**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

**KHOA CÔNG NGHỆ THÔNG TIN 1**

****

**BÀI TẬP CƠ SỞ DỮ LIỆU PHÂN TÁN**

**BÀI TẬP 1: THIẾT KẾ CƠ SỞ DỮ LIỆU QUẢN LÝ CỬA HÀNG TRỰC TUYẾN**

|  |  |
| --- | --- |
| **Giảng viên hướng dẫn** | **: TS NGUYỄN THỊ THU THỦY** |
| **Họ và tên sinh viên** | **: ĐINH QUANG HƯNG** |
| **Mã sinh viên** | **: B22DCCN407** |
| **Lớp** | **: D22CQCN11-B** |
| **Nhóm** | **: 08** |

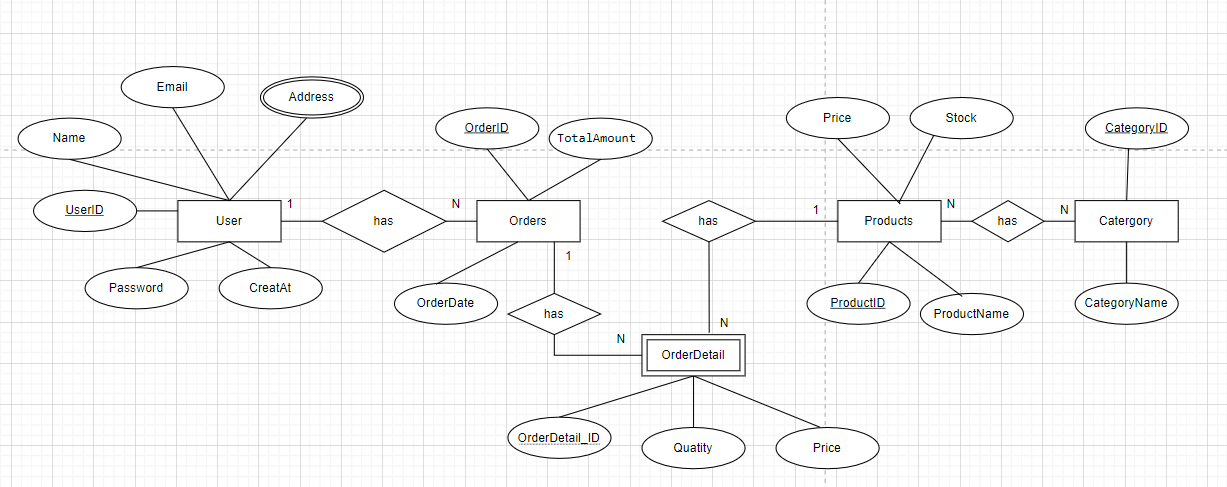
***Hà Nội – 2025***

**NỘI DUNG**

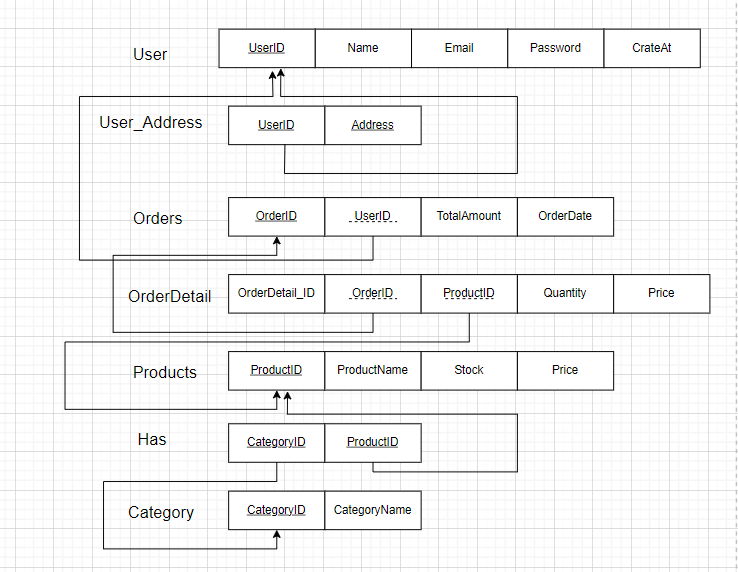
**Phần 1: Tạo 1 hệ CSDL Đơn giản**

Một cửa hàng trực tuyến cho phép người dùng đăng ký tài khoản, mua hàng và theo dõi đơn hàng. Mỗi người dùng(User) có các thuộc tính: UserID, Name, Email, Password, CreatAt, Address.Trong đó thì mỗi người dùng sẽ có nhiều địa chỉ(Address). Đơn hàng(Order) có các thuộc tính: OrderID, TotalAmount, OrderDate, mỗi người dùng có thể có nhiều đơn hàng, mỗi đơn hàng chỉ của 1 người dùng. Sản phẩm (Products) gồm có các thuộc tính: ProductID, Price, Stock, ProductName. Thể loại sản phẩm(Category) gồm có các thuộc tính: CategoryID, CategoryName. Mỗi thể loại có nhiều loại sản phẩm khác nhau, và mỗi sản phẩm có nhiều thể loại . Mỗi đơn hàng(Orders) có nhiều chi tiết đơn hàng(OrderDetail) và mỗi sản phẩm(Product) có thể xuất hiện trong nhiều chi tiết đơn hàng. Chi tiết đơn hàng có thể có những thuộc tính: Price, Quanlity.

**Phần 2: Sơ đồ ERD:**



**Phần 3:Ánh xạ sang lược đồ quan hệ:**



-Quan hệ N-N giữa Categories và Products được tách thành 1 bảng riêng với lý do:

+ Tránh trùng lặp dữ liệu.

+ Tránh việ khó mở rộng: Nếu một sản phẩm thuộc nhiều danh mục (ví dụ: iPhone vừa là "Điện thoại" vừa là "Thiết bị cao cấp"), ta phải thêm nhiều dòng cho cùng một sản phẩm trong bảng Products, hoặc thêm nhiều cột (như CategoryID1, CategoryID2, ...), dẫn đến thiết kế không linh hoạt và phức tạp.

+ Tránh vi phạm chuẩn hóa: Chuẩn hóa cơ sở dữ liệu yêu cầu loại bỏ trùng lặp dữ liệu và đảm bảo mỗi bảng chỉ chứa thông tin liên quan trực tiếp đến thực thể của nó. Quan hệ N-N không thể được biểu diễn trực tiếp trong một bảng mà vẫn tuân thủ các dạng chuẩn như 3NF

-Thực thể OrderDetail là thực thể yếu mặc dù OrderDetailID là duy nhất trong bảng OrderDetails, nhưng trong thực tế, nó không mang ý nghĩa định danh đầy đủ nếu không gắn với OrderID và khóa của thực thể mạnh (OrderID) là một phần không thể thiếu để xác định thực thể yếu OrderDetail.

**Phần 4: Luyện tập việc cài đặt với hệ quản trị CSDL SQL Server**

**SQL Server:** là một hệ quản trị cơ sở dữ liệu quan hệ do Microsoft phát triển. Nó được thiết kế để lưu trữ, quản lý và truy xuất dữ liệu một cách hiệu quả, phục vụ cho nhiều loại ứng dụng từ nhỏ đến lớn, như ứng dụng doanh nghiệp, phân tích dữ liệu, và trí tuệ nhân tạo. SQL Server hỗ trợ ngôn ngữ truy vấn SQL

-- Tạo CSDL

CREATE DATABASE ShopDB;

-- Bảng Users

CREATE TABLE Users (

UserID INT PRIMARY KEY,

Name VARCHAR(100) NOT NULL,

Email VARCHAR(100) UNIQUE NOT NULL,

Password VARCHAR(255) NOT NULL,

CreatedAt DATE

);

--Bảng User\_Address

CREATE TABLE User\_Address (

UserID INT,

Addres VARCHAR(100),

FOREIGN KEY(UserID) REFERENCES Users(UserID),

PRIMARY KEY(UserID,Addres)

)

-- Bảng Categories

CREATE TABLE Categories (

CategoryID INT PRIMARY KEY,

CategoryName VARCHAR(100) NOT NULL

);

-- Bảng Products

CREATE TABLE Products (

ProductID INT PRIMARY KEY,

ProductName VARCHAR(255) NOT NULL,

Price INT NOT NULL,

Stock INT NOT NULL

);

-- Bảng Orders

CREATE TABLE Orders (

OrderID INT PRIMARY KEY,

UserID INT,

OrderDate DATE,

TotalAmount INT NOT NULL,

FOREIGN KEY (UserID) REFERENCES Users(UserID)

);

-- Bảng OrderDetails

CREATE TABLE OrderDetails (

OrderDetailID INT PRIMARY KEY,

OrderID INT,

ProductID INT,

Quantity INT NOT NULL,

Price INT NOT NULL,

FOREIGN KEY (OrderID) REFERENCES Orders(OrderID),

FOREIGN KEY (ProductID) REFERENCES Products(ProductID)

);

-- Bảng Has

CREATE TABLE Has (

CategoryID INT,

ProductID INT,

FOREIGN KEY (CategoryID) REFERENCES Categories(CategoryID),

FOREIGN KEY (ProductID) REFERENCES Products(ProductID)

);

**Phần 5: Thêm dữ liệu vào SQL Server**

-- Du lieu cho bang Users

INSERT INTO Users (UserID, Name, Email, Password, CreatedAt) VALUES

(1, 'Nguyen Van A', 'nguyenvana@gmail.com', 'pass123', '2024-01-15'),

(2, 'Tran Thi B', 'tranthib@gmail.com', 'abc456', '2024-02-20'),

(3, 'Le Van C', 'levanc@gmail.com', 'xyz789', '2024-03-10'),

(4, 'Pham Thi D', 'phamthid@gmail.com', 'def012', '2024-04-05'),

(5, 'Hoang Van E', 'hoangvane@gmail.com', 'ghi345', '2024-05-12'),

(6, 'Bui Thi F', 'buithif@gmail.com', 'jkl678', '2024-06-18'),

(7, 'Dang Van G', 'dangvang@gmail.com', 'mno901', '2024-07-23'),

(8, 'Vu Thi H', 'vuthih@gmail.com', 'pqr234', '2024-08-30'),

(9, 'Do Van I', 'dovani@gmail.com', 'stu567', '2024-09-15'),

(10, 'Ngo Thi K', 'ngothik@gmail.com', 'vwx890', '2024-10-20');

-- Du lieu cho bang User\_Address

INSERT INTO User\_Address (UserID, Addres) VALUES

(1, '123 Duong Lang, Ha Noi'),

(2, '45 Nguyen Trai, TP.HCM'),

(3, '78 Hung Vuong, Da Nang'),

(4, '12 Le Loi, Hue'),

(5, '90 Tran Phu, Nha Trang'),

(6, '34 Bach Dang, Hai Phong'),

(7, '56 Pham Ngoc Thach, Can Tho'),

(8, '89 Ly Thuong Kiet, Vung Tau'),

(9, '23 Nguyen Hue, Quy Nhon'),

(10, '67 Dien Bien Phu, Da Lat');

-- Du lieu cho bang Categories

INSERT INTO Categories (CategoryID, CategoryName) VALUES

(1, 'Dien thoai'),

(2, 'Laptop'),

(3, 'May tinh bang'),

(4, 'Phu kien'),

(5, 'Tai nghe'),

(6, 'Loa'),

(7, 'Dong ho thong minh'),

(8, 'May anh'),

(9, 'Tivi'),

(10, 'Thiet bi gia dung');

-- Du lieu cho bang Products

INSERT INTO Products (ProductID, ProductName, Price, Stock) VALUES

(1, 'iPhone 14 Pro', 25000000, 50),

(2, 'Samsung Galaxy S23', 20000000, 60),

(3, 'MacBook Air M2', 30000000, 30),

(4, 'Dell XPS 13', 28000000, 25),

(5, 'iPad Pro 2023', 22000000, 40),

(6, 'Samsung Tab S9', 18000000, 45),

(7, 'AirPods Pro', 5500000, 100),

(8, 'Sony WH-1000XM5', 8500000, 80),

(9, 'Apple Watch Series 8', 12000000, 35),

(10, 'Canon EOS R50', 23000000, 20);

-- Du lieu cho bang Orders

INSERT INTO Orders (OrderID, UserID, OrderDate, TotalAmount) VALUES

(1, 1, '2025-01-05', 25000000),

(2, 2, '2025-01-06', 20000000),

(3, 3, '2025-01-07', 30000000),

(4, 4, '2025-01-08', 28000000),

(5, 5, '2025-01-09', 22000000),

(6, 6, '2025-01-10', 18000000),

(7, 7, '2025-01-11', 5500000),

(8, 8, '2025-01-12', 8500000),

(9, 9, '2025-01-13', 12000000),

(10, 10, '2025-01-14', 23000000);

-- Du lieu cho bang OrderDetails

INSERT INTO OrderDetails (OrderDetailID, OrderID, ProductID, Quantity, Price) VALUES

(1, 1, 1, 1, 25000000),

(2, 2, 2, 1, 20000000),

(3, 3, 3, 1, 30000000),

(4, 4, 4, 1, 28000000),

(5, 5, 5, 1, 22000000),

(6, 6, 6, 1, 18000000),

(7, 7, 7, 1, 5500000),

(8, 8, 8, 1, 8500000),

(9, 9, 9, 1, 12000000),

(10, 10, 10, 1, 23000000);

-- Du lieu cho bang Has

INSERT INTO Has (CategoryID, ProductID) VALUES

(1, 1),

(1, 2),

(2, 3),

(2, 4),

(3, 5),

(3, 6),

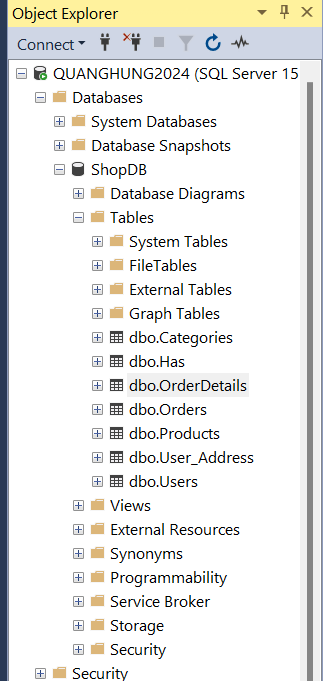
(5, 7),

(5, 8),

(7, 9),

(8, 10);

**Các bảng sau khi thêm dữ liệu:**



**Phần 6: Luyện tập truy vấn dữ liệu**

--Truy vấn 1:Tìm người dùng chi tiêu nhiều hơn trung bình tổng chi tiêu của tất cả người dùng

SELECT u.UserID, u.Name, SUM(o.TotalAmount) AS TongChiTieu

FROM Users u

JOIN Orders o ON u.UserID = o.UserID

GROUP BY u.UserID, u.Name

HAVING SUM(o.TotalAmount) > (

SELECT AVG(TongChiTieuTrungBinh)

FROM (

SELECT SUM(TotalAmount) AS TongChiTieuTrungBinh

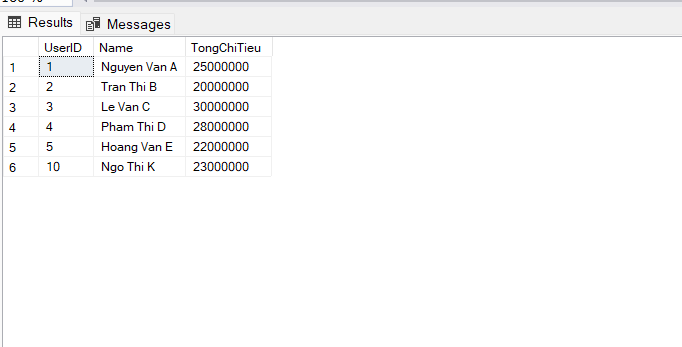
FROM Orders

GROUP BY UserID

) AS TrungBinh

);

Kết quả:



-- Truy vấn 2:Đếm số lượng sản phẩm trong mỗi danh mục

SELECT c.CategoryName, COUNT(p.ProductID) AS TotalProducts

FROM Categories c

JOIN Has h ON c.CategoryID = h.CategoryID

JOIN Products p ON h.ProductID = p.ProductID

GROUP BY c.CategoryName;

Kết quả:



-- Truy vấn 3:Tìm những danh mục có nhiều sản phẩm nhất

SELECT TOP 1 WITH TIES c.CategoryName, COUNT(p.ProductID) AS TotalProducts

FROM Categories c

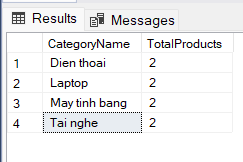
JOIN Has h ON c.CategoryID = h.CategoryID

JOIN Products p ON h.ProductID = p.ProductID

GROUP BY c.CategoryName

ORDER BY TotalProducts DESC

Kết quả:



-- Truy vấn 4:Tìm người dùng chi tiêu nhiều nhất trong mỗi địa chỉ

SELECT ua.Addres, u.Name, SUM(o.TotalAmount) AS TotalSpent

FROM Users u

JOIN Orders o ON u.UserID = o.UserID

JOIN User\_Address ua ON u.UserID = ua.UserID

GROUP BY ua.Addres, u.UserID, u.Name

HAVING SUM(o.TotalAmount) = (

SELECT TOP 1 SUM(o2.TotalAmount)

FROM Orders o2

JOIN User\_Address ua2 ON o2.UserID = ua2.UserID

WHERE ua2.Addres = ua.Addres

GROUP BY o2.UserID

);

Kết quả:



-- Truy vấn 5:Liệt kê các đơn hàng cùng thông tin người mua, danh mục sản phẩm, ngày đặt

SELECT o.OrderID, u.Name, o.OrderDate, c.CategoryName

FROM Orders o

JOIN OrderDetails od ON o.OrderID = od.OrderID

JOIN Has h ON od.ProductID = h.ProductID

JOIN Categories c ON h.CategoryID = c.CategoryID

JOIN Users u ON o.UserID = u.UserID

ORDER BY o.OrderDate;

Kết quả:



-- Truy vấn 6:Tìm người dùng sống ở Hà Nội và đã đặt hàng

SELECT DISTINCT u.Name, u.Email, ua.Addres

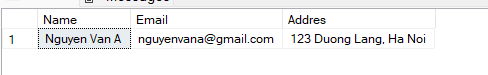
FROM Users u

JOIN User\_Address ua ON u.UserID = ua.UserID

JOIN Orders o ON u.UserID = o.UserID

WHERE ua.Addres LIKE '%Ha Noi';

Kết quả:



-- Truy vấn 7:Lấy thông tin người dùng và sản phẩm họ đã mua có giá trên 10 triệu

SELECT u.Name, u.Email, p.ProductName, p.Price

FROM Users u

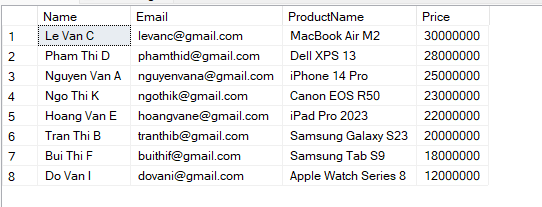
JOIN Orders o ON u.UserID = o.UserID

JOIN OrderDetails od ON o.OrderID = od.OrderID

JOIN Products p ON od.ProductID = p.ProductID

WHERE p.Price > 10000000

ORDER BY p.Price DESC;



-- Truy vấn 8:Sửa thông tin người dùng

UPDATE User\_Address

SET Addres = '100 Nguyen Trai, TP.HCM'

WHERE UserID = 2;

-- Truy vấn 9:Xóa thông tin người dùng

DELETE FROM User\_Address WHERE UserID = 10;

DELETE FROM Orders WHERE UserID = 10;

DELETE FROM Users WHERE UserID = 10;

-- Truy vấn 10:Tìm các sản phẩm có tổng giá trị bán ra lớn hơn 25 triệu

SELECT p.ProductName, p.Price, SUM(od.Quantity \* od.Price) AS TotalSalesValue

FROM Products p

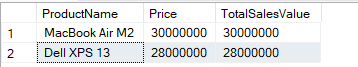
JOIN OrderDetails od ON p.ProductID = od.ProductID

GROUP BY p.ProductID, p.ProductName, p.Price

HAVING SUM(od.Quantity \* od.Price) > 25000000

ORDER BY TotalSalesValue DESC;

Kết quả:



**Tối ưu hóa câu truy vấn của SQL Server:** SQL Server sử dụng một thành phần gọi là Query Optimizer (Trình tối ưu hóa truy vấn) để phân tích và tạo ra kế hoạch thực thi (execution plan) hiệu quả nhất cho mỗi truy vấn. Quá trình tối ưu hóa này dựa trên nhiều yếu tố như cấu trúc dữ liệu, thống kê, và tài nguyên hệ thống

+ **Query Optimizer và Execution Plan**

Query Optimizer là một phần của công cụ cơ sở dữ liệu (Database Engine). Khi bạn gửi một truy vấn SQL, nó sẽ:

Phân tích cú pháp (Parsing): Kiểm tra cú pháp của truy vấn.

Chuyển thành cây truy vấn (Query Tree): Biểu diễn truy vấn dưới dạng logic nội bộ.

Tối ưu hóa: Tạo ra nhiều kế hoạch thực thi khả thi, sau đó chọn kế hoạch có chi phí thấp nhất (dựa trên CPU, I/O, bộ nhớ).

Tạo Execution Plan: Kế hoạch thực thi cuối cùng được lưu vào bộ nhớ cache để tái sử dụng nếu truy vấn lặp lại.

Execution Plan bao gồm các bước như cách truy cập bảng (scan hay seek), thứ tự JOIN, và cách tính toán (ví dụ: GROUP BY).

+ **Sử dụng thống kê (Statistics)**

SQL Server duy trì thống kê về dữ liệu trong bảng và chỉ mục (index), ví dụ: phân bố giá trị trong cột, số lượng dòng, hay độ duy nhất của dữ liệu.

Query Optimizer dùng thông tin này để:

-Ước tính số dòng trả về (cardinality).

-Quyết định chọn Index Seek (tìm kiếm nhanh) hay Table Scan (quét toàn bảng).

Ví dụ: Nếu truy vấn WHERE Price > 20000000 trên bảng Products, SQL Server sẽ kiểm tra thống kê để xem có bao nhiêu dòng thỏa mãn điều kiện và liệu có nên dùng index trên cột Price không.

+ **Các chiến lược tối ưu hóa**

Chọn phương pháp truy cập dữ liệu:

Table Scan: Quét toàn bảng nếu không có index phù hợp hoặc bảng nhỏ.

Index Seek: Sử dụng index để truy cập trực tiếp vào dữ liệu cần thiết.

Index Scan: Quét toàn bộ index nếu điều kiện lọc không đủ chọn lọc.

Tối ưu hóa JOIN:

Nested Loop Join: Tốt cho bảng nhỏ hoặc khi một bảng có ít dòng.

Merge Join: Hiệu quả khi cả hai bảng đã được sắp xếp.

Hash Join: Dùng bảng băm tạm thời, phù hợp khi bảng lớn và không có index tốt.

SQL Server tự động chọn loại JOIN dựa trên kích thước bảng và điều kiện lọc.

Song song hóa (Parallelism):

Với hệ thống nhiều CPU, SQL Server có thể chia nhỏ truy vấn thành các tác vụ chạy song song, sau đó tổng hợp kết quả.

Ví dụ: Thay vì lọc sau khi JOIN, SQL Server có thể đẩy điều kiện WHERE xuống trước để giảm số dòng cần xử lý.

+ **Sử dụng Index**

Index (chỉ mục) là yếu tố quan trọng để tăng tốc truy vấn:

Clustered Index: Sắp xếp dữ liệu vật lý theo cột (mỗi bảng chỉ có một).

Non-Clustered Index: Lưu riêng một cấu trúc chỉ mục với con trỏ đến dữ liệu.

Query Optimizer kiểm tra các index có sẵn và quyết định sử dụng nếu nó giảm chi phí I/O.

+ **Bộ nhớ cache (Plan Cache)**

Sau khi tạo execution plan, SQL Server lưu nó trong Plan Cache. Nếu truy vấn tương tự được gửi lại, nó sẽ tái sử dụng plan này thay vì tối ưu hóa lại, tiết kiệm thời gian.

+ **Cập nhật thống kê và tái tối ưu**

Nếu dữ liệu thay đổi nhiều (thêm, xóa, sửa), thống kê có thể lỗi thời. SQL Server tự động hoặc thủ công (qua lệnh UPDATE STATISTICS) cập nhật thống kê để đảm bảo tối ưu hóa chính xác.