CHƯƠNG 3. SQL Primer: Data Definition Language (DDL)

CÁC ĚỐI TƯỢNG CƠ SỞ DỮ LIỆU

1. **TẠO CƠ SỞ DỮ LIỆU**: Ěể tạo được cơ sở dữ liệu, ta login vào với loginname là sa hay với 1 loginname có quyền tạo cơ sở dữ liệu.

Cú pháp: Create Database <tenCSDL>

Ví dụ1: Create Database QLVT

* Trên dƿa sẽ có 2 files: qlvt\_data.mdf và qlvt\_log.ldf

Ngoài ra, ta có thể chỉ định các tham số cần thiết cho việc tạo cơ sở dữ liệu, chẳng hạn như:

# Create Database QLVT On Primary

( Name = Qlvt1,

Filename = 'c:\thu\data\qlvt1.mdf' , Size = 10MB ,

MaxSize = 100MB, FileGrowth = 10MB)

# Log On

( Name = QlvtlLog,

Filename = 'c:\thu\data\qlvt1\_log.ldf' , Size = 10MB ,

MaxSize = 100MB, FileGrowth = 10MB)

Các giá trị mặc định:

* Size: dựa vào kích thước của file model.mdf
* MaxSize : không giới hạn
* FileGrowth : 10%

Sau khi tạo xong 1 cơ sở dữ liệu, ta có thể:

\* Thêm 1 file vào cơ sở dữ liệu đã được tạo:

# Alter Database QLVT Add File

( Name = Qlvt2,

Filename = 'c:\thu\data\qlvt2.**Ndf**' , Size = 10MB ,

MaxSize = 100MB, FileGrowth = 10MB)

* Xóa 1 file khỏi cơ sở dữ liệu QLVT:

# Alter Database QLVT

**Remove File Qlvt2**

* Hiệu chỉnh lại các tùy chọn đã định:

# Alter Database QLVT

**Modify File**

( Name = Qlvt1, FileGrowth = 5MB)

\* Xóa toàn bộ cơ sở dữ liệu : **Drop Database QLVT**

**Lưu ý**: Ta có thể tạo 1 cơ sở dữ liệu trong SQL Server qua công cụ Enterprise Manager (xem V trong chương 2)

# TẠO TABLE:

Cho cơ sở dữ liệu QLVT, trong đó có các Table sau :

# Table Nhanvien

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Type** | **Constraint** |
| MANV | int | Primary key |
| HO | nvarchar(40) | Not Null |
| TEN | nvarchar(10) | Not Null |
| PHAI | nvarchar(3) | Default : ‘Nam’ |
| DIACHI | nvarchar(50) | Not Null, Default : ‘ ‘ |
| NGAYSINH | Date |  |
| LUONG | Money | >=5000000 và <=20000000  Default : 5000000 |
| GHICHU | nText |  |

* 1. **Table Kho:**

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Type** | **Constraint** |
| MAKHO | nChar(2) | Primary Key |
| TENKHO | nvarchar(30) | Unique, Not Null |
| DIACHI | nvarchar(70) | Not Null |

* 1. **Table Vattu:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field Name** | **Type** | **Constraint** | | |
| MAVT | nChar(4) | Primary Key | | |
| TENVT | nvarchar(30) | Unique, Not Null | | |
| DVT | nvarchar(15) | Default : ‘ ‘; Description vị tính | : | đơn |

* 1. **Table Phatsinh**:

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Type** | **Constraint** |
| PHIEU | nChar(8) | Primary Key |
| NGAY | DateTime | Default : GETDATE() |
| LOAI | nChar(1) | chỉ nhận ‘N’, ‘X’, ‘T’, ‘C’  Default : ‘N’ |
| HOTEN | nvarchar (40) | Description: ho ten khach hang |
| MANV | int | Foreign key |

-**Table CT\_Phatsinh**:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Field Name** | **Type** | **Properties** |  |
| PHIEU | nChar(8) | Foreign key |
| MAVT | nChar(4) | Foreign key |
| SOLUONG | int | >0 |
| DONGIA | money | >0 |
| MAKHO | nChar(2) | Foreign Key |
| LYDO | varchar(30) |  |

Khóa chính : **PHIEU+MAVT**

Cú pháp: Ta xem cú pháp của lệnh Tạo Table qua ví dụ sau: Ví dụ 2: Tạo table Vattu với khóa chính là MAVT.

# USE QLVT

**Create Table Vattu**

( MAVT nChar (4) ,

TENVT nVarChar (30) unique Not Null, DVT nVarChar (15) Default ' '

**Constraint** PK\_Vattu **Primary Key** (MAVT) )

Ví dụ 3: Tạo cấu trúc Table NhanVien

# Create Table QLVT.dbo.Nhanvien

( MANV int **Primary key**,

HO nVarChar (40) Not Null,

TEN nVarChar(10) Not Null,

Phai nVarchar(3) Not Null Default 'Nam' ,

DIACHI nVarChar (50) Default ' ' , NGAYSINH Date ,

LUONG Money Default 800000

Check ( Luong >=800000 And Luong <=6000000) ,

GHICHU nText )

Trong lệnh Create Table, ta có thể :

# Tạo ràng buộc về khóa ngoại và field tự động tính.

Ví dụ 4: Xem ví dụ tạo table CT\_Phatsinh, trong đó có thêm field

TRIGIA = SOLUONG\*DONGIA:

Create Table QLVT.dbo.CT\_PhatSinh

( PHIEU nChar(8) Not Null,

MAVT nChar(4) Not Null,

MAKHO nChar(2) Not Null,

SOLUONG int Not Null Check (SOLUONG > 0 ) , DONGIA float Not Null Check (DONGIA > 0 ) , **TRIGIA As SOLUONG \* DONGIA ,**

LYDO varchar(30)

**Constraint** PK\_CTPS Primary Key (PHIEU, MAVT) ,

# Constraint FK\_CTPS\_PS Foreign Key (PHIEU) References dbo.PHATSINH(PHIEU)

**ON DELETE CASCADE ON UPDATE CASCADE,**

Constraint FK\_CTPS\_VT Foreign Key (MAVT) References dbo.VATTU(MAVT)

ON DELETE CASCADE ON UPDATE CASCADE,

Constraint FK\_CTPS\_KHO Foreign Key (MAKHO) References dbo.KHO(MAKHO)

ON DELETE CASCADE ON UPDATE CASCADE

# )

1. **Tạo Unique Key :**

Ví dụ 5: Cho tên kho là field không được trùng

….

# Constraint UK\_TENKHO UNIQUE (TENKHO)

…

# Kiểm tra dữ liệu ở mức record: Constraint <tênRB> Check (đk)

<đk> trong Constraint Check chỉ hoạt động khi nào ta ghi dữ liệu đang nhập vào

table.

# Bỏ qua việc kiểm tra các ràng buộc trong nhân bản dữ liệu: Constraint <tênRB> Check Not For Replication (đk)

Sau khi tạo xong Table, ta có thể:

# \* Thêm hay xóa 1 ràng buộc:

Ví dụ 6: Xóa ràng buộc FK\_CTPS\_Kho trong table CT\_PhatSinh

# Alter Table QLVT.dbo.CT\_PhatSinh Drop Constraint FK\_CTPS\_Kho

* Thêm ràng buộc về khóa ngoại trên field MAKHO của table CT\_PhatSinh.

# Alter Table QLVT.dbo.CT\_PhatSinh

**Add Constraint FK\_CTPS\_KHO Foreign Key (MAKHO) References dbo.KHO(MAKHO)**

Lưu ý: Ěể kiểm tra tất cả các records đã có trong table phải thỏa 1 ràng buộc nào đó thì ta thêm vào tùy chọn **With Check**

# Ví dụ 7: Alter Table QLVT.dbo.CT\_PhatSinh With check

**Check Constraint FK\_CTPS\_KHO**

* **Thêm, Xóa, thay đổi field trong Table:**
* Thêm 1 field :

Ví dụ 8: Thêm Field Dongia vào Table Vattu

# Alter Table QLVT.dbo.Vattu

**Add Dongia Int Check ( Dongia > 0)**

- Thay đổi kiểu của Field

Ví dụ 9: Cho độ rộng của TENVT là 40 ký tự

Alter Table QLVT.dbo.VATTU

Alter Column TENVT Varchar (40) Not Null

Lưu ý: ta không thể thay đổi trên các field có kiểu Text, Image, Timestamp, các field tự động tính và các field dùng trong việc nhân bản dữ liệu. Ta cǜng không thể thay đổi

trên các field có dùng trong field tự động tính, dùng trong 1 ràng buộc hay trong 1 chỉ mục.

- Xóa 1 field:

Ví dụ 10: Xóa field DONGIA trong table VATTU

# Alter Table QLVT.dbo.VATTU Drop Column Dongia

* **Xóa Table:** xóa table VATTU khỏi cơ sở dữ liệu QLVT.

# Drop Table QLVT.dbo.VATTU

1. **TẠO VIEW:** Trong SQL, 1 view là 1 đối tượng xuất hiện như table ảo để user thực hiện 1 query hay thực hiện trong 1 chương trình ứng dụng. Tuy nhiên, view không chứa dữ liệu, mà chứa mã lệnh xử lý dữ liệu. View được định nghƿa trên 1 hay nhiều table cơ sở (hay trên các view khác), và vì vậy ta có thể xem như view là 1 phương tiện để truy nhập tới các table cơ sở. Ta có thể dùng View thực hiện các việc sau:

Hay:

* Chọn ra 1 tập các records trong 1 table cơ sở
* Tạo ra các cột mới dựa trên các cột đã có trong các table cơ sở
* Kết nối nhiều records từ các table cơ sở thành 1 record trong view
* Nối các record từ nhiều table
* Thay đổi dữ liệu trên table .

Ví dụ 11: Chọn ra các nhân viên Nam trong cơ sở dữ liệu QLVT

# Create View V\_NV\_nam As

Select \* From Nhanvien Where Phai='Nam'

# Create View V\_NV\_nam As

Select \* From **[QLVT].**dbo.Nhanvien Where Phai='Nam'

Sau khi cho lệnh này thực hiện, ta sẽ có 1 View tên V\_NV\_Nam trong đối tượng View. Và từ đây về sau, ta có thể dùng nó như 1 table trong 1 phát biểu SQL khác.

Ví dụ 12: Tĕng lương nam nhân viên tên Thu thêm 1%

# Update V\_NV\_Nam

Set Luong = Luong + Luong \*0.01 Where Ten=’Thu’

Ta lưu ý rằng trong điều kiện của lệnh Update ta không đưa vào điều kiện Phai='Nam' vì điều kiện này ta đã có trong View V\_NV\_Nam, và do đó, ta sẽ kế thừa điều kiện Phai='Nam' trong câu lệnh Update.

Ví dụ 13: Tạo View tên NV\_Nam\_Luong2000000 chứa các nam nhân viên có lương 2000000 đồng

Create View NV\_Nam\_Luong2000000 AS

SELECT \* FROM V\_NV\_NAM WHERE LUONG =2000000

# \* Dữ liệu trong View có thể lấy từ nhiều Table:

Ví dụ 14. Tạo view tên Vattu\_Nhap có các cột sau: Số phiếu, Tên vt, Solg, dongia, tên kho, Họ NV, Tên NV

Create View Vattu\_Nhap As

Select dbo.Phatsinh.Phieu, Tenvt, Soluong, dongia, Ho, Ten From dbo.Phatsinh, dbo.CT\_Phatsinh,

dbo.Vattu, dbo.Nhanvien

Where dbo.Phatsinh.Phieu = dbo.CT\_Phatsinh.Phieu AND dbo.CT\_Phatsinh.MAVT = dbo.Vattu.MAVT AND dbo.Phatsinh.MANV = dbo.Nhanvien.MANV AND Loai = 'N'

***\* Nếu mệnh đề Select trong 1 view có dùng Group by thì view đó có thuộc tính Read Only.***

Ví dụ 15: Tạo View tên V\_TriGia\_Phieu cho biết trị giá của các phiếu xuất đã lập.

Create View V\_TriGia\_Phieu As

Select Phieu, Sum(SOLG \* DONGIA) as ThanhTien From CT\_Phatsinh Where Loai=’X’

Group By Phieu

# \* View chứa Subquery cǜng có thuộc tính ReadOnly

**\* With Check Option**: Nếu trong View có thêm tùy chọn With Check Option thì các lệnh Insert, Update trên view này sẽ không thực hiện trên các Field làm vi phạm điều kiện của View.

Ví dụ 16: Ta có View sau: Create View NV\_nam As

Select \* From [QLVT].dbo.Nhanvien Where Phai='Nam'

# With Check Option

thì câu lệnh sau sẽ không thực hiện:

Update **Nv\_nam**

Set **Phai ='Nu'**

Lưu ý: Tùy chọn này sẽ ảnh hưởng trên tất cả View được tạo từ View chứa nó.

\* **Derived Column** : field mới được xây dựng từ các field đã có .

Ví dụ 17: Tạo View ghép họ và tên nhân viên lại thành 1 field

Create View dbo.Luong\_nv As

Select MANV, HOTEN=HO + ' '+ TEN , LUONG From Nhanvien

**\* Xóa 1 View**: **Drop** View <tên view>

# TẠO INDEX:

Thông thường, khi ta tạo table có khóa chính (Primary Key) hay Unique thì SQL Server đã tự động tạo 1 chỉ mục trên table đó. Nhưng trong thực tế, đôi khi ta muốn tạo 1 chỉ mục trên các field không phải là khóa để tĕng hiệu quả truy xuất; lúc này, ta dùng lệnh Create Index

Cú pháp:

# Create [Clustered] Index <tên Index> On < tên csdl >.dbo.< tên table> (danh sách các field trong index)

**[On <tên file nhóm chứa index>]**

Ví dụ 18: Tạo chỉ mục tên IX\_Tenho\_nv theo thứ tự Ten tĕng dần, trùng tên thì sắp qua họ.

Create Index IX\_Tenho\_nv

On QLVT.dbo.Nhanvien (Ten, Ho)

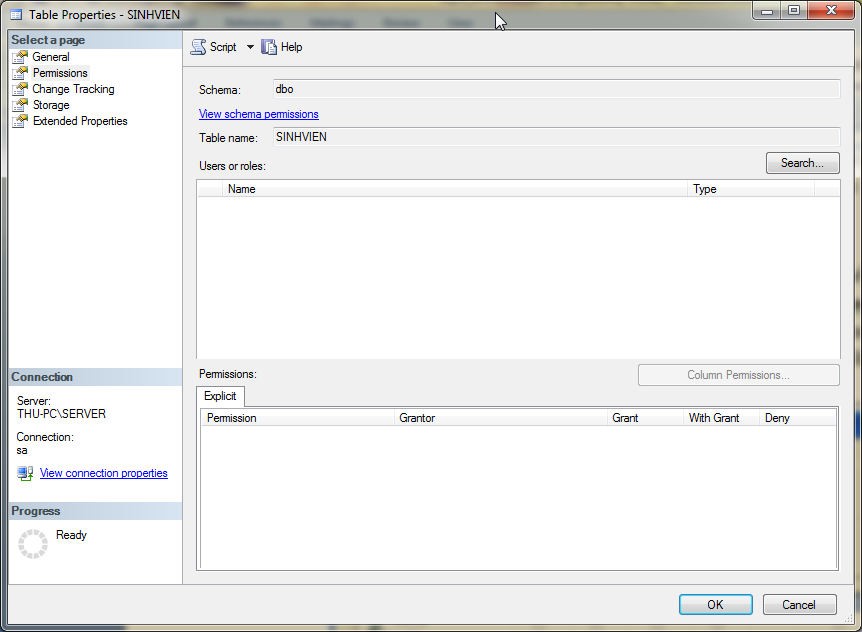
* **Muốn xóa 1 index** : Drop Index Table\_name.Index\_name
* Liệt kê danh sách nhân viên theo thứ tự ten, ho:

Select \* from Nhanvien **with (index**(IX\_Tenho\_nv))

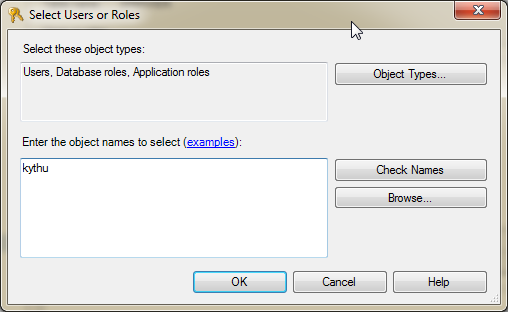
**Lưu y**: *Phải thận trọng khi dùng lệnh Drop vì khi ta xóa 1 đối tượng thì tất cả các đối tượng có liên quan đến đối tượng đó cǜng sẽ mất theo. Ví dụ như khi ta xóa 1 table thì các constraint, index định nghƿa trên table đó cǜng sẽ mất theo.*

# CẤP VÀ XÓA QUYỀN TRÊN TABLE CHO USER:

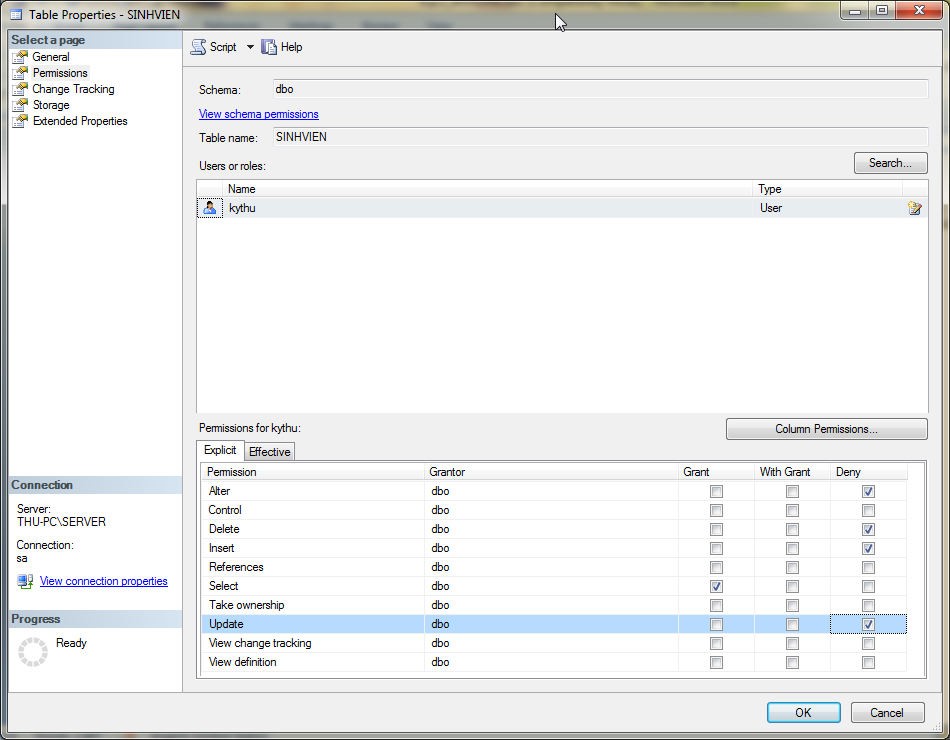
Muốn cấp, thu hồi quyền thực hiện các lệnh trên từng table, ta có thể dùng công cụ Enterprise Manager. Giả sử trong cơ sở dữ liệu QLDSV, ta đã có 1 **user kythu** và 1 nhóm **PDT**. Ta muốn cấp quyền/ thu hồi quyền select , insert, update, delete, exec cho user/nhóm nào trên table Sinhvien thì: right click trên table Sinhvien, chọn Properties / Permissions:



Click Search, để chọn User hoặc Role: click nút Browse để tìm danh sách các User/Role



Sau đó, ta đánh dấu check trên từng quyền muốn cấm (Deny) hoặc cho phép (Grant)



Ngoài ra, ta còn có thể thực hiện việc cấp/ xóa quyền qua lệnh Grant / Revoke

# Cấp quyền:

Grant <danh sách các quyền> On <tên đối tượng>

To <danh sách các users>

- Bảng sau đây cho ta biết từ khóa sử dụng trong danh sách các quyền :

|  |  |
| --- | --- |
| **Từ khóa** | **Lệnh Sql được quyền thực hiện hay bị cấm thực hiện trên đối tượng** |
| Delete | Delete |
| Insert | Insert |
| References | Add 1 Foreign Key |
| Select | Select , Create View |
| Update | Update |
| With Grant | Grant |
| Option | Revoke |
| All | Tất cả các lệnh |

- Tên đối tượng : có thể là tên của table, view, Stored Procedure

Ví dụ 19: Cho user lnkthu chỉ được quyền xem dữ liệu trên các table Nhanvien

# Grant Select On Nhanvien To lnkthu

1. **Thu hồi quyền:**

Revoke <danh sách các quyền>

On <tên đối tượng>

To <danh sách các users>

# \* Column-Level Security:

Lệnh Grant/ Revoke còn cho phép ta cấp quyền trên từng Field của table. Ěiều này sẽ giúp người quản trị cơ sở dữ liệu bảo mật dữ liệu cao hơn.

Ví dụ 20: Cho user lnkthu chỉ được quyền hiệu chỉnh dữ liệu trên các field (HO, TEN, PHAI, DIACHI, NGAYSINH, GHICHU) của table Nhanvien.

# Grant Select

(MANV, HO, TEN, PHAI, DIACHI, NGAYSINH, LUONG, GHICHU)

On Nhanvien To lnkthu

# Grant Update

(HO, TEN, PHAI, DIACHI, NGAYSINH, GHICHU)

On Nhanvien To lnkthu

# \* Cấp/ thu hồi quyền tạo các đối tượng:

Ví dụ 21: Người chủ cơ sở dữ liệu có thể cấp quyền tạo View, Stored Procedure cho 1 user nào đó (lnkthu) qua câu lệnh sau:

# Grant Create View, Create Procedure

**To lnkthu**

Các lệnh SQL sau có thể dùng được trong lệnh Grant/Revoke:

# Create Database, Create Table, Create View, Create Procedure, Create Default, Create Rule, Backup Database, Backup Log.

Lưu ý: Trong mỗi cơ sở dữ liệu đều có 1 tập các table hệ thống, các table này lưu trữ các thông tin mà ta đã mô tả trong quá trình tạo cơ sở dữ liệu (cấp phát vùng nhớ cho cơ sở dữ liệu, các cột của table, các ràng buộc, và chỉ mục…. )

# Bảng sau đây cho ta biết các table hệ thống đó (gọi là catalog)

|  |  |
| --- | --- |
| **Table** | **Nội dung** |
| sysallocations | Cấp phát vùng nhớ cho file cơ sở dữ liệu |
| syscolumns | Mô tả định nghƿa các cột |
| syscomments | Mô tả views, rules, default, trigger, check constraint, và  Stored Procedure. |
| sysconstraint | Anh xạ các constraint tới các đối tượng. |
| sysfiles, sysfiles1 | Thông tin về tên file cơ sở dữ liệu trên dƿa, và vị trí của  chúng |
| sysforeignkeys | Các định nghƿa về khóa ngoại |
| sysmembers | Thông tin về thành viên của nhóm |
| sysusers | Chứa các user và các roles |

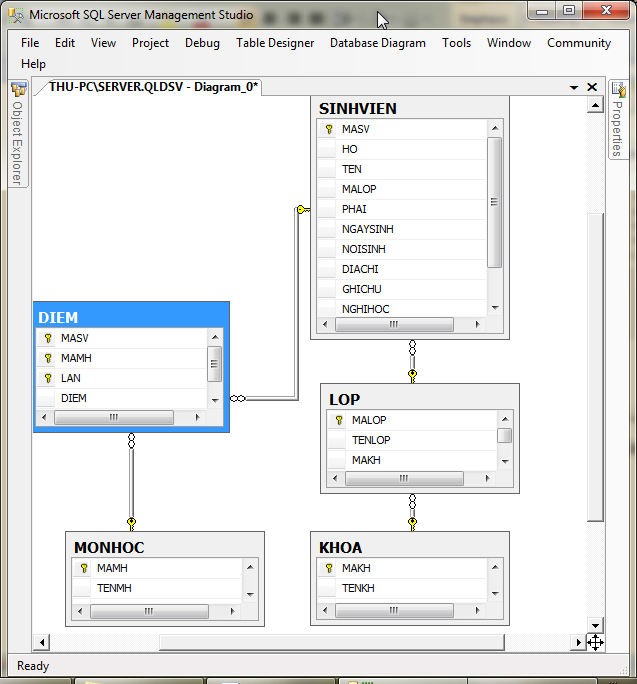
**VI. TẠO DIAGRAMS**: Diagram là đối tượng dùng để tạo, quản lý, và xem các đối tượng cơ sở dữ liệu dưới dạng sơ đồ. Khi ta tạo các khóa ngoại thì thực chất ta đã thiết lập đối tượng diagram bằng lệnh.

Khi tạo các mối quan hệ giữa các table trên diagram, SQL Server sẽ tự động phát sinh các trigger kiểm tra các ràng buộc dữ liệu về khóa ngoại tương ứng, và điều này giúp bảo vệ sự toàn vẹn về dữ liệu.

Cách tạo:

Right click trên đối tượng Diagram/ New Database Diagram, chọn các table cần tạo mối quan hệ, sau đó ta dùng mouse drag tên field là khóa chính trên table A qua field ở table còn lại.

Sau khi hoàn thành, ta có diagram sau:



**Lưu ý**: Trong diagram, ta có thể hiển thị cấu trúc của table theo nhiều dạng khác nhau, và thêm vào các chú giải.