Chương 1

TỔNG QUAN VỀ CƠ SỞ DỮ LIỆU

- Giới thiệu các vấn đề căn bản về cơ sở dữ liệu:
  - Các cơ sở dữ liệu được xây dựng như thế nào?
  - Cách tạo các bảng dữ liệu và quan hệ?
  - Cách xây dựng các câu lệnh truy vấn?
  - Các giao dịch cơ sở dữ liệu là gì?
- Giúp sinh viên tìm hiểu cách sử dụng:
  - Chương trình SQL Server Manager Studio 2012
  - Microsoft Visual Studio .Net 2013
- Khám phá cơ sở dữ liệu Northwind

- Cơ sở dữ liệu và các khái niệm
- Ngôn ngữ truy vấn có cấu trúc
- Tạo các đối tượng trong cơ sở dữ liệu
  - Bảng (Table)
  - Khung nhìn (View)
  - Thủ tục lưu trữ, hàm và Triggers
  - Ràng buộc (Constraints)
  - Chỉ mục (Index)

- Giao dịch cơ sở dữ liệu
- Sử dụng Microsoft SQL Server 2012
- Khám phá cơ sở dữ liệu Northwind

- Một cơ sở dữ liệu là một tập các thông tin có tổ chức

- Cơ sở dữ liệu quan hệ là tập hợp các thông tin có liên quan với nhau và được tổ chức thành các cấu trúc được gọi là bảng (***Table***)
  - Table = Rows + Columns

- Để truy xuất đến một cơ sở dữ liệu, chúng ta dùng ngôn ngữ truy vấn có cấu trúc (***SQL – Structured Query Language***)

- Hệ thống được dùng để quản lý thông tin trong cơ sở dữ liệu gọi là hệ quản trị cơ sở dữ liệu.

- Trong trường hợp cơ sở dữ liệu điện tử, hệ quản trị cơ sở dữ liệu là phần mềm quản lý thông tin trong bộ nhớ và các tập tin máy tính.

- Một số hệ quản trị cơ sở dữ liệu thông dụng là Microsoft SQL Server, Oracle và DB2.

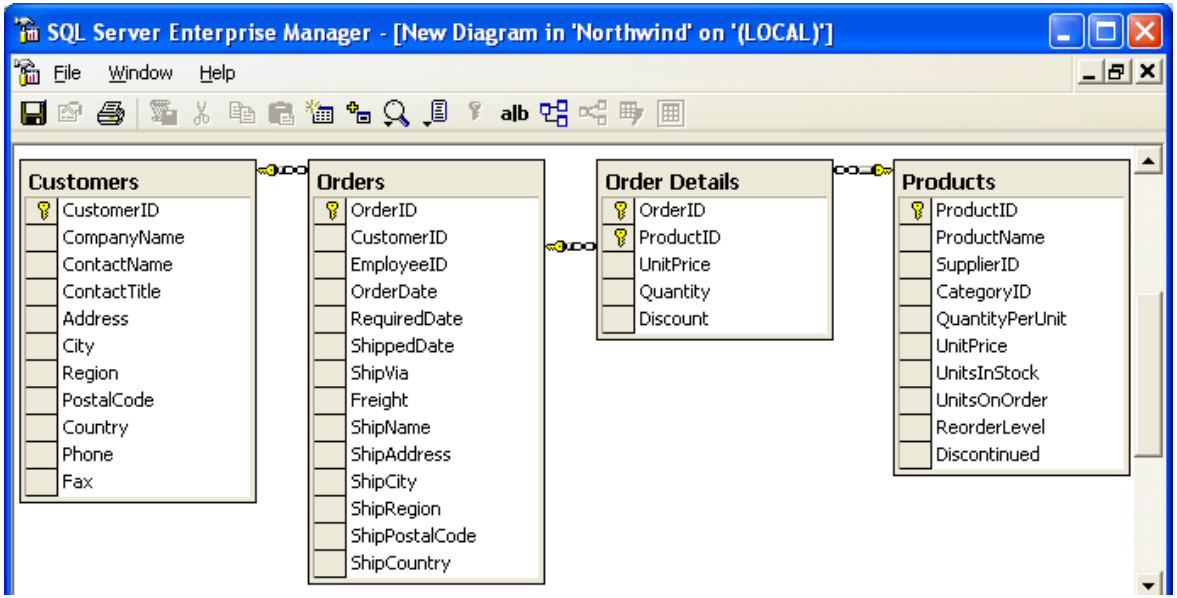
- Infomix
- Dbms
- XTF
- Orache
- PostgreSQL
- SQLServer
- SQL Anywhere
- DB2
- MySQL
- FireBird
- Progress
- Synergy
- Virtuaso
- Ingres

- Phân biệt sự khác nhau giữa hai khái niệm: Một cơ sở dữ liệu là một tập thông tin có cấu trúc còn hệ quản trị cơ sở dữ liệu là phần mềm cho phép lưu trữ và cung cấp các công cụ để thao tác trên các thông tin được lưu trữ đó.

- Lược đồ cơ sở dữ liệu (***database schema***) là thuật ngữ dùng để chỉ cấu trúc của dữ liệu, bao gồm cả định nghĩa về các bảng, các cột tạo nên cơ sở dữ liệu.

- Khóa chính: là một hay nhiều cột của bảng và giá trị trên các cột này xác định duy nhất một mẫu tin trong bảng.
- Khóa kép (***composite key***): là khóa chính được tạo bởi nhiều cột.
- Giá trị của khóa chính trên mỗi mẫu tin trong bảng phải là duy nhất.
- Một cơ sở dữ liệu có thể có nhiều bảng và các bảng lại có quan hệ với nhau.
- Khóa ngoại: là tập các cột của một bảng có tham chiếu đến tập các cột của bảng khác.

- Các quan hệ một-nhiều được tạo nên bởi các khóa ngoại (***foreign keys***).
- Bảng chứa khóa ngoại còn được gọi là ***bảng con.*** Bảng chứa cột mà khóa ngoại tham chiếu đến gọi là ***bảng cha***.
- Khóa ngoại có thể xem như một con trỏ từ bảng con đến bảng cha.



- Giá trị null là giá trị rỗng hoặc giá trị chưa biết.
  - Columns: null or not-null

- Một chỉ mục (***index***) trong một bảng của cơ sở dữ liệu được sử dụng để tìm kiếm các dòng cụ thể trong một bảng
  - Nhược điểm: làm chậm quá trình INSERT dữ liệu

- Những trường hợp nên dùng chỉ mục
  - Một câu truy vấn trả về không quá 10% toàn bộ số hàng trong một bảng.
  - Cột cần làm chỉ mục phải có một miền giá trị lớn.
  - Các cột chứa các giá trị duy nhất trên mỗi dòng.

- Kiểu dữ liệu là loại giá trị mà bạn có thể lưu trữ trong một cột của bảng

bigint	char	ntext
int	Varchar	binary
smallint	text	varbinary
tinyint	nchar	image
bit	float	cursor
Decimal	real	sql_variant
Numeric	datetime	table
money	smalldatetime	timestamp
smallmoney	nvarchar	uniqueidentifier

NGÔN NGỮ TRUY VẤN CÓ CẤU TRÚC SQL

- SQL là một ngôn ngữ chuẩn cho phép truy xuất đến các cơ sở dữ liệu quan hệ

- 2 Loại câu lệnh SQL thông dụng:
  - DML – Data Manipulation Language
  - DDL – Data Definition Language

- Các lệnh ***DML*** cho phép bạn truy xuất, thêm, hiệu chỉnh và xóa các dòng được lưu trong cơ sở dữ liệu.

- Các lệnh ***DDL*** cho phép bạn tạo ra các cấu trúc cơ sở dữ liệu như bảng, khung nhìn,…

- ***SELECT***: Lấy các dòng từ một hay nhiều bảng
- ***INSERT***: Thêm một hay nhiều dòng mới vào một bảng
- ***UPDATE***: Cập nhật giá trị cho một hay nhiều dòng của bảng
- ***DELETE***: Xóa một hay nhiều dòng khỏi một bảng

- Lấy các dòng từ một hay nhiều bảng bằng lệnh SELECT
- Cú pháp

```
SELECT [ALL | DISTINCT][TOP n] danh_sách_chọn
[INTO tên_bảng_mới]
FROM danh_sách_bảng/khung_nhìn
[WHERE   điều_kiện]
[GROUP BY danh_sách_cột]
[HAVING điều_kiện]
[ORDER BY cột_sắp_xếp]
[COMPUTE danh_sách_hàm_gộp [BY danh_sách_cột]]
```

- Toán tử so sánh (dùng sau mệnh đề WHERE hoặc HAVING)

Các toán tử so sánh

Toán tử	ý nghĩa
=	Bằng
>	Lớn hơn
<	Nhỏ hơn
>=	Lớn hơn hoặc bằng
<=	Nhỏ hơn hoặc bằng
<>	Khác
!>	Không lớn hơn
!<	Không nhỏ hơn

- Ký tự đại diện (dùng để kiểm tra khuôn dạng của dữ liệu)

Ký tự đại diện	ý nghĩa
%	Chuỗi ký tự bất kỳ gồm không hoặc nhiều ký tự
_	Ký tự đơn bất kỳ
[]	Ký tự đơn bất kỳ trong giới hạn được chỉ định (ví dụ [a-f]) hay một tập (ví dụ [abcdef])
[^]	Ký tự đơn bất kỳ không nằm trong giới hạn được chỉ định (ví dụ [^a-f] hay một tập (ví dụ [^abcdef]).

- Các hàm gộp - được sử dụng để tính giá trị thống kê cho toàn bảng hoặc trên mỗi nhóm dữ liệu

Hàm gộp	Chức năng
SUM([ALL | DISTINCT] *biểu_thức*)	Tính tổng các giá trị.
AVG([ALL | DISTINCT] *biểu_thức*)	Tính trung bình của các giá trị
COUNT([ALL | DISTINCT] *biểu_thức*)	Đếm số các giá trị trong biểu thức.
COUNT(*)	Đếm số các dòng được chọn.
MAX(*biểu_thức*)	Tính giá trị lớn nhất
MIN(*biểu_thức*)	Tính giá trị nhỏ nhất

Trong đó:

- Hàm SUM và AVG chỉ làm việc với các biểu thức số.
- Hàm SUM, AVG, COUNT, MIN và MAX bỏ qua các giá trị NULL khi tính toán.
- Hàm COUNT(*) không bỏ qua các giá trị NULL.

- Bổ sung dữ liệu bằng câu lệnh INSERT
- Cú pháp 1: Bổ sung một dòng vào bảng
  - INSERT INTO tên_bảng[(danh_sách_cột)] VALUES (danh_sách_trị)
- Cú pháp 2: Bổ sung nhiều dòng vào bảng
  - INSERT INTO tên_bảng[(danh_sách_cột)] câu_lệnh_SELECT
- Phần nằm trong cặp dấu [ ] có thể có hoặc không.
- Nếu có, kết quả của câu lệnh SELECT phải có số cột bằng với số cột được chỉ định trong bảng đích và phải tương thích về kiểu dữ liệu.

- Cập nhật dữ liệu dùng lệnh UPDATE

```
UPDATE tên_bảng
SET tên_cột = biểu_thức
[, ..., tên_cột_k = biểu_thức_k]
[FROM danh_sách_bảng]
[WHERE điều_kiện]
```

- Xóa dữ liệu bằng lệnh DELETE

```
DELETE FROM tên_bảng
FROM danh_sách_bảng ]
[ WHERE điều_kiện ]
```

- Khi sử dụng các lệnh DML để thêm, xóa hay cập nhật dữ liệu, cấn phải bảo đảm 2 nguyên tắc sau:
  - Khóa chính của một dòng luôn chứa một giá trị duy nhất.
  - Khóa ngoại của một dòng trong bảng con luôn tham chiếu đến một giá trị tồn tại trong bảng cha.
- Các tình huống
  - Thêm: trùng giá trị trên các cột khóa chính
  - Cập nhật: giá trị trên các cột khóa chính
  - Xóa: dòng đang có dòng khác tham chiếu đến.

- Một giao dịch (transaction) là một chuỗi một hoặc nhiều câu lệnh SQL được kết hợp lại với nhau thành một khối (đơn vị) công việc.

- Đặc điểm của các câu lệnh SQL trong giao dịch:
  - Có mối quan hệ tương đối mật thiết với nhau
  - Thực hiện các thao tác độc lập
  - Tất cả các câu lệnh đòi hỏi hoặc phải thực thi trọn vẹn (commit) hoặc không một câu lệnh nào được thực thi (rollback).

- Việc kết hợp các câu lệnh lại với nhau trong một giao dịch nhằm đảm bảo tính toàn vẹn dữ liệu và khả năng phục hồi dữ liệu.

- Tính chất
  - Tính nguyên tử (Atomicity): Một giao dịch được xem như một câu lệnh đơn.
  - Tính nhất quán (Consistency): Sau khi giao dịch kết thúc, sự toàn vẹn của dữ liệu phải được bảo toàn.
  - Tính độc lập (Isolation): Một giao dịch khi được thực thi đồng thời với những giao dịch khác trên cùng hệ thống không chịu bất kỳ sự ảnh hưởng nào của các giao dịch đó.
  - Tính bền vững (Durability): Sau khi một giao dịch đã thực hiện thành công, mọi tác dụng mà nó đã tạo ra phải tồn tại bền vững trong cơ sở dữ liệu, cho dù là hệ thống có bị lỗi đi chăng nữa.

- Giao dịch SQL được định nghĩa dựa trên các câu lệnh xử lý giao dịch sau đây:
  - BEGIN TRANSACTION: Bắt đầu một giao dịch
  - SAVE TRANSACTION: Đánh dấu một vị trí trong giao dịch (gọi là điểm đánh dấu).
  - ROLLBACK TRANSACTION: Quay lui trở lại đầu giao dịch hoặc một điểm đánh dấu trước đó trong giao dịch.
  - COMMIT TRANSACTION: Đánh dấu điểm kết thúc một giao dịch. Khi câu lệnh này thực thi cũng có nghĩa là giao dịch đã thực hiện thành công.
  - ROLLBACK [WORK]: Quay lui trở lại đầu giao dịch.
  - COMMIT [WORK]: Đánh dấu kết thúc giao dịch.

- Theo mặc định, giao dịch sẽ bị hủy nếu không thực hiện lệnh COMMIT để xác nhận các thay đổi.

- Bạn có thể kiểm tra các lỗi trong giao dịch trước khi quyết định thực hiện lệnh COMMIT hay ROLLBACK bằng cách dùng hàm @@ERROR.

- Hàm này trả về giá trị 0 nếu một lệnh được thực thi và không gây ra lỗi.

- Ví dụ:

```
BEGIN TRANSACTION MyTransaction;

INSERT INTO Customers ( CustomerID, CompanyName )
VALUES ( 'SYCOM', 'Steve Yellow Company' );

INSERT INTO Orders ( CustomerID )
VALUES ( 'SYCOM' );

IF @@Error = 0
    COMMIT TRANSACTION MyTransaction;
ELSE
    ROLLBACK TRANSACTION MyTransaction;
```

- Các lệnh DDL – Data Definition Language – cho phép tạo ra các cấu trúc của cơ sở dữ liệu như
  - Bảng (Tables)
  - Chỉ mục (Indexes)
  - Khung nhìn (views)
  - Thủ tục & hàm
  - Triggers, …

- Để tạo, thay đổi và xóa các đối tượng này, ta dùng các lệnh CREATE, ALTER và DROP.

```
CREATE TABLE Persons (
 PersonID int CONSTRAINT PK_Persons PRIMARY KEY,
 FirstName nvarchar(15) NOT NULL,
 LastName nvarchar(15) NOT NULL,
 DateOfBirth datetime
);


ALTER TABLE Persons
ADD Address nvarchar(50);


ALTER TABLE Persons
DROP COLUMN Address;

ALTER TABLE Persons
ADD EmployerID nchar(5) CONSTRAINT FK_Persons_Customers REFERENCES
Customers(CustomerID);

DROP TABLE Persons
```

```
CREATE INDEX LastNameIndex
ON Persons(LastName);

DROP INDEX Persons.LastNameIndex
```

- Khung nhìn là một bảng "ảo" trong cơ sở dữ liệu có nội dung được định nghĩa thông qua một truy vấn (câu lệnh SELECT)

- Phân biệt với bảng (tables)
  - Giống nhau: có tên & cùng cấu trúc (cột).
  - Khác nhau: về mặt lưu trữ dữ liệu

- Ưu điểm của khung nhìn
  - Bảo mật dữ liệu
  - Đơn giản hoá các thao tác truy vấn dữ liệu
  - Tập trung và đơn giản hóa dữ liệu
  - Độc lập dữ liệu

- Câu lệnh CREATE VIEW được sử dụng để tạo ra khung nhìn và có cú pháp như sau:

```
CREATE VIEW tên_khung_nhìn[(danh_sách_tên_cột)]
AS
    câu_lệnh_SELECT
```

- Số cột của khung nhìn bằng với số cột trong kết quả của câu truy vấn SELECT
- Không thể qui định ràng buộc và tạo chỉ mục cho khung nhìn.
- Câu lệnh SELECT với mệnh đề COMPUTE ... BY không được sử dụng để định nghĩa khung nhìn.

- Phải đặt tên cho các cột của khung nhìn trong các trường hợp sau đây:
  - Trong kết quả của câu lệnh SELECT có ít nhất một cột được sinh ra bởi một biểu thức (tức là không phải là một tên cột trong bảng cơ sở) và cột đó không được đặt tiêu đề.
  - Tồn tại hai cột trong kết quả của câu lệnh SELECT có cùng tiêu đề cột

```
CREATE VIEW ListProducts
AS
      SELECT ProductID, ProductName, UnitsInStock
            CategoryName, CompanyName AS Supplier
      FROM Products pro, Categories cat, Suppliers sup
      WHERE pro.CategoryID = cat.CategoryID
            AND pro.SupplierID = sup.SupplierID
```

- Một thủ tục là một đối tượng trong cơ sở dữ liệu bao gồm một tập nhiều câu lệnh SQL được nhóm lại với nhau thành một nhóm với những khả năng sau:
  - Có thể sử dụng các cấu trúc điều khiển (IF, WHILE, FOR).
  - Có thể sử dụng các biến như trong ngôn ngữ lập trình nhằm lưu giữ các giá trị tính toán được, các giá trị được truy xuất được từ cơ sở dữ liệu.
  - Có thể nhận các tham số truyền vào cũng như có thể trả về các giá trị thông qua các tham số (như trong các ngôn ngữ lập trình).
  - Khi một thủ tục lưu trữ đã được định nghĩa, nó có thể được gọi thông qua tên thủ tục

- Tạo thủ tục

```
CREATE PROCEDURE tên_thủ_tục [ (danh_sách_tham_số) ]
[ WITH RECOMPILE | ENCRYPTION | RECOMPILE , ENCRYPTION ]
AS
        Các_câu_lệnh_của_thủ_tục
```

  - Cách 1: Tên_thủ_tục [danh_sách_các_đối_số]
  - Cách 2: EXECUTE Tên_thủ_tục [danh_sách_các_đối_số]

- Số lượng các đối số cũng như thứ tự của chúng phải phù hợp với số lượng và thứ tự của các tham số khi định nghĩa thủ tục.

```
CREATE PROCEDURE AddProduct
  @MyProductName nvarchar(40),
  @MySupplierID int,
  @MyCategoryID int,
  @MyQuantityPerUnit nvarchar(20),
  @MyUnitPrice money,
  @MyUnitsInStock smallint,
  @MyUnitsOnOrder smallint,
  @MyReorderLevel smallint,
  @MyDiscontinued bit
AS
DECLARE @ProductID int

  -- insert a row into the Products table

  INSERT INTO Products (
    ProductName, SupplierID, CategoryID, QuantityPerUnit,
    UnitPrice, UnitsInStock, UnitsOnOrder, ReorderLevel,
    Discontinued
  ) VALUES (
    @MyProductName, @MySupplierID, @MyCategoryID, @MyQuantityPerUnit,
    @MyUnitPrice, @MyUnitsInStock, @MyUnitsOnOrder, @MyReorderLevel,
    @MyDiscontinued
  )

  -- use the @@IDENTITY function to get the last inserted
  -- identity value, which in this case is the ProductID of
  -- the new row in the Products table
  SET @ProductID = @@IDENTITY
```

- Hàm là đối tượng cơ sở dữ liệu tương tự như thủ tục.

- Điểm khác biệt giữa hàm và thủ tục:
  - Hàm trả về một giá trị thông qua tên hàm còn thủ tục thì không.
  - Điều này cho phép ta sử dụng hàm như là một thành phần của một biểu thức

```
CREATE FUNCTION tên_hàm ( [danh_sách_tham_số] )
RETURNS ( kiểu_trả_về_của_hàm )
AS
BEGIN
      các_câu_lệnh_của_hàm
END
```

- Hàm trả về dữ liệu kiểu bảng (hay còn gọi là hàm nội tuyến)

- Q

```
CREATE FUNCTION tên_hàm ( [danh_sách_tham_số] )
RETURNS @biến_bảng TABLE định_nghĩa_bảng
AS
      BEGIN
            các_câu_lệnh_trong_thân_hàm
            RETURN
      END
```

ETURNS TABLE.

RETURN xác định SELECT.

  - Ngoài ra, không sử dụng bất kỳ câu lệnh nào khác trong phần thân của hàm.

```
CREATE FUNCTION DiscountPrice(@OriginalPrice money, @Discount float)
RETURNS money
AS
BEGIN

  RETURN @OriginalPrice * @Discount
END

-- Tính tiền giảm với UnitPrice = 10 và Discount = 0.3
SELECT dbo.DiscountPrice(10, 0.3);

-- Tính tiền giảm với Discount = 0.3 và UnitPrice
-- lấy từ bảng Products của csdl Northwind
SELECT dbo.DiscountPrice(UnitPrice, 0.3), UnitPrice
FROM Products
WHERE ProductID = 1;

-- Tương tự
DECLARE @MyDiscountFactor float
SET @MyDiscountFactor = 0.3
SELECT dbo.DiscountPrice(UnitPrice, @MyDiscountFactor), UnitPrice
FROM Products
WHERE ProductID = 1;
```

```
CREATE FUNCTION ProductsToBeReordered2(@ReorderLevel int)
RETURNS @MyProducts table
(

  ProductID int,
  ProductName nvarchar(40),
  UnitsInStock smallint,
  Reorder nvarchar(3)
)
AS
BEGIN

  -- retrieve rows from the Products table and
  -- insert them into the MyProducts table,
  -- setting the Reorder column to 'No'
  INSERT INTO @MyProducts
    SELECT ProductID, ProductName, UnitsInStock, 'No'
    FROM Products;

  -- update the MyProducts table, setting the
  -- Reorder column to 'Yes' when the UnitsInStock
  -- column is less than or equal to @ReorderLevel
  UPDATE @MyProducts
  SET Reorder = 'Yes'
  WHERE UnitsInStock <= @ReorderLevel

  RETURN
END
```

```
SELECT * FROM ProductsToBeReordered2(20);
```

- Thường được sử dụng để đảm bảo tính toàn vẹn dữ liệu trong cơ sở dữ liệu

- Trigger cũng chứa một tập các câu lệnh SQL và tập các câu lệnh này sẽ được thực thi khi trigger được gọi.

- Phân biệt giữa thủ tục và trigger
  - Thủ tục lưu trữ được thực thi khi người sử dụng có lời gọi đến chúng
  - Trigger được "gọi" tự động khi xảy ra những thao tác làm thay đổi dữ liệu trong các bảng

- Mỗi một trigger được tạo ra và gắn liền với một bảng nào đó trong cơ sở dữ liệu.
- Khi dữ liệu trong bảng bị thay đổi (tức là khi bảng chịu tác động của các câu lệnh INSERT, UPDATE hay DELETE) thì trigger sẽ được tự đông kích hoạt.

```
CREATE TRIGGER tên_trigger
ON tên_bảng
FOR {[INSERT][,][UPDATE][,][DELETE]}
AS
      [IF UPDATE(tên_cột)
      [AND UPDATE(tên_cột)|OR UPDATE(tên_cột)]
      ...]
     các_câu_lệnh_của_trigger
```

- Ví dụ

```
CREATE TABLE mathang
(
      mahang NVARCHAR(5) PRIMARY KEY,    /*mã hàng*/
      tenhang NVARCHAR(50) NOT NULL,    /*tên hàng*/
      soluong INT,    /*số lượng hàng hiện có*/
)

CREATE TABLE  nhatkybanhang
(
      stt    INT IDENTITY PRIMARY KEY,
      ngay  DATETIME,  /*ngày bán hàng*/
      nguoimua NVARCHAR(30), /*tên người mua hàng*/
      mahang  NVARCHAR(5)    /*mã mặt hàng được bán*/
       FOREIGN KEY REFERENCES mathang(mahang),
      soluong  INT,    /*giá bán hàng*/
      giaban  MONEY  /*số lượng hàng được bán*/
)
```

```
CREATE TRIGGER trg_nhatkybanhang_insert
ON nhatkybanhang
FOR INSERT
AS
    UPDATE mathang
    SET mathang.soluong = mathang.soluong - inserted.soluong
    FROM mathang INNER JOIN inserted
                 ON mathang.mahang=inserted.mahang
```

Mã Hàng	Tên Hàng	Số Lượng
H1	Xà phòng	30
H2	Kem đánh răng	45

```
INSERT INTO nhatkybanhang
     (ngay,nguoimua,mahang,soluong,giaban)
VALUES('5/5/2004','Tran Ngoc Thanh','H1',10,5200)
```

- Tạo bảng
- Viết truy vấn SQL
- Tạo thủ tục
- Tạo Trigger

HẾT CHƯƠNG 1