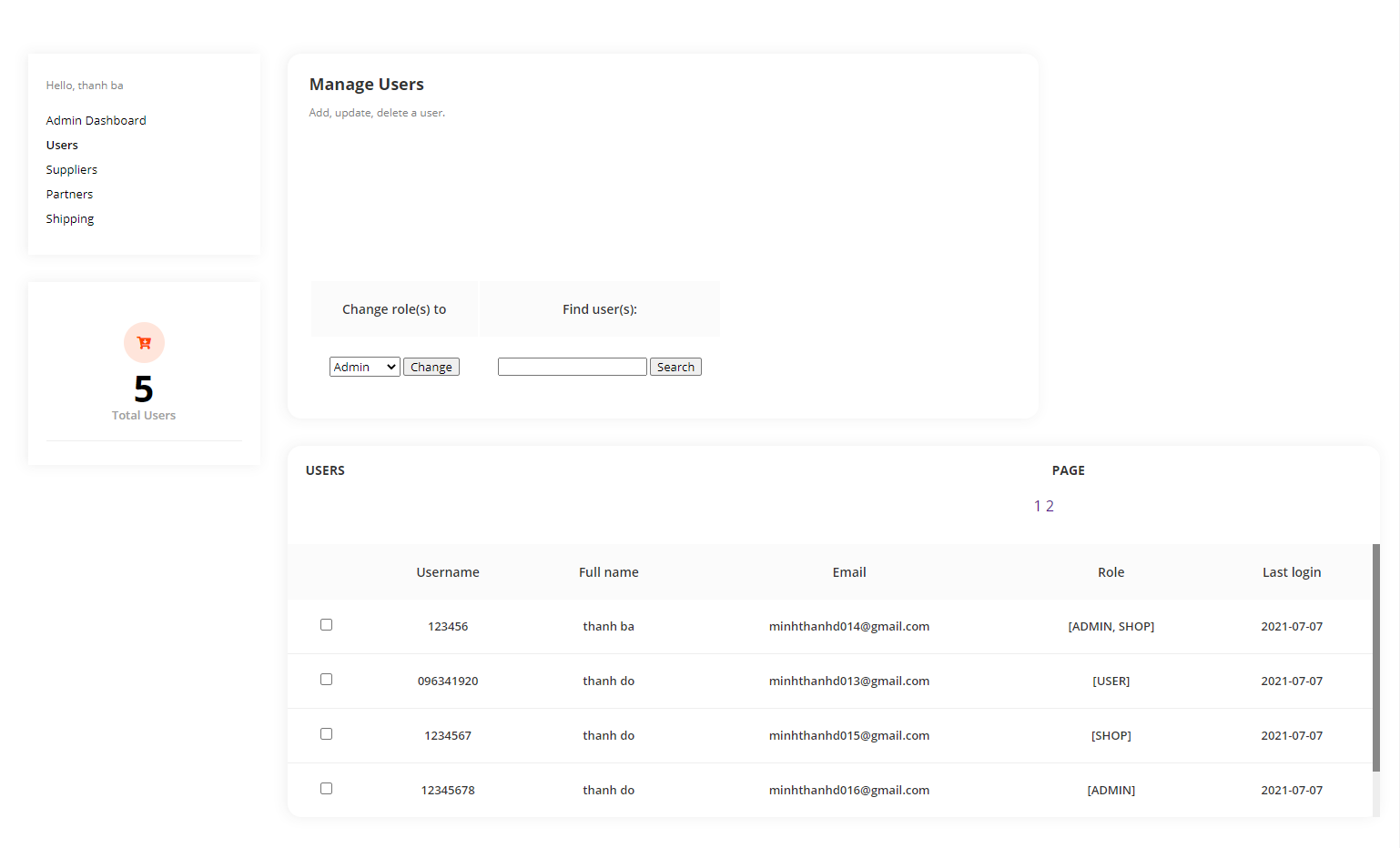
* Thông tin cá nhân:
* 
* Hiển thị danh sách order:
* 

**Admin:**

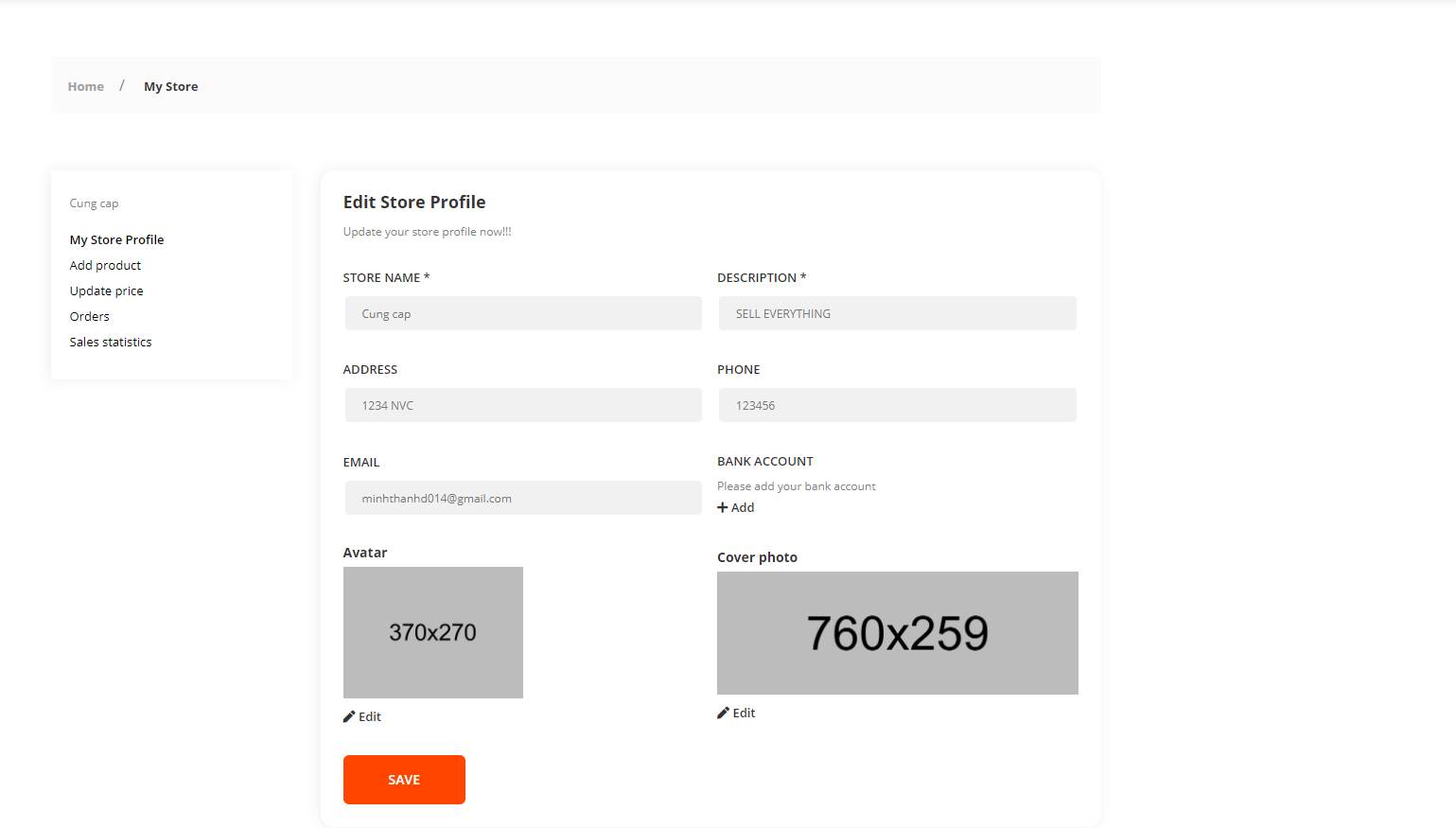
Admin dashboard:



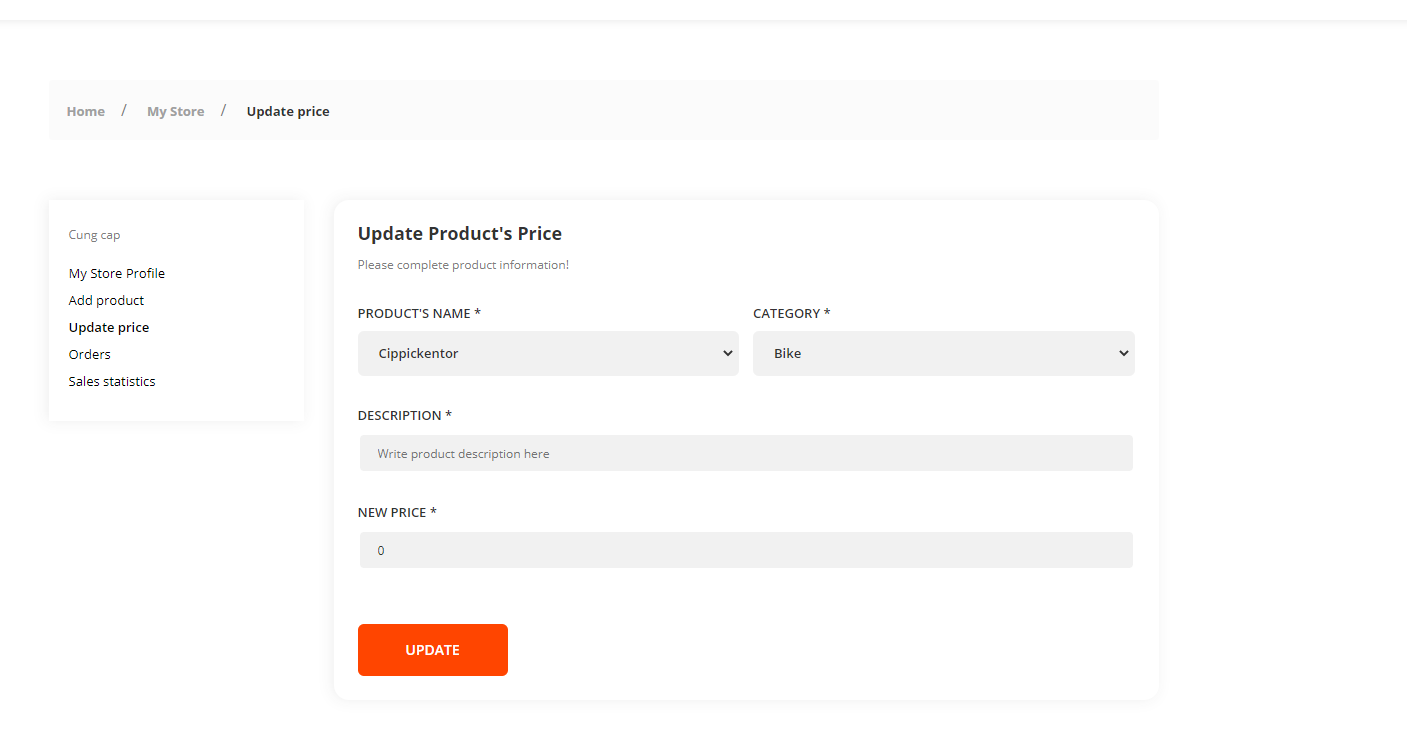
Danh sách user:



**Chủ shop:**



* Thay đổi giá:



1. **Partiton**

**Phân chia dữ liệu – Partition**

1. **Partition:**

* *Partition (Phân chia dữ liệu):* là một giải pháp chia nhỏ một Database lớn thành nhiều Database nhỏ, ta có thể phân tách từng bảng hoặc cả một DB ra nhiều phần nhỏ đặt ở nhiều máy chủ (server) khác nhau. Điều này sẽ giúp cho hệ thống DB của chúng ta đạt được các tính chất khả năng bảo trì, hiệu xuất, tính sẵn sàng, và cân bằng tải của ứng dụng. Và giải pháp này cũng giảm chi phí cũng như tính mở rộng để scale up DB bằng cách dùng nhiều server nhỏ gộp lại hơn là nâng cấp một server lớn.
* Partition function: Ánh xạ các dòng của bảng hoặc các chỉ mục trong phân mảnh dựa theo tiêu chí phân mảnh. Qui định giá trị biên cho các đoạn. Hệ thống dựa vào hàm này để xác định đoạn mà mỗi bản ghi thuộc vào.
* Partition Scheme: Ánh xạ các phân mảnh của bảng vào các filegroup (mỗi filegroup chứa một phân mảnh)

1. **Lợi ích của việc sử dụng Partition:**

* Cải thiện khả năng co giãn và khả năng quản lý của các bảng lớn (large table) và các bảng có các kiểu truy cập khác nhau.
* Khi các bảng và các chỉ mục quá lớn, việc phân mảnh giúp chia dữ liệu thành cá phần nhỏ hơn, có thể quản lý được.
* Nếu 1 bảng lớn tồn tại trên hệ thống nhiều CPUs, việc phân mảnh bảng sẽ giúp tăng hiệu suất khi thực hiện song song
* *Việc backup/restore* một đoạn mà không ảnh hưởng đến các đoạn còn lại.

🡺 Phân chia dữ liệu sẽ giúp cho hệ thống DB đạt được các tính chất khả năng bảo trì, hiệu xuất, tính sẵn sàng, và cân bằng tải của ứng dụng. Và giải pháp này cũng giảm chi phí cũng như tính mở rộng để scale up DB bằng cách dùng nhiều server nhỏ gộp lại hơn là nâng cấp một server lớn.

1. **Các kiểu phân chia dữ liệu:**

* Phân chia theo chiều ngang (horizontal partition):
* Phân chia các dòng trong một table thành nhiều table khác nhau.
* Tình huống áp dụng: khi nhiều người dùng khác nhau cần truy cập các dòng dữ liệu khác nhau.
* Ưu điểm: tối ưu hoá tốc độ truy cập dữ liệu
* Nhược điểm:

Phức tạp khi phải truy cập toàn bộ dữ liệu.

Nếu không phân chia cẩn thận sẽ dễ gây mất cân bằng giữa các DB.

* Phân chia theo chiều dọc (vertical partition):
* Phân chia một cấu trúc luận lý thành những cấu trúc lưu trữ vật lý khác nhau. Phân chia dựa vào thuộc tính (cột) của bảng thành nhiều bảng khác nhau.
* Tình huống áp dụng: chia bảng thành các loại cấu trúc có giá trị sử dụng khác nhau hoặc cấu trúc truy cập thường xuyên và không thường xuyên.
* Ưu điểm: Dễ thực hiện và không ảnh hưởng đến ứng dụng
* Nhược điểm: Khi hệ thống lớn thì dữ liệu cũng lớn theo, do đó phải thực hiện phân chia nhiều lần.

1. **Phân chia dữ liệu bảng Order (Hoá đơn) theo chiều ngang**

**Bước 1: Tạo các file group**

* Tạo các filegroup để chứa các phân mảnh sau khi đã phân chia.
* Group 1 (Primary): lưu các đơn hàng trước năm 2020 (ngày 31/12/2019 trở về trước)
* Group 2 (FG2): lưu các đơn hàng trong năm 2020 (từ ngày 1/1/2020 đến 31/12/2020)
* Group 3 (FG3): lưu các đơn hàng sau năm 2020 (từ ngày 1/1/2021 trở về sau)

**CREATE DATABASE** **DB\_Order\_Partition**

ON PRIMARY

(NAME= 'Order\_Partition\_1', FILENAME='D:\PartitionDB\FG1\Order\_Partition\_1.mdf', SIZE=2, MAXSIZE=100, FILEGROWTH=1 ),

FILEGROUP FG2

(NAME = 'Order\_Partition \_2', FILENAME = 'D:\PartitionDB\FG2\Order\_Partition\_2.ndf', SIZE = 2, MAXSIZE=100, FILEGROWTH=1 ),

FILEGROUP FG3

(NAME = 'Order\_Partition\_3', FILENAME='D:\PartitionDB\FG3\Order\_Partition\_3.ndf', SIZE = 2, MAXSIZE=100, FILEGROWTH=1 )

**GO**

**Bước 2: Tạo Partition Function**

* Tạo các giá trị biên cho mỗi bảng dựa vào cột Date trong bảng chính (Order)
* Hàm Partition function ‘OrderDatePartition’ tạo được 3 bảng nhỏ chia dữ liệu theo các mốc thời gian: trước ngày 1/1/2020; từ 1/1/2020 đến 31/12/2020; từ 1/1/2021 trở về sau.
* RANGE RIGHT nghĩa là lấy < or >=

LEFT lấy <= and >

|  |  |
| --- | --- |
| <1/1/2020 | Order\_Partition\_1 |
| >=1/1/2020 and <31/12/2020 | Order\_Partition\_2 |
| >=1/1/2021 | Order\_Partition\_3 |

**Use** DB\_Order\_Partition

**GO**

**CREATE PARTITION FUNCTION** OrderDatePartition (datetime)

AS

RANGE RIGHT

FOR VALUES ( '2020-01-01', '2021-01-01')

**GO**

**Bước 3: Tạo partition schema**

* Hàm Partition Scheme ‘OrderDatePartitionScheme’ dùng hàm ‘OrderDatePartition’ để “gắn” các đoạn vào từng filegroup

**Use** DB\_Order\_Partition

**GO**

**CREATE PARTITION SCHEME** OrderDatePartitionScheme

AS

PARTITION OrderDatePartition

TO ([PRIMARY], FG2, FG3 )

**GO**

**Bước 4: Tạo partitioned table**

* Tạo một bảng mới từ Partition Scheme, bảng này sẽ lọc ra những dữ liệu cần tìm kiếm nhiều khi truy vấn gồm mã đơn, ngày lập hoá đơn và thành tiền.
* Vì ngày lập hoá đơn không phải khoá chính nên sẽ phải tạo thêm một index cho bảng mới theo cột ngày lập (date) để truy vấn được dễ dàng và nhanh chóng hơn.

**Use** DB\_Order\_Partition

**GO**

**CREATE TABLE** Order\_Partition\_Table

(

**IdOrder** nvarchar(10) PRIMARY KEY,

**date** datetime,

**total\_money** int(15)

)

**ON** OrderDatePartitionScheme (date);

**GO**

**CREATE INDEX** idx\_date\_order\_partition

**ON** Order\_Partition\_Table (date);

**GO**