**Kiểu dữ liệu chuỗi kí tự trong SQL Server**

Dưới đây là các kiểu dữ liệu thuộc nhóm chuỗi kí tự trong SQL Server (Transact-SQL).

|  |  |  |
| --- | --- | --- |
| Cú pháp kiểu dữ liệu | Kích thước tối đa | Giải thích |
| CHAR(kich\_thuoc) | Tối đa 8000 kí tự. | kich\_thuoc là số kí tự lưu trữ.  Độ dài cố định.  Thêm dấu cách về bên phải để bù phần trống cho đủ số kí tự.  Không chứa kí tự Unicode. |
| VARCHAR(kich\_thuoc) hoặc VARCHAR(toi\_da) | Tối đa 8000 kí tự hoặc theo số tối đa. | kich\_thuoc là số kí tự lưu trữ.  Độ dài tùy biến.  Nếu chỉ định là toi\_da thì tối đa là 2GB.  Không chứa kí tự Unicode. |
| TEXT | Tối đa 2GB. | Độ dài tùy biến.  Không chứa kí tự Unicode. |
| NCHAR(kich\_thuoc) | Tối đa 4000 kí tự. | Độ dài cố định.  Kí tự Unicode. |
| NVARCHAR(kich\_thuoc) hoặc NVARCHAR(toi\_da) | Tối đa 4000 kí tự hoặc theo số tối đa. | kich\_thuoc là số kí tự lưu trữ.  Độ dài tùy biến.  Nếu số toi\_da được chi định thì số kí tự tối đa là 2GB.  Kí tự Unicode. |
| NTEXT | Tối đa 1.073.741.823 byte. | Độ dài tùy biến.  Kí tự Unicode. |
| BINARY(kich\_thuoc) | Tối đa 8000 kí tự. | kich\_thuoc là số kí tự lưu trữ.  Độ dài cố định.  Thêm dấu cách để bù phần trống cho đủ số kí tự.  Dữ liệu nhị phân. |
| VARBINARY(kich\_thuoc) hoặc VARBINARY(toi\_da) | Tối đa 8000 kí tự hoặc theo số tối đa. | kich\_thuoc là số kí tự lưu trữ.  Độ dài tùy biến.  Nếu chỉ định là toi\_da thì tối đa là 2GB.  Dữ liệu nhị phân. |
| IMAGE | kích thước tối đa là 2GB. | Độ dài tùy biến.  Dữ liệu nhị phân. |

**[Kiểu dữ liệu trong SQL](https://quantrimang.com/hoc/kieu-du-lieu-trong-sql-142478" \o "Kiểu dữ liệu trong SQL)**

Kiểu dữ liệu dạng số trong SQL Server

Dưới đây là danh sách các kiểu dữ liệu thuộc dạng số trong SQL Server (Transact-SQL).

|  |  |  |
| --- | --- | --- |
| Cú pháp kiểu dữ liệu | Kích thước tối đa | Giải thích |
| BIT | số nguyên 0, 1 hoặc NULL |  |
| TINYINT | từ 0 đến 255 |  |
| SMALLINT | từ -32768 đến 32767 |  |
| INT | -2,147,483,648 đến 2,147,483,647 |  |
| BIGINT | từ -9,223,372,036,854,775,808 đến 9,223,372,036,854,775,807 |  |
| DECIMAL(m,d) | m mặc định là 18 nếu không được chỉ định cụ thể.  d mặc định là 0 nếu không được chỉ định cụ thể. | m là tổng số lượng các số còn d là số lượng các số nằm sau dấu phẩy. |
| DEC(m,d) | m mặc định là 18 nếu không được chỉ định cụ thể.  d mặc định là 0 nếu không được chỉ định cụ thể. | m là tổng số lượng các số còn d là số lượng các số nằm sau dấu phẩy.  Đồng nghĩa với kiểu dữ liệu DECIMAL. |
| NUMERIC(m,d) | m mặc định là 18 nếu không được chỉ định cụ thể.  d mặc định là 0 nếu không được chỉ định cụ thể. | m là tổng số lượng các số còn d là số lượng các số nằm sau dấu phẩy.  Đồng nghĩa với kiểu dữ liệu DECIMAL. |
| FLOAT(n) | số dấu phẩy động n mặc định là 53 nếu không được chỉ định cụ thể. | n là số lượng của số bit lưu trữ trong một kí hiệu hóa học. |
| REAL | tương đương với FLOAT(24) |  |
| SMALLMONEY | từ - 214,748.3648 đến 214,748.3647 |  |
| MONEY | từ -922,337,203,685,477.5808 đến 922,337,203,685,477.5807 |  |

Kiểu dữ liệu ngày tháng/thời gian trong SQL Server

Dưới đây là danh sách các kiểu dữ liệu ngày tháng/thời gian trong SQL Server (Transact-SQL)

|  |  |  |
| --- | --- | --- |
| Cú pháp kiểu dữ liệu | Kích thước tối đa | Giải thích |
| DATE | giá trị từ '0001-01-01' đến '9999-12-31. | hiển thị dưới dạng ‘YYYY-MM-DD’ |
| DATETIME | Ngày lấy từ '1753-01-01 00:00:00' to '9999-12-31 23:59:59'.  Giờ lấy từ '00:00:00' to '23:59:59:997' | hiển thị dưới dạng ‘YYYY-MM-DD hh:mm:ss[.mmm] |
| DATETIME2(chính xác tới số thập phân của giây) | giá trị lấy từ '0001-01-01' đến '9999-12-31'.  Thời gian lấy từ '00:00:00' đến '23:59:59:9999999'. | hiển thị dưới dạng 'YYYY-MM-DD hh:mm:ss[.số giây thập phân]' |
| SMALLDATETIME | giá trị lấy từ '1900-01-01' đến '2079-06-06'.  Thời gian lấy từ '00:00:00' đến '23:59:59'. | hiển thị dưới dạng 'YYYY-MM-DD hh:mm:ss |
| TIME | giá trị lấy từ '00:00:00.0000000' đến '23:59:59.9999999'.  Ngày lấy từ '0001-01-01' đến '9999-12-31'. | hiển thị dưới dạng 'YYYY-MM-DD hh:mm:ss[.nnnnnnn]' |
| DATETIMEOFFSET (chính xác tới số thập phân của giây) | giá trị thời gian lấy từ '00:00:00' đến '23:59:59:9999999'.  Múi giờ lấy từ -14:00 đến +14:00. | hiển thị dưới dạng YYYY-MM-DD hh:mm:ss[.nnnnnnn]' [{+|-}hh:mm] |

**Các kiểu dữ liệu khác trong SQL Server**

|  |  |
| --- | --- |
| Kiểu dữ liệu | Mô tả |
| sql\_variant | Chứa tới 8,000 byte dữ liệu của các kiểu khác nhau, ngoại trừ text, ntext, và timestamp |
| uniqueidentifier | Lưu trữ một mã định danh duy nhất trên toàn cục (GUID) |
| xml | Lưu trữ dữ liệu định dạng XML. Tối đa 2GB |
| cursor | Lưu trữ một tham chiếu tới con trỏ được dùng cho các hoạt động database |
| table | Lưu trữ một bộ kết quả để xử lý sau |

**Các thuộc tính trong sql server**

|  |  |  |
| --- | --- | --- |
| Thuộc tính | Mô tả | Ví dụ |
| PRIMARY KEY | Thuộc tính này đánh dấu một hoặc một số trường trong một bảng làm khóa chính. Nó đảm bảo tính duy nhất của các giá trị trong trường đó và cung cấp một cách nhanh chóng để tìm kiếm dữ liệu. | CREATE TABLE Customers  (  CustomerID INT PRIMARY KEY,  CustomerName VARCHAR(255)  ) |
| FOREIGN KEY: | Thuộc tính này được sử dụng để xác định mối quan hệ giữa hai bảng. Nó đảm bảo tính nhất quán và thực hiện các ràng buộc giữa các bảng liên quan | CREATE TABLE Orders  (  OrderID INT PRIMARY KEY,  CustomerID INT,  OrderDate DATE,  FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)  ) |
| UNIQUE | Thuộc tính này đảm bảo tính duy nhất của các giá trị trong trường, tương tự như PRIMARY KEY, nhưng không yêu cầu giá trị duy nhất đó là khóa chính | CREATE TABLE Products  (  ProductID INT UNIQUE,  ProductName VARCHAR(255)  ) |
| DEFAULT | Thuộc tính này cho phép bạn xác định giá trị mặc định cho một trường khi một bản ghi mới được thêm vào bảng và giá trị cho trường đó không được cung cấp | CREATE TABLE Employees  (  EmployeeID INT PRIMARY KEY,  EmployeeName VARCHAR(255),  EmploymentDate DATE DEFAULT GETDATE()  ) |
| CHECK | Thuộc tính này cho phép bạn xác định một điều kiện hoặc quy tắc kiểm tra dữ liệu trước khi nó được thêm vào hoặc cập nhật trong trường | CREATE TABLE Students  (  StudentID INT PRIMARY KEY,  StudentName VARCHAR(255),  Age INT CHECK (Age >= 18)  ) |
| NULL/NOT NULL: | Thuộc tính này quy định liệu trường có thể chứa giá trị NULL hay không. NOT NULL yêu cầu trường luôn phải chứa giá trị, trong khi NULL cho phép trường chứa giá trị null. | CREATE TABLE Orders  (  OrderID INT PRIMARY KEY,  OrderDate DATE,  CustomerID INT NOT NULL  ) |
| IDENTITY | Thuộc tính này cho phép bạn tự động tạo ra các giá trị số nguyên duy nhất và tăng dần cho một trường. | CREATE TABLE Products  (  ProductID INT IDENTITY(1,1) PRIMARY KEY,  ProductName VARCHAR(255)  ) |
| COLLATE | Thuộc tính này xác định cách so sánh và sắp xếp dữ liệu trong trường văn bản. Nó có thể được sử dụng để đảm bảo tính duy nhất của giá trị và sắp xếp dữ liệu theo thứ tự mong muốn. | CREATE TABLE Names  (  NameID INT PRIMARY KEY,  FullName NVARCHAR(255) COLLATE Latin1\_General\_CS\_AS  ) |
| CLUSTERED/NONCLUSTERED INDEX: | Thuộc tính này đánh dấu một trường hoặc nhóm trường để tạo chỉ mục, giúp tăng cường hiệu suất truy xuất dữ liệu. | CREATE TABLE Orders  (  OrderID INT PRIMARY KEY CLUSTERED,  CustomerID INT,  OrderDate DATE  )  CREATE NONCLUSTERED INDEX IX\_CustomerID ON Orders(CustomerID) |
| ENCRYPTED | Thuộc tính này cho phép mã hóa dữ liệu trong trường, bảo vệ dữ liệu trước khi lưu trữ nó trong cơ sở dữ liệu. | CREATE TABLE CreditCards  (  CardNumber VARCHAR(16) ENCRYPTED WITH (COLUMN\_ENCRYPTION\_KEY = MyEncryptionKey) NOT NULL  ) |

**Hàm hệ thống trong sql server**

|  |  |  |
| --- | --- | --- |
| SUM() | Tính tổng giá trị các cột số. | SELECT SUM(Gia) AS TongGia FROM SanPham; |
| AVG() | Tính trung bình giá trị các cột số. | SELECT AVG(Diem) AS DiemTrungBinh FROM SinhVien; |
| COUNT() | Đếm số dòng hoặc giá trị không NULL trong một cột. | SELECT COUNT(\*) AS TongSoKhachHang FROM KhachHang; |
| MAX() | Tìm giá trị lớn nhất trong một cột. | SELECT MAX(Luong) AS LuongCaoNhat FROM NhanVien; |
| MIN() | Tìm giá trị nhỏ nhất trong một cột. | SELECT MIN(Tuoi) AS TuoiNhoNhat FROM HocSinh; |
| CONCAT() | Kết hợp các chuỗi thành một chuỗi lớn. | SELECT CONCAT(Ho, ' ', Ten) AS HoTen FROM NhanVien; |
| LEN() | Đếm số ký tự trong một chuỗi. | SELECT LEN(Description) AS DescriptionLength  FROM Products; |
| SUBSTRING() | Trích xuất một phần của chuỗi. | SELECT SUBSTRING(DienThoai, 1, 5) AS KyTuDauTien FROM KhachHang; |
| CHARINDEX() | Tìm vị trí đầu tiên của một chuỗi con trong chuỗi khác. | SELECT CHARINDEX('apple', Description) AS Position  FROM Products; |
| LOWER() và UPPER() | Chuyển đổi chuỗi thành chữ thường hoặc chữ hoa | SELECT LOWER(ProductName) AS LowercaseName, UPPER(ProductName) AS UppercaseName  FROM Products; |
| GETDATE() | Lấy thời gian và ngày hiện tại. | SELECT GETDATE() AS CurrentDateTime; |
| CURRENT\_TIMESTAMP | Tương tự như GETDATE(), lấy thời gian và ngày hiện tại. | SELECT CURRENT\_TIMESTAMP AS CurrentTime; |
| SYSDATETIME() | Lấy thời gian và ngày hiện tại với độ chính xác cao hơn so với GETDATE() | SELECT SYSDATETIME() AS CurrentDateTime; |
| DATEPART() | Trích xuất một phần cụ thể của một ngày hoặc thời gian, chẳng hạn như năm, tháng hoặc ngày. | SELECT DATEPART(YEAR, OrderDate) AS OrderYear, DATEPART(MONTH, OrderDate) AS OrderMonth  FROM Orders; |
| DATEDIFF() | Tính toán khoảng thời gian giữa hai ngày hoặc thời gian | SELECT DATEDIFF(DAY, OrderDate, ShippedDate) AS DaysToShip  FROM Orders; |
| CONVERT() | Chuyển đổi một giá trị từ một kiểu dữ liệu sang một kiểu dữ liệu khác, chẳng hạn như chuyển đổi ngày tháng sang chuỗi | SELECT CONVERT(VARCHAR, OrderDate, 103) AS FormattedDate  FROM Orders; |
| FORMAT() | Định dạng ngày tháng hoặc thời gian thành một chuỗi theo định dạng cụ thể. | SELECT FORMAT(OrderDate, 'dd/MM/yyyy') AS FormattedDate  FROM Orders; |
| COALESCE() | Trả về giá trị đầu tiên không phải NULL trong danh sách các biểu thức. | SELECT COALESCE(FirstName, LastName, 'N/A') AS DisplayName  FROM Customers; |
| NULLIF() | So sánh hai giá trị và trả về NULL nếu chúng bằng nhau, hoặc giá trị đầu tiên nếu chúng khác nhau. | SELECT NULLIF(Salary, 0) AS ValidSalary  FROM Employees; |
| ISNULL() | Thay thế giá trị NULL bằng một giá trị khác nếu nó là NULL. | SELECT ISNULL(PhoneNumber, 'Not available') AS ContactNumber  FROM Contacts; |
| LTRIM() và RTRIM() | Loại bỏ khoảng trắng không cần thiết từ đầu hoặc cuối của chuỗi. | SELECT LTRIM(RTRIM(StreetAddress)) AS CleanAddress  FROM Customers; |
| LEFT() và RIGHT() | Trích xuất một số lượng ký tự từ đầu hoặc cuối của một chuỗi. | SELECT LEFT(PhoneNumber, 3) AS AreaCode, RIGHT(PhoneNumber, 4) AS LastFourDigits  FROM Contacts; |
| PATINDEX() | Tìm vị trí của một biểu thức chính quy trong một chuỗi. | SELECT PATINDEX('%app%', Description) AS PatternPosition  FROM Products; |
| DATEADD() | Thêm một khoảng thời gian vào một ngày hoặc thời gian. | SELECT DATEADD(DAY, 7, OrderDate) AS DeliveryDate  FROM Orders; |
| STUFF() | Thay thế một phần của chuỗi bằng một chuỗi khác | SELECT STUFF(PhoneNumber, 4, 3, 'XXX') AS MaskedPhoneNumber  FROM Customers; |
| ABS() | Trả về giá trị tuyệt đối của một số. | SELECT ABS(-5) AS AbsoluteValue |
| SIGN() | Trả về dấu của một số (-1 nếu âm, 0 nếu bằng 0, 1 nếu dương). | SELECT SIGN(-5) AS SignValue |

Quy tắc chuyển đổi ERD sang mô hình dữ liệu quan hệ

1. Các Entity(Thực thể)

Mỗi entity sẽ được chuyển đổi  thành 1 bảng. Trong đó:

Các thuộc tính của ERD sẽ trở thành các trường của bảng.

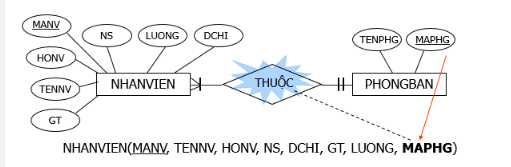
Thuộc tính khoá chính của ERD sẽ trở thành khoá chính của bảng

2. Kiểm tra mỗi quan hệ giữa các thực thể

**a. Mối quan hệ 1 - n:**

Đối với mối quan hệ 1 - nhiều chúng ta sẽ bổ sung khóa chính của bảng 1 làm khóa ngoại của bảng nhiều.

Ví dụ thực thể PhongBan và thực thể NhanVien có mối quan hệ 1 - nhiều (1 phòng ban có nhiều nhân viên) thì khi đó bảng NhanVien sẽ có thêm một khóa ngoại là MaPHG (đây là khóa chính Bảng phòng ban)



Từ hình trên ta sẽ thu được 2 bảng:

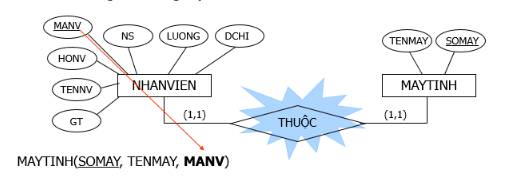
NHANVIEN(MaNV, HoNV, TenNV, GT, NS, Luong, DiaChi, MaPHG)

PHONGBAN(MaPHG, TenPHG)

**b. Mối quan hệ 1 - 1:**

Đối với mối quan hệ 1 - 1 chúng ta sẽ bổ sung khóa chính của bên này làm khóa ngoại của bảng bên kia

Ví dụ thực thể NhanVien có mối quan hệ 1 - 1 với thực thể MayTinh (Mỗi nhân viên chỉ có 1 máy tính) thì chúng ta sẽ bổ sung thêm MaNV làm khóa ngoại của bảng MayTinh



Từ hình trên ta sẽ thu được kết quả là 2 bảng:

NHANVIEN(MaNV, HoNV, TenNV, GT, NS, Luong, DiaChi)

MAYTINH(SoMay, TenMay, MaNV)

**c. Mối quan hệ n - m(nhiều - nhiều):**

Đối với mối quan n - m để chuyển đổi từ ERD sang mô hình dữ liệu quan hệ sẽ phải làm các bước như sau:

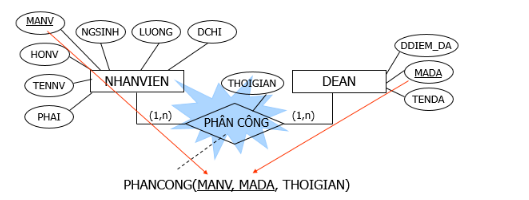
Bước 1: Tạo ra bảng trung gian được sinh ra từ mối quan hệ n - m với chứa khóa chính của 2 bảng chuyển hóa thành khóa ngoại của bảng trung gian cùng với các thuộc tính của mối quan hệ

Bước 2: Xác định khóa trung gian như sau:

Nếu cặp khóa ngoại đủ điều kiện để đảm bảo thành khóa chính thì sẽ chọn cặp khóa ngoại thành khóa chính

Nếu cặp khóa ngoại không đủ điều kiện để đảm bảo tính duy nhất và not null thì phải tạo thêm một trường mới là mã của bảng trung gian làm khóa chính

Ví dụ thực thể NhanVien có mối quan hệ n - m với thực thể DuAn (Một nhân viên có thể làm nhiều dự án và một dự án có thể do nhiều nhân viên thực hiện) thì chúng ta sẽ làm như sau:



Từ hình trên ta thu được các bảng như sau:

NHANVIEN(MaNV, HoNV, TenNV, GT, NS, Luong, DiaChi)

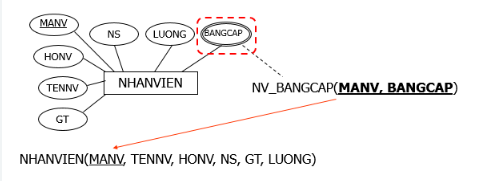
DEAN(MaDA, TenDA, DDiem\_DA)

PHANCONG(MaNV, MaDA, ThoiGian)

**d. Thuộc tính đa trị**

Đối với các thuộc tính đa trị của một thực thể chúng ta sẽ chuyển đổi thành một quan hệ mới (bảng mới) có khóa chính bao gồm 2 thuộc tính là thuộc tính đa trị và thuộc tính khóa của thực thể đồng thời loại bỏ thuộc tính đó khỏi thực thể cũ.

Ví dụ: Thực thể NhanVien có thuộc tính đa trị là bằng cấp khi đó chúng ta chuyển đổi sang mô hình dữ liệu quan hệ sẽ sinh thêm một bảng có tên là NV\_BangCap với khóa chính là MaNV và BangCap và đồng thời xóa bỏ trường BangCap khỏi bảng NhanVien



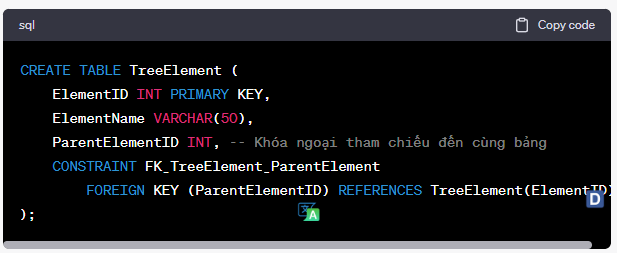
Từ hình trên chúng ta thu được 2 bảng:

NHANVIEN(MaNV, HoNV, TenNV, GT, NS, Luong, DiaChi)

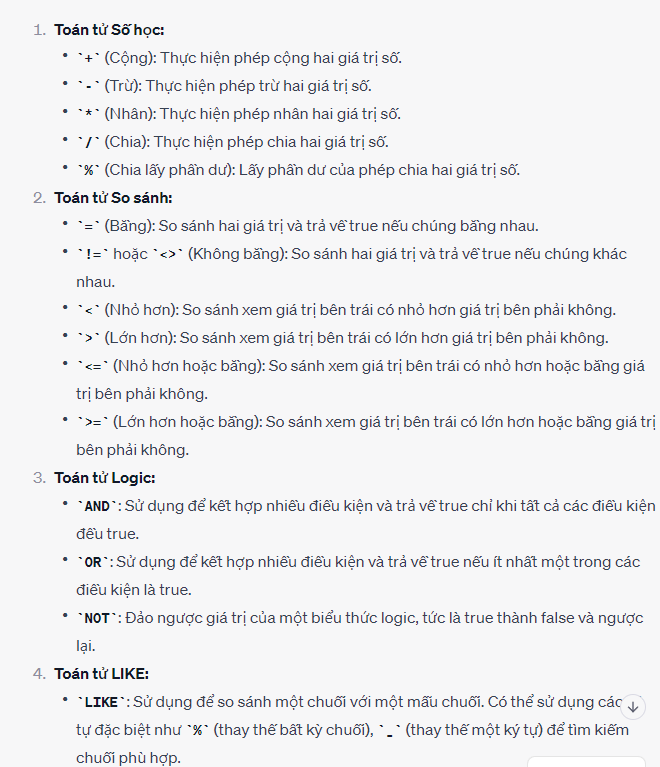
NV\_BANGCAP(MaNV, BangCap)

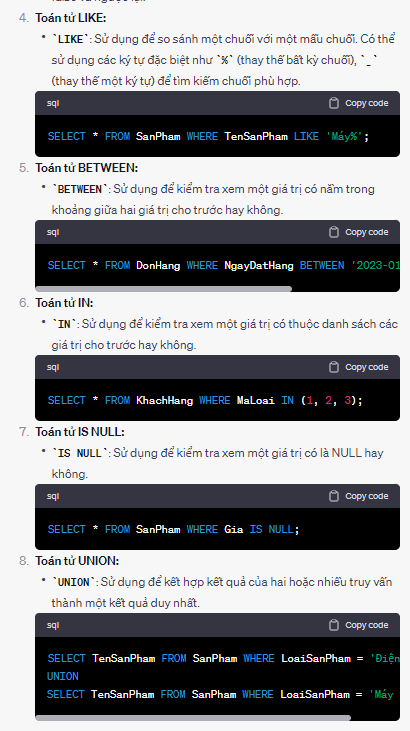
**mối quan hệ đệ quy**

Mối quan hệ đệ quy (Recursive Relationship): Mối quan hệ đệ quy xảy ra khi một thực thể trong bảng có mối quan hệ với chính nó thông qua một hoặc nhiều thuộc tính của nó. Ví dụ, biểu diễn cây phân cấp, danh mục sản phẩm, hoặc cây gia phả.



**Các loại toán tử**





**Các loại join trong sql server**

|  |  |  |
| --- | --- | --- |
| INNER JOIN: | INNER JOIN (hoặc chỉ JOIN) được sử dụng để kết hợp các dòng dữ liệu từ hai bảng dựa trên một điều kiện kết nối. Chỉ các hàng có giá trị kết nối trùng khớp trong cả hai bảng sẽ được trả về. | SELECT Orders.OrderID, Customers.CustomerName  FROM Orders  INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID; |
| LEFT JOIN (hoặc LEFT OUTER JOIN): | LEFT JOIN kết hợp tất cả các hàng từ bảng bên trái (bảng trái) và các hàng từ bảng bên phải (bảng phải) dựa trên điều kiện kết nối. Nếu không có giá trị kết nối trong bảng phải, các cột từ bảng phải sẽ trả về giá trị NULL. | SELECT Customers.CustomerName, Orders.OrderID  FROM Customers  LEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerID; |
| RIGHT JOIN (hoặc RIGHT OUTER JOIN): | RIGHT JOIN tương tự như LEFT JOIN, nhưng nó kết hợp tất cả các hàng từ bảng bên phải và các hàng từ bảng bên trái. Nếu không có giá trị kết nối trong bảng trái, các cột từ bảng trái s | SELECT Customers.CustomerName, Orders.OrderID  FROM Customers  RIGHT JOIN Orders ON Customers.CustomerID = Orders.CustomerID; |
| FULL JOIN (hoặc FULL OUTER JOIN): | FULL JOIN kết hợp tất cả các hàng từ cả bảng bên trái và bảng bên phải. Nó trả về tất cả các hàng từ cả hai bảng, và nếu không có giá trị kết nối trong một bảng, các cột từ bảng đó sẽ trả về giá trị NULL. | SELECT Customers.CustomerName, Orders.OrderID  FROM Customers  FULL JOIN Orders ON Customers.CustomerID = Orders.CustomerID; |
| SELF JOIN: | SELF JOIN là một phép nối mà bạn kết hợp một bảng với chính nó. Điều này thường được sử dụng khi bạn muốn so sánh dữ liệu trong cùng một bảng và tạo mối quan hệ giữa các hàng trong bảng đó. | SELECT e1.EmployeeName, e2.EmployeeName AS Supervisor  FROM Employees e1  LEFT JOIN Employees e2 ON e1.SupervisorID = e2.EmployeeID; |
| CROSS JOIN: | CROSS JOIN tạo một kết hợp (cross product) của tất cả các hàng từ bảng bên trái với tất cả các hàng từ bảng bên phải. Nó tạo ra một số lượng hàng bằng tích của số hàng trong hai bảng. | SELECT Customers.CustomerName, Products.ProductName  FROM Customers  CROSS JOIN Products; |

Các câu lệnh thay đổi

|  |  |
| --- | --- |
| câu lệnh thêm cột vào một bảng nào đó | ALTER TABLE table\_name  ADD column\_name data\_type; |
| thay đổi kiểu dữ liệu | ALTER TABLE table\_name  ALTER COLUMN column\_name new\_data\_type; |
| xoá một hàng nào đó | DELETE FROM SanPham  WHERE ID = 5; |
| sửa dữ liệu một hàng nào đó | UPDATE table\_name  SET column1 = value1, column2 = value2, ...  WHERE condition; |

**Stored Procedure**

Stored Procedure (Thủ tục lưu trữ) trong SQL Server là một tập hợp các câu lệnh SQL được lưu trữ và có thể được gọi và thực thi bất cứ khi nào bạn cần. Stored Procedure thường được sử dụng để thực hiện các tác vụ xử lý dữ liệu phức tạp, truy vấn, cập nhật và thao tác với cơ sở dữ liệu.

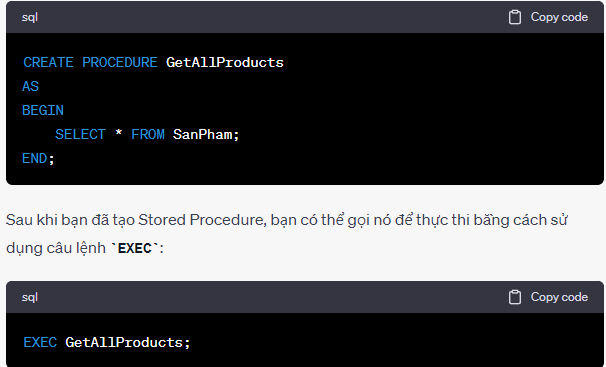
CREATE PROCEDURE procedure\_name

AS

BEGIN

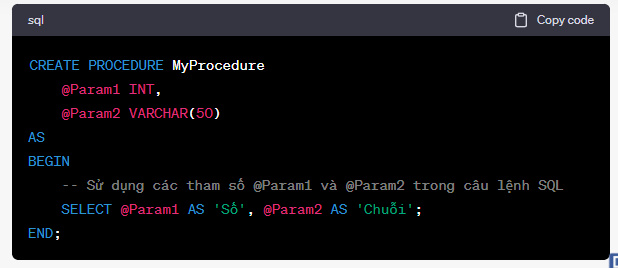
-- Các câu lệnh SQL và xử lý tại đây

END;



**Stored Procedure có tham số truyền vào**

Stored Procedure có thể có hoặc không có tham số truyền vào, tùy thuộc vào yêu cầu của bạn. Một Stored Procedure có thể được thiết kế để nhận tham số đầu vào để thực hiện các hành động dựa trên giá trị đó, hoặc nó có thể không nhận bất kỳ tham số nào.



**view trong sql server**

"view" (chế độ xem) là một đối tượng cơ sở dữ liệu ảo, được tạo dựa trên một hoặc nhiều bảng hoặc view khác. Một view là một câu truy vấn đã được đặt tên và được lưu trữ trong cơ sở dữ liệu, cho phép bạn truy cập và truy vấn dữ liệu một cách thuận tiện và bảo mật. View thường được sử dụng để giới hạn quyền truy cập dữ liệu và để ẩn thông tin nhạy cảm.

CREATE VIEW view\_name

AS

SELECT column1, column2, ...

FROM table\_name

WHERE condition;



**Trigger trong sql server**

"trigger" (hoặc "bắn lệnh") là một đối tượng cơ sở dữ liệu được sử dụng để tự động thực thi một loạt các câu lệnh SQL khi xảy ra một sự kiện nào đó trong cơ sở dữ liệu. Trigger thường được sử dụng để theo dõi và phản ứng tự động khi dữ liệu trong bảng bị thay đổi.

CREATE TRIGGER trigger\_name

ON table\_name

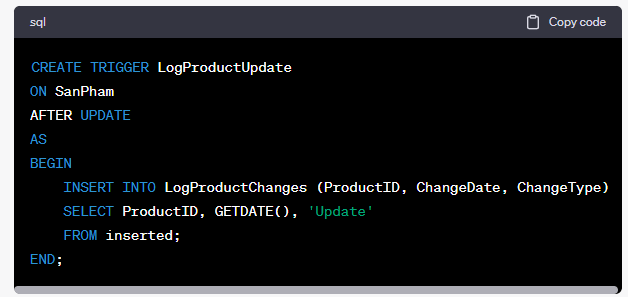
AFTER|INSTEAD OF {INSERT, UPDATE, DELETE}

AS

BEGIN

-- Các câu lệnh SQL và xử lý tại đây

END;



**function (hàm) trong sql server**

function (hàm) là một đối tượng cơ sở dữ liệu được sử dụng để thực hiện các tính toán hoặc xử lý dữ liệu và trả về một giá trị. Có hai loại chính của function trong SQL Server:

User-Defined Function (UDF): Đây là các hàm được tạo ra bởi người dùng, và chúng có thể sử dụng trong các câu lệnh SQL tương tự như các hàm tích hợp (built-in functions). UDF có thể trả về một giá trị hoặc một bảng dữ liệu (table-valued function).

Scalar Function (UDF trả về giá trị): Scalar function là một loại UDF trả về một giá trị duy nhất, chẳng hạn như một số, một chuỗi, hoặc một ngày tháng.

Table-Valued Function (UDF trả về bảng): Table-valued function là một loại UDF trả về một bảng dữ liệu, cho phép bạn thực hiện truy vấn và lấy dữ liệu từ nó.

