BÁO CÁO BÀI TẬP NHÓM BẢN CHÍNH THỨC

Học phần: Thực hành CSDL – IT3290

Nhóm 4 – 156778 – GV: TS. Nguyễn Thị Oanh

BÀI TOÁN QUẢN LÝ SIÊU THỊ

Hà Nội, 6/2025

MỤC LỤC

A. THÔNG TIN NHÓM

Họ tên SV	**MSSV**	**STT theo DS lớp**
Lương Quý Hiếu (NT)	20235711	14
Lê Vũ Nguyên Hoàng	20235723	15
Phạm Đào Việt Hoàng	20235727	17

B. THÔNG TIN BÀI TẬP

Đề tài: CSDL cho quản lí siêu thị.

Nhiệm vụ:

+ Mô tả được nghiệp vụ các hoạt động diễn ra trong siêu thị.

+ Đưa ra các chức năng cần có của ứng dụng sẽ xây dựng, có sự phân quyền giữa các chủ thể tham gia.

+ Xây dựng sơ đồ thực thể liên kết và sơ đồ quan hệ.

Giải quyết:

1. Tổng hợp tài liệu – Hướng dẫn sử dụng:

a. Tài liệu bao gồm:

- Báo cáo chính thức định dạng PDF: bao gồm toàn bộ thông tin của bài tập lớn.
- Slide thuyết trình buổi thi CK môn học: phục vụ cho mục đích thuyết trình bài thi cuối kỳ.
- File Group4_156778_Query.sql tổng hợp tài liệu 30 yêu cầu truy vấn của 3 sinh viên trong nhóm.
- File g4.market.sql lưu toàn bộ câu lệnh SQL khởi tạo CSDL và view, trigger lưu ở trên tài khoản admin_market và CSDL market.
- File CSDL_20242_OanhNT chứa source code chương trình Java để chạy chương trình thử nghiệm trên máy. Để chạy được chương trình, người dùng phải đáp ứng 2 điều kiện:
    1. Điều kiện cần: có cài sẵn phần mềm chạy ngôn ngữ Java, có sử dụng JavaFX để chạy FXML phục vụ vấn đề giao diện, trước khi chạy phải import CSDL từ

file g4.market.sql về máy chạy trên hệ quản trị CSDL postgreSQL bằng giao diện dòng lệnh hoặc giao diện pgAdmin4.

2. Điều kiện đủ: sau khi tải file về máy, người dùng tìm đến file Main_App.java để bấm nút Run chương trình. Giao diện mở đầu sẽ là màn hình đăng nhập, người dùng có quyền test trên ứng dụng với 3 vai trò: customer, employee và admin.

+ Đối với customer: tài khoản là customer01 -> customer04, mật khẩu mặc định là 123456.
+ Đối với employee: tài khoản là employee01 -> employee04, mật khẩu mặc định là abc123.
+ Đối với admin chỉ có 1 tài khoản duy nhất: admin01, mật khẩu mặc định là admin123.

Các giá trị từ 1 đến 4 là các tài khoản test để truy cập vào theo customer_id và employee_id từ 1 đến 4.

Toàn bộ chương trình được lưu ở trên GitHub, giảng viên và các bạn có thể truy cập bằng 2 cách sau (hiện tại đang để chế độ public):

Cách 1: HieuLuong1/CSDL.20242.156778: Dự án code CSDL và chương trình Java. Hoặc:
https://github.com/hieuluong1/csdl.20242.156778/tree/main

Cách 2: Quét chương trình thông qua mã QR Code:

3. **Hướng dẫn sử dụng báo cáo:**

- Báo cáo cho biết thông tin cá nhân của các thành viên trong nhóm 4_156778_IT3290. Về nội dung chuyên môn, báo cáo sẽ trình bày lần lượt từ mô tả nghiệp vụ bài toán quản lí siêu thị, mô tả chức năng của hệ thống, thiết kế sơ đồ thực thể liên kết, chuyển sang sơ đồ quan hệ. Ngoài ra ở mục số 6 đã tổng hợp 30 câu hỏi truy vấn và phân tích hiệu năng của từng câu lệnh của mỗi thành viên. Mục nhận xét – nâng cao là những gì đã làm được của nhóm, những ý tưởng và gợi mở còn có thể phát triển của dự án này và những khó khăn mà nhóm gặp phải trong quá trình thiết kế. Ở phần cuối cùng, nhóm xin gửi đến giảng viên TS. Nguyễn Thị Oanh lời cảm ơn chân thành đối với nhóm trong học kì 2024.2.
- Phần phụ lục là các cập nhật sau mỗi phiên bản (mỗi phiên bản đều được gửi trên Teams của nhóm).

2. Mô tả nghiệp vụ:

Nghiệp vụ được chia thành 3 luồng hoạt động chính gồm hoạt động mua bán hàng hóa, hoạt động nhập hàng và quản lí hàng hóa và hoạt động quản lí nhân sự của siêu thị.

A. Mua bán hàng:

a. Quản lí thông tin khách hàng:

- Khi khách đến mua hàng, thu ngân có thể tạo hồ sơ khách hàng, mỗi **khách hàng** được lưu bởi 1 mã khách hàng khác nhau, tên, SDT, email (nếu có) của khách.
- Khách hàng có thể kiểm tra được lịch sử mua hàng của mình.

b. Bán hàng và thanh toán:

- Khách hàng mang giỏ hàng đến quầy thu ngân, nhân viên sẽ quét mã hàng hóa, nhập số lượng bán ra, hệ thống tự động đưa ra số tiền niêm yết của từng sản phẩm * số lượng sản phẩm đó, thuế VAT sau đó xuất ra tổng tiền trên **hóa đơn**. Hiện tại, siêu thị chưa áp dụng hình thức Khuyến mãi nào.
- Hóa đơn của khách hàng được hiển thị trên màn hình bao gồm thông tin khách hàng, ngày giờ mua hàng, tên siêu thị (nếu siêu thị rộng có thể thêm quầy số bao nhiêu), nhân viên thu ngân là ai, thông tin sản phẩm bán ra và số tiền cần chi trả.
- Sau đó, khách hàng lựa chọn hình thức thanh toán. Siêu thị nhận được 3 hình thức thanh toán bao gồm tiền mặt, tiền chuyển khoản hoặc ví điện tử. Khách hàng thanh toán đúng số tiền gửi, nhân viên xác nhận thanh toán, hệ thống ghi nhận đơn hàng, gửi số tiền thu được đến nghiệp vụ doanh thu, gửi số hàng đã bán đến nghiệp vụ quản lí kho cập nhật số lượng...

B. Nhập hàng và Quản lí kho:

c. Quản lí hàng hóa và kho:

- Cứ 1-2 tuần cố định, nhân viên sẽ tiến hành kiểm tra lại **kho** và **hàng hóa** tồn, hàng sắp hết ở trong kho. Từ đó người nhân viên có thể cập nhật kho (còn loại hàng nào, số lượng bao nhiêu, của **nhà cung cấp** (NCC) nào, chất lượng hàng, HSD...). Bên cạnh đó, để quản lí số lượng thật chặt chẽ hơn nữa để tránh thất thoát sản phẩm, quản lí cũng cần phải đi kiểm kê thực tế và đột xuất. Khi kiểm kê số lượng thực tế, người quản lý cần có 1 phiếu kiểm tra cụ thể, trên đó là đầy đủ thông tin mã đơn kiểm kê, ngày giờ, lô hàng đã kiểm, số lượng thực tế, số lượng được lưu trên hệ thống.
- Mỗi sản phẩm của siêu thị sẽ được phân loại vào từng chủng loại cụ thể như Sữa và sản phẩm chế biến từ sữa, Rau củ quả, Thịt...
- Hàng hóa khi được đưa ra khỏi kho, hệ thống sẽ cập nhật số lượng còn lại trong kho. Siêu thị bán hàng theo hình thức nhập nhiều bán 1 (FIFO), nghĩa là có thể nhập nhiều hàng hoặc cùng 1 loại hàng nhưng nhiều lô một lúc, sau đó sẽ bán lần lượt từng lô cho đến khi lô đó hết rồi mới thêm bằng lô khác, ưu tiên theo ngày nhập, nếu ngày nhập bằng nhau thì siêu thị phải có trách nhiệm lưu trên hệ thống số thứ tự của lô, lô có số thứ tự nhỏ hơn sẽ bán trước.
- Đối với những loại hàng hóa có HSD, siêu thị cần phải kiểm tra nghiêm ngặt sản phẩm theo từng lô hàng nhập về của sản phẩm đó. Giả sử sản phẩm Sữa Ông Thọ nhập về từ lô SD1111 có hạn sử dụng là 30/6/2025, thì người quản lý và nhân viên

phải nắm được lô đó nhập từ khi nào, nhập số lượng bao nhiêu, còn bao nhiêu sản phẩm chưa bán trong lô đó để có thể chủ động trong vấn đề xử lí lô hàng tồn khi gần hết HSD.

d. Nhập hàng:

- Nhân viên tạo phiếu nhập hàng đến từng NCC về tên hàng, số lượng, địa điểm, thời gian nhận hàng. Siêu thị nhập từ mỗi nhà cung cấp nhiều sản phẩm và cũng không độc quyền sản phẩm nào, do đó mỗi sản phẩm cũng có nhập từ nhiều nhà cung cấp khác nhau.
- NCC gửi tới siêu thị hóa đơn thanh toán hàng hóa nhập bao gồm các thông tin về NCC (tên, địa chỉ, email, SDT khi cần làm việc với ai), thông tin lô hàng, ngày nhập hàng, ngày hết hạn (nếu có), đơn giá, thuế VAT, tổng tiền phải thanh toán.
- Khi hàng về, nhân viên kiểm tra hàng hóa theo yêu cầu của người quản lí thông qua 1 biên bản nhập hàng, biên bản này là sự tổng hợp của tờ phiếu nhập hàng với hóa đơn nhập hàng. Nghĩa là đối với mỗi sản phẩm và số lượng yêu cầu, siêu thị sẽ kiểm tra xem đã đủ lô và số lượng nhận về chưa, nếu đủ rồi thì đó là những lô nào, HSD đến bao giờ?..

C. Quản lí chung:

e. Quản lí nhân viên:

- Khi **nhân viên** được nhận vào hệ thống, siêu thị và người quản lí cần thiết lập và lưu trữ mã nhân viên, thông tin (mã CCCD, tên, tuổi, ngày sinh, giới tính, địa chỉ nhà, SDT và email liên lạc nhân viên đó).
- Quản lí cập nhật lịch làm việc theo tháng của nhân viên lên hệ thống. Mỗi ca làm việc có thể có nhiều nhân viên. Mỗi ngày, quản lí có thể điểm danh nhân viên và kèm ghi chú thưởng phạt theo ca làm (đồng phục, đầu tóc...) đối với mỗi nhân viên thông qua hệ thống giám sát. Nhân viên có thể xem được lịch làm việc của bản thân, xem được tình trạng ngày hôm đó (đi làm/ nghỉ/ muộn/ phép). Đây là cơ sở để nhà quản lí nắm được ngày công và ngày nghỉ, chế độ lương, thưởng phạt của nhân viên.
- Nếu có việc nghỉ phải báo lại cho quản lí trước thời gian nghỉ (bao lâu) để bố trí nhân sự thay thế, trừ trường hợp các lí do đặc biệt, ... để quản lí tính vào ngày phép và chấm công.
- Khi nhập hóa đơn thanh toán hay báo cáo kho, tên của nhân viên cũng sẽ được hiển thị cụ thể để nhà quản lí nắm được thông tin bán hàng, có thể truy xuất nếu có vấn đề bất ngờ gì đó.

f. Quản lí doanh số:

- Siêu thị cần quản lí được số tiền nhận/chi trong từng tháng.
- Về hoạt động chi, siêu thị quản lí được số tiền đã xuất ra để nhập hàng từng NCC, tính được bảng **lương** của nhân viên (dựa trên ngày công, thưởng phạt theo quy định,

số tiền theo ngày, thưởng phạt), các chi phí phát sinh sửa chữa hoặc chi phí ổn định cần chi hàng tháng.

- Về hoạt động nhận, siêu thị quản lí được số tiền nhận của mỗi đơn hàng, cập nhật được theo ngày giờ cụ thể số tiền nhận được.

3. Mô tả chức năng:

Cần xây dựng 1 hệ thống phân quyền cho 3 đối tượng sử dụng chính gồm khách hàng, nhân viên, quản lí siêu thị.

a. Đối với khách hàng

- Ở màn hình khách hàng, khách có thể nhìn thấy 2 nút bấm "*Thông tin khách hàng*" và "*Lịch sử mua hàng*". Khi bấm vào nút 1 sẽ hiển thị toàn bộ thông tin bản thân, do siêu thị chỉ bán trực tiếp nên nếu muốn sửa thông tin gì khách hàng hãy đến siêu thị để nhờ nhân viên chỉnh sửa. Đối với nút số 2 khi bấm vào sẽ hiển thị toàn bộ các hóa đơn mua hàng bao gồm mã hóa đơn, thời gian mua, tổng tiền đã chi trả cho từng hóa đơn. Ngoài ra khi bấm vào phần xem chi tiết mỗi hóa đơn, khách hàng có thể xem được đầy đủ từng sản phẩm và số lượng, tổng tiền đã mua của sản phẩm đó.

Chức năng	**Đầu vào**	**Đầu ra**
Thông tin khách hàng		Toàn bộ thông tin của Khách lưu trong CSDL.
Lịch sử mua hàng	Chọn 1 bản ghi	Chi tiết các sản phẩm và số lượng, giá tiền đã mua các sản phẩm.

b. Đối với nhân viên

- Nhân viên sẽ có các chức năng chính bao gồm Thông tin cá nhân, Lịch làm việc, Quản lí khách hàng, Quản lí kho, Tạo hóa đơn, Nhập hàng, Nhà cung cấp.
- Đối với chức năng *"Thông tin cá nhân"* khi bấm vào sẽ hiển thị đầy đủ thông tin của nhân viên đó lên hệ thống, nhân viên chỉ có thể xem không thể sửa các thông tin này, cũng như chỉ xem được của bản thân không xem được của người khác.
- Đối với chức năng *"Lịch làm việc"*, nhân viên có thể theo dõi được lịch làm việc của bản thân trong tháng cụ thể giờ cụ thể, xem được đi làm bao nhiêu buổi, vắng bao nhiêu buổi. Nhân viên có thể chọn lọc theo tháng, năm để lọc theo lịch làm việc của tháng, năm đó. Bấm vào nút xem chi tiết để xem cụ thể từng ngày. Nhân viên có thể lựa chọn nút *"Xem lương"* ở ngay bên cạnh trong tháng đó trong trường hợp quản lí đã cho phép xem. Bảng lương sẽ bao gồm đầy đủ mọi thông tin của nhân viên được lưu trên hệ thống siêu thị.
- Đối với chức năng *"Quản lí khách hàng"* khi bấm vào, nhân viên có thể tìm kiếm được khách hàng trong hệ thống bằng tên hoặc số điện thoại của họ, có thể tạo mới

khách hàng nếu họ mới đến đây lần đầu hoặc cập nhật thông tin khách hàng nếu bấm vào ô đó.

- Chức năng *"Quản lí kho"* cho phép nhân viên xem và tìm kiếm được các sản phẩm của siêu thị hiện có. Ngoài ra khi lựa chọn phần *"Thêm sản phẩm"*, nhân viên cũng có thể tạo mới các sản phẩm được thêm vào trong siêu thị. Ở đây, nhân viên cũng có thể xem được các mặt hàng mới được nhập về thông qua chức năng cùng tên *"Hàng mới nhập"*. Chức năng này cho phép nhân viên xem được các lô hàng mới cập nhật của siêu thị và có thể tìm kiếm theo ngày hoặc theo nhà cung cấp.
- Chức năng *"Tạo hóa đơn"* sẽ được sử dụng khi khách mua hàng, chỉ đơn giản là tạo hóa đơn và các thông tin cơ bản trong hóa đơn đó. Nhân viên điền thông tin sản phẩm, sau đó hiển thị ra các lô hàng và số lượng còn của chúng (trên thực tế siêu thị đã có gắn nhãn các lô được đưa ra quầy ngày hôm đó còn trong phạm vi môn học này nhóm sẽ lựa chọn hình thức chọn lô hàng coi như nhân viên biết lô nào đã được bán ra), các sản phẩm này khi bấm vào sẽ được lựa chọn và điền đầy đủ các thông tin khác, nhân viên có thể tạo 1 đơn hàng mới.
- Ở phần *"Nhập hàng"* nhân viên có quyền tạo 1 biên bản nhập hàng sau khi hàng hóa đến, nhân viên nhập vào các lô được nhận về trong ngày hôm đó, đếm đúng số lượng rồi điền vào máy, sau đó sẽ được lưu trên hệ thống.
- Ở phần *"Nhà cung cấp"*, nhân viên có thể xem được chi tiết các nhà cung cấp và các thông tin của họ.

Chức năng	**Đầu vào**	**Đầu ra**
Thông tin cá nhân		Toàn bộ thông tin của nhân viên.
Lịch làm việc	Chọn tháng, năm	Thời gian làm việc cụ thể, điểm danh từng bản ghi cụ thể (D, M, V, P)
Lương	*Chỉ xem được khi thông báo có lương của tháng đó, quản lí đã tạo 1 bảng lương.	Chi tiết bảng lương tháng.
Tìm kiếm khách hàng	SDT Khách hàng	Thông tin Khách Hàng
Tạo Khách hàng	Họ tên, SDT, Email nếu có	Khách hàng được tạo.
Quản lí kho	Tìm kiếm tên sản phẩm	Thông tin sản phẩm.
Thêm hàng hóa	Nhập thông tin hàng hóa	Hàng hóa mới được tạo.
Hàng mới nhập		Toàn bộ các lô hàng mới nhập về.

Lọc theo	Tick chọn Ngày nhập/NCC/HSD	Điền lên thanh tìm kiếm, kết quả trả về theo yêu cầu.
Cập nhật/Xóa sản phẩm	Thông tin sản phẩm, số tiền, chủng loại mới	Kết quả mong muốn.
Tạo hóa đơn	Các thông tin của hóa đơn, sản phẩm trong hóa đơn đó.	Hóa đơn mới được tạo.
Nhập hàng	Thông tin biên bản nhập hàng, các lô được nhập.	Biên bản được nhập.
Nhà cung cấp		Xem thông tin Nhà cung cấp.

c. Đối với nhà quản lí

- Quản lí sẽ có các chức năng chính bao gồm Lịch làm việc, Quản lí khách hàng, Quản lí kho, Lương, Tạo hóa đơn, Nhập hàng, Nhà cung cấp, Biên bản kiểm kê, xử lí.
- Đối với chức năng *"Lịch làm việc"*, quản lí có thể tìm kiếm theo thông tin của người nhân viên đó. Ngoài ra khi bấm vào từng dòng nhân viên và từng cột giá trị của nhân viên, quản lí có thể cập nhật thông tin của người đó cho đúng với sự thay đổi giá trị. Quản lí có thể thêm 1 nhân viên mới vào hệ thống. Hơn nữa, quản lí cũng có thể bấm vào nút *"Thêm lịch làm việc"* để chọn giờ làm và lịch làm cho nhân viên trong tháng đó. Khi bấm vào đây và chọn các giá trị tháng, người nhân viên, ngày đầu tiên làm việc của tháng và lịch làm việc, quản lí có thể gán lịch làm việc cho nhân viên. Ngoài ra, sau khi có lịch làm việc rồi, quản lí có thể chọn ngày làm việc rồi lựa chọn tình trạng của ngày hôm đó để *"Điểm danh"* cho nhân viên.
- Chức năng *"Lương"* cho phép quản lí xem được chi tiết các bảng lương. Quản lí có thể lựa chọn tháng năm cụ thể, sau đó bấm *"Thêm bảng lương mới"* , quản lí có thể bấm chọn các mã nhân viên, các thông tin cần thiết để bảng lương được tạo lập thành công.
- Đối với chức năng *"Quản lí khách hàng"* khi bấm vào, quản lí có thể tìm kiếm được khách hàng trong hệ thống bằng tên hoặc số điện thoại của họ, có thể tạo mới khách hàng nếu họ mới đến đây lần đầu.
- Chức năng *"Quản lí kho"* cho phép quản lí xem và tìm kiếm được các sản phẩm của siêu thị hiện có. Ngoài ra khi lựa chọn phần thêm sản phẩm, quản lí cũng có thể tạo mới các sản phẩm được thêm vào trong siêu thị. Ở đây, quản lí cũng có thể xem được các mặt hàng mới được nhập về thông qua chức năng cùng tên *"Hàng mới*

nhập". Chức năng này cho phép quản lí xem được các lô hàng mới cập nhật của siêu thị.

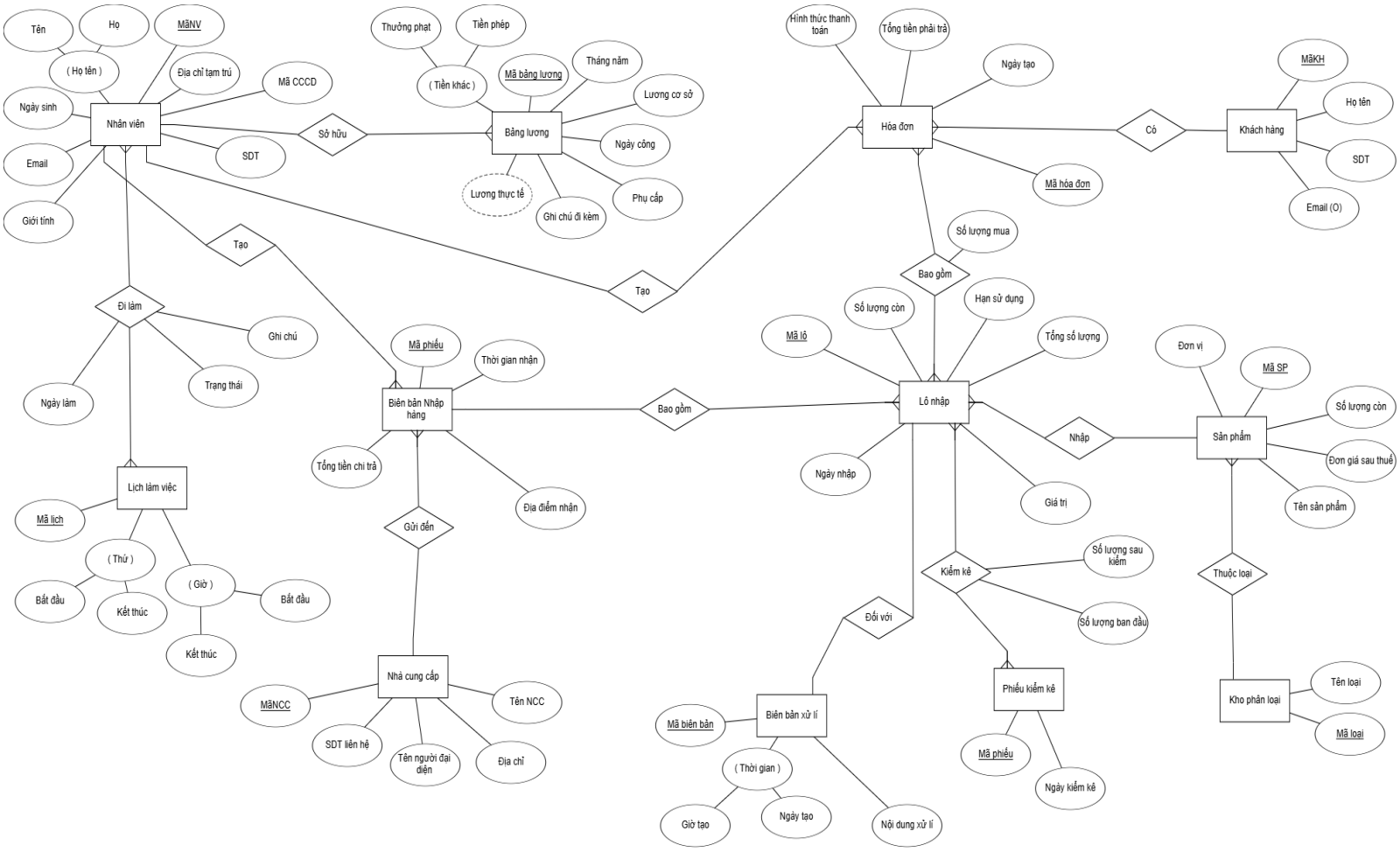
- Chức năng *"Tạo hóa đơn"* sẽ được sử dụng khi khách mua hàng, chỉ đơn giản là tạo hóa đơn và các thông tin cơ bản trong hóa đơn đó. Quản lí điền tên sản phẩm, sau đó hiển thị ra các lô hàng và số lượng còn của chúng, các sản phẩm này khi bấm vào sẽ được lựa chọn và điền đầy đủ các thông tin khác, nhân viên có thể tạo 1 đơn hàng mới.
- Ở phần *"Nhập hàng"*, quản lí có quyền tạo 1 biên bản nhập hàng sau khi hàng hóa đến, nhân viên nhập vào các lô được nhận về trong ngày hôm đó, đếm đúng số lượng rồi điền vào máy, sau đó sẽ được lưu trên hệ thống.
- Quản lí còn có thêm 1 quyền nữa đó là *"Nhà cung cấp"* đến của siêu thị. Quản lí có thể thêm, xóa các thông tin của nhà cung cấp đến với siêu thị.
- Chức năng *"Tạo biên bản"* cho phép quản lí tạo 2 loại biên bản thường dùng trong siêu thị: biên bản kiểm kê và biên bản xử lí hàng hóa. Đối với *"Biên bản kiểm kê"*, khi lựa chọn tạo 1 biên bản mới, khi quản lí chọn mã lô mà mình muốn kiểm tra thì ngay lập tức hiển thị số lượng nhập vào và số lượng còn của lô đó, tuy nhiên khi có thêm 1 ô trống để quản lí có thể cập nhật vào đó số lượng thực tế mà nhà quản lí đã đi kiểm tra. Bấm lưu kết quả để tạo mới 1 bản ghi. Đối với *"Biên bản xử lí"*, nhà quản lí thêm đầy đủ các thông tin như ngày giờ, mã lô, nội dung xử lí để tiến hành xử lí lô hàng đó, bấm lưu để tạo mới 1 bản ghi xử lí sản phẩm.

Chức năng	**Đầu vào**	**Đầu ra**
Quản lí nhân viên	Họ tên NV	Thông tin nhân viên.
Thêm lịch làm việc	Tháng, nhân viên, lịch làm việc, ngày bắt đầu làm	Nhân viên đã được gán lịch làm việc.
Điểm danh	Ngày, tên NV, trạng thái	Kết quả điểm danh nhân viên sẽ nhìn thấy được.
Lương	Cập nhật, chỉnh sửa bảng lương	Bảng lương mới được tạo.
Siêu nhân viên		Nhân viên chăm chỉ nhất của tháng: đi làm đủ từ 25 buổi và không vắng buổi nào trong tháng đó.
Thêm bảng lương mới		Tạo thêm 1 bảng lương rỗng chờ quản lí cập nhật.
Quản lí kho	Tìm kiếm tên sản phẩm	Thông tin sản phẩm.
Thêm hàng hóa	Nhập thông tin hàng hóa	Hàng hóa mới được tạo.

Lọc theo	Tick chọn Ngày nhập/NCC/HSD	Điền lên thanh tìm kiếm, kết quả trả về theo yêu cầu.
Hàng mới nhập		Toàn bộ các lô hàng mới nhập về.
Cập nhật/Xóa sản phẩm	Thông tin sản phẩm, số tiền, chủng loại mới	Kết quả mong muốn.
Tạo hóa đơn	Các thông tin của hóa đơn, sản phẩm trong hóa đơn đó.	Hóa đơn mới được tạo.
Nhập hàng	Thông tin biên bản nhập hàng, các lô được nhập.	Biên bản được nhập.
Nhà cung cấp		Xem thông tin Nhà cung cấp.
Biên bản xử lí	Tạo biên bản xử lí, chọn 1 lô hàng, điền nội dung xử lí.	Biên bản được khởi tạo.
Biên bản kiểm kê	Tạo 1 biên bản kiểm kê, chọn nhiều lô hàng, điều chỉnh giá trị thực tế	Biên bản được khởi tạo.

4. Sơ đồ thực thể liên kết:

Từ nghiệp vụ chi tiết, nhóm đã xây dựng được lên sơ đồ thực thể liên kết.



Một số giải thích thực thể:

- Khách hàng: là người mua hàng khi đến siêu thị, được lưu trữ thông tin gồm Mã KH, Họ tên, SDT và Email nếu có.
- Hóa đơn: là biên bản giao dịch mỗi lần sử dụng dịch vụ của khách hàng khi đến siêu thị, hóa đơn cần phải cho biết Mã hóa đơn, Ngày tạo, Tổng tiền phải trả và Hình thức thanh toán.
- Lô nhập là các lô hàng chứa các sản phẩm của siêu thị, trên lô nhập là Mã lô, ngày nhập, số lượng còn, hạn sử dụng, tổng số lượng và giá trị của lô.

- Sản phẩm của siêu thị bao gồm Mã sản phẩm, Tên sản phẩm, Số lượng còn, Đơn vị và Đơn giá sau thuế.
- Kho phân loại là các loại hàng chung của sản phẩm, bao gồm Mã kho và Tên kho.
- Phiếu kiểm kê là biên bản mà quản lí có thể nhập để đánh giá kiểm tra sản phẩm trong kho theo từng lô, bao gồm Mã phiếu và Ngày kiểm kê.
- Biên bản xử lí là các biên bản dùng để xử lí sản phẩm bị hỏng hoặc hết hạn sử dụng, cần xử lí sản phẩm ngay lập tức và có nội dung cụ thể. Biên bản xử lí bao gồm Mã biên bản, Ngày tạo, Giờ tạo, Nội dung xử lí.
- Biên bản nhập hàng là biên bản sau cùng trong nghiệp vụ nhập hàng mà nhân viên phải quan tâm bao gồm Mã phiếu, Thời gian nhận, Địa điểm nhận, Tổng số tiền chi trả.
- Nhà cung cấp là thông tin các đối tác mà siêu thị hợp tác, bao gồm Mã NCC, Tên NCC, Tên người đại diện, SDT liên hệ, Địa chỉ.
- Nhân viên là những người làm việc cho siêu thị, được siêu thị quản lí bằng Mã NV, Họ tên đầy đủ, Ngày sinh, Giới tính, Địa chỉ tạm trú, SDT, Mã CCCD, Email.
- Bảng lương là tổng hợp lương của nhân viên, được phân biệt bởi Mã bảng lương, Tháng năm, Lương cơ sở, Ngày công, Phụ cấp, Thưởng phạt, Tiền phép, Ghi chú đi kèm, Lương thực tế.
- Lịch làm việc là chi tiết các giờ làm và ngày làm của nhân viên, bao gồm Mã lịch, Thứ bắt đầu, Thứ kết thúc, Giờ bắt đầu, Giờ kết thúc.

Các liên kết trong sơ đồ:

Thực thể A	**Liên kết**	**Thực thể B**
Khách hàng	Có (1 – n)	Hóa đơn
Hóa đơn	Bao gồm (n – n)	Lô nhập
Lô nhập	Nhập (n – 1)	Sản phẩm
Sản phẩm	Thuộc loại (n – 1)	Kho phân loại
Phiếu kiểm kê	Kiểm kê (n – n)	Lô nhập
Biên bản xử lí	Xử lí (1 – 1)	Lô nhập
Biên bản nhập hàng	Bao gồm (1 – n)	Lô nhập
Biên bản nhập hàng	Gửi đến (n – 1)	Nhà cung cấp
Nhân viên	Tạo (1 – n)	Hóa đơn
Nhân viên	Tạo (1 – n)	Biên bản nhập hàng
Nhân viên	Đi làm (n – n)	Lịch làm việc
Nhân viên	Sở hữu (1 – n)	Bảng lương

Bảng trên có 7 liên kết bậc 2, 0 liên kết bậc 1 (đệ quy).

Dựa vào bảng trên, đối với các liên kết n – n như Bao gồm, Đi làm có thể tạo ra 1 quan hệ mới, các liên kết 1 – n, n – 1, 1 – 1 có thể tạo thành khóa ngoài trong sơ đồ quan hệ.

5. Sơ đồ quan hệ:

Sơ đồ quan hệ được chuyển từ sơ đồ thực thể liên kết:

Nhân viên (MãNV, Họ, Tên, Ngày sinh, Email, Giới tính, Địa chỉ tạm trú, SDT, Mã CCCD)

Lịch làm việc (Mã lich, Thứ bắt đầu, Thứ kết thúc, Giờ bắt đầu, Giờ kết thúc)

Đi làm (*MãNV, Mã lịch,* Ngày làm, Trạng thái, Ghi chú)

Bảng lương (Mã bảng lương, Tháng năm, Lương cơ sở, Ngày công, Phụ cấp, Ghi chú đi kèm, Thưởng phạt, Tiền phép, Lương thực tế, *MãNV*)

Khách hàng (MãKH, Họ tên, SDT, Email)

Hóa đơn (Mã hóa đơn, Ngày tạo, Tổng tiền phải trả, Hình thức thanh toán, *MãKH, MãNV tạo*)

Lô nhập (Mã lô, Ngày nhập, Hạn sử dụng, Tổng số lượng, Số lượng còn, *Mã SP, Mã Nhập hàng,* Giá trị)

Bao gồm (*Mã hóa đơn*, *Mã lô*, Số lượng mua)

Sản phẩm (Mã SP, Tên sản phẩm, Đơn vị, Đơn giá sau thuế, Số lượng còn, *Mã loại*)

Kho phân loại (Mã loại, Tên loại)

Phiếu kiểm kê (Mã phiếu, Ngày kiểm kê)

Kiểm tra(*Mã phiếu, Mã lô*, Số lượng sau kiểm, Số lượng ban đầu)

Phiếu nhập hàng (Mã phiếu, Thời gian nhập, Địa điểm nhận, Tổng tiền chi trả, *MãNV, Mã NCC*)

Nhà cung cấp (MãNCC, Tên NCC, Địa chỉ, Tên người đại diện, SDT liên hệ)

Biên bản xử lí (Mã biên bản, Ngày tạo, Giờ tạo, Nội dụng xử lí, *Mã lô*)

- **Biên dịch CSDL sang bản tiếng Anh để đưa vào lập trình đảm bảo tính thống nhất giữa các thành viên trong nhóm:**

Employee (EmployeeID, Lastname, Firstname, DOB, Email, Gender, Address, Phone, IdentityID)

Schedule (ScheduleID, FromDay, ToDay, TimeStart, TimeEnd)

Working (*EmployeeID, ScheduleID,* WorkDate, Status, Note)

Salary (SalaryID, MonthYear, BasicSalary, Workdays, Bonus, Note, Reward/Punish, Leavepay, ActualSalary, *EmployeeID*)

Customer (CustomerID, Fullname, Phone, Email)

Orders (OrderID, Orderdate, Totalamount, Method, *CustomerID, EmployeeID*)

Batch (Batch, ImportDate, Expiration, Totalnumber, QuantInStock, *ProductID, ImportID,* Value)

OrderDetails (*OrderID*, *BatchID*, Quantity)

Products (ProductID, Name, Unit, Price, QuantInStock, *Category*)

Categories (Category, Title)

CheckReport (ReportID, Checkdate)
CheckDetails (*ReportID, Batch*, RealQuant, QuantInStock)
ImportReport (ImportID, Date, Address, TotalAmount, *EmployeeID, SupplierID*)
Supplier (SupplierID, Name, Address, ContactName, Contact)
IncidentReport (IncidentID, Date, Hours, Content, *Batch*)
Lưu ý: phần xem thông tin các cột trong bảng khi đưa lên PostgreSQL đã được mô tả chi tiết ở File g4.market.sql; phần báo cáo này chỉ thể hiện các thuộc tính trong từng quan hệ và các liên kết giữa các bảng để đảm bảo thuộc tính khóa chính và khóa ngoại trong sơ đồ quan hệ.

6. Câu lệnh Truy vấn – Phân tích Hiệu năng:

a. Lương Quý Hiếu (20235711):

1. Cho biết tổng số tiền thu được trong việc bán hàng trong ngày 5/6/2025:

```
select sum(total_amount)

from orders

where order_date = '2025/06/05';
```

- Có thể sử dụng index B-Tree trong trường hợp này và nên sử dụng index trên bảng orders, trên cột order_date. Lí do: Phép toán = thỏa mãn điều kiện sử dụng của index B-Tree, bảng orders là bảng nhiều dữ liệu cần quản lí, cột ngày tháng lại ít bị cập nhật giá trị lại được người dùng thường xuyên truy vấn nên hoàn toàn hợp lí. Câu lệnh tạo index:

```
create index idx_orders_date on orders using btree (order_date);
```

Khi đã có index theo ngày, nó sẽ giúp lọc ra nhanh chóng theo điểm các hóa đơn thỏa mãn.

2. Cho biết các khách hàng đã mua hàng ở cửa hàng vào ngày 5/6/2025. Thông tin in ra gồm mã khách hàng, họ tên đầy đủ của khách hàng đó. Lưu ý nếu khách hàng mua từ 2 lần trên 1 ngày cũng chỉ hiện thông tin 1 lần.

Cách 1:

```
select distinct customer_id, fullname

from customer

where customer_id in (select customer_id from orders where order_date = '2025/06/05');
```

Cách 2:

```
select distinct customer_id, fullname

from customer
```

```
join orders using (customer_id)
where order_date = '2025/06/05';
```

- Như đã biết, việc tạo 1 index B-Tree hoặc index Hash theo order_date trên bảng orders là hoàn toàn hợp lí và nên làm, do nó thỏa mãn cả về điều kiện kích hoạt index theo giá trị cụ thể, bảng nhiều dữ liệu cần tổ chức và cột order_date ít khi bị cập nhật dữ liệu (hết 1 ngày mới cập nhật 1 lần, mà khi có giá trị mới thì luôn lớn hơn cũ nên B-Tree không mất quá nhiều công tổ chức lại thứ tự) giúp lọc ra những bản ghi phù hợp nhanh chóng. Câu lệnh tạo index tương tự câu 1:

  ```
  create index idx_orders_date on orders using btree (order_date);
  ```

  Tuy nhiên, ở 2 cách viết trên thì cách số 1 dùng truy vấn con, nó sẽ lọc ra các bản ghi customerid thỏa mãn, sau đó sẽ có thêm 1 bước gom nhóm theo các giá trị trùng nhau rồi mới so sánh với các bản ghi customer_id ở câu truy vấn chính, còn cách số 2 thì bộ xử lí câu hỏi sẽ dùng index lọc trên bảng orders trước theo điều kiện phép chọn, rồi mới thực hiện phép join với bảng orders với các giá trị customer_id thỏa mãn. Cả 2 cách đều có thể dùng index tuy nhiên cách số 2 nhanh hơn cách 1 do máy giảm được 1 bước gom nhóm các kết quả trùng trả về trong in.

  Ngoài ra, index B-Tree sẽ đa dụng hơn index Hash trong trường hợp này do việc tìm kiếm trên khoảng và điểm, nếu muốn tái sử dụng code để thống kê trong tháng, trong năm thì B-Tree sẽ đa dụng hơn.

3. Đưa ra sản phẩm được bán nhiều nhất trong tháng 6/2025.

Cách 1:

```
with temp as(
        select product_id, product_name, sum(quantity) as banra
        from orders
        join order_details using (order_id)
        join batch using (batch_id)
        join products using (product_id)
        where order_date between '2025/06/01' and '2025/06/30'
        group by product_id, product_name)
select product_id, product_name, banra
from temp
where banra = (select max(banra) from temp);
```

Cách 2:

```
with temp as(
        select product_id, product_name, sum(quantity) as banra
        from orders
        join order_details using (order_id)
        join batch using (batch_id)
        join products using (product_id)
        where order_date between '2025/06/01' and '2025/06/30'
        group by product_id, product_name)
select product_id, product_name, banra
from temp
where banra >= all(select banra from temp);
```

- Đối với câu hỏi này, ta nhận xét bảng orders vốn là bảng có nhiều dữ liệu, cột order_date được khởi tạo ít thay đổi, rõ ràng đặt index B-Tree trên cột order_date của bảng orders là vẫn hợp lí do nó đánh dấu trên khoảng >= và <= để tận dụng tối ưu sức mạnh của index. Cách đánh chỉ mục:

  ```
  create index idx_orders_date on orders using btree (order_date);
  ```

  Đối với 2 cách ở trên, cách số 1 theo thời gian lâu dài sẽ hiệu quả hơn cách số 2. Lí do nằm ở chỗ, cách số 1 sử dụng câu lệnh tính max chỉ 1 lần, rồi câu lệnh chính sẽ thực hiện so sánh xem bản ghi nào có giá trị bằng với max như các phép toán chọn thông thường theo quy tắc lưu trữ dạng đường ống. Còn ở cách số 2, mỗi lần đọc 1 bản ghi ở câu lệnh chính, banra sẽ phải duyệt qua với mọi giá trị banra ở bảng temp xem có thỏa mãn không mới in ra, nếu càng nhiều sản phẩm thì mỗi lần so sánh vậy sẽ rất chậm (theo quy tắc dạng vật chất hóa => liên quan đến bộ nhớ => chậm hơn là chắc chắn).

4. Thống kê số nhân viên mỗi ca làm trong tháng 6/2025.

```
select schedule_id, start_day, end_day, start_time, end_time, count(distinct employee_id) as soluong
from schedule
join working using (schedule_id)
where work_date between '2025/06/01' and '2025/06/30'
group by schedule_id;
```

- Có thể và nên sử dụng index B-Tree ở trên cột work_date của bảng working. Do cột work_date chứa các giá trị ngày tháng, trung bình 1 ngày có thể nhập đến 20 bản ghi nên số lượng bản ghi đủ lớn, dữ liệu trên cột lại không cần thay đổi nhiều nên cần thiết phải tạo 1 chỉ mục theo ngày:

  create index idx_working_date on working using btree (work_date);

  Index B-Tree sẽ thực hiện vai trò của nó để gom khoảng giá trị thỏa mãn để có thể lọc ra các giá trị trong tháng đó. (Số bản ghi trong 1 tháng lại đủ nhỏ so với số bản ghi tổng thể, rất thuận lợi cho việc gọi chỉ mục này).

5. **Tạo 1 khung nhìn cho biết các loại sản phẩm hiện có trong siêu thị và số lượng của mỗi loại. Sắp xếp kết quả theo thứ tự giảm dần về số lượng.**

```
create or replace view categories_number as
        select category_id, category_name, count(product_id) as soluong
        from categories
        join products using (category_id)
        group by category_id, category_name
        order by soluong desc;
```

6. **Cho biết nhân viên chăm chỉ nhất siêu thị vào tháng 5/2025 để quản lí tặng thưởng. Nhân viên chăm chỉ là nhân viên đi làm đủ từ 25 buổi/ tháng nhưng không được Vắng (nghỉ không phép) buổi nào.**

```
with temp as(
        select employee_id, count(*) as dilam
        from working
        where status = 'D' and (work_date between '2025/05/01' and '2025/05/31')
        group by employee_id
        having count(*) >= 25),
temp_absent as(
        select employee_id, count(*) as vang
        from working
        where status = 'V' and (work_date between '2025/05/01' and '2025/05/31')
        group by employee_id)
select employee_id, concat(lastname, ' ', firstname) as fullname, dilam
```

from employee

join temp using (employee_id)

left join temp_absent using (employee_id)

where dilam = (select max(dilam) from temp) and vang is null;

Câu này có thể cải thiện hiệu năng câu lệnh truy vấn bằng cách đặt index trên 2 điều kiện ở phép where trong bảng tạm. Vấn đề cần thiết ở đây là nên đặt index composite theo 2 cột (status, work_date) hay (work_date, status) hay 2 index trên 2 cột đơn lẻ status với workdate.

--create index idx_working_status_date on working using btree (status, work_date);

--create index idx_date_status on working using btree (work_date, status);

create index idx_working_date on working using btree (work_date);

--create index idx_working_status on working using btree (status);

Nhận xét do cột status ít dữ liệu khác nhau ('D', 'M', 'V', 'P') nên thứ tự bị lặp lại nhiều, việc tạo thêm 1 index ở đây rất lãng phí tài nguyên bộ nhớ. Index đơn trên work_date theo em đáp ứng được nhất trong trường hợp này, nó sẽ giúp lọc ra khoảng ngày tháng năm thỏa mãn (index theo ngày này còn được sử dụng nhiều trong các yêu cầu khác) rồi sau đó lấy ra các bản ghi có status phù hợp.

Đối với 2 index composite, hệ thống đòi hỏi phép where phải tuân theo thứ tự, nghĩa là (status, work_date) thì được chấp nhận, còn ngược lại thì không. Tuy nhiên do đã phân tích ở trên, việc quét toàn bộ trên status sẽ tối ưu hơn đặt index ở đây, nên do đó index composit (status, workdate) sẽ không được tối ưu. Nên đặt trên 1 cột work_date là phù hợp nhất.

7. **Viết hàm trả về các đơn hàng mà khách hàng đã mua với đầu vào là số điện thoại của khách, kết quả trả về là mã hóa đơn, ngày mua hàng, hình thức thanh toán, tổng số tiền phải trả.**

create or replace function order_bought(in v_phone varchar(20)) returns table (orderid int, orderdate date, method varchar(50), totalamount numeric(12,2)) as

$$

begin

return query select order_id, order_date, payment_method, total_amount

from orders

join customer using (customer_id)

where phone = v_phone;

```
end;
$$ language plpgsql;
select * from order_bought('0909123456');
```

8. Cho biết Công ty TNHH Sao Mai đã cung cấp những sản phẩm gì cho siêu thị.

```
select distinct product_id, product_name
from products
join batch using (product_id)
join import_reports using (import_id)
join suppliers using (supplier_id)
where supplier_name = 'Cong ty TNHH Sao Mai';
```

- Có thể đặt index trên supplier_name để truy vấn hiệu quả hơn đối với phép toán = đối với bảng suppliers nhiều dữ liệu. Câu lệnh tạo truy vấn:

```
create index idx_suppliers_name on suppliers using btree (supplier_name);
```

Tuy nhiên trên thực tế người ta chỉ lưu đầy đủ chứ tìm kiếm thì hiếm khi. Thường khi tìm tên công ty người ta chỉ tìm Sao Mai, hoặc cùng lắm là Công ty Sao Mai, do đó dòng where trên thực tế sẽ có dạng là (tùy vào tên của công ty nữa):

```
where supplier_name like 'Sao Mai%';
where supplier_name like '%Sao Mai';
where supplier_name like '%Sao Mai%';
```

hoặc thậm chí dùng where lower(supplier_name) like ‘%sao mai%’;

Như vậy trong 4 cách trên thì có đến 3 cách thường xuyên được dùng nhưng lại cấm index B-Tree (%name), do đó ở đây việc đặt index sẽ không tối ưu về việc sử dụng, mà lại mất thêm chi phí để bảo trì.

Một lí do nữa đó là cột supplier_name thường được lưu bằng varchar dài hoặc dạng text, khi đó sẽ mất thêm 1 bước tính toán giá trị xâu để lưu được khóa của index, rồi khi truy vấn lại mất công so sánh với các giá trị gây mất thời gian. Do đó viết truy vấn như câu trên kia đặt thì được nhưng không nên đặt index ở đây.

9. Viết 1 hàm có đầu vào là mã nhân viên, ngày đi làm và trả về số đơn mà nhân viên đó đã thực hiện trong ngày hôm đó.

```
create or replace function total_order (in v_employeeid int, v_date date) returns integer as
$$
```

```
declare v_count integer;
begin
        select into v_count count(order_id)
        from orders
        where employee_id = v_employeeid and order_date = v_date;
        return v_count;
end;
$$ language plpgsql;
select total_order(1, '2025/06/09');
```

10. **Thống kê các lô hàng có thể hết HSD trong tháng 7/2025. Thông tin lô hàng gồm mã lô, tên sản phẩm, số sản phẩm còn, hạn sử dụng của lô sắp xếp theo thứ tự tăng dần.**

```
select batch_id, product_name, batch.quantity_in_stock, expiration_date
from batch
join products using (product_id)
where expiration_date <= '2025/07/31'
order by expiration_date asc;
```

- Có thể và nên đặt index trên cột expiration_date của bảng batch.
- Lí do: index B-Tree hoàn toàn thỏa mãn điều kiện tìm khoảng giá trị <=, dữ liệu trên cột này cũng không được thay đổi và có sự đa dạng liên tục; bảng batch lại có số lượng lớn các lô hàng nhập về mỗi ngày. Đối với người sử dụng, nhất là ở vị trí quản lí, việc theo dõi các sản phẩm sắp hết HSD là rất cần thiết và thường xuyên, truy vấn sẽ được gọi nhiều lần và liên tục => đặt index là hoàn toàn chính xác.

```
create index idx_batch_exp on batch using btree (expiration_date);
```

11. **Đặt trigger để quản lí việc khi 1 lô hàng mới được nhập thì số lượng tồn kho của 1 sản phẩm sẽ được tăng lên tương ứng, kiểm tra cả trường hợp khi lô được bán ra hoặc tiêu hủy.**

INSERT:

```
create or replace function tf_after_insert_batch() returns trigger as
$$
begin
        update products
        set quantity_in_stock = quantity_in_stock + new.total_quantity
        where product_id = new.product_id;
```

```
        return new;
end;
$$ language plpgsql;

create trigger tg_after_insert_batch
after insert on batch
for each row
execute procedure tf_after_insert_batch();
```

DELETE:

```
create or replace function tf_after_delete_batch() returns trigger as
$$
begin
        update products
        set quantity_in_stock = quantity_in_stock - old.quantity_in_stock
        where product_id = old.product_id;
        return old;
end;
$$ language plpgsql;

create trigger tg_after_delete_batch
after delete on batch
for each row
execute procedure tf_after_delete_batch();
```

UPDATE:

```
create or replace function tf_after_update_batch() returns trigger as
$$
begin
        update products
        set quantity_in_stock = quantity_in_stock - old.quantity_in_stock +
new.quantity_in_stock
        where product_id = new.product_id;
        return new;
end;
$$ language plpgsql;

create trigger tg_after_update_batch
after update on batch
for each row
when (new.quantity_in_stock is distinct from old.quantity_in_stock)
execute procedure tf_after_update_batch();
```

12. **Đặt trigger để đảm bảo mỗi khi 1 sản phẩm trong lô hàng được bán ra thì số lượng tồn kho của lô đó giảm tương ứng.**

```
create or replace function tf_after_insert_order_details() returns trigger as
$$
begin
        update batch
        set quantity_in_stock = quantity_in_stock - new.quantity
        where batch_id = new.batch_id;
        return new;
end;
$$ language plpgsql;
create trigger tg_after_insert_order_details
after insert on order_details
for each row
execute procedure tf_after_insert_order_details();
```

13. Cho biết từ đầu năm 2025 có những lô hàng nào bị sai khác số lượng? Sắp xếp mức độ nghiêm trọng theo (số lượng thực tế - số lượng hệ thống) giảm dần.

```
select batch_id, product_name, check_details.quantity_in_stock, real_quantity, (check_details.quantity_in_stock - real_quantity) as thatthoat
from check_reports
join check_details using (report_id)
join batch using (batch_id)
join products using (product_id)
where check_date >= '2025/01/01' and (check_details.quantity_in_stock - real_quantity) > 0
order by thatthoat desc;
```

Có thể đặt Index B-Tree trên cột check_date do index B-Tree có thể tìm kiếm được thông tin theo khoảng để cải thiện tốc độ truy vấn. Câu lệnh tạo index:

```
create index idx_check_date on (check_reports) using btree (check_date);
```

Nhưng không nên đặt index ở trên bảng này. Do bảng check_reports tương đối nhỏ, cộng thêm việc ít khi truy vấn này được thực thi (trừ trường hợp siêu thị thường xuyên bị sai khác số lượng), nên việc duy trì 1 index ở đây là không cần thiết. Ngoài

ra em cũng đã cho chạy thử thì hệ thống vẫn ưu tiên dùng duyệt toàn bộ thay vì dùng index để lọc giá trị.

b. Lê Vũ Nguyên Hoàng (20235723):

1. Cho biết 5 sản phẩm có đơn giá cao nhất trong siêu thị

```
SELECT product_id, product_name, unit, price_with_tax
FROM products
ORDER BY price_with_tax DESC
LIMIT 5;
```

Vì truy vấn cần sắp xếp toàn bộ bảng theo price_with_tax giảm dần rồi lấy 5 dòng đầu tiên, việc tạo chỉ mục DESC trên cột này giúp truy cập trực tiếp các phần tử lớn nhất mà không cần sắp xếp toàn bảng.CREATE INDEX idx_products_price_desc ON products USING btree (price_with_tax DESC);

2. Hiển thị toàn bộ danh sách hóa đơn gồm mã hóa đơn, ngày đặt hàng, tên khách hàng, tổng tiền và hình thức thanh toán

```
SELECT o.order_id, o.order_date, c.fullname AS customer_name, o.total_amount, o.payment_method
FROM orders o
JOIN customer c ON o.customer_id = c.customer_id;
```

- Cách viết khác dùng subquery

```
SELECT o.order_id, o.order_date, (SELECT fullname FROM customer c WHERE c.customer_id = o.customer_id) AS customer_name, o.total_amount, o.payment_method
FROM orders o;
```

- Ở cách viết 2, hệ thống quét toàn bộ bảng orders và đối với mỗi bản ghi, hệ thống quét tiếp toàn bộ bảng customer để tìm bản ghi có customer_id trùng khớp. Do đó ở cách viết 1, sử dụng JOIN để ghép 2 bảng với nhau sẽ có tốc độ nhanh hơn

3. Thống kê số loại sản phẩm, số lượng hàng tồn kho của mỗi danh mục hàng

```
SELECT ca.category_name, COUNT(p.product_id) AS so_loai_san_pham, SUM(p.quantity_in_stock) AS tong_so_hang_trong_kho
FROM categories ca
LEFT JOIN products p ON ca.category_id = p.category_id
```

GROUP BY ca.category_name;

- Hệ thống quét toàn bộ bảng products để tìm các bản ghi có category_id tương ứng với từng bản ghi của bảng categories, sau đó nhóm các bản ghi theo danh mục hàng, rồi thực hiện COUNT và tính SUM

4. Thống kê số lần nhập hàng từ các nhà cung cấp, và đưa ra giá trị đơn nhập cao nhất và thấp nhất của từng nhà cung cấp

SELECT s.supplier_id, s.supplier_name, COUNT(ir.import_id) AS so_lan_nhap, MAX(ir.total_amount) AS tong_cao_nhat, MIN(ir.total_amount) AS tong_thap_nhat

FROM suppliers s

LEFT JOIN import_reports ir ON s.supplier_id = ir.supplier_id

GROUP BY s.supplier_id, s.supplier_name;

- Hệ thống quét toàn bộ bảng import_reports để tìm các bản ghi có supplier_id tương ứng với mỗi bản ghi của bảng suppliers, nhóm các bản ghi theo nhà cung cấp rồi COUNT, tìm MAX, MIN

5. Đưa ra danh sách khách hàng đã đến siêu thị mua hàng vào tháng 5 năm 2025

SELECT DISTINCT c.customer_id, c.fullname

FROM customer c

JOIN orders o ON c.customer_id = o.customer_id

WHERE o.order_date BETWEEN '2025-05-01' AND '2025-05-31'

ORDER BY c.fullname;

Truy vấn lọc các đơn hàng trong khoảng thời gian tháng 5/2025, đây là khoảng nhỏ so với toàn bộ dữ liệu.

Do đó, tạo chỉ mục với order_date giúp hệ thống nhanh chóng xác định các bản ghi phù hợp mà không cần quét toàn bộ bảng.

Đồng thời, customer_id cũng thường xuyên được sử dụng nên có thể tạo chỉ mục kết hợp để tăng tốc độ lọc thời gian và ghép bảng

CREATE INDEX idx_orders_date_customer ON orders(order_date, customer_id);

6. Liệt kê chi tiết các biên bản kiểm kê trong năm 2025

SELECT cr.report_id, cr.check_date, b.batch_id, b.product_id, b.quantity_in_stock AS so_luong_ghi_nhan, cd.real_quantity AS so_luong_thuc_te

```
FROM check_reports cr
JOIN check_details cd ON cr.report_id = cd.report_id
JOIN batch b ON cd.batch_id = b.batch_id
WHERE cr.check_date >= '2025-01-01'
ORDER BY cr.check_date, cr.report_id, b.batch_id;
```

Truy vấn lọc theo check_date trong một khoảng thời gian cụ thể, nên có thể tạo chỉ mục giúp truy vấn phạm vi hiệu quả.

```
CREATE INDEX idx_checkreports_date ON check_reports(check_date);
```

7. Tính lợi nhuận (thu - chi) của siêu thị trong tháng 5/2025

```
WITH thu AS (
  SELECT SUM(total_amount) AS tong_thu
  FROM orders
  WHERE order_date BETWEEN '2025-05-01' AND '2025-05-31'
),
chi_luong AS (
  SELECT SUM(actual_salary) AS tong_chi_luong
  FROM salary
  WHERE monthyear = '5/2025'
),
chi_nhap AS (
  SELECT SUM(total_amount) AS tong_chi_nhap
  FROM import_reports
  WHERE import_date BETWEEN '2025-05-01' AND '2025-05-31'
)
SELECT
  thu.tong_thu,
  chi_luong.tong_chi_luong,
  chi_nhap.tong_chi_nhap,
```

thu.tong_thu - chi_luong.tong_chi_luong - chi_nhap.tong_chi_nhap AS loi_nhuan

FROM thu, chi_luong, chi_nhap;

Các truy vấn con đều lọc theo khoảng thời gian ngắn, do đó có thể tạo chỉ mục trên các cột thời gian trong các bảng để tăng hiệu suất lọc thời gian.

```
CREATE INDEX idx_orders_date ON orders(order_date);

CREATE INDEX idx_salary_monthyear ON salary(monthyear);

CREATE INDEX idx_import_date ON import_reports(import_date);
```

8. Tạo bảng view thống kê tổng chi tiêu của mỗi khách hàng

```
CREATE OR REPLACE VIEW customer_spending_view AS

SELECT c.customer_id, c.fullname, COUNT(o.order_id) AS so_don_hang, SUM(o.total_amount) AS tong_chi_tieu

FROM customer c

LEFT JOIN orders o ON c.customer_id = o.customer_id

GROUP BY c.customer_id, c.fullname;
```

9. Hàm thống kê số ngày đi làm đúng giờ, đi muộn và vắng của nhân viên theo ID trong tháng 5 năm 2025

```
CREATE OR REPLACE FUNCTION employee_attendance_summary(v_employee_id INT)

RETURNS TABLE (

  employee_id INT,

  so_ngay_dung_gio INT,

  so_ngay_di_muon INT,

  so_ngay_vang INT

) AS

$$

BEGIN

  RETURN QUERY

  SELECT

    w.employee_id,
```

```
    COUNT(*) FILTER (WHERE w.status = 'D')::INT AS so_ngay_dung_gio,
    COUNT(*) FILTER (WHERE w.status = 'M')::INT AS so_ngay_di_muon,
    COUNT(*) FILTER (WHERE w.status = 'V')::INT AS so_ngay_vang
  FROM working w
  WHERE w.employee_id = v_employee_id
   AND w.work_date BETWEEN DATE '2025-05-01' AND DATE '2025-05-
31'
  GROUP BY w.employee_id;
END;
$$ LANGUAGE plpgsql;
SELECT * FROM employee_attendance_summary(1);
```

- Khi nhập vào employee_id, hệ thống sẽ quét bảng working để tìm ra bản ghi có employee_id tương ứng và có ngày nằm trong tháng 5 năm 2025, sau đó thống kê trạng thái từng ngày theo 3 loại đúng giờ, muộn, vắng và trả kết quả ra một bảng mới

10. Hàm đưa ra các sản phẩm sắp hết hạn sau số ngày nhập vào

```
CREATE OR REPLACE FUNCTION expiring_batches(n_days INT)
RETURNS TABLE (
  batch_id INT,
  product_name VARCHAR(100),
  expiration_date DATE,
  quantity_in_stock INT,
  days_left INT
) AS
$$
BEGIN
  RETURN QUERY
  SELECT
    b.batch_id,
    p.product_name,
```

```
    b.expiration_date,
    b.quantity_in_stock,
    (b.expiration_date - CURRENT_DATE)::INT AS days_left
  FROM batch b
  JOIN products p ON b.product_id = p.product_id
  WHERE b.expiration_date BETWEEN CURRENT_DATE AND CURRENT_DATE + n_days;
END;
$$ LANGUAGE plpgsql;
SELECT * FROM expiring_batches(200);
```

Vì truy vấn lọc theo cột expiration_date trong một khoảng giới hạn (ngày hiện tại đến n ngày sau), nên có thể tạo chỉ mục để tăng hiệu suất lọc thời gian.

```
CREATE INDEX idx_batch_expiration_date ON batch(expiration_date);
```

11. Thống kê số đơn hàng theo từng ngày

```
SELECT order_date, COUNT(order_id) AS so_don_hang, SUM(total_amount)
AS tong_doanh_thu
FROM orders
GROUP BY order_date
ORDER BY order_date;
```

Vì truy vấn nhóm và sắp xếp theo order_date, hệ thống cần quét và nhóm theo cột này. Do đó có thể tạo chỉ mục với order_date để tăng hiệu suất nhóm và sắp xếp.

```
CREATE INDEX idx_orders_orderdate ON orders(order_date);
```

12. Tạo view tổng nhập hàng theo tháng

```
CREATE OR REPLACE VIEW monthly_import_summary AS
SELECT
  TO_CHAR(import_date, 'YYYY-MM') AS thang_nhap,
  COUNT(import_id) AS so_lan_nhap,
  SUM(total_amount) AS tong_gia_tri_nhap
FROM import_reports
GROUP BY TO_CHAR(import_date, 'YYYY-MM')
ORDER BY thang_nhap;
```

Truy vấn tổng hợp theo tháng dựa vào import_date, nên tạo chỉ mục trên cột này sẽ giúp hệ thống tăng tốc độ nhóm theo tháng.

```
CREATE INDEX idx_importreports_importdate ON import_reports(import_date);
```

13. Tạo view xem các sản phẩm bán chạy nhất

```
CREATE OR REPLACE VIEW top_selling_products AS
SELECT
   p.product_id,
   p.product_name,
   SUM(od.quantity) AS tong_so_luong_ban
FROM products p
JOIN batch b ON p.product_id = b.product_id
JOIN order_details od ON b.batch_id = od.batch_id
GROUP BY p.product_id, p.product_name
ORDER BY tong_so_luong_ban DESC;
```

14. Liệt kê các lô hàng đã bị lập biên bản xử lý kèm tên sản phẩm

```
SELECT
   ir.incident_id,
   ir.report_date,
   ir.report_time,
   ir.description,
   b.batch_id,
   b.expiration_date,
   p.product_name
FROM incident_reports ir
JOIN batch b ON ir.batch_id = b.batch_id
JOIN products p ON b.product_id = p.product_id;
```

15. Thống kê tổng lương thực tế đã trả cho từng nhân viên trong năm 2025

```
SELECT
   e.employee_id,
   e.lastname || ' ' || e.firstname AS hoten,
   SUM(s.actual_salary) AS tong_luong
FROM salary s
JOIN employee e ON s.employee_id = e.employee_id
WHERE s.monthyear LIKE '%2025%'
GROUP BY e.employee_id, hoten;
```

c. Phạm Đào Việt Hoàng (20235727):

1. Tổng số đơn hàng mỗi ngày trong tháng 6/2025

```
SELECT
  order_date,
```

```
  COUNT(*) AS num_orders
FROM orders
WHERE order_date BETWEEN '2025-06-01' AND '2025-06-30'
GROUP BY order_date
ORDER BY order_date;
```

Điều kiện lọc order_date BETWEEN … bao phủ 30 ngày (~30/365 ≈ 8%) khá nhỏ so với toàn bộ tổng số đơn hàng

Index B-Tree trên order_date sẽ nhanh chóng khoanh vùng các bản ghi trong khoảng, tránh quét toàn bộ.

```
CREATE INDEX idx_orders_orderdate ON orders(order_date);
```

2. Danh sách sản phẩm có tồn kho dưới 10 đơn vị

```
SELECT
  b.product_id,
  p.product_name,
  SUM(b.quantity_in_stock) AS total_in_stock
  FROM batch b
  JOIN products p USING(product_id)
  GROUP BY b.product_id, p.product_name
  HAVING SUM(b.quantity_in_stock) < 10;
```

Có trên batch(product_id): product_id là FK, B-Tree index giúp join nhanh với products và gom nhóm trên cột này. Khi GROUP BY, engine có thể sử dụng index ordering của product_id để nhóm liên tục mà không phải sort ngoài.

```
CREATE INDEX idx_batch_product ON batch(product_id);
```

3. Top 3 khách hàng chi tiêu cao nhất

```
SELECT c.customer_id, c.fullname, SUM(o.total_amount) AS total_spent
FROM customer c
JOIN orders o USING(customer_id)
GROUP BY c.customer_id, c.fullname
```

```
ORDER BY total_spent DESC
LIMIT 3;
```

Có thể dùng index vì Join theo customer_id diễn ra nhiều, B-Tree trên orders(customer_id) cho phép nhanh chóng tìm tất cả đơn của một khách mà không full scan orders. Đồng thời, khi thực hiện GROUP BY và ORDER BY, engine có thể tận dụng phần ordering trong index để tạo trước dãy giá trị customer_id, giảm chi phí sort kết quả cuối.

```
CREATE INDEX idx_orders_customer ON orders(customer_id);
```

4. **Sản phẩm chưa bán (không xuất hiện trong order_details)**

```
SELECT p.product_id, p.product_name
FROM products p
WHERE NOT EXISTS (
  SELECT 1
  FROM batch b
  JOIN order_details od USING(batch_id)
  WHERE b.product_id = p.product_id
);
```

Có thể dùng index vì Ở mệnh đề con, b.product_id = p.product_id sẽ lặp qua từng p; nếu index trên batch(product_id), hệ thống chỉ theo index tìm liền các batch cùng sản phẩm rồi kiểm tra tiếp order_details. B-Tree cho phép range scan nhanh nhóm batch của mỗi sản phẩm. Mặt khác, index trên order_details(batch_id) (FK) vốn đã có sẵn cũng hỗ trợ join tiếp.

```
CREATE INDEX idx_batch_product ON batch(product_id);
```

5. **Doanh thu theo nhóm danh mục trong tháng 5/2025**

```
SELECT c.category_name, SUM(od.quantity*p.price_with_tax) AS revenue

FROM orders o

JOIN order_details od USING(order_id)

JOIN batch b USING(batch_id)

JOIN products p USING(product_id)

JOIN categories c USING(category_id)

WHERE o.order_date BETWEEN '2025-05-01' AND '2025-05-31'

GROUP BY c.category_name

ORDER BY revenue DESC;
```

Có thể dùng index vì orders(order_date): B-Tree index tối ưu truy vấn range tháng 5, khoanh vùng chỉ ~8% dữ liệu.

order_details(order_id): index giúp join nhanh orders → order_details.

Các join tiếp theo trên batch, products, categories đều tận dụng FK index. Nhờ đó, bộ máy planner chọn kế hoạch index-nested-loop thay vì full scan nhiều bảng.

CREATE INDEX idx_orders_date ON orders(order_date);

CREATE INDEX idx_od_order ON order_details(order_id);

6. **Liệt kê sản phẩm và tổng số lượng bán được ngoài tháng 5/2025:**

```
SELECT
  p.product_id,
  p.product_name,
  SUM(od.quantity) AS total_quantity_sold
FROM orders o
JOIN order_details od USING(order_id)
JOIN batch b USING(batch_id)
JOIN products p USING(product_id)
WHERE o.order_date < '2025-05-01' OR o.order_date > '2025-05-31'
GROUP BY p.product_id, p.product_name
ORDER BY total_quantity_sold DESC;
```

Không thể dùng index vì số lượng sản phẩm được bán ngoài tháng 5 là rất lớn so với số lượng sản phẩm bán trong tháng 5. Do đó khi sử dụng index sẽ chậm hơn rất nhiều so với việc tìm kiếm tuần tự. Ngoài ra đây còn có phép OR, một phép toán không sử dụng được index

7. **Danh sách nhân viên từng làm ca “15:00–22:00”**

```
SELECT DISTINCT e.employee_id, e.lastname||' '||e.firstname AS fullname
FROM working w
JOIN schedule s USING(schedule_id)
JOIN employee e USING(employee_id)
WHERE s.start_time='15:00:00' AND s.end_time='22:00:00';
```

Có thể dùng index vì Hai điều kiện s.start_time và s.end_time cùng lúc phù hợp cho composite index (start_time,end_time). B-Tree composite giúp planner nhanh xác định ca đúng khung, chỉ scan những leaf page cần, rồi mới join tiếp đến working và employee.

CREATE INDEX idx_schedule_time ON schedule(start_time,end_time);

8. Số đơn hàng theo phương thức thanh toán

SELECT payment_method, COUNT(*) AS cnt

FROM orders

GROUP BY payment_method;

Không thể dùng index vì payment_method chỉ vài giá trị (ví dụ 'Tien mat', 'Chuyen khoan'), selectivity rất thấp, B-Tree index phân tán kém, và optimizer ưu tiên full table scan nhanh hơn lookup index rồi đọc hàng loạt pages lặp lại nhiều.

9. Danh sách lô hàng sắp hết hạn trong 7 ngày tới

SELECT batch_id, expiration_date, quantity_in_stock

FROM batch

WHERE expiration_date BETWEEN CURRENT_DATE AND CURRENT_DATE+INTERVAL'7 days';

Có thể dùng index vì Khoảng 7 ngày rất nhỏ so với thời gian tồn tại lô hàng (có thể nhiều tháng/năm), selectivity cao. B-Tree trên batch(expiration_date) giúp range scan chính xác, chỉ truy cập leaf pages chứa các ngày sắp hết, tránh full scan.

CREATE INDEX idx_batch_expiry ON batch(expiration_date);

10. Tổng số lô hàng nhập theo nhà cung cấp

SELECT s.supplier_id, s.supplier_name, COUNT(im.import_id) AS import_count

FROM suppliers s

LEFT JOIN import_reports im USING(supplier_id)

GROUP BY s.supplier_id, s.supplier_name

ORDER BY import_count DESC;

Có thể dùng index vì Join dựa trên supplier_id – B-Tree trên import_reports(supplier_id) giúp nhanh chóng tập hợp các phiếu nhập của mỗi supplier, sau đó GROUP BY & ORDER BY trên kết quả nhóm cũng tận dụng ordering trong index phần nào.

CREATE INDEX idx_import_supplier ON import_reports(supplier_id);

11. Doanh thu trung bình mỗi đơn hàng theo tháng năm 2025

```
SELECT to_char(order_date,'MM/YYYY') AS monthyear, AVG(total_amount) AS
avg_revenue
FROM orders
WHERE order_date BETWEEN '2025-01-01' AND '2025-12-31'
GROUP BY monthyear
ORDER BY monthyear;
```

Có thể dùng index vì Dù dùng to_char trong SELECT, điều kiện WHERE order_date BETWEEN … vẫn tận dụng B-Tree trên orders(order_date) để lọc trước dữ liệu cho một năm. Sau đó, grouping trên kết quả con đã được giới hạn.

```
CREATE INDEX idx_orders_date_fullyear ON orders(order_date);
```

12. Nhân viên không từng làm ca nào trong tháng 5/2025

```
SELECT e.employee_id, e.lastname||' '||e.firstname AS fullname
FROM employee e
WHERE NOT EXISTS (
  SELECT 1
  FROM working w
  WHERE w.employee_id=e.employee_id
    AND w.work_date BETWEEN '2025-05-01' AND '2025-05-31'
);
```

Có thể dùng index vì Correlated subquery lọc theo w.employee_id + range w.work_date rất phù hợp với composite index (employee_id, work_date) trên working. B-Tree composite cung cấp lookup nhanh những ngày của mỗi nhân viên, giúp NOT EXISTS loại ngay những e có lịch, hiệu quả hơn full scan.

```
CREATE INDEX idx_working_emp_date ON working(employee_id, work_date);
```

C. NHẬN XÉT – NÂNG CAO:

1. Kết quả tự đánh giá:

a. Kết quả chung:

- Đã biết cách viết bài mô tả nghiệp vụ, phân biệt với mô tả chức năng ứng dụng. Phần này nhóm cũng đã có thêm 1 hoạt động trải nghiệm đó là đi phỏng vấn từ các nhân viên ở siêu thị, nhờ đó am hiểu thêm về nghiệp vụ thực tế mà các chị phải làm mỗi ngày. Đồng thời việc mua 1 hóa đơn của siêu thị cũng là tài liệu quý giá để nhóm tham khảo những gì mà người mua cần quan tâm trên hóa đơn.
- Sơ đồ thực thể liên kết tự thiết kế hoàn toàn.
- Viết 10 câu truy vấn cho mỗi thành viên, có giải thích kèm chú ý về cách đặt chỉ mục nếu có.
- Xây dựng được 1 chương trình kết nối CSDL market.

b. Điểm mới của bài toán:

- Bài toán xây dựng CSDL quản lí siêu thị có sự mở rộng hơn về quản lí HSD, nhờ có sự trợ giúp của giảng viên về ý tưởng lô nên nhóm đã thực hiện bổ sung. Ngoài ra đối với mỗi lô nhập còn có cả biên bản kiểm kê, biên bản xử lí lô hỏng.
- Mở rộng từ phiếu nhập hàng thành biên bản nhập hàng: đây là cụm từ chính xác từ siêu thị để quản lí cụ thể các sản phẩm lô được nhập khi nào.
- Phần quản lí chung có thêm mục Lịch làm việc và Lương thưởng cho nhân viên.

2. Nâng cao – Gợi mở:

- Nếu siêu thị phát triển lâu dài, hoàn toàn có thể thêm phần Khuyến mãi và Điểm thưởng khách hàng vào để có thể phát triển.
- Phần bảng lương còn nhập thủ công, chưa có đủ kịp thời gian để thiết kế nhập tự động cho đúng với tính chất thuộc tính tự suy.

3. Khó khăn:

- Chắc chắn phần khó nhất trong CSDL chính là vẽ được sơ đồ ERD. Sau 1 buổi được cô chữa trên lớp, nhóm cũng đã phải tự về vẽ lại toàn bộ sơ đồ để theo đúng tính chất mô tả nghiệp vụ.
- Phần kết nối CSDL và xây dựng chương trình tương đối nhiều mà bài hướng dẫn kết nối trong chương trình lại ở khá muộn, do đó nhóm phải gấp rút thực hiện nên vẫn còn nhiều chỗ chưa thực sự mượt mà.

4. Phân công nhiệm vụ:

- Nhóm 4 hoàn toàn không có phân công nhiệm vụ, nhóm trưởng đã xác định tinh thần này ngay từ đầu. Các bạn phải lắng nghe nhau trong mọi công việc và tham gia ở mọi khâu của bài toán thiết kế. Ở bước thiết kế (B.1 – B.4) thông thường cứ sau 1 tuần nhóm trưởng sẽ tổ chức buổi họp, đề ra các nhiệm vụ cần làm trong tuần (viết nghiệp vụ, chức năng, vẽ ERD...) và các bạn trong nhóm chủ động báo cáo. Mọi người đều phải tự làm để thực sự hiểu về hệ thống và CSDL mà mình thiết kế để có thể bổ sung và tranh biện với nhau. Ở phần lập trình và khai thác dữ liệu, các thành viên xác định nhiệm vụ là cần đưa market lên máy tính, viết 10 câu truy vấn, xây dựng 1 chương trình. Sẽ có người đảm nhiệm phần giao diện, có người làm phần code và CSDL. Tuy nhiên trong quá trình làm luôn hỏi đáp, bổ sung cho nhau và thậm chí nếu đang làm mà người này bận thì người kia hoàn toàn sẵn sàng làm tiếp các phần còn lại. Do vậy, phần phân công nhiệm vụ không có sự rõ ràng mà làm chung tất cả nên nhóm cũng chưa biết nên ghi thế nào cho đúng, nếu có thì chắc ở phần cuối cùng nhóm trưởng sẽ đảm nhiệm phần báo cáo với slide thuyết trình, nhưng trong thời gian đó thì các bạn cũng đóng vai trò tester kiểm định chương trình Java và bổ sung các chức năng như View, Trigger nếu còn thiếu.

5. Phụ lục:

***Nội dung cập nhật so với bản cũ:**

Bản hiện tại: Bổ sung hoàn chỉnh nội dung theo yêu cầu để nộp chính thức.

Bản chính thức 9/5:

Cập nhật phiên bản tiếng Anh cho CSDL để bắt đầu code.

Bản 4:

Cập nhật lại thông tin báo cáo và loại bỏ câu chuyện cho báo cáo nghiêm túc.

Bản 3:

- Nghiệp vụ: bổ sung thêm 1 câu chuyện cho dễ tưởng tượng, sau này đến báo cáo chính sẽ xóa. Bổ sung thêm một số nghiệp vụ quan trọng: quản lí lô cho vấn đề hạn sử dụng; quản lí phiếu kiểm kê, biên bản hủy bỏ hàng hóa...
- Chức năng: bổ sung chức năng xem thống kê các sản phẩm sắp hết, các sản phẩm sắp hết HSD, sản phẩm không bán được.
- Cập nhật lại sơ đồ ERD, sơ đồ quan hệ dựa trên quan hệ ngữ nghĩa nghiệp vụ, chưa áp dụng kiến thức phụ thuộc hàm do chưa học.

Bản 2:

- Sửa đổi sơ đồ ERD phần doanh thu.

Bản 1:

- Chỉnh sửa lại chi tiết phần mô tả chức năng theo cấu trúc Input – Output đối với từng chức năng cụ thể.
- Hiện tại nội dung liên quan đến Khuyến mãi bị loại bỏ.
- Xây dựng sơ đồ thực thể liên kết và sơ đồ quan hệ.

Lời kết

Nhóm xin chân thành cảm ơn giảng viên TS. Nguyễn Thị Oanh đã nhiệt tình với lớp mỗi tuần học. Đặc biệt là sau buổi nhận xét trình bày sản phẩm thử, nhờ đó mà các thành viên đã có một bài học đáng nhớ về việc tự thiết kế sản phẩm mà không dựa vào 1 DB mẫu nào có sẵn, phải tự lên ý tưởng và xóa đi vẽ lại sơ đồ từ đầu... Càng thiết kế lại thấy mình càng chưa đủ, những kiến thức ở những buổi lí thuyết còn ít quá. Vậy mà đến buổi cuối cùng, các thành viên cũng tự vẽ được 1 DB "nhà làm" (xin tự nhận là "chỉn chu") để tự tin đi báo cáo. Học ở giảng viên, nhóm cũng học được nhiều bài học về cách tổ chức công việc, sự nghiêm túc trong bài giảng và lắng nghe bài trình bày của sinh viên. Nhóm chúc giảng viên luôn nhiều sức khỏe, đạt được nhiều thành công hơn nữa trong sự nghiệp trồng "trẻ lớn".

Nhóm chúc cô Oanh một Hè vui!

Nhóm 4_156778_IT3290