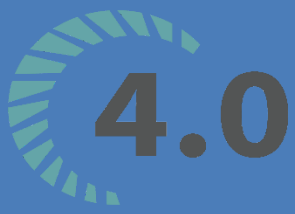
BỘ MÔN HỆ THỐNG THÔNG TIN – KHOA CÔNG NGHỆ THÔNG TIN

ĐẠI HỌC KHOA HỌC TỰ NHIÊN THÀNH PHỐ HỒ CHÍ MINH, ĐẠI HỌC QUỐC GIA TP HCM

MÔN CƠ SỞ DỮ LIỆU NÂNG CAO



Sinh viên thực hiện: 22127115 - Trần Trung Hiếu

22127138 - Đào Ngọc Hưng

22127329 - Phạm Ngọc Phú

22127335 - Nguyễn Trọng Phúc

22127354 - Phan Vũ Anh Quang

GV phụ trách: Cơ sở dữ liệu nâng cao, Tiết Gia Hồng

ĐỒ ÁN/BÀI TẬP MÔN HỌC - CƠ SỞ DỮ LIỆU NÂNG CAO

HỌC KỲ I – NĂM HỌC 2024-2025

BẢNG THÔNG TIN CHI TIẾT NHÓM

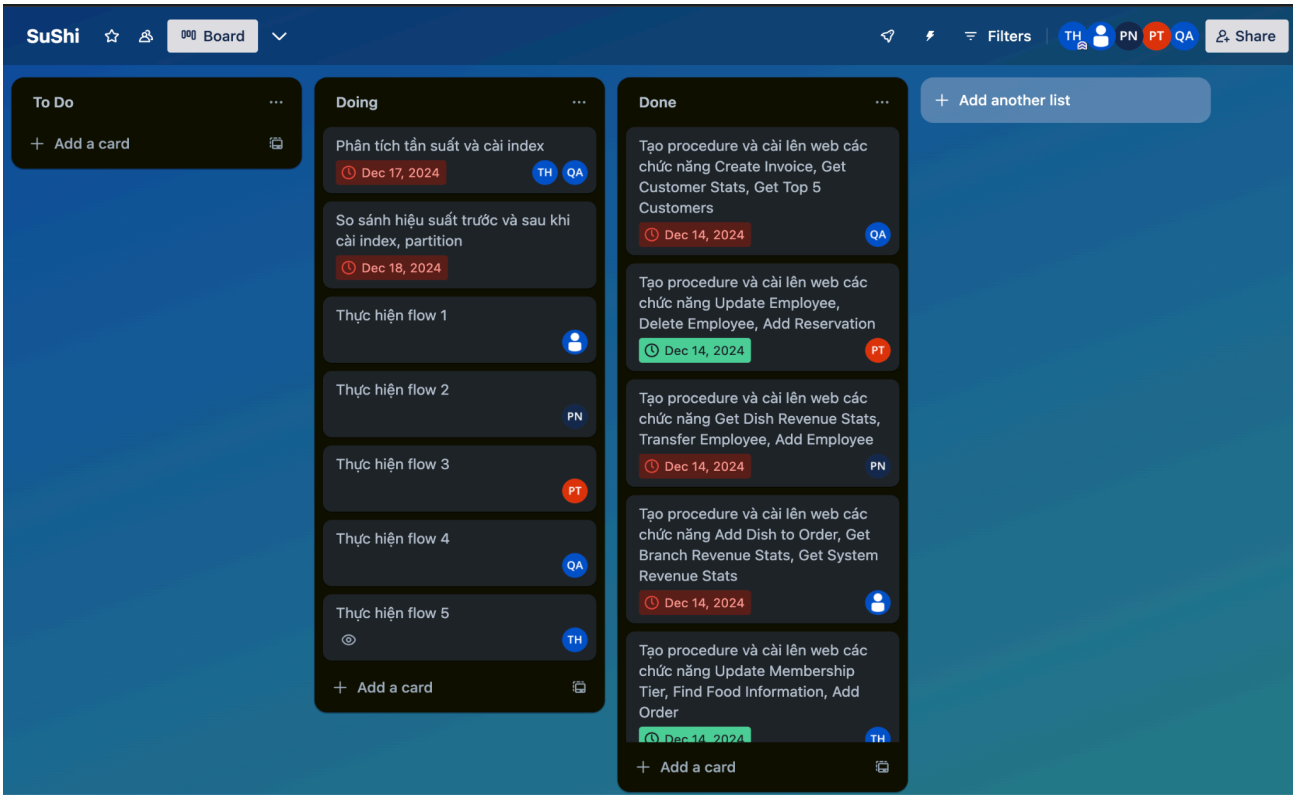
Mã nhóm:	11		
Tên nhóm:	Mười một		
Số lượng:	**5**		
MSSV	**Họ tên**	**Email**	**Điện thoại**
22127115	Trần Trung Hiếu	tthieu22@clc.fitus.edu.vn	
22127138	Đào Ngọc Hưng	dnhung22@clc.fitus.edu.vn	
22127329	Phạm Ngọc Phú	pnphu22@clc.fitus.edu.vn	
22127335	Nguyễn Trọng Phúc	ntphuc22@clc.fitus.edu.vn	
22127354	Phan Vũ Anh Quang	pvaquang22@clc.fitus.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**
Thiết kế CSDL mức quan niệm (ER)	Cả nhóm	100%	10/10
Chuyển qua lược đồ quan hệ, đánh giá dạng chuẩn "Nhân viên"	22127138 - Đào Ngọc Hưng	100%	10/10
Chuyển qua lược đồ quan hệ, đánh giá dạng chuẩn "Hóa đơn, phiếu, đánh giá"	22127115 - Trần Trung Hiếu	100%	10/10
Chuyển qua lược đồ quan hệ, đánh giá dạng chuẩn "Khách hàng offline"	22127354 - Phan Vũ Anh Quang	100%	10/10
Chuyển qua lược đồ quan hệ, đánh giá dạng chuẩn "Khách hàng online"	22127329 - Phạm Ngọc phú	100%	10/10
Chuyển qua lược đồ quan hệ, đánh giá dạng chuẩn "Menu"	22127335 - Nguyễn Trọng Phúc	100%	10/10
Bổ sung ràng buộc	22127115 - Trần Trung Hiếu	100%	10/10
Bổ sung chi tiết quy trình	22127354 - Phan Vũ Anh Quang	100%	10/10

Kiểm tra dạng chuẩn và nâng dạng chuẩn (nếu cần thiết)	22127138 - Đào Ngọc Hưng	100%	10/10
Thực hiện quy trình Phục vụ nhân viên (cài procedure, chức năng web, phân tích hiệu suất index)	22127138 - Đào Ngọc Hưng	100%	10/10
THực hiện quy trình trải nghiệm khách hàng (cài procedure, chức năng web, phân tích hiệu suất index)	22127329 - Phạm Ngọc phú	100%	10/10
Thực hiện quy trình quản lý doanh thu hệ thống (cài procedure, chức năng web, phân tích hiệu suất index)	22127335 - Nguyễn Trọng Phúc	100%	10/10
Thực hiện quy trình thống kê và quản lý khách hàng (cài procedure, chức năng web, phân tích hiệu suất index)	22127354 - Phan Vũ Anh Quang	100%	10/10
Thực hiện quy trình quản lý nhân sự (cài procedure, chức năng web, phân tích hiệu suất index)	22127115 - Trần Trung Hiếu	100%	10/10
Phân tích truy vấn	22127115 - Trần Trung Hiếu	100%	10/10
Cài index	22127354 - Phan Vũ Anh Quang	100%	10/10
Set up web	22127138 - Đào Ngọc Hưng	100%	10/10

Thêm UI	22127354 - Phan Vũ Anh Quang 22127138 - Đào Ngọc Hưng	100%	10/10
Báo cáo	22127115 - Trần Trung Hiếu 22127329 - Phạm Ngọc phú	100%	10/10

Công Cụ Quản Lý Công Việc: Trello



YÊU CẦU ĐỒ ÁN- BÀI TẬP

Loại bài tập	**□ Lý thuyết • Thực hành • Đồ án □ Bài tập**
Ngày bắt đầu	**02/11/2024**
Ngày kết thúc	**02/1/2024**

A. Yêu cầu của Đồ án/Bài tập

I. Mô tả:

1. **Quy trình Quản lý Nhân sự: Thêm, Cập nhật, Xóa và Điều Chuyển Nhân Viên**

   Quy trình **Quản lý nhân viên** là một phần quan trọng trong hệ thống quản lý của chuỗi nhà hàng SushiX, nhằm đảm bảo việc tổ chức và vận hành đội ngũ nhân viên hiệu quả. Quy trình này tập trung vào các thao tác liên quan đến việc quản lý thông tin nhân sự và lịch sử làm việc tại các chi nhánh.

2. **Quy trình xem thông tin chi nhánh và tìm kiếm món ăn: thể hiện danh sách chi nhánh, thông tin của từng chi nhánh, xem thực đơn của từng chi nhánh, tìm kiếm món ăn và giá.**

   Quy trình **xem thông tin chi nhánh và tìm kiếm món ăn** là một phần quan trọng trong hệ thống quản lý của chuỗi nhà hàng SushiX, nhằm cho phép người dùng khi vào web có thể nhìn thấy các chi nhánh khác của hệ thống nhà hàng SushiX. Người dùng có thể xem thông tin của từng chi nhánh như tên chi nhánh, địa chỉ, giờ mở/đóng cửa, bãi đổ xe máy/ ô tô, khu vực. Sau đó người dùng có thể xem thực đơn của từng chi nhánh, sau đó chọn món ăn và hiển thị giá món trong hệ thống nhà hàng SushiX.

3. **Quy trình quản lý doanh thu: quản lý doanh thu theo hệ thống, quản lý doanh thu theo chi nhánh và quản lý doanh thu theo món.**

   Quy trình **quản lý doanh thu** là một phần quan trọng trong hệ thống quản lý của chuỗi nhà hàng SushiX, giúp người quản lý theo dõi và đánh giá hiệu quả hoạt động kinh doanh. Chức năng này bao gồm việc quản lý doanh thu toàn hệ thống, doanh thu từng chi nhánh, và doanh thu theo món ăn.

4. **Quy trình đặt bàn: thực hiện đặt bàn trước thông qua web.**

   Quy trình **đặt bàn** là một phần quan trọng trong hệ thống quản lý của chuỗi nhà hàng SushiX, nhằm cho phép người dùng khi vào web có thể đặt bàn ở một chi nhánh trong hệ thống nhà hàng SushiX.

5. **Quy trình thống kê: thực hiện thống kê top 5 khách hàng chi tiêu nhiều nhất mỗi chi nhánh, thống kê lượt phục vụ.**

   Quy trình **Thống kê** là một phần quan trọng trong hệ thống quản lý của chuỗi nhà hàng SushiX, nhằm cho phép người quản lý của chi nhánh hay hệ thống có thể thống kê như

top 5 khách hàng chi tiêu nhiều nhất để xem khách hàng thân thiết và tạo các giảm giá riêng cho họ hoặc thống kê lượt phục vụ trong hệ thống nhà hàng SushiX.

6. **Quy trình cập nhật hạng thẻ:**

Quy trình **cập nhật hạng thẻ** là một phần quan trọng trong hệ thống quản lý của chuỗi nhà hàng SushiX, nhằm cho phép người quản lý của chi nhánh hay hệ thống có thể cập nhật hạng thẻ cho người dùng trong hệ thống nhà hàng SushiX.

7. **Quy trình phục vụ khách hàng của nhân viên:**

Quy trình **phục vụ khách hàng** là một phần quan trọng trong hệ thống quản lý của chuỗi nhà hàng SushiX, được thiết kế để hỗ trợ nhân viên nhà hàng thực hiện các nhiệm vụ phục vụ khách hàng một cách chuyên nghiệp, hiệu quả và chính xác. Từ khi khách hàng bước vào nhà hàng đến khi hoàn tất thanh toán, nhân viên sử dụng hệ thống để xử lý các công việc cụ thể theo từng bước, đảm bảo trải nghiệm khách hàng tốt nhất.

II. Danh sách các yêu cầu chức năng

1. **Quy trình Quản lý Nhân sự: Thêm, Cập nhật, Xóa và Điều Chuyển Nhân Viên**
   - Đăng nhập **(sp_AuthenticateUser)**
   - Quản lý thêm nhân viên **(sp_AddEmployee)**
   - Cập nhật thông tin nhân viên **(sp_UpdateEmployee)**
   - Xóa thông tin nhân viên **(sp_DeleteEmployee)**
   - Điều chuyển nhân viên **(sp_TransferEmployee)**
2. **Quy trình Quản lý Doanh thu: Quản lý doanh thu theo hệ thống, chi nhánh và khu vực**
   - Đăng nhập **(sp_AuthenticateUser)**
   - Quản lý doanh thu theo hệ thống **(sp_GetSystemRevenueStats)**
   - Quản lý doanh thu theo chi nhánh **(sp_GetBranchRevenueStats)**
   - Quản lý doanh thu theo món ăn **(sp_GetDishRevenueStats)**
3. **Quy trình xem thông tin chi nhánh và tìm kiếm món ăn: thể hiện danh sách chi nhánh, thông tin của từng chi nhánh, xem thực đơn của từng chi nhánh, tìm kiếm món ăn và giá.**
   - Truy cập vào trang chủ.
   - Bấm vào chức năng danh sách chi nhánh.
   - Sẽ chuyển hướng tới trang danh sách chi nhánh.
   - Xem thông tin của từng chi nhánh.
   - Bấm vào chức năng thực đơn.
   - Sẽ chuyển hướng tới trang để xem thực đơn, ta có thể lọc theo chi nhánh phục vụ, khu vực, mục thực đơn. (**sp_FindFoodInformationFromRegionOrBranch**).

- Xem thông tin của tên món, giá tiền, chi nhánh phục vụ món ăn đó.

4. **Quy trình đặt bàn: thực hiện đặt bàn trước.**
   - Đăng nhập vào với chức năng người dùng/nhân viên **(sp_AuthenticateUser)**
   - Chọn chi nhánh muốn đặt bàn. (**sp_AddReservation**)
   - Chọn bàn muốn đặt.
   - Chọn ngày đặt, giờ đến.
   - Điền số lượng khách đến.
   - Điền ghi chú nếu có.
5. **Quy trình thống kê: thực hiện thống kê top 5 khách hàng chi tiêu nhiều nhất mỗi chi nhánh, thống kê lượt phục vụ.**
   - Đăng nhập vào với chức năng quản lý **(sp_AuthenticateUser).**
   - Chọn chức năng thống kê top 5 khách hàng trong phần khách hàng (**sp_GetTop5CustomersByBranch**).
   - Về trang chủ quản lý.
   - Chọn chức năng thống kê lượt phục vụ **(sp_GetCustomerStatsFromInvoice).**
   - Ta sẽ có các thống kê.
6. **Quy trình cập nhật hạng thẻ:**
   - Đăng nhập vào với chức năng quản lý **(sp_AuthenticateUser).**
   - Chọn chức năng cập nhật hạng thẻ **(sp_UpdateMembershipTiers)**.(chức năng cập nhật hạng thẻ theo đúng logic ta sẽ dùng **server agent** trong **SSMS** để tạo job thực hiện stored procedure tự động 1 năm 1 lần nhưng phiên bản free hiện tại không cung cấp sử dụng **sql server agent** nên nhóm cho nó thành 1 chức năng và quản lý sẽ thực hiện mỗi cuối năm).
   - Hiển thị thông báo cập nhật thành công.
7. **Quy trình phục vụ khách hàng của nhân viên:**
   - Đăng nhập vào với chức năng người dùng/nhân viên (**sp_AuthenticateUser**).
   - Chỉnh trạng thái bàn (trống/đang phục vụ).
   - Tạo phiếu đặt món (**sp_addOrder**).
   - Thêm món vào phiếu đặt món (**sp_addDishToOrder**)
   - Tạo hóa đơn thanh toán (**sp_CreateInvoice**)

III. Danh sách các yêu cầu phi chức năng

1. **Hiệu Suất:**
   - Hệ thống phải xử lý tối đa 100.000 dòng dữ liệu cho một số bảng quan trọng.
   - Thời gian phản hồi các truy vấn tìm kiếm và thống kê phải dưới 2 giây.
2. **Tính Nhất Quán:**
   - Đảm bảo dữ liệu luôn nhất quán qua các giao dịch và các bảng liên quan.

- Kiểm tra ràng buộc khóa ngoại để duy trì tính toàn vẹn của dữ liệu.

3. **Tối Ưu Hóa Truy Vấn:**
   - Sử dụng các giải pháp tối ưu hóa truy vấn như chỉ mục, phân vùng dữ liệu để cải thiện hiệu suất truy vấn.
   - Thực hiện các thử nghiệm hiệu suất để đảm bảo hệ thống đáp ứng yêu cầu về tốc độ và khả năng xử lý dữ liệu lớn.

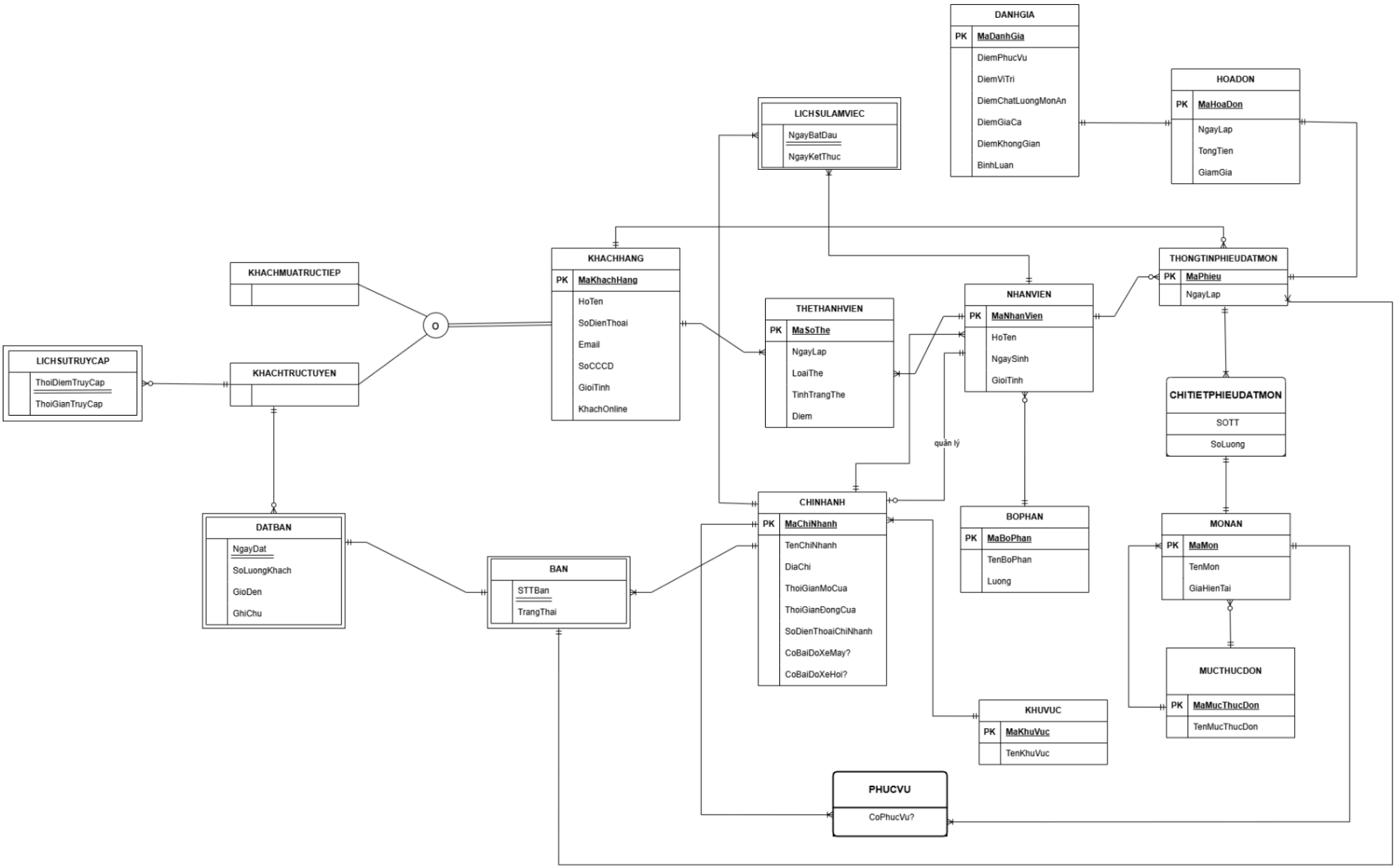
B. Kết quả

I. Thiết kế dữ liệu quan niệm

Link draw.io:
https://drive.google.com/file/d/1rdoXm6dZjwhYq41oEuyqMBtERk1ZcNHt/view?usp=sharing

KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
227 Nguyễn Văn Cừ, Phường 4, Quận 5, TP.HCM
Điện Thoại: (08) 38.354.266 - Fax:(08) 38.350.096

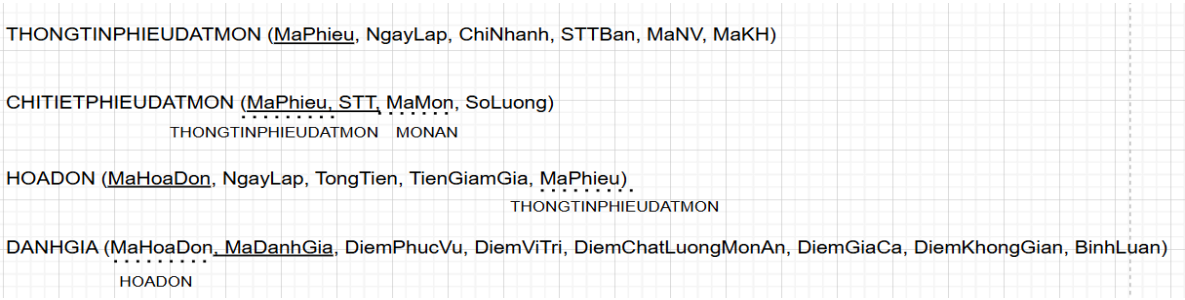
4.0



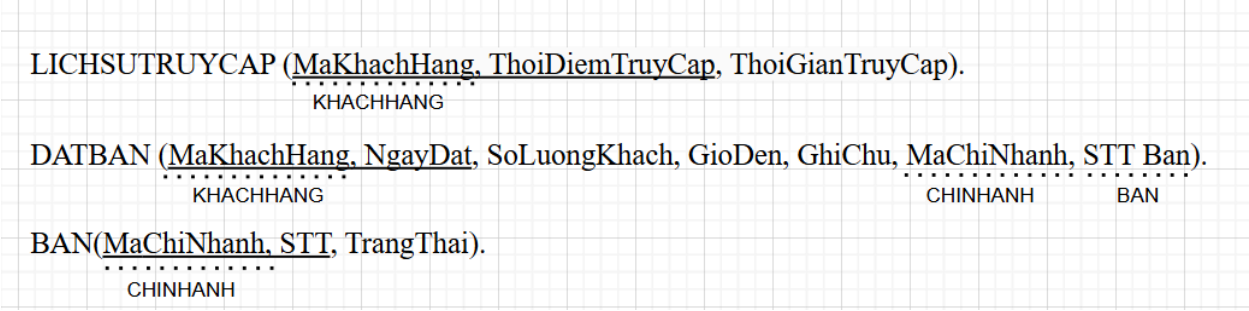
II. Thiết kế dữ liệu logic

1. Chuyển lược đồ quan hệ

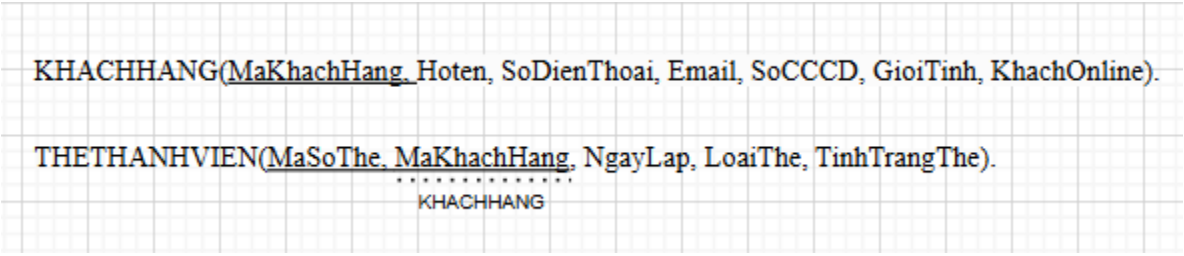
Hóa đơn, phiếu, đánh giá (thongtinphieudatmon, chitietphieudatmon, hoadon, danhgia)



Khách online (lichsutruycap, datban, ban)

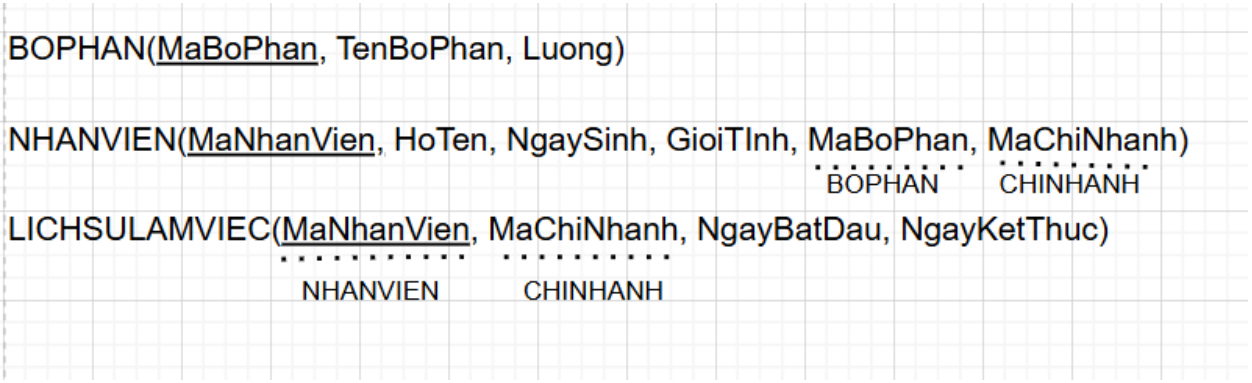


Khách hàng offline (khachhang, thethanhvien)



Menu (monan, mucthucdon, phucvu, chinhanh, khuvuc)

Nhân viên (nhanvien, bophan, lichsulamviec)



2. Bổ sung các ràng buộc không biểu diễn được trên lược đồ.

- Ràng buộc về nâng hạng, giữ hạng, và hạ hạng thẻ thành viên dựa trên tổng giá trị tiêu dùng tích lũy trong khoảng thời gian 1 năm:
  - **Chi tiết:**
  - Để nâng hạng lên thẻ SILVER hoặc GOLD, khách hàng phải đạt tổng giá trị tiêu dùng tích lũy nhất định trong 1 năm kể từ ngày đạt hạng hiện tại.
  - Để giữ hạng, khách hàng cũng phải đạt mức tiêu dùng tích lũy trong 1 năm.
  - Nếu không đạt, thẻ sẽ hạ hạng hoặc trở về mức ban đầu
- Ràng buộc khách hàng không có nhiều hơn một thẻ thành viên đang hoạt động:
- Mỗi nhân viên chỉ có thể làm việc tại một chi nhánh vào một thời điểm:
- Thực đơn các chi nhánh cùng khu vực phải giống nhau.

3. Dạng chuẩn

3.1. Bảng THONGTINPHIEUDATMON

Thuộc tính:

- Khóa chính: MaPhieu
- Các thuộc tính: NgayLap, ChiNhanh, STTBan, MaNV, MaKH

Phụ thuộc hàm:

- MaPhieu → NgayLap, ChiNhanh, STTBan, MaNV, MaKH

Dạng chuẩn:

- Đạt **1NF** vì các giá trị là nguyên tử.
- Đạt **2NF** vì mọi thuộc tính không khóa phụ thuộc hoàn toàn vào khóa chính MaPhieu.
- Đạt **3NF** vì không có phụ thuộc bắc cầu giữa các thuộc tính.

3.2. Bảng CHITIETPHIEUDATMON

Thuộc tính:

- Khóa chính: (MaPhieu, STT)
- Các thuộc tính: MaMon, SoLuong

Phụ thuộc hàm:

- MaPhieu, STT → MaMon, SoLuong

Dạng chuẩn:

- Đạt **1NF** vì các giá trị là nguyên tử.
- Đạt **2NF** vì các thuộc tính không khóa (MaMon, SoLuong) phụ thuộc hoàn toàn vào khóa chính (MaPhieu, STT).
- Đạt **3NF** vì không có phụ thuộc bắc cầu giữa các thuộc tính.

3.3. Bảng HOADON

Thuộc tính:

- Khóa chính: MaHoaDon
- Các thuộc tính: NgayLap, TongTien, TienGiamGia, MaPhieu

Phụ thuộc hàm:

- MaHoaDon → NgayLap, TongTien, TienGiamGia, MaPhieu

Dạng chuẩn:

- Đạt **1NF** vì các giá trị là nguyên tử.
- Đạt **2NF** vì các thuộc tính không khóa phụ thuộc hoàn toàn vào khóa chính MaHoaDon.
- Đạt **3NF** vì không có phụ thuộc bắc cầu giữa các thuộc tính.

3.4. Bảng DANHGIA

Thuộc tính:

- Khóa chính: MaHoaDon, MaDanhGia
- Các thuộc tính: DiemPhucVu, DiemViTri, DiemChatLuongMonAn, DiemGiaCa, DiemKhongGian, BinhLuan

Phụ thuộc hàm:

- MaHoaDon, MaDanhGia → DiemPhucVu, DiemViTri, DiemChatLuongMonAn, DiemGiaCa, DiemKhongGian, BinhLuan

Dạng chuẩn:

- Đạt **1NF** vì các giá trị là nguyên tử.
  - Đạt **2NF** vì các thuộc tính không khóa phụ thuộc hoàn toàn vào khóa chính (MaHoaDon, MaDanhGia).
  - Đạt **3NF** vì không có phụ thuộc bắc cầu giữa các thuộc tính.

3.5. Bảng LICHSUTRUYCAP

Thuộc tính:

- Khóa chính: (MaKhachHang, ThoiDiemTruyCap)
- Các thuộc tính: ThoiGianTruyCap

Phụ thuộc hàm:

- MaKhachHang, ThoiDiemTruyCap → ThoiGianTruyCap

Dạng chuẩn:

- Đạt **1NF** vì các giá trị là nguyên tử.
- Đạt **2NF** vì các thuộc tính không khóa phụ thuộc hoàn toàn vào khóa chính (MaKhachHang, ThoiDiemTruyCap).
- Đạt **3NF** vì không có phụ thuộc bắc cầu giữa các thuộc tính.

3.6. Bảng DATBAN

Thuộc tính:

- Khóa chính: (MaKhachHang, NgayDat)
- Các thuộc tính: SoLuongKhach, GioDen, GhiChu, MaChiNhanh, STTBan

Phụ thuộc hàm:

- MaKhachHang, NgayDat → SoLuongKhach, GioDen, GhiChu, MaChiNhanh, STTBan

Dạng chuẩn:

- Đạt **1NF** vì các giá trị là nguyên tử.
- Đạt **2NF** vì các thuộc tính không khóa phụ thuộc hoàn toàn vào khóa chính (MaKhachHang, NgayDat).
- Đạt **3NF** vì không có phụ thuộc bắc cầu giữa các thuộc tính.

3.7. Bảng KHACHHANG

Thuộc tính:

- Khóa chính: MaKhachHang
- Các thuộc tính: HoTen, SoDienThoai, Email, SoCCCD, GioiTinh, KhachOnline

Phụ thuộc hàm:

- MaKhachHang → HoTen, SoDienThoai, Email, SoCCCD, GioiTinh, KhachOnline

Dạng chuẩn:

- Đạt **1NF** vì các giá trị là nguyên tử.

- Đạt **2NF** vì các thuộc tính không khóa phụ thuộc hoàn toàn vào khóa chính MaKhachHang.
- Đạt **3NF** vì không có phụ thuộc bắc cầu giữa các thuộc tính.

3.8. Bảng NHANVIEN

Thuộc tính:

- Khóa chính: MaNhanVien
- Các thuộc tính: HoTen, NgaySinh, GioiTinh, MaBoPhan, MaChiNhanh

Phụ thuộc hàm:

- MaNhanVien → HoTen, NgaySinh, GioiTinh, MaBoPhan, MaChiNhanh

Dạng chuẩn:

- Đạt **1NF** vì các giá trị là nguyên tử.
- Đạt **2NF** vì các thuộc tính không khóa phụ thuộc hoàn toàn vào khóa chính MaNhanVien.
- Đạt **3NF** vì không có phụ thuộc bắc cầu giữa các thuộc tính.

3.9. Bảng BAN

Thuộc tính:

- Khóa chính: MaChiNhanh, STT
- Các thuộc tính: TrangThai

Phụ thuộc hàm:

- {MaChiNhanh, STT} → TrangThai

Dạng chuẩn:

- Đạt **1NF** vì các giá trị là nguyên tử.
- Đạt **2NF** vì mọi thuộc tính không khóa phụ thuộc hoàn toàn vào khóa chính {MaChiNhanh, STT}.
- Đạt **3NF** vì không có phụ thuộc bắc cầu giữa các thuộc tính.

3.10. Bảng THETHANHVIEN

Thuộc tính:

- Khóa chính: MaSoThe
- Các thuộc tính: MaKhachHang, NgayLap, LoaiThe, TinhTrangThe

Phụ thuộc hàm:

- MaSoThe → MaKhachHang, NgayLap, LoaiThe, TinhTrangThe

Dạng chuẩn:

- Đạt **1NF** vì các giá trị là nguyên tử.
- Đạt **2NF** vì mọi thuộc tính không khóa phụ thuộc hoàn toàn vào khóa chính MaSoThe.
- Đạt **3NF** vì không có phụ thuộc bắc cầu giữa các thuộc tính.

3.11. Bảng MONAN

Thuộc tính:

- Khóa chính: MaMon
- Các thuộc tính: MaMucThucDon, TenMon, GiaHienTai

Phụ thuộc hàm:

- MaMon → MaMucThucDon, TenMon, GiaHienTai

Dạng chuẩn:

- Đạt **1NF** vì các giá trị là nguyên tử.
- Đạt **2NF** vì mọi thuộc tính không khóa phụ thuộc hoàn toàn vào khóa chính MaMon.
- Đạt **3NF** vì không có phụ thuộc bắc cầu giữa các thuộc tính.

3.12. Bảng MUCTHUCDON

Thuộc tính:

- Khóa chính: MaMucThucDon
- Các thuộc tính: TenMucThucDon

Phụ thuộc hàm:

- MaMucThucDon → TenMucThucDon

Dạng chuẩn:

- Đạt **1NF** vì các giá trị là nguyên tử.
- Đạt **2NF** vì mọi thuộc tính không khóa phụ thuộc hoàn toàn vào khóa chính MaMucThucDon.
- Đạt **3NF** vì không có phụ thuộc bắc cầu giữa các thuộc tính.

3.13. Bảng PHUCVU

Thuộc tính:

- Khóa chính: MaChiNhanh, MaMon
- Các thuộc tính: CoPhucVu?

Phụ thuộc hàm:

- {MaChiNhanh, MaMon} → CoPhucVu?

Dạng chuẩn:

- Đạt **1NF** vì các giá trị là nguyên tử.
- Đạt **2NF** vì mọi thuộc tính không khóa phụ thuộc hoàn toàn vào khóa chính {MaChiNhanh, MaMon}.
- Đạt **3NF** vì không có phụ thuộc bắc cầu giữa các thuộc tính.

3.14. Bảng CHINHANH

Thuộc tính:

- Khóa chính: MaChiNhanh
- Các thuộc tính: MaKhuVuc, QuanLyChiNhanh, TenChiNhanh, DiaChi, ThoiGianMoCua, ThoiGianDongCua, SoDienThoaiChiNhanh, CoBaiDoXeMay?, CoBaiDoXeHoi?

Phụ thuộc hàm:

- MaChiNhanh → MaKhuVuc, QuanLyChiNhanh, TenChiNhanh, DiaChi, ThoiGianMoCua, ThoiGianDongCua, SoDienThoaiChiNhanh, CoBaiDoXeMay?, CoBaiDoXeHoi?

Dạng chuẩn:

- Đạt **1NF** vì các giá trị là nguyên tử.
- Đạt **2NF** vì mọi thuộc tính không khóa phụ thuộc hoàn toàn vào khóa chính MaChiNhanh.
- Đạt **3NF** vì không có phụ thuộc bắc cầu giữa các thuộc tính.

3.15. Bảng KHUVUC

Thuộc tính:

- Khóa chính: MaKhuVuc
- Các thuộc tính: TenKhuVuc

Phụ thuộc hàm:

- MaKhuVuc → TenKhuVuc

Dạng chuẩn:

- Đạt **1NF** vì các giá trị là nguyên tử.
- Đạt **2NF** vì mọi thuộc tính không khóa phụ thuộc hoàn toàn vào khóa chính MaKhuVuc.
- Đạt **3NF** vì không có phụ thuộc bắc cầu giữa các thuộc tính.

3.16. Bảng BOPHAN

Thuộc tính:

- Khóa chính: MaBoPhan
- Các thuộc tính: TenBoPhan, Luong

Phụ thuộc hàm:

- MaBoPhan → TenBoPhan, Luong

Dạng chuẩn:

- Đạt **1NF** vì các giá trị là nguyên tử.
- Đạt **2NF** vì mọi thuộc tính không khóa phụ thuộc hoàn toàn vào khóa chính MaBoPhan.
- Đạt **3NF** vì không có phụ thuộc bắc cầu giữa các thuộc tính.

3.17. Bảng LICHSULAMVIEC

Thuộc tính:

- Khóa chính: MaNhanVien, MaChiNhanh
- Các thuộc tính: NgayBatDau, NgayKetThuc

Phụ thuộc hàm:

- {MaNhanVien, MaChiNhanh} → NgayBatDau, NgayKetThuc

Dạng chuẩn:

- Đạt **1NF** vì các giá trị là nguyên tử.

- Đạt **2NF** vì mọi thuộc tính không khóa phụ thuộc hoàn toàn vào khóa chính {MaNhanVien, MaChiNhanh}.
- Đạt **3NF** vì không có phụ thuộc bắc cầu giữa các thuộc tính.

Tổng kết

Tất cả các bảng đều đạt **3NF** vì:

- Mỗi bảng thỏa mãn **1NF**: Các thuộc tính chứa giá trị nguyên tử.
- Mỗi bảng thỏa mãn **2NF**: Không có thuộc tính không khóa nào phụ thuộc một phần vào khóa chính.
- Mỗi bảng thỏa mãn **3NF**: Không có phụ thuộc bắc cầu giữa các thuộc tính không khóa.

III. Thiết kế vật lý (Thông tin tần suất, cài chỉ mục, phân tích, kiểm chứng hiệu quả chỉ mục, …)

1. Thông tin tần suất, phân tích chọn chỉ mục

Bảng phân tích truy vấn: Link

Dựa trên bảng truy vấn và tần suất truy vấn từ các **stored procedures**, nhận thấy rằng các bảng sau đây có tần suất truy vấn cao và kích thước dữ liệu lớn:

- **KHACHHANG**: Truy vấn tìm kiếm khách hàng dựa trên **MaKH**.
- **MONAN**: Truy vấn tìm kiếm thông tin món ăn dựa trên **MaMon**, **TenMon**.
- **THONGTINPHIEUDATMON**: Truy vấn dựa trên **MaPhieu**, **MaChiNhanh**, **STTBan**, và **NgayLap**.
- **CHITIETPHIEUDATMON**: Truy vấn dựa trên **MaPhieu**, **MaMon**.
- **HOADON**: Truy vấn tính toán doanh thu, lọc hóa đơn dựa trên **NgayLap**.

Giải pháp nhóm đề xuất:

Bảng	Chỉ mục áp dụng	Phân vùng dữ liệu
KHACHHANG	MaKH (Clustered Index)	Không áp dụng
MONAN	MaMon (Clustered Index), TenMon (Non-Clustered Index)	Không áp dụng
THONGTINPHIEUDATMON	MaPhieu(Clustered Index), (MaChiNhanh, STTBan)(Composite Index), NgayLap	Không áp dụng

	(Non-Clustered Index)	
CHITIETPHIEUDATMON	(MaPhieu, MaMon)(Composite Index)	Không áp dụng
HOADON	MaPhieu (Clustered Index), NgayLap (Non-clustered Index)	Không áp dụng

2. Phân tích, đánh giá và so sánh hiệu suất trước và sau khi cài index

2.1. Quy trình Quản lý Nhân sự: Thêm, Cập nhật, Xóa và Điều Chuyển Nhân Viên

Quy trình hiện tại sử dụng các bảng liên quan đến nhân sự như:

- NHANVIEN: Lưu trữ (Update, Insert, Read, Delete) thông tin nhân viên.
- LICHSULAMVIEC: Theo dõi lịch sử làm việc (Insert, Update) của nhân viên tại các chi nhánh.
- CHINHANH: Quản lý thông tin chi nhánh.

Dữ liệu các bảng trên không quá lớn, tần suất thực hiện các chức năng không lớn. Do đó, hiệu suất trước và sau cài index sẽ cho kết quả không khác biệt nhiều.

2.2. Quy trình xem thông tin chi nhánh và tìm kiếm món ăn: thể hiện danh sách chi nhánh, thông tin của từng chi nhánh, xem thực đơn của từng chi nhánh, tìm kiếm món ăn và giá.

Quy trình hiện tại sử dụng các bảng liên quan đến thông tin chi nhánh như:

- CHINHANH: Xem thông tin (Read) về các chi nhánh như tên chi nhánh, địa chỉ, giờ mở/đóng cửa, khu vực.
- MONAN: Xem danh sách (Read) các món ăn cùng giá và danh mục thực đơn tương ứng.
- CHINHANH_MON: Xem (Read) các món ăn của từng chi nhánh cụ thể, xác định món có phục vụ tại chi nhánh đó hay không.

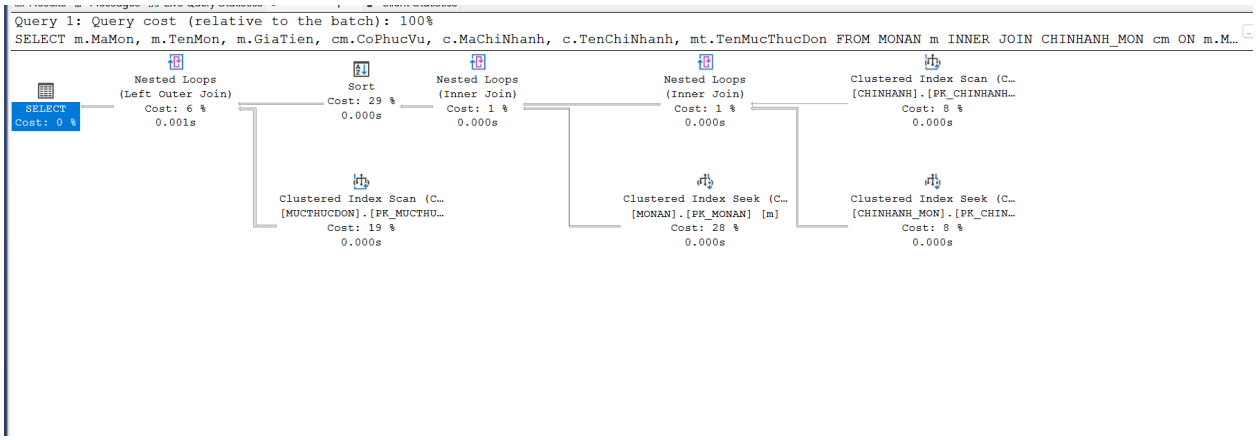
Do đó, có thể thực hiện so sánh hiệu suất trước và sau khi tối ưu vì dữ liệu hiện tại có dựa trên truy vấn tận dụng chỉ mục.

sp_FindFoodInformationFromRegionOrBranch

EXEC sp_FindFoodInformationFromRegionOrBranch '1', '1'

Có Index:

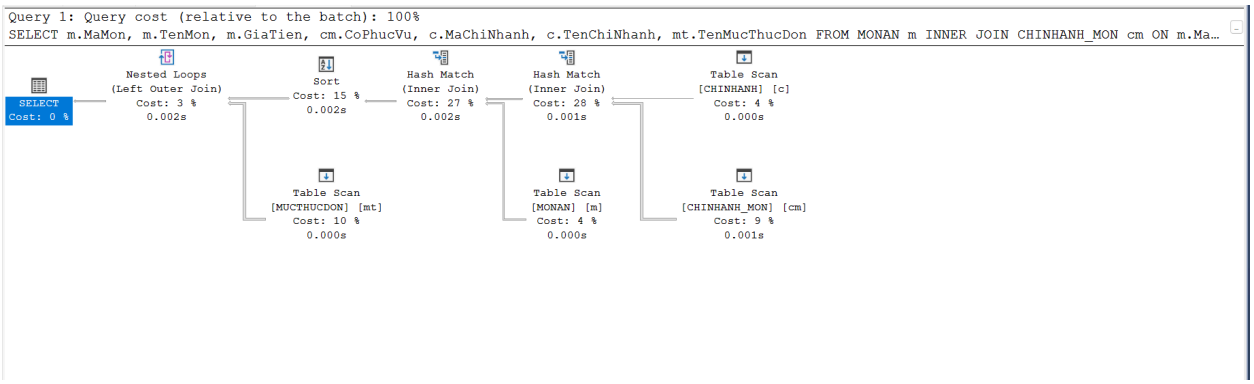
- Thời gian thực thi: 213ms.
- Kế hoạch thực thi:
    - Clustered Index Seek:
        - Được sử dụng trên bảng `MONAN`, `CHINHANH_MON`, và `CHINHANH`.
        - Hiệu quả: Clustered Index Seek giúp lọc nhanh dữ liệu từ khóa chính hoặc cột có index, giảm chi phí truy xuất I/O.
        - Chi phí trên từng bước:
            - `Clustered Index Seek` (bảng MONAN): Chi phí 28%.
            - `Clustered Index Seek` (bảng CHINHANH_MON): Chi phí 8%.
            - `Clustered Index Seek` (bảng CHINHANH): Chi phí 8%.
    - Nested Loop:
        - Được sử dụng để kết hợp dữ liệu giữa các bảng (`MONAN`, `CHINHANH_MON`, và `CHINHANH`).
        - Nested Loop là hiệu quả khi kích thước dữ liệu nhỏ và các bảng được lọc bởi Index Seek.
- Hiệu quả:
    - Việc sử dụng Clustered Index Seek và Nested Loop giúp giảm thời gian thực thi đáng kể.
    - Truy vấn tận dụng các index hiện có, tối ưu hiệu suất truy cập dữ liệu.



Không Có Index:

- Thời gian thực thi: 206ms.
- Kế hoạch thực thi:
    - Table Scan:
        - Các bảng `MONAN`, `CHINHANH_MON`, `CHINHANH` và `MUCHUCDON` thực hiện quét toàn bộ bảng.

        - Hạn chế: Không có index dẫn đến việc phải quét toàn bộ dữ liệu, tốn kém tài nguyên và tăng chi phí I/O.
    - Hash Match:
        - Được sử dụng thay thế cho Nested Loop. Tuy nhiên, Hash Match thường có chi phí cao hơn khi xử lý dữ liệu lớn hoặc không có index hỗ trợ.
- Hạn chế:
    - Table Scan làm tăng lượng dữ liệu đọc từ đĩa và làm tăng chi phí CPU.
    - Hiệu suất bị giới hạn khi không có cơ chế lọc dữ liệu hiệu quả.



Kết luận

Hiệu quả của Index:

Mặc dù thời gian thực thi không có sự chênh lệch lớn (213ms so với 206ms), điều này có thể do dataset hiện tại tương đối nhỏ.

Với dữ liệu lớn hơn, sự khác biệt sẽ trở nên rõ ràng hơn khi Clustered Index Seek mang lại hiệu suất cao hơn nhiều so với Table Scan.

2.3. Quy trình quản lý doanh thu: quản lý doanh thu theo hệ thống, quản lý doanh thu theo chi nhánh và quản lý doanh thu theo món.

Quy trình hiện tại sử dụng các bảng liên quan đến quản lý doanh thu như:

- CHINHANH: Xem thông tin (Read) về các chi nhánh để quản lý theo chi nhánh.
- THONGTINPHIEUDATMON: Xem thông tin (Read) phiếu đặt món để tiến hành quản lý theo hệ thống và chi nhánh.
- CHITIETPHIEUDATMON: Xem thông tin (Read) chi tiết các món ăn trong phiếu đặt món để tiến hành quản lý theo món ăn.
- MONAN: Xem thông tin (Read) về món ăn.
- HOADON: Xem thông tin (Read) hóa đơn.

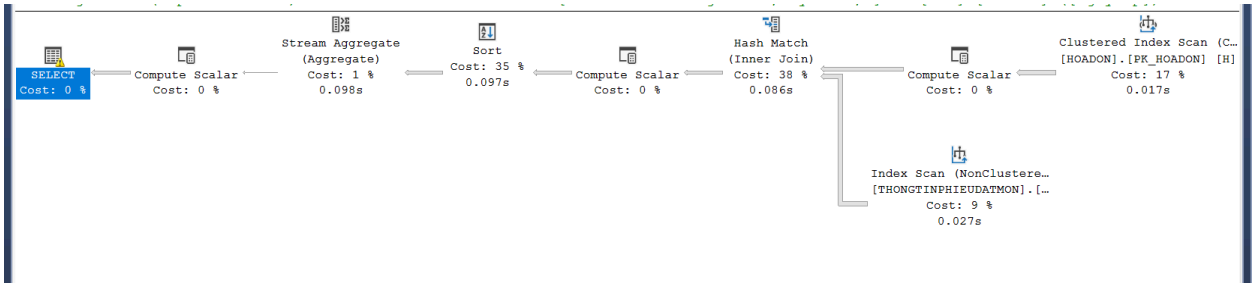
Do đó, có thể thực hiện so sánh hiệu suất trước và sau khi tối ưu vì dữ liệu hiện tại có dựa trên truy vấn tận dụng chỉ mục.

sp_GetSystemRevenueStats

EXEC sp_GetSystemRevenueStats '2023-01-01 00:00:00','2023-01-15 00:00:00', 'DAY'

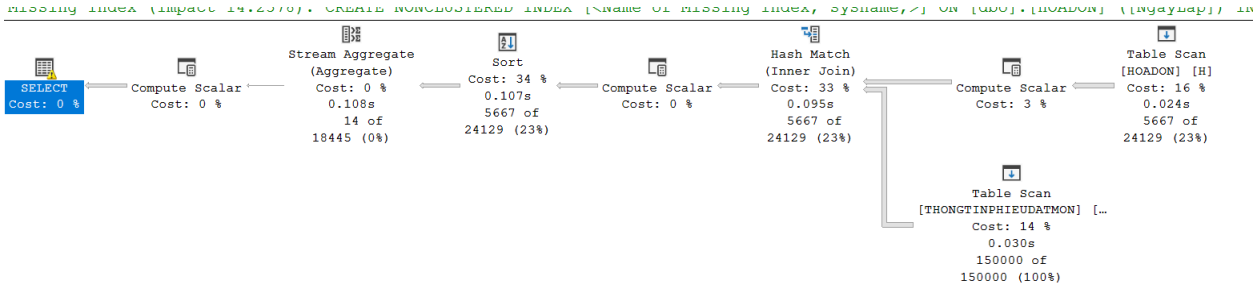
Có Index

- **Thời gian:** 246ms.
- **Kế hoạch thực thi:**
  - **I/O thấp:** Sử dụng `Index Scan` và `Clustered Index Scan` giúp lọc dữ liệu hiệu quả.
  - **Tiêu thụ tài nguyên:** Ít tài nguyên CPU/I/O, tốt cho truy vấn định kỳ.



Không có Index

- **Thời gian:** 260ms.
- **Kế hoạch thực thi:**
  - **I/O cao:** Dùng `Table Scan`, tốn tài nguyên I/O và CPU.



Kết luận: Với tần suất 7 ngày/lần, thời gian thực thi có thể không quá chênh lệch. Tuy nhiên, có index trên cột NgayLap vẫn là lựa chọn tốt vì:

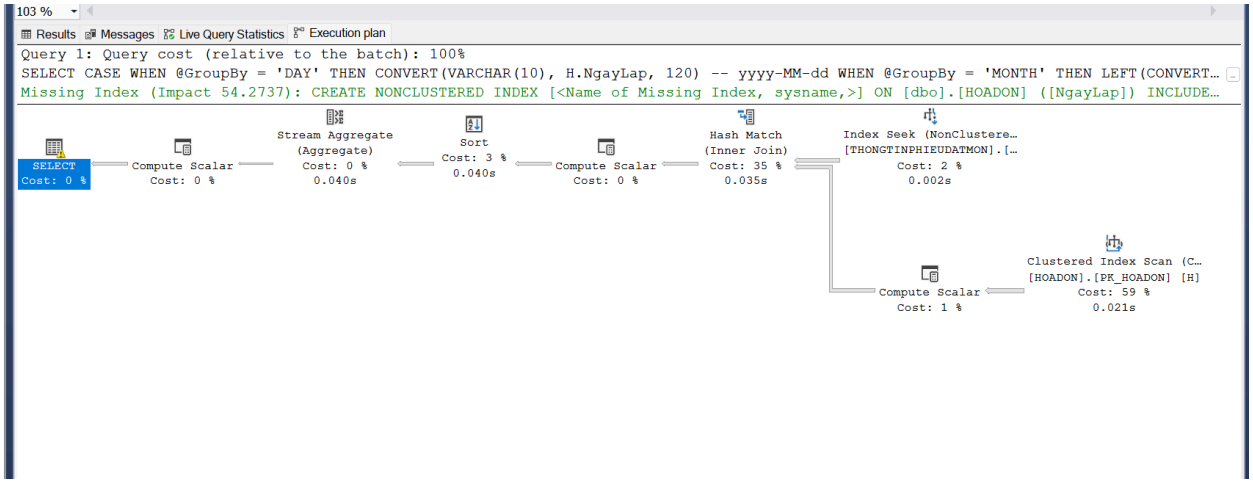
- Dữ liệu tăng trưởng: Index sẽ giúp duy trì hiệu suất khi lượng dữ liệu lớn dần.
- Tài nguyên hệ thống: Giảm I/O và CPU, tránh ảnh hưởng đến các tác vụ khác khi hệ thống chạy nhiều tác vụ đồng thời.

sp_GetBranchRevenueStats

EXEC sp_GetBranchRevenueStats '1', '2023-01-01 00:00:00','2023-03-01 00:00:00', 'DAY'

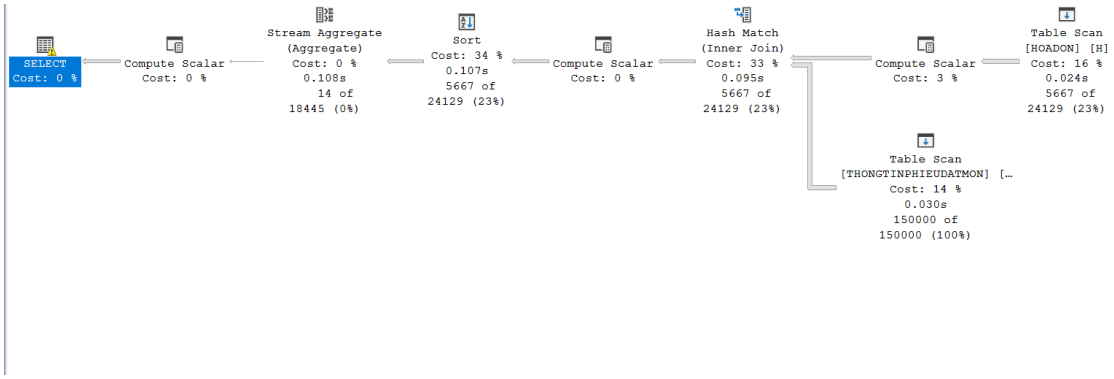
Có Index

- **Thời gian:** 152ms.
- **Kế hoạch thực thi:**
  - **Clustered Index Scan:** Quét `PK_HOADON` (59% chi phí).
  - **Non-Clustered Index Seek:** Tìm nhanh trên `THONGTINPHIEUDATMON` (2% chi phí).
  - Giảm tải I/O và CPU, tối ưu khi lọc theo `NgayLap`.



Không có Index

- **Thời gian:** 223ms.
- **Kế hoạch thực thi:**
  - **Table Scan:** Quét toàn bộ bảng, chi phí cao.
  - Tăng tải tài nguyên, hiệu suất giảm khi dữ liệu lớn.



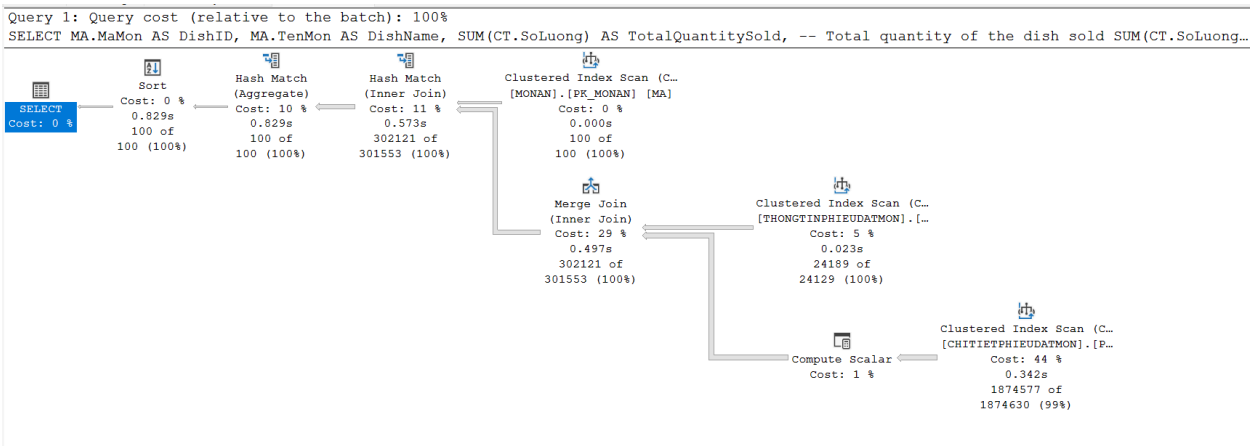
Kết luận: Có index cải thiện hiệu suất rõ rệt, giảm thời gian thực thi từ 223ms xuống 152ms (giảm 31.8%), giảm tải tài nguyên I/O và CPU. Khuyến nghị duy trì index hiện tại

sp_GetDishRevenueStats

EXEC sp_GetDishRevenueStats '2023-01-01 00:00:00', '2023-02-01 00:00:00'

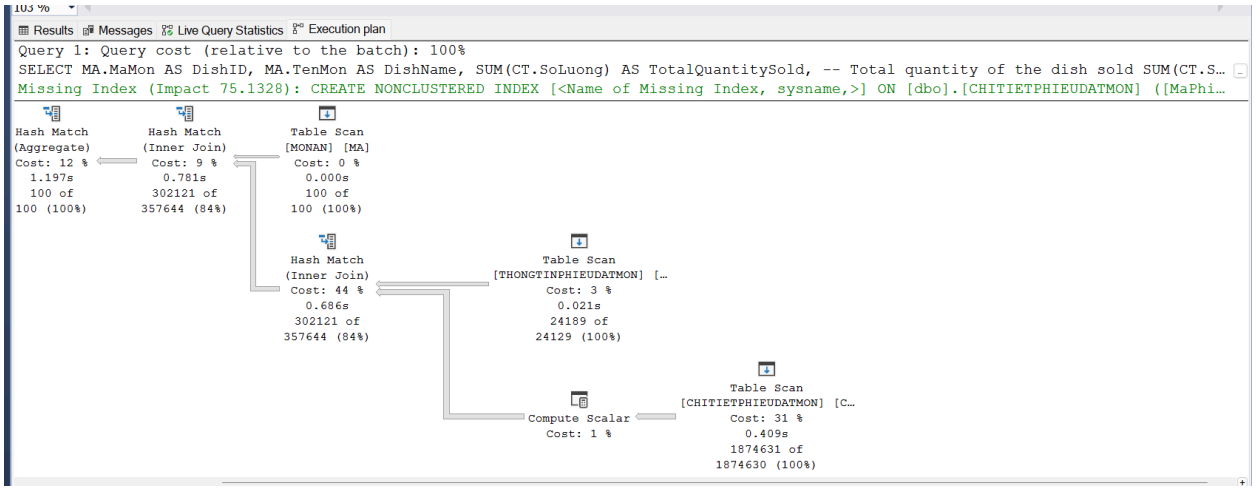
Có Index

- **Thời gian:** 1102ms.
- **Kế hoạch thực thi:**
  - **Clustered Index Scan:** Quét `PK_MONAN` trên bảng `MONAN` (0% chi phí).
  - **Clustered Index Scan:** Truy xuất dữ liệu nhanh trên `CHITIETPHIEUDATMON` (44% chi phí).
  - **Clustered Index Scan:** Truy xuất dữ liệu từ `THONGTINPHIEUDATMON` (5% chi phí).
  - **Merge Join và Hash Match:** Kết hợp dữ liệu giữa các bảng với tổng chi phí lớn nhất (29% và 12%).
  - **Hiệu quả:** Index hỗ trợ giảm thời gian và chi phí I/O so với trường hợp không có index.



Không có Index

- **Thời gian:** 2757ms.
- **Kế hoạch thực thi:**
  - **Table Scan:** Quét toàn bộ bảng `CHITIETPHIEUDATMON`, `MONAN`, và `THONGTINPHIEUDATMON`.
  - **Hash Match:** Kết hợp dữ liệu giữa các bảng, chi phí cao do không có bộ lọc trước.
  - **Hạn chế:**
    - Không có index dẫn đến quét toàn bộ bảng, tốn tài nguyên I/O và CPU.
    - Thời gian thực thi dài gấp **2.5 lần** so với khi có index.



Kết luận: Có index giúp giảm thời gian thực thi từ **2757ms** xuống còn **1102ms** (giảm 60%). Việc sử dụng `Clustered Index Scan` và tránh `Table Scan` tiết kiệm đáng kể tài nguyên

I/O và CPU. Nên duy trì các index hiện tại trên bảng CHITIETPHIEUDATMON và THONGTINPHIEUDATMON.

2.4. Quy trình đặt bàn: thực hiện đặt bàn trước.

Quy trình hiện tại sử dụng các bảng liên quan đến việc thực hiện đặt bàn như:

- CHINHANH: Xem thông tin (Read) các chi nhánh để tiến hành đặt bàn.
- KHACHHANG: Xem và quản lý thông tin khách hàng (Read) để liên kết với việc thực hiện đặt bàn.
- BAN: Xem và quản lý thông tin (Read) về các bàn tại mỗi chi nhánh, trong đó cần xác nhận và kiểm tra trạng thái của bàn để tiến hành thực hiện đặt bàn.
- DATBAN: Thêm và lưu trữ thông tin (Insert) về việc thực hiện đặt bàn của khách hàng.

Do đó, có thể thực hiện so sánh hiệu suất trước và sau khi tối ưu vì dữ liệu hiện tại có dựa trên truy vấn tận dụng chỉ mục.

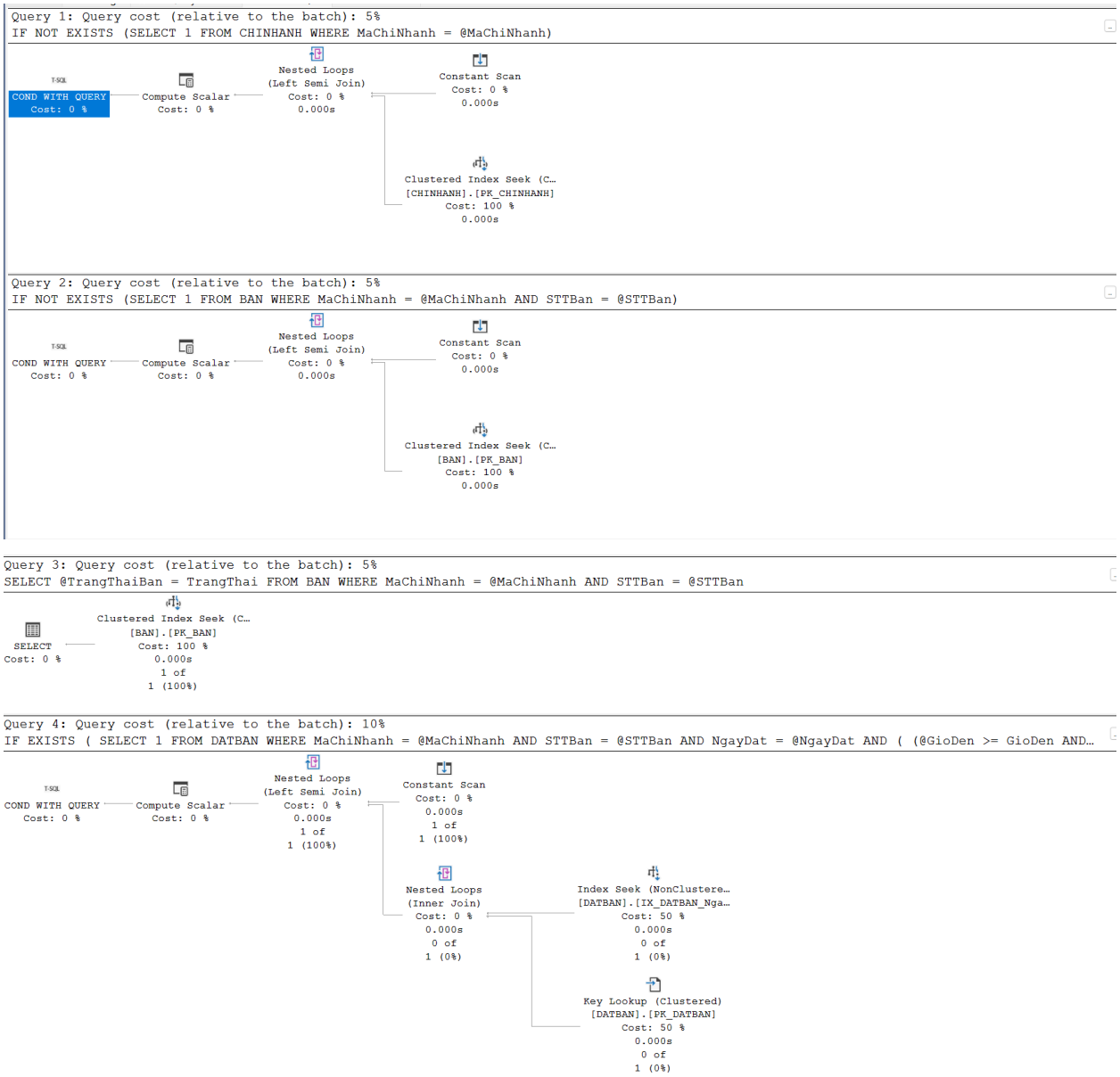
sp_AddReservation

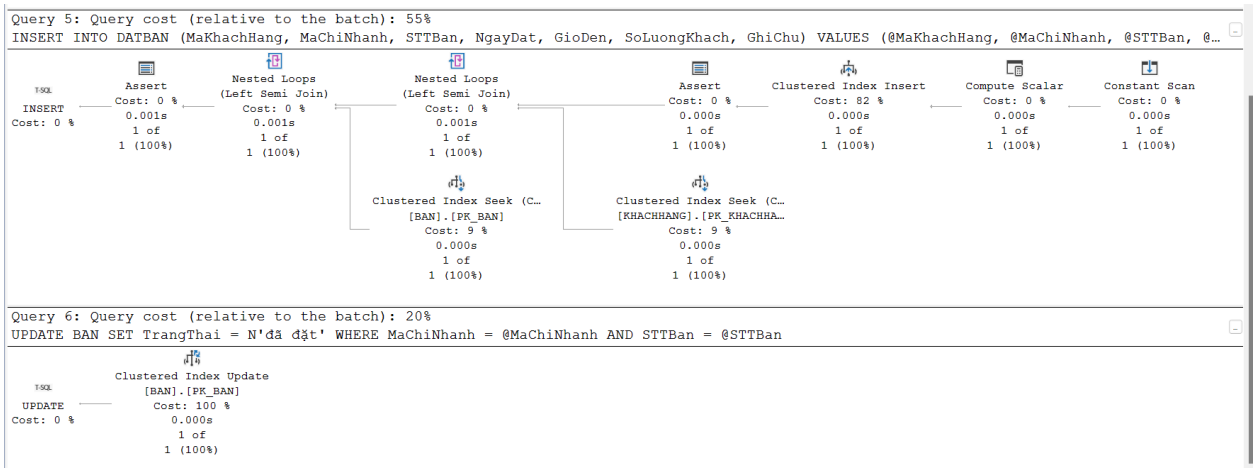
EXEC sp_AddReservation '1', '1', '1', '2025-01-02', '2025-01-02 12:00:00', '5', 'sinh nhat'

có index

- Thời gian thực thi: 2184ms
- Kế hoạch thực thi:
  - Clustered Index Seek:
    - Được sử dụng trên bảng BAN, KHACHHANG, và CHINHANH.
    - Hiệu quả: Clustered Index Seek giúp lọc nhanh dữ liệu từ khóa chính hoặc cột có index, giảm chi phí truy xuất I/O.
  - Index Seek (NonClustered) + Key Lookup (Clustered)
    - Index giúp tăng tốc đáng kể việc kiểm tra trùng lặp trên bảng DATBAN.
    - Non-clustered index on DATBAN for columns such as NgayDat.
  - Clustered Index Update:
    - Index giúp cập nhật dữ liệu nhanh hơn trên bảng BAN.
  - Clustered Index Insert:
    - Index giúp thêm mới bản ghi vào bảng DATBAN.
  - Nested Loop:
    - Được sử dụng để kết hợp dữ liệu giữa các bảng (CHINHANH, DATBAN, và BAN).
    - Nested Loop là hiệu quả khi kích thước dữ liệu nhỏ và các bảng được lọc bởi Index Seek.

- Hiệu quả:
  - Việc sử dụng Clustered Index Seek/Update, Index Seek và Nested Loop giúp giảm thời gian thực thi đáng kể.
  - Truy vấn tận dụng các index hiện có, tối ưu hiệu suất truy cập dữ liệu.
  - Tối ưu hóa khi lọc theo các cột `MaChiNhanh`, `STTBan`, `NgayDat`, và `GioDen`.
  - Non-Clustered Index Seek**:** Tìm nhanh trên `DATBAN`.
  - Giảm tải I/O và CPU, tối ưu khi lọc theo `NgayDat`.

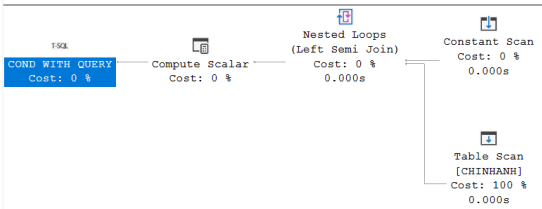




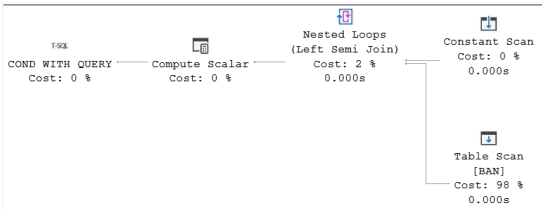
không index

- Thời gian thực thi: 2250ms
- Kế hoạch thực thi:
  - Table Scan:
    - Quét toàn bộ bảng `CHINHANH` và `BAN` để kiểm tra sự tồn tại của chi nhánh và bàn.
    - Quét toàn bộ bảng `DATBAN` để kiểm tra trùng lặp thời gian đặt bàn.
    - Việc sử dụng table scan dẫn tới đọc các dữ liệu không cần thiết.
  - Nested Loops:
    - việc không sử dụng index gây nên không hiệu quả cao trong việc nối các bảng.
- Hạn chế:
  - Table Scan làm tăng lượng dữ liệu đọc từ đĩa và làm tăng chi phí CPU.
  - Hiệu suất bị giới hạn khi không có cơ chế lọc dữ liệu hiệu quả.

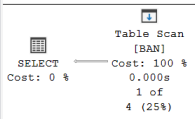
Query 1: Query cost (relative to the batch): 1%
IF NOT EXISTS (SELECT 1 FROM CHINHANH WHERE MaChiNhanh = @MaChiNhanh)



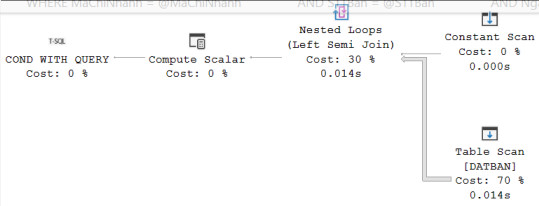
Query 2: Query cost (relative to the batch): 1%
IF NOT EXISTS (SELECT 1 FROM BAN WHERE MaChiNhanh = @MaChiNhanh AND STTBan = @STTBan)



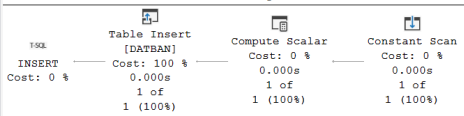
Query 3: Query cost (relative to the batch): 1%
SELECT @TrangThaiBan = TrangThai FROM BAN WHERE MaChiNhanh = @MaChiNhanh AND STTBan = @STTBan



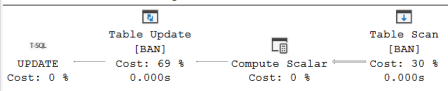
Query 4: Query cost (relative to the batch): 94%
IF EXISTS (SELECT 1 FROM DATBAN WHERE MaChiNhanh = @MaChiNhanh AND STTBan = @STTBan AND NgayDat = @NgayDat AND ((@GioDen >= GioDen AND @Gi…
Missing Index (Impact 99.3039): CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>] ON [dbo].[DATBAN] ([MaChiNhanh],[STTBan],[Ngay…



Query 5: Query cost (relative to the batch): 2%
INSERT INTO DATBAN (MaKhachHang, MaChiNhanh, STTBan, NgayDat, GioDen, SoLuongKhach, GhiChu) VALUES (@MaKhachHang, @MaChiNhanh, @STTBan, @Nga…



Query 6: Query cost (relative to the batch): 2%
UPDATE BAN SET TrangThai = N'đã đặt' WHERE MaChiNhanh = @MaChiNhanh AND STTBan = @STTBan



Kết luận

Hiệu quả của Index:

Mặc dù thời gian thực thi không có sự chênh lệch lớn (2184ms so với 2250ms), điều này có thể do dataset hiện tại tương đối nhỏ.

Với dữ liệu lớn hơn, sự khác biệt sẽ trở nên rõ ràng hơn khi Clustered Index Seek và NonCluster Index mang lại hiệu suất cao hơn nhiều so với Table Scan.

2.5. Quy trình thống kê: thực hiện thống kê top 5 khách hàng chi tiêu nhiều nhất mỗi chi nhánh, thống kê lượt phục vụ.

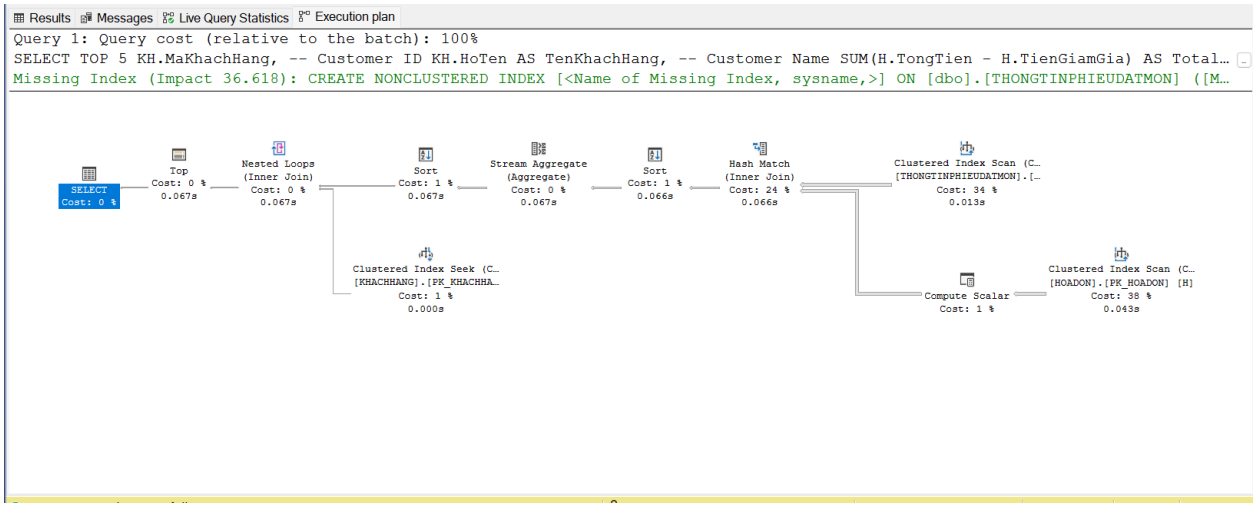
Quy trình hiện tại sử dụng các bảng liên quan đến thống kê top 5 khách hàng như:

- KHACHHANG: Xem thông tin (Read) các khách hàng nằm trong top 5 khách hàng có tổng chi tiêu cao nhất thuộc từng chi nhánh.
- THONGTINPHIEUDATMON: Xem thông tin (Read) các phiếu đặt món để liên kết các khách hàng nằm trong top chi tiêu và các hóa đơn của họ.
- HOADON: Xem các hóa đơn (Read) của các khách hàng nằm trong top chi tiêu để tiến hành thống kê top 5.

Do đó, có thể thực hiện so sánh hiệu suất trước và sau khi tối ưu vì dữ liệu hiện tại có dựa trên truy vấn tận dụng chỉ mục.

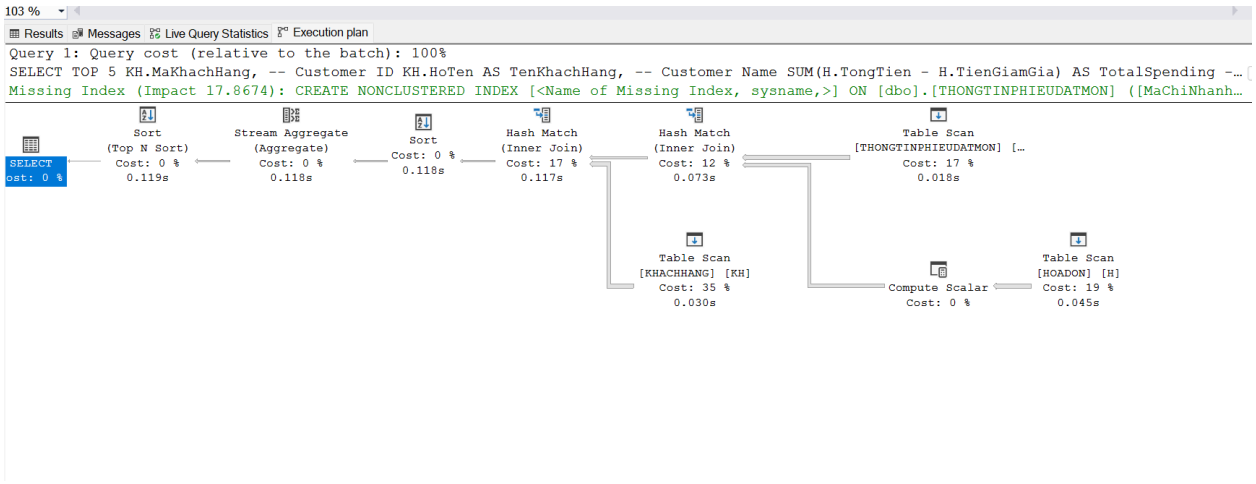
sp_GetTop5CustomersByBranch
EXEC sp_GetTop5CustomersByBranch '1', '2023', '1'
Có Index

- Thời gian: 160ms.
- Kế hoạch thực thi:
  - Clustered Index Seek: Truy xuất nhanh trên `PK_HOADON` và `THONGTINPHIEUDATMON` (38% và 34% chi phí).
  - Nested Loop và Hash Match: Kết hợp dữ liệu giữa các bảng với chi phí thấp (24%).
  - Hiệu quả: Sử dụng `Index Seek` giúp lọc dữ liệu nhanh chóng, giảm chi phí I/O và CPU.



Không có Index

- **Thời gian: 891ms.**
- Kế hoạch thực thi:
  - Table Scan: Quét toàn bộ bảng `HOADON`, `THONGTINPHIEUDATMON`, và `KHACHHANG`.
  - Hash Match: Kết hợp dữ liệu, chi phí cao do không có bộ lọc từ index.
  - Hạn chế:
    - `Table Scan` tốn tài nguyên và tăng thời gian thực thi gấp 5.5 lần so với trường hợp có index.



Kết luận

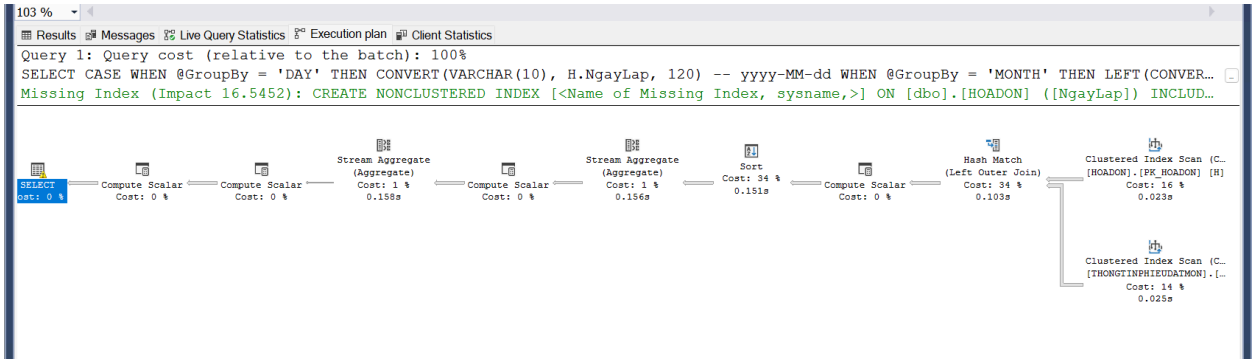
- Hiệu quả của index: Index giúp giảm thời gian thực thi từ 891ms xuống 160ms (giảm 82%), tối ưu đáng kể khi lọc dữ liệu theo chi nhánh và khách hàng.
- Tần suất sử dụng: Với tần suất 1 tháng 1 lần, thời gian thực thi không ảnh hưởng nhiều trong ngắn hạn, nhưng việc có index giúp giảm tải hệ thống, đặc biệt quan trọng khi dữ liệu tăng trưởng.
- Khuyến nghị:
    - Giữ Clustered Index trên `PK_HOADON` và thêm Non-Clustered Index trên `MaChiNhanh` để cải thiện thêm hiệu suất.
    - Theo dõi hiệu suất định kỳ và điều chỉnh index khi dữ liệu thay đổi lớn.

sp_GetCustomerStatsFromInvoice

EXEC sp_GetCustomerStatsFromInvoice '2023-01-01 00:00:00', '2023-03-01 00:00:00', 'DAY'

Có Index

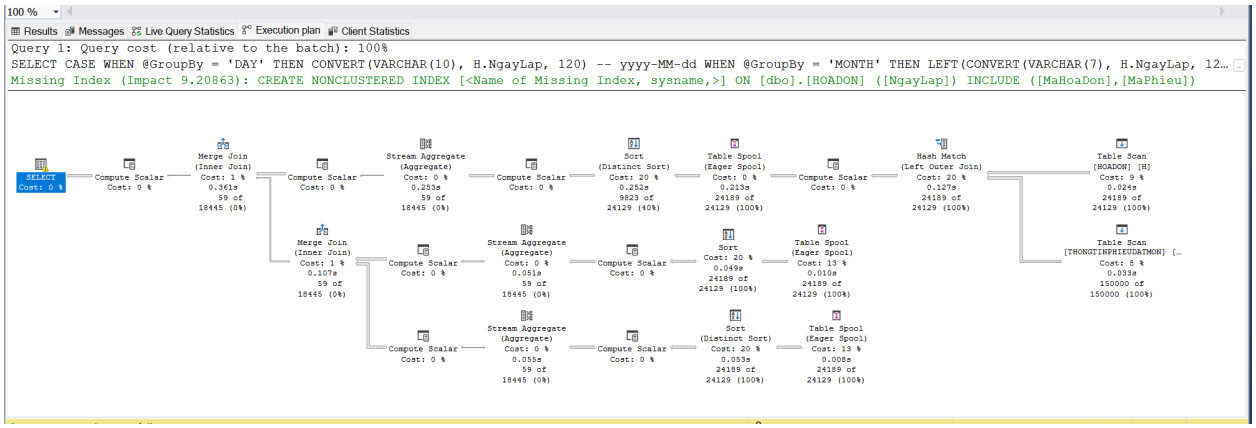
- **Thời gian:** 288ms.
- **Kế hoạch thực thi:**
    - Clustered Index Scan: Quét chỉ mục cụm trên `HOADON` và `THONGTINPHIEUDATMON` (16% và 11% chi phí).
    - Hash Match: Kết hợp dữ liệu từ các bảng với chi phí 34%.
    - Stream Aggregate: Tổng hợp dữ liệu với chi phí thấp.
    - Hiệu quả: Chỉ mục giúp giảm thời gian truy xuất, sử dụng `Clustered Index Scan` thay vì `Table Scan`, tiết kiệm I/O và CPU.



Không có Index

- **Thời gian:** 572ms.
- **Kế hoạch thực thi:**

- Table Scan: Quét toàn bộ bảng `HOADON`, `THONGTINPHIEUDATMON` với chi phí cao.
- Hash Match: Xử lý kết hợp dữ liệu giữa các bảng, chi phí tăng do không có bộ lọc từ index.
- Tăng tài nguyên sử dụng: Quét toàn bộ bảng làm tăng đáng kể tài nguyên I/O và CPU, kéo dài thời gian thực thi.



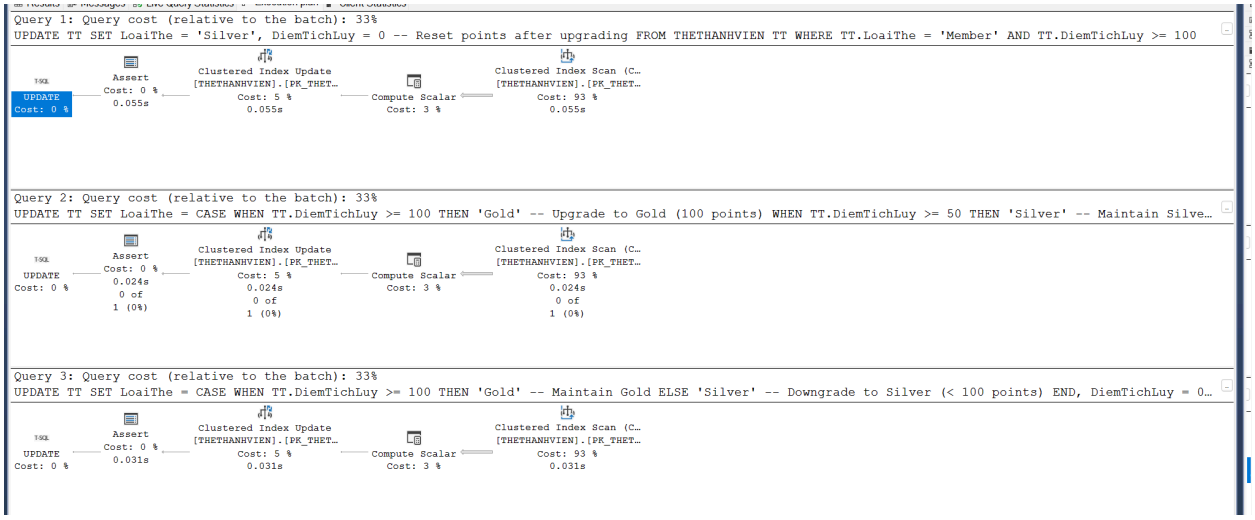
Nhận xét: Việc sử dụng index giúp giảm thời gian thực thi từ 572ms xuống 288ms (giảm gần 50%), tối ưu rõ rệt khi lọc và tính toán dữ liệu.
Tần suất sử dụng 15 lần/ngày: Với tần suất này, việc có index giúp giảm tải đáng kể cho hệ thống, đặc biệt khi truy vấn lặp lại nhiều lần trong ngày. Nếu không có index, thời gian thực thi sẽ lâu hơn, gây tắc nghẽn và làm tăng tải cho hệ thống.
Nên duy trì các index hiện có trên `NgayLap` và các cột liên quan để tối ưu hóa truy vấn.

sp_UpdateMembershipTiers

EXEC sp_UpdateMembershipTiers

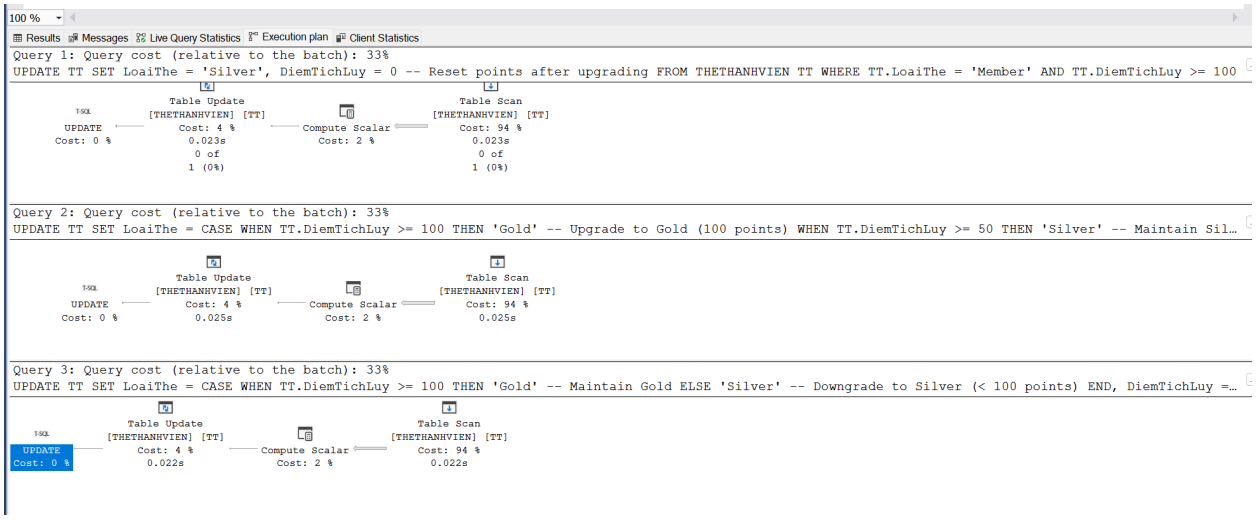
Có Index

- **Thời gian: 931ms.**
- **Kế hoạch thực thi:**
  - Clustered Index Seek: Sử dụng `Index Seek` trên bảng `THETHANHVIEN` để truy xuất các hàng phù hợp với điều kiện lọc (`LoaiThe`, `DiemTichLuy`).
  - Cập nhật (Update): Thao tác cập nhật nhanh chóng nhờ giảm số lượng bản ghi cần xử lý sau khi sử dụng chỉ mục.
  - Hiệu quả: Chỉ mục giúp tìm nhanh các bản ghi phù hợp mà không cần quét toàn bộ bảng, giảm tải I/O và CPU.



Không có Index

- **Thời gian:** 1879ms.
- **Kế hoạch thực thi:**
  - Table Scan: Quét toàn bộ bảng `THETHANHVIEN` để tìm các bản ghi phù hợp, gây lãng phí tài nguyên và tăng thời gian thực thi.
  - Cập nhật (Update): Tốn nhiều thời gian hơn vì phải xử lý tất cả bản ghi mà không có bộ lọc từ index.



Kết luận: Hiệu quả của index: Có index giúp giảm thời gian thực thi từ 1879ms xuống 931ms (giảm 50%), đặc biệt hiệu quả khi xử lý điều kiện lọc (`LoaiThe`, `DiemTichLuy`).
Có thể giữ Clustered Index trên `PK_THETHANHVIEN` và tạo thêm Non-Clustered Index trên `LoaiThe` và `DiemTichLuy` để cải thiện hiệu suất hơn nữa.

2.6. Quy trình cập nhật hạng thẻ:

Quy trình hiện tại sử dụng các bảng liên quan đến việc cập nhật hạng thẻ như:

- THETHANHVIEN: chứa thông tin về thẻ thành viên của khách hàng để có thể tiến hành thực hiện thao tác cập nhật cấp độ (Update) cho thẻ khi đạt điều kiện.

Do đó, không thể thực hiện so sánh hiệu suất trước và sau khi tối ưu vì dữ liệu hiện tại dựa trên truy vấn không tận dụng được chỉ mục.

2.7. Quy trình phục vụ khách hàng của nhân viên

Quy trình hiện tại sử dụng các bảng liên quan đến việc cập nhật hạng thẻ như:

- THONGTINPHIEUDATMON: Thêm và lưu trữ thông tin (Insert) của phiếu đặt món khi tiến hành tạo phiếu đặt món, cũng như để tạo hóa đơn từ thông tin được lưu trữ (Read) trên phiếu đặt món.
- MONAN: Xem các món ăn (Read) được thêm vào phiếu đặt món và liên kết với hóa đơn để lấy thông tin (Read) về các món ăn khi thanh toán.
- CHITIETPHIEUDATMON: Thêm và lưu trữ thông tin (Insert) các món ăn được thêm vào phiếu đặt món và liên kết với hóa đơn để lấy thông tin (Read) khi thanh toán.
- CHINHANH: Xem thông tin (Read) chi nhánh của hóa đơn được tạo.
- BAN: Cập nhật trạng thái (Update) cho bàn sau khi đã tạo hóa đơn và thanh toán.
- HOADON: Thêm và lưu trữ thông tin hóa đơn (Insert) của khách hàng để tiến hành thanh toán.

Do đó, có thể thực hiện so sánh hiệu suất trước và sau khi tối ưu vì dữ liệu hiện tại có dựa trên truy vấn tận dụng chỉ mục.

sp_addOrder

EXEC sp_addOrder '2025-01-02', '1', '1', '1', '1'

Có index
excution time: 249
Kế hoạch thực thi:

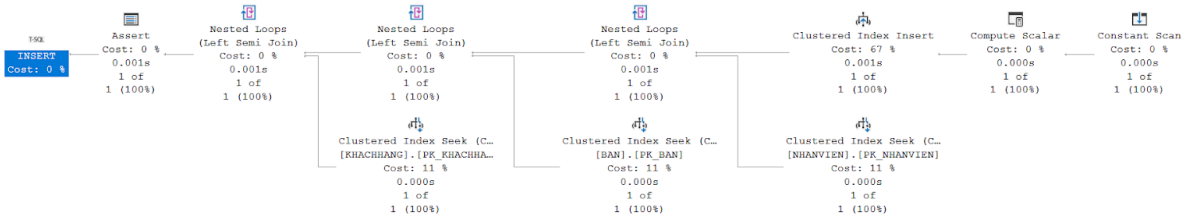
- Clustered Index Seek:

- Được sử dụng trên các bảng [KHACHHANG], [BAN], và [NHANVIEN] để truy xuất các hàng phù hợp.
- Hiệu quả cao nhờ sử dụng chỉ mục trên các bảng liên quan.
- Tiết kiệm I/O và giảm thời gian tìm kiếm dữ liệu cần thiết.

- Nested Loops:
  - Dùng để liên kết giữa các bảng dựa trên chỉ mục, giảm thiểu overhead khi xử lý số lượng lớn dữ liệu.
- Clustered Index Insert:
  - Thao tác chèn dữ liệu vào bảng chính (Clustered Index). Nhanh chóng do các hàng cần chèn đã được xác định rõ ràng thông qua chỉ mục.

Hiệu quả:

- Chỉ mục hỗ trợ tối ưu hóa truy vấn, đảm bảo rằng chỉ các hàng liên quan được xử lý.
- Thời gian thực thi nhanh hơn đáng kể nhờ vào việc tận dụng chỉ mục để hạn chế thao tác quét toàn bộ bảng.



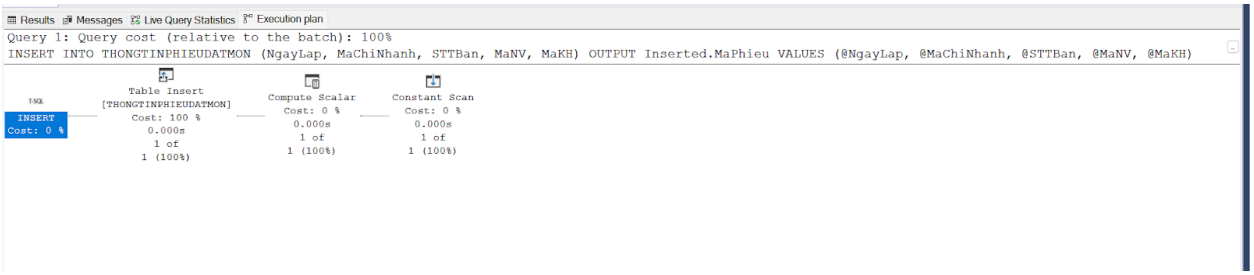
Không index
excution time: 230
Kế hoạch thực thi:

- **Table Insert:**
  - Chèn dữ liệu trực tiếp vào bảng [THONGTINPHIEUDATMON].
  - Không sử dụng chỉ mục, dẫn đến việc xử lý mọi hàng trong bảng, dù không cần thiết.

- **Table Scan:**
  - Quét toàn bộ bảng để tìm dữ liệu, gây lãng phí tài nguyên (CPU và I/O).
  - Điều này có thể làm tăng thời gian xử lý trong trường hợp bảng chứa lượng dữ liệu lớn.

Hạn chế:

- Không có chỉ mục nên hiệu suất bị giảm nếu dữ liệu tăng trưởng, dẫn đến việc quét bảng toàn bộ (Table Scan).
- Phụ thuộc nhiều vào kích thước dữ liệu: nếu bảng lớn, thời gian thực thi sẽ tăng lên đáng kể.



Kết luận:

1. **Hiệu quả của index:**
   - Với index, thời gian thực thi là 249ms, so với 230ms khi không có index. Tuy sự khác biệt nhỏ, nhưng với dữ liệu lớn hơn, sự tối ưu này sẽ càng rõ ràng.
   - Index giúp giảm tải xử lý, đặc biệt khi có nhiều điều kiện lọc.
2. **Khuyến nghị:**
   - Giữ Clustered Index trên các khóa chính của bảng liên quan ([KHACHHANG], [BAN], [NHANVIEN]).
   - Tạo thêm Non-Clustered Index trên các cột thường được truy vấn (ví dụ: MaChiNhanh, NgayLap) để cải thiện hiệu suất chèn và tìm kiếm.
   - Dữ liệu tiếp tục tăng trưởng, việc quản lý index sẽ cần được thực hiện định kỳ để duy trì hiệu suất ổn định.

sp_addDishToOrder

EXEC sp_addDishToOrder '1', '1', '2'

Có index

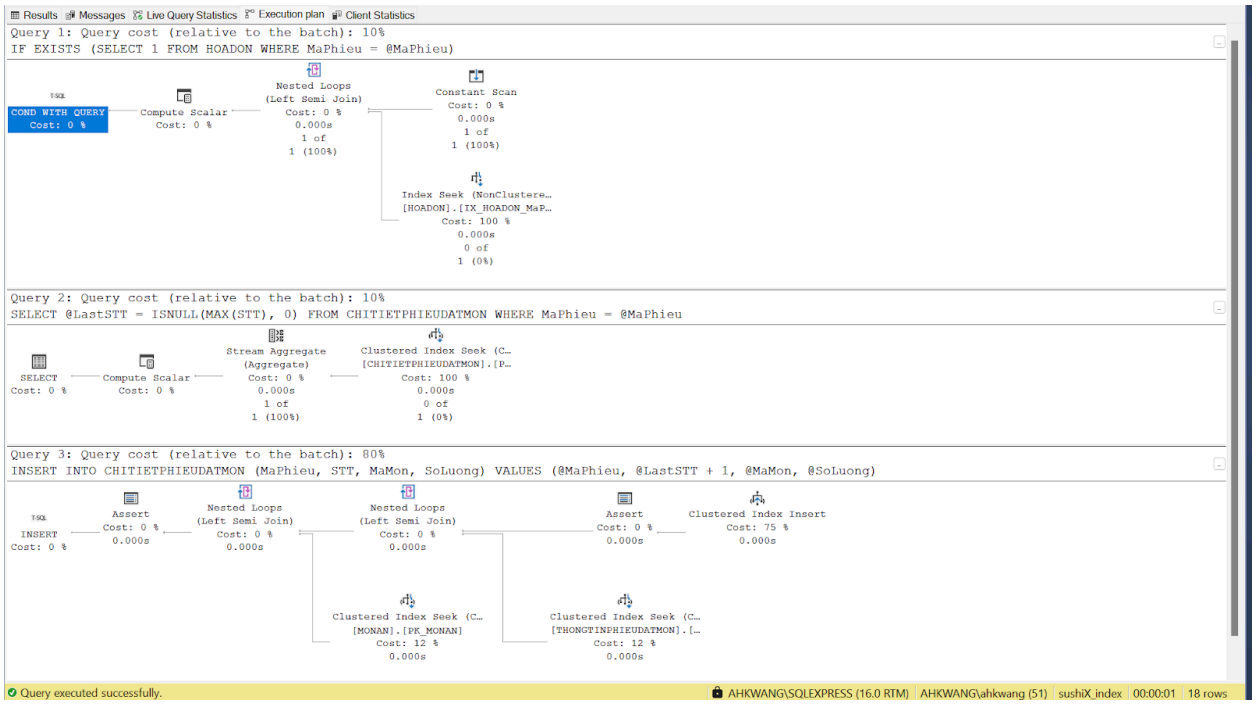
excution time: 918ms

Kế hoạch thực thi:

1. Query 1: IF EXISTS (SELECT 1 FROM HOADON WHERE MaPhieu = @MaPhieu)
    - Index Seek:
        - Sử dụng Non-Clustered Index IX_HOADON_MaPhieu để tìm nhanh các hàng dựa trên MaPhieu.
        - Tiết kiệm I/O và giảm thời gian xử lý.
    - Nested Loops:
        - Kết nối hiệu quả giữa các bảng và điều kiện tìm kiếm.
2. Query 2: SELECT @LastSTT = ISNULL(MAX(STT), 0) FROM CHITIETPHIEUDATMON WHERE MaPhieu = @MaPhieu
    - Clustered Index Seek:
        - Sử dụng Clustered Index trên CHITIETPHIEUDATMON để tìm giá trị STT lớn nhất.
        - Giảm đáng kể thời gian xử lý nhờ không cần quét toàn bộ bảng.
    - Stream Aggregate:
        - Tính toán giá trị MAX nhanh chóng nhờ chỉ xử lý các hàng phù hợp với điều kiện.
3. Query 3: INSERT INTO CHITIETPHIEUDATMON (...)
    - Clustered Index Seek:
        - Tìm kiếm nhanh trong bảng MONAN để xác minh dữ liệu cần thiết.
    - Clustered Index Insert:
        - Thao tác chèn dữ liệu vào bảng CHITIETPHIEUDATMON với hiệu suất cao nhờ xác định hàng cần chèn thông qua chỉ mục.

Hiệu quả:

- Sử dụng chỉ mục giúp giảm đáng kể thời gian xử lý từng truy vấn.
- Giảm tối đa số lượng I/O và tài nguyên CPU cần thiết.



Không index
excution time: 1305ms
Kế hoạch thực thi:

1. Query 1: IF EXISTS (SELECT 1 FROM HOADON WHERE MaPhieu = @MaPhieu)
    - Table Scan:
        - Quét toàn bộ bảng HOADON để tìm MaPhieu, gây lãng phí tài nguyên và thời gian.
    - SQL Server khuyến nghị tạo Non-Clustered Index trên HOADON.MaPhieu.
2. Query 2: SELECT @LastSTT = ISNULL(MAX(STT), 0) FROM CHITIETPHIEUDATMON WHERE MaPhieu = @MaPhieu
    - Table Scan:
        - Quét toàn bộ bảng CHITIETPHIEUDATMON để tìm giá trị MAX của STT, làm tăng thời gian xử lý.
    - SQL Server khuyến nghị tạo Non-Clustered Index trên CHITIETPHIEUDATMON.MaPhieu.
3. Query 3: INSERT INTO CHITIETPHIEUDATMON (...)
    - Table Insert:
        - Chèn dữ liệu trực tiếp mà không sử dụng chỉ mục. Điều này làm tăng overhead khi bảng chứa nhiều dữ liệu.

Hạn chế:

- Không có chỉ mục dẫn đến quét bảng (Table Scan), ảnh hưởng lớn đến hiệu suất với dữ liệu lớn.
- Thời gian thực thi tăng đáng kể so với trường hợp có index.

Kết luận

1. Hiệu quả của index:
    - Thời gian thực thi giảm từ 1305ms (không có index) xuống 918ms (có index), tức giảm 30%.
    - Các chỉ mục hỗ trợ tối ưu hóa tìm kiếm và chèn dữ liệu, giảm số lượng I/O cần thiết.

sp_CreateInvoice

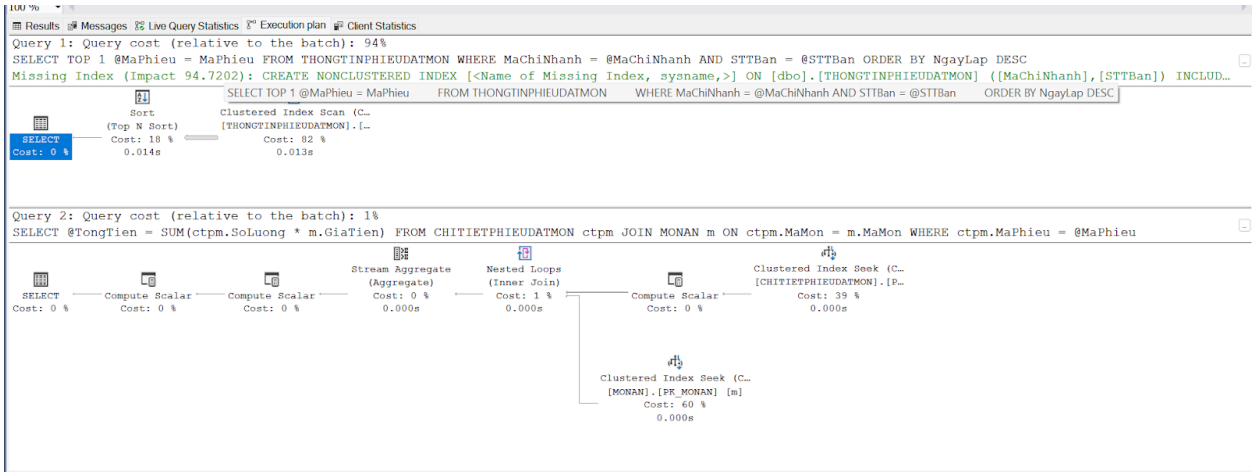
```
DECLARE @MaHoaDon INT; -- Biến để nhận OUTPUT

EXEC sp_CreateInvoice

    @MaChiNhanh = 1, -- Giá trị mẫu cho chi nhánh

    @STTBan = 1,     -- Giá trị mẫu cho số thứ tự bàn

    @TienGiamGia = 1000, -- Giá trị giảm giá mẫu (nếu không có, có thể bỏ qua vì có giá trị mặc định)

    @MaHoaDon = @MaHoaDon OUTPUT; -- Nhận giá trị MaHoaDon từ OUTPUT
```

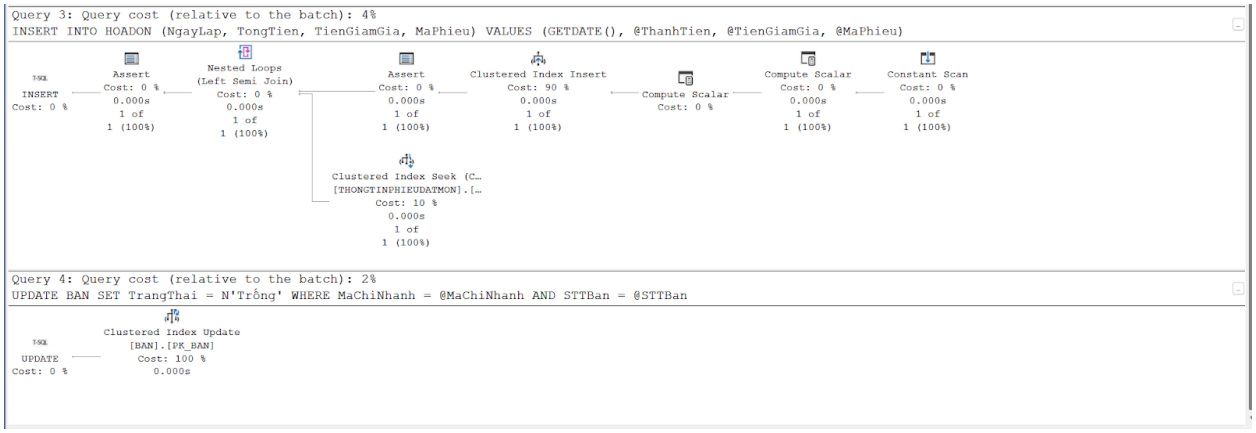
Có index
excution time: 1217ms
Kế hoạch thực thi:

1. Query 1: SELECT TOP 1 @MaPhieu = MaPhieu FROM THONGTINPHIEUDATMON WHERE ... ORDER BY NgayLap DESC
    - Clustered Index Seek:
        - Sử dụng chỉ mục để tìm kiếm nhanh trên bảng THONGTINPHIEUDATMON.
        - Thao tác sắp xếp (Sort) được hỗ trợ nhờ index, tiết kiệm tài nguyên.
    - SQL Server khuyến nghị thêm Non-Clustered Index để tối ưu hơn nữa.
2. Query 2: SELECT @TongTien = SUM(ctpm.SoLuong * m.GiaTien) ...
    - Clustered Index Seek:
        - Sử dụng chỉ mục trên bảng CHITIETPHIEUDATMON và MONAN để thực hiện tính toán SUM.
    - Stream Aggregate:
        - Tối ưu hóa tính toán tổng (SUM) nhờ chỉ xử lý các hàng được lọc.
3. Query 3: INSERT INTO HOADON ...
    - Clustered Index Insert:
        - Chèn dữ liệu vào bảng HOADON với hiệu suất cao.
        - Sử dụng Nested Loops và Clustered Index Seek để tìm kiếm dữ liệu liên quan.
4. Query 4: UPDATE BAN SET TrangThai = ...
    - Clustered Index Update:
        - Cập nhật trạng thái bảng BAN nhanh chóng nhờ chỉ mục.

Hiệu quả:

- Index tối ưu hóa tìm kiếm và cập nhật dữ liệu.
- Thời gian xử lý từng query giảm đáng kể.





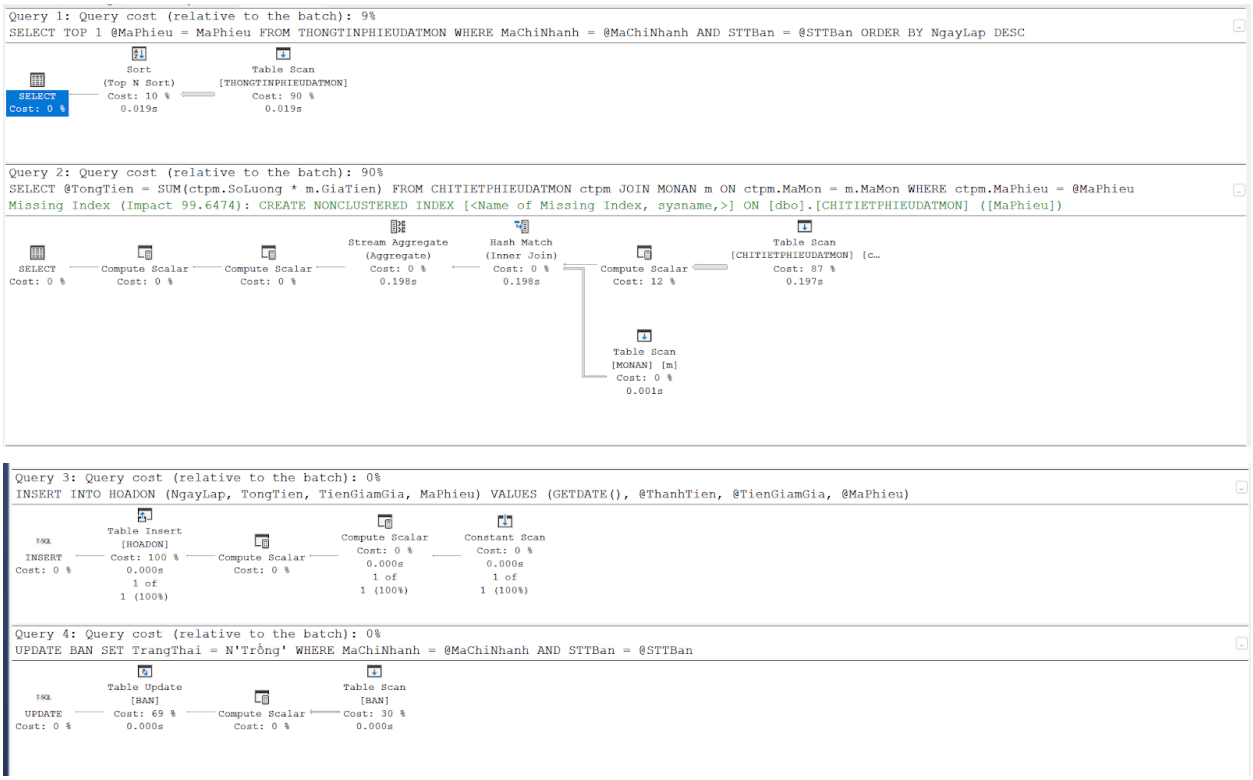
Không index
excution time: 1616
Kế hoạch thực thi:

1. Query 1: SELECT TOP 1 @MaPhieu = MaPhieu FROM THONGTINPHIEUDATMON WHERE ... ORDER BY NgayLap DESC
    - Table Scan:
        - Quét toàn bộ bảng THONGTINPHIEUDATMON để tìm MaPhieu.
        - Không có chỉ mục dẫn đến sắp xếp tốn kém tài nguyên.
    - SQL Server khuyến nghị thêm Non-Clustered Index để cải thiện hiệu suất.

2. Query 2: SELECT @TongTien = SUM(ctpm.SoLuong * m.GiaTien) ...
    - Table Scan:
        - Quét toàn bộ bảng CHITIETPHIEUDATMON và MONAN.
    - Hash Match:
        - Dùng để thực hiện join giữa CHITIETPHIEUDATMON và MONAN, nhưng tiêu tốn nhiều tài nguyên hơn Nested Loops.
3. Query 3: INSERT INTO HOADON ...
    - Table Insert:
        - Chèn dữ liệu trực tiếp mà không có hỗ trợ từ chỉ mục, gây overhead lớn.
4. Query 4: UPDATE BAN SET TrangThai = ...
    - Table Scan:
        - Quét toàn bộ bảng BAN để cập nhật dữ liệu, làm tăng thời gian xử lý.

Hạn chế:

- Table Scan và Hash Match làm tăng đáng kể thời gian thực thi.
- Hiệu suất kém hơn nhiều so với trường hợp có index.



Kết luận

1. Hiệu quả của index:
    - Thời gian xử lý giảm từ 1616ms (không có index) xuống còn 1217ms (có index), giảm khoảng 25%.
    - Index giúp giảm thiểu Table Scan, tăng hiệu suất đáng kể.
2. Khuyến nghị:
    - Tạo thêm Non-Clustered Index trên:
        - THONGTINPHIEUDATMON.MaChiNhanh, STTBan, và NgayLap
        - CHITIETPHIEUDATMON.MaPhieu
    - Tối ưu join giữa CHITIETPHIEUDATMON và MONAN bằng cách sử dụng chỉ mục.
    - Bảo trì (rebuild/reorganize) index thường xuyên để duy trì hiệu suất.

sp_AuthenticateUser: full table scan, no index, không xét