Tuần 1

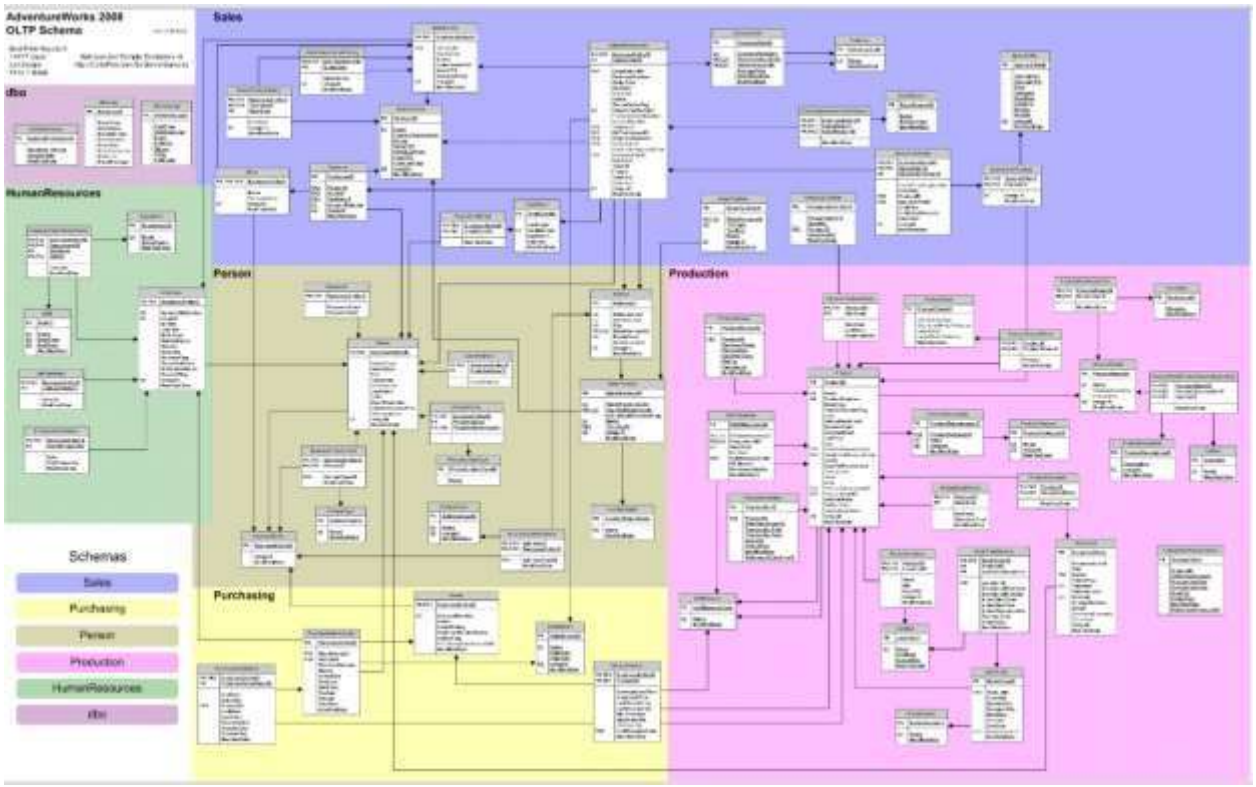
Module 1. Tìm hiểu Cơ sở dữ liệu AdventureWork2008 - Tạo và quản trị cơ sở dữ liệu – Database file – File group

Mục tiêu:

- Tìm hiểu cơ sở dữ liệu AdventureWork2008R2
- Tạo và quản trị cơ sở dữ liệu Database
- Tạo và quản lý các Table (Bảng dữ liệu)
- Tạo lược đồ quan hệ (Relationship Diagram)
- Tạo, sửa, xóa và áp dụng các kiểu dữ liệu trong SQL Server 2008R2
- Biết một số thủ tục trợ giúp về Database và Datatype
- Xây dựng các ràng buộc (Constraint) cho các bảng
- Thêm, sửa, xóa và truy vấn dữ liệu

Mô tả CSDL AdventureWork 2008R2: Gồm 6 lược đồ

- Dbo
- HumanResource
- Person
- Production
- Purchasing
- Sales



AdventureWorks là một cơ sở dữ liệu mẫu được tạo ra để sử dụng trong giảng dạy mỗi phiên bản của **Microsoft SQL Server**.

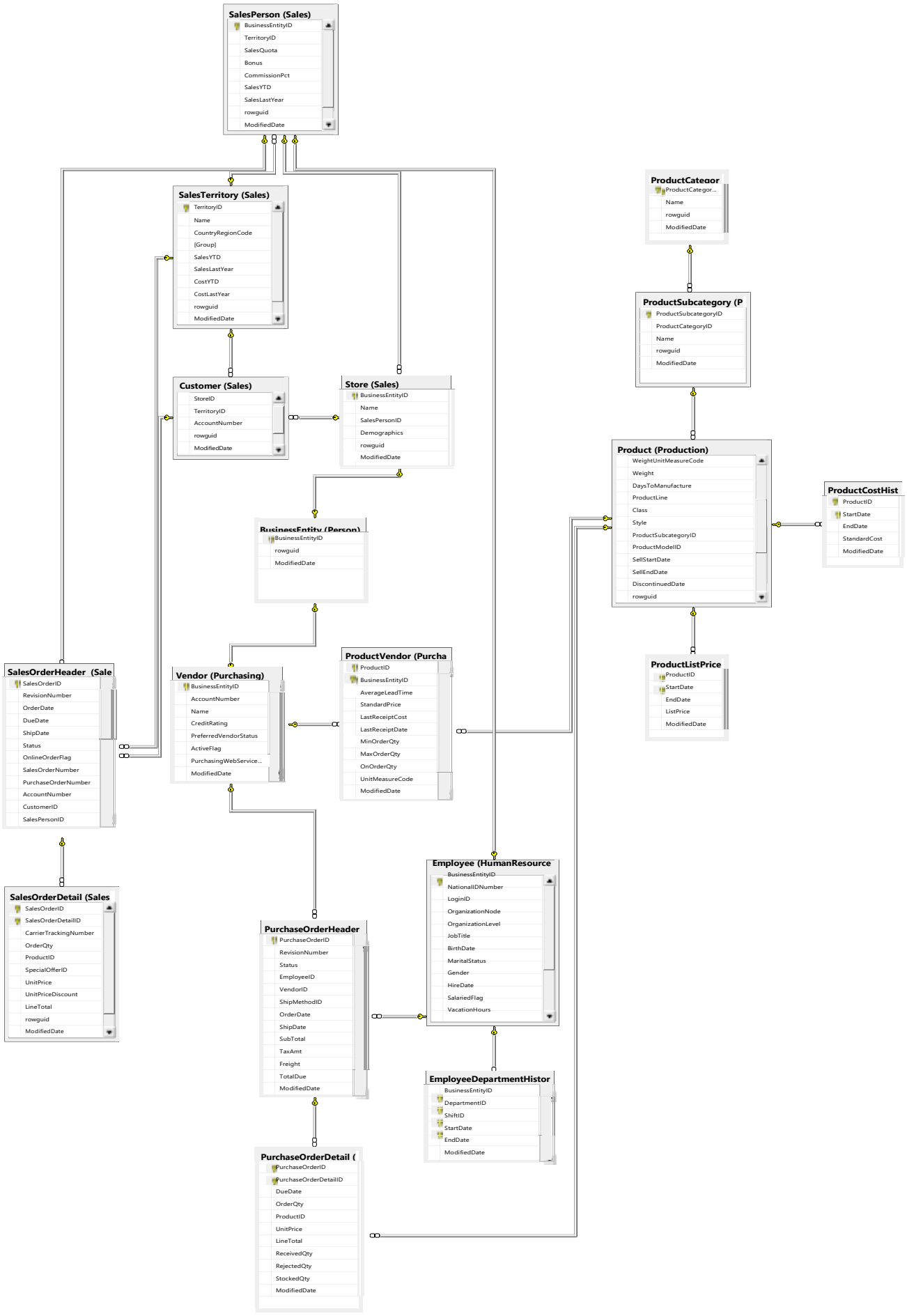
AdventureWorks là một cơ sở dữ liệu quản lý bán hàng của một công ty đa quốc gia chuyên sản xuất và bán các mặt hàng kim loại và xe đạp thể thao đến các thị trường Bắc Mỹ, châu Âu và châu Á. Cơ sở hoạt động của nó nằm ở Bothell, Washington, nhưng nhân viên của họ và các đội bán hàng của các khu vực được bố trí trên khắp cơ sở trên thị trường của họ. Công ty đang tìm kiếm để mở rộng thị phần của mình bằng cách nhắm đến khách hàng tốt nhất của họ, và mở rộng sản phẩm sẵn có của mình thông qua một trang web bên ngoài.

Tổ chức và ý nghĩa của các lược đồ trong CSDL:

Schema	Contains objects related to	Examples
HumanResources	Employees of Adventure Works Cycles.	Employee Table Department Table
Person	Names and addresses of individual customers, vendors, and employees.	Contact Table Address Table StateProvince Table
Production	Products manufactured and sold by Adventure Works Cycles.	BillOfMaterials Table Product Table WorkOrder Table
Purchasing	Vendors from who parts and products are purchased.	PurchaseOrderDetail Table PurchaseOrderHeader Table Vendor Table
Sales	Customers and sales-related data.	Customer Table SalesOrderDetail Table SalesOrderHeader Table

I. Thực hiện các thao tác trên cơ sở dữ liệu AdventureWorks2008R2

1. Tạo một diagram gồm các bảng liên quan đến **Customer** và **Product** như hình ở trang sau:


SalesPerson (Sales)
BusinessEntityID
TerritoryID
SalesQuota
Bonus
CommissionPct
SalesYTD
SalesLastYear
rowguid
ModifiedDate
SalesTerritory (Sales)
TerritoryID
Name
CountryRegionCode
[Group]
SalesYTD
SalesLastYear
CostYTD
CostLastYear
rowguid
ModifiedDate
ProductCategor
ProductCategor...
Name
rowguid
ModifiedDate
ProductSubcategory (P
ProductSubcategoryID
ProductCategoryID
Name
rowguid
ModifiedDate
Customer (Sales)
StoreID
TerritoryID
AccountNumber
rowguid
ModifiedDate
Store (Sales)
BusinessEntityID
Name
SalesPersonID
Demographics
rowguid
ModifiedDate
Product (Production)
WeightUnitMeasureCode
Weight
DaysToManufacture
ProductLine
Class
Style
ProductSubcategoryID
ProductModelID
SellStartDate
SellEndDate
DiscontinuedDate
rowguid
ProductCostHist
ProductID
StartDate
EndDate
StandardCost
ModifiedDate
BusinessEntity (Person)
BusinessEntityID
rowguid
ModifiedDate
SalesOrderHeader (Sale
SalesOrderID
RevisionNumber
OrderDate
DueDate
ShipDate
Status
OnlineOrderFlag
SalesOrderNumber
PurchaseOrderNumber
AccountNumber
CustomerID
SalesPersonID
Vendor (Purchasing)
BusinessEntityID
AccountNumber
Name
CreditRating
PreferredVendorStatus
ActiveFlag
PurchasingWebService...
ModifiedDate
ProductVendor (Purcha
ProductID
BusinessEntityID
AverageLeadTime
StandardPrice
LastReceiptCost
LastReceiptDate
MinOrderQty
MaxOrderQty
OnOrderQty
UnitMeasureCode
ModifiedDate
ProductListPrice
ProductID
StartDate
EndDate
ListPrice
ModifiedDate
SalesOrderDetail (Sales
SalesOrderID
SalesOrderDetailID
CarrierTrackingNumber
OrderQty
ProductID
SpecialOfferID
UnitPrice
UnitPriceDiscount
LineTotal
rowguid
ModifiedDate
Employee (HumanResource
BusinessEntityID
NationalIDNumber
LoginID
OrganizationNode
OrganizationLevel
JobTitle
BirthDate
MaritalStatus
Gender
HireDate
SalariedFlag
VacationHours
PurchaseOrderHeader
PurchaseOrderID
RevisionNumber
Status
EmployeeID
VendorID
ShipMethodID
OrderDate
ShipDate
SubTotal
TaxAmt
Freight
TotalDue
ModifiedDate
EmployeeDepartmentHistor
BusinessEntityID
DepartmentID
ShiftID
StartDate
EndDate
ModifiedDate
PurchaseOrderDetail (
PurchaseOrderID
PurchaseOrderDetailID
DueDate
OrderQty
ProductID
UnitPrice
LineTotal
ReceivedQty
RejectedQty
StockedQty
ModifiedDate

2. Tìm hiểu field liên kết giữa các bảng (Khóa chính và khóa ngoại)

[Production].[Product]	[Person].[BusinessEntity]
[Production].[ProductCostHistory]	[Sales].[Store]
[Production].[ProductListPriceHistory]	[Sales].[Customer]
[Production].[ProductSubcategory]	[Sales].[SalesTerritory]
[Production].[ProductCategory]	[Sales].[SalesPerson]
[Purchasing].[PurchaseOrderDetail]	[Sales].[SalesOrderHeader]
[Purchasing].[PurchaseOrderHeader]	[Sales].[SalesOrderDetail]
[Purchasing].[Vendor]	[HumanResources].[Employee]
[Purchasing].[ProductVendor]	[HumanResources].[EmployeeDepartmentHistory]

Sử dụng T-SQL tạo một cơ sở dữ liệu mới tên **SmallWorks**, với 2 file group tên **SWUserData1** và **SWUserData2,** lưu theo đường dẫn T:\HoTen\TenTapTin.

```
CREATE DATABASE SmallWorks
ON PRIMARY
(
      NAME = 'SmallWorksPrimary',
      FILENAME = 'T:\HoTen\SmallWorks.mdf',
      SIZE = 10MB,
      FILEGROWTH = 20%,
      MAXSIZE = 50MB
),
FILEGROUP SWUserData1
(
      NAME = 'SmallWorksData1',
      FILENAME = 'T:\HoTen\SmallWorksData1.ndf',
      SIZE = 10MB,
      FILEGROWTH = 20%,
      MAXSIZE = 50MB
),
```

```
FILEGROUP SWUserData2
(
      NAME = 'SmallWorksData2',
      FILENAME = 'T:\HoTen\SmallWorksData2.ndf',
      SIZE = 10MB,
      FILEGROWTH = 20%,
      MAXSIZE = 50MB
)
LOG ON
(
      NAME = 'SmallWorks_log',
      FILENAME = 'T:\HoTen\SmallWorks_log.ldf',
      SIZE = 10MB,
      FILEGROWTH = 10%,
      MAXSIZE = 20MB
)
```

3. Dùng **SSMS** để xem kết quả: Click phải trên tên của CSDL vừa tạo
    a. Chọn **filegroups,** quan sát kết quả:
        - Có bao nhiêu filegroup, liệt kê tên các filegroup hiện tại
        - Filegroup mặc định là gì?
    b. Chọn **file**, quan sát có bao nhiêu **database file?**
4. Dùng **T-SQL** tạo thêm một filegroup tên **Test1FG1** trong **SmallWorks,** sau đó add thêm 2 file **filedat1.ndf** và **filedat2.ndf** dung lượng 5MB vào filegroup **Test1FG1**. Dùng SSMS xem kết quả.
5. Dùng T-SQL tạo thêm một một file thứ cấp **filedat3.ndf** dung lượng 3MB trong filegroup **Test1FG1**. Sau đó sửa kích thước tập tin này lên 5MB. Dùng SSMS xem kết quả. Dùng T-SQL xóa file thứ cấp **filedat3.ndf.** Dùng SSMS xem kết quả
6. Xóa filegroup Test1FG1? Bạn có xóa được không? Nếu không giải thích? Muốn xóa được bạn phải làm gì?
7. Xem lại thuộc tính (properties) của CSDL SmallWorks bằng cửa sổ thuộc tính properties và bằng thủ tục hệ thống sp_helpDb, sp_spaceUsed, sp_helpFile. Quan sát và cho biết các trang thể hiện thông tin gì?.
8. Tại cửa sổ **properties** của CSDL **SmallWorks**, chọn thuộc tính **ReadOnly**, sau đó đóng cửa sổ **properties**. Quan sát màu sắc của CSDL. Dùng lệnh T-SQL gỡ bỏ thuộc tính **ReadOnly** và đặt thuộc tính cho phép nhiều người sử dụng CSDL SmallWorks.
9. Trong CSDL **SmallWorks**, tạo 2 bảng mới theo cấu trúc như sau:

```
CREATE TABLE dbo.Person
(
        PersonID int NOT NULL,
        FirstName varchar(50) NOT NULL,
        MiddleName varchar(50) NULL,
        LastName varchar(50) NOT NULL,
        EmailAddress nvarchar(50) NULL
) ON SWUserData1
-----------------------
CREATE TABLE dbo.Product
(
        ProductID int NOT NULL,
        ProductName varchar(75) NOT NULL,
        ProductNumber nvarchar(25) NOT NULL,
        StandardCost money NOT NULL,
        ListPrice money NOT NULL
) ON SWUserData2
```

10. Chèn dữ liệu vào 2 bảng trên, lấy dữ liệu từ bảng **Person** và bảng **Product** trong AdventureWorks2008 (lưu ý: chỉ rõ tên cơ sở dữ liệu và lược đồ), dùng lệnh **Insert…Select...** Dùng lệnh Select * để xem dữ liệu trong 2 bảng **Person** và bảng **Product** trong **SmallWorks.**
11. Dùng SSMS, Detach cơ sở dữ liệu **SmallWorks** ra khỏi phiên làm việc của SQL.
12. Dùng SSMS, Attach cơ sở dữ liệu **SmallWorks** vào SQL

BÀI TẬP VỀ NHÀ:

Yêu cầu sinh viên làm và nộp lại cho giáo viên trước buổi học hôm sau

I. **Dùng T-SQL tạo CSDL T:\HoTen\Sales, các thông số tùy ý, trong CSDL Sales thực hiện các công việc sau:**

1. Tạo các kiểu dữ liệu người dùng sau:

Name	Schema	Data Type	Length	Storage	Allow Nulls
Mota	dbo	nvarchar	40	50	Yes
IDKH	dbo	char	10	10	No
DT	dbo	char	12	50	yes

2. Tạo các bảng theo cấu trúc sau:

SanPham	
Attribute name	Datatype
Masp	char(6)
TenSp	varchar(20)
NgayNhap	Date
DVT	char(10)
SoLuongTon	Int
DonGiaNhap	money

HoaDon	
Attribute name	Datatype
MaHD	Char(10)
NgayLap	Date
NgayGiao	Date
Makh	IDKH
DienGiai	Mota

KhachHang	
MaKH	IDKH
TenKH	Nvarchar(30
Diachi	Nvarchar(40
Dienthoai	DT

ChiTietHD	
Attribute name	Datatype
MaHD	Char(10)
Masp	Char(6)
Soluong	int

3. Trong Table HoaDon, sửa cột DienGiai thành nvarchar(100).
4. Thêm vào bảng SanPham cột TyLeHoaHong float
5. Xóa cột NgayNhap trong bảng SanPham
6. Tạo các ràng buộc khóa chính và khóa ngoại cho các bảng trên
7. Thêm vào bảng **HoaDon** các ràng buộc sau:
   - NgayGiao >= NgayLap
   - MaHD gồm 6 ký tự, 2 ký tự đầu là chữ, các ký tự còn lại là số
   - Giá trị mặc định ban đầu cho cột NgayLap luôn luôn là ngày hiện hành
8. Thêm vào bảng **Sản phẩm** các ràng buộc sau:
   - **SoLuongTon** chỉ nhập từ 0 đến 500
   - **DonGiaNhap** lớn hơn 0
   - Giá trị mặc định cho **NgayNhap** là ngày hiện hành
   - DVT chỉ nhập vào các giá trị 'KG', 'Thùng', 'Hộp', 'Cái'
9. Dùng lệnh T-SQL nhập dữ liệu vào 4 table trên, dữ liệu tùy ý, chú ý các ràng buộc của mỗi Table
10. Xóa 1 hóa đơn bất kỳ trong bảng HoaDon. Có xóa được không? Tại sao? Nếu vẫn muốn xóa thì phải dùng cách nào?
11. Nhập 2 bản ghi mới vào bảng ChiTietHD với MaHD = 'HD999999999' và MaHD='1234567890'. Có nhập được không? Tại sao?
12. Đổi tên CSDL **Sales** thành **BanHang**
13. Tạo thư mục T:\QLBH, chép CSDL BanHang vào thư mục này, bạn có sao chép được không? Tại sao? Muốn sao chép được bạn phải làm gì? Sau khi sao chép, bạn thực hiện Attach CSDL vào lại SQL.

14. Tạo bản BackUp cho CSDL BanHang
15. Xóa CSDL BanHang
16. Phục hồi lại CSDL BanHang.

Module 2. Thao tác dữ liệu

Mục tiêu:

- Ôn lại các cấu trúc câu lệnh select có lệnh gom nhóm, Subquery

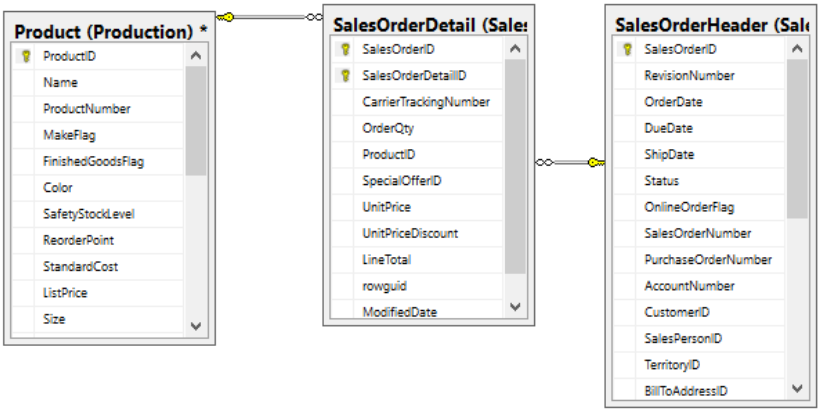
I) Câu lệnh SELECT sử dụng các hàm thống kê với các mệnh đề Group by và Having:

1) Liệt kê danh sách các hóa đơn (SalesOrderID) lập trong tháng 6 năm 2008 có tổng tiền >70000, thông tin gồm SalesOrderID, Orderdate, **SubTotal,** trong đó SubTotal =SUM(OrderQty*UnitPrice).
2) Đếm tổng số khách hàng và tổng tiền của những khách hàng thuộc các quốc gia có mã vùng là US (lấy thông tin từ các bảng *Sales.SalesTerritory, Sales.Customer, Sales.SalesOrderHeader, Sales.SalesOrderDetail*). Thông tin bao gồm TerritoryID, tổng số khách hàng (**CountOfCust**), tổng tiền (**SubTotal**) với SubTotal = SUM(OrderQty*UnitPrice)
3) Tính tổng trị giá của những hóa đơn với *Mã theo dõi giao hàng* (CarrierTrackingNumber) có 3 ký tự đầu là 4BD, thông tin bao gồm SalesOrderID, CarrierTrackingNumber, SubTotal=SUM(OrderQty*UnitPrice)
4) Liệt kê các sản phẩm (Product) có đơn giá (UnitPrice)<25 và số lượng bán trung bình >5, thông tin gồm ProductID, Name, AverageOfQty.
5) Liệt kê các công việc (JobTitle) có tổng số nhân viên >20 người, thông tin gồm JobTitle, CountOfPerson=Count(*)
6) Tính tổng số lượng và tổng trị giá của các sản phẩm do các nhà cung cấp có tên kết thúc bằng 'Bicycles' và tổng trị giá > 800000, thông tin gồm BusinessEntityID, Vendor_Name, ProductID, SumOfQty, SubTotal
   (sử dụng các bảng [Purchasing].[Vendor], [Purchasing].[PurchaseOrderHeader] và [Purchasing].[PurchaseOrderDetail])
7) Liệt kê các sản phẩm có trên 500 đơn đặt hàng trong quí 1 năm 2008 và có tổng trị giá >10000, thông tin gồm ProductID, Product_Name, CountOfOrderID và SubTotal

8) Liệt kê danh sách các khách hàng có trên 25 hóa đơn đặt hàng từ năm 2007 đến 2008, thông tin gồm mã khách (PersonID) , họ tên (FirstName +' '+ LastName as FullName), Số hóa đơn (CountOfOrders).
9) Liệt kê những sản phẩm có tên bắt đầu với 'Bike' và 'Sport' có tổng số lượng bán trong mỗi năm trên 500 sản phẩm, thông tin gồm ProductID, Name, CountOfOrderQty, Year. (*Dữ liệu lấy từ các bảng Sales.SalesOrderHeader, Sales.SalesOrderDetail và Production.Product*)
10) Liệt kê những phòng ban có lương (Rate: lương theo giờ) trung bình >30, thông tin gồm Mã phòng ban (DepartmentID), tên phòng ban (Name), Lương trung bình (AvgofRate). Dữ liệu từ các bảng *[HumanResources].[Department]*, *[HumanResources].[EmployeeDepartmentHistory]*, *[HumanResources].[EmployeePayHistory]*.

II) **Subquery**

1) Liệt kê các sản phẩm gồm các thông tin **Product Names** và **Product ID** có trên 100 đơn đặt hàng trong tháng 7 năm 2008



2) Liệt kê các sản phẩm (ProductID, Name) có số hóa đơn đặt hàng nhiều nhất trong tháng 7/2008
3) Hiển thị thông tin của khách hàng có số đơn đặt hàng nhiều nhất, thông tin gồm: CustomerID, Name, CountOfOrder
4) Liệt kê các sản phẩm (ProductID, Name) thuộc mô hình sản phẩm áo dài tay với tên bắt đầu với "Long-Sleeve Logo Jersey", dùng phép IN và EXISTS, (*sử dụng bảng Production.Product và Production.ProductModel*)

5) Tìm các mô hình sản phẩm (ProductModelID) mà giá niêm yết (list price) tối đa cao hơn giá trung bình của tất cả các mô hình.
6) Liệt kê các sản phẩm gồm các thông tin ProductID, Name, có tổng số lượng đặt hàng > 5000 (dùng IN, EXISTS)
7) Liệt kê những sản phẩm (ProductID, UnitPrice) có đơn giá (UnitPrice) cao nhất trong bảng Sales.SalesOrderDetail
8) Liệt kê các sản phẩm không có đơn đặt hàng nào thông tin gồm ProductID, Nam; dùng 3 cách **Not in**, **Not exists** và **Left join**.
9) Liệt kê các nhân viên không lập hóa đơn từ sau ngày 1/5/2008, thông tin gồm EmployeeID, FirstName, LastName (dữ liệu từ 2 bảng HumanResources.Employees và Sales.SalesOrdersHeader)
10) Liệt kê danh sách các khách hàng (CustomerID, Name) có hóa đơn đặt hàng trong năm 2007 nhưng không có hóa đơn đặt hàng trong năm 2008.

Tuần 3

BÀI TẬP VỀ NHÀ: Hiện thực các ràng buộc toàn vẹn dữ liệu

Yêu cầu sinh viên tự làm và nộp lại cho giáo viên ở buổi thực hành tuần sau

Mục tiêu:

- Thực hiện các ràng buộc toàn vẹn dữ liệu: Primary key,Foreign key, Domain, Check, Default.
- Tìm hiểu **cascading constraint** trong thao tác update và delete

1) Tạo hai bảng mới trong cơ sở dữ liệu AdventureWorks2008 theo cấu trúc sau:

```
create table
MyDepartment (
     DepID smallint not null primary
     key, DepName nvarchar(50),
     GrpName
     nvarchar(50)
)
create table MyEmployee (
     EmpID int not null primary
     key, FrstName nvarchar(50),
     MidName
```

```
        nvarchar(50),
        LstName
        nvarchar(50),
        DepID smallint not null foreign key
        references MyDepartment(DepID)
    )
```

2) Dùng lệnh **insert <TableName1> select <fieldList> from <TableName2>** chèn dữ liệu cho bảng MyDepartment, lấy dữ liệu từ bảng **[HumanResources].[Department].**
3) Tương tự câu 2, chèn 20 dòng dữ liệu cho bảng MyEmployee lấy dữ liệu từ 2 bảng

    `[Person].[Person]` và `[HumanResources].[EmployeeDepartmentHistory]`
4) Dùng lệnh **delete** xóa 1 record trong bảng **MyDepartment** với DepID=1, có thực hiện được không? Vì sao?
5) Thêm một **default constraint** vào field **DepID** trong bảng **MyEmployee**, với giá trị mặc định là **1.**
6) Nhập thêm một record mới trong bảng MyEmployee, theo cú pháp sau: `insert into MyEmployee (EmpID, FrstName, MidName, LstName) values(1, 'Nguyen','Nhat','Nam').` Quan sát giá trị trong field **depID** của record mới thêm.
7) Xóa foreign key constraint trong bảng MyEmployee, thiết lập lại khóa ngoại DepID tham chiếu đến DepID của bảng MyDepartment với thuộc tính **on delete set default**.
8) Xóa một record trong bảng MyDepartment có DepID=7, quan sát kết quả trong hai bảng MyEmployee và MyDepartment
9) Xóa **foreign key** trong bảng **MyEmployee.** Hiệu chỉnh ràng buộc khóa ngoại **DepID** trong bảng **MyEmployee,** thiết lập thuộc tính **on delete cascade** và **on update cascade**
10) Thực hiện xóa một record trong bảng **MyDepartment** với DepID =3, có thực hiện được không?
11) Thêm ràng buộc **check** vào bảng MyDepartment tại field GrpName, chỉ cho phép nhận thêm những Department thuộc group Manufacturing

12) Thêm ràng buộc **check** vào bảng [HumanResources].[Employee], tại cột BirthDate, chỉ cho phép nhập thêm nhân viên mới có tuổi từ 18 đến 60

Module 3. View

Mục tiêu:

- Tạo view, thao tác trên view
- Tìm hiểu các thuộc tính của view

1) Tạo view **dbo.vw_Products** hiển thị danh sách các sản phẩm từ bảng Production.Product và bảng Production.ProductCostHistory. Thông tin bao gồm ProductID, Name, Color, Size, Style, StandardCost, EndDate, StartDate
2) Tạo view List_Product_View chứa danh sách các sản phẩm có trên 500 đơn đặt hàng trong quí 1 năm 2008 và có tổng trị giá >10000, thông tin gồm ProductID, Product_Name, CountOfOrderID và SubTotal.
3) Tạo view dbo.vw_CustomerTotals hiển thị tổng tiền bán được (total sales) từ cột TotalDue của mỗi khách hàng (customer) theo tháng và theo năm. Thông tin gồm CustomerID, YEAR(OrderDate) AS OrderYear, MONTH(OrderDate) AS OrderMonth, SUM(TotalDue).
4) Tạo view trả về tổng số lượng sản phẩm (Total Quantity) bán được của mỗi nhân viên theo từng năm. Thông tin gồm SalesPersonID, OrderYear, sumOfOrderQty
5) Tạo view ListCustomer_view chứa danh sách các khách hàng có trên 25 hóa đơn đặt hàng từ năm 2007 đến 2008, thông tin gồm mã khách (PersonID) , họ tên (FirstName +' '+ LastName as FullName), Số hóa đơn (CountOfOrders).
6) Tạo view ListProduct_view chứa danh sách những sản phẩm có tên bắt đầu với 'Bike' và 'Sport' có tổng số lượng bán trong mỗi năm trên 50 sản phẩm, thông tin gồm ProductID, Name, SumOfOrderQty, Year. (*dữ liệu lấy từ các bảng Sales.SalesOrderHeader, Sales.SalesOrderDetail, và Production.Product*)
7) Tạo view List_department_View chứa danh sách các phòng ban có lương (Rate: lương theo giờ) trung bình >30, thông tin gồm Mã phòng ban (DepartmentID), tên phòng ban (Name), Lương trung bình (AvgOfRate). Dữ liệu từ các bảng *[HumanResources].[Department]*,

[HumanResources].[EmployeeDepartmentHistory], *[HumanResources].[EmployeePayHistory]*.

8) Tạo view **Sales.vw_OrderSummary** với từ khóa **WITH ENCRYPTION** gồm OrderYear (năm của ngày lập), OrderMonth (tháng của ngày lập), OrderTotal (tổng tiền). Sau đó xem thông tin và trợ giúp về mã lệnh của view này
9) Tạo view **Production.vwProducts** với từ khóa **WITH SCHEMABINDING** gồm ProductID, Name, StartDate,EndDate,ListPrice của bảng Product và bảng ProductCostHistory. Xem thông tin của View. Xóa cột ListPrice của bảng Product. Có xóa được không? Vì sao?
10) Tạo view view_Department với từ khóa **WITH CHECK OPTION** chỉ chứa các phòng thuộc nhóm có tên (GroupName) là "Manufacturing" và "Quality Assurance", thông tin gồm: DepartmentID, Name, GroupName.
    a. Chèn thêm một phòng ban mới thuộc nhóm không thuộc hai nhóm "Manufacturing" và "Quality Assurance" thông qua view vừa tạo. Có chèn được không? Giải thích.
    b. Chèn thêm một phòng mới thuộc nhóm "Manufacturing" và một phòng thuộc nhóm "Quality Assurance".
    c. Dùng câu lệnh Select xem kết quả trong bảng Department.

Module 4. Batch, Stored Procedure, Function

Mục tiêu:

Hiểu và biết cách lập trình trong SQL

- Viết các batch

Tạo và thực thi các loại stored procedure và function

- Stored Procedure
  - Tham số input và output
- Function gồm 3 loại
  - Scalar Function
  - Table valued Function
  - Multi statement table valued Function

Tuần 4

I) Batch

1) Viết một batch khai báo biến @tongsoHD chứa tổng số hóa đơn của sản phẩm có ProductID='778'; nếu @tongsoHD>500 thì in ra chuỗi "Sản phẩm 778 có trên 500 đơn hàng", ngược lại thì in ra chuỗi "Sản phẩm 778 có ít đơn đặt hàng"
2) Viết một đoạn Batch với tham số @makh và @n chứa số hóa đơn của khách hàng @makh, tham số @nam chứa năm lập hóa đơn (ví dụ @nam=2008), nếu @n>0 thì in ra chuỗi: "Khách hàng @makh có @n hóa đơn trong năm 2008" ngược lại nếu @n=0 thì in ra chuỗi "Khách hàng @makh không có hóa đơn nào trong năm 2008"
3) Viết một batch tính số tiền giảm cho những hóa đơn (SalesOrderID) có tổng tiền>100000, thông tin gồm [SalesOrderID], SubTotal=SUM([LineTotal]), Discount (tiền giảm), với Discount được tính như sau:
   - Những hóa đơn có SubTotal<100000 thì không giảm,
   - SubTotal từ 100000 đến <120000 thì giảm 5% của SubTotal
   - SubTotal từ 120000 đến <150000 thì giảm 10% của SubTotal
   - SubTotal từ 150000 trở lên thì giảm 15% của SubTotal

*(Gợi ý: Dùng cấu trúc **Case… When …Then …**)*

4) Viết một Batch với 3 tham số: @mancc, @masp, @soluongcc, chứa giá trị của các field [ProductID],[BusinessEntityID],[OnOrderQty], với giá trị truyền cho các biến @mancc, @masp (vd: @mancc=1650, @masp=4), thì chương trình sẽ gán giá trị tương ứng của field [OnOrderQty] cho biến @soluongcc, nếu @soluongcc trả về giá trị là null thì in ra chuỗi "Nhà cung cấp 1650 không cung cấp sản phẩm 4", ngược lại (vd: @soluongcc=5) thì in chuỗi "Nhà cung cấp 1650 cung cấp sản phẩm 4 với số lượng là 5"

(Gợi ý: Dữ liệu lấy từ [Purchasing].[ProductVendor])

5) Viết một batch thực hiện tăng lương giờ (Rate) của nhân viên trong [HumanResources].[EmployeePayHistory] theo điều kiện sau: Khi tổng lương giờ của tất cả nhân viên Sum(Rate)<6000 thì cập nhật tăng lương giờ lên 10%, nếu sau khi cập nhật mà lương giờ cao nhất của nhân viên >150 thì dừng.

```
WHILE (SELECT SUM(rate) FROM
[HumanResources].[EmployeePayHistory])<6000
BEGIN
      UPDATE [HumanResources].[EmployeePayHistory]
      SET rate = rate*1.1
      IF (SELECT MAX(rate)FROM
      [HumanResources].[EmployeePayHistory]) > 150
         BREAK
      ELSE
          CONTINUE
END
```

Tuần 5

II) Stored Procedure:

1) Viết một thủ tục tính tổng tiền thu (TotalDue) của mỗi khách hàng trong một tháng bất kỳ của một năm bất kỳ (tham số tháng và năm) được nhập từ bàn phím, thông tin gồm: CustomerID, SumOfTotalDue =Sum(TotalDue)

2) Viết một thủ tục dùng để xem doanh thu từ đầu năm cho đến ngày hiện tại của một nhân viên bất kỳ, với một tham số đầu vào và một tham số đầu ra. Tham số @SalesPerson nhận giá trị đầu vào theo chỉ định khi gọi thủ tục, tham số

@SalesYTD được sử dụng để chứa giá trị trả về của thủ tục.

3) Viết một thủ tục trả về một danh sách ProductID, ListPrice của các sản phẩm có giá bán không vượt quá một giá trị chỉ định (tham số input @MaxPrice).

4) Viết thủ tục tên **NewBonus** cập nhật lại tiền thưởng (Bonus) cho 1 nhân viên bán hàng (SalesPerson), dựa trên tổng doanh thu của nhân viên đó. Mức thưởng mới bằng mức thưởng hiện tại cộng thêm 1% tổng doanh thu. Thông tin bao gồm

   [SalesPersonID], NewBonus (thưởng mới), SumOfSubTotal. Trong đó:

   SumOfSubTotal =sum(SubTotal)

   NewBonus = Bonus+ sum(SubTotal)*0.01

5) Viết một thủ tục dùng để xem thông tin của nhóm sản phẩm (ProductCategory) có tổng số lượng (OrderQty) đặt hàng cao nhất trong một năm tùy ý (tham số input), thông tin gồm: ProductCategoryID, Name, SumOfQty. Dữ liệu từ bảng

   ProductCategory, ProductSubCategory, Product và SalesOrderDetail.

   (Lưu ý: dùng Sub Query)

6) Tạo thủ tục đặt tên là **TongThu** có tham số vào là mã nhân viên, tham số đầu ra là tổng trị giá các hóa đơn nhân viên đó bán được. Sử dụng lệnh RETURN để trả về trạng thái thành công hay thất bại của thủ tục.

7) Tạo thủ tục hiển thị tên và số tiền mua của cửa hàng mua nhiều hàng nhất theo năm đã cho.

8) Viết thủ tục **Sp_InsertProduct** có tham số dạng input dùng để chèn một mẫu tin vào bảng Production.Product. Yêu cầu: chỉ thêm vào các trường có giá trị not null và các field là khóa ngoại.

9) Viết thủ tục **XoaHD**, dùng để xóa 1 hóa đơn trong bảng Sales.SalesOrderHeader khi biết SalesOrderID. Lưu ý : trước khi xóa mẫu tin trong Sales.SalesOrderHeader thì phải xóa các mẫu tin của hoá đơn đó trong Sales.SalesOrderDetail.

10) Viết thủ tục **Sp_Update_Product** có tham số ProductId dùng để tăng listprice lên 10% nếu sản phẩm này tồn tại, ngược lại hiện thông báo không có sản phẩm này.

III) Function

– **Scalar Function**

1) Viết hàm tên **CountOfEmployees** (dạng scalar function) với tham số @mapb, giá trị truyền vào lấy từ field [DepartmentID], hàm trả về số nhân viên trong phòng ban tương ứng. Áp dụng hàm đã viết vào câu truy vấn liệt kê danh sách các phòng ban với số nhân viên của mỗi phòng ban, thông tin gồm: [DepartmentID], Name, countOfEmp với countOfEmp= **CountOfEmployees(**[DepartmentID]**).**

   (Dữ liệu lấy từ bảng [HumanResources].[EmployeeDepartmentHistory] và [HumanResources].[Department])

2) Viết hàm tên là **InventoryProd** (dạng scalar function) với tham số vào là @ProductID và @LocationID trả về số lượng tồn kho của sản phẩm trong khu vực tương ứng với giá trị của tham số

   (Dữ liệu lấy từ bảng[Production].[ProductInventory])

3) Viết hàm tên **SubTotalOfEmp** (dạng scalar function) trả về tổng doanh thu của một nhân viên trong một tháng tùy ý trong một năm tùy ý, với tham số vào @EmplID, @MonthOrder, @YearOrder

   (Thông tin lấy từ bảng [Sales].[SalesOrderHeader])

Tuần 6

– **Table Valued Functions:**

4) Viết hàm **SumOfOrder** với hai tham số @thang và @nam trả về danh sách các hóa đơn (SalesOrderID) lập trong tháng và năm được truyền vào từ 2 tham số @thang và @nam, có tổng tiền >70000, thông tin gồm SalesOrderID, OrderDate, SubTotal, trong đó SubTotal =sum(OrderQty*UnitPrice).

5) Viết hàm tên **NewBonus** tính lại tiền thưởng (Bonus) cho nhân viên bán hàng (SalesPerson), dựa trên tổng doanh thu của mỗi nhân viên, mức thưởng mới bằng mức thưởng hiện tại tăng thêm 1% tổng doanh thu, thông tin bao gồm [SalesPersonID], NewBonus (thưởng mới), SumOfSubTotal. Trong đó:
   - SumOfSubTotal =sum(SubTotal),
   - NewBonus = Bonus+ sum(SubTotal)*0.01

6) Viết hàm tên **SumOfProduct** với tham số đầu vào là @MaNCC (VendorID),

hàm dùng để tính tổng số lượng (SumOfQty) và tổng trị giá (SumOfSubTotal) của các sản phẩm do nhà cung cấp @MaNCC cung cấp, thông tin gồm ProductID, SumOfProduct, SumOfSubTotal

(sử dụng các bảng [Purchasing].[Vendor] [Purchasing].[PurchaseOrderHeader] và [Purchasing].[PurchaseOrderDetail])

7) Viết hàm tên **Discount_Func** tính số tiền giảm trên các hóa đơn(SalesOrderID), thông tin gồm SalesOrderID, [SubTotal], Discount; trong đó Discount được tính như sau:

Nếu [SubTotal]<1000 thì Discount=0

Nếu 1000<=[SubTotal]<5000 thì Discount = 5%[SubTotal]

Nếu 5000<=[SubTotal]<10000 thì Discount = 10%[SubTotal]

Nếu [SubTotal>=10000 thì Discount = 15%[SubTotal]

Gợi ý: Sử dụng Case.. When … Then …

```
case
when SubTotal<1000 then 0
when SubTotal>=1000 and SubTotal<5000 then [SubTotal]*0.05
when SubTotal>=5000 and SubTotal<10000 then  [SubTotal]*0.1
else SubTotal*0.15
end
```

(Sử dụng dữ liệu từ bảng [Sales].[SalesOrderHeader])

8) Viết hàm **TotalOfEmp** với tham số @MonthOrder, @YearOrder để tính tổng doanh thu của các nhân viên bán hàng (SalePerson) trong tháng và năm được truyền vào 2 tham số, thông tin gồm [SalesPersonID], Total, với Total=Sum([SubTotal])

– **Multi-statement Table Valued Functions:**

9) Viết lại các câu 5,6,7,8 bằng Multi-statement table valued function

10) Viết hàm tên **SalaryOfEmp** trả về kết quả là bảng lương của nhân viên, với tham số vào là @MaNV (giá trị của [BusinessEntityID]), thông tin gồm BusinessEntityID, FName, LName, Salary (giá trị của cột Rate).

– Nếu giá trị của tham số truyền vào là Mã nhân viên khác Null thì kết quả là bảng lương của nhân viên đó.

Ví dụ thực thi hàm: `select * from SalaryOfEmp(288)`

	ID	FName	LName	Salary
1	288	Rachel	Valdez	25.3846

Kết quả là:

- Nếu giá trị truyền vào là Null thì kết quả là bảng lương của tất cả nhân viên

  Ví dụ: thực thi hàm `select * from SalaryOfEmp(Null)`

  Kết quả là 316 record

	ID	FName	LName	Salary
1	1	Ken	Sánchez	138.05
2	2	Terri	Duffy	69.8077
3	3	Roberto	Tamburello	47.5961
4	4	Rob	Walters	9.482
5	4	Rob	Walters	26.092
6	4	Rob	Walters	32.8308
7	5	Gail	Erickson	35.9615
8	6	Jossef	Goldberg	35.9615
9	7	Dylan	Miller	55.5289

(Dữ liệu lấy từ 2 bảng [HumanResources].[EmployeePayHistory] và [Person].[Person])

Module 5. TRIGGER

Mục tiêu**: Sinh viên hiểu được khái niệm vai trò của trigger trong CSDL, phân biệt được các loại trigger, hiện thực được các loại trigger trên CSDL **AdventureWorks

- ***Trigger** chỉ có thể được kích hoạt một cách tự động bởi một trong các lệnh **Insert, Update, Delete**.*
- *Khi một **trigger** được kích hoạt:*
  - *Dữ liệu được **insert** hoặc **update** sẽ được chứa trong bảng **Inserted***
  - *Dữ liệu bị **delete** được chứa trong bảng **Deleted**.*

  Inserted và Deleted** là 2 bảng tạm chỉ có giá trị bên trong trigger, thường dùng để kiểm tra dữ liệu trước khi **commit** hay **roll back
- *Có thể áp dụng trigger cho **View***
- *Loại trigger:*
  - ***INSTEAD OF trigger**: bỏ qua các hành động kích hoạt trigger (các thao tác insert, delete, update), thay vào đó nó sẽ thực hiện các câu lệnh bên trong trigger. Thường dùng trong **View** với các chức năng sau:*
    - *Cập nhật nhiều bảng trong view cùng một lúc*
    - *Tăng điều kiện ràng buộc trên các thuộc tính so với CHECK*
    - *Đánh giá trạng thái của bảng trước và sau khi cập nhật dữ liệu.*
    - *Cho phép từ chối một phần các câu lệnh và thực thi phần còn lại*
  - ***AFTER trigger (FOR):** được thực thi sau khi các câu lệnh SQL thực thi thành công. AFTER trigger là trigger mặc định khi dùng từ FOR.*
    - *Không dùng trong view*

1. Tạo một **Instead of** trigger thực hiện trên **view**. Thực hiện theo các bước sau:
   - Tạo mới 2 bảng **M_Employees** và **M_Department** theo cấu trúc sau:

```
create table M_Department
(
    DepartmentID int not null primary key,
    Name nvarchar(50),
    GroupName nvarchar(50)
)
```

```
create table M_Employees
(
        EmployeeID int not null primary key,
        Firstname nvarchar(50),
        MiddleName nvarchar(50),
        LastName nvarchar(50),
        DepartmentID int foreign key references M_Department(DepartmentID)
)
```

- Tạo một **view** tên **EmpDepart_View** bao gồm các field: EmployeeID, FirstName, MiddleName, LastName, DepartmentID, Name, GroupName, dựa trên 2 bảng **M_Employees** và **M_Department.**
- Tạo một trigger tên **InsteadOf_Trigger** thực hiện trên view **EmpDepart_View,** dùng để chèn dữ liệu vào các bảng **M_Employees** và **M_Department** khi chèn một record mới thông qua view **EmpDepart_View.**

Dữ liệu test:

insert **EmpDepart_view** values(1, 'Nguyen','Hoang','Huy', 11,'Marketing','Sales')

Kết quả:

	DepartmentID	Name	groupName
1	11	Marketing	Sales

	EmployeeID	Firtname	MiddleName	LastName	DepartmentID
1	1	Nguyen	Hoang	Huy	11

2. Tạo một trigger thực hiện trên bảng **MSalesOrders** có chức năng thiết lập độ ưu tiên của khách hàng (**CustPriority**) khi người dùng thực hiện các thao tác **Insert, Update** và **Delete** trên bảng **MSalesOrders** theo điều kiện như sau:
   - Nếu tổng tiền **Sum(SubTotal)** của khách hàng dưới 10,000 $ thì độ ưu tiên của khách hàng **(CustPriority)** là 3
   - Nếu tổng tiền **Sum(SubTotal)** của khách hàng từ 10,000 $ đến dưới 50000 $ thì độ ưu tiên của khách hàng **(CustPriority)** là 2
   - Nếu tổng tiền Sum(SubTotal) của khách hàng từ 50000 $ trở lên thì độ ưu tiên của khách hàng **(CustPriority)** là 1

Các bước thực hiện:

- Tạo bảng **MCustomers** và **MSalesOrders** theo cấu trúc

  sau:
```
create table MCustomer
(
        CustomerID int not null primary key,
        CustPriority int
```

```
)
create table MSalesOrders
(
        SalesOrderID int not null primary key,
        OrderDate date,
        SubTotal money,
        CustomerID int foreign key references MCustomer(CustomerID) )
```

- Chèn dữ liệu cho bảng **MCustomers**, lấy dữ liệu từ bảng **Sales.Customer**, nhưng chỉ lấy CustomerID>30100 và CustomerID<30118, cột CustPriority cho giá trị null.
- Chèn dữ liệu cho bảng **MSalesOrders**, lấy dữ liệu từ bảng **Sales.SalesOrderHeader,** chỉ lấy những hóa đơn của khách hàng có trong bảng khách hàng.
- Viết trigger để lấy dữ liệu từ 2 bảng inserted và deleted.
- Viết câu lệnh kiểm tra việc thực thi của trigger vừa tạo bằng cách chèn thêm hoặc xóa hoặc update một record trên bảng **MSalesOrders**

3. Viết một trigger thực hiện trên bảng **MEmployees** sao cho khi người dùng thực hiện chèn thêm một nhân viên mới vào bảng **MEmployees** thì chương trình cập nhật số nhân viên trong cột **NumOfEmployee** của bảng **MDepartment.** Nếu tổng số nhân viên của phòng tương ứng <=200 thì cho phép chèn thêm, ngược lại thì hiển thị thông báo "Bộ phận đã đủ nhân viên" và hủy giao tác. Các bước thực hiện:

- Tạo mới 2 bảng **MEmployees** và **MDepartment** theo cấu trúc sau:

```
create table MDepartment
(
        DepartmentID int not null primary key,
        Name nvarchar(50),
        NumOfEmployee int
)
create table MEmployees
(
        EmployeeID int not null,
        FirstName nvarchar(50),
        MiddleName nvarchar(50),
        LastName nvarchar(50),
        DepartmentID int foreign key references MDepartment(DepartmentID),
        constraint pk_emp_depart primary key(EmployeeID, DepartmentID)
)
```

- Chèn dữ liệu cho bảng **MDepartment**, lấy dữ liệu từ bảng **Department, cột** NumOfEmployee gán giá trị NULL, bảng **MEmployees** lấy từ bảng **EmployeeDepartmentHistory**
- Viết trigger theo yêu cầu trên và viết câu lệnh hiện thực trigger

4. Bảng **[Purchasing].[Vendor]**, chứa thông tin của nhà cung cấp, thuộc tính **CreditRating** hiển thị thông tin đánh giá mức tín dụng, có các giá trị:

   1 = Superior

   2 = Excellent

   3 = Above average

   4 = Average

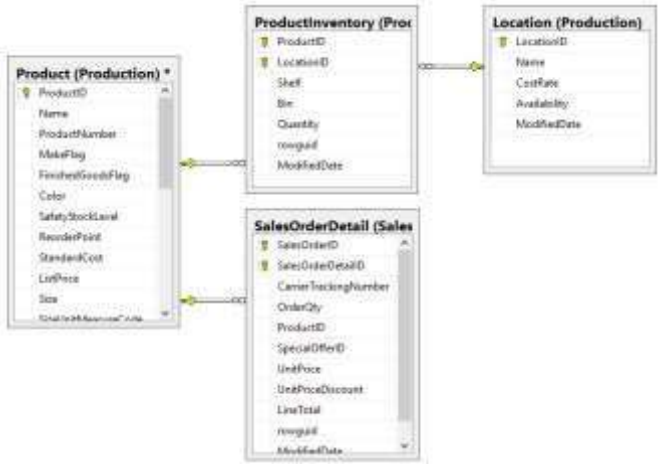
   5 = Below average

Viết một trigger nhằm đảm bảo khi chèn thêm một record mới vào bảng **[**Purchasing].[PurchaseOrderHeader**],** nếu Vender có CreditRating=5 thì hiển thị thông báo không cho phép chèn và đồng thời hủy giao tác.

Dữ liệu test

INSERT INTO Purchasing.PurchaseOrderHeader (RevisionNumber, Status, EmployeeID, VendorID, ShipMethodID, OrderDate, ShipDate, SubTotal, TaxAmt, Freight) VALUES (2 ,3, 261, 1652, 4 ,GETDATE() ,GETDATE() , 44594.55, ,3567.564, ,1114.8638);

5. Viết một trigger thực hiện trên bảng **ProductInventory** (lưu thông tin số lượng sản phẩm trong kho). Khi chèn thêm một đơn đặt hàng vào bảng **SalesOrderDetail** với số lượng xác định trong field **OrderQty**, nếu số lượng trong kho **Quantity> OrderQty** thì cập nhật lại số lượng trong kho **Quantity= Quantity- OrderQty**, ngược lại nếu **Quantity=0** thì xuất thông báo "*Kho hết hàng*" và đồng thời *hủy giao tác*.



Sơ đồ của các bảng liên quan.

6. Tạo trigger cập nhật tiền thưởng (**Bonus**) cho nhân viên bán hàng **SalesPerson**, khi người dùng chèn thêm một record mới trên bảng **SalesOrderHeader**, theo quy định như sau: Nếu tổng tiền bán được của nhân viên có hóa đơn mới nhập vào bảng **SalesOrderHeader** có giá trị >10000000 thì tăng tiền thưởng lên 10% của mức thưởng hiện tại. Cách thực hiện:
   - Tạo hai bảng mới **M_SalesPerson** và **M_SalesOrderHeader**

```
create table M_SalesPerson
(
	SalePSID int not null primary key,
	TerritoryID int,
	BonusPS money
)
create table M_SalesOrderHeader
(
	SalesOrdID int not null primary key,
	OrderDate date,
	SubTotalOrd money,
	SalePSID int foreign key references M_SalesPerson(SalePSID)
)
```

   - Chèn dữ liệu cho hai bảng trên lấy từ **SalesPerson** và **SalesOrderHeader** chọn những field tương ứng với 2 bảng mới tạo.
   - Viết trigger cho thao tác **insert** trên bảng **M_SalesOrderHeader,** khi trigger thực thi thì dữ liệu trong bảng **M_SalesPerson** được cập nhật.

Module 6. ROLE - PERMISSION

Sử dụng SSMS (Sql Server Management Studio) để thực hiện các thao tác sau:

1) Đăng nhập vào SQL bằng SQL Server authentication, tài khoản sa. Sử dụng T-SQL.
2) Tạo hai login SQL server Authentication **User2** và **User3**
3) Tạo một database user **User2** ứng với login **User2** và một database user **User3** ứng với login **User3** trên CSDL AdventureWorks2008.
4) Tạo 2 kết nối đến server thông qua login **User2** và **User3**, sau đó thực hiện các thao tác truy cập CSDL của 2 user tương ứng (VD: thực hiện câu Select). Có thực hiện được không?
5) Gán quyền select trên Employee cho User2, kiểm tra kết quả. Xóa quyền select trên Employee cho User2. Ngắt 2 kết nối của User2 và User3
6) Trở lại kết nối của sa, tạo một user-defined database Role tên Employee_Role trên CSDL AdventureWorks2008, sau đó gán các quyền Select, Update, Delete cho Employee_Role.
7) Thêm các User2 và User3 vào Employee_Role. Tạo lại 2 kết nối đến server thông qua login User2 và User3 thực hiện các thao tác sau:
    a) Tại kết nối với User2, thực hiện câu lệnh Select để xem thông tin của bảng Employee
    b) Tại kết nối của User3, thực hiện cập nhật JobTitle=’Sale Manager’ của nhân viên có BusinessEntityID=1
    c) Tại kết nối User2, dùng câu lệnh Select xem lại kết quả.
    d) Xóa role Employee_Role, (quá trình xóa role ra sao?)

Module 7. TRANSACTION

Mục tiêu:

- *Sinh viên hiểu khái niệm về transaction, 4 thuộc tính cơ bản của transaction, cơ chế hoạt động của transaction*
- *Hiện thực được transaction trong một ngữ cảnh cụ thể*

I. SINGLE TRANSACTION

Autocommit mode là chế độ quản lý giao dịch mặc định của SQL Server Database Engine. Mỗi lệnh Transact-SQL được Commit hoặc Rollback khi nó hoàn thành.

1) Thêm vào bảng **Department** một dòng dữ liệu tùy ý bằng câu lệnh INSERT..VALUES…
   a) Thực hiện lệnh chèn thêm vào bảng Department một dòng dữ liệu tùy ý bằng cách thực hiện lệnh **Begin tran** và **Rollback,** dùng câu lệnh Select * From Department xem kết quả.
   b) Thực hiện câu lệnh trên với lệnh **Commit** và kiểm tra kết quả.
2) Tắt chế độ **autocommit** của SQL Server (**SET IMPLICIT_TRANSACTIONS ON**). Tạo đoạn **batch** gồm các thao tác:
   - Thêm một dòng vào bảng **Department**
   - Tạo một bảng **Test** (ID int, Name nvarchar(10))
   - Thêm một dòng vào **Test**
   - ROLLBACK;
   - Xem dữ liệu ở bảng **Department** và **Test** để kiểm tra dữ liệu, giải thích kết quả.
3) Viết đoạn batch thực hiện các thao tác sau (lưu ý thực hiện lệnh SET XACT_ABORT ON: nếu câu lệnh T-SQL làm phát sinh lỗi run-time, toàn bộ giao dịch được chấm dứt và Rollback)
   - Câu lệnh SELECT với phép chia 0 :SELECT 1/0 as Dummy
   - Cập nhật một dòng trên bảng **Department** với DepartmentID=’9’ (id này không tồn tại)
   - Xóa một dòng không tồn tại trên bảng **Department** (DepartmentID =’66’)
   - Thêm một dòng bất kỳ vào bảng **Department**
   - COMMIT;

Thực thi đoạn batch, quan sát kết quả và các thông báo lỗi và giải thích kết quả.

4) Thực hiện lệnh SET XACT_ABORT OFF (những câu lệnh lỗi sẽ rollback, transaction vẫn tiếp tục) sau đó thực thi lại các thao tác của đoạn batch ở câu 3. Quan sát kết quả và giải thích kết quả?

Tuần 9

II. CONCURRENT TRANSACTIONS (Các giao tác đồng thời)

1) Tạo bảng **Accounts** (AccountID int NOT NULL PRIMARY KEY,
balance int NOT NULL
CONSTRAINT unloanable_account CHECK (balance >= 0)

Chèn dữ liệu:

```
INSERT INTO Accounts (acctID,balance) VALUES (101,1000);
INSERT INTO Accounts (acctID,balance) VALUES (202,2000);
```

2) **SET TRANSACTION ISOLATION LEVEL**

```
SET TRANSACTION ISOLATION LEVEL
    { READ UNCOMMITTED
    | READ COMMITTED
    | REPEATABLE READ
    | SNAPSHOT
    | SERIALIZABLE
    }[ ; ]
```

- **READ UNCOMMITTED**: có thể đọc những dòng đang được hiệu chỉnh bởi các transaction khác nhưng chưa commit
- **READ COMMITTED:** không thể đọc những dòng đang hiệu chỉnh bởi những transaction khác mà chưa commit

3) Mở 2 cửa sổ Query của SQL server, thiết lập **SET TRANSACTION ISOLATION LEVEL READ COMMITTED** ở cả 2 cửa sổ (tạm gọi là client A bên trái, và client B bên phải)

- **B1: Client A, client B: cùng thực hiện lệnh SELECT trên bảng Accounts với AccountID** =101
- **B2: Client A** cập nhật account trên **AccountID** =101, balance =1000-200
- **B3: Client B** cập nhật account trên **AccountID** =101, balance =1000-500
- **B4: Client A**: SELECT trên Accounts với **AccountID** =101; COMMIT;

- **B5: Client B**: SELECT trên Accounts với **AccountID** =101; COMMIT;

Quan sát kết quả hiển thị và giải thích.

4) Thiết lập **ISOLATION LEVEL REPEATABLE READ** *(không thể đọc được dữ liệu đã được hiệu chỉnh nhưng chưa commit bởi các transaction khác và không có transaction khác có thể hiệu chỉnh dữ liệu đã được đọc bởi các giao dịch hiện tại cho đến transaction hiện tại hoàn thành)* ở 2 client. Thực hiện yêu cầu sau:

- **B1: Client A, client B**: cùng thực hiện lệnh SELECT trên bảng Accounts với **AccountID** =101
- **B2: Client A** cập nhật accounts trên **AccountID** =101, balance =1000-200
- **B3: Client B** cập nhật accounts trên **AccountID** =101, balance =1000-500.
- **B4: Client A**: SELECT trên Accounts với **AccountID** =101; COMMIT;

Quan sát kết quả hiển thị và giải thích.

5) Giả sử có 2 giao dịch chuyển tiền từ tài khoản 101 và 202 như sau:

- Client A chuyển 100$ từ tài khoản 101 sang 202
- Client B chuyển 200$ từ tài khoản 202 sang 101.

Viết các lệnh tương ứng ở 2 client để kiểm soát các giao dịch xảy ra đúng

6) Xóa tất cả dữ liệu của bảng Accounts. Thêm lại các dòng mới

```
INSERT INTO Accounts (AccountID ,balance) VALUES (101,1000);
INSERT INTO Accounts (AccountID ,balance) VALUES (202,2000);
```

- **B1: Client A**: cập nhật balance của account giảm đi 100 cho **AccountID** =101, cập nhật balance của account tăng lên 100 cho **AccountID** =202
- **B2: Client B**: thiết lập ISOLATION LEVEL READ UNCOMMITTED

```
SELECT * FROM Accounts;
COMMIT;
```

– **B3: Client A:**

```
ROLLBACK;

SELECT * FROM Accounts;
COMMIT;
```

Quan sát kết quả và giải thích.

7) Xóa tất cả dữ liệu của bảng Account, thêm lại các dòng mới

```
INSERT INTO Accounts (AccountID ,balance) VALUES (101,1000);

INSERT INTO Accounts (AccountID  ,balance) VALUES (202,2000);
```

- **B1: Client A**: thiết lập ISOLATION LEVEL REPEATABLE READ;

Lấy ra các Accounts có Balance>1000

- **B2: Client B:**

```
INSERT INTO Accounts (AccountID ,balance) VALUES

(303,3000);

COMMIT;
```

- **B3: Client A:**

```
SELECT * FROM Accounts WHERE balance > 1000;

COMMIT;
```

Quan sát kết quả và giải thích.

Module 8. Bảo trì cơ sở dữ liệu

Mục tiêu:

- *Backup và Recovery cơ sở dữ liệu*

1. Trong SQL Server, tạo thiết bị backup có tên adv2008back lưu trong thư mục T:\backup\adv2008back.bak
2. Attach CSDL AdventureWorks2008, chọn mode recovery cho CSDL này là full, rồi thực hiện full backup vào thiết bị backup vừa tạo
3. Mở CSDL AdventureWorks2008, tạo một transaction giảm giá tất cả mặt hàng xe đạp trong bảng Product xuống $15 nếu tổng trị giá các mặt hàng xe đạp không thấp hơn 60%.
4. Thực hiện các backup sau cho CSDL AdventureWorks2008, tất cả backup đều lưu vào thiết bị backup vừa tạo
    a. Tạo 1 differential backup
    b. Tạo 1 transaction log backup
5. (Lưu ý ở bước 7 thì CSDL AdventureWorks2008 sẽ bị xóa. Hãy lên kế hoạch phục hồi cơ sở dữ liệu cho các hoạt động trong câu 5, 6).
   Xóa mọi bản ghi trong bảng Person.EmailAddress, tạo 1 transaction log backup
6. Thực hiện lệnh:
    a. Bổ sung thêm 1 số phone mới cho nhân viên có mã số business là 10000 như sau:
       INSERT INTO Person.PersonPhone VALUES (10000,'123-456-7890',1,GETDATE())
    b. Sau đó tạo 1 differential backup cho AdventureWorks2008 và lưu vào thiết bị backup vừa tạo.
    c. Chú ý giờ hệ thống của máy.
       Đợi 1 phút sau, xóa bảng Sales.ShoppingCartItem
7. Xóa CSDL AdventureWorks2008
8. Để khôi phục lại CSDL:
    a. Như lúc ban đầu (trước câu 3) thì phải restore thế nào?
    b. Ở tình trạng giá xe đạp đã được cập nhật và bảng Person.EmailAddress vẫn còn nguyên chưa bị xóa (trước câu 5) thì cần phải restore thế nào?
    c. Đến thời điểm đã được chú ý trong câu 6c thì thực hiện việc restore lại CSDL AdventureWorks2008 ra sao?
9. **Thực hiện đoạn lệnh sau**:

```
CREATE DATABASE Plan2Recover;
USE Plan2Recover;

CREATE TABLE T1 (
```

```
        PK INT Identity PRIMARY KEY,
        Name VARCHAR(15)
        );
GO

INSERT T1 VALUES ('Full');
GO

BACKUP DATABASE Plan2Recover
TO DISK = 'T:\P2R.bak'
WITH NAME = 'P2R_Full',
INIT;
```

Tiếp tục thực hiện các lệnh sau:

```
INSERT T1 VALUES ('Log 1');
GO

BACKUP Log Plan2Recover
TO DISK ='T:\P2R.bak'
WITH NAME = 'P2R_Log';
```

Tiếp tục thực hiện các lệnh sau:

```
INSERT T1 VALUES ('Log 2');
GO

BACKUP Log Plan2Recover
TO DISK ='T:\P2R.bak'
WITH NAME = 'P2R_Log';
```

Xóa CSDL vừa tạo, rồi thực hiện quá trình khôi phục như sau:

```
Use Master;

RESTORE DATABASE Plan2Recover
FROM DISK = 'T:\P2R.bak'
With FILE = 1, NORECOVERY;

RESTORE LOG Plan2Recover
FROM DISK ='T:\P2R.bak'
With FILE = 2, NORECOVERY;

RESTORE LOG Plan2Recover
FROM DISK ='T:\P2R.bak'
With FILE = 3, RECOVERY;
```

Tuần 10

KIỂM TRA BÀI TẬP THỰC HÀNH