





**PHẠM HỮU KHANG (Chủ biên)  
ĐOÀN THIỆN NGÂN (Hiệu đính)**

# SQL Server

## Lập trình 2005

## Thủ tục và hàm

NHÀ XUẤT BẢN LAO ĐỘNG - XÃ HỘI

**GIỚI THIỆU**3 

## GIỚI THIỆU

Sau khi tham khảo kiểu dữ liệu, từ khóa, biểu thức bảng và phát biểu SQL trong tập 1, chương 8 của tập 2 tiếp tục giới thiệu các đặc điểm mới về loại hàm có sẵn và hàm do người sử dụng định nghĩa trong SQL Server 2005.

Sang chương 9, bạn tiếp tục tìm hiểu cách khai báo và sử dụng phát biểu SQL phức tạp cùng với việc định nghĩa phát biểu SQL động nhằm cung cấp giải pháp cho những trường hợp không thể sử dụng phát biểu SQL thông thường; cũng trong chương này bạn có thể tìm thấy hai hàm PIVOT cho phép luân chuyển dữ liệu từ chiều dọc theo chiều ngang và hàm UNPIVOT thực hiện quá trình ngược lại.

Bước sang chương 10, bạn có thể tìm hiểu cách khai báo và sử dụng biến trong lập trình T-SQL. Bên cạnh đó bạn cũng tìm thấy kiểu dữ liệu Table và phát biểu điều khiển, đặc biệt trong chương này bạn sẽ tìm thấy tính đa dạng trong truy vấn dữ liệu khi làm việc với phép toán CASE.

Tiếp tục sang chương 11, bạn sẽ khám phá cấu trúc và chức năng của thủ tục nội tại, trong đó những tính năng mới được thể hiện qua việc khai báo theo nhóm, các loại tham số, mượn quyền thực thi và lợi ích khi sử dụng thủ tục nội tại trong ứng dụng thực tế.

Sang chương 12, bạn tiếp tục khám phá DML Trigger và DDL Trigger; trong đó loại DDL Trigger là đối tượng mới giới thiệu trong SQL Server 2005 cho phép bạn quản lý biến động của Server và cơ sở dữ liệu SQL Server. Ngoài ra, bạn cũng tìm hiểu 3 đối tượng DEFAULTS, RULES và TYPES trong chương 13.

Giáo trình bao gồm 6 chương và ứng dụng nhấn mạnh xuyên suốt từ kiểu dữ liệu, hàm, phát biểu SQL động, phát biểu SQL phức tạp, phát biểu điều khiển, thủ tục nội tại, trigger giúp cho bạn sử dụng chúng trong ứng dụng thực tế và nhiều kỹ thuật quan trọng khác cùng với nhiều ví dụ chi tiết, diễn giải rõ ràng.

Cuốn sách “SQL Server 2005 - Lập trình Thủ tục và Hàm” nằm trong bộ giáo trình SQL Server 2005 bao gồm nhiều cuốn từ lập trình T-SQL, lập trình Thủ tục và Hàm, lập trình nâng cao, ứng dụng SQL Server 2005 trong hệ thống kế toán và quản trị cơ sở dữ liệu SQL Server 2005.

**MK.PUB**

## HƯỚNG DẪN SỬ DỤNG CD ĐÍNH KÈM THEO SÁCH

Để sử dụng các ví dụ trong CD đính kèm theo sách, trước tiên bạn chép hai tập tin AccountSystem.mdf và AccountSystem.ldf vào ổ đĩa cứng, sử dụng Management Studio để tạo cơ sở dữ liệu AccountSystem dạng Attach bằng cách thực hiện như sau:

- Chọn ngăn Databases và R-Click.
- Chọn Attach... rồi chọn nút Add trong cửa sổ vừa xuất hiện.
- Chọn tập tin cơ sở dữ liệu AccountSystem.mdf và nhấn nút OK.
- Kiểm tra đường dẫn cơ sở dữ liệu sẽ lưu trữ và nhấn nút OK.

Nếu muốn phục hồi cơ sở dữ liệu từ tập tin .bak, bạn có thể thực hiện các bước như sau:

- Chọn ngăn Databases và R-Click.
- Chọn Restore Database rồi đặt tên AccountSystem trong phần "To database".
- Chọn tùy chọn From device rồi trỏ đến tập tin cơ sở dữ liệu bạn backup là AccountSystem.bak và nhấn nút OK.
- Kiểm tra đường dẫn cơ sở dữ liệu sẽ lưu và nhấn nút OK.

Sau đó, bạn chép thư mục Projects vào ổ đĩa cứng, trong mỗi cuốn sách có nhiều dự án ứng với các phần trong hệ thống kế toán và ví dụ tham khảo cho cuốn sách được quản lý bằng Management Studio. Để mở giải pháp (Solution) trong Management Studio, bạn chọn vào File | Open Project | Solution và chọn vào tập tin AccountSystemSoln.

Ví dụ đính kèm theo sách được tổ chức theo từng Project, bao gồm nhiều Project ứng với chức năng trong hệ thống kế toán, sau khi mở Solution trong Management Studio, bạn có thể chọn từng Project để thực thi phát biểu SQL bằng cách khởi động cửa sổ Query hoặc chọn tên tập tin .sql rồi Double Click.

**THƯ NGỎ**

5 

# THƯ NGỎ

**Kính thưa quý Bạn đọc gần xa!**

Trước hết, Ban xuất bản xin bày tỏ lòng biết ơn và niềm vinh hạnh được đồng đảo Bạn đọc nhiệt tình ủng hộ tủ sách MK.PUB.

Trong thời gian qua chúng tôi rất vui và cảm ơn các Bạn đã gửi e-mail đóng góp nhiều ý kiến quý báu cho tủ sách.

Mục tiêu và phương châm phục vụ của chúng tôi là:

- *Lao động khoa học nghiêm túc.*
- *Chất lượng và ngày càng chất lượng hơn.*
- *Tất cả vì Bạn đọc.*

**Một lần nữa, Ban xuất bản MK.PUB xin kính mời quý Bạn đọc tiếp tục tham gia cùng chúng tôi để nâng cao chất lượng sách.** Cụ thể:

Trong quá trình sử dụng sách, nếu quý Bạn phát hiện thấy bất kỳ sai sót nào (dù nhỏ) xin đánh dấu, ghi chú nhận xét ý kiến của Bạn ra bên cạnh rồi gửi cuốn sách này cho chúng tôi theo địa chỉ:

**Nhà sách Minh Khai**

249 Nguyễn Thị Minh Khai, Q.1, Tp. Hồ Chí Minh.

E-mail: [mk.book@minhkhai.com.vn](mailto:mk.book@minhkhai.com.vn) hoặc [mk.pub@minhkhai.com.vn](mailto:mk.pub@minhkhai.com.vn)

Chúng tôi xin hoàn lại cước phí bưu điện và gửi trả lại Bạn cuốn sách cùng tên. Ngoài ra chúng tôi còn gửi tặng Bạn một cuốn sách khác trong tủ sách MK.PUB. Bạn có thể chọn cuốn sách này theo danh mục thích hợp sẽ gửi tới Bạn.

Với mục đích ngày càng nâng cao chất lượng tủ sách MK.PUB, chúng tôi rất mong nhận được sự hợp tác nhiệt tình của quý Bạn đọc gần xa.

**"MK.PUB cùng Bạn đọc đồng hành"** để nâng cao chất lượng sách.

Một lần nữa chúng tôi xin chân thành cảm ơn.

**MK.PUB**

**MỤC LỤC**7 **MỤC LỤC**

|   |           |
|---|-----------|
| <b>GIỚI THIỆU .....</b>                                       | <b>3</b>  |
| <b>THƯ NGỎ.....</b>   | <b>5</b>  |
| <b>MỤC LỤC.....</b>   | <b>7</b>  |
| <b>Chương 8: GIỚI THIỆU HÀM TRONG SQL SERVER 2005 .....</b>   | <b>13</b> |
| 1. Khái niệm hàm do người sử dụng định nghĩa.....             | 13        |
| 1.1. Phát biểu tạo hàm do người sử dụng định nghĩa.....       | 15        |
| 1.2. Phát biểu thay đổi hàm do người sử dụng định nghĩa ..... | 27        |
| 1.3. Phát biểu xóa hàm do người sử dụng định nghĩa .....      | 33        |
| 2. Hàm hệ thống trả về giá trị đơn (Scalar Functions) .....   | 34        |
| 2.1. Nhóm hàm Configuration .....                             | 34        |
| 2.2. Nhóm hàm Cursor .....                                    | 35        |
| 2.3. Nhóm hàm Date và Time .....                              | 36        |
| 2.4. Nhóm hàm Mathematical .....                              | 36        |
| 2.5. Nhóm hàm Metadata .....                                  | 37        |
| 2.6. Nhóm hàm Security .....                                  | 39        |
| 2.7. Nhóm hàm String .....                                    | 41        |
| 2.8. Nhóm hàm System .....                                    | 43        |
| 2.9. Nhóm hàm Statistical .....                               | 50        |
| 3. Hàm trả về đối tượng là tập mẫu tin (Rowset) .....         | 51        |
| 3.1. Hàm OPENQUERY .....                                      | 51        |
| 3.2. Hàm OPENROWSET .....                                     | 57        |

|   |            |
|---|------------|
| <b>3.3. Hàm OPENDATASOURCE.....</b>                           | <b>60</b>  |
| <b>3.4. Hàm OPENXML .....</b>                                 | <b>61</b>  |
| <b>4. Hàm trả về giá trị tổng hợp (Aggregate) .....</b>       | <b>65</b>  |
| <b>5. Hàm trả về khoảng giá trị (Ranking).....</b>            | <b>67</b>  |
| <b>5.1. Hàm RANK.....</b>                                     | <b>67</b>  |
| <b>5.2. Hàm NTILE .....</b>                                   | <b>69</b>  |
| <b>5.3. Hàm DENSE_RANK.....</b>                               | <b>71</b>  |
| <b>5.4. Hàm ROW_NUMBER.....</b>                               | <b>73</b>  |
| <b>6. Kết chương .....</b>                                    | <b>75</b>  |
| <b>Chương 9: PHÁT BIỂU TRUY VẤN DỮ LIỆU NÂNG CAO.....</b>     | <b>77</b>  |
| <b>1. Phát biểu truy vấn động.....</b>                        | <b>77</b>  |
| <b>2. Làm việc với nhiều mệnh đề .....</b>                    | <b>83</b>  |
| <b>2.1. Sử dụng phép toán UNION.....</b>                      | <b>83</b>  |
| <b>2.2. Mệnh đề JOIN với phát biểu SELECT.....</b>            | <b>85</b>  |
| <b>2.3. Sử dụng hàm CASE.....</b>                             | <b>88</b>  |
| <b>3. Phép toán PIVOT và UNPIVOT .....</b>                    | <b>93</b>  |
| <b>3.1. Phép toán PIVOT.....</b>                              | <b>94</b>  |
| <b>3.2. Phép toán UNPIVOT .....</b>                           | <b>97</b>  |
| <b>4. Kết chương .....</b>                                    | <b>99</b>  |
| <b>Chương 10: KHAI BÁO BIẾN VÀ PHÁT BIỂU ĐIỀU KHIỂN .....</b> | <b>101</b> |
| <b>1. Khai báo và sử dụng biến.....</b>                       | <b>101</b> |
| <b>1.1. Khai báo biến .....</b>                               | <b>101</b> |
| <b>1.2. Phát biểu SET .....</b>                               | <b>104</b> |
| <b>1.3. Phát biểu SELECT .....</b>                            | <b>106</b> |

**MỤC LỤC**

|   |            |
|---|------------|
| 2. Biến kiểu Table.....                                 | 109        |
| 3. Phát biểu điều khiển .....                           | 112        |
| 3.1. Phát biểu điều khiển IF..ELSE.....                 | 112        |
| 3.2. Phát biểu điều khiển BEGIN..END .....              | 114        |
| 3.3. Phát biểu điều khiển WHILE.....                    | 115        |
| 3.4. Phát biểu điều khiển RETURN.....                   | 124        |
| 3.5. Phát biểu điều khiển TRY..CATCH.....               | 125        |
| 3.6. Phát biểu điều khiển WAITFOR .....                 | 130        |
| 3.7. Phát biểu điều khiển GOTO.....                     | 133        |
| 4. Hàm CASE.....  | 135        |
| 5. Kết chương .....                                     | 140        |
| <b>Chương 11: KHÁM PHÁ THỦ TỤC NỘI TẠI.....</b>         | <b>141</b> |
| 1. Giới thiệu thủ tục nội tại.....                      | 141        |
| 1.1. Các loại thủ tục nội tại .....                     | 142        |
| 1.2. Cú pháp tạo thủ tục nội tại .....                  | 144        |
| 1.3. Thủ tục đơn giản .....                             | 145        |
| 1.4. Thủ tục nội tại cùng tên (nhóm thủ tục) .....      | 146        |
| 1.5. Thực thi thủ tục .....                             | 148        |
| 2. Thủ tục nội tại với tham số.....                     | 149        |
| 2.2. Gọi thủ tục trong thủ tục.....                     | 154        |
| 2.3. Mã hóa thủ tục nội tại .....                       | 156        |
| 3. Mượn quyền thực thi thủ tục .....                    | 157        |
| 4. Thủ tục với giá trị trả về.....                      | 158        |
| 5. Thủ tục nội tại để thêm, xóa, cập nhật dữ liệu ..... | 162        |

|  |            |
|--|------------|
| 5.1. Thủ tục nội tại để thêm dữ liệu.....                  | 166        |
| 5.2. Thủ tục nội tại để cập nhật dữ liệu.....              | 177        |
| 5.3. Thủ tục nội tại để xóa dữ liệu.....                   | 178        |
| 6. Thủ tục nội tại để truy vấn dữ liệu.....                | 180        |
| 7. Kết chương .....  | 187        |
| <b>Chương 12: KHÁM PHÁ TRIGGER.....</b>                    | <b>189</b> |
| 1. Giới thiệu thủ tục nội tại.....                         | 189        |
| 2. DML Trigger.....  | 191        |
| 2.1. Tại sao sử dụng DML Trigger.....                      | 191        |
| 2.2. Cấu trúc của Trigger.....                             | 193        |
| 2.3. Trigger cho hành động thêm mẫu tin.....               | 196        |
| 2.4. Trigger cho hành động cập nhật mẫu tin.....           | 199        |
| 2.5. Trigger cho hành động xóa mẫu tin.....                | 201        |
| 3. DDL Trigger .....                                       | 209        |
| 3.1. Tại sao sử dụng DDL Trigger .....                     | 209        |
| 3.2. Cấu trúc của Trigger.....                             | 209        |
| 3.3. Danh sách DDL Trigger .....                           | 214        |
| 3.4. Tạo DDL Trigger .....                                 | 216        |
| 3.5. Hàm EVENTDATA.....                                    | 216        |
| 3.6. Trigger kiểm soát cơ sở dữ liệu hiện hành .....       | 218        |
| 3.7. Tạo Trigger để kiểm soát Login User trên Server ..... | 223        |
| 4. So sánh DML Trigger và Constraint.....                  | 225        |
| 5. Kết chương .....  | 226        |

**MỤC LỤC**11 

|  |            |
|--|------------|
| <b>Chương 13: ĐỐI TƯỢNG DEFAULT, RULE, TYPE.....</b>       | <b>227</b> |
| 1. Đối tượng Default .....                                 | 227        |
| 2. Đối tượng Rule .....                                    | 231        |
| 3. Đối tượng Type .....                                    | 233        |
| 4. Kết chương .....  | 238        |
| <b>Ứng dụng: SỬ DỤNG THỦ TỤC NỘI TẠI TRONG ỨNG DỤNG ..</b> | <b>239</b> |
| 1. Phân kế toán công nợ phải thu.....                      | 239        |
| 1.1. <i>Bảng Customers</i> .....                           | 240        |
| 1.2. <i>Bảng SalesInvoiceTypes</i> .....                   | 242        |
| 1.3. <i>Bảng SalesInvoiceBatchs</i> .....                  | 245        |
| 1.4. <i>Bảng SalesInvoices</i> .....                       | 247        |
| 1.5. <i>Bảng SalesInvoiceDetails</i> .....                 | 251        |
| 1.6. <i>Thủ tục hỗ trợ làm báo cáo</i> .....               | 253        |
| 2. Phân kế toán công nợ phải trả.....                      | 267        |
| 2.1. <i>Bảng Suppliers</i> .....                           | 268        |
| 2.2. <i>Bảng PurchaseInvoiceTypes</i> .....                | 270        |
| 2.3. <i>Bảng PurchaseInvoiceBatchs</i> .....               | 273        |
| 2.4. <i>Bảng PurchaseInvoices</i> .....                    | 275        |
| 2.5. <i>Bảng PurchaseInvoiceDetails</i> .....              | 280        |
| 2.6. <i>Thủ tục hỗ trợ làm báo cáo</i> .....               | 283        |
| 3. Phân kế toán tổng hợp.....                              | 295        |
| 3.1. <i>Bảng Receipts</i> .....                            | 295        |
| 3.2. <i>Bảng ReceiptTypes</i> .....                        | 300        |
| 3.3. <i>Bảng ReceiptBatchs</i> .....                       | 302        |

**M<sup>o</sup> 12****MỤC LỤC**

|   |     |
|---|-----|
| <i>3.4. Bảng Payments.....</i>                | 304 |
| <i>3.5. Bảng PaymentTypes.....</i>            | 309 |
| <i>3.6. Bảng PaymentBatches.....</i>          | 311 |
| <i>3.7. Báo cáo tổng hợp thu và chi .....</i> | 314 |
| 4. Phân kế toán xuất nhập tồn .....           | 321 |
| <i>4.1. Phân nhập kho.....</i>                | 322 |
| <i>4.2. Phân xuất kho .....</i>               | 334 |
| <i>4.3. Tình hình tồn kho.....</i>            | 346 |
| 5 Kết chương .....                            | 351 |

Chương 8: Giới thiệu hàm trong SQL Server 2005

## **Chương 8:**

## **GIỚI THIỆU HÀM TRONG SQL SERVER 2005**

## Tóm tắt chương 8

Bạn đã tham khảo cách sử dụng phát biểu SQL cùng với các mệnh đề và phép toán cũng như một số hàm cơ bản để truy vấn và kết xuất dữ liệu theo yêu cầu của nghiệp vụ kế toán dựa trên những dữ liệu mẫu trong tập 1.

Trong chương đầu của tập 2, bạn sẽ tiếp tục tìm hiểu về cách khai báo hàm người dùng và những hàm có sẵn của SQL Server 2005 khi sử dụng phát biểu SELECT để sắp xếp, kết hợp, trích lọc, tính toán dữ liệu.

Các vấn đề chính sẽ được đề cập:

- ✓ Khái niệm hàm do người sử dụng định nghĩa.
  - ✓ Hàm trả về giá trị đơn (Scalar).
  - ✓ Hàm trả về đối tượng là tập mẫu tin (Rowset).
  - ✓ Hàm trả về giá trị tổng hợp (Aggregate).
  - ✓ Hàm trả về khoảng giá trị (Ranking).

## **1. KHÁI NIỆM HÀM DO NGƯỜI SỬ DỤNG ĐỊNH NGHĨA**

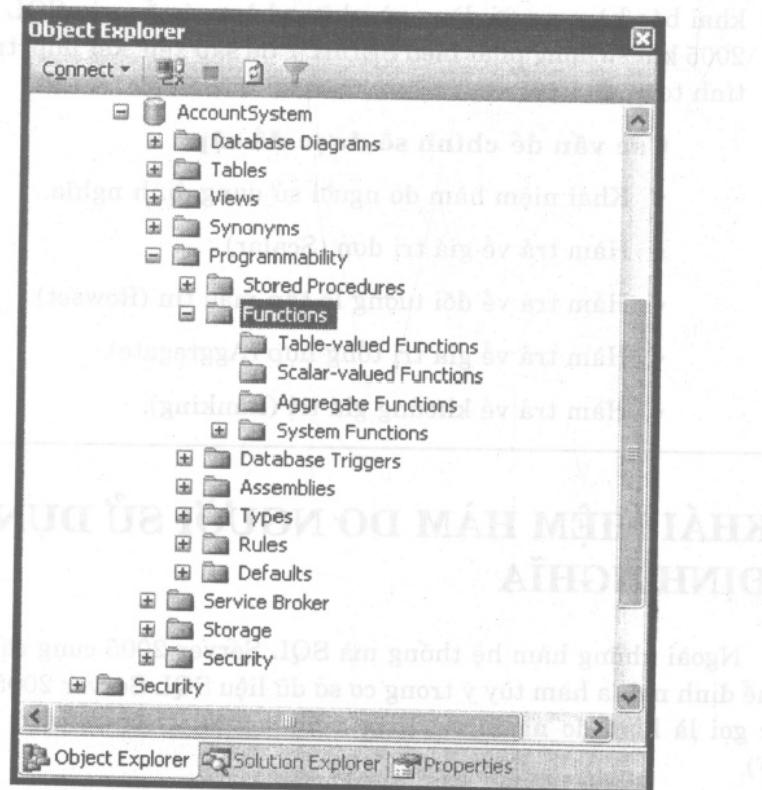
Ngoài những hàm hệ thống mà SQL Server 2005 cung cấp sẵn, bạn có thể định nghĩa hàm tùy ý trong cơ sở dữ liệu SQL Server 2005, hàm này được gọi là hàm do người sử dụng định nghĩa (User-defined Function - UDF).

Chú ý: Bạn có thể tìm thấy các ví dụ trình bày của chương này nằm trong tập tin có tên User-Defined Functions.sql, ScalarFunctions.sql, RowsetFunctions.sql và RankingFunctions.sql.

UDF được xem như thủ tục có thể thực thi việc tính toán và trả về giá trị đơn hay đối tượng Table. UDF đã được giới thiệu lần đầu tiên trong phiên bản Microsoft SQL Server 2000 và bạn có thể sử dụng hàm trong phát biểu T-SQL để tính toán dữ liệu trên cột hay tạo ra ràng buộc trong khi truy vấn dữ liệu.

SQL Server 2005 tiếp tục giới thiệu sự tích hợp hàm với CLR (common language runtime) thuộc Framework 2.0 để cho phép bạn tạo ra UDF bằng ngôn ngữ lập trình .NET 2005 mà bạn chọn trong bộ Visual Studio 2005. Chẳng hạn, bạn sử dụng ngôn ngữ lập trình C# 2005, Visual Basic 2005, J# 2005 hay C++ 2005.

Hàm do bạn tạo ra được lưu giữ trong ngăn Programmability | Functions trông giống như hình 8-1.



**Hình 8-1:** Hàm do người sử dụng định nghĩa.

Chú ý: Để tìm hiểu ngôn ngữ lập trình C#, bạn có thể tìm đọc bộ sách C# 2005 từ cấp độ cơ bản đến nâng cao do tác giả Phạm Hữu Khang biên soạn được nhà sách Minh Khai phát hành trong năm 2006 và 2007.

## 1.1. Phát biểu tạo hàm do người sử dụng định nghĩa

Để tạo hàm dạng UDF, bạn có thể sử dụng phát biểu CREATE FUNCTION.

Tuy nhiên, do cú pháp của phát biểu CREATE FUNCTION khá phức tạp, chúng ta có thể tìm hiểu từng phần của phát biểu này thông qua từng trường hợp cụ thể.

### 1.1.1. Hàm trả về giá trị đơn

Khi bạn có nhu cầu khai báo hàm có kết quả trả về là giá trị đơn (Scalar) trong SQL Server 2005 thì sử dụng cú pháp như sau:

```
CREATE FUNCTION [ schema_name. ] function_name
( [ { @parameter_name [ AS ] [ type_schema_name. ]
      parameter_data_type
      [ = default ] }
      [ ,...n ]
    ]
)
RETURNS return_data_type
[ WITH <function_option> [ , . . . n ] ]
[ AS ]
BEGIN
    function_body
    RETURN scalar_expression
END
[ : ]
```

Trong đó, schema\_name ứng với tên cơ sở dữ liệu, tài khoản người sử dụng dùng để đăng nhập cơ sở dữ liệu; function\_name là tên hàm cần tạo; @parameter\_name ứng với tham số truyền vào hàm và parameter\_data\_type là kiểu dữ liệu của giá trị truyền vào cho tham số.

Phát biểu RETURNS cho phép bạn khai báo kiểu dữ liệu thông qua return\_data\_type và nội dung của hàm được khai báo trong phần BEGIN và END.

Kiểu dữ liệu của giá trị trả về từ hàm bao gồm cả kiểu dữ liệu CLR do người sử dụng định nghĩa ngoại trừ kiểu timestamp.

M 16

**Chương 8: Giới thiệu hàm trong SQL Server 2005**

**Chú ý:** Bạn có thể tham khảo kiểu dữ liệu được giới thiệu trong SQL Server 2005 đã được trình bày trong chương 4 của tập 1.

Chẳng hạn, bạn có thể khai báo hàm udfPreviousMonth nhận tham số là chuỗi gồm 7 ký tự ứng với tháng của năm hiện hành và trả về chuỗi là tháng trước đó như ví dụ 8-1.

**Ví dụ 8-1: Khai báo hàm trả về giá trị đơn**

```
CREATE FUNCTION udfPreviousMonth
(
    @CurrentMonthYear char(7)
)
RETURNS char(7)
WITH EXECUTE AS CALLER
AS
BEGIN
    -- Khai báo hai biến ứng với tháng và năm
    DECLARE @Month TINYINT
    DECLARE @Year SMALLINT
    -- Khai báo hai biến ứng với tháng và năm

    SET @Month = CAST(LEFT(@CurrentMonthYear, 2) AS TINYINT)
    SET @Year = CAST(RIGHT(@CurrentMonthYear, 4) AS
SMALLINT)
    -- Nếu là tháng 1 thì tháng trước đó là tháng 12 của năm trước

    IF ( @Month != 1 )
        SET @Month = @Month - 1
    ELSE
        BEGIN
            SET @Month = 12
            SET @Year = @Year - 1
        END
    RETURN(right('0'+ltrim(@Month),2) + '/' + ltrim(@Year))
END;
GO
```

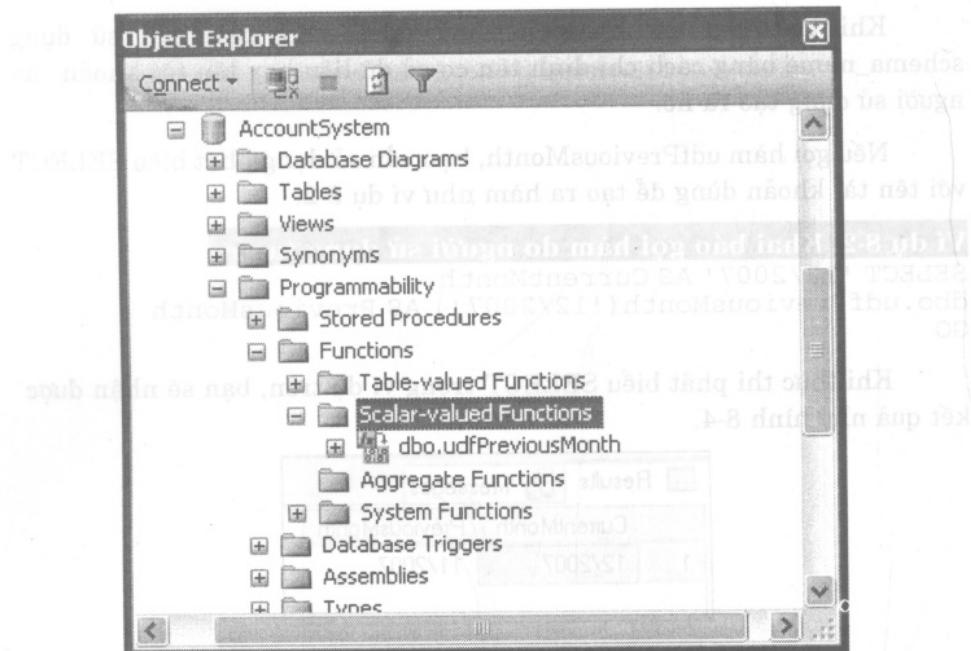
Sau khi thực thi phát biểu tạo hàm trên, hàm udfPreviousMonth sẽ được tạo ra trong cơ sở dữ liệu AccountSystem, bạn có thể tìm thấy hàm này trong ngăn Functions như hình 8-2.

**Chú ý:** Hàm này được sử dụng cho chức năng tính tồn kho và công nợ thu và chi.

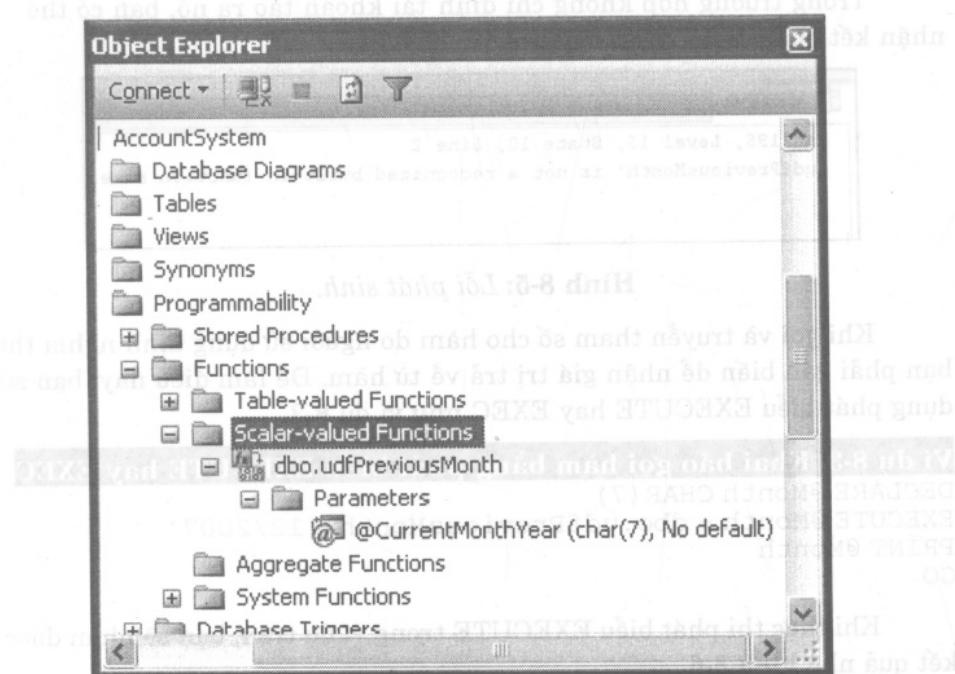
Để biết hàm có bao nhiêu tham số, bạn có thể chọn vào ngăn Parameters, danh sách tham số của hàm sẽ trình bày tương tự như hình 8-3.

Chương 8: Giới thiệu hàm trong SQL Server 2005

17 ®



**Hình 8-2:** Hàm *udfPreviousMonth*.



**Hình 8-3:** Tham số của hàm.

Khi gọi hàm do người sử dụng khai báo, bạn cần sử dụng schema\_name bằng cách chỉ định tên cơ sở dữ liệu hay tên tài khoản của người sử dụng tạo ra nó.

Nếu gọi hàm udfPreviousMonth, bạn cần sử dụng phát biểu SELECT với tên tài khoản dùng để tạo ra hàm như ví dụ 8-2.

**Ví dụ 8-2: Khai báo gọi hàm do người sử dụng tạo ra**

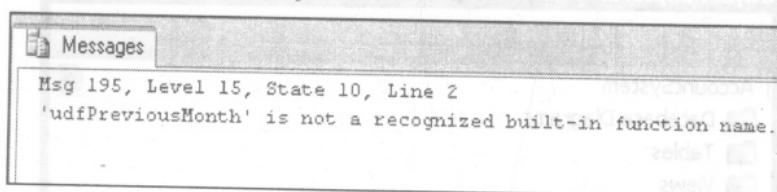
```
SELECT '12/2007' AS CurrentMonth,
dbo.udfPreviousMonth('12/2007') AS PreviousMonth
GO
```

Khi thực thi phát biểu SELECT trong ví dụ trên, bạn sẽ nhận được kết quả như hình 8-4.

|   | CurrentMonth | PreviousMonth |
|---|--------------|---------------|
| 1 | 12/2007      | 11/2007       |

**Hình 8-4: Gọi hàm udfPreviousMonth.**

Trong trường hợp không chỉ định tài khoản tạo ra nó, bạn có thể nhận kết quả với lỗi trình bày như hình 8-5.



**Hình 8-5: Lỗi phát sinh.**

Khi gọi và truyền tham số cho hàm do người sử dụng định nghĩa thì bạn phải gán biến để nhận giá trị trả về từ hàm. Để làm điều này, bạn sử dụng phát biểu EXECUTE hay EXEC như ví dụ 8-3.

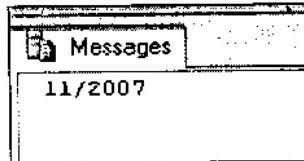
**Ví dụ 8-3: Khai báo gọi hàm bằng phát biểu EXECUTE hay EXEC**

```
DECLARE @Month CHAR(7)
EXECUTE @Month = dbo.udfPreviousMonth '12/2007'
PRINT @Month
GO
```

Khi thực thi phát biểu EXECUTE trong ví dụ trên, bạn sẽ nhận được kết quả như hình 8-6.

**Chương 8: Giới thiệu hàm trong SQL Server 2005**

19

**Hình 8-6: Gọi hàm bằng phát biểu EXECUTE.**

Nếu khai báo giá trị mặc định cho tham số trong hàm do người sử dụng định nghĩa, bạn có thể truyền giá trị cho tham số hoặc không.

Chẳng hạn, bạn khai báo hàm có giá trị mặc định cho tham số ProvinceId để lấy ra tổng số hóa đơn mua hàng của khách hàng cho một tỉnh thành như ví dụ 8-4.

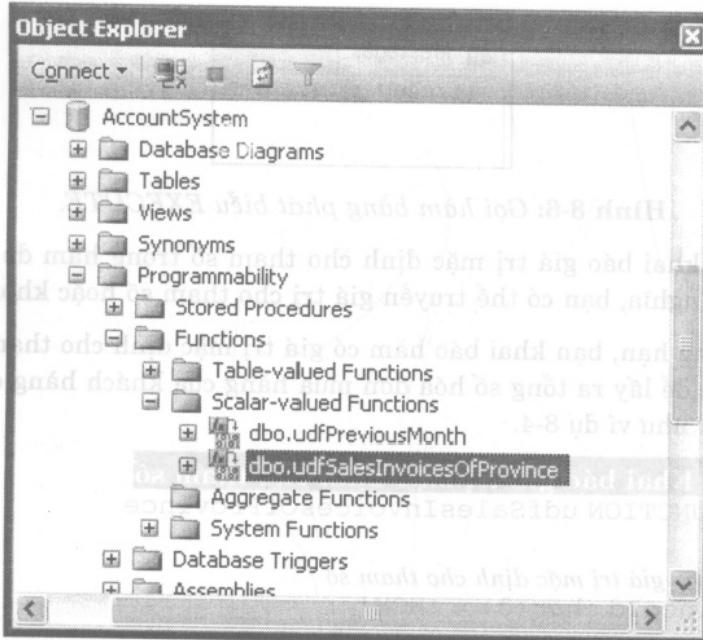
**Ví dụ 8-4: Khai báo giá trị mặc định cho tham số**

```
CREATE FUNCTION udfSalesInvoicesOfProvince
(
    -- Khai báo giá trị mặc định cho tham số
    @ProvinceId char(3) = 'HCM'
)
RETURNS INT
WITH EXECUTE AS CALLER
AS
BEGIN
    -- Khai báo biến nắm giữ tổng số hóa đơn bán hàng
    DECLARE @NumberOfInvoices INT
    -- Gán biến nắm giữ tổng số hóa đơn bán hàng
    SELECT @NumberOfInvoices = COUNT(S.CustomerId)
    FROM Customers C, SalesInvoices S
    WHERE C.CustomerId = S.CustomerId
        AND ProvinceId = @ProvinceId
    RETURN (@NumberOfInvoices)
END;
GO
```

Sau khi thực thi phát biểu CREATE FUNCTION của ví dụ trên, bạn có thể tìm thấy hàm udfSalesInvoicesOfProvince vừa tạo trong ngăn Scalar-valued functions như hình 8-7.



Chương 8: Giới thiệu hàm trong SQL Server 2005



**Hình 8-7:** Hàm *udfSalesInvoicesOfProvince*.

Dể gọi hàm udfSalesInvoicesOfProvince, bạn chỉ định tham số tỉnh thành thông qua cột ProvinceId như ví dụ 8-5.

Ví dụ 8-5: Khai báo goi hàm udfSalesInvoicesOfProvince

```
SELECT ProvinceId, ProvinceName,
      dbo.udfSalesInvoicesOfProvince(ProvinceId) As Invoices
  FROM Provinces
 GO
```

Nếu thực thi phát biểu SELECT trong ví dụ trên, bạn có thể tìm thấy kết quả trình bày tương tự như hình 8-8.

|   | ProvinceId | ProvinceName     | Invoices |
|---|------------|------------------|----------|
| 1 | BDU        | Bình Dương       | 0        |
| 2 | DNA        | Đồng Nai         | 1        |
| 3 | HAN        | Hà Nội           | 2        |
| 4 | HCM        | Hồ Chí Minh      | 10       |
| 5 | TTH        | Thừa Thiên - Huế | 0        |

**Hình 8-8:** Danh sách tỉnh thành.

## Chương 8: Giới thiệu hàm trong SQL Server 2005

21 M®

Tuy nhiên, trong trường hợp không chỉ định tham số cho hàm `udfSalesInvoicesOfProvince`, bạn phải sử dụng từ khóa `DEFAULT` như ví dụ 8-6.

**Ví dụ 8-6: Khai báo gọi hàm với giá trị mặc định**

```
SELECT ProvinceId, ProvinceName,
dbo.udfSalesInvoicesOfProvince(DEFAULT) As Invoices
FROM Provinces
GO
```

Nếu thực thi phát biểu `SELECT` với từ khóa `DEFAULT` trong ví dụ trên, bạn có thể tìm thấy kết quả trình bày tương tự như hình 8-9.

|   | ProvinceId | ProvinceName     | Invoices |
|---|------------|------------------|----------|
| 1 | BDU        | Bình Dương       | 10       |
| 2 | DNA        | Đồng Nai         | 10       |
| 3 | HAN        | Hà Nội           | 10       |
| 4 | HCM        | Hồ Chí Minh      | 10       |
| 5 | TTH        | Thừa Thiên - Huế | 10       |

**Hình 8-9: Danh sách tỉnh thành.**

Chú ý: Khi sử dụng từ khóa `DEFAULT` cho trường hợp tham số có giá trị mặc định, giá trị tại cột `Invoices` sẽ là 10 ứng với số lượng hóa đơn của tỉnh thành có mã là `HCM`.

### 1.1.2. Hàm trả về đối tượng Table

Ngoài loại hàm do người sử dụng định nghĩa trong phần trên, bạn có thể khai báo hàm có kiểu dữ liệu trả về là đối tượng Table bằng cú pháp như sau:

```
CREATE FUNCTION [ schema_name. ] function_name
( [ { @parameter_name [ AS ] [ type_schema_name. ]
parameter_data_type
[ = default ] }
[ ,...n ]
]
)
RETURNS TABLE
[ WITH <function_option> [ ,...n ] ]
[ AS ]
RETURN [ ( ) select_stmt [ ) ]
[ ; ]
```

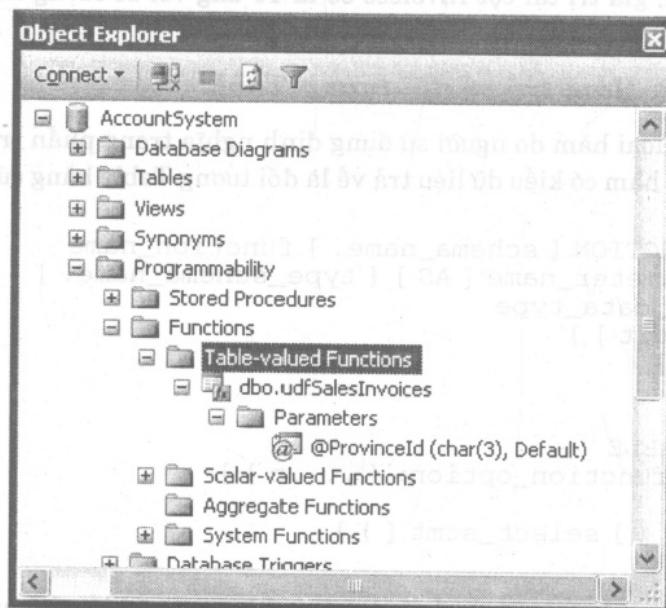
Tương tự như hàm trả về giá trị đơn, hàm trả về kiểu đối tượng Table sẽ là phát biểu `SELECT` trong phần khai báo `RETURN`.

Chẳng hạn, bạn khai báo hàm có tên udfSalesInvoices, nhận tham số là mã tỉnh thành rồi sử dụng biểu thức bảng với phát biểu WITH như ví dụ 8-7.

#### Ví dụ 8-7: Khai báo hàm trả về đối tượng Table

```
CREATE FUNCTION udfSalesInvoices
(
    @ProvinceId char(3) = 'HCM'
)
RETURNS TABLE
AS
RETURN
(
    WITH CustomerList
    AS
    (
        SELECT CustomerId
        FROM Customers
        WHERE ProvinceId = @ProvinceId
    )
    SELECT InvoiceNo, DueDate, CustomerId
    FROM SalesInvoices
    WHERE CustomerId in (SELECT * FROM CustomerList)
)
GO
```

Sau khi thực thi phát biểu CREATE FUNCTION trong ví dụ trên, bạn có thể tìm thấy hàm vừa tạo trong ngăn Functions như hình 8-10.



Hình 8-10: Hàm trả về Table.

**Chương 8: Giới thiệu hàm trong SQL Server 2005**

23

Để gọi hàm có giá trị trả về là đối tượng TABLE, bạn có thể sử dụng phát biểu SELECT với mệnh đề FROM như ví dụ 8-8.

**Ví dụ 8-8: Khai báo gọi hàm udfSalesInvoices**

```
SELECT *
FROM dbo.udfSalesInvoices ('HCM')
GO
```

Khi thực thi phát biểu SELECT trong ví dụ trên, bạn có thể tìm thấy kết quả trình bày như hình 8-11.

|    | InvoiceNo  | DueDate             | CustomerId |
|----|------------|---------------------|------------|
| 1  | SI00000001 | 2007-10-10 00:00:00 | A0001      |
| 2  | SI00000002 | 2007-10-11 00:00:00 | A0002      |
| 3  | SI00000004 | 2007-10-13 00:00:00 | A0001      |
| 4  | SI00000006 | 2007-10-14 00:00:00 | A0005      |
| 5  | SI00000008 | 2007-10-17 00:00:00 | A0007      |
| 6  | SI00000009 | 2007-10-18 00:00:00 | A0008      |
| 7  | SI00000010 | 2007-10-19 00:00:00 | A0001      |
| 8  | SI00000011 | 2007-10-19 00:00:00 | A0002      |
| 9  | SI00000012 | 2007-10-20 00:00:00 | A0001      |
| 10 | SI00000013 | 2007-10-20 00:00:00 | A0005      |

**Hình 8-11: Kết quả gọi hàm udfSalesInvoices.**

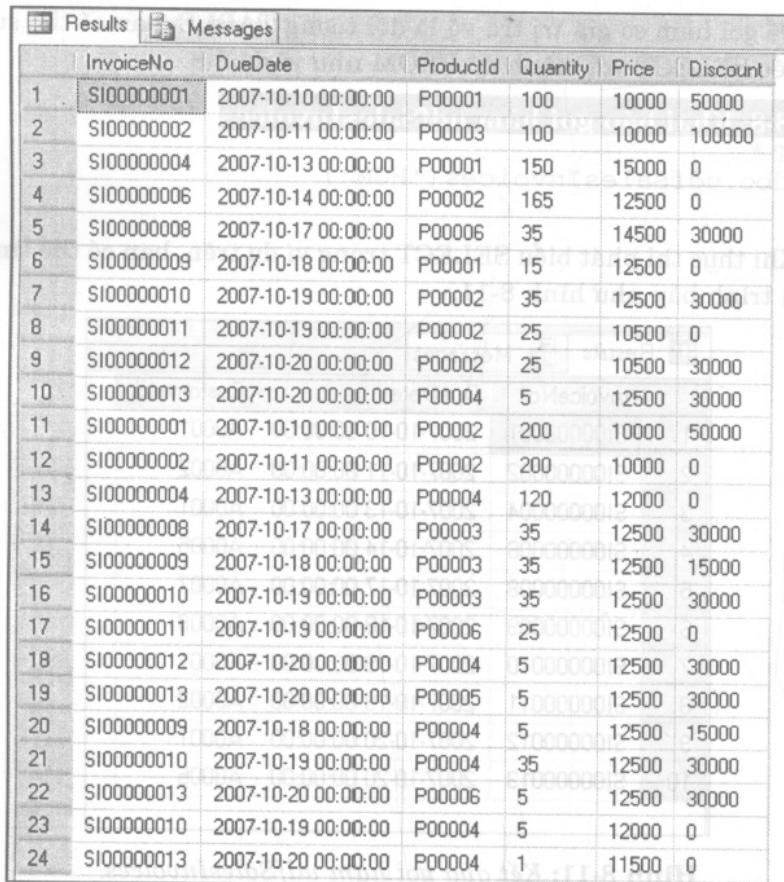
Do hàm udfSalesInvoices trả về tập dữ liệu, nên bạn có thể sử dụng mệnh đề JOIN hay WHERE để kết hợp với các bảng dữ liệu khác.

Chẳng hạn, hàm udfSalesInvoices trả về danh sách hóa đơn bán hàng được liên kết với bảng danh sách hóa đơn bán hàng chi tiết tương tự như khai báo trong ví dụ 8-9.

**Ví dụ 8-9: Khai báo mệnh đề INNER JOIN với hàm udfSalesInvoices**

```
SELECT F.InvoiceNo, DueDate,
D.ProductId, Quantity, Price, Discount
FROM dbo.udfSalesInvoices ('HCM') F
INNER JOIN SalesInvoiceDetails D
ON F.InvoiceNo = D.InvoiceNo
GO
```

Khi thực thi phát biểu SELECT với mệnh đề INNER JOIN trong ví dụ trên, bạn có thể tìm thấy kết quả trình bày như hình 8-12.



|    | InvoiceNo  | DueDate             | ProductId | Quantity | Price | Discount |
|----|------------|---------------------|-----------|----------|-------|----------|
| 1  | S100000001 | 2007-10-10 00:00:00 | P00001    | 100      | 10000 | 50000    |
| 2  | S100000002 | 2007-10-11 00:00:00 | P00003    | 100      | 10000 | 100000   |
| 3  | S100000004 | 2007-10-13 00:00:00 | P00001    | 150      | 15000 | 0        |
| 4  | S100000006 | 2007-10-14 00:00:00 | P00002    | 165      | 12500 | 0        |
| 5  | S100000008 | 2007-10-17 00:00:00 | P00006    | 35       | 14500 | 30000    |
| 6  | S100000009 | 2007-10-18 00:00:00 | P00001    | 15       | 12500 | 0        |
| 7  | S100000010 | 2007-10-19 00:00:00 | P00002    | 35       | 12500 | 30000    |
| 8  | S100000011 | 2007-10-19 00:00:00 | P00002    | 25       | 10500 | 0        |
| 9  | S100000012 | 2007-10-20 00:00:00 | P00002    | 25       | 10500 | 30000    |
| 10 | S100000013 | 2007-10-20 00:00:00 | P00004    | 5        | 12500 | 30000    |
| 11 | S100000001 | 2007-10-10 00:00:00 | P00002    | 200      | 10000 | 50000    |
| 12 | S100000002 | 2007-10-11 00:00:00 | P00002    | 200      | 10000 | 0        |
| 13 | S100000004 | 2007-10-13 00:00:00 | P00004    | 120      | 12000 | 0        |
| 14 | S100000008 | 2007-10-17 00:00:00 | P00003    | 35       | 12500 | 30000    |
| 15 | S100000009 | 2007-10-18 00:00:00 | P00003    | 35       | 12500 | 15000    |
| 16 | S100000010 | 2007-10-19 00:00:00 | P00003    | 35       | 12500 | 30000    |
| 17 | S100000011 | 2007-10-19 00:00:00 | P00006    | 25       | 12500 | 0        |
| 18 | S100000012 | 2007-10-20 00:00:00 | P00004    | 5        | 12500 | 30000    |
| 19 | S100000013 | 2007-10-20 00:00:00 | P00005    | 5        | 12500 | 30000    |
| 20 | S100000009 | 2007-10-18 00:00:00 | P00004    | 5        | 12500 | 15000    |
| 21 | S100000010 | 2007-10-19 00:00:00 | P00004    | 35       | 12500 | 30000    |
| 22 | S100000013 | 2007-10-20 00:00:00 | P00006    | 5        | 12500 | 30000    |
| 23 | S100000010 | 2007-10-19 00:00:00 | P00004    | 5        | 12000 | 0        |
| 24 | S100000013 | 2007-10-20 00:00:00 | P00004    | 1        | 11500 | 0        |

Hình 8-12: Mệnh đề INNER JOIN với hàm udfSalesInvoices.

### 1.1.3. Hàm có nhiều phát biểu SQL

Ngoài hai cách khai báo hàm như trình bày ở trên, bạn có thể khai báo hàm với nhiều phát biểu SQL trong phần thân của nó bằng cách sử dụng cú pháp như sau:

```

CREATE FUNCTION [ schema_name. ] function_name
( [ { @parameter_name [ AS ]
      [ type_schema_name. ] parameter_data_type
      [ = default ] }
      [ ,...n ]
)
RETURNS @return_variable TABLE < table_type_definition >
[ WITH <function_option> [ ,...n ] ]
[ AS ]
BEGIN

```

Chương 8: Giới thiệu hàm trong SQL Server 2005

25

```
        function_body  
    RETURN  
END  
[ ; ]
```

Trong đó, dữ liệu trả về nằm trong biến @return\_variable có kiểu dữ liệu là đối tượng TABLE.

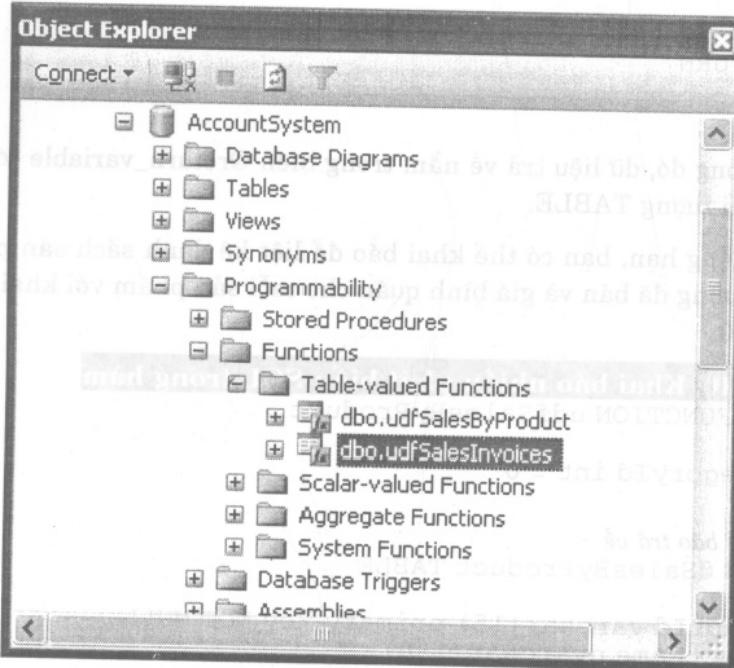
Chẳng hạn, bạn có thể khai báo để liệt kê danh sách sản phẩm với tổng số lượng đã bán và giá bình quân cho mỗi sản phẩm với khai báo như ví dụ 8-10.

Ví dụ 8-10: Khai báo nhiều phát biểu SQL trong hàm

```
CREATE FUNCTION udfSalesByProduct
(
    @CategoryId int = 0
)
-- Khai báo trả về
RETURNS @SalesByProduct TABLE
(
    ProductId varchar(15) primary key NOT NULL,
    ProductName nvarchar(150),
    TotalQuantity int,
    AveragePrice int
)
AS
-- Khai báo nội dung hàm
BEGIN
    INSERT @SalesByProduct
    SELECT D.ProductId, ProductNameInVietnamese,
        SUM(Quantity), AVG(Price)
    FROM Products P, SalesInvoiceDetails D
    WHERE P.ProductId = D.ProductId
    AND CategoryId = CASE @CategoryId WHEN 0
        THEN CategoryId ELSE @CategoryId END
    GROUP BY D.ProductId, ProductNameInVietnamese
    RETURN
END;
GO
```

Sau khi thực thi phát biểu CREATE FUNCTION trong ví dụ trên, hàm udfSalesByProduct tạo ra sẽ nằm trong ngăn Table-Valued Functions như hình 8-13.

Chú ý: Bạn có thể tìm hiểu chi tiết phát biểu CASE và kiểu dữ liệu TABLE trong chương trình bày thủ tục nội tại.



Hình 8-13: Hàm udfSalesByProduct.

Để khai báo gọi hàm udfSalesByProduct với từ khóa DEFAULT, bạn có thể khai báo như ví dụ 8-11.

#### Ví dụ 8-11: Khai báo gọi hàm udfSalesByProduct

```
SELECT * FROM dbo.udfSalesByProduct (DEFAULT)
GO
```

Bạn có thể tìm thấy danh sách sản phẩm cùng với số lượng bán khi thực thi phát biểu SELECT với hàm udfSalesByProduct như hình 8-14.

| ProductId | ProductName                          | TotalQuantity | AveragePrice |
|-----------|--------------------------------------|---------------|--------------|
| 1         | Túi xách                             | 515           | 13125        |
| 2         | Túi xách dùng cho học sinh nữ        | 750           | 10857        |
| 3         | Túi xách dùng cho học sinh nam       | 625           | 11666        |
| 4         | Túi áo mưa                           | 176           | 12214        |
| 5         | Túi xách dùng cho Máy tính           | 130           | 13000        |
| 6         | Túi xách dùng cho Điện thoại di động | 90            | 13500        |

Hình 8-14: Gọi hàm udfSalesByProduct.

Tuy nhiên, nếu khai báo gọi hàm udfSalesByProduct với mã loại sản phẩm là 1, bạn có thể khai báo như ví dụ 8-12.

#### Ví dụ 8-12: Khai báo gọi hàm udfSalesByProduct

```
SELECT * FROM dbo.udfSalesByProduct(1)
GO
```

Khi đó, bạn có thể tìm thấy danh sách sản phẩm cùng với số lượng bán nếu thực thi phát biểu SELECT để gọi hàm udfSalesByProduct với tham số là 1 như hình 8-15.

| ProductId | ProductName                           | TotalQuantity | AveragePrice |
|-----------|---------------------------------------|---------------|--------------|
| 1         | P00001 Túi xách                       | 515           | 13125        |
| 2         | P00002 Túi xách dùng cho học sinh nữ  | 750           | 10857        |
| 3         | P00003 Túi xách dùng cho học sinh nam | 625           | 11666        |
| 4         | P00004 Túi áo mưa                     | 176           | 12214        |

Hình 8-15: Gọi hàm udfSalesByProduct.

Chú ý: Bạn có thể tìm hiểu chi tiết phát biểu CREATE FUNCTION để tạo hàm CLR trong cuốn: SQL Server 2005 - Lập trình nâng cao.

### 1.2. Phát biểu thay đổi hàm do người sử dụng định nghĩa

Trong trường hợp cần thay đổi cấu trúc của hàm do người sử dụng định nghĩa, bạn có thể sử dụng phát biểu ALTER FUNCTION.

Tương tự như trường hợp khai báo phát biểu CREATE FUNCTION, đối với trường hợp này bạn cũng có thể phân ra ba trường hợp.

#### 1.2.1. Hàm trả về giá trị đơn

Khi bạn có nhu cầu khai báo thay đổi cấu trúc hàm có giá trị trả về là giá trị đơn trong SQL Server 2005 thì sử dụng cú pháp như sau:

```
ALTER FUNCTION [ schema_name. ] function_name
( [ { @parameter_name [ AS ] type_schema_name. ]
parameter_data_type
[ = default ] }
[ ,...n ]
)
RETURNS return_data_type
[ WITH <function_option> [ ,...n ] ]
```

**M<sup>®</sup> 28****Chương 8: Giới thiệu hàm trong SQL Server 2005**

```
[ AS ]
BEGIN
    function_body
    RETURN scalar_expression
END
[ ; ]
```

Trong đó, schema\_name ứng với tên cơ sở dữ liệu, tài khoản người sử dụng dùng để đăng nhập cơ sở dữ liệu, function\_name là tên hàm cần thay đổi, @parameter\_name ứng với tham số truyền vào hàm và parameter\_data\_type là kiểu dữ liệu cho giá trị truyền vào cho tham số.

Chẳng hạn, bạn có thể khai báo thay đổi hàm udfSalesInvoicesOfProvince đang tồn tại với tham số là chuỗi gồm 7 ký tự có giá trị mặc định là chuỗi rỗng thay vì 'HCM' như ví dụ 8-1.

**Ví dụ 8-13: Khai báo thay đổi hàm udfSalesInvoicesOfProvince**

```
ALTER FUNCTION udfSalesInvoicesOfProvince
(
    @ProvinceId char(3) = ''
)
RETURNS INT
WITH EXECUTE AS CALLER
AS
BEGIN
    DECLARE @NumberOfInvoices INT
    IF (@ProvinceId = '')
        BEGIN
            SELECT @NumberOfInvoices = COUNT(S.CustomerId)
            FROM Customers C, SalesInvoices S
            WHERE C.CustomerId = S.CustomerId
            AND (ProvinceId = 'HCM' OR ProvinceId = 'HAN' )
        END
    ELSE
        BEGIN
            SELECT @NumberOfInvoices = COUNT(S.CustomerId)
            FROM Customers C, SalesInvoices S
            WHERE C.CustomerId = S.CustomerId
            AND ProvinceId = @ProvinceId
        END
    RETURN (@NumberOfInvoices)
END;
GO
```

Sau khi thực thi phát biểu trên, hàm udfSalesInvoicesOfProvince sẽ được thay đổi cấu trúc trong cơ sở dữ liệu AccountSystem.

Khi gọi hàm udfSalesInvoicesOfProvince vừa thay đổi cấu trúc với từ khóa DEFAULT truyền vào hàm ứng với tham số như ví dụ 8-14.

**Chương 8: Giới thiệu hàm trong SQL Server 2005**

29

**Ví dụ 8-14: Khai báo gọi hàm udfSalesInvoicesOfProvince**

```
SELECT ProvinceId, ProvinceName,
dbo.udfSalesInvoicesOfProvince(DEFAULT) As Invoices
FROM Provinces
GO
```

Khi thực thi phát biểu SELECT với hàm udfSalesInvoicesOfProvince trong ví dụ trên, bạn sẽ nhận được kết quả như hình 8-16.

|   | ProvinceId | ProvinceName     | Invoices |
|---|------------|------------------|----------|
| 1 | BDU        | Bình Dương       | 10       |
| 2 | DNA        | Đồng Nai         | 10       |
| 3 | HAN        | Hà Nội           | 10       |
| 4 | HCM        | Hồ Chí Minh      | 10       |
| 5 | TTH        | Thừa Thiên - Huế | 10       |

**Hình 8-16: Gọi hàm udfSalesInvoicesOfProvince.**

### 1.2.2. Hàm trả về đối tượng Table

Tương tự như trường hợp hàm trả về giá trị đơn, bạn có thể khai báo để thay đổi cấu trúc hàm có kiểu dữ liệu trả về là đối tượng Table như sau:

```
ALTER FUNCTION [ schema_name. ] function_name
( [ { @parameter_name [ AS ] [ type_schema_name. ]
parameter_data_type
[ = default ] }
[ ,...n ]
)
RETURNS TABLE
[ WITH <function_option> [ ,...n ] ]
[ AS ]
RETURN [ ( ) select_stmt [ ) ]
[ ; ]
```

Chẳng hạn, bạn khai báo thay đổi cấu trúc hàm có tên udfSalesInvoices không tham số và sử dụng biểu thức bảng với phát biểu WITH như ví dụ 8-15.

**Ví dụ 8-15: Khai báo thay đổi hàm trả về đối tượng Table**

```
ALTER FUNCTION udfSalesInvoices ()
RETURNS TABLE
AS
RETURN
```

M® 30

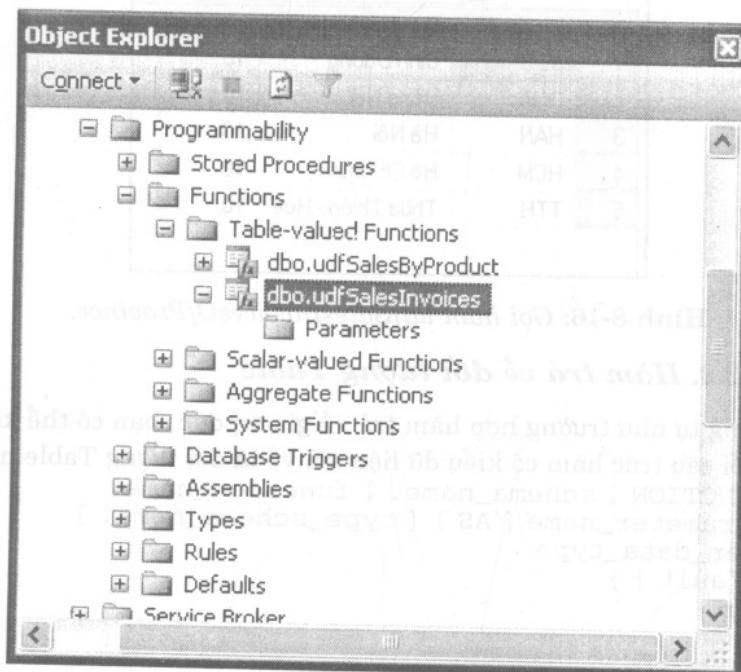
**Chương 8: Giới thiệu hàm trong SQL Server 2005**

```

SELECT InvoiceNo, DueDate, CustomerId
FROM SalesInvoices
WHERE CustomerId in
    (SELECT CustomerId FROM Customers)
)
GO

```

Sau khi thực thi phát biểu ALTER FUNCTION trong ví dụ trên, bạn có thể tìm thấy hàm vừa tạo không có tham số trong ngăn Functions như hình 8-17.



**Hình 8-17: Hàm không tham số trả về Table.**

Chú ý: Để gọi hàm có giá trị trả về là đối tượng TABLE, bạn có thể sử dụng phát biểu SELECT với mệnh đề FROM như ví dụ 8-16.

**Ví dụ 8-16: Khai báo gọi hàm udfSalesInvoices không tham số**

```

SELECT * FROM dbo.udfSalesInvoices()
GO

```

Khi thực thi phát biểu SELECT trong ví dụ trên, bạn có thể tìm thấy kết quả trình bày như hình 8-18.

## Chương 8: Giới thiệu hàm trong SQL Server 2005

31

|    | InvoiceNo  | DueDate             | CustomerId |
|----|------------|---------------------|------------|
| 1  | SI00000001 | 2007-10-10 00:00:00 | A0001      |
| 2  | SI00000002 | 2007-10-11 00:00:00 | A0002      |
| 3  | SI00000003 | 2007-10-12 00:00:00 | A0003      |
| 4  | SI00000004 | 2007-10-13 00:00:00 | A0001      |
| 5  | SI00000005 | 2007-10-14 10:00:00 | A0004      |
| 6  | SI00000006 | 2007-10-14 00:00:00 | A0005      |
| 7  | SI00000007 | 2007-10-17 00:00:00 | A0006      |
| 8  | SI00000008 | 2007-10-17 00:00:00 | A0007      |
| 9  | SI00000009 | 2007-10-18 00:00:00 | A0008      |
| 10 | SI00000010 | 2007-10-19 00:00:00 | A0001      |
| 11 | SI00000011 | 2007-10-19 00:00:00 | A0002      |
| 12 | SI00000012 | 2007-10-20 00:00:00 | A0001      |
| 13 | SI00000013 | 2007-10-20 00:00:00 | A0005      |

Hình 8-18: Kết quả gọi hàm udfSalesInvoices.

**1.2.3. Hàm với nhiều phát biểu SQL**

Bạn cũng có thể sử dụng phát biểu ALTER FUNCTION để khai báo thay đổi cấu trúc hàm với nhiều phát biểu SQL trong phần thân với cú pháp như sau:

```
ALTER FUNCTION [ schema_name. ] function_name
( [ { @parameter_name [ AS ]
      [ type_schema_name. ] parameter_data_type
      [ = default ] }
      [ ,...n ]
    ]
)
RETURNS @return_variable TABLE <table_type_definition>
[ WITH <function_option> [ ,...n ] ]
[ AS ]
BEGIN
    function_body
    RETURN
END
[ ; ]
```

Chẳng hạn, bạn có thể khai báo thay đổi hàm đã tồn tại với tên để liệt kê danh sách sản phẩm với tổng số lượng đã bán và giá bình quân cho mỗi sản phẩm với khai báo như ví dụ 8-10.

M® 32

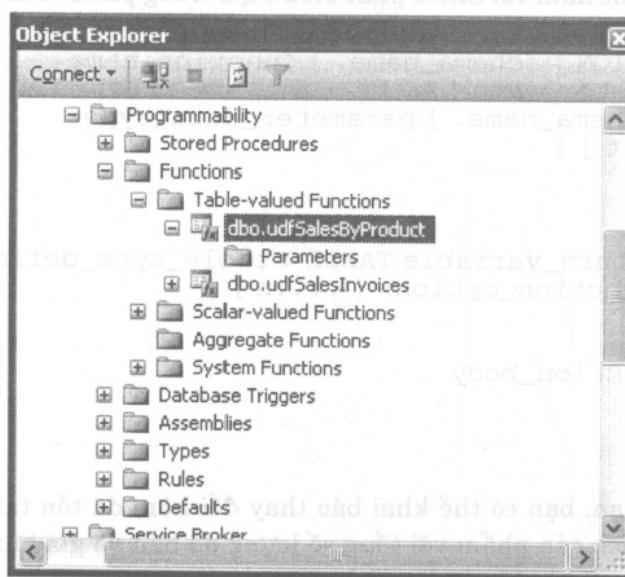
**Chương 8: Giới thiệu hàm trong SQL Server 2005****Ví dụ 8-17: Khai báo nhiều phát biểu SQL trong hàm**

```

ALTER FUNCTION udfSalesByProduct()
RETURNS @SalesByProduct TABLE
(
    -- Khai báo trả về đối tượng Table
    ProductId varchar(15) primary key NOT NULL,
    ProductName nvarchar(150),
    TotalQuantity int,
    AveragePrice int,
    TotalDiscount int
)
AS
BEGIN
    -- Khai báo thêm dữ liệu vào đối tượng Table
    INSERT @SalesByProduct
    SELECT D.ProductId, ProductNameInVietnamese,
           SUM(Quantity), AVG(Price), SUM(Discount)
    FROM Products P, SalesInvoiceDetails D
    WHERE P.ProductId = D.ProductId
    GROUP BY D.ProductId, ProductNameInVietnamese
    RETURN
END;
GO

```

Sau khi thực thi phát biểu ALTER FUNTION, bạn có thể tìm thấy hàm udfSalesByProduct không tham số trong ngăn Table-valued Functions như hình 8-19.

**Hình 8-19: Hàm đã thay đổi.**

Chương 8: Giới thiệu hàm trong SQL Server 2005

Để khai báo gọi hàm udfSalesByProduct không tham số, bạn có thể khai báo như ví dụ 8-18.

Ví dụ 8-18: Khai báo goi hàm udfSalesByProduct

```
SELECT * FROM dbo.udfSalesByProduct()  
GO
```

Bạn có thể tìm thấy danh sách sản phẩm cùng với số lượng bán khi thực thi phát biểu SELECT để gọi hàm udfSalesByProduct như hình 8-20.

|   | ProductId | ProductName                          | TotalQuantity | AveragePrice | TotalDiscount |
|---|-----------|--------------------------------------|---------------|--------------|---------------|
| 1 | P00001    | Túi xách                             | 515           | 13125        | 100000        |
| 2 | P00002    | Túi xách dùng cho học sinh nữ        | 750           | 10857        | 160000        |
| 3 | P00003    | Túi xách dùng cho học sinh nam       | 625           | 11666        | 280000        |
| 4 | P00004    | Túi áo mưa                           | 176           | 12214        | 105000        |
| 5 | P00005    | Túi xách dùng cho Máy tính           | 130           | 13000        | 90000         |
| 6 | P00006    | Túi xách dùng cho Điện thoại di động | 90            | 13500        | 70000         |

**Hình 8-20:** Gọi hàm không tham số.

### 1.3. Phát biểu xóa hàm do người sử dụng định nghĩa

Tương tự như các đối tượng cơ sở dữ liệu khác, bạn cũng có thể sử dụng phát biểu DROP FUNCTION để loại bỏ hàm ra khỏi cơ sở dữ liệu hiện hành.

Chẳng hạn, bạn khai báo để xóa hàm có tên udfSalesByProduct đang tồn tại trong cơ sở dữ liệu AccountSystem, bằng cách sử dụng cú pháp tương tự như ví dụ 8-19.

#### Ví dụ 8-19: Khai báo xóa hàm

```
DROP FUNCTION udfSalesByProduct  
GO
```

**Chú ý:** Nếu bạn xóa hàm không tồn tại trong cơ sở dữ liệu thì lỗi sẽ phát sinh tương tự như hình 8-21.

```

DROP FUNCTION udfHUUKHANGDOTCOM
GO
Msg 3701, Level 11, State 5, Line 1
Cannot drop the function 'udfHUUKHANGDOTCOM',
because it does not exist or you do not have permission.

```

Hình 8-21: Xóa hàm không tồn tại trong cơ sở dữ liệu.

## 2. HÀM HỆ THỐNG TRẢ VỀ GIÁ TRỊ ĐƠN (SCALAR FUNCTIONS)

SQL Server 2005 giới thiệu nhiều nhóm hàm mà khi bạn sử dụng nó sẽ trả về giá trị đơn như: Configuration, Cursor, Date và Time, Mathematical, Metadata, Security, String, System, Statistical, Text và Image.

### 2.1. Nhóm hàm Configuration

Nhóm hàm Configuration (cấu hình) bao gồm một số hàm (được xem như các biến toàn cục của SQL Server 2000) như: @@DATEFIRST, @@DBTS, @@LANGID, @@LANGUAGE, @@LOCK\_TIMEOUT, @@MAX\_CONNECTIONS, @@MAX\_PRECISION, @@OPTIONS, @@REMSERVER, @@SERVERNAME, @@SERVICENAME, @@SPID, @@TEXTSIZE, @@VERSION, @@NESTLEVEL cho phép bạn sử dụng để lấy thông tin về cấu hình hiện hành của SQL Server 2005.

Chẳng hạn, bạn có thể sử dụng hàm @@SERVERNAME bằng cách khai báo tương tự như ví dụ 8-20.

#### Ví dụ 8-20: Khai báo gọi hàm @@SERVERNAME

```

SELECT @@SERVERNAME AS SERVERNAME
GO

```

Khi thực thi phát biểu SELECT trong ví dụ trên, bạn có thể tìm thấy tên của máy cài đặt SQL Server 2005 như hình 8-22.

**Chương 8: Giới thiệu hàm trong SQL Server 2005**

35



|   |  | Results    | Messages |
|---|--|------------|----------|
|   |  | SERVERNAME |          |
| 1 |  | mysolution |          |

**Hình 8-22: Gọi hàm @@SERVERNAME.****2.2. Nhóm hàm Cursor**

Nhóm hàm của đối tượng CURSOR (kiểu con trỏ) bao gồm một số hàm như: @@CURSOR\_ROWS, CURSOR\_STATUS, @@FETCH\_STATUS cho phép bạn sử dụng để lấy thông tin về con trỏ hiện hành.

Chẳng hạn, bạn có thể sử dụng hàm @@FETCH\_STATUS bằng cách khai báo tương tự như ví dụ 8-21.

**Ví dụ 8-21: Khai báo gọi hàm @@FETCH\_STATUS**

```
DECLARE BalanceCursor CURSOR FOR
SELECT IssueID, ReceiptAmount, PaymentAmount
FROM dbo.MonthlyCashBalances;
OPEN BalanceCursor;
FETCH NEXT FROM BalanceCursor;
WHILE @@FETCH_STATUS = 0
BEGIN
    FETCH NEXT FROM BalanceCursor;
END;
CLOSE BalanceCursor;
DEALLOCATE BalanceCursor;
GO
```

Khi thực thi phát biểu SELECT với kiểu CURSOR trong ví dụ trên, bạn có thể tìm thấy kết quả trình bày như hình 8-23.

|  |  |  | Results | Messages      |               |
|--|--|--|---------|---------------|---------------|
|  |  |  | IssueID | ReceiptAmount | PaymentAmount |
|  |  |  |         |               |               |

**Hình 8-23: Gọi hàm @@FETCH\_STATUS.**

Chú ý: Chúng ta sẽ tìm hiểu chi tiết về các hàm trên trong chương trình bày kiểu dữ liệu CURSOR ở cuốn tiếp theo: SQL Server 2005 - Lập trình nâng cao.

### 2.3. Nhóm hàm Date và Time

Nhóm hàm Date và Time (kiểu DATETIME và SMALLDATETIME) bao gồm một số hàm như: DATEADD, DATEDIFF, DATENAME, DATEPART, DAY, MONTH, YEAR, GETDATE, GETUTCDATE cho phép bạn sử dụng để lấy thông tin về thời gian.

Chẳng hạn, bạn có thể sử dụng hàm GETDATE, DAY, MONTH, YEAR để lấy giá trị thời gian hiện hành bằng cách khai báo tương tự như ví dụ 8-22.

#### Ví dụ 8-22: Khai báo gọi hàm GETDATE, DAY, MONTH, YEAR

```
SELECT GETDATE() AS TODAY,
       DAY(GETDATE()) AS CURRENTDAY,
       MONTH(GETDATE()) AS CURRENTMONTH,
       YEAR(GETDATE()) AS CURRENTYEAR
GO
```

Khi thực thi phát biểu SELECT với các hàm trong ví dụ trên, bạn có thể tìm thấy kết quả trình bày như hình 8-24.

|   | TODAY                   | CURRENTDAY | CURRENTMONTH | CURRENTYEAR |
|---|-------------------------|------------|--------------|-------------|
| 1 | 2007-09-23 18:30:42.857 | 23         | 9            | 2007        |

Hình 8-24: Gọi hàm thời gian.

### 2.4. Nhóm hàm Mathematical

Nhóm hàm Mathematical (toán học) bao gồm một số hàm thường sử dụng như: FLOOR, ROUND, SQRT, SQUARE cho phép bạn sử dụng để lấy tính toán số học.

Chẳng hạn, bạn có thể sử dụng hàm SQRT, SQUARE để trình bày giá trị trong bảng SalesPrices như ví dụ 8-23.

#### Ví dụ 8-23: Khai báo gọi hàm SQRT, SQUARE

```
SELECT DateOfPrice, ProductId, Price,
       SQRT(Price) AS SQRTOfPrice,
       SQUARE(Price) AS SQUAREOfPrice
FROM SalesPrices
GO
```

Khi thực thi phát biểu SELECT với hai hàm SQRT, SQUARE trong ví dụ trên, bạn có thể tìm thấy kết quả trình bày như hình 8-25.

**Chương 8: Giới thiệu hàm trong SQL Server 2005**

37

The screenshot shows a SQL query results grid with four columns: DateOfPrice, ProductId, Price, and two calculated columns: SQRTOfPrice and SQUAREOfPrice. The data consists of three rows:

|   | DateOfPrice         | ProductId | Price | SQRTOfPrice      | SQUAREOfPrice |
|---|---------------------|-----------|-------|------------------|---------------|
| 1 | 2007-10-01 00:00:00 | P00001    | 10000 | 100              | 100000000     |
| 2 | 2007-10-01 00:00:00 | P00002    | 10100 | 100.498756211209 | 102010000     |
| 3 | 2007-10-01 00:00:00 | P00003    | 10100 | 100.498756211209 | 102010000     |

**Hình 8-25: Gọi hàm SQRT, SQUARE.****2.5. Nhóm hàm Metadata**

Nhóm hàm Metadata (siêu dữ liệu) bao gồm một số hàm thường sử dụng như: DB\_ID, OBJECT\_ID, COL\_NAME, DB\_NAME, OBJECT\_NAME, FILE\_ID, FILE\_NAME cho phép bạn sử dụng để lấy thông tin của cơ sở dữ liệu và đối tượng cơ sở dữ liệu.

Chẳng hạn, bạn có thể sử dụng hàm DB\_NAME để trình bày tên của cơ sở dữ liệu hiện hành và hàm OBJECT\_ID để trình bày mã của đối tượng cơ sở dữ liệu có tên là 'Customers' như ví dụ 8-24.

**Ví dụ 8-24: Khai báo gọi hàm DB\_NAME, OBJECT\_ID**

```
SELECT DB_NAME() AS [Current Database];
SELECT OBJECT_ID('Customers')
GO
```

Khi thực thi phát biểu SELECT với hai hàm DB\_NAME, OBJECT\_ID trong ví dụ trên, bạn có thể tìm thấy kết quả trình bày như hình 8-26.

The screenshot shows a SQL query results grid with two columns: Current Database and [No column name]. The data consists of two rows:

| Current Database | [No column name] |
|------------------|------------------|
| AccountSystem    | 1559676604       |

**Hình 8-26: Gọi hàm DB\_NAME, OBJECT\_ID.**

**Chú ý:** Bạn có thể liệt kê danh sách cơ sở dữ liệu bằng cách sử dụng phát biểu như ví dụ 8-25.

**Ví dụ 8-25: Khai báo liệt kê danh sách cơ sở dữ liệu**

```
SELECT database_id, name as database_name
FROM SYS.DATABASES
GO
```

Khi thực thi phát biểu SELECT với bảng sys.databases trong ví dụ trên, bạn có thể tìm thấy kết quả trình bày danh sách cơ sở dữ liệu trong SQL Server 2005 như hình 8-26-1.

|    | database_id | database_name           |
|----|-------------|-------------------------|
| 1  | 1           | master                  |
| 2  | 2           | tempdb                  |
| 3  | 3           | model                   |
| 4  | 4           | msdb                    |
| 5  | 5           | AdventureWorks          |
| 6  | 6           | AccountSystem           |
| 7  | 7           | HRSQL                   |
| 8  | 8           | RecruitVietnam          |
| 9  | 9           | AccountingSystem        |
| 10 | 10          | KHANGDB                 |
| 11 | 11          | VietnameseDatabase      |
| 12 | 12          | HumanResourceAndPayroll |
| 13 | 13          | ContactLists            |

**Hình 8-26-1: Danh sách cơ sở dữ liệu.**

Tương tự như vậy, bạn có thể liệt kê danh sách đối tượng cơ sở dữ liệu bằng cách sử dụng phát biểu như ví dụ 8-26.

**Ví dụ 8-26: Khai báo liệt kê danh sách đối tượng cơ sở dữ liệu**

```
SELECT object_id, name as object_name
FROM SYS.objects
WHERE schema_id=1 and parent_object_id=0
ORDER BY CREATE_DATE DESC
GO
```

Khi thực thi phát biểu SELECT với bảng sys.objects trong ví dụ trên, bạn có thể tìm thấy kết quả trình bày danh sách đối tượng cơ sở dữ liệu trong SQL Server 2005 như hình 8-27.

**Chương 8: Giới thiệu hàm trong SQL Server 2005**

39

|    | object_id  | object_name                   |
|----|------------|-------------------------------|
| 1  | 2039678314 | CloseMonthCashBalanceDetails  |
| 2  | 1927677915 | CloseMonthCashBalances        |
| 3  | 1559676604 | Customers                     |
| 4  | 1543676547 | vwARBAndSalesAmount           |
| 5  | 1447676205 | CloseAccountPayableDetails    |
| 6  | 1351675863 | CloseAccountReceivableDetails |
| 7  | 1303675692 | CloseAccountReceivable        |
| 8  | 1143675122 | CloseAccountPayable           |
| 9  | 1063674837 | Categories                    |
| 10 | 1172199226 | udfSalesByProduct             |
| 11 | 1076198884 | udfSalesInvoices              |
| 12 | 1044198770 | udfSalesInvoicesOfProvince    |
| 13 | 1012198656 | udfPreviousMonth              |
| 14 | 756197744  | Exports                       |
| 15 | 516196889  | ExportDetails                 |
| 16 | 420196547  | ImportDetails                 |
| 17 | 308196148  | Stocks                        |
| 18 | 180195692  | CloseInventoryControl         |
| 19 | 84195350   | MonthlyInventoryControl       |

**Hình 8-27: Danh sách đối tượng cơ sở dữ liệu.**

## 2.6. Nhóm hàm Security

Nhóm hàm Security (bảo mật) bao gồm một số hàm thường sử dụng như: CURRENT\_USER, USER\_ID, USER\_NAME, SESSION\_USER, IS\_MEMBER, SUSER\_ID, SUSER\_SNAME, SYSTEM\_USER, IS\_SRVROLEMEMBER, SCHEMA\_NAME, SCHEMA\_ID, SETUSER, PERMISSIONS cho phép bạn sử dụng để lấy thông tin của người sử dụng và nhóm quyền sử dụng.

Chẳng hạn, bạn có thể sử dụng hàm CURRENT\_USER để trình bày tài khoản đang đăng nhập và hàm USER\_NAME để trình bày tên tài khoản thứ i như ví dụ 8-27.

**Ví dụ 8-27: Khai báo gọi hàm USER\_NAME() và CURRENT\_USER**

```
SELECT USER_NAME(),
CURRENT_USER,
USER_NAME(0);
GO
```

Khi thực thi phát biểu SELECT với hai hàm USER\_NAME(), CURRENT\_USER trong ví dụ trên, bạn có thể tìm thấy kết quả trình bày như hình 8-28.

|   | (No column name) | (No column name) | (No column name) |
|---|------------------|------------------|------------------|
| 1 | dbo              | dbo              | public           |

**Hình 8-28: Gọi hàm USER\_NAME(), CURRENT\_USER.**

Chú ý: Bạn có thể sử dụng hàm CURRENT\_USER khi khai báo tạo bảng dữ liệu với giá trị mặc định ứng với cột lưu giữ tài khoản người sử dụng.

Chẳng hạn, bạn khai báo phát biểu CREATE TABLE để tạo bảng dữ liệu có tên ExchangeRate với cột UserName được khai báo giá trị mặc định là hàm CURRENT\_USER như ví dụ 8-28.

#### Ví dụ 8-28: Sử dụng hàm CURRENT\_USER

```
CREATE TABLE ExchangeRate
(
    CurrencyID      CHAR (3) NOT NULL,
    DateOfRate      SMALLDATETIME NOT NULL DEFAULT GETDATE(),
    Rate            DECIMAL DEFAULT 0 NOT NULL,
    UserName        varchar(20) NOT NULL DEFAULT CURRENT_USER
    PRIMARY KEY (CurrencyID, DateOfRate)
)
GO
```

Bằng cách sử dụng phát biểu INSERT để thêm mới dữ liệu vào bảng ExchangeRate như ví dụ 8-29.

#### Ví dụ 8-29: Khai báo thêm dữ liệu vào bảng ExchangeRate

```
INSERT INTO ExchangeRate
(CurrencyID, DateOfRate, Rate)
VALUES ('USD', GETDATE(), 16200)
GO
```

Sau khi thêm thành công, bạn có thể liệt kê cột dữ liệu UserName bằng cách sử dụng phát biểu SELECT và kết quả trình bày như hình 8-29.

|   | user_NAME() | CURRENT_USER | CURRENT_USER_ID |
|---|-------------|--------------|-----------------|
| 1 | sa          | sa           | 1               |

**Chương 8:** Giới thiệu hàm trong SQL Server 2005

41

| CurrencyID | DateOfRate | Rate                | UserName |
|------------|------------|---------------------|----------|
| 1          | USD        | 2007-09-24 13:46:00 | dbo      |

**Hình 8-29:** Sử dụng hàm CURRENT\_USER.

Bạn có thể liệt kê danh sách người sử dụng trong cơ sở dữ liệu bằng cách sử dụng phát biểu như ví dụ 8-30.

**Ví dụ 8-30: Khai báo liệt kê danh sách người sử dụng**

```
SELECT uid, name FROM sys.sysusers
WHERE islogin = 1
GO
```

Khi thực thi phát biểu SELECT với bảng sys.sysusers trong ví dụ trên, bạn có thể tìm thấy kết quả trình bày danh sách người sử dụng trong SQL Server 2005 như hình 8-30.

| uid | name                     |
|-----|--------------------------|
| 1   | dbo                      |
| 2   | guest                    |
| 3   | INFORMATION_SCHEMA       |
| 4   | sys                      |
| 5   | MYSOLUTION\Administrator |
| 6   | khangdbuser              |
| 7   | PurchasingUser           |

**Hình 8-30:** Danh sách người sử dụng.**2.7. Nhóm hàm String**

Nhóm hàm String (chuỗi ứng với char hay varchar) bao gồm một số hàm thường sử dụng như: CHAR, NCHAR, LEFT, RIGHT, LEN, LTRIM, RTRIM, STR, SUBSTRING, UPPER, LOWER, PADINDEX cho phép bạn sử dụng để xử lý chuỗi.

Chẳng hạn, bạn có thể sử dụng hàm CHAR để trình bày dấu nháy đơn trong phát biểu SQL động như ví dụ 8-31.

#### Ví dụ 8-31: Khai báo gọi hàm CHAR

```
DECLARE @query NVARCHAR(500);
SET @query = N'SELECT CustomerId, ContactName,
SET @query = @query + N'ContactTitle'
SET @query = @query + N'FROM Customers WHERE '
SET @query = @query + N'ProvinceId=' + CHAR(39)
SET @query = @query + N'HCM' + CHAR(39)
EXEC sp_executesql @query;
GO
```

Khi thực thi phát biểu SELECT trong ví dụ trên, bạn có thể tìm thấy kết quả trình bày như hình 8-31.

**Hình 8-31: Gọi hàm CHAR.**

Trong trường hợp bạn muốn cắt hay loại bỏ khoảng trắng trong chuỗi, bạn có thể sử dụng hàm LTRIM, RTRIM hay RIGHT và LEFT tương tự như ví dụ 8-32.

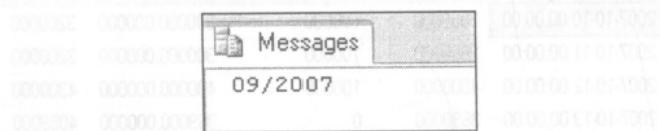
#### Ví dụ 8-32: Khai báo sử dụng hàm LTRIM, LEFT và RIGHT

```
DECLARE @CurrentMonthYear char(7)
SET @CurrentMonthYear = '10/2007'
DECLARE @Month TINYINT
DECLARE @Year SMALLINT
SET @Month = CAST(LEFT(@CurrentMonthYear, 2) AS TINYINT)
SET @Year = CAST(RIGHT(@CurrentMonthYear, 4) AS SMALLINT)
IF (@Month != 1 )
    SET @Month = @Month - 1
ELSE
    BEGIN
        SET @Month = 12
        SET @Year = @Year - 1
    END
PRINT right('0'+ ltrim(@Month),2) +'/' + ltrim(@Year)
GO
```

**Chương 8:** Giới thiệu hàm trong SQL Server 2005

43

Khi thực thi phát biểu SELECT trong ví dụ trên, bạn có thể tìm thấy kết quả trình bày như hình 8-32.



**Hình 8-32:** Gọi hàm LTRIM, LEFT và RIGHT.

## 2.8. Nhóm hàm System

Tương tự như nhóm hàm Configuration, nhóm hàm System (hệ thống) bao gồm một số hàm thường sử dụng như: APP\_NAME, CAST, CONVERT, HOST\_NAME, IDENTITY, ISDATE, ISNULL, ISNUMERIC, NEWID, @@ROWCOUNT, NULLIF cho phép bạn sử dụng để thực hiện một số ý định như: Đếm số hàng dữ liệu, kiểm tra số, tạo số tự động.

Chẳng hạn, bạn khai báo phát biểu SELECT để trình bày danh sách ngày xuất hóa đơn và số tiền bán hàng như ví dụ 8-33.

### Ví dụ 8-33: Khai báo doanh thu bán hàng

```
WITH Sales
AS
(
    SELECT DueDate AS SalesDate,
           SUM(Quantity*Price) AS SalesAmount,
           SUM(Discount) AS DiscountAmount,
           SUM(Quantity*Price*VATRate/100) AS VATAmount
      FROM SalesInvoices S, SalesInvoiceDetails D
     WHERE S.InvoiceNo = D.InvoiceNo
   GROUP BY DueDate
)
SELECT SalesDate, SalesAmount, DiscountAmount, VATAmount,
       SalesAmount + VATAmount - DiscountAmount AS SalesAmount
  FROM Sales
   GO
```

Khi thực thi phát biểu SELECT với biểu thức bảng trong ví dụ trên, kết quả trình bày như hình 8-33.

|    | SalesDate           | SalesAmount | DiscountAmount | VATAmount     | SalesAmount |
|----|---------------------|-------------|----------------|---------------|-------------|
| 1  | 2007-10-10 00:00:00 | 3000000     | 100000         | 300000.000000 | 3200000     |
| 2  | 2007-10-11 00:00:00 | 3000000     | 100000         | 300000.000000 | 3200000     |
| 3  | 2007-10-12 00:00:00 | 4000000     | 100000         | 400000.000000 | 4300000     |
| 4  | 2007-10-13 00:00:00 | 3690000     | 0              | 369000.000000 | 4059000     |
| 5  | 2007-10-14 00:00:00 | 2062500     | 0              | 206250.000000 | 2268750     |
| 6  | 2007-10-14 10:00:00 | 5250000     | 105000         | 525000.000000 | 5670000     |
| 7  | 2007-10-17 00:00:00 | 2995000     | 130000         | 299500.000000 | 3164500     |
| 8  | 2007-10-18 00:00:00 | 687500      | 30000          | 68750.000000  | 726250      |
| 9  | 2007-10-19 00:00:00 | 1947500     | 90000          | 194750.000000 | 2052250     |
| 10 | 2007-10-20 00:00:00 | 524000      | 150000         | 52400.000000  | 426400      |

Hình 8-33: Doanh thu bán hàng.

Nhìn vào hình trên, cột SalesDate sẽ trình bày thời gian bao gồm giờ, phút và giây, trong trường hợp trình bày chuỗi ngày tháng và năm, bạn sử dụng hàm CONVERT như ví dụ 8-34.

#### Ví dụ 8-34: Khai báo gọi hàm CONVERT

```
WITH Sales
AS
(
    SELECT Convert(char(10), DueDate, 103) AS SalesDate,
           SUM(Quantity*Price) AS SalesAmount,
           SUM(Discount) AS DiscountAmount,
           SUM(Quantity*Price*VATRate/100) AS VATAmount
      FROM SalesInvoices S, SalesInvoiceDetails D
     WHERE S.InvoiceNo = D.InvoiceNo
   GROUP BY Convert(char(10), DueDate, 103)
)
SELECT SalesDate, SalesAmount, VATAmount, DiscountAmount, VATAmount,
       SalesAmount + VATAmount - DiscountAmount AS SalesAmount
  FROM Sales
   GO
```

Khi thực thi phát biểu SELECT trong ví dụ trên, bạn có thể tìm thấy kết quả trình bày như hình 8-34.

**Chương 8: Giới thiệu hàm trong SQL Server 2005**45 

|   | SalesDate  | SalesAmount | DiscountAmount | VATAmount     | SalesAmount |
|---|------------|-------------|----------------|---------------|-------------|
| 1 | 10/10/2007 | 3000000     | 100000         | 300000.000000 | 3200000     |
| 2 | 11/10/2007 | 3000000     | 100000         | 300000.000000 | 3200000     |
| 3 | 12/10/2007 | 4000000     | 100000         | 400000.000000 | 4300000     |
| 4 | 13/10/2007 | 3690000     | 0              | 369000.000000 | 4059000     |
| 5 | 14/10/2007 | 7312500     | 105000         | 731250.000000 | 7938750     |
| 6 | 17/10/2007 | 2995000     | 130000         | 299500.000000 | 3164500     |
| 7 | 18/10/2007 | 687500      | 30000          | 68750.000000  | 726250      |
| 8 | 19/10/2007 | 1947500     | 90000          | 194750.000000 | 2052250     |
| 9 | 20/10/2007 | 524000      | 150000         | 52400.000000  | 426400      |

**Hình 8-34: Gọi hàm CONVERT.**

Chú ý: Hàm CONVERT cho phép chúng ta chuyển đổi kiểu dữ liệu này sang kiểu dữ liệu khác. Tuy nhiên, có một số kiểu dữ liệu không thể chuyển sang kiểu dữ liệu khác.

CONVERT ( data\_type [ ( length ) ] , expression [ , style ] )

Trong cột VATAmount, bạn có thể nhìn thấy nhiều số lẻ thập phân, nếu muốn định dạng kiểu số nguyên thì bạn cũng có thể sử dụng hàm CAST như ví dụ 8-35.

**Ví dụ 8-35: Khai báo sử dụng hàm CAST**

```
WITH Sales AS
(
    SELECT Convert(char(10), DueDate, 103) AS SalesDate,
           SUM(Quantity*Price) AS SalesAmount,
           SUM(Discount) AS DiscountAmount,
           SUM(Quantity*Price*VATRate/100) AS VATAmount
      FROM SalesInvoices S, SalesInvoiceDetails D
     WHERE S.InvoiceNo = D.InvoiceNo
   GROUP BY Convert(char(10), DueDate, 103)
)
SELECT SalesDate, SalesAmount, DiscountAmount,
       CAST(VATAmount AS INT) AS VATAmount,
       SalesAmount + VATAmount - DiscountAmount AS SalesAmount
  FROM Sales
 GO
```

Khi thực thi phát biểu SELECT trong ví dụ trên, bạn có thể tìm thấy kết quả trình bày như hình 8-35.

|   | SalesDate  | SalesAmount | DiscountAmount | VATAmount | SalesAmount |
|---|------------|-------------|----------------|-----------|-------------|
| 1 | 10/10/2007 | 3000000     | 100000         | 300000    | 3200000     |
| 2 | 11/10/2007 | 3000000     | 100000         | 300000    | 3200000     |
| 3 | 12/10/2007 | 4000000     | 100000         | 400000    | 4300000     |
| 4 | 13/10/2007 | 3690000     | 0              | 369000    | 4059000     |
| 5 | 14/10/2007 | 7312500     | 105000         | 731250    | 7938750     |
| 6 | 17/10/2007 | 2995000     | 130000         | 299500    | 3164500     |
| 7 | 18/10/2007 | 687500      | 30000          | 68750     | 726250      |
| 8 | 19/10/2007 | 1947500     | 90000          | 194750    | 2052250     |
| 9 | 20/10/2007 | 524000      | 150000         | 52400     | 426400      |

**Hình 8-35:** Sử dụng hàm CAST.

Chú ý: Tương tự như hàm CONVERT, hàm CAST cho phép chúng ta chuyển đổi kiểu dữ liệu này sang kiểu dữ liệu khác.  
CAST ( expression AS data\_type [ (length) ] )

Nếu bảng dữ liệu có cột kiểu số nguyên là số tự động tăng, bạn có thể sử dụng hàm @@IDENTITY để lấy ra số tự động tăng vừa phát sinh sau khi mẫu tin vừa thêm.

Chẳng hạn, bạn khai báo phát biểu INSERT để thêm mẫu tin vào bảng Categories và lấy ra số tự động tăng vừa tạo ra như ví dụ 8-36.

#### Ví dụ 8-36: Khai báo sử dụng hàm @@IDENTITY

```
INSERT INTO Categories(CategoryNameInVietnamese)
VALUES (N'Túi xách học sinh nam')
SELECT @@IDENTITY
GO
```

Khi thực thi phát biểu INSERT trong ví dụ trên, mẫu tin mới thêm vào sẽ có giá trị tại cột CategoryId là 5 như hình 8-36.

|                  |   |
|------------------|---|
| (No column name) | 5 |
| 1                |   |

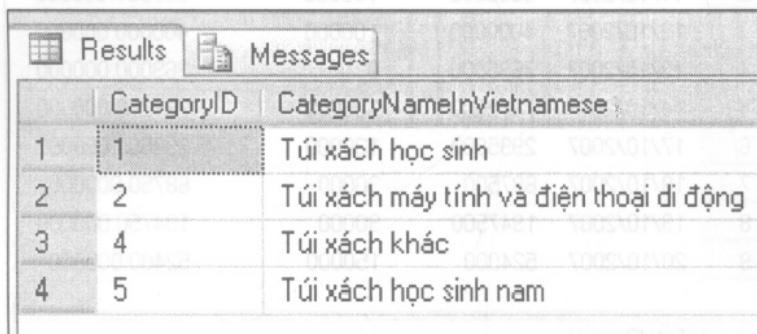
**Hình 8-36:** Số tự động tăng.

Bạn có thể kiểm tra dữ liệu trong bảng Categories bằng cách khai báo phát biểu SELECT như ví dụ 8-37.

**Chương 8: Giới thiệu hàm trong SQL Server 2005**47 **Ví dụ 8-37: Khai báo kiểm tra dữ liệu**

```
SELECT * FROM Categories
GO
```

Mẫu tin vừa thêm vào có giá trị tại cột CategoryId là 5 như hình 8-37.



|   | CategoryID | CategoryNameInVietnamese                |
|---|------------|---|
| 1 | 1          | Túi xách học sinh                       |
| 2 | 2          | Túi xách máy tính và điện thoại di động |
| 3 | 4          | Túi xách khác                           |
| 4 | 5          | Túi xách học sinh nam                   |

**Hình 8-37: Mẫu tin mới thêm vào.**

Để biết được số lượng mẫu tin vừa truy vấn, bạn có thể sử dụng hàm @@ROWCOUNT.

Trong một vài trường hợp làm việc với phát biểu SELECT, bạn nên sử dụng hàm này tương tự như ví dụ 8-38.

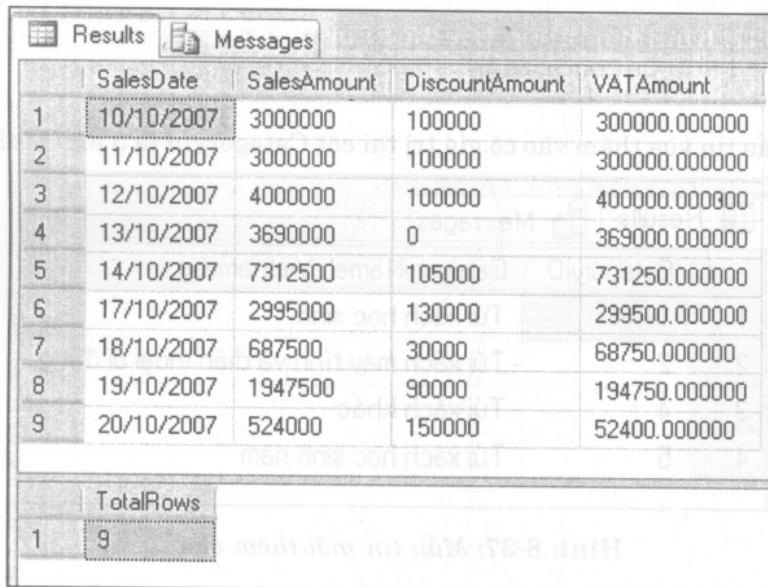
**Ví dụ 8-38: Khai báo sử dụng hàm @@ROWCOUNT**

```
SELECT Convert(char(10), DueDate, 103) AS SalesDate,
SUM(Quantity*Price) AS SalesAmount,
SUM(Discount) AS DiscountAmount,
SUM(Quantity*Price*VATRate/100) AS VATAmount
FROM SalesInvoices S, SalesInvoiceDetails D
WHERE S.InvoiceNo = D.InvoiceNo
GROUP BY Convert(char(10), DueDate, 103)
SELECT @@ROWCOUNT AS TotalRows
GO
```

Khi thực thi phát biểu SELECT với mệnh đề GROUP BY trong ví dụ trên, bạn có thể tìm thấy kết quả trình bày như hình 8-38.

Trong đó, số 9 là tổng số mẫu tin mà phát biểu SELECT trước đó thực thi.

Nếu bạn có nhu cầu tạo ra giá trị duy nhất có kiểu uniqueidentifier thì sử dụng hàm NEWID.



The screenshot shows the SQL Server Management Studio interface with the 'Results' tab selected. It displays a table with four columns: SalesDate, SalesAmount, DiscountAmount, and VATAmount. Below the table, a single row labeled 'TotalRows' contains the value '9'.

|   | SalesDate  | SalesAmount | DiscountAmount | VATAmount     |
|---|------------|-------------|----------------|---------------|
| 1 | 10/10/2007 | 3000000     | 100000         | 300000.000000 |
| 2 | 11/10/2007 | 3000000     | 100000         | 300000.000000 |
| 3 | 12/10/2007 | 4000000     | 100000         | 400000.000000 |
| 4 | 13/10/2007 | 3690000     | 0              | 369000.000000 |
| 5 | 14/10/2007 | 7312500     | 105000         | 731250.000000 |
| 6 | 17/10/2007 | 2995000     | 130000         | 299500.000000 |
| 7 | 18/10/2007 | 687500      | 30000          | 68750.000000  |
| 8 | 19/10/2007 | 1947500     | 90000          | 194750.000000 |
| 9 | 20/10/2007 | 524000      | 150000         | 52400.000000  |

| TotalRows |
|-----------|
| 1 9       |

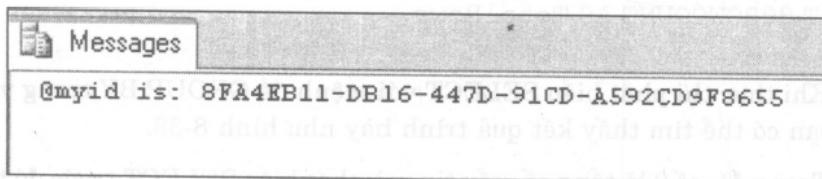
**Hình 8-38:** Sử dụng hàm @@ROWCOUNT.

Chẳng hạn, bạn có thể tạo ra giá trị kiểu uniqueidentifier bằng cách khai báo như ví dụ 8-39.

#### Ví dụ 8-39: Khai báo tạo giá trị duy nhất kiểu uniqueidentifier

```
DECLARE @myid uniqueidentifier
SET @myid = NEWID()
PRINT '@myid is: ' + CONVERT(varchar(255), @myid)
GO
```

Khi thực thi tập lệnh trên, bạn có thể tìm thấy giá trị vừa tạo tương tự như hình 8-39.



The screenshot shows the SQL Server Management Studio interface with the 'Messages' tab selected. It displays a single message line: '@myid is: 8FA4EB11-DB16-447D-91CD-A592CD9F8655'.

**Hình 8-39:** Tạo giá trị duy nhất.

Ngoài cách tạo ra giá trị duy nhất có kiểu uniqueidentifier, bạn có thể sử dụng hàm NEWID để lấy ra số mẫu tin một cách ngẫu nhiên.

Chẳng hạn, để lấy ra ngẫu nhiên 5 sản phẩm bán trong tháng, bạn có thể khai báo như ví dụ 8-40.

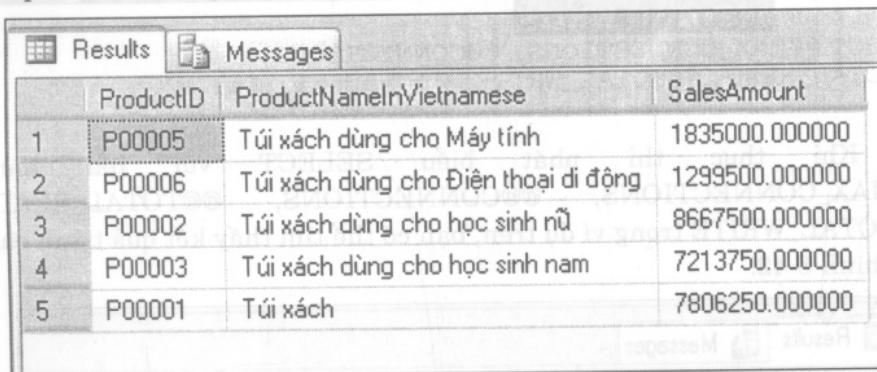
## Chương 8: Giới thiệu hàm trong SQL Server 2005

49 M®

**Ví dụ 8-40: Khai báo hàm NEWID**

```
SELECT TOP(5) P.ProductID, ProductNameInVietnamese,
SUM(Quantity*Price*(1+VATRate/100)-Discount) As SalesAmount
FROM Products P, SalesInvoiceDetails D
WHERE P.ProductID = D.ProductID
GROUP BY P.ProductID, ProductNameInVietnamese
ORDER BY NEWID()
GO
```

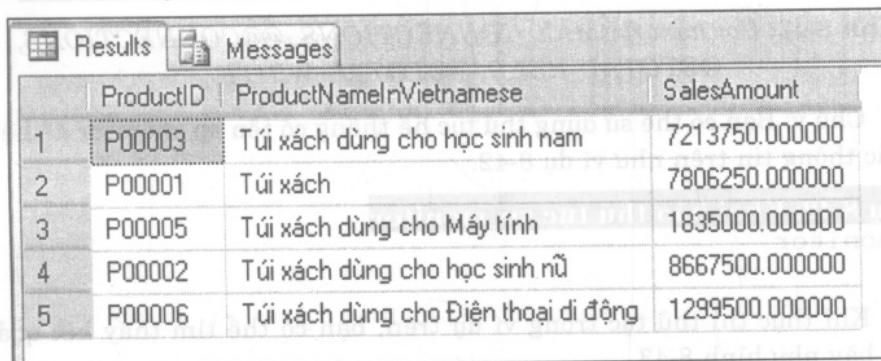
Nếu bạn thực thi phát biểu SELECT trong ví dụ trên lần thứ nhất, kết quả sẽ trình bày như hình 8-40.



|   | ProductID | ProductNameInVietnamese              | SalesAmount    |
|---|-----------|--------------------------------------|----------------|
| 1 | P00005    | Túi xách dùng cho Máy tính           | 1835000.000000 |
| 2 | P00006    | Túi xách dùng cho Điện thoại di động | 1299500.000000 |
| 3 | P00002    | Túi xách dùng cho học sinh nữ        | 8667500.000000 |
| 4 | P00003    | Túi xách dùng cho học sinh nam       | 7213750.000000 |
| 5 | P00001    | Túi xách                             | 7806250.000000 |

**Hình 8-40:** Lấy ngẫu nhiên ra 5 mẫu tin.

Nếu bạn thực thi phát biểu SELECT trong ví dụ trên lần thứ hai, kết quả sẽ trình bày như hình 8-41.



|   | ProductID | ProductNameInVietnamese              | SalesAmount    |
|---|-----------|--------------------------------------|----------------|
| 1 | P00003    | Túi xách dùng cho học sinh nam       | 7213750.000000 |
| 2 | P00001    | Túi xách                             | 7806250.000000 |
| 3 | P00005    | Túi xách dùng cho Máy tính           | 1835000.000000 |
| 4 | P00002    | Túi xách dùng cho học sinh nữ        | 8667500.000000 |
| 5 | P00006    | Túi xách dùng cho Điện thoại di động | 1299500.000000 |

**Hình 8-41:** Lấy ngẫu nhiên ra 5 mẫu tin.

## 2.9. Nhóm hàm Statistical

Nhóm hàm Statistical (thống kê) bao gồm một số hàm thường sử dụng như: @@CONNECTIONS, @@CPU\_BUSY, @@IO\_BUSY, @@TOTAL\_READ, @@TOTAL\_WRITE cho phép bạn sử dụng để lấy thông tin về hệ thống.

Chẳng hạn, bạn có thể sử dụng hàm @@CONNECTIONS, @@TOTAL\_READ, @@TOTAL\_WRITE để trình bày số lượng kết nối như ví dụ 8-41.

### Ví dụ 8-41: Khai báo gọi hàm @@CONNECTIONS, @@TOTAL\_READ, @@TOTAL\_WRITE

```
SELECT @@MAX_CONNECTIONS, @@CONNECTIONS,
@@TOTAL_READ, @@TOTAL_WRITE
GO
```

Khi thực thi phát biểu SELECT với hai hàm @@MAX\_CONNECTIONS, @@CONNECTIONS, @@TOTAL\_READ, @@TOTAL\_WRITE trong ví dụ trên, bạn có thể tìm thấy kết quả trình bày như hình 8-42.

|   | [No column name] | [No column name] | [No column name] | [No column name] |
|---|------------------|------------------|------------------|------------------|
| 1 | 32767            | 2733             | 1050             | 187              |

Hình 8-42: Gọi hàm @@MAX\_CONNECTIONS, @@CONNECTIONS, @@TOTAL\_READ, @@TOTAL\_WRITE.

Chú ý: Bạn có thể sử dụng thủ tục hệ thống có tên sp\_monitor để liệt kê các thông tin trên như ví dụ 8-42.

### Ví dụ 8-42: Khai báo thủ tục sp\_monitor

```
sp_monitor
GO
```

Khi thực thi thủ tục trong ví dụ trên, bạn có thể tìm thấy kết quả trình bày như hình 8-43.

## Chương 8: Giới thiệu hàm trong SQL Server 2005

51



|   |            | Results                 | Messages                |
|---|------------|-------------------------|-------------------------|
|   |            | last_run                | current_run             |
| 1 |            | 2007-09-24 17:59:35.950 | 2007-09-24 18:15:45.030 |
|   |            | seconds                 | 970                     |
|   |            | cpu_busy                | idle                    |
| 1 | 266(94)-9% | 58(19)-1%               | 11223(824)-84%          |
|   |            | packets_received        | packets_sent            |
| 1 | 2839(214)  | 2836(214)               | 0(0)                    |
|   |            | packet_errors           |                         |
|   |            | total_read              | total_write             |
| 1 | 1054(9)    | 187(1)                  | 0(0)                    |
|   |            | connections             |                         |
| 1 | 2759(223)  |                         |                         |

Hình 8-43: Gọi thủ tục sp\_monitor.

### 3. HÀM TRẢ VỀ ĐỐI TƯỢNG LÀ TẬP MẪU TIN (ROWSET)

Ngoài hàm trả về giá trị đơn vừa trình bày ở phần trên, nếu bạn có nhu cầu lặp ra tập dữ liệu thì sử dụng một số hàm như: OPENQUERY, OPENROWSET, OPENDATASOURCE, OPENXML.

Ngoài ra, các hàm này cho phép bạn đang kết nối cơ sở dữ liệu này mà có thể truy vấn dữ liệu từ cơ sở dữ liệu khác.

#### 3.1. Hàm OPENQUERY

Hàm OPENQUERY cho phép thực thi phát biểu SQL thông qua liên kết SQL Server thiết lập trong phần Linked Servers.

```
OPENQUERY ( linked_server , 'query' )
```

Để thực hiện việc truy vấn dữ liệu bằng hàm OPENQUERY, trước tiên bạn tạo liên kết đến cơ sở dữ liệu SQL Server 2005 thứ hai bằng thủ tục hệ thống hay MS, ví dụ Server cài đặt cơ sở dữ liệu SQL Server 2005 có tên KHANGCOMPUTER.

Nếu sử dụng thủ tục hệ thống sp\_addlinkedserver, bạn khai báo như ví dụ 8-43.

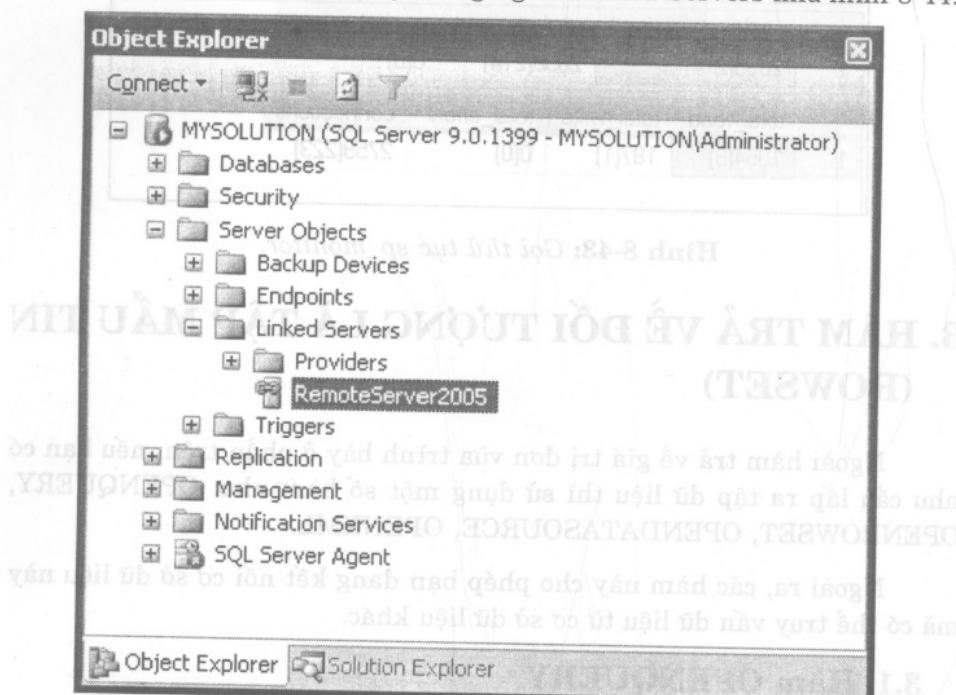
#### Ví dụ 8-43: Khai báo thực thi thủ tục sp\_addlinkedserver

```
EXEC sp_addlinkedserver
    @server = 'RemoteServer2005',
    @srvproduct = '' ,
```

```

@provider = 'SQLNCLI',
@provstr = 'SERVER=khangcomputer,
            Database=RecruitVietnam;
            UID=sa;PWD=sa'
GO
    
```

Sau khi thực thi phát biểu trong ví dụ thành công, bạn có thể tìm thấy SQL Server 2005 đăng ký trong ngăn Linked Servers như hình 8-44.



**Hình 8-44:** Tạo liên kết Server.

Trong trường hợp muốn tạo liên kết đến SQL Server 2000, bạn có thể khai báo thực thi thủ tục `sp_addlinkedserver` như ví dụ 8-44.

#### Ví dụ 8-44: Khai báo thực thi thủ tục `sp_addlinkedserver`

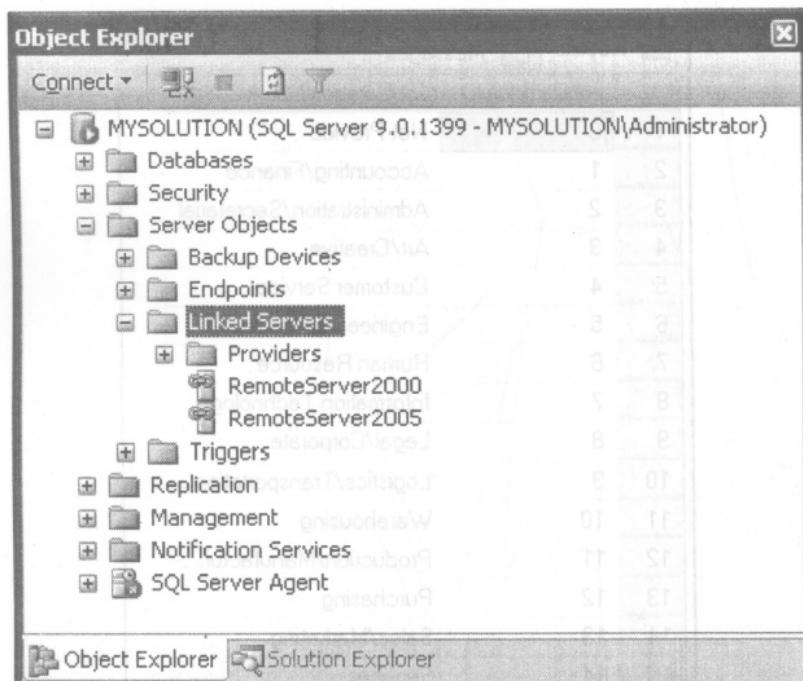
```

EXEC sp_addlinkedserver
    @server = 'RemoteServer2000',
    @srvproduct = '',
    @provider = 'SQLOLEDB',
    @provstr = 'SERVER=SQLServer2000,
                Database=RecruitVietnam;UID=sa;PWD=sa'
GO
    
```

Sau khi thực thi thành công ví dụ trên, bạn có thể tìm thấy SQL Server 2000 đăng ký trong ngăn Linked Servers như hình 8-45.

## Chương 8: Giới thiệu hàm trong SQL Server 2005

53



Hình 8-45: Đăng ký SQL Server 2000.

Sau khi bạn đăng ký SQL Server 2005 thứ hai vào SQL Server 2005 đang làm việc, bạn có thể sử dụng hàm OPENQUERY như ví dụ 8-45.

**Ví dụ 8-45: Khai báo hàm OPENQUERY**

```
SELECT * FROM
OPENQUERY(RemoteServer2005,
'SELECT * FROM dbo.tblJobCategories')
GO
```

Nếu thực thi phát biểu SELECT với hàm OPENQUERY trong ví dụ trên, bạn có thể tìm thấy kết quả trình bày là danh sách mẫu tin trong bảng tblJobCategories như hình 8-46.

**Chú ý:** Bạn có thể tìm thấy bảng dữ liệu có tên tblJobCategories trong cơ sở dữ liệu RecruitVietnam đính kèm theo đĩa.

Để khắc phục lỗi này, bạn có thể chọn vào Start | All Programs | Microsoft SQL Server 2005 | Configuration Tools | SQL Server Browser. Área Configuration cửa sổ xuất hiện như hình 8-47.



|    | JobCategoryID | JobCategoryName            |
|----|---------------|----------------------------|
| 1  | 0             | Not Provide                |
| 2  | 1             | Accounting/Finance         |
| 3  | 2             | Administration/Secretarial |
| 4  | 3             | Art/Creative               |
| 5  | 4             | Customer Service           |
| 6  | 5             | Engineering/Technical      |
| 7  | 6             | Human Resource             |
| 8  | 7             | Information Technology     |
| 9  | 8             | Legal/Corporate            |
| 10 | 9             | Logistics/Transportation   |
| 11 | 10            | Warehousing                |
| 12 | 11            | Production/Manufacturing   |
| 13 | 12            | Purchasing                 |
| 14 | 13            | Sales/Marketing            |
| 15 | 14            | Services                   |
| 16 | 15            | Others                     |

**Hình 8-46:** Gọi hàm OPENQUERY.

Trong trường hợp bạn liên kết đến SQL Server 2005 phiên bản SQLEXPRESS, sau khi đăng ký thành công bạn gọi phát biểu SELECT với hàm OPENQUERY như ví dụ trên, lỗi sẽ phát sinh như sau do phiên bản này mặc định không cho phép người sử dụng truy cập dữ liệu từ xa.

May khac am khong the phap nguoi su dung truy cap OLE DB provider "SQLNCLI11" for linked server

"RemoteServer2005" returned message "Login timeout expired"

OLE DB provider "SQLNCLI" for linked server

"RemoteServer2005"

returned message "An error has occurred while establishing a connection to the server. When connecting to SQL Server 2005.

this failure may be caused by the fact that under the default settings SQL Server does not allow remote connections.".

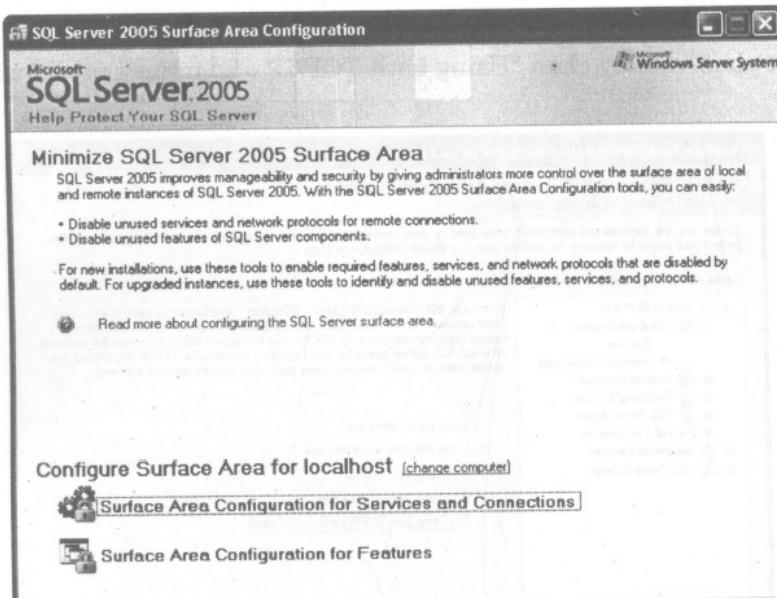
Msg 53, Level 16, State 1, Line 0  
Named Pipes Read

Named Pipes Provider: Could not open a connection to SQL Server [53].

Dashed and dotted lines indicate the boundaries of the two clusters.

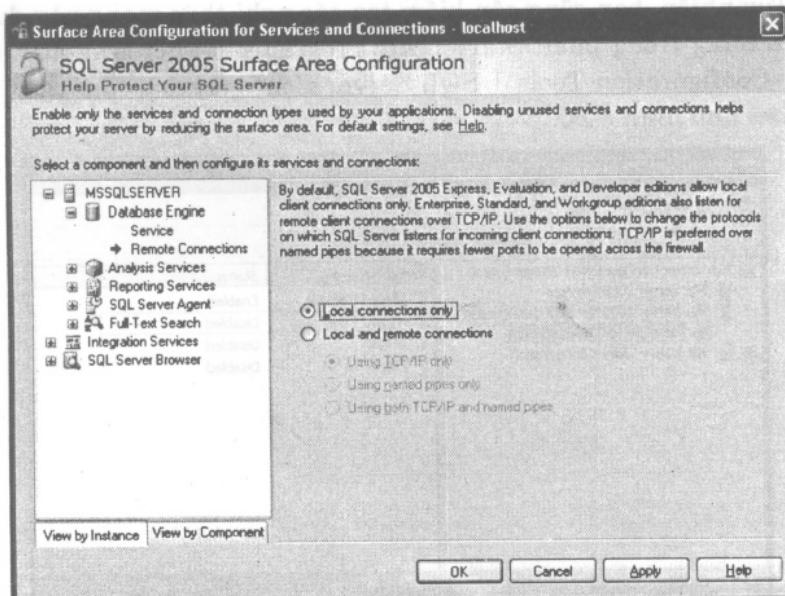
Để khắc phục lỗi trên, bạn có thể chọn vào Start | All Programs Microsoft SQL Server 2005 | Configuration Tools | SQL Server Surface Area Configuration cửa sổ xuất hiện như hình 8.47.

## Chương 8: Giới thiệu hàm trong SQL Server 2005

55 

Hình 8-47: Cửa sổ SQL Server Surface Area Configuration.

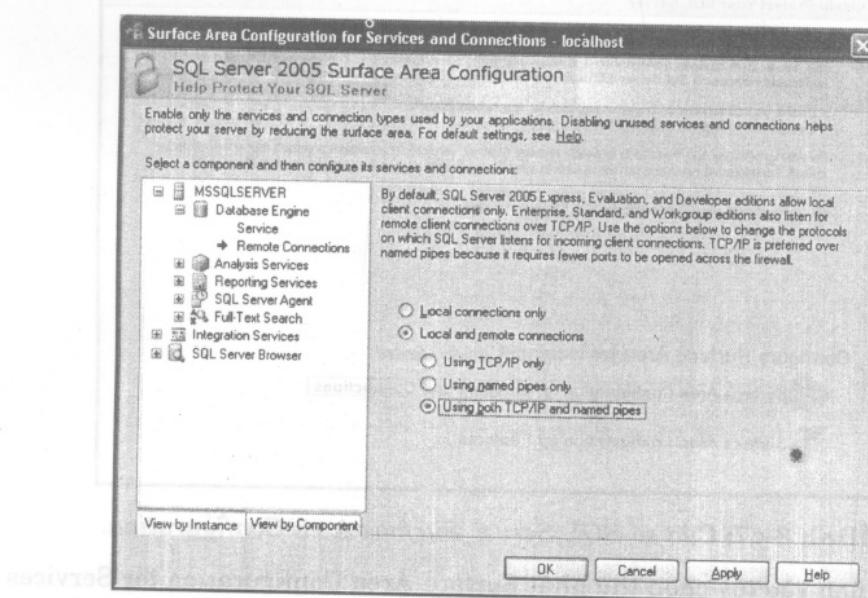
Chọn vào tùy chọn thứ nhất Surface Area Configuration for Services and Connections rồi tiếp tục chọn vào Remote Connections, cửa sổ xuất hiện như hình 8-48.



Hình 8-48: Chọn Remote Connections.

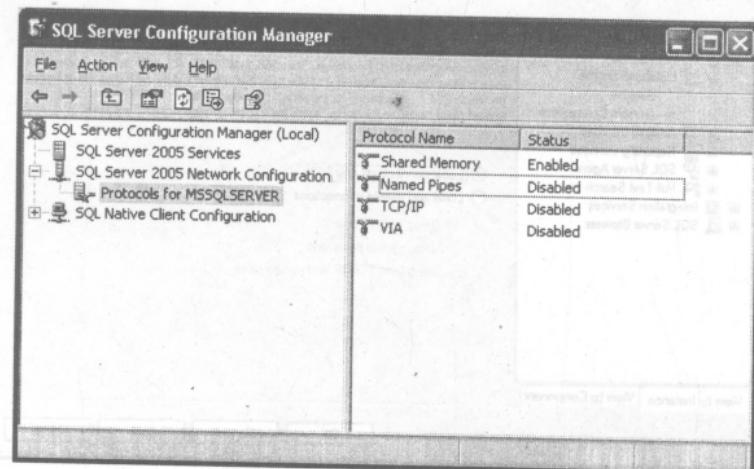
**Chương 8: Giới thiệu hàm trong SQL Server 2005**

Bạn có thể chọn vào tùy chọn “Local and remote connections”, rồi tiếp tục chọn vào tùy chọn “Using both TCP/IP and named pipes” như hình 8-49.



**Hình 8-49:** Cho phép truy cập từ xa.

Tuy nhiên, bạn cũng cần kiểm tra các nghi thức mạng này được cho phép sử dụng trong phần Start | All Programs | Microsoft SQL Server 2005 | Configuration Tools | SQL Server Configuration Manager, cửa sổ xuất hiện như hình 8-50.



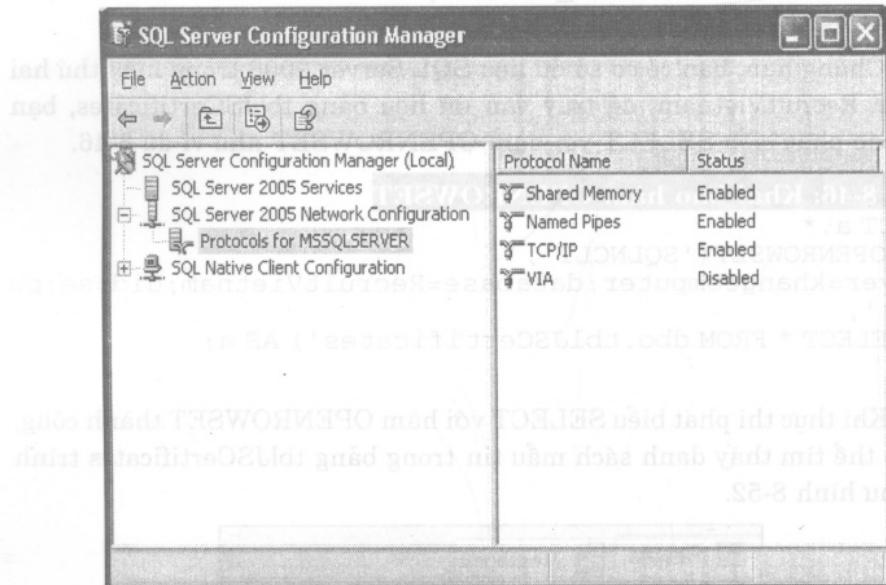
**Hình 8-50:** Cho phép sử dụng nghi thức mạng.

**Chương 8: Giới thiệu hàm trong SQL Server 2005**

57



Bằng cách chuyển trạng thái từ Disabled sang Enabled cho hai thức mạng là TCP/IP và Named Pipes, cửa sổ xuất hiện như hình 8-51.



**Hình 8-51:** Cho phép sử dụng nghi thức mạng.

Lưu ý: Để tìm hiểu thêm về cách liên kết đến với nhiều loại cơ sở dữ liệu khác nhau từ cơ sở dữ liệu SQL Server 2005, bạn có thể tham khảo chương 23 trong tập kế tiếp.

### 3.2. Hàm OPENROWSET

Để sử dụng hàm OPENROWSET, bạn khai báo cú pháp tương tự như sau:

```

OPENROWSET
( { 'provider_name' , { 'datasource' ; 'user_id' ;
'password'
| 'provider_string' }
, { [ catalog. ] [ schema. ] object
| 'query'
}
| BULK 'data_file' , { FORMATFILE = 'format_file_path' [ <bulk_options> ]
| SINGLE_BLOB | SINGLE_CLOB | SINGLE_NCLOB }
} )
<bulk_options> ::=
[ , CODEPAGE = { 'ACP' | 'OEM' | 'RAW' | 'code_page' } ]
[ , ERRORFILE = 'file_name' ]

```

```
[ , FIRSTROW = first_row ]
[ , LASTROW = last_row ]
[ , MAXERRORS = maximum_errors ]
[ , ROWS_PER_BATCH = rows_per_batch ]
```

Chẳng hạn, bạn có cơ sở dữ liệu SQL Server 2005 trong máy thứ hai với tên RecruitVietnam, để truy vấn dữ liệu bảng tblJSCertificates, bạn khai báo phát biểu SELECT với hàm OPENROWSET như ví dụ 8-46.

#### Ví dụ 8-46: Khai báo hàm OPENROWSET

```
SELECT a.*  
FROM OPENROWSET( 'SQLNCLI',  
'server=khangcomputer;database=RecruitVietnam;uid=sa;pw  
d=sa;',  
'SELECT * FROM dbo.tblJSCertificates') AS a;  
GO
```

Khi thực thi phát biểu SELECT với hàm OPENROWSET thành công, bạn có thể tìm thấy danh sách mẫu tin trong bảng tblJSCertificates trình bày như hình 8-52.

| CertificateID | CertificateName        |
|---------------|------------------------|
| 1             | Not Provide            |
| 2             | P.h.D                  |
| 3             | Master                 |
| 4             | Bachelor/Honors Degree |
| 5             | Polytechnic/Diploma    |
| 6             | Pre-U/Junior College   |
| 7             | Post-Secondary         |
| 8             | Secondary/High School  |
| 9             | Primary/Junior High    |
| 10            | Others                 |

Hình 8-52: Danh sách mẫu tin trong bảng tblJSCertificates.

Tuy nhiên, bạn có thể nhận thấy lỗi phát sinh trông giống như sau:

```
Msg 15281, Level 16, State 1, Line 1
SQL Server blocked access to STATEMENT
'OpenRowset/OpenDatasource' of component 'Ad Hoc
Distributed Queries' because this component is turned off
as part of the security configuration for this server.
A system administrator can enable the use of 'Ad Hoc
Distributed Queries' by using sp_configure.
```

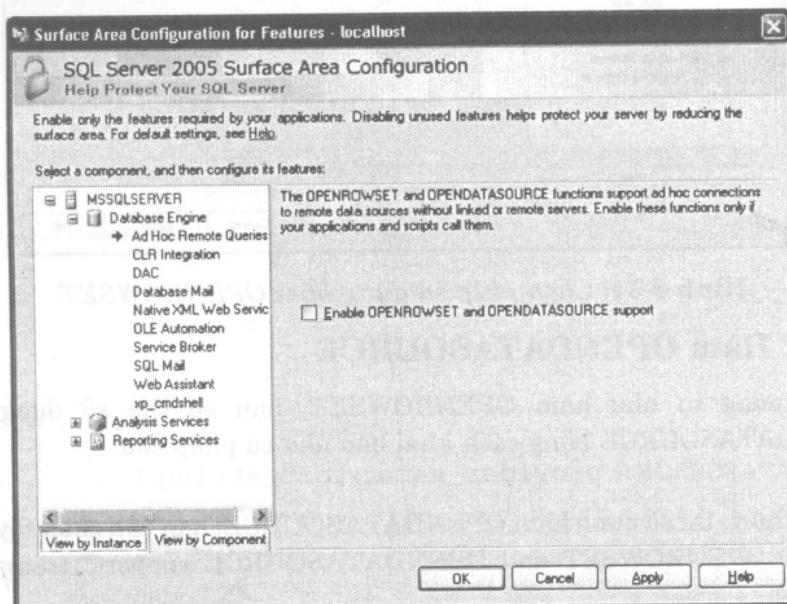
**Chương 8: Giới thiệu hàm trong SQL Server 2005**

59

For more information about enabling "Ad Hoc Distributed Queries", see "Surface Area Configuration" in SQL Server Books Online.

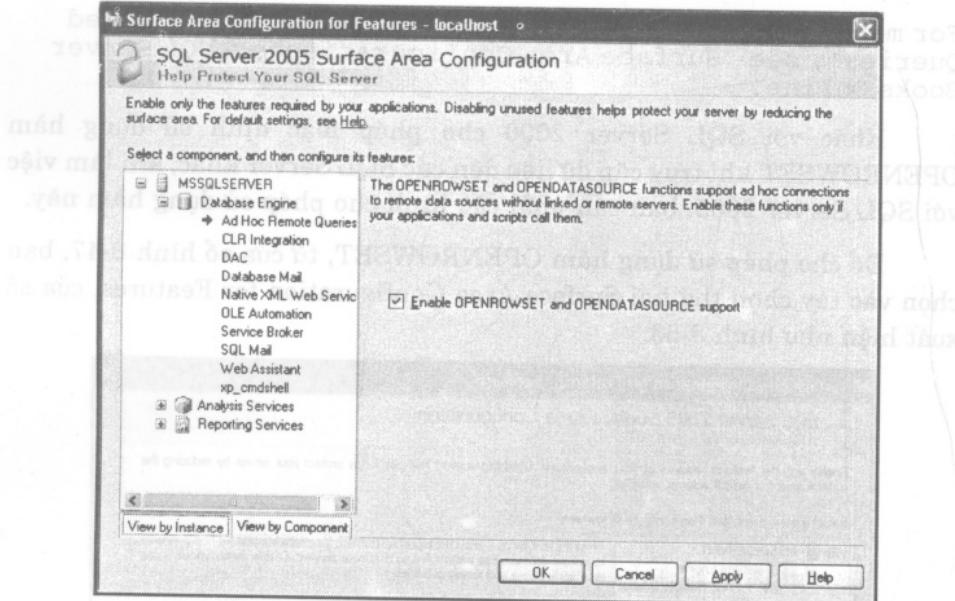
Khác với SQL Server 2000 cho phép mặc định sử dụng hàm OPENROWSET khi truy cập dữ liệu đến các SQL Server khác, khi làm việc với SQL Server 2005, bạn cần phải cấu hình cho phép sử dụng hàm này.

Để cho phép sử dụng hàm OPENROWSET, từ cửa sổ hình 8-47, bạn chọn vào tùy chọn thứ hai Surface Area Configuration for Features, cửa sổ xuất hiện như hình 8-53.



**Hình 8-53: Cửa sổ Surface Area Configuration for Features.**

Chọn vào tùy chọn "Enable OPENROWSET and OPENDATASOURCE support" như hình 8-54 rồi nhấn nút OK hoặc Apply.



**Hình 8-54:** Chọn phép sử dụng hàm OPENROWSET.

### 3.3. Hàm OPENDATASOURCE

Tương tự như hàm OPENROWSET, bạn có thể sử dụng hàm OPENDATASOURCE bằng cách khai báo như cú pháp sau:

```
OPENDATASOURCE ( provider_name, init_string )
```

**Chú ý:** Để sử dụng hàm OPENDATASOURCE, bạn cần chọn tùy chọn “Enable OPENROWSET and OPENDATASOURCE support” trong hình 8-47-2.

Chẳng hạn, bạn có thể khai báo phát biểu SELECT với hàm OPENDATASOURCE để truy vấn mẫu tin trong bảng Categories như ví dụ 8-47

#### Ví dụ 8-47: Khai báo hàm OPENDATASOURCE

```
SELECT CategoryId, categoryNameInVietnamese  
FROM OPENDATASOURCE('SQLNCL1',  
    'Data Source=mysolution;Integrated Security=SSPI')  
    .AccountSystem.dbo.Categories  
GO
```

Khi thực thi phát biểu trong ví dụ trên, kết quả sẽ trình bày như hình

**Chương 8: Giới thiệu hàm trong SQL Server 2005**

61

| CategoryId | categoryNameInVietnamese                |
|------------|---|
| 1          | Túi xách học sinh                       |
| 2          | Túi xách máy tính và điện thoại di động |
| 3          | Túi xách khác                           |
| 4          | Túi xách học sinh nam                   |

**Hình 8-55: Sử dụng hàm OPENDATASOURCE.****3.4. Hàm OPENXML**

Bạn có thể thực thi phát biểu SELECT với mệnh đề FOR XML và chỉ định chế độ xuất dữ liệu bằng RAW, AUTO, EXPLICIT hay PATH với kết quả trả về có định dạng XML.

Loại phát biểu SELECT với mệnh đề FOR XML có thể khai báo trong thủ tục nội tại hay hàm do người dùng định nghĩa.

Chẳng hạn, bạn khai báo phát biểu SELECT để lấy ra dữ liệu trong bảng Customers và SalesInvoices ra định dạng XML như ví dụ 8-48.

**Ví dụ 8-48: Khai báo mệnh đề FOR XML**

```
SELECT Customers.CustomerId, CompanyNameInVietnamese,
       ProductId, Quantity, Price, Discount, VATRate
  FROM Customers INNER JOIN SalesInvoices
    ON Customers.CustomerId = SalesInvoices.CustomerId
   INNER JOIN SalesInvoiceDetails
    ON SalesInvoices.InvoiceNo =
       SalesInvoiceDetails.InvoiceNo
 WHERE ProvinceId= 'HAN'
FOR XML AUTO
GO
```

Khi thực phát biểu SELECT và mệnh đề FOR XML với AUTO, bạn có thể tìm thấy kết quả có định dạng XML như sau:

```
<Customers CustomerId="A0003"
CompanyNameInVietnamese="Công ty Trách Nhiệm Hữu Hạn
Kodaka Vietnam">
  <SalesInvoiceDetails ProductId="P00002" Quantity="100"
Price="10000" Discount="50000" VATRate="10" />
</Customers>
<Customers CustomerId="A0004"
CompanyNameInVietnamese="Công ty Trách Nhiệm Hữu Hạn
E-Google Vietnam">
  <SalesInvoiceDetails ProductId="P00001" Quantity="250"
Price="15000" Discount="50000" VATRate="10" />
</Customers>
```

```
<Customers CustomerId="A0003"
CompanyNameInVietnamese="Công ty Trách Nhiệm Hữu Hạn
Kodaka Vietnam">
  <SalesInvoiceDetails ProductId="P00003" Quantity="300"
Price="10000" Discount="50000" VATRate="10" />
</Customers>
<Customers CustomerId="A0004"
CompanyNameInVietnamese="Công ty Trách Nhiệm Hữu Hạn
E-Google Vietnam">
  <SalesInvoiceDetails ProductId="P00003" Quantity="120"
Price="12500" Discount="55000" VATRate="10" />
</Customers>
```

Để xuất ra kết quả là mỗi mảng tin ứng với một node của XML thì bạn khai báo như ví dụ 8-49.

**Ví dụ 8-49: Khai báo từ khóa RAW**

```
SELECT Customers.CustomerId, CompanyNameInVietnamese,
ProductID, Quantity, Price, Discount, VATRate
FROM Customers INNER JOIN SalesInvoices
ON Customers.CustomerId = SalesInvoices.CustomerId
INNER JOIN SalesInvoiceDetails
ON SalesInvoices.InvoiceNo =
SalesInvoiceDetails.InvoiceNo
WHERE ProvinceId= 'HAN'
FOR XML RAW
GO
```

Khi thực thi phát biểu **SELECT** trên, kết quả trình bày là mỗi node ứng với mỗi mẫu tin.

```
<row CustomerId="A0003" CompanyNameInVietnamese="Công ty  
Trách Nhiệm Hữu Hạn Kodaka Vietnam" ProductId="P00002"  
Quantity="100" Price="10000" Discount="50000"  
VATRate="10" />  
<row CustomerId="A0004" CompanyNameInVietnamese="Công ty  
Trách Nhiệm Hữu Hạn E-Google Vietnam" ProductId="P00001"  
Quantity="250" Price="15000" Discount="50000"  
VATRate="10" />  
<row CustomerId="A0003" CompanyNameInVietnamese="Công ty  
Trách Nhiệm Hữu Hạn Kodaka Vietnam" ProductId="P00003"  
Quantity="300" Price="10000" Discount="50000"  
VATRate="10" />  
<row CustomerId="A0004" CompanyNameInVietnamese="Công ty  
Trách Nhiệm Hữu Hạn E-Google Vietnam" ProductId="P00003"  
Quantity="120" Price="12500" Discount="55000"  
VATRate="10" />
```

Trong trường hợp bạn muốn kết quả trình bày mỗi node ứng với một mẩu tin, mỗi mẩu tin bao gồm nhiều cột dữ liệu ứng với từng node con thì khai báo từ khóa PATH.

**Chương 8: Giới thiệu hàm trong SQL Server 2005**

63

**Ví dụ 8-50: Khai báo từ khóa PATH**

```
SELECT Customers.CustomerId, CompanyNameInVietnamese,
ProductID, Quantity, Price, Discount, VATRate
FROM Customers INNER JOIN SalesInvoices
ON Customers.CustomerId = SalesInvoices.CustomerId
INNER JOIN SalesInvoiceDetails
ON SalesInvoices.InvoiceNo =
SalesInvoiceDetails.InvoiceNo
WHERE ProvinceId= 'HAN'
FOR XML PATH
GO
```

Khi thực thi phát biểu SELECT trong ví dụ trên, bạn có thể tìm thấy kết quả trình bày như sau:

```
<row>
<CustomerId>A0003</CustomerId>
<CompanyNameInVietnamese>Công ty Trách Nhiệm Hữu Hạn
Kodaka Vietnam</CompanyNameInVietnamese>
<ProductId>P00002</ProductId>
<Quantity>100</Quantity>
<Price>10000</Price>
<Discount>50000</Discount>
<VATRate>10</VATRate>
</row>
<row>
<CustomerId>A0004</CustomerId>
<CompanyNameInVietnamese>Công ty Trách Nhiệm Hữu Hạn
E-Google Vietnam</CompanyNameInVietnamese>
<ProductId>P00001</ProductId>
<Quantity>250</Quantity>
<Price>15000</Price>
<Discount>50000</Discount>
<VATRate>10</VATRate>
</row>
<row>
<CustomerId>A0003</CustomerId>
<CompanyNameInVietnamese>Công ty Trách Nhiệm Hữu Hạn
Kodaka Vietnam</CompanyNameInVietnamese>
<ProductId>P00003</ProductId>
<Quantity>300</Quantity>
<Price>10000</Price>
<Discount>50000</Discount>
<VATRate>10</VATRate>
</row>
<row>
<CustomerId>A0004</CustomerId>
<CompanyNameInVietnamese>Công ty Trách Nhiệm Hữu Hạn
E-Google Vietnam</CompanyNameInVietnamese>
<ProductId>P00003</ProductId>
<Quantity>120</Quantity>
```

**M® 64****Chương 8: Giới thiệu hàm trong SQL Server 2005**

```
<Price>12500</Price>
<Discount>55000</Discount>
<VATRate>10</VATRate>
</row>
```

Thay vì sử dụng phát biểu SELECT với mệnh đề FOR XML, bạn có thể sử dụng hàm OPENXML để đọc dữ liệu từ chuỗi định dạng XML với cú pháp như sau:

```
OPENXML( @idoc int [ in ] ,
rowpattern nvarchar [ in ] , [ flags byte [ in ] ] )
[ WITH ( SchemaDeclaration | TableName ) ]
```

Chẳng hạn, bạn khai báo tạo ra bảng dữ liệu có cột dữ liệu được lấy ra từ chuỗi XML như ví dụ 8-51.

**Ví dụ 8-51: Khai báo bảng dữ liệu**

```
DECLARE @idoc int
DECLARE @doc varchar(1000)
SET @doc =
<ROOT>
<Customer>
<Order OrderID="SI00000001">
    <CustomerID="A0001" OrderDate="2007-09-10">
        <OrderDetail ProductID="P00001" Quantity="12"/>
        <OrderDetail ProductID="P00002" Quantity="10"/>
    </Order>
</Customer>
<Customer>
    <Order OrderID="SI00000002">
        <CustomerID="A0002" OrderDate="2007-09-10">
            <OrderDetail ProductID="P00001" Quantity="10"/>
            <OrderDetail ProductID="P00002" Quantity="20"/>
        </Order>
</Customer>
</ROOT>' 
EXEC sp_xml_preparedocument @idoc OUTPUT, @doc
SELECT *
FROM OPENXML (@idoc,
'/ROOT/Customer/Order/OrderDetail',2)
WITH (OrderID varchar(10) '...@OrderID',
CustomerID char(5) '...@CustomerID',
OrderDate datetime '...@OrderDate',
ProductID varchar(10) '@ProductID',
Quantity int '@Quantity')
GO
```

Khi thực thi phát biểu SQL trên, bạn nhận kết quả trình bày như hình 8-56.

## Chương 8: Giới thiệu hàm trong SQL Server 2005

65 

|   | OrderID    | CustomerID | OrderDate               | ProductID | Quantity |
|---|------------|------------|-------------------------|-----------|----------|
| 1 | S100000001 | A0001      | 2007-09-10 00:00:00.000 | P00001    | 12       |
| 2 | S100000001 | A0001      | 2007-09-10 00:00:00.000 | P00002    | 10       |
| 3 | S100000002 | A0002      | 2007-09-10 00:00:00.000 | P00001    | 10       |
| 4 | S100000002 | A0002      | 2007-09-10 00:00:00.000 | P00002    | 20       |

Hình 8-56: Sử dụng hàm OPENXML.

Chú ý: Để tìm hiểu chi tiết cách truy cập dữ liệu XML bằng đối tượng ADO.NET, bạn có thể tìm đọc quyển sách “C# 2005 - Tập 4: Lập trình cơ sở dữ liệu” do nhà sách Minh Khai phát hành trong năm 2007.

## 4. HÀM TRẢ VỀ GIÁ TRỊ TỔNG HỢP (AGGREGATE)

Khi cần tổng hợp hay thống kê dữ liệu, bạn có thể sử dụng hàm như: SUM, COUNT, AVG, MAX, MIN, CHECKSUM, CHECKSUM\_AGG, STDEV, STDEVP, COUNT\_BIG, VAR, GROUPING, VARP có kết hợp với mệnh đề GROUP BY.

5 hàm cơ bản thường được sử dụng nhiều nhất khi khai báo phát biểu SELECT với mệnh đề GROUP BY là SUM, COUNT, AVG, MAX, MIN.

Chẳng hạn, bạn có thể thống kê số lượng, giá và tiền bán hàng bằng cách khai báo mặc định GROUP BY như ví dụ 8-52.

### Ví dụ 8-52: Khai báo 5 hàm cơ bản

```
SELECT D.ProductId,
       COUNT(*) As NumberOfProduct,
       SUM(Quantity) As TotalQuantity,
       MAX(Price) As MAXPrice,
       MIN(Price) As MINPrice,
       AVG(Price) As AveragePrice,
       SUM(Quantity*Price) As TotalAmount,
       SUM(Discount) As TotalDiscount,
       SUM(VATRate*Quantity*Price) As VATAmount
  FROM Products P
 INNER JOIN SalesInvoiceDetails D
    ON P.ProductId = D.ProductId
 GROUP BY D.ProductId
GO
```

Khi thực thi phát biểu SELECT với mệnh đề GROUP BY cùng với 5 hàm cơ bản, kết quả trình bày như hình 8-57.

## Chương 8: Giới thiệu hàm trong SQL Server 2005

| ProductId | Nu.    | Tot. | MAX | MINPrice | AveragePrice | TotalAmount  | TotalDiscount | VATAmount |
|-----------|--------|------|-----|----------|--------------|--------------|---------------|-----------|
| 1         | P00001 | 4    | 515 | 15000    | 10000        | 13125.000000 | 7187500       | 100000    |
| 2         | P00002 | 7    | 750 | 12500    | 10000        | 10857.142857 | 8025000       | 160000    |
| 3         | P00003 | 6    | 625 | 12500    | 10000        | 11666.666666 | 6812500       | 280000    |
| 4         | P00004 | 7    | 176 | 12500    | 11500        | 12214.285714 | 2136500       | 105000    |
| 5         | P00005 | 2    | 130 | 13500    | 12500        | 13000.000000 | 1750000       | 90000     |
| 6         | P00006 | 4    | 90  | 14500    | 12500        | 13500.000000 | 1245000       | 70000     |

Hình 8-57: Sử dụng 5 hàm cơ bản.

Trong trường hợp sử dụng mệnh đề GROUP BY và WITH ROLLUP, bạn có thể sử dụng hàm GROUPING như ví dụ 8-53.

**Ví dụ 8-53: Khai báo sử dụng hàm GROUPING**

```
SELECT D.ProductId,
       COUNT(*) As NumberOfProduct,
       SUM(Quantity) As TotalQuantity,
       AVG(Price) As AveragePrice,
       SUM(Quantity*Price) As TotalAmount,
       SUM(Discount) As TotalDiscount,
       SUM(VATRate*Quantity*Price) As VATAmount,
       GROUPING(D.ProductId)
  FROM Products P
 INNER JOIN SalesInvoiceDetails D
    ON P.ProductId = D.ProductId
 GROUP BY D.ProductId
 WITH ROLLUP
GO
```

Nếu thực thi phát biểu SELECT trong ví dụ trên, bạn có thể tìm thấy kết quả trình bày như hình 8-58.

| ProductId | Nu.    | Total | AveragePrice | TotalAmount  | TotalDiscount | VATAmount | GROUPINGProduct |
|-----------|--------|-------|--------------|--------------|---------------|-----------|-----------------|
| 1         | P00001 | 4     | 515          | 13125.000000 | 7187500       | 100000    | 71875000 0      |
| 2         | P00002 | 7     | 750          | 10857.142857 | 8025000       | 160000    | 80250000 0      |
| 3         | P00003 | 6     | 625          | 11666.666666 | 6812500       | 280000    | 68125000 0      |
| 4         | P00004 | 7     | 176          | 12214.285714 | 2136500       | 105000    | 21365000 0      |
| 5         | P00005 | 2     | 130          | 13000.000000 | 1750000       | 90000     | 17500000 0      |
| 6         | P00006 | 4     | 90           | 13500.000000 | 1245000       | 70000     | 12450000 0      |
|           | NULL   | 30    | 2286         | 12133.333333 | 27156500      | 805000    | 271565000 1     |

Hình 8-58: Sử dụng hàm GROUPING.

Nếu bạn có nhu cầu tính toán giá trị phương sai của cột dữ liệu giá trị số thì sử dụng hàm VAR tương tự như ví dụ 8-54.

**Ví dụ 8-54: Khai báo hàm VAR**

```
SELECT D.ProductId,
       SUM(Quantity) As TotalQuantity,
```

**Chương 8: Giới thiệu hàm trong SQL Server 2005**

67



```

    AVG(Price) As AveragePrice,
    SUM(Quantity*Price) As TotalAmount,
    SUM(Discount) As TotalDiscount,
    VAR(Discount) As VarDiscount
  FROM Products P
  INNER JOIN SalesInvoiceDetails D
  ON P.ProductId = D.ProductId
  GROUP BY D.ProductId
GO
  
```

Chú ý: VAR là hàm thống kê tính phương sai (variance) của một đại lượng ngẫu nhiên.

Nếu thực thi phát biểu SELECT trong ví dụ trên, bạn có thể tìm thấy kết quả trình bày như hình 8-59.

| ProductId | TotalQuantity | AveragePrice | TotalAmount | TotalDiscount | VarDiscount      |
|-----------|---------------|--------------|-------------|---------------|------------------|
| 1 P00001  | 515           | 13125.000000 | 7187500     | 100000        | 83333333.333333  |
| 2 P00002  | 750           | 10857.142857 | 8025000     | 160000        | 523809523.809524 |
| 3 P00003  | 625           | 11666.666666 | 6812500     | 280000        | 896666666.666667 |
| 4 P00004  | 176           | 12214.285714 | 2136500     | 105000        | 225000000        |
| 5 P00005  | 130           | 13000.000000 | 1750000     | 90000         | 450000000        |
| 6 P00006  | 90            | 13500.000000 | 1245000     | 70000         | 225000000        |

Hình 8-59: Sử dụng hàm VAR.

## 5. HÀM TRẢ VỀ KHOẢNG GIÁ TRỊ (RANKING)

Hàm thuộc nhóm Ranking bao gồm (RANK, NTILE, DENSE\_RANK, ROW\_NUMBER) trả về giá trị ứng với thứ bậc cho mỗi hàng trong tập dữ liệu. Tùy thuộc vào hàm mà bạn sử dụng, một số mẫu tin có thể nhận cùng một thứ bậc do chúng có cùng giá trị.

### 5.1. Hàm RANK

Để tìm hiểu hàm RANK, trước tiên bạn liệt kê danh sách sản phẩm nhập kho trong bảng ImportDetails ứng với kho ST001 với phát biểu SELECT như ví dụ 8-55.

#### Ví dụ 8-55: Liệt kê danh sách sản phẩm trong kho ST001

```

SELECT P.ProductID, P.ProductNameInVietnamese,
D.StockID, D.Quantity
FROM ImportDetails D JOIN Products P
ON D.ProductID = P.ProductID
  
```

```
WHERE D.StockID = 'ST001'
ORDER BY P.ProductNameInVietnamese
GO
```

Khi thực thi phát biểu SELECT trên, bạn có thể tìm thấy danh sách mẫu tin trình bày như hình 8-60.

| ProductID | ProductNameInVietnamese        | Stock... | Quantity |
|-----------|--------------------------------|----------|----------|
| P00004    | Túi áo mưa                     | ST001    | 400      |
| P00001    | Túi xách                       | ST001    | 500      |
| P00001    | Túi xách                       | ST001    | 400      |
| P00003    | Túi xách dùng cho học sinh nam | ST001    | 700      |
| P00002    | Túi xách dùng cho học sinh nữ  | ST001    | 600      |
| P00002    | Túi xách dùng cho học sinh nữ  | ST001    | 450      |
| P00005    | Túi xách dùng cho Máy tính     | ST001    | 200      |

Hình 8-60: Sản phẩm nhập kho ST001.

Nếu bạn muốn trình bày thứ bậc của từng sản phẩm dựa trên số lượng nhập trong kho ST001 thì sử dụng hàm RANK và mệnh đề ORDER BY ứng với mã kho như ví dụ 8-56.

#### Ví dụ 8-56: Khai báo sử dụng hàm RANK

```
SELECT P.ProductID, P.ProductNameInVietnamese,
D.StockID, D.Quantity,
RANK() OVER (PARTITION BY D.StockID order by D.Quantity) as
RANK
FROM ImportDetails D JOIN Products P
ON D.ProductID = P.ProductID
WHERE D.StockID = 'ST001'
ORDER BY RANK, P.ProductNameInVietnamese
GO
```

Khi thực thi phát biểu SELECT trong ví dụ trên, thứ bậc của sản phẩm trong kho ST001 trình bày như hình 8-61.

| ProductID | ProductNameInVietnamese        | Stock... | Quantity | RANK |
|-----------|--------------------------------|----------|----------|------|
| P00005    | Túi xách dùng cho Máy tính     | ST001    | 200      | 1    |
| P00004    | Túi áo mưa                     | ST001    | 400      | 2    |
| P00001    | Túi xách                       | ST001    | 400      | 2    |
| P00002    | Túi xách dùng cho học sinh nữ  | ST001    | 450      | 4    |
| P00001    | Túi xách                       | ST001    | 500      | 5    |
| P00002    | Túi xách dùng cho học sinh nữ  | ST001    | 600      | 6    |
| P00003    | Túi xách dùng cho học sinh nam | ST001    | 700      | 7    |

Hình 8-61: Thứ bậc của sản phẩm.

**Chương 8: Giới thiệu hàm trong SQL Server 2005**

69



Trong ví dụ trên, mẫu tin thứ 2 và 3 có cùng thứ bậc là 2, bậc kế tiếp là 4 thay vì 3 cho mẫu tin thứ 4 bởi vì hàm RANK không tính giá trị liên tục, thay vào đó nó tính theo số lần cấp phát thứ bậc.

## 5.2. Hàm NTILE

Tương tự như hàm RANK, nhưng hàm NTILE cho phép bạn chỉ định số ứng với thứ bậc cho trước, từ đó các mẫu tin trong bảng dữ liệu được trình bày dựa trên thứ bậc này. Số thứ bậc được đánh liên tục và số mẫu tin được chia đều cho các thứ bậc ưu tiên từ số thứ bậc nhỏ tăng dần.

Chẳng hạn, bạn liệt kê danh sách sản phẩm nhập vào kho ST002 bằng phát biểu SELECT như ví dụ 8-57.

### Ví dụ 8-57: Khai báo liệt kê sản phẩm nhập trong kho ST002

```
SELECT P.ProductID, P.ProductNameInVietnamese,
D.StockID, D.Quantity
FROM ImportDetails D JOIN Products P
ON D.ProductID = P.ProductID
WHERE D.StockID = 'ST002'
ORDER BY D.Quantity, P.ProductNameInVietnamese
GO
```

Kết quả trình bày như hình 8-62 là 11 mẫu tin được sắp xếp tăng dần theo số lượng khi thực thi phát biểu trong ví dụ trên.

|    | ProductID | ProductNameInVietnamese              | Stock... | Quantity |
|----|-----------|--------------------------------------|----------|----------|
| 1  | P00004    | Túi áo mưa                           | ST002    | 50       |
| 2  | P00002    | Túi xách dùng cho học sinh nữ        | ST002    | 50       |
| 3  | P00007    | Túi xách dùng cho PC                 | ST002    | 50       |
| 4  | P00005    | Túi xách dùng cho Máy tính           | ST002    | 70       |
| 5  | P00002    | Túi xách dùng cho học sinh nữ        | ST002    | 100      |
| 6  | P00006    | Túi xách dùng cho Điện thoại di động | ST002    | 150      |
| 7  | P00007    | Túi xách dùng cho PC                 | ST002    | 150      |
| 8  | P00001    | Túi xách                             | ST002    | 200      |
| 9  | P00006    | Túi xách dùng cho Điện thoại di động | ST002    | 200      |
| 10 | P00001    | Túi xách                             | ST002    | 400      |
| 11 | P00007    | Túi xách dùng cho PC                 | ST002    | 700      |

Hình 8-62: Danh sách sản phẩm nhập vào kho ST002.

Trong trường hợp cấp số bậc cho từng sản phẩm trong kho ST002, bạn sử dụng hàm RANK như ví dụ 8-58.

**Ví dụ 8-58: Khai báo hàm RANK**

```

SELECT P.ProductID, P.ProductNameInVietnamese, D.StockID, D.Quantity,
RANK() OVER (PARTITION BY D.StockID ORDER BY D.Quantity) AS
RANK
FROM ImportDetails D JOIN Products P
ON D.ProductID = P.ProductID
WHERE D.StockID = 'ST002'
ORDER BY D.Quantity, P.ProductNameInVietnamese
GO
    
```

Khi thực thi phát biểu SELECT trong ví dụ trên, kết quả như hình 8-63.

|    | ProductID | ProductNameInVietnamese              | StockID | Quantity | RANK |
|----|-----------|--------------------------------------|---------|----------|------|
| 1  | P00004    | Túi áo mưa                           | ST002   | 50       | 1    |
| 2  | P00002    | Túi xách dùng cho học sinh nữ        | ST002   | 50       | 1    |
| 3  | P00007    | Túi xách dùng cho PC                 | ST002   | 50       | 1    |
| 4  | P00005    | Túi xách dùng cho Máy tính           | ST002   | 70       | 4    |
| 5  | P00002    | Túi xách dùng cho học sinh nữ        | ST002   | 100      | 5    |
| 6  | P00006    | Túi xách dùng cho Điện thoại di động | ST002   | 150      | 6    |
| 7  | P00007    | Túi xách dùng cho PC                 | ST002   | 150      | 6    |
| 8  | P00001    | Túi xách                             | ST002   | 200      | 8    |
| 9  | P00006    | Túi xách dùng cho Điện thoại di động | ST002   | 200      | 8    |
| 10 | P00001    | Túi xách                             | ST002   | 400      | 10   |
| 11 | P00007    | Túi xách dùng cho PC                 | ST002   | 700      | 11   |

Hình 8-63: Cấp phát thứ bậc.

Tuy nhiên, nếu muốn giới hạn thứ bậc của chúng trong khoảng 5 giá trị, bạn sử dụng hàm NTILE như ví dụ 8-59.

**Ví dụ 8-59: Khai báo sử dụng hàm NTILE**

```

SELECT P.ProductID, P.ProductNameInVietnamese,
D.StockID, D.Quantity,
NTILE(5) OVER (PARTITION BY D.StockID ORDER BY D.Quantity) AS NTILE
FROM ImportDetails D JOIN Products P
ON D.ProductID = P.ProductID
WHERE D.StockID = 'ST002'
ORDER BY D.Quantity, P.ProductNameInVietnamese
GO
    
```

**Chương 8:** Giới thiệu hàm trong SQL Server 2005

71

Khi thực thi phát biểu SELECT với hàm NTILE có giá trị là 5, bạn có thể tìm thấy kết quả trình bày như hình 8-64.

Chú ý: Số mẫu tin là 11, 11 chia 5 được 2 dư 1 nên bình quân mỗi bậc có 2 mẫu tin, dư 1 mẫu tin ưu tiên đưa vào bậc 1.

|    | ProductID | ProductNameInVietnamese              | StockID | Quantity | NTILE |
|----|-----------|--------------------------------------|---------|----------|-------|
| 1  | P00004    | Túi áo mưa                           | ST002   | 50       | 1     |
| 2  | P00002    | Túi xách dùng cho học sinh nữ        | ST002   | 50       | 1     |
| 3  | P00007    | Túi xách dùng cho PC                 | ST002   | 50       | 1     |
| 4  | P00005    | Túi xách dùng cho Máy tính           | ST002   | 70       | 2     |
| 5  | P00002    | Túi xách dùng cho học sinh nữ        | ST002   | 100      | 2     |
| 6  | P00006    | Túi xách dùng cho Điện thoại di động | ST002   | 150      | 3     |
| 7  | P00007    | Túi xách dùng cho PC                 | ST002   | 150      | 3     |
| 8  | P00001    | Túi xách                             | ST002   | 200      | 4     |
| 9  | P00006    | Túi xách dùng cho Điện thoại di động | ST002   | 200      | 4     |
| 10 | P00001    | Túi xách                             | ST002   | 400      | 5     |
| 11 | P00007    | Túi xách dùng cho PC                 | ST002   | 700      | 5     |

Hình 8-64: Sử dụng hàm NTILE.

### 5.3. Hàm DENSE\_RANK

Để tìm hiểu hàm DENSE\_RANK, trước tiên bạn liệt kê danh sách sản phẩm nhập kho trong bảng ImportDetails có sử dụng hàm RANK với phát biểu SELECT như ví dụ 8-60.

#### Ví dụ 8-60: Khai báo liệt kê danh sách sản phẩm

```
SELECT P.ProductID, P.ProductNameInVietnamese,
D.StockID, D.Quantity
FROM ImportDetails D JOIN Products P
ON D.ProductID = P.ProductID
ORDER BY D.Quantity, P.ProductNameInVietnamese
GO
```

Khi thực thi phát biểu SELECT trên, bạn có thể tìm thấy danh sách mẫu tin trình bày như hình 8-65.

Trong ví dụ trên, mẫu tin thứ 1, 2, 3 có cùng thứ bậc là 1, bậc kế tiếp là 4 thay vì 2 cho mẫu tin thứ 4 bởi vì hàm RANK không tính giá trị liên tục mà nó tính theo số lần cấp phát thứ bậc.

Nếu sử dụng hàm DENSE\_RANK, bạn có thể tìm thấy thứ bậc có giá trị liên tục.

The screenshot shows the SSMS Results pane displaying a table of product data. The table has columns: ProductID, ProductNameInVietnamese, StockID, Quantity, and RANK. The RANK column uses the DENSE\_RANK function. The data includes various products like 'Túi áo mưa', 'Túi xách', and 'Túi xách dùng cho học sinh' across different stock locations (ST001, ST002) and quantities.

|    | ProductID | ProductNameInVietnamese        | Stock... | Quantity | RANK |
|----|-----------|--------------------------------|----------|----------|------|
| 1  | P00004    | Túi áo mưa                     | ST002    | 50       | 1    |
| 2  | P00002    | Túi xách dùng cho học sinh nữ  | ST002    | 50       | 1    |
| 3  | P00007    | Túi xách dùng cho PC           | ST002    | 50       | 1    |
| 4  | P00005    | Túi xách dùng cho Máy tính     | ST002    | 70       | 4    |
| 5  | P00002    | Túi xách dùng cho học sinh nữ  | ST002    | 100      | 5    |
| 6  | P00006    | Túi xách dùng cho Điện thoại   | ST002    | 150      | 6    |
| 7  | P00007    | Túi xách dùng cho PC           | ST002    | 150      | 6    |
| 8  | P00001    | Túi xách                       | ST002    | 200      | 8    |
| 9  | P00006    | Túi xách dùng cho Điện thoại   | ST002    | 200      | 8    |
| 10 | P00005    | Túi xách dùng cho Máy tính     | ST001    | 200      | 1    |
| 11 | P00004    | Túi áo mưa                     | ST001    | 400      | 2    |
| 12 | P00001    | Túi xách                       | ST001    | 400      | 2    |
| 13 | P00001    | Túi xách                       | ST002    | 400      | 10   |
| 14 | P00002    | Túi xách dùng cho học sinh nữ  | ST001    | 450      | 4    |
| 15 | P00001    | Túi xách                       | ST001    | 500      | 5    |
| 16 | P00002    | Túi xách dùng cho học sinh nữ  | ST001    | 600      | 6    |
| 17 | P00003    | Túi xách dùng cho học sinh ... | ST001    | 700      | 7    |
| 18 | P00007    | Túi xách dùng cho PC           | ST002    | 700      | 11   |

Hình 8-65: Sản phẩm nhập kho.

Để làm điều này, bạn khai báo phát biểu SELECT với hàm DENSE\_RANK như ví dụ 8-61.

#### Ví dụ 8-61: Khai báo sử dụng hàm DENSE\_RANK

```
SELECT P.ProductID, P.ProductNameInVietnamese,
D.StockID, D.Quantity,
DENSE_RANK() OVER (PARTITION BY D.StockID ORDER BY
D.Quantity) AS DENSERANK
FROM ImportDetails D JOIN Products P
ON D.ProductID = P.ProductID
ORDER BY D.Quantity, P.ProductNameInVietnamese
GO
```

Khi thực thi phát biểu SELECT trên, bạn có thể tìm thấy danh sách mẫu tin có thứ bậc liên tục trình bày như hình 8-66.

Chương 8: Giới thiệu hàm trong SQL Server 2005

73

|    | ProductID | ProductNameInVietnamese              | Stock... | Quantity | DENSE RANK |
|----|-----------|--------------------------------------|----------|----------|------------|
| 1  | P00004    | Túi áo mưa                           | ST002    | 50       | 1          |
| 2  | P00002    | Túi xách dùng cho học sinh nữ        | ST002    | 50       | 1          |
| 3  | P00007    | Túi xách dùng cho PC                 | ST002    | 50       | 1          |
| 4  | P00005    | Túi xách dùng cho Máy tính           | ST002    | 70       | 2          |
| 5  | P00002    | Túi xách dùng cho học sinh nữ        | ST002    | 100      | 3          |
| 6  | P00006    | Túi xách dùng cho Điện thoại di động | ST002    | 150      | 4          |
| 7  | P00007    | Túi xách dùng cho PC                 | ST002    | 150      | 4          |
| 8  | P00001    | Túi xách                             | ST002    | 200      | 5          |
| 9  | P00006    | Túi xách dùng cho Điện thoại di động | ST002    | 200      | 5          |
| 10 | P00005    | Túi xách dùng cho Máy tính           | ST001    | 200      | 1          |
| 11 | P00004    | Túi áo mưa                           | ST001    | 400      | 2          |
| 12 | P00001    | Túi xách                             | ST001    | 400      | 2          |
| 13 | P00001    | Túi xách                             | ST002    | 400      | 6          |
| 14 | P00002    | Túi xách dùng cho học sinh nữ        | ST001    | 450      | 3          |
| 15 | P00001    | Túi xách                             | ST001    | 500      | 4          |
| 16 | P00002    | Túi xách dùng cho học sinh nữ        | ST001    | 600      | 5          |
| 17 | P00003    | Túi xách dùng cho học sinh nam       | ST001    | 700      | 6          |
| 18 | P00007    | Túi xách dùng cho PC                 | ST002    | 700      | 7          |

**Hình 8-66:** Sản phẩm nhập kho.

Trong ví dụ trên, mẫu tin thứ 1, 2, 3 có cùng thứ bậc là 1, bậc kế tiếp là 2 thay vì 4 cho mẫu tin thứ 4 bởi vì hàm DENSE\_RANK tính giá trị liên tục theo số lần cấp phát thứ bậc.

#### 5.4. Hàm ROW NUMBER

Hàm ROW\_NUMBER trả về số thứ tự (số đầu tiên bắt đầu) cho mẫu tin trong tập mẫu tin mà phát biểu SELECT trả về. Hàm này thường được sử dụng để phân trang cho tập mẫu tin cần lấy ra.

Chẳng hạn, khi liệt kê danh sách sản phẩm nhập kho, bạn sử dụng hàm ROW NUMBER để tạo ra số thứ tự như ví dụ 8-62.

#### **Ví dụ 8-62: Khai báo hàm BROW NUMBER**

```
SELECT P.ProductID, P.ProductNameInVietnamese,  
D.StockID, D.Quantity,  
ROW_NUMBER() OVER(ORDER BY D.Quantity DESC) AS RowNumber
```

## Chương 8: Giới thiệu hàm trong SQL Server 2005

```
FROM ImportDetails D JOIN Products P
ON D.ProductID = P.ProductID
ORDER BY RowNumber, P.ProductNameInVietnamese
GO
```

Khi thực thi phát biểu SELECT với hàm ROW\_NUMBER, bạn có thể tìm thấy số thứ tự của mẫu tin sắp xếp theo chiều tăng dần như hình 8-67.

| ProductID | ProductNameInVietnamese              | Stock... | Quantity | RowNumber |
|-----------|--------------------------------------|----------|----------|-----------|
| 1 P00007  | Túi xách dùng cho PC                 | ST002    | 700      | 1         |
| 2 P00003  | Túi xách dùng cho học sinh nam       | ST001    | 700      | 2         |
| 3 P00002  | Túi xách dùng cho học sinh nữ        | ST001    | 600      | 3         |
| 4 P00001  | Túi xách                             | ST001    | 500      | 4         |
| 5 P00002  | Túi xách dùng cho học sinh nữ        | ST001    | 450      | 5         |
| 6 P00001  | Túi xách                             | ST001    | 400      | 6         |
| 7 P00001  | Túi xách                             | ST002    | 400      | 7         |
| 8 P00004  | Túi áo mưa                           | ST001    | 400      | 8         |
| 9 P00006  | Túi xách dùng cho Điện thoại di động | ST002    | 200      | 9         |
| 10 P00005 | Túi xách dùng cho Máy tính           | ST001    | 200      | 10        |
| 11 P00001 | Túi xách                             | ST002    | 200      | 11        |
| 12 P00007 | Túi xách dùng cho PC                 | ST002    | 150      | 12        |
| 13 P00006 | Túi xách dùng cho Điện thoại di động | ST002    | 150      | 13        |
| 14 P00002 | Túi xách dùng cho học sinh nữ        | ST002    | 100      | 14        |
| 15 P00005 | Túi xách dùng cho Máy tính           | ST002    | 70       | 15        |
| 16 P00004 | Túi áo mưa                           | ST002    | 50       | 16        |
| 17 P00007 | Túi xách dùng cho PC                 | ST002    | 50       | 17        |
| 18 P00002 | Túi xách dùng cho học sinh nữ        | ST002    | 50       | 18        |

Hình 8-67: Sử dụng hàm ROW\_NUMBER.

Nếu có nhu cầu lấy ra số lượng mẫu tin ứng với số bắt đầu và kết thúc, bạn có thể kết hợp với biểu thức bảng bằng cách sử dụng phát biểu WITH như ví dụ 8-63.

#### Ví dụ 8-63: Khai báo lấy ra số mẫu tin chỉ định

```
WITH SALES
AS
(
    SELECT P.ProductID, P.ProductNameInVietnamese,
    D.StockID, D.Quantity,
    ROW_NUMBER() OVER(ORDER BY D.Quantity DESC) AS RowNumber
```

**Chương 8: Giới thiệu hàm trong SQL Server 2005**

75

```

FROM ImportDetails D JOIN Products P
ON D.ProductID = P.ProductID
)
SELECT * FROM SALES
WHERE RowNumber BETWEEN 5 AND 10
ORDER BY RowNumber
GO
  
```

Khi thực thi phát biểu SELECT với hàm ROW\_NUMBER, bạn có thể lấy ra số mẫu tin từ số 5 đến số 10 ứng với thứ tự của mẫu tin đã sắp xếp theo chiều tăng dần như hình 8-68.

|   | ProductID | ProductName\ Vietnamese              | Stock... | Quantity | RowNumber |
|---|-----------|--------------------------------------|----------|----------|-----------|
| 1 | P00002    | Túi xách dùng cho học sinh nữ        | ST001    | 450      | 5         |
| 2 | P00001    | Túi xách                             | ST001    | 400      | 6         |
| 3 | P00001    | Túi xách                             | ST002    | 400      | 7         |
| 4 | P00004    | Túi áo mưa                           | ST001    | 400      | 8         |
| 5 | P00006    | Túi xách dùng cho Điện thoại di động | ST002    | 200      | 9         |
| 6 | P00005    | Túi xách dùng cho Máy tính           | ST001    | 200      | 10        |

**Hình 8-68:** Biểu thức bảng.

Bằng cách sử dụng hàm ROW\_NUMBER và biểu thức bảng, bạn có thể phân trang dữ liệu ngay bên trong cơ sở dữ liệu trước khi chuyển đến đối tượng ADO.NET.

Nếu bạn đã làm quen với ADO.NET 2005 trong ASP.NET 2005, để có thể lấy ra tập dữ liệu khi người sử dụng chọn trang thứ i trên trang ASP.NET, bạn sẽ sử dụng phương thức Fill của đối tượng SqlDataAdapter để điền dữ liệu từ thủ tục nội tại vào đối tượng DataTable thay vì đọc dữ liệu và phân trang bằng đối tượng SqlDataAdapter.

## 6. KẾT CHƯƠNG

Trong chương này, chúng ta tập trung tìm hiểu các hàm có sẵn của SQL Server, hàm do người sử dụng định nghĩa và hàm tạo ra bằng ngôn ngữ lập trình T-SQL.

 76

## Chương 8: Giới thiệu hàm trong SQL Server 2005

Chúng ta sẽ tiếp tục tìm hiểu chi tiết về các phát biểu truy vấn nâng cao và câu truy vấn động trong chương kế tiếp trước khi tìm hiểu cách khai báo biến, phát biểu điều khiển và thủ tục nội tại.

Ngoài ra, bạn sẽ tìm hiểu hàm mới được giới thiệu trong SQL Server 2005 là EVENTDATA, nó dùng để nắm giữ thông tin thay đổi cấu trúc cơ sở dữ liệu và Server khi sử dụng DDL Trigger trong chương 12.

Chương 9:

# PHÁT BIỂU TRUY VẤN DỮ LIỆU NÂNG CAO

## Tóm tắt chương 9

Trong chương trước, bạn đã tìm hiểu các hàm có sẵn và hàm do người sử dụng định nghĩa; SQL Server 2005 cho phép bạn tạo ra các phát biểu với nhiều cấu trúc đặc biệt để có thể kết xuất dữ liệu theo định dạng phức tạp.

Trong chương này chúng ta cùng tìm hiểu một số phép toán được giới thiệu trong SQL Server 2005 như PIVOT và UNPIVOT dùng để chuyển dữ liệu chiều ngang sang chiều dọc và ngược lại.

Các vấn đề chính sẽ được đề cập:

- ✓ Phát biểu truy vấn động.
  - ✓ Làm việc với nhiều mệnh đề.
  - ✓ Phép toán PIVOT và UNPIVOT

## 1. PHÁT BIỂU TRUY VẤN ĐÔNG

Khi làm việc với SQL Server, để truy cập đối tượng Table hay View, bạn cần chỉ định tên của chúng dạng tường minh. Trong một vài trường hợp, chúng ta không biết trước tên đối tượng cần truy cập, chính vì vậy bạn cần phải sử dụng câu truy vấn động.

**Chú ý:** Bạn có thể tìm thấy các ví dụ trình bày của chương này nằm trong tập tin có tên **DynamicQuery.sql**.

Ví dụ, bạn khai báo câu truy vấn động dạng SELECT có cú pháp tương tự như sau:

```
DECLARE @SQL NVARCHAR(500)  
DECLARE @TableName VARCHAR(50)
```

```

SET @TableName = 'Customers'
    SET @SQL = N'SELECT * FROM '
    SET @SQL = @SQL + @TableName + CHAR(13)
    PRINT @SQL
GO

```

Nếu thực thi đoạn chương trình trên, bạn có thể tìm thấy kết quả là phát biểu SQL dạng SELECT sẽ được thực thi có cú pháp như sau:

```
SELECT * FROM Customers
```

Với cách khai báo câu truy vấn theo hình thức trên, bạn có thể tạo ra phát biểu SQL có cấu trúc đặc biệt và sử dụng phát biểu EXECUTE (EXEC) hay thủ tục hệ thống sp\_executesql để thực thi chúng.

Chú ý, cú pháp của phát biểu EXECUTE dùng để thực thi chuỗi hay biến có cấu trúc câu SQL sẽ trình bày như sau:

```

{ EXEC | EXECUTE }
  ( { @string_variable | [ N ]'tsql_string' } [ + . . . n ] )
  [ AS { LOGIN | USER } = ' name ' ]
[ ; ]

```

Tuy nhiên, nếu chuỗi SQL là thủ tục nội tại hay hàm, bạn cần sử dụng cú pháp EXECUTE như sau:

```

{ EXEC | EXECUTE } ]
{
  { @return_status = }
  { module_name [ ;number ] | @module_name_var }
  [ [ @parameter = ] { value
    | @variable [ OUTPUT ]
    | [ DEFAULT ]
  }
  ]
  [ , . . . n ]
  [ WITH RECOMPILE ]
}
[ ; ]

```

Tương tự như vậy, bạn có thể sử dụng thủ tục nội tại hệ thống sp\_executesql thay vì phát biểu EXECUTE với cấu trúc như sau:

```

sp_executesql [ @stmt = ] stmt
[
  { , [ @params=] N'@parameter_name data_type [ [ OUT
  [ PUT ] [ , . . . n ] ]
  { , [ @param1 = ] 'value1' [ , . . . n ] }
]

```

Chẳng hạn, bạn có thể truyền tên bảng và cột dữ liệu vào thủ tục nội tại rồi sử dụng phát biểu EXEC để thực thi chúng như ví dụ 9-1.

**Chương 9: Phát biểu truy vấn dữ liệu nâng cao**

79

**Ví dụ 9-1: Khai báo câu truy vấn động**

```

CREATE PROC SPExecuteWithTableName
    @TableName VARCHAR(100),
    @ColumnName VARCHAR(100)
AS
    DECLARE @SQL NVARCHAR(500);
    SET @SQL = N'SELECT ' + @ColumnName + ' FROM ' + @TableName;
    SET @SQL = @SQL + CHAR(13);
    PRINT @SQL;
    EXEC (@SQL);
GO

```

Sau khi thực thi phát biểu CREATE PROC trong ví dụ trên, bạn khai báo để gọi thủ tục SPExecuteWithTableName như ví dụ 9-2.

**Ví dụ 9-2: Khai báo gọi SPExecuteWithTableName**

```

SPExecuteWithTableName 'Banks', '*'
GO

```

Nếu thực thi phát biểu trên, bạn sẽ tìm thấy danh sách mẫu tin trong bảng Banks như hình 9-1.

| BankID | BankName | BankAddress  | ProvinceID |
|--------|----------|--|------------|
| 1      | ACB      | Asia Commercial Bank<br>Lầu 2 - 30 Mạc Đĩnh Chi, Quận... | HCM        |
| 2      | SBA      | Southern Bank<br>1 Lê Lai                                | HCM        |
| 3      | UAB      | USA-Asia Commercial Bank<br>2 Le Duan St., Dist.1        | HCM        |

**Hình 9-1: Danh sách mẫu tin trong bảng Banks.**

Tuy nhiên, bạn cũng có thể sử dụng thủ tục hệ thống có tên sp\_executesql bằng cách khai báo tương tự như ví dụ 9-3.

**Ví dụ 9-3: Khai báo thực thi câu SQL động**

```

CREATE PROC SPExecuteSQLWithTableName
    @TableName VARCHAR(100),
    @ColumnName VARCHAR(100)
AS
    DECLARE @SQL NVARCHAR(500);
    SET @SQL = N'SELECT ' + @ColumnName + ' FROM ' + @TableName;
    SET @SQL = @SQL + CHAR(13);
    PRINT @SQL;
    EXEC sp_executesql @SQL;
GO

```

Sau khi thực thi phát biểu CREATE PROC trong ví dụ trên, bạn khai báo để gọi thủ tục SPExecuteSQLWithTableName như ví dụ 9-4.

**Ví dụ 9-4: Khai báo gọi SPExecuteWithTableName**

```
SPExecuteSQLWithTableName 'Customers',
'CustomerId,CompanyNameInVietnamese'
GO
```

Nếu thực thi phát biểu trên, bạn sẽ tìm thấy danh sách mẫu tin trong bảng Customers như hình 9-2.

| CustomerId | CompanyNameInVietnamese                       |
|------------|---|
| 1 A0001    | Công ty Trách Nhiệm Hữu Hạn Microsoft Vietnam |
| 2 A0002    | Công ty Trách Nhiệm Hữu Hạn IBM Vietnam       |
| 3 A0003    | Công ty Trách Nhiệm Hữu Hạn Kodaka Vietnam    |
| 4 A0004    | Công ty Trách Nhiệm Hữu Hạn E-Google Vietnam  |
| 5 A0005    | Công ty Cổ phần Suzumi Vietnam                |
| 6 A0006    | Tập đoàn UCIA USA                             |
| 7 A0007    | Công ty Đa quốc gia UFCA                      |
| 8 A0008    | Công ty Cổ phần ReruitVietnam                 |
| 9 A0009    | Trung tâm giáo dục Vietnam                    |
| 10 A0010   | Công ty Trách Nhiệm Hữu Hạn Hot Getways       |

Query executed successfully.

**Hình 9-2:** Danh sách mẫu tin trong bảng Customers.

Giả sử trong thực tế, những trường hợp cập nhật hay xóa thông tin khi người sử dụng chọn các CheckBox trên điều khiển lưới được thiết kế tương tự như hình 9-3.

| #                        | CompanyId | Company Name                          |
|--------------------------|-----------|---------------------------------------|
| <input type="checkbox"/> | A0001     | <u>Cong ty Saigon Pharma</u>          |
| <input type="checkbox"/> | A0002     | <u>Cong ty Saigon Insurance (GOV)</u> |
| <input type="checkbox"/> | A0003     | <u>Cong ty Saigon Insurance (PRI)</u> |

Add New    Delete    Print    Save to Excel

**Hình 9-3:** Danh sách khách hàng.

**Chương 9:** Phát biểu truy vấn dữ liệu nâng cao

81 M®

Nếu họ nhấn nút Delete sau khi chọn hai mã công ty như hình 9-4.

| COMPANY LIST                        |           |                                       |
|-------------------------------------|-----------|---------------------------------------|
| #                                   | CompanyId | Company Name                          |
| <input type="checkbox"/>            | A0001     | <u>Cong ty Saigon Pharma</u>          |
| <input checked="" type="checkbox"/> | A0002     | <u>Cong ty Saigon Insurance (GOV)</u> |
| <input checked="" type="checkbox"/> | A0003     | <u>Cong ty Saigon Insurance (PRI)</u> |

**Add New**    **Delete**    **Print**    **Save to Excel**

**Hình 9-4:** Chọn mã công ty để xóa.

Để xóa hai mẫu tin này, bạn có thể sử dụng phát biểu DELETE ứng với mã A0002 và A0003 (lấy ra từ điều khiển CheckBox) như ví dụ 9-5.

**Ví dụ 9-5: Khai báo xóa mẫu tin**

```
DELETE * FROM Customers
WHERE CustomerId in ('A0002', 'A0003')
```

Tuy nhiên, nếu bạn sử dụng thủ tục nội tại thay vì dùng phát biểu SQL dạng DELETE như ví dụ trên thì khai báo tham số ứng với chuỗi dạng nhiều mã khách hàng.

Tương tự như trường hợp trên, nếu bạn có nhu cầu chuyển trạng thái của phiếu nhập kho từ 1 sang giá trị 0 ứng với mã phiếu xuất chọn theo hình thức như hình 9-4.

**Ví dụ 9-6: Khai báo cập nhật phiếu nhập kho**

```
CREATE PROC SPExecuteSQLForImports
    @ImportNo VARCHAR(100)
AS
    DECLARE @SQL NVARCHAR(500);

    SET @SQL = N'Update Imports Set ImportDiscontinued = 0';
    SET @SQL = @SQL + N' WHERE ImportNo IN (' + CHAR(39);
    SET @SQL = @SQL + REPLACE(@ImportNo, ',', ',') + CHAR(39) + ')';
    PRINT @SQL;
    EXEC sp_executesql @SQL;
GO
```

Khi gọi thủ tục này từ cửa sổ MS, bạn có thể khai báo tương tự như ví dụ 9-7.

**Ví dụ 9-7: Khai báo gọi thủ tục nội tại SPExecuteSQLForImports**

```
SPExecuteSQLForImports 'IM0001, IM0002'
GO
```

Tương tự như vậy, bạn cũng có thể khai báo xóa danh sách mẫu tin trong bảng xuất kho ứng với mã đã chọn từ CheckBox như ví dụ 9-8.

**Ví dụ 9-8: Khai báo thủ tục nội tại xóa**

```
CREATE PROC SPExecuteForExports
    @ExportNo VARCHAR(100)
AS
    DECLARE @SQL NVARCHAR(500);

    SET @SQL = N'Delete From Exports '
    SET @SQL = @SQL + N' WHERE ExportNo IN (' + CHAR(39)
    SET @SQL = @SQL + REPLACE(@ExportNo,
        ', ', CHAR(39) + ',' + CHAR(39)) + CHAR(39) + ')';
    PRINT @SQL
    EXECUTE (@SQL);
GO
```

Chú ý: Bạn có thể sử dụng hàm REPLACE và CHAR để thay thế dấu phẩy thành hai dấu nháy đơn và dấu phẩy để bảo đảm chuỗi SQL động có cú pháp hợp lệ.

Nếu gọi thủ tục SPExecuteForExports hay SPExecuteSQLForImports từ ngôn ngữ lập trình C#, bạn có thể khai báo tương tự như ví dụ 9-9.

**Ví dụ 9-9: Khai báo gọi thủ tục nội tại bằng ngôn ngữ lập trình C#**

```
string exportNo = "EX0001, EX0002";
string sqlConnectionString = "server=(local); " +
    "database=AccountSystem; " +
    "Integrated Security=true";
using (SqlConnection sqlConnection = new
    SqlConnection(sqlConnectionString))
{
    sqlConnection.Open();
    SqlCommand sqlCommand = new SqlCommand();
    sqlCommand.CommandText =
        "SPExecuteForExports";
    sqlCommand.Connection = sqlConnection;
    sqlCommand.CommandType =
        CommandType.StoredProcedure;
    sqlCommand.Parameters.AddWithValue(
        "@ExportNo", exportNo);
    sqlCommand.ExecuteNonQuery();
}
```

**Chương 9: Phát biểu truy vấn dữ liệu nâng cao**

83

Trong trường hợp phát biểu SQL có sử dụng tham số, bạn có thể khai báo tương tự như ví dụ 9-9-1.

**Ví dụ 9-9-1: Khai báo sử dụng tham số**

```
EXECUTE sp_executesql
N'SELECT * FROM Customers
WHERE ProvinceId = @ProvinceId',
N'@ProvinceId Char(3)',
@ProvinceId = 'HCM';
```

Lưu ý: Để tham khảo chi tiết về đối tượng ADO.NET 2.0, bạn có thể tìm đọc C# 2005 - Tập 4 quyển 1: *Lập trình cơ sở dữ liệu* do nhà sách Minh Khai phát hành.

## 2. LÀM VIỆC VỚI NHIỀU MỆNH ĐỀ

### 2.1. Sử dụng phép toán UNION

Trong tập 1, bạn đã tìm hiểu chi tiết về biểu thức bảng với phát biểu WITH, phép toán UNION để kết hợp dữ liệu từ nhiều bảng khác nhau.

Với phép toán UNION, bạn có thể tổng hợp dữ liệu từ nhiều bảng nhằm kết xuất kết quả như ý thay vì tính toán nhiều bước. Một trong những ví dụ nổi bật nhất được sử dụng để tính công nợ thu, công nợ chi, tình hình tồn quỹ, xuất nhập tồn và ngay cả các tính toán liên quan đến thông tin đầu kỳ.

Chẳng hạn, có nhiều cách để tính tồn kho, nhưng bạn nên sử dụng sức mạnh của T-SQL trong SQL Server 2005 thay vì dùng ngôn ngữ lập trình khác.

Thủ hình dung giải pháp khi bạn muốn tính tình hình tồn kho trong doanh nghiệp có hệ thống nhiều kho hàng, cách suy nghĩ thông thường là lấy số lượng tồn kho đầu kỳ cộng với số lượng nhập trong kỳ trừ đi số lượng xuất trong kỳ, chúng ta sẽ có số lượng tồn kho cuối kỳ.

Do dữ liệu đang nằm trên ba bảng khác nhau là CloseInventoryControl (tồn kho đầu kỳ), ImportDetails (nhập trong kỳ), ExportDetails (xuất trong kỳ).

Thay vì phải xử lý trên ba bảng, bạn chỉ cần sử dụng phép toán UNION với từ ALL để tổng kết dữ liệu bằng biểu thức bảng như ví dụ 9-10.

**M<sup>®</sup> 84****Chương 9: Phát biểu truy vấn dữ liệu nâng cao****Ví dụ 9-10: Khai báo tổng kết dữ liệu**

WITH ImportAndExport

AS

(

-- Dữ liệu đầu kỳ

```
select ProductId, StockId,
EndQuantity AS BeginQuantity,
0 Import, 0 Export from CloseInventoryControl
where CloseMonth='09/2007'
```

-- Dữ liệu nhập trong kỳ

```
UNION ALL
select ProductId, StockId, 0,
SUM(Quantity) , 0
FROM ImportDetails
GROUP BY ProductId, StockId
```

-- Dữ liệu xuất trong kỳ

```
UNION ALL
select ProductId, StockId, 0,
0 Import, SUM(Quantity) AS Export
FROM ExportDetails
GROUP BY ProductId, StockId
```

)

-- Tổng kết dữ liệu cuối trong kỳ

```
SELECT StockId, P.ProductID,
ProductNameInVietnamese,
SUM(BeginQuantity) as BeginQtty,
SUM(Import) ImportQtty,
SUM(Export) ExportQtty,
SUM(BeginQuantity) +
SUM(Import) - SUM(Export) As BalanceQtty
FROM Products P, ImportAndExport IE
WHERE P.ProductID= IE.ProductID
GROUP BY StockId, P.ProductID,
ProductNameInVietnamese
GO
```

Nếu thực thi đoạn phát biểu trong ví dụ trên, bạn sẽ có kết quả ứng với tình hình tồn kho như hình 9-5.

**Chương 9: Phát biểu truy vấn dữ liệu nâng cao**

85



| StockId | ProductId | ProductNameInVietnamese        | BeginQty | ImportQty | ExportQty | BalanceQty |
|---------|-----------|--------------------------------|----------|-----------|-----------|------------|
| 1       | ST001     | Túi xách                       | 100      | 900       | 115       | 885        |
| 2       | ST002     | Túi xách                       | 150      | 600       | 400       | 350        |
| 3       | ST001     | Túi xách dùng cho học sinh nữ  | 150      | 1050      | 665       | 535        |
| 4       | ST002     | Túi xách dùng cho học sinh nữ  | 250      | 150       | 85        | 315        |
| 5       | ST001     | Túi xách dùng cho học sinh ... | 300      | 700       | 625       | 375        |
| 6       | ST002     | Túi xách dùng cho học sinh ... | 350      | 0         | 0         | 350        |
| 7       | ST001     | Túi áo mưa                     | 350      | 400       | 125       | 625        |
| 8       | ST002     | Túi áo mưa                     | 50       | 50        | 51        | 49         |
| 9       | ST001     | Túi xách dùng cho Máy tính     | 150      | 200       | 125       | 225        |
| 10      | ST002     | Túi xách dùng cho Máy tính     | 0        | 70        | 5         | 65         |
| 11      | ST005     | Túi xách dùng cho Máy tính     | 150      | 0         | 0         | 150        |
| 12      | ST001     | Túi xách dùng cho Điện thoại   | 450      | 0         | 0         | 450        |
| 13      | ST002     | Túi xách dùng cho Điện thoại   | 0        | 350       | 90        | 260        |
| 14      | ST005     | Túi xách dùng cho Điện thoại   | 10       | 0         | 0         | 10         |
| 15      | ST001     | Túi xách dùng cho PC           | 500      | 0         | 0         | 500        |
| 16      | ST002     | Túi xách dùng cho PC           | 0        | 900       | 0         | 900        |
| 17      | ST005     | Túi xách dùng cho PC           | 650      | 0         | 0         | 650        |

**Hình 9-5: Tình hình tồn kho.**

**Chú ý:** Cách lưu và tính toán tồn kho theo giải pháp trên chỉ tối ưu khi số lượng mẫu tin trong cơ sở dữ liệu có giới hạn vài ngàn.

Nếu số lượng mẫu tin lớn đến hàng triệu thì bạn không thể sử dụng cách lưu trữ và tính toán như trên, thay vào đó bạn cần tổ chức lại cách tính tổng số lượng nhập, xuất mỗi khi hai nghiệp vụ này phát sinh.

## 2.2. Mệnh đề JOIN với phát biểu SELECT

Bạn cũng tìm hiểu cách khai báo mệnh đề INNER JOIN và LEFT JOIN giữa các bảng dữ liệu có quan hệ với nhau. Chẳng hạn, trong trường hợp tổng hợp doanh thu bán hàng, bạn khai báo như ví dụ 9-11.

### Ví dụ 9-11: Khai báo doanh thu bán hàng

```

SELECT C.CustomerID, CompanyNameInVietnamese,
SUM(Quantity*Price*(1+VATRate/100)-Discount)
As SalesAmount
FROM Customers C
LEFT JOIN SalesInvoices S
ON C.CustomerID = S.CustomerID
LEFT JOIN SalesInvoiceDetails D
ON S.InvoiceNo = D.InvoiceNo
GROUP BY C.CustomerID, CompanyNameInVietnamese
GO
    
```

Giả sử chúng ta có  $N$  khách hàng trong bảng Customers, i hóa đơn bán hàng trong bảng SalesInvoices và j mẫu tin ứng với chi tiết hóa đơn bán hàng trong bảng SalesInvoiceDetails.

Nếu thực thi phát biểu SELECT với mệnh đề LEFT JOIN trong ví dụ trên, chúng ta sẽ có j mẫu tin cộng thêm số lượng mẫu tin trong  $N$  có mã không tồn tại trong i mẫu tin, kết quả trình bày như hình 9-6.

|    | CustomerID | CompanyName<br>in Vietnamese                    | SalesQuantity | SalesAmount    |
|----|------------|---|---------------|----------------|
| 1  | A0001      | Công ty Trách Nhiệm Hữu Hạn Macrosocial Vietnam | 710           | 8976250.000000 |
| 2  | A0002      | Công ty Trách Nhiệm Hữu Hạn IBM Vietnam         | 350           | 3832500.000000 |
| 3  | A0003      | Công ty Trách Nhiệm Hữu Hạn Kodaka Vietnam      | 400           | 4300000.000000 |
| 4  | A0004      | Công ty Trách Nhiệm Hữu Hạn E-Google Vietnam    | 370           | 5670000.000000 |
| 5  | A0005      | Công ty Cổ phần Suzumi Vietnam                  | 181           | 2397650.000000 |
| 6  | A0006      | Tập đoàn UCIA USA                               | 150           | 2185000.000000 |
| 7  | A0007      | Công ty Đa quốc gia UFCACI                      | 70            | 979500.000000  |
| 8  | A0008      | Công ty Cổ phần RecruitVietnam                  | 55            | 726250.000000  |
| 9  | A0009      | Trung tâm giáo dục Vietnam                      | NULL          | NULL           |
| 10 | A0010      | Công ty Trách Nhiệm Hữu Hạn Hot Getways         | NULL          | NULL           |

Hình 9-6: Doanh thu bán hàng.

Thông thường, bạn sử dụng mệnh đề JOIN giữa hai bảng, trong trường hợp này bạn cũng có thể áp dụng mệnh đề JOIN ứng với phát biểu SELECT khác.

Chẳng hạn, bạn khai báo doanh thu bán hàng từ hai bảng có tên SalesInvoices và SalesInvoiceDetails như ví dụ 9-12.

#### Ví dụ 9-12: Khai báo doanh thu bán hàng

```
SELECT ProductId, SUM(Quantity) AS Quantity,
       SUM(Quantity*Price*(1
+VATRate/100)-Discount) AS SalesAmount
  FROM SalesInvoices S
 INNER JOIN SalesInvoiceDetails D
    ON S.InvoiceNo = D.InvoiceNo
 GROUP BY ProductId
 GO
```

Nếu thực thi phát biểu SELECT trên, bạn có thể tìm thấy danh sách ứng với số lượng và số tiền bán như hình 9-7.

**Chương 9:** Phát biểu truy vấn dữ liệu nâng cao

87 M®

| ProductID | Quantity | SalesAmount    |
|-----------|----------|----------------|
| 1 P00001  | 515      | 7806250.000000 |
| 2 P00002  | 750      | 8667500.000000 |
| 3 P00003  | 625      | 7213750.000000 |
| 4 P00004  | 176      | 2245150.000000 |
| 5 P00005  | 130      | 1835000.000000 |
| 6 P00006  | 90       | 1299500.000000 |

**Hình 9-7:** Doanh thu bán hàng.

Tuy nhiên, bạn cũng có thể khai báo mệnh đề LEFT JOIN với phát biểu SELECT trong ví dụ 9-12 như ví dụ 9-13.

**Ví dụ 9-13: Khai báo mệnh đề LEFT JOIN với phát biểu SELECT**

```
SELECT C.ProductID, ProductNameInVietnamese,
Quantity, SalesAmount
FROM Products C LEFT JOIN
(Select ProductID, SUM(Quantity) As Quantity,
SUM(Quantity*Price*(1
+VATRate/100)-Discount) AS SalesAmount
from SalesInvoices S
INNER JOIN SalesInvoiceDetails D
ON S.InvoiceNo = D.InvoiceNo
GROUP BY ProductID ) M
ON C.ProductID = M.ProductID
GO
```

Nếu thực thi phát biểu SELECT trên, bạn có thể tìm thấy danh sách mã sản phẩm ứng với số lượng và số tiền bán như hình 9-8.

| ProductID | ProductNameInVietnamese          | Quantity | SalesAmount    |
|-----------|----------------------------------|----------|----------------|
| 1 P00001  | Túi xách                         | 515      | 7806250.000000 |
| 2 P00002  | Túi xách dùng cho học sinh nữ    | 750      | 8667500.000000 |
| 3 P00003  | Túi xách dùng cho học sinh nam   | 625      | 7213750.000000 |
| 4 P00004  | Túi áo mưa                       | 176      | 2245150.000000 |
| 5 P00005  | Túi xách dùng cho Máy tính       | 130      | 1835000.000000 |
| 6 P00006  | Túi xách dùng cho Điện thoại ... | 90       | 1299500.000000 |
| 7 P00007  | Túi xách dùng cho PC             | NULL     | NULL           |
| 8 P00008  | Túi xách dùng cho TV             | NULL     | NULL           |

**Hình 9-8:** Áp dụng mệnh đề JOIN với phát biểu SELECT.

### 2.3. Sử dụng hàm CASE

Khi làm việc với trang tìm kiếm dữ liệu, do chúng ta cung cấp nhiều tiêu chí tìm kiếm, nên bạn nên sử dụng hàm CASE để xét trường giá trị ứng với tiêu chí tìm kiếm mà người dùng không cung cấp.

Chú ý: Bạn có thể tìm hiểu hai cú pháp của hàm CASE trong chương kế tiếp.

Giả sử bạn có trang tìm kiếm khách hàng và hóa đơn bán hàng với bốn tùy chọn InvoiceNo, InvoiceDate, CustomerId và CustomerName như hình 9-9.

Hàm CASE: Đầu vào và kết quả

|               |                      |
|---------------|----------------------|
| CustomerId:   | <input type="text"/> |
| CustomerName: | <input type="text"/> |
| InvoiceNo:    | <input type="text"/> |
| InvoiceDate:  | <input type="text"/> |
| To:           | <input type="text"/> |
| <b>SEARCH</b> |                      |

**Hình 9-9: Trang tìm kiếm.**

Với giao diện tìm kiếm như trên, bạn cần có phát biểu SQL dạng tìm kiếm với phép toán AND cho các tiêu chí tìm kiếm thì khai báo tương tự như ví dụ 9-14.

#### Ví dụ 9-14: Khai báo phép toán WHERE và AND

WHERE Discontinued=0

AND S.CustomerID =

```
CASE @CustomerID
  WHEN '' THEN S.CustomerID
  ELSE @CustomerID END
```

AND CompanyNameInVietnamese LIKE '%' +

```
CASE @CustomerName
  WHEN '' THEN @CustomerName
  ELSE @CustomerName END + '%'
```

AND S.InvoiceNo =

```
CASE @InvoiceNo
  WHEN '' THEN S.InvoiceNo
  ELSE @InvoiceNo END
```

AND S.DueDate >= CAST(

```
CASE @InvoiceDateFrom
  WHEN '' THEN '1/1/2000'
```

**Chương 9: Phát biểu truy vấn dữ liệu nâng cao**

89

```

ELSE @InvoiceDateFrom END
AS SMALLDATETIME)
AND S.DueDate <= CAST(
CASE @InvoiceDateTo
WHEN '' THEN '1/1/2079'
ELSE @InvoiceDateTo END
AS SMALLDATETIME)
GO

```

Trong đó, các biến tương ứng với tiêu chí tìm kiếm được khai báo như biến hay tham số nếu tạo thủ tục nội tại.

**Ví dụ 9-15: Khai báo biến**

-- Khai báo biến

```

DECLARE @CustomerID VARCHAR(10)
DECLARE @CustomerName NVARCHAR(150)
DECLARE @InvoiceNo VARCHAR(10)
DECLARE @InvoiceDateFrom VARCHAR(15)
DECLARE @InvoiceDateTo VARCHAR(15)

```

-- Gán giá trị cho biến

```

SET @CustomerID = ''
SET @CustomerName = N'%Cô phàn%'
SET @InvoiceNo = ''
SET @InvoiceDateFrom = ''
SET @InvoiceDateTo = ''

```

Sau đó, bạn khai báo phát biểu SELECT với mệnh đề JOIN giữa ba bảng dữ liệu liên quan là Customers, SalesInvoices và SalesInvoiceDetails.

**Ví dụ 9-16: Khai báo phát biểu SELECT**

```

DECLARE @CustomerID VARCHAR(10)
DECLARE @CustomerName NVARCHAR(150)
DECLARE @InvoiceNo VARCHAR(10)
DECLARE @InvoiceDateFrom VARCHAR(15)
DECLARE @InvoiceDateTo VARCHAR(15)
SET @CustomerID = ''
SET @CustomerName = N'%Cô phàn%'
SET @InvoiceNo = ''
SET @InvoiceDateFrom = ''
SET @InvoiceDateTo = ''
SELECT C.CustomerID, CompanyNameInVietnamese,
S.InvoiceNo, Quantity*Price AS Amount, Discount,
Quantity*Price*(1+VATRate/100)-Discount
As SalesAmount
FROM Customers C INNER JOIN SalesInvoices S
ON C.CustomerID = S.CustomerID
INNER JOIN SalesInvoiceDetails D

```

M® 90

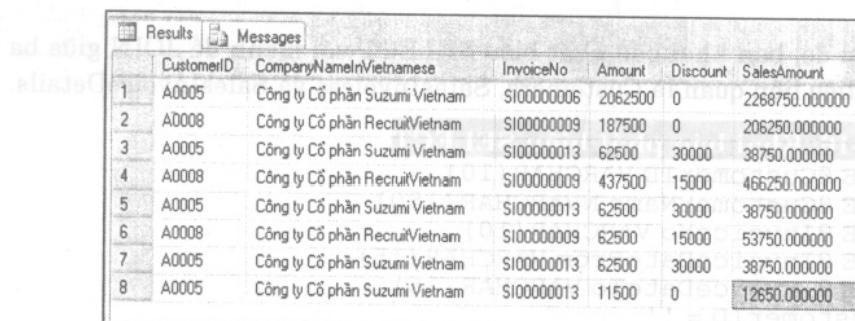
**Chương 9:** Phát biểu truy vấn dữ liệu nâng cao

```

ON S.InvoiceNo = D.InvoiceNo
WHERE Discontinued = 0
AND S.CustomerID =
CASE @CustomerID
WHEN '' THEN S.CustomerID
ELSE @CustomerID END
AND CompanyNameInVietnamese LIKE
CASE @CustomerName
WHEN '' THEN @CustomerName
ELSE @CustomerName END
AND S.InvoiceNo =
CASE @InvoiceNo
WHEN '' THEN S.InvoiceNo
ELSE @InvoiceNo END
AND S.DueDate >= CAST(
CASE @InvoiceDateFrom
WHEN '' THEN '1/1/2000'
ELSE @InvoiceDateFrom END
AS SMALLDATETIME)
AND S.DueDate <= CAST(
CASE @InvoiceDateTo
WHEN '' THEN '1/1/2079'
ELSE @InvoiceDateTo END
AS SMALLDATETIME)
GO

```

Bạn có thể gọi phát biểu SELECT trên, kết quả sẽ trình bày như hình 9-11.



|   | CustomerID | CompanyNameInVietnamese        | InvoiceNo | Amount  | Discount | SalesAmount    |
|---|------------|--------------------------------|-----------|---------|----------|----------------|
| 1 | A0005      | Công ty Cổ phần Suzumi Vietnam | S10000006 | 2062500 | 0        | 2268750.000000 |
| 2 | A0008      | Công ty Cổ phần RecruitVietnam | S10000009 | 187500  | 0        | 206250.000000  |
| 3 | A0005      | Công ty Cổ phần Suzumi Vietnam | S10000013 | 62500   | 30000    | 38750.000000   |
| 4 | A0008      | Công ty Cổ phần RecruitVietnam | S10000009 | 437500  | 15000    | 466250.000000  |
| 5 | A0005      | Công ty Cổ phần Suzumi Vietnam | S10000013 | 62500   | 30000    | 38750.000000   |
| 6 | A0008      | Công ty Cổ phần RecruitVietnam | S10000009 | 62500   | 15000    | 53750.000000   |
| 7 | A0005      | Công ty Cổ phần Suzumi Vietnam | S10000013 | 62500   | 30000    | 38750.000000   |
| 8 | A0005      | Công ty Cổ phần Suzumi Vietnam | S10000013 | 11500   | 0        | 12650.000000   |

**Hình 9-11: Gọi phát biểu SQL.**

Để sử dụng tốt phát biểu SQL của ví dụ trên trong SQL Server, bạn nên khai báo chúng như một thủ tục nội tại.

**Ví dụ 9-17: Khai báo thủ tục nội tại**

```

CREATE PROC SearchSalesInvoice
@CustomerID VARCHAR(10),
@CustomerName NVARCHAR(150),
@InvoiceNo VARCHAR(10),

```

**Chương 9: Phát biểu truy vấn dữ liệu nâng cao**

91

```

@InvoiceDateFrom VARCHAR(15),
@InvoiceDateTo VARCHAR(15)
AS
    SELECT C.CustomerID, CompanyNameInVietnamese,
    S.InvoiceNo, Quantity * Price AS Amount, Discount,
    Quantity*Price*(1+VATRate/100)-Discount
    As SalesAmount
    FROM Customers C INNER JOIN SalesInvoices S
    ON C.CustomerID = S.CustomerID
    INNER JOIN SalesInvoiceDetails D
    ON S.InvoiceNo = D.InvoiceNo
    WHERE Discontinued=0
    AND S.CustomerID =
        CASE @CustomerID
        WHEN '' THEN S.CustomerID
        ELSE @CustomerID END
    AND CompanyNameInVietnamese LIKE
        CASE @CustomerName
        WHEN '' THEN @CustomerName
        ELSE @CustomerName END
    AND S.InvoiceNo =
        CASE @InvoiceNo
        WHEN '' THEN S.InvoiceNo
        ELSE @InvoiceNo END
    AND S.DueDate >= CAST(
        CASE @InvoiceDateFrom
        WHEN '' THEN '1/1/2000'
        ELSE @InvoiceDateFrom END
        AS SMALLDATETIME)
    AND S.DueDate <= CAST(
        CASE @InvoiceDateTo
        WHEN '' THEN '1/1/2079'
        ELSE @InvoiceDateTo END
        AS SMALLDATETIME)
GO

```

Chú ý: Để tham khảo chi tiết về thủ tục nội tại, bạn có thể đọc chương

12.

Khi triệu gọi thủ tục nội tại trên, bạn có thể truyền các giá trị nhập trên màn hình vào các tham số tương ứng như ví dụ 9-18.

**Ví dụ 9-18: Khai báo triệu gọi thủ tục nội tại**

```

SearchSalesInvoice '',N'%Cô phèn%', 'SI00000009', '', ''
GO

```

Nếu thực thi thủ tục nội tại trên, bạn có thể tìm thấy kết quả như hình 9-12.

## Chương 9: Phát biểu truy vấn dữ liệu nâng cao

|   | CustomerID | CompanyNameInVietnamese        | InvoiceNo  | Amount | Discount | SalesAmount   |
|---|------------|--------------------------------|------------|--------|----------|---------------|
| 1 | A0008      | Công ty Cổ phần RecruitVietnam | SI00000009 | 187500 | 0        | 206250.000000 |
| 2 | A0008      | Công ty Cổ phần RecruitVietnam | SI00000009 | 437500 | 15000    | 466250.000000 |
| 3 | A0008      | Công ty Cổ phần RecruitVietnam | SI00000009 | 62500  | 15000    | 53750.000000  |

**Hình 9-12:** Gọi thủ tục tìm kiếm.

Trong trường hợp bạn muốn tìm kiếm theo từ khóa chung cho 3 tiêu chí CustomerId, CompanyName và InvoiceNo như hình 9-13.

Keyword:

**SEARCH**

**Hình 9-13:** Trang tìm kiếm theo từ khóa.

Đối với trường hợp này, bạn khai báo thủ tục trên sẽ được như ví dụ 9-19.

**Ví dụ 9-19: Khai báo thủ tục tìm kiếm theo từ khóa**

```
CREATE PROC SearchSalesInvoiceByKeyword
@Keyword NVARCHAR(150)
AS
SELECT C.CustomerID, CompanyNameInVietnamese,
S.InvoiceNo, Quantity * Price AS Amount, Discount,
Quantity*Price*(1+VATRate/100)-Discount
As SalesAmount
FROM Customers C INNER JOIN SalesInvoices S
ON C.CustomerID = S.CustomerID
INNER JOIN SalesInvoiceDetails D
ON S.InvoiceNo = D.InvoiceNo
WHERE Discontinued=0
AND
(S.CustomerID LIKE @Keyword
OR CompanyNameInVietnamese LIKE @Keyword
OR S.InvoiceNo LIKE @Keyword
)
GO
```

Khi triệu gọi thủ tục trên, bạn có thể truyền các giá trị nhập trên màn hình vào các tham số tương ứng như ví dụ 9-20.

**Ví dụ 9-20: Khai báo triệu gọi thủ tục nội tại**

```
SearchSalesInvoiceByKeyword N'%Đa quốc gia%'
GO
```

**Chương 9: Phát biểu truy vấn dữ liệu nâng cao**

93

Nếu thực thi thủ tục nội tại trên, bạn có thể tìm thấy kết quả như hình 9-13-1.

| CustomerID | CompanyNameInVietnamese  | InvoiceNo  | Amount | Discount | SalesAmount   |
|------------|--------------------------|------------|--------|----------|---------------|
| 1 A0007    | Công ty Đa quốc gia UFCA | SI00000008 | 507500 | 30000    | 528250.000000 |
| 2 A0007    | Công ty Đa quốc gia UFCA | SI00000008 | 437500 | 30000    | 451250.000000 |

**Hình 9-13-1: Gọi thủ tục tìm kiếm theo từ khóa.**

Nếu bạn triệu gọi thủ tục nội tại trên với mã khách hàng là A0002 thì khai báo như ví dụ 9-21.

**Ví dụ 9-21: Khai báo tìm kiếm theo mã khách hàng**

```
SearchSalesInvoiceByKeyword 'A0002'
GO
```

Nếu thực thi khai báo trên thì kết quả trình bày như hình 9-14.

| CustomerID | CompanyNameInVietnamese                 | InvoiceNo  | Quantity | Price | Discount | SalesAmount    |
|------------|---|------------|----------|-------|----------|----------------|
| 1 A0002    | Công ty Trách Nhiệm Hữu Hạn IBN Vietnam | SI00000002 | 100      | 10000 | 100000   | 1000000.000000 |
| 2 A0002    | Công ty Trách Nhiệm Hữu Hạn IBN Vietnam | SI00000011 | 25       | 10500 | 0        | 288750.000000  |
| 3 A0002    | Công ty Trách Nhiệm Hữu Hạn IBN Vietnam | SI00000002 | 200      | 10000 | 0        | 2200000.000000 |
| 4 A0002    | Công ty Trách Nhiệm Hữu Hạn IBN Vietnam | SI00000011 | 25       | 12500 | 0        | 343750.000000  |

**Hình 9-14: Tìm kiếm theo mã công ty.**

Nếu bạn triệu gọi thủ tục nội tại trên với mã số hóa đơn bán hàng là SI00000013 thì khai báo như ví dụ 9-22.

**Ví dụ 9-22: Khai báo tìm kiếm theo mã số hóa đơn bán hàng**

```
SearchSalesInvoiceByKeyword 'SI00000013'
GO
```

Nếu thực thi khai báo trên thì kết quả trình bày như hình 9-15.

| CustomerID | CompanyNameInVietnamese        | InvoiceNo  | Amount | Discount | SalesAmount  |
|------------|--------------------------------|------------|--------|----------|--------------|
| 1 A0005    | Công ty Cổ phần Suzumi Vietnam | SI00000013 | 62500  | 30000    | 38750.000000 |
| 2 A0005    | Công ty Cổ phần Suzumi Vietnam | SI00000013 | 62500  | 30000    | 38750.000000 |
| 3 A0005    | Công ty Cổ phần Suzumi Vietnam | SI00000013 | 62500  | 30000    | 38750.000000 |
| 4 A0005    | Công ty Cổ phần Suzumi Vietnam | SI00000013 | 11500  | 0        | 12650.000000 |

**Hình 9-15: Tìm kiếm theo mã số hóa đơn bán hàng.**

### 3. PHÉP TOÁN PIVOT VÀ UNPIVOT

Bạn có thể sử dụng phép toán quan hệ PIVOT và UNPIVOT để chế tác dữ liệu từ giá trị của biểu thức bảng vào bảng dữ liệu khác.

### 3.1. Phép toán PIVOT

Phép toán PIVOT cho phép luân phiên giá trị của biểu thức bảng bằng cách thống nhất giá trị từ một cột trong biểu thức thành nhiều cột khi kết xuất dữ liệu và thực hiện quá trình kết hợp cho những cột dữ liệu còn lại tương tự như mệnh đề GROUP BY.

Chẳng hạn, chúng ta tổng hợp danh sách mẫu tin trong bảng ImportDetails như ví dụ 9-23.

#### Ví dụ 9-23: Khai báo tổng hợp số lượng nhập

```
SELECT StockId, ProductId,
SUM(Quantity) AS TotalQuantity
FROM ImportDetails
GROUP BY StockId, ProductId
ORDER BY StockId, ProductId
GO
```

Nếu thực thi phát biểu SELECT với mệnh đề GROUP BY trên, bạn sẽ tìm thấy dữ liệu xuất kho như hình 9-16.

|    | StockId | ProductId | TotalQuantity |
|----|---------|-----------|---------------|
| 1  | ST001   | P00001    | 900           |
| 2  | ST001   | P00002    | 1050          |
| 3  | ST001   | P00003    | 700           |
| 4  | ST001   | P00004    | 400           |
| 5  | ST001   | P00005    | 200           |
| 6  | ST002   | P00001    | 600           |
| 7  | ST002   | P00002    | 150           |
| 8  | ST002   | P00004    | 50            |
| 9  | ST002   | P00005    | 70            |
| 10 | ST002   | P00006    | 350           |
| 11 | ST002   | P00007    | 900           |

Hình 9-16: Tổng hợp mẫu tin trong bảng ImportDetails.

Nếu bạn muốn tổng hợp dữ liệu nhập kho theo hai chiều ứng với mã sản phẩm (chiều dọc) và mã kho (chiều ngang) thì sử dụng phép toán PIVOT như ví dụ 9-24.

#### Ví dụ 9-24: Khai báo phép toán PIVOT

```
SELECT ProductId, [ST001] AS Stock1,
[ST002] AS Stock2, [ST003] AS Stock3
FROM (SELECT StockId, ProductId, Quantity
```

**Chương 9: Phát biểu truy vấn dữ liệu nâng cao**

95

```
FROM ImportDetails) p
PIVOT
(
    SUM (Quantity)
    FOR StockId IN
    ( [ST001], [ST002], [ST003] )
)
AS pvt
ORDER BY ProductId
GO
```

Trong đó, bạn sử dụng hàm SUM để tính tổng số lượng nhập theo từng mã sản phẩm cho từng mã kho.

Khi thực thi phát biểu SELECT với phép toán PIVOT trên, bạn có thể tìm thấy kết quả số lượng nhập phân bố như hình 9-17.

| ProductId | Stock1 | Stock2 | Stock3 |
|-----------|--------|--------|--------|
| 1 P00001  | 900    | 600    | NULL   |
| 2 P00002  | 1050   | 150    | NULL   |
| 3 P00003  | 700    | NULL   | NULL   |
| 4 P00004  | 400    | 50     | NULL   |
| 5 P00005  | 200    | 70     | NULL   |
| 6 P00006  | NULL   | 350    | NULL   |
| 7 P00007  | NULL   | 900    | NULL   |

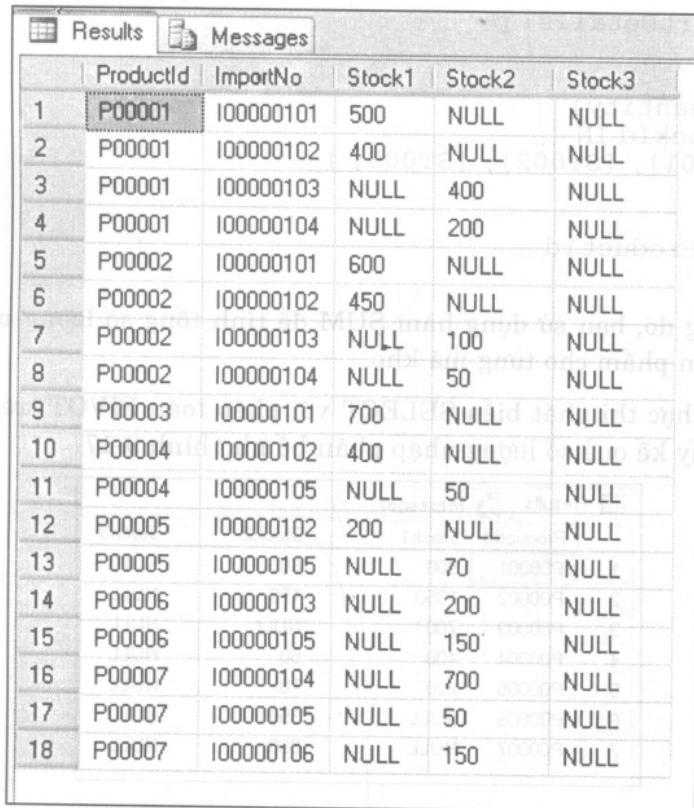
Hình 9-17: Sử dụng hàm PIVOT.

Trong trường hợp bạn muốn liệt kê cột mã phiếu nhập thì khai báo như ví dụ 9-25.

**Ví dụ 9-25: Khai báo liệt kê nhiều cột dữ liệu**

```
SELECT ProductId, ImportNo, [ST001] AS Stock1,
[ST002] AS Stock2, [ST003] AS Stock3
FROM (SELECT ImportNo, StockId, ProductId, Quantity
FROM ImportDetails) p
PIVOT
(
    SUM (Quantity)
    FOR StockId IN
    ( [ST001], [ST002], [ST003] )
)
AS pvt
ORDER BY ProductId
GO
```

Khi thực thi phát biểu SELECT với phép toán PIVOT trên, bạn có thể tìm thấy kết quả số lượng nhập phân bố như hình 9-18.



|    | ProductId | ImportNo  | Stock1 | Stock2 | Stock3 |
|----|-----------|-----------|--------|--------|--------|
| 1  | P00001    | I00000101 | 500    | NULL   | NULL   |
| 2  | P00001    | I00000102 | 400    | NULL   | NULL   |
| 3  | P00001    | I00000103 | NULL   | 400    | NULL   |
| 4  | P00001    | I00000104 | NULL   | 200    | NULL   |
| 5  | P00002    | I00000101 | 600    | NULL   | NULL   |
| 6  | P00002    | I00000102 | 450    | NULL   | NULL   |
| 7  | P00002    | I00000103 | NULL   | 100    | NULL   |
| 8  | P00002    | I00000104 | NULL   | 50     | NULL   |
| 9  | P00003    | I00000101 | 700    | NULL   | NULL   |
| 10 | P00004    | I00000102 | 400    | NULL   | NULL   |
| 11 | P00004    | I00000105 | NULL   | 50     | NULL   |
| 12 | P00005    | I00000102 | 200    | NULL   | NULL   |
| 13 | P00005    | I00000105 | NULL   | 70     | NULL   |
| 14 | P00006    | I00000103 | NULL   | 200    | NULL   |
| 15 | P00006    | I00000105 | NULL   | 150    | NULL   |
| 16 | P00007    | I00000104 | NULL   | 700    | NULL   |
| 17 | P00007    | I00000105 | NULL   | 50     | NULL   |
| 18 | P00007    | I00000106 | NULL   | 150    | NULL   |

Hình 9-18: Trình bày hai cột dữ liệu.

Nếu có nhu cầu sử dụng biểu thức bảng với phát biểu WITH thì khai báo như ví dụ 9-26.

#### Ví dụ 9-26: Khai báo biểu thức bảng

```
WITH PivotTable
AS
(
    SELECT ProductId, [ST001] AS Stock1, [ST002] AS Stock2, [ST003] AS Stock3
    FROM (SELECT StockId, ProductId, Quantity
    FROM ImportDetails) p
    PIVOT
    (
        SUM (Quantity)
        FOR StockId IN
        ( [ST001], [ST002], [ST003] )
    )
    AS PvtTable
)
SELECT ProductId,
```

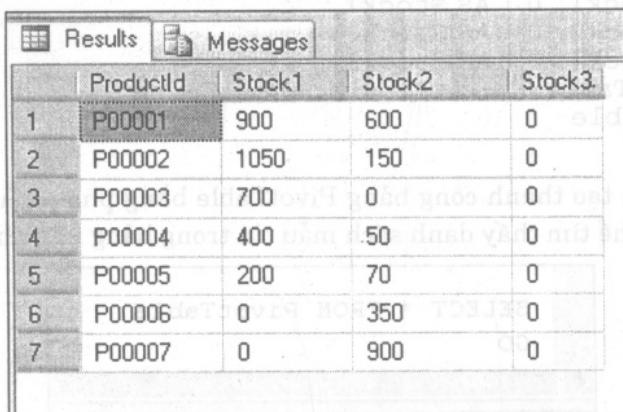
**Chương 9: Phát biểu truy vấn dữ liệu nâng cao**97 

```

ISNULL(Stock1, 0) AS Stock1,
ISNULL(Stock2, 0) AS Stock2,
ISNULL(Stock3, 0) AS Stock3
FROM PivotTable
GO

```

Khi thực thi phát biểu SELECT với phép toán PIVOT trên, bạn có thể tìm thấy kết quả số lượng nhập phân bố như hình 9-19.



|   | ProductId | Stock1 | Stock2 | Stock3 |
|---|-----------|--------|--------|--------|
| 1 | P00001    | 900    | 600    | 0      |
| 2 | P00002    | 1050   | 150    | 0      |
| 3 | P00003    | 700    | 0      | 0      |
| 4 | P00004    | 400    | 50     | 0      |
| 5 | P00005    | 200    | 70     | 0      |
| 6 | P00006    | 0      | 350    | 0      |
| 7 | P00007    | 0      | 900    | 0      |

Hình 9-19: Số lượng nhập cho mỗi kho.

### 3.2. Phép toán UNPIVOT

Trong khi hàm PIVOT thực hiện quá trình luân phiên dữ liệu theo nhóm từ bảng này sang bảng khác thì hàm UNPIVOT thực hiện quá trình ngược lại với hàm PIVOT bằng cách luân chuyển giá trị của cột bảng thành cột giá trị.

Nói cách khác, hàm UNPIVOT đảo nghịch hàm PIVOT bằng cách chuyển cột thành hàng.

Giả sử, chúng ta tạo ra bảng dữ liệu có cấu trúc như bảng đã sử dụng hàm PIVOT bằng cách thực thi phát biểu SELECT với mệnh đề INTO như ví dụ 9-27.

#### Ví dụ 9-27: Khai báo tạo bảng dữ liệu

```

WITH PvtTable
As
(
    SELECT ProductId, [ST001] AS Stock1,
           [ST002] AS Stock2, [ST003] AS Stock3
    FROM (SELECT StockId, ProductId, Quantity
          FROM ImportDetails) p
    PIVOT

```

M® 98

**Chương 9:** Phát biểu truy vấn dữ liệu nâng cao

```

(
    SUM (Quantity)
    FOR StockId IN
    ( [ST001], [ST002], [ST003] )
)
AS PvtTable
) a nhanh TOWER phai gian luy TOWER vao tinh tinh id cua
SELECT ProductId, ISNULL(Stock1, 0 ) AS Stock1,
ISNULL(Stock2, 0 ) AS Stock2,
ISNULL(Stock3, 0 ) AS Stock3
INTO PivotTable
FROM PvtTable
GO

```

Sau khi tạo thành công bảng PivotTable bằng phát biểu trong ví dụ trên, bạn có thể tìm thấy danh sách mẫu tin trong bảng này như hình 9-20.

|   | ProductId | Stock1 | Stock2 | Stock3 |
|---|-----------|--------|--------|--------|
| 1 | P00001    | 900    | 900    | 900    |
| 2 | P00002    | 1050   | 1050   | 1050   |
| 3 | P00003    | 700    | 700    | 700    |
| 4 | P00004    | 400    | 400    | 400    |
| 5 | P00005    | 200    | 200    | 200    |
| 6 | P00006    | 0      | 0      | 0      |
| 7 | P00007    | 0      | 0      | 0      |

**Hình 9-20:** Danh sách mẫu tin trong bảng PivotTable.

Nếu bạn khai báo hàm UNPIVOT để chuyển dữ liệu trong cột Stock1, Stock2 và Stock3 thành hàng thì khai báo như ví dụ 9-28.

**Ví dụ 9-28: Khai báo hàm UNPIVOT**

```

SELECT ProductId, StockId, Imports
FROM
    (SELECT *
     FROM PivotTable) p
UNPIVOT
    (Imports FOR StockId IN
        (Stock1, Stock2, Stock3))

```

**Chương 9:** Phát biểu truy vấn dữ liệu nâng cao

```
) AS UnpivotTable
GO
```

Khi thực thi phát biểu SELECT với hàm UNPIVOT, bạn có thể tìm thấy danh sách mẫu tin trong bảng này như hình 9-21.

|    | ProductId | StockId | Imports |
|----|-----------|---------|---------|
| 1  | P00001    | Stock1  | 900     |
| 2  | P00001    | Stock2  | 900     |
| 3  | P00001    | Stock3  | 900     |
| 4  | P00002    | Stock1  | 1050    |
| 5  | P00002    | Stock2  | 1050    |
| 6  | P00002    | Stock3  | 1050    |
| 7  | P00003    | Stock1  | 700     |
| 8  | P00003    | Stock2  | 700     |
| 9  | P00003    | Stock3  | 700     |
| 10 | P00004    | Stock1  | 400     |
| 11 | P00004    | Stock2  | 400     |
| 12 | P00004    | Stock3  | 400     |
| 13 | P00005    | Stock1  | 200     |
| 14 | P00005    | Stock2  | 200     |
| 15 | P00005    | Stock3  | 200     |
| 16 | P00006    | Stock1  | 0       |
| 17 | P00006    | Stock2  | 0       |
| 18 | P00006    | Stock3  | 0       |
| 19 | P00007    | Stock1  | 0       |
| 20 | P00007    | Stock2  | 0       |
| 21 | P00007    | Stock3  | 0       |

Hình 9-21: Hàm UNPIVOT.

## 4. KẾT CHƯƠNG

Chúng ta vừa tìm hiểu cách khai báo và thực thi phát biểu SQL động bằng thủ tục hệ thống sp\_executesql và phát biểu EXECUTE.

 100

## Chương 9: Phát biểu truy vấn dữ liệu nâng cao

Bạn cũng tìm hiểu một số phát biểu SQL dạng phức tạp như kết hợp mệnh đề JOIN với tập dữ liệu tạo ra từ phát biểu SELECT thay vì TABLE hay VIEW.

Ngoài ra, bạn cũng tìm hiểu cách luân chuyển dữ liệu từ bảng giá trị thành cột bằng hàm PIVOT và cách chuyển cột dữ liệu thành hàng bằng hàm UNPIVOT.

Trong chương kế tiếp, chúng ta tiếp tục tìm hiểu cách khai báo biến, phát biểu điều khiển và chi tiết cách sử dụng hàm CASE.

## Chương 10:

# KHAI BÁO BIẾN VÀ PHÁT BIỂU ĐIỀU KHIỂN

### Tóm tắt chương 10

Trong chương này, chúng ta tiếp tục tìm hiểu biến cục bộ, cách khai và gán giá trị cho biến, phép toán, biến kiểu TABLE, phát biểu điều khiển trong SQL Server 2005 trước khi tìm hiểu thủ tục nội tại.

#### Các vấn đề chính sẽ được đề cập:

- ✓ Khai báo và sử dụng biến.
- ✓ Biến kiểu Table.
- ✓ Phát biểu điều khiển.
- ✓ Hàm CASE.

## 1. KHAI BÁO VÀ SỬ DỤNG BIẾN

Để khai báo biến trong SQL Server 2005, bạn sử dụng từ khóa DECLARE, tên biến phải bắt đầu bằng tiền tố @ rồi mới đến tên của biến.

Chú ý: Bạn có thể tìm thấy các ví dụ trình bày của chương này nằm trong tập tin có tên VariablesAndConditionStatement.sql.

### 1.1. Khai báo biến

Biến đầy đủ sẽ có cú pháp như sau:

```
DECLARE
  {{ @local_variable [AS] data_type }}
```

**M® 102****Chương 10: Khai báo biến và phát biểu điều khiển**

Tuy nhiên, bạn cũng có thể không sử dụng từ khóa AS và cần chỉ định chiều dài của dữ liệu nếu kiểu dữ liệu yêu cầu xác định chiều dài.

```
DECLARE {{ @local_variable data_type[(length)] }}
```

Trong đó, @local\_variable là tên biến, data\_type là các kiểu dữ liệu trong SQL Server ngoại trừ text, ntext hay image. Tất cả các biến sau khi khai báo sẽ có giá trị khởi tạo mặc định là NULL.

Tầm vực của biến chỉ có giới hạn trong lô (batch) gồm các phát biểu và kết thúc bằng lệnh GO.

```
DECLARE @Status bit  
GO
```

Chẳng hạn, bạn có thể khai báo biến ứng với các kiểu dữ liệu tương tự như ví dụ 10-1.

**Ví dụ 10-1: Khai báo biến và xuất giá trị biến.**

```
DECLARE @Status bit  
DECLARE @CustomerId CHAR(5)  
DECLARE @TotalQuantity INT  
DECLARE @TotalDiscount DECIMAL(18)  
DECLARE @TotalVATAmount DECIMAL  
DECLARE @Amount DECIMAL  
DECLARE @TotalAmount DECIMAL  
PRINT @Status  
PRINT @CustomerId  
PRINT @TotalQuantity  
PRINT @TotalDiscount  
PRINT @TotalVATAmount  
PRINT @Amount  
PRINT @TotalAmount  
GO
```

Trong trường hợp muốn khai báo nhiều biến trên một hàng, bạn sử dụng cú pháp tương tự như ví dụ 10-2.

**Ví dụ 10-2: Khai báo biến trên cùng một hàng**

```
DECLARE @Status bit, @CustomerId CHAR(5)  
DECLARE @TotalQuantity INT  
DECLARE @TotalDiscount DECIMAL(18)  
DECLARE @TotalVATAmount DECIMAL  
DECLARE @Amount DECIMAL, @TotalAmount DECIMAL  
PRINT @Status  
PRINT @CustomerId  
PRINT @TotalQuantity  
PRINT @TotalDiscount  
PRINT @TotalVATAmount  
PRINT @Amount
```

**Chương 10: Khai báo biến và phát biểu điều khiển**

103

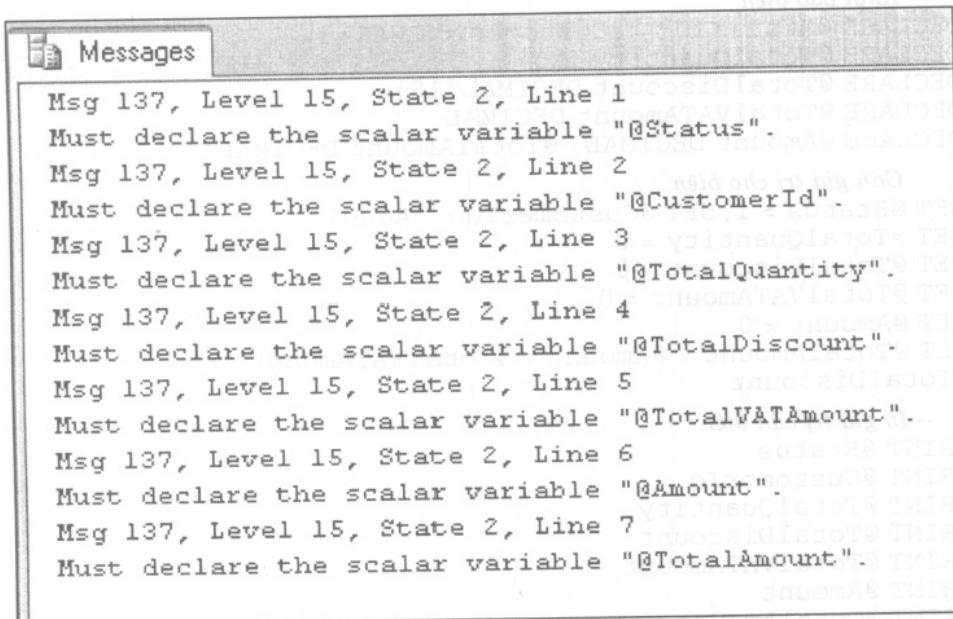
```
PRINT @TotalAmount
GO
```

Như giới thiệu ở trên, biến cục bộ chỉ có tầm vực trong lô, chính vì vậy lô sẽ phát sinh nếu bạn khai báo và sử dụng biến như ví dụ 10-3.

**Ví dụ 10-3: Khai báo và sử dụng biến**

```
DECLARE @Status bit, @CustomerId CHAR(5)
DECLARE @TotalQuantity INT
DECLARE @TotalDiscount DECIMAL(18)
DECLARE @TotalVATAmount DECIMAL
DECLARE @Amount DECIMAL, @TotalAmount DECIMAL
GO
PRINT @Status
PRINT @CustomerId
PRINT @TotalQuantity
PRINT @TotalDiscount
PRINT @TotalVATAmount
PRINT @Amount
PRINT @TotalAmount
GO
```

Khi thực thi lô phát biểu khai báo và in giá trị của biến, lỗi sẽ phát sinh như hình 10-1.



**Hình 10-1: Lỗi phát sinh do tầm vực của biến.**

**Chú ý:** Một số hàm có sẵn trong SQL Server 2005 sẽ được nhận dạng với hai ký tự @ như: @@IDENTITY, ROWCOUNT, ... đã trình bày trong chương trước.

Sau khi khai báo biến, bạn có thể sử dụng phát biểu SET hay SELECT để gán giá trị cho biến.

## 1.2. Phát biểu SET

Trong trường hợp sử dụng phát biểu SET thì khai báo theo cú pháp như sau:

```
SET
{ @local_variable
  [ :: property_name | field_name ] = expression | udt_name
{ . | :: }
method_name(argument [ ,...n ] )
```

Cụ thể phát biểu SET cho phép bạn gán giá trị cho biến tương tự như ví dụ 10-4.

### Ví dụ 10-4: Khai báo gán giá trị cho biến

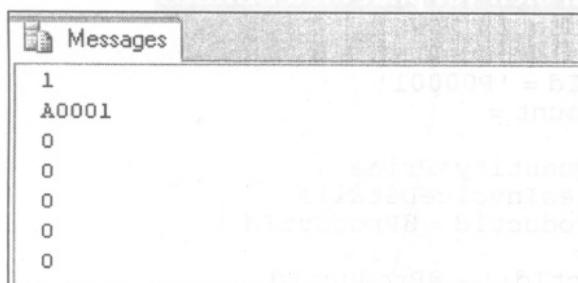
```
-- Khai báo biến
DECLARE @Status bit, @CustomerId CHAR(5)
DECLARE @TotalQuantity INT
DECLARE @TotalDiscount DECIMAL(18)
DECLARE @TotalVATAmount DECIMAL
DECLARE @Amount DECIMAL, @TotalAmount DECIMAL
-- Gán giá trị cho biến
SET @Status = 1; SET @CustomerId = 'A0001'
SET @TotalQuantity = 0
SET @TotalDiscount = 0
SET @TotalVATAmount = 0
SET @Amount = 0
SET @TotalAmount = @Amount + @TotalVATAmount -
@TotalDiscount
-- In giá trị của biến
PRINT @Status
PRINT @CustomerId
PRINT @TotalQuantity
PRINT @TotalDiscount
PRINT @TotalVATAmount
PRINT @Amount
PRINT @TotalAmount
GO
```

**Chú ý:** Bạn có thể sử dụng phát biểu SET trên cùng một hàng bằng cách sử dụng dấu chấm phẩy để phân cách.

**Chương 10: Khai báo biến và phát biểu điều khiển**105 

```
SET @Status = 1; SET @CustomerId = 'A0001'
```

Khi thực thi phát biểu trong ví dụ trên, bạn có thể tìm thấy kết quả trình bày như hình 10-2.



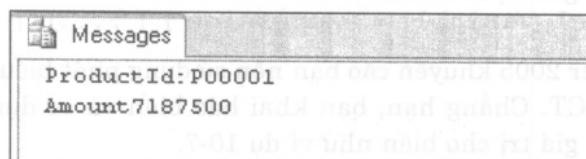
**Hình 10-2: Kết quả trình bày giá trị của biến.**

Ngoài việc gán giá trị cụ thể cho biến, bạn có thể sử dụng phát biểu SET để gán giá trị lấy từ phát biểu SELECT tương tự như ví dụ 10-5.

**Ví dụ 10-5: Gán biến với phát biểu SELECT**

```
DECLARE @ProductId CHAR(10)
DECLARE @TotalAmount DECIMAL
SET @ProductId = 'P00001'
SET @TotalAmount =
(
    SELECT SUM(Quantity*Price)
    FROM SalesInvoiceDetails
    WHERE ProductId = @ProductId
)
PRINT 'ProductId: ' + @ProductId
PRINT 'Amount' + LTRIM(@TotalAmount)
GO
```

Nếu thực thi phát biểu trong ví dụ trên, tổng số tiền bán của sản phẩm có mã là P00001 trình bày như hình 10-3.



**Hình 10-3: Sử dụng phát biểu SET với phát biểu SELECT.**

Chú ý: Khi sử dụng phát biểu SET với phát biểu SELECT, bạn bảo đảm phát biểu SELECT này trả về giá trị đơn. Nếu phát biểu SELECT trả về nhiều giá trị thì lỗi sẽ phát sinh.

M® 106

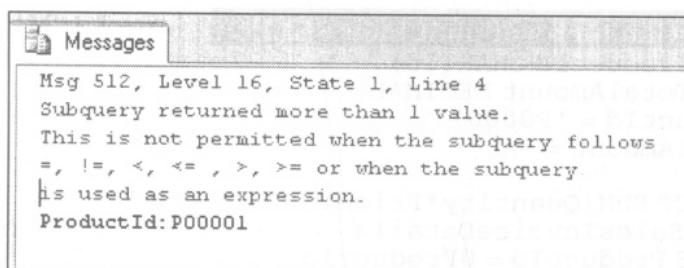
**Chương 10: Khai báo biến và phát biểu điều khiển**

Chẳng hạn, bạn khai báo phát biểu SELECT với nhiều giá trị trả về như ví dụ 10-6.

**Ví dụ 10-6: Gán biến với phát biểu SELECT**

```
DECLARE @ProductId CHAR(10)
DECLARE @TotalAmount DECIMAL
SET @ProductId = 'P00001'
SET @TotalAmount =
(
    SELECT Quantity*Price
    FROM SalesInvoiceDetails
    WHERE ProductId = @ProductId
)
PRINT 'ProductId:' + @ProductId
PRINT 'Amount' + LTRIM(@TotalAmount)
GO
```

Khi thực thi phát biểu trong ví dụ trên, lỗi sẽ phát sinh tương tự như hình 10-4.



**Hình 10-4: Lỗi do phát biểu SELECT trả về nhiều giá trị.**

### 1.3. Phát biểu SELECT

Ngoài cách sử dụng phát biểu SET, bạn có thể sử dụng phát biểu SELECT để gán giá trị cho biến.

```
SELECT { @local_variable = expression } [ ,...n ] [ ; ]
```

SQL Server 2005 khuyến cáo bạn nên sử dụng phát biểu SET thay vì phát biểu SELECT. Chẳng hạn, bạn khai báo biến và sử dụng phát biểu SELECT để gán giá trị cho biến như ví dụ 10-7.

**Ví dụ 10-7: Sử dụng phát biểu SELECT để gán giá trị cho biến**

```
DECLARE @CustomerId varchar(10)
DECLARE @TotalAmount DECIMAL
SELECT @CustomerId = 'A0001'
SELECT @TotalAmount =10000
SELECT @CustomerId AS CustomerId, @TotalAmount AS
TotalAmount
GO
```

## Chương 10: Khai báo biến và phát biểu điều khiển

107

Khi thực thi phát biểu trong ví dụ trên, kết quả trình bày tương tự như hình 10-5.

| CustomerId | TotalAmount |
|------------|-------------|
| 1          | A0001       |

Hình 10-5: Sử dụng phát biểu SELECT.

Nếu bạn gán giá trị cho nhiều biến, bạn phải sử dụng phát biểu SET cho mỗi biến, sử dụng dấu chấm phẩy để phân cách hai phát biểu SET như ví dụ 10-4.

Trong trường hợp sử dụng phát biểu SELECT để gán giá trị cho biến, bạn có thể gán cùng một lúc nhiều biến.

#### Ví dụ 10-8: Gán giá trị cho nhiều biến

-- Khai báo biến

```
DECLARE @CustomerId varchar(10)
DECLARE @TotalQuantity INT
DECLARE @TotalAmount DECIMAL
```

-- Gán giá trị cho biến

```
SELECT @CustomerId = 'A0001', @TotalQuantity=100, @TotalAmount =10000
```

-- Trình bày giá trị của biến

```
SELECT @CustomerId AS CustomerId,
@TotalAmount AS TotalAmount,
@TotalQuantity AS TotalQuantity
GO
```

Khi thực thi phát biểu trong ví dụ trên, kết quả trình bày tương tự như hình 10-6.

| CustomerId | TotalAmount | TotalQuantity |
|------------|-------------|---------------|
| 1          | A0001       | 10000         |

Hình 10-6: Sử dụng phát biểu SELECT.

Tương tự như trường hợp phát biểu SET và giá trị lấy ra từ phát biểu SELECT, bạn cũng có thể sử dụng phát biểu SELECT để lấy giá trị từ phát biểu SELECT khác như ví dụ 10-9.

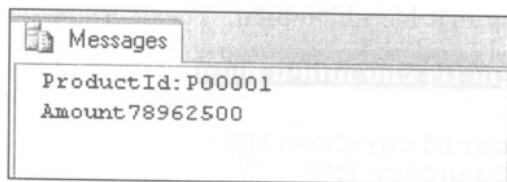
**Ví dụ 10-9: Gán giá trị bằng lệnh SELECT với một phát biểu****SELECT khác**

```

DECLARE @ProductId CHAR(10)
DECLARE @TotalAmount DECIMAL
SET @ProductId = 'P00001'
SELECT @TotalAmount =
(
    SELECT SUM((1+VATRate)*Quantity*Price-Discount)
    FROM SalesInvoiceDetails
    WHERE ProductId = @ProductId
)
PRINT 'ProductId: ' + @ProductId
PRINT 'Amount' + LTRIM(@TotalAmount)
GO

```

Khi thực thi phát biểu trong ví dụ trên, kết quả trình bày tương tự như hình 10-7.



**Hình 10-7: Sử dụng phát biểu SELECT.**

Một trong những điểm mạnh của phát biểu SELECT khi sử dụng nó để gán giá trị cho biến là cùng một lúc bạn có thể lấy giá trị từ cơ sở dữ liệu và gán vào nhiều biến.

Chẳng hạn, để lấy ra tổng số lượng, tiền bán và tiền thuế VAT rồi gán vào 3 biến tương ứng thì bạn khai báo như ví dụ 10-10.

**Ví dụ 10-10: Gán nhiều giá trị bằng lệnh SELECT**

```

DECLARE @ProductId CHAR(10)
DECLARE @TotalQuantity INT
DECLARE @TotalDiscount DECIMAL
DECLARE @VATAmount DECIMAL
DECLARE @TotalAmount DECIMAL
SET @ProductId = 'P00001'
SELECT
    @TotalQuantity = SUM(Quantity),
    @TotalAmount = SUM(Quantity*Price),
    @TotalDiscount = SUM(Discount),
    @VATAmount = SUM(VATRate*Quantity*Price)
    FROM SalesInvoiceDetails
    WHERE ProductId = @ProductId
PRINT 'ProductId: ' + @ProductId
PRINT 'Quantity:' + LTRIM(@TotalQuantity)

```

**Chương 10:** Khai báo biến và phát biểu điều khiển

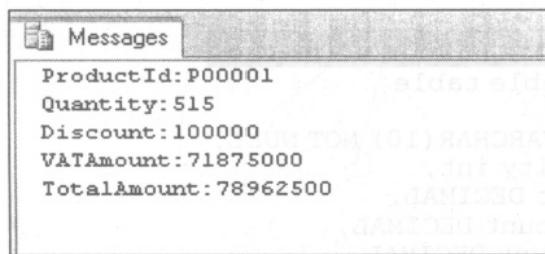
109

```

PRINT 'Discount:' + LTRIM(@TotalDiscount)
PRINT 'VATAmount:' + LTRIM(@VATAmount)
PRINT 'TotalAmount:' + LTRIM(@TotalAmount +
    @VATAmount - @TotalDiscount)
GO

```

Nếu thực thi phát biểu trong ví dụ trên, kết quả trình bày tương tự như hình 10-8.



**Hình 10-8: Sử dụng phát biểu SELECT.**

## 2. BIẾN KIẾU TABLE

Một trong kiểu dữ liệu mới giới thiệu trong SQL Server 2005 là kiểu TABLE, bạn có thể khai báo biến kiểu TABLE với cú pháp như sau:

```

DECLARE
    {{ @table_variable_name < table_type_definition > }
    } [ ,...n]

< table_type_definition > ::= TABLE ( { < column_definition > | < table_constraint > }
[ ,... ] )
< column_definition > ::= [ column_name { scalar_data_type | AS
computed_column_expression } [ COLLATE collation_name ]
[ [ DEFAULT constant_expression ] | IDENTITY
[ ( seed,increment ) ] ]
[ ROWGUIDCOL ]
[ < column_constraint > ]
< column_constraint > ::= [ [ NULL | NOT NULL ]
| [ PRIMARY KEY | UNIQUE ]
| CHECK ( logical_expression )
]
< table_constraint > ::= { { PRIMARY KEY | UNIQUE } ( column_name [ ,... ] )
| CHECK ( search_condition )
}

```

M® 110

**Chương 10: Khai báo biến và phát biểu điều khiển**

Cú pháp để khai báo biến kiểu TABLE khá phức tạp, chúng ta sẽ tìm hiểu cách làm việc với đối tượng này qua từng ví dụ.

**Chú ý:** Chúng ta sẽ tìm hiểu biến kiểu TABLE và biến kiểu CURSOR trong tập kế tiếp: *SQL Server 2005 - Lập trình nâng cao*.

Chẳng hạn, bạn khai báo biến đối tượng Table bao gồm các cột dữ liệu như ví dụ 10-11.

**Ví dụ 10-11: Khai báo tạo kiểu TABLE**

```
DECLARE @MyTable table
(
    ProductId VARCHAR(10) NOT NULL,
    TotalQuantity int,
    TotalAmount DECIMAL,
    TotalVATAmount DECIMAL,
    TotalDiscount DECIMAL
)
```

Bạn có thể sử dụng phát biểu SELECT để truy vấn dữ liệu trong bảng dữ liệu ứng với biến kiểu đối tượng TABLE vừa tạo như ví dụ 10-12.

**Ví dụ 10-12: Khai báo và truy vấn đối tượng TABLE**

```
DECLARE @MyTable table
(
    ProductId VARCHAR(10) NOT NULL,
    TotalQuantity int,
    TotalAmount DECIMAL,
    TotalVATAmount DECIMAL,
    TotalDiscount DECIMAL
)
SELECT * FROM @MyTable;
GO
```

Khi thực thi phát biểu DECLARE và SELECT trong ví dụ trên, bạn có thể tìm thấy kết quả trình bày như hình 10-9.

| ProductId | TotalQuantity | TotalAmount | TotalVATAmount | TotalDiscount |
|-----------|---------------|-------------|----------------|---------------|
|           |               |             |                |               |

**Hình 10-9: Truy vấn dữ liệu trong biến đối tượng TABLE.**

Bạn có thể thêm dữ liệu vào biến đối tượng TABLE bằng cách khai báo phát biểu INSERT với đối tượng TABLE.

**Ví dụ 10-13: Khai báo thêm dữ liệu vào biến TABLE với INSERT**

```
DECLARE @MyTable table
```

**Chương 10: Khai báo biến và phát biểu điều khiển**

111

```

(
    ProductId VARCHAR(10) NOT NULL,
    TotalQuantity int,
    TotalAmount DECIMAL,
    TotalVATAmount DECIMAL,
    TotalDiscount DECIMAL
)
INSERT INTO @MyTable
VALUES ('A0001', 10, 1000, 100, 0)
SELECT * FROM @MyTable;
GO
  
```

Khi thực thi phát biểu DECLARE và SELECT trong ví dụ trên, bạn có thể tìm thấy kết quả trình bày như hình 10-10.

| ProductId | TotalQuantity | TotalAmount | TotalVATAmount | TotalDiscount |
|-----------|---------------|-------------|----------------|---------------|
| 1 A0001   | 10            | 1000        | 100            | 0             |

**Hình 10-10: Thêm dữ liệu vào biến đối tượng TABLE với INSERT.**

Ngoài ra, bạn cũng có thể thêm dữ liệu vào biến đối tượng TABLE từ phát biểu SELECT như cách khai báo phát biểu INSERT và SELECT với đối tượng TABLE như ví dụ 10-14.

**Ví dụ 10-14: Khai báo thêm dữ liệu vào biến TABLE với SELECT**

```

DECLARE @MyTable table
(
    ProductId VARCHAR(10) NOT NULL,
    TotalQuantity int,
    TotalAmount DECIMAL,
    TotalVATAmount DECIMAL,
    TotalDiscount DECIMAL
)
INSERT INTO @MyTable
SELECT
    ProductId,
    SUM(Quantity),
    SUM(Quantity*Price),
    SUM(VATRate*Quantity*Price),
    SUM(Discount)
FROM SalesInvoiceDetails
GROUP BY ProductId;
SELECT * FROM @MyTable;
GO
  
```

Khi thực thi phát biểu DECLARE, INSERT và SELECT trong ví dụ trên, bạn có thể tìm thấy kết quả trình bày như hình 10-11.

|   | ProductId | TotalQuantity | TotalAmount | TotalVATAmount | TotalDiscount |
|---|-----------|---------------|-------------|----------------|---------------|
| 1 | P00001    | 515           | 7187500     | 71875000       | 100000        |
| 2 | P00002    | 750           | 8025000     | 80250000       | 160000        |
| 3 | P00003    | 625           | 6812500     | 68125000       | 280000        |
| 4 | P00004    | 176           | 2136500     | 21365000       | 105000        |
| 5 | P00005    | 130           | 1750000     | 17500000       | 90000         |
| 6 | P00006    | 90            | 1245000     | 12450000       | 70000         |

Hình 10-11: Thêm dữ liệu vào biến đối tượng TABLE với SELECT.

### 3. PHÁT BIỂU ĐIỀU KHIỂN

SQL Server 2005 giới thiệu phát biểu điều khiển bao gồm IF..ELSE, BEGIN..END, WHILE, RETURN, TRY..CATCH, WAITFOR, CONTINUE, BREAK và GOTO.

#### 3.1. Phát biểu điều khiển IF..ELSE

Tương tự như các ngôn ngữ lập trình khác, SQL Server 2005 giới thiệu phát biểu rẽ nhánh IF..ELSE với cấu trúc như sau:

```
IF Boolean_expression
  { sql_statement | statement_block }
[ ELSE
  { sql_statement | statement_block } ]
```

Trong đó, Boolean\_expression là biểu thức luận lý trả về giá trị True hay False. Chẳng hạn, bạn khai báo sử dụng phát biểu rẽ nhánh IF..ELSE như ví dụ 10-15.

##### Ví dụ 10-15: Khai báo phát biểu IF..ELSE

```
DECLARE @ProductId CHAR(10)
DECLARE @TotalQuantity INT
SET @ProductId = 'P00001'
SELECT
  @TotalQuantity = SUM(Quantity)
  FROM SalesInvoiceDetails
  WHERE ProductId = @ProductId
PRINT 'ProductId:' + @ProductId
```

**Chương 10: Khai báo biến và phát biểu điều khiển**

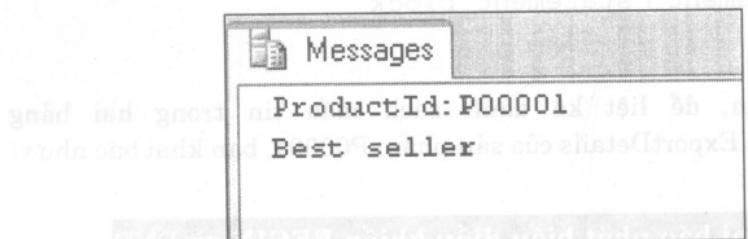
113

```

if @TotalQuantity>500
    PRINT 'Best seller'
else
    PRINT 'Normal seller'
GO

```

Khi thực thi phát biểu trong ví dụ trên, bạn có thể tìm thấy kết quả trình bày như hình 10-12.



**Hình 10-12: Khai báo sử dụng phát biểu IF..ELSE.**

Bạn cũng có thể sử dụng hàm EXISTS với phát biểu IF thông tin như ví dụ 10-16.

**Ví dụ 10-16: Khai báo hàm EXISTS trong phát biểu IF..ELSE**

```

DECLARE @ProductId CHAR(10)
SET @ProductId = 'P00001'
if EXISTS(SELECT *
        FROM SalesInvoiceDetails
        WHERE ProductId = @ProductId)
    SELECT * FROM ExportDetails
        WHERE ProductId = @ProductId
GO

```

Khi thực thi phát biểu IF với hàm EXISTS trong ví dụ trên, bạn có thể tìm thấy kết quả trình bày như hình 10-13.

**Hình 10-13: Khai báo sử dụng phát biểu IF..ELSE.**

### 3.2. Phát biểu điều khiển BEGIN..END

Khi bên trong phát biểu IF..ELSE, WHILE, TRY..CATCH hay chuyển tác (TRANSACTION) hay nhóm phát biểu SQL, bạn có thể sử dụng phát biểu điều khiển BEGIN..END.

```
BEGIN [yêu cầu bắt buộc]
  {
    sql_statement | statement_block
  }
END
```

Chẳng hạn, để liệt kê danh sách mẫu tin trong hai bảng ImportDetails và ExportDetails của sản phẩm P00001, bạn khai báo ví dụ 10-17.

#### Ví dụ 10-17: Khai báo phát biểu điều khiển BEGIN và END

```
DECLARE @ProductId CHAR(10)
SET @ProductId = 'P00001'
if exists(SELECT *
          FROM SalesInvoiceDetails
          WHERE ProductId = @ProductId)
BEGIN
  SELECT StockId, Quantity AS ExportQuantity,
         0 As ImportQuantity
  FROM ExportDetails
  WHERE ProductId = @ProductId
  UNION ALL
  SELECT StockId, 0, Quantity AS Import
  FROM ImportDetails
  WHERE ProductId = @ProductId
END
GO
```

Nếu thực thi phát biểu trong ví dụ trên, bạn có thể tìm thấy kết quả trình bày như hình 10-14.

| StockId | ExportQuantity | ImportQuantity |
|---------|----------------|----------------|
| 1 ST001 | 100            | 0              |
| 2 ST002 | 150            | 0              |
| 3 ST002 | 250            | 0              |
| 4 ST001 | 15             | 0              |
| 5 ST001 | 0              | 500            |
| 6 ST001 | 0              | 400            |
| 7 ST002 | 0              | 400            |
| 8 ST002 | 0              | 200            |

Hình 10-14: Sử dụng phát biểu BEGIN và END.

### 3.3. Phát biểu điều khiển WHILE

Phát biểu điều khiển WHILE cho phép chúng ta lặp lại thực thi tập lệnh cho đến khi biểu thức kiểm tra là False.

```
WHILE Boolean_expression
  { sql_statement | statement_block }
  [ BREAK ]
  { sql_statement | statement_block }
  [ CONTINUE ]
  { sql_statement | statement_block }
```

Chẳng hạn, bạn có thể khai báo sử dụng phát biểu điều khiển WHILE như ví dụ 10-18.

#### Ví dụ 10-18: Khai báo sử dụng phát biểu điều khiển WHILE

```
DECLARE @count int
SET @count = 0
WHILE @count<10
BEGIN
  SET @count = @count + 1
  PRINT @count
END
GO
```

Khi thực thi phát biểu trong ví dụ trên, bạn có thể tìm thấy giá trị của biến @count trình bày như hình 10-15.

```
1
2
3
4
5
6
7
8
9
10
```

Hình 10-15: Giá trị của biến @count.

Bạn cũng có thể sử dụng phát biểu điều khiển WHILE với phát biểu SQL dạng SELECT để tính tổng quỹ trong tháng. Để làm điều này, trước tiên bạn khai báo đoạn chương trình với biểu thức bảng như ví dụ 10-19.

**M<sup>®</sup> 116****Chương 10: Khai báo biến và phát biểu điều khiển WHILE****-- Khai báo biến**

```
DECLARE @CurrentMonth CHAR(7)
DECLARE @PreviousMonth CHAR(7)
SET @CurrentMonth = '10/2007'
SET @PreviousMonth = dbo.udfPreviousMonth('10/2007');
```

**-- Tổng hợp dữ liệu**

```
WITH BalanceOfToday
AS
(
    SELECT '01/' + @CurrentMonth AS DueDate,
    N 'Tồn quỹ đầu kỳ' AS DescriptionInVietnamese,
    0 as Receipt, 0 as Payment,
    EndingAmount AS BalanceAmount
    FROM CloseMonthCashBalances
    WHERE CloseMonth=@PreviousMonth
    UNION ALL
    SELECT Convert(char(11), ReceiptDate, 106) AS DueDate,
    DescriptionInVietnamese, ReceiptAmount,
    0, 0
    FROM Receipts
    UNION ALL
    SELECT Convert(char(11), PaymentDate, 106) AS DueDate,
    DescriptionInVietnamese, 0, PaymentAmount, 0
    FROM Payments
)
```

**-- Trình bày dữ liệu tổng hợp**

```
SELECT DueDate, DescriptionInVietnamese,
Receipt, Payment, BalanceAmount
,ROW_NUMBER() OVER(ORDER BY DueDate) AS RowNumber
FROM BalanceOfToday
ORDER BY RowNumber ASC
```

Nếu thực thi các phát biểu SELECT trên, bạn có thể tìm thấy danh sách tác vụ thu và chi tiền trình bày như hình 10-16.

## Chương 10: Khai báo biến và phát biểu điều khiển

117 

|    | DueDate     | DescriptionInVietnamese                      | Receipt | Payment | BalanceAmount | RowNumber |
|----|-------------|--|---------|---------|---------------|-----------|
| 1  | 01/10/2007  | Tồn quỹ đầu kỳ                               | 0       | 0       | 100000000     | 1         |
| 2  | 06 Oct 2007 | Trả tiền mua hàng Ajinomoto Việt Nam         | 0       | 5030000 | 0             | 2         |
| 3  | 06 Oct 2007 | Trả tiền mua hàng Suzumi Việt Nam            | 0       | 5132500 | 0             | 3         |
| 4  | 06 Oct 2007 | Trả tiền mua hàng                            | 0       | 700000  | 0             | 4         |
| 5  | 07 Oct 2007 | Trả tiền mua hàng                            | 0       | 1980000 | 0             | 5         |
| 6  | 07 Oct 2007 | Trả tiền mua hàng Ajinomoto Việt Nam         | 0       | 700000  | 0             | 6         |
| 7  | 07 Oct 2007 | Trả tiền mua hàng                            | 0       | 200000  | 0             | 7         |
| 8  | 08 Oct 2007 | Trả tiền mua hàng                            | 0       | 200000  | 0             | 8         |
| 9  | 09 Oct 2007 | Trả tiền mua hàng                            | 0       | 1500000 | 0             | 9         |
| 10 | 09 Oct 2007 | Trả tiền mua hàng                            | 0       | 1500000 | 0             | 10        |
| 11 | 09 Oct 2007 | Trả tiền mua hàng                            | 0       | 1500000 | 0             | 11        |
| 12 | 10 Oct 2007 | Trả tiền mua hàng                            | 0       | 3500000 | 0             | 12        |
| 13 | 10 Oct 2007 | Trả tiền mua hàng                            | 0       | 2500000 | 0             | 13        |
| 14 | 10 Oct 2007 | Thu tiền bán hàng của khách hàng ...         | 500000  | 0       | 0             | 14        |
| 15 | 11 Oct 2007 | Thu tiền bán hàng của khách hàng ...         | 450000  | 0       | 0             | 15        |
| 16 | 11 Oct 2007 | Thu tiền bán hàng của khách hàng ...         | 1055000 | 0       | 0             | 16        |
| 17 | 11 Oct 2007 | Trả tiền mua hàng                            | 0       | 6177500 | 0             | 17        |
| 18 | 12 Oct 2007 | Thu tiền bán hàng của khách hàng ...         | 1500000 | 0       | 0             | 18        |
| 19 | 13 Oct 2007 | Thu tiền tạm ứng mua hàng của khách hàng ... | 3000000 | 0       | 0             | 19        |
| 20 | 13 Oct 2007 | Thu tiền bán hàng của khách hàng ...         | 2000000 | 0       | 0             | 20        |
| 21 | 13 Oct 2007 | Thu tiền bán hàng của khách hàng ...         | 1200000 | 0       | 0             | 21        |
| 22 | 14 Oct 2007 | Thu tiền bán hàng của khách hàng ...         | 1450000 | 0       | 0             | 22        |
| 23 | 14 Oct 2007 | Thu tiền bán hàng của khách hàng ...         | 1650000 | 0       | 0             | 23        |
| 24 | 17 Oct 2007 | Thu tiền bán hàng của khách hàng ...         | 2000000 | 0       | 0             | 24        |
| 25 | 17 Oct 2007 | Thu tiền bán hàng của khách hàng ...         | 790500  | 0       | 0             | 25        |
| 26 | 17 Oct 2007 | Thu tiền bán hàng của khách hàng ...         | 1000000 | 0       | 0             | 26        |
| 27 | 18 Oct 2007 | Thu tiền bán hàng của khách hàng ...         | 1000000 | 0       | 0             | 27        |

Hình 10-16: Tình hình thu và trả tiền.

Trong hình trên, nếu bạn muốn cập nhật giá trị tại cột BalanceAmount của hàng thứ 2 trở đi thì sử dụng phát biểu điều khiển WHILE và hàm EXISTS như ví dụ 10-20.

**Ví dụ 10-20: Khai báo cập nhật cột BalanceAmount**

-- Khai báo biến

```
DECLARE @CurrentMonth CHAR(7)
DECLARE @PreviousMonth CHAR(7)
SET @CurrentMonth = '10/2007'
SET @PreviousMonth = dbo.udfPreviousMonth('10/2007');
```

-- Tổng hợp dữ liệu

```
WITH BalanceOfToday
AS
(
    UPDATE #BalanceTemp SET BalanceAmount = 0
    SELECT '01/' + @CurrentMonth AS DueDate,
    N'Tồn quỹ đầu kỳ' AS DescriptionInVietnamese,
    0 as Receipt, 0 as Payment,
    EndingAmount As BalanceAmount
    FROM CloseMonthCashBalances
    INSERT INTO #BalanceTemp
    SELECT * FROM #BalanceTemp
)
```

**M<sup>®</sup> 118****Chương 10: Khai báo biến và phát biểu điều khiển**

```

WHERE CloseMonth=@PreviousMonth
UNION ALL
SELECT Convert(char(11), ReceiptDate, 106) AS DueDate,
DescriptionInVietnamese, ReceiptAmount,
0, 0
FROM Receipts
UNION ALL
SELECT Convert(char(11), PaymentDate, 106) AS DueDate,
DescriptionInVietnamese, 0, PaymentAmount, 0
FROM Payments
)
SELECT DueDate, DescriptionInVietnamese,
Receipt, Payment, BalanceAmount
, ROW_NUMBER() OVER(ORDER BY DueDate) AS RowNumber
-- Thêm tổng hợp dữ liệu vào bảng tạm

INTO #Balances
FROM BalanceOfToday
ORDER BY RowNumber ASC
DECLARE @count int
DECLARE @rptAmount int, @totalRptAmt int
DECLARE @pmtAmount int, @totalPmtAmt int
DECLARE @balanceAmount int
SET @balanceAmount = 0
SET @totalRptAmt = 0
SET @totalPmtAmt = 0
SET @count = 1
-- Duyệt qua từng mẩu tin
WHILE(exists(SELECT * FROM #Balances WHERE
RowNumber=@count+1))
BEGIN
-- Nếu mẩu tin ứng với tồn quỹ đầu kỳ
SELECT @balanceAmount=BalanceAmount FROM #Balances
WHERE RowNumber=@count
-- Nếu mẩu tin kế tiếp tồn tại
SELECT @rptAmount = Receipt, @pmtAmount = Payment
FROM #Balances WHERE RowNumber=@count +1
SET @balanceAmount = @balanceAmount + @rptAmount -
@pmtAmount
SET @totalRptAmt = @totalRptAmt + @rptAmount
SET @totalPmtAmt = @totalPmtAmt + @pmtAmount
-- Cập nhật giá trị cho cột BalanceAmount
UPDATE #Balances SET BalanceAmount = @balanceAmount
WHERE RowNumber=@count+1
SET @count = @count + 1
END
-- Thêm mẩu tin ứng với tồn quỹ cuối kỳ
INSERT INTO #Balances

```

**Chương 10: Khai báo biến và phát biểu điều khiển**

119

```
VALUES (LTRIM(dbo.udfLastDay (@CurrentMonth))
+ '/' + @CurrentMonth ,N'Tồn quỹ đầu kỳ' ,
@totalRptAmt, @totalPmtAmt, @balanceAmount, @count +1)
```

-- Trình bày tình hình tồn quỹ

```
SELECT DueDate, DescriptionInVietnamese,
Receipt, Payment, BalanceAmount
FROM #Balances ORDER BY RowNumber ASC
GO
```

-- Xóa bảng tạm

```
DROP TABLE #Balances
GO
```

Khi thực thi phát biểu điều khiển WHILE và phát biểu SELECT trong ví dụ trên, kết quả trình bày như hình 10-17.

|    | DueDate     | DescriptionInVietnamese                  | Receipt | Payment | BalanceAmount |
|----|-------------|--|---------|---------|---------------|
| 1  | 01/10/2007  | Tồn quỹ đầu kỳ                           | 0       | 0       | 100000000     |
| 2  | 06 Oct 2007 | Trả tiền mua hàng Ajinomoto Việt Nam     | 0       | 5030000 | 94970000      |
| 3  | 06 Oct 2007 | Trả tiền mua hàng Suzumi Việt Nam        | 0       | 5132500 | 89837500      |
| 4  | 06 Oct 2007 | Trả tiền mua hàng                        | 0       | 7000000 | 82837500      |
| 5  | 07 Oct 2007 | Trả tiền mua hàng                        | 0       | 1980000 | 80857500      |
| 6  | 07 Oct 2007 | Trả tiền mua hàng Ajinomoto Việt Nam     | 0       | 7000000 | 73857500      |
| 7  | 07 Oct 2007 | Trả tiền mua hàng                        | 0       | 2000000 | 71857500      |
| 8  | 08 Oct 2007 | Trả tiền mua hàng                        | 0       | 2000000 | 69857500      |
| 9  | 09 Oct 2007 | Trả tiền mua hàng                        | 0       | 1500000 | 68357500      |
| 10 | 09 Oct 2007 | Trả tiền mua hàng                        | 0       | 1500000 | 66857500      |
| 11 | 09 Oct 2007 | Trả tiền mua hàng                        | 0       | 1500000 | 65357500      |
| 12 | 10 Oct 2007 | Trả tiền mua hàng                        | 0       | 3500000 | 61857500      |
| 13 | 10 Oct 2007 | Trả tiền mua hàng                        | 0       | 2500000 | 59357500      |
| 14 | 10 Oct 2007 | Thu tiền bán hàng của khách hàng ...     | 500000  | 0       | 59857500      |
| 15 | 11 Oct 2007 | Thu tiền bán hàng của khách hàng ...     | 450000  | 0       | 60307500      |
| 16 | 11 Oct 2007 | Thu tiền bán hàng của khách hàng ...     | 1055000 | 0       | 61362500      |
| 17 | 11 Oct 2007 | Trả tiền mua hàng                        | 0       | 6177500 | 55185000      |
| 18 | 12 Oct 2007 | Thu tiền bán hàng của khách hàng ...     | 1500000 | 0       | 56685000      |
| 19 | 13 Oct 2007 | Thu tiền tạm ứng mua hàng của khách hàng | 3000000 | 0       | 59685000      |
| 20 | 13 Oct 2007 | Thu tiền bán hàng của khách hàng ...     | 2000000 | 0       | 61685000      |
| 21 | 13 Oct 2007 | Thu tiền bán hàng của khách hàng ...     | 1200000 | 0       | 62885000      |
| 22 | 14 Oct 2007 | Thu tiền bán hàng của khách hàng ...     | 1450000 | 0       | 64335000      |
| 23 | 14 Oct 2007 | Thu tiền bán hàng của khách hàng ...     | 1650000 | 0       | 65985000      |
| 24 | 17 Oct 2007 | Thu tiền bán hàng của khách hàng ...     | 2000000 | 0       | 67985000      |
| 25 | 17 Oct 2007 | Thu tiền bán hàng của khách hàng ...     | 790500  | 0       | 68775500      |
| 26 | 17 Oct 2007 | Thu tiền bán hàng của khách hàng ...     | 1000000 | 0       | 69775500      |
| 27 | 18 Oct 2007 | Thu tiền bán hàng của khách hàng ...     | 1000000 | 0       | 70775500      |

**Hình 10-17: Sử dụng phát biểu điều khiển WHILE.**

Lưu ý: Bạn có thể sử dụng lệnh SET NOCOUNT để tắt hay mở việc trả về thông báo số dòng có tác động như ví dụ 10-21.

**M® 120****Chương 10: Khai báo biến và phát biểu điều khiển****Ví dụ 10-21: Khai báo và khởi tạo Constructor thứ ba**

SET NOCOUNT OFF

-- Khai báo biến

```
DECLARE @CurrentMonth CHAR(7)
DECLARE @PreviousMonth CHAR(7)
SET @CurrentMonth = '10/2007'
SET @PreviousMonth = dbo.udfPreviousMonth('10/2007');
```

-- Tổng hợp dữ liệu

```
WITH BalanceOfToday
AS
(
    SELECT '01/' + @CurrentMonth AS DueDate,
    N'Tồn quỹ đầu kỳ' AS DescriptionInVietnamese,
    0 as Receipt, 0 as Payment,
    EndingAmount As BalanceAmount
    FROM CloseMonthCashBalances
    WHERE CloseMonth=@PreviousMonth
    UNION ALL
    SELECT Convert(char(11), ReceiptDate, 106) AS DueDate,
    DescriptionInVietnamese, ReceiptAmount,
    0, 0
    FROM Receipts
    UNION ALL
    SELECT Convert(char(11), PaymentDate, 106) AS DueDate,
    DescriptionInVietnamese, 0, PaymentAmount, 0
    FROM Payments
)
SELECT DueDate, DescriptionInVietnamese,
Receipt, Payment, BalanceAmount
, ROW_NUMBER() OVER(ORDER BY DueDate) AS RowNumber
-- Thêm tổng hợp dữ liệu vào bảng tạm
INTO #Balances
FROM BalanceOfToday
ORDER BY RowNumber ASC
DECLARE @count int
DECLARE @rptAmount int, @totalRptAmt int
DECLARE @pmtAmount int, @totalPmtAmt int
DECLARE @balanceAmount int
SET @balanceAmount = 0
SET @totalRptAmt = 0
SET @totalPmtAmt = 0
SET @count = 1
-- Duyệt qua từng mẫu tin
WHILE(exists(SELECT * FROM #Balances WHERE
RowNumber=@count+1))
BEGIN
    -- Nếu mẫu tin ứng với tồn quỹ đầu kỳ
```

**Chương 10: Khai báo biến và phát biểu điều khiển**121 

```

SELECT @balanceAmount = BalanceAmount FROM #Balances
WHERE RowNumber=@count

-- Nếu mẫu tin kế tiếp tồn tại
SELECT @rptAmount = Receipt, @pmtAmount = Payment
FROM #Balances WHERE RowNumber=@count +1
SET @balanceAmount = @balanceAmount + @rptAmount -
@pmtAmount
SET @totalRptAmt = @totalRptAmt + @rptAmount
SET @totalPmtAmt = @totalPmtAmt + @pmtAmount

-- Cập nhật giá trị cho cột BalanceAmount
UPDATE #Balances SET BalanceAmount = @balanceAmount
WHERE RowNumber=@count+1
SET @count = @count + 1

END

-- Thêm mẫu tin ứng với tồn quỹ cuối kỳ
INSERT INTO #Balances
VALUES (LTRIM(dbo.udfLastDay(@CurrentMonth))
+ ' / ' + @CurrentMonth ,N'Tồn quỹ đầu kỳ',
@totalRptAmt, @totalPmtAmt, @balanceAmount, @count +1)

-- Trình bày tình hình tồn quỹ
SELECT DueDate, DescriptionInVietnamese,
Receipt, Payment, BalanceAmount
FROM #Balances ORDER BY RowNumber ASC
GO

-- Xóa bảng tạm
DROP TABLE #Balances
GO
SET NOCOUNT ON

```

Trong đó, hàm udfLastDay được khai báo trong tập tin User-Defined Functions.sql với cấu trúc như ví dụ 10-22.

**Ví dụ 10-22: Khai báo hàm udfLastDay**

```

CREATE FUNCTION [dbo].[udfLastDay]
(
    @CurrentMonthYear char(7)
)
RETURNS tinyint
WITH EXECUTE AS CALLER
AS
BEGIN
    DECLARE @Month TINYINT
    DECLARE @Year SMALLINT
    DECLARE @Day TINYINT
    SET @Month = CAST(LEFT(@CurrentMonthYear, 2) AS TINYINT)
    SET @Year = CAST(RIGHT(@CurrentMonthYear, 4) AS
SMALLINT)
    IF (@Month in (1,3,5,7,8,10,12))

```

M® 122

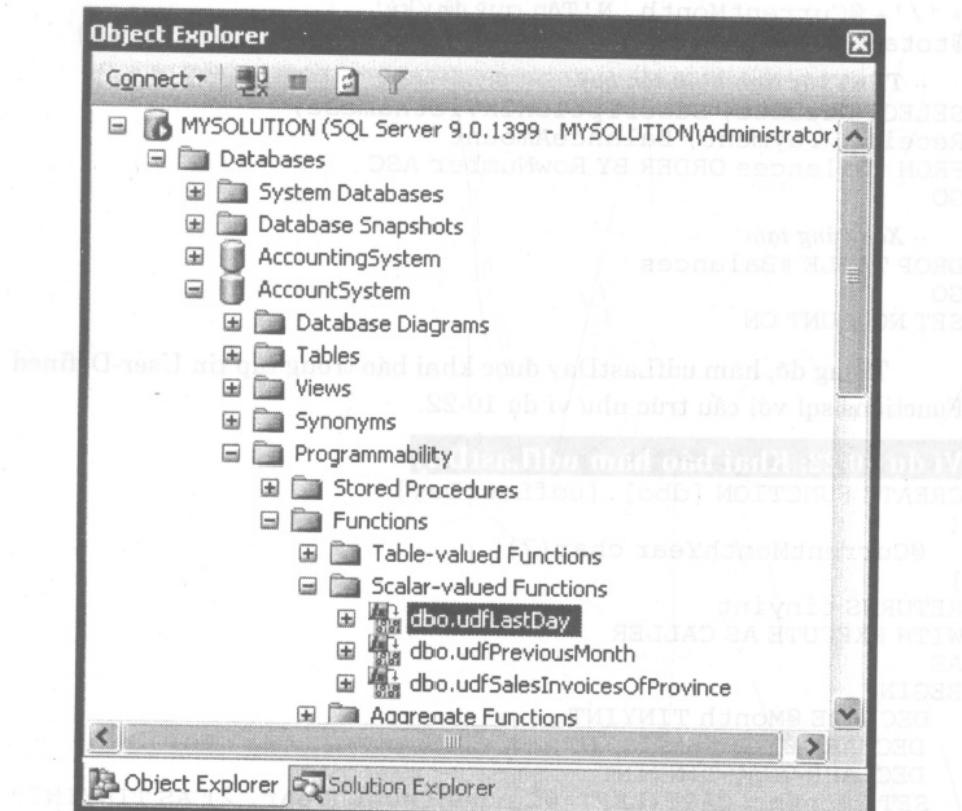
**Chương 10: Khai báo biến và phát biểu điều khiển**

```

SET @Day = 31
ELSE
BEGIN
    IF (@Month = 2)
    BEGIN
        IF @Year%4 = 0 AND @Year%100 = 0
            SET @Day = 29
        ELSE
            SET @Day = 28
    END
    ELSE
        SET @Day = 30
END
RETURN (@Day)
END;

```

Khi thực thi phát biểu CREATE FUNCTION trên, bạn có thể tìm thấy tên hàm udfLastDay xuất hiện trong ngắn như hình 10-18.



**Hình 10-18: Tạo hàm thành công.**

### 3.3.1. Phát biểu điều khiển CONTINUE

Phát biểu CONTINUE cho phép bạn bỏ qua các khai báo ngay sau nó trong vòng lặp WHILE, phát biểu này thường được sử dụng với phát biểu điều khiển WHILE.

Chẳng hạn, để in ra các số chẵn từ 1 đến 10, bạn khai báo phát biểu điều khiển WHILE và CONTINUE như ví dụ 10-23.

#### Ví dụ 10-23: Khai báo sử dụng CONTINUE

```
DECLARE @count int
DECLARE @total int
SET @count = 0
SET @total = 10
WHILE @count<10
BEGIN
    SET @count = @count + 1
    IF (@count%2=0)
        CONTINUE
    SET @total = @total + @count
END
SELECT @total As 'Total'
GO
```

Nếu thực thi phát biểu WHILE trong ví dụ trên, bạn có thể tìm thấy kết quả trình bày như hình 10-19.

| Total |
|-------|
| 35    |

Hình 10-19: Sử dụng phát biểu CONTINUE.

### 3.3.2. Phát biểu điều khiển BREAK

Phát biểu BREAK cho phép bạn thoát ra khỏi phát biểu vòng lặp hay nhánh, phát biểu này thường được sử dụng với phát biểu điều khiển WHILE.

Chẳng hạn, để in ra các số chẵn từ 1 đến 10, bạn khai báo phát biểu điều khiển WHILE và BREAK như ví dụ 10-24.

#### Ví dụ 10-24: Khai báo sử dụng phát biểu BREAK

```
DECLARE @count int
DECLARE @total int
SET @count = 100
SET @total = 10
WHILE @count>0
BEGIN
```

M® 124

**Chương 10: Khai báo biến và phát biểu điều khiển**

```

SET @count = @count + 1
SET @total = @total + 10
IF (@total>20)
    BREAK
END
SELECT @total As 'Total'
GO

```

Nếu thực thi phát biểu WHILE trong ví dụ trên, bạn có thể tìm thấy kết quả trình bày như hình 10-20.

| Total | 30 |
|-------|----|
| 1     |    |

Hình 10-20: Sử dụng phát biểu BREAK.

### 3.4. Phát biểu điều khiển RETURN

Phát biểu RETURN cho phép thoát khỏi lô phát biểu truy vấn hay thủ tục nội tại không điều kiện với cú pháp như sau:

```
RETURN [ integer_expression ]
```

Chẳng hạn, bạn khai báo sử dụng phát biểu RETURN như ví dụ 10-25.

#### Ví dụ 10-25: Khai báo sử dụng phát biểu RETURN

```

DECLARE @count int
DECLARE @total int
SET @count = 100
SET @total = 10
WHILE @count>0
BEGIN
    SET @count = @count + 1
    SET @total = @total + 10
    IF (@total>20)
        RETURN
END
GO

```

Lưu ý: Khi khai báo phát biểu RETURN, bạn có thể không chỉ định giá trị sau phát biểu RETURN nếu bạn đang làm việc với lô phát biểu.

Tuy nhiên, trong trường hợp chỉ định giá trị số nguyên, phát biểu RETURN phải được khai báo trong thủ tục nội tại. Chúng ta sẽ tìm hiểu chi tiết phát biểu RETURN với thủ tục nội tại trong chương kế tiếp.

### 3.5. Phát biểu điều khiển TRY..CATCH

Tương tự như giải pháp kiểm soát lỗi trong ngôn ngữ lập trình C# hay C++, bạn có thể phát biểu TRY..CATCH với cấu trúc như sau:

```
BEGIN TRY
    { sql_statement | statement_block }
END TRY
BEGIN CATCH
    { sql_statement | statement_block }
END CATCH
[ ; ]
```

Bạn có thể khai báo nhóm phát biểu SQL trong khối TRY và khai báo khác khi lỗi xảy ra trong khối CATCH.

Giả sử, bạn có cấu trúc bảng dữ liệu có tên Balances như ví dụ 10-26.

#### Ví dụ 10-26: Khai báo tạo bảng dữ liệu Balances

```
CREATE TABLE Balances
(
    DueDate CHAR(11),
    DescriptionInVietnamese NVARCHAR(150),
    Receipt DECIMAL,
    Payment DECIMAL,
    BalanceAmount DECIMAL
    CONSTRAINT BalanceAmount_check
    CHECK (BalanceAmount >= 0),
    RowNumber int
)
GO
```

Kế đến, bạn khai báo đoạn chương trình bao gồm các phát biểu điều khiển và phát biểu truy vấn để tính tồn quỹ như ví dụ 10-27.

#### Ví dụ 10-27: Khai báo tính tồn quỹ

```
SET NOCOUNT OFF
DELETE FROM Balances;
DECLARE @CurrentMonth CHAR(7)
DECLARE @PreviousMonth CHAR(7)
SET @CurrentMonth = '10/2007'
SET @PreviousMonth = dbo.udfPreviousMonth('10/2007');
WITH BalanceOfToday
AS
(
    SELECT '01/' + @CurrentMonth AS DueDate,
    N'Tồn quỹ đầu kỳ' AS DescriptionInVietnamese,
    0 AS Receipt, 0 AS Payment, 3500000 AS BalanceAmount
    FROM CloseMonthCashBalances
    WHERE CloseMonth=@PreviousMonth
    UNION ALL
```

**M<sup>®</sup> 126****Chương 10: Khai báo biến và phát biểu điều khiển**

```

SELECT Convert(char(11), ReceiptDate, 106) AS DueDate,
DescriptionInVietnamese, ReceiptAmount,
0, 0
FROM Receipts
UNION ALL
SELECT Convert(char(11), PaymentDate, 106) AS DueDate,
DescriptionInVietnamese, 0, PaymentAmount, 0
FROM Payments
)
INSERT INTO Balances
SELECT DueDate, DescriptionInVietnamese,
Receipt, Payment, BalanceAmount
, ROW_NUMBER() OVER(ORDER BY DueDate) AS RowNumber
FROM BalanceOfToday
ORDER BY RowNumber ASC
DECLARE @count int
DECLARE @rptAmount int, @totalRptAmt int
DECLARE @pmtAmount int, @totalPmtAmt int
DECLARE @balanceAmount int
SET @balanceAmount = 0
SET @totalRptAmt = 0
SET @totalPmtAmt = 0
SET @count = 1
WHILE(exists(SELECT * FROM Balances WHERE
RowNumber=@count+1))
BEGIN
    SELECT @balanceAmount= BalanceAmount FROM Balances
    WHERE RowNumber=@count
    SELECT @rptAmount = Receipt, @pmtAmount = Payment
    FROM Balances WHERE RowNumber=@count +1
    SET @balanceAmount = @balanceAmount + @rptAmount -
    @pmtAmount
    SET @totalRptAmt = @totalRptAmt + @rptAmount
    SET @totalPmtAmt = @totalPmtAmt + @pmtAmount
    UPDATE Balances SET BalanceAmount = @balanceAmount
    WHERE RowNumber=@count+1
    SET @count = @count + 1
END
INSERT INTO Balances
VALUES(LTRIM(dbo.udfLastDay(@CurrentMonth))
+ '/' + @CurrentMonth ,N'Tôn quỹ cuối kỳ',
@totalRptAmt, @totalPmtAmt, @balanceAmount, @count +1)

SELECT DueDate, DescriptionInVietnamese,
Receipt, Payment, BalanceAmount
FROM Balances ORDER BY RowNumber ASC
SET NOCOUNT ON
GO

```

Trong đó, tôn quỹ đầu kỳ là 7000000, khi cập nhật cột BalanceAmount, giá trị có thể là âm, khi đó lỗi sẽ phát sinh như hình 10-21.

## Chương 10: Khai báo biến và phát biểu điều khiển

127



```
(1 row(s) affected)

(1 row(s) affected)
Msg 547, Level 16, State 0, Line 50
The UPDATE statement conflicted with the
CHECK constraint "BalanceAmount_check".
The conflict occurred in database "AccountSystem",
table "dbo.Balances", column 'BalanceAmount'.
The statement has been terminated.

Msg 547, Level 16, State 0, Line 50
The UPDATE statement conflicted with the
CHECK constraint "BalanceAmount_check".
The conflict occurred in database "AccountSystem",
table "dbo.Balances", column 'BalanceAmount'.
```

**Query completed with errors.**

Hình 10-21: Lỗi phát sinh do CONSTRAINT.

Kết quả trình bày ứng với các mẫu tin cập nhật thành công, những mẫu tin cập nhật không thành công sẽ có giá trị là 0 tại cột BalanceAmount như hình 10-22.

|    | DueDate     | DescriptionInVietnamese              | Rece... | Payment | BalanceAmount |
|----|-------------|--------------------------------------|---------|---------|---------------|
| 1  | 01/10/2007  | Tồn quỹ đầu kỳ                       | 0       | 0       | 35000000      |
| 2  | 06 Oct 2007 | Trả tiền mua hàng Ajinomoto Việt Nam | 0       | 5030000 | 29970000      |
| 3  | 06 Oct 2007 | Trả tiền mua hàng Suzumi Việt Nam    | 0       | 5132500 | 24837500      |
| 4  | 06 Oct 2007 | Trả tiền mua hàng                    | 0       | 7000000 | 17837500      |
| 5  | 07 Oct 2007 | Trả tiền mua hàng                    | 0       | 1980000 | 15857500      |
| 6  | 07 Oct 2007 | Trả tiền mua hàng Ajinomoto Việt Nam | 0       | 7000000 | 8857500       |
| 7  | 07 Oct 2007 | Trả tiền mua hàng                    | 0       | 2000000 | 6857500       |
| 8  | 08 Oct 2007 | Trả tiền mua hàng                    | 0       | 2000000 | 4857500       |
| 9  | 09 Oct 2007 | Trả tiền mua hàng                    | 0       | 1500000 | 3357500       |
| 10 | 09 Oct 2007 | Trả tiền mua hàng                    | 0       | 1500000 | 1857500       |
| 11 | 09 Oct 2007 | Trả tiền mua hàng                    | 0       | 1500000 | 357500        |
| 12 | 10 Oct 2007 | Trả tiền mua hàng                    | 0       | 3500000 | 0             |
| 13 | 10 Oct 2007 | Trả tiền mua hàng                    | 0       | 2500000 | 0             |

Hình 10-22: Mẫu tin cập nhật thành công.

Để tránh lỗi phát sinh, bạn có thể cài đặt phát biểu TRY..CATCH như ví dụ 10-28.

**M® 128****Chương 10: Khai báo biến và phát biểu điều khiển****Ví dụ 10-28: Khai báo phát biến TRY..CATCH**

```

SET NOCOUNT OFF
DELETE FROM Balances;
DECLARE @CurrentMonth CHAR(7)
DECLARE @PreviousMonth CHAR(7)
SET @CurrentMonth = '10/2007'
SET @PreviousMonth = dbo.udfPreviousMonth('10/2007');

-- Khai báo biến thức bảng
WITH BalanceOfToday
AS
(
    SELECT '01/' + @CurrentMonth AS DueDate,
    N'Tồn quỹ đầu kỳ' As DescriptionInVietnamese,
    0 as Receipt, 0 as Payment, 35000000 As BalanceAmount
    FROM CloseMonthCashBalances
    WHERE CloseMonth=@PreviousMonth
    UNION ALL
    SELECT Convert(char(11), ReceiptDate, 106) AS DueDate,
    DescriptionInVietnamese, ReceiptAmount,
    0, 0
    FROM Receipts
    UNION ALL
    SELECT Convert(char(11), PaymentDate, 106) AS DueDate,
    DescriptionInVietnamese, 0, PaymentAmount, 0
    FROM Payments
)
INSERT INTO Balances
SELECT DueDate, DescriptionInVietnamese,
Receipt, Payment, BalanceAmount
, ROW_NUMBER() OVER(ORDER BY DueDate) AS RowNumber
FROM BalanceOfToday
ORDER BY RowNumber ASC
DECLARE @count int
DECLARE @rptAmount int, @totalRptAmt int
DECLARE @pmtAmount int, @totalPmtAmt int
DECLARE @balanceAmount int
SET @balanceAmount = 0
SET @totalRptAmt = 0
SET @totalPmtAmt = 0
SET @count = 1
WHILE(exists(SELECT * FROM Balances
WHERE RowNumber=@count+1))
BEGIN
    SELECT @balanceAmount= BalanceAmount FROM Balances
    WHERE RowNumber=@count
    SELECT @rptAmount = Receipt, @pmtAmount = Payment
    FROM Balances
    WHERE RowNumber=@count +1
    SET @balanceAmount = @balanceAmount + @rptAmount -
    @pmtAmount
)

```

## Chương 10: Khai báo biến và phát biểu điều khiển

129 M®

```

SET @totalRptAmt = @totalRptAmt + @rptAmount
SET @totalPmtAmt = @totalPmtAmt + @pmtAmount
-- Cài đặt phát biểu TRY..CATCH
BEGIN TRY
    UPDATE Balances
    SET BalanceAmount = @balanceAmount
    WHERE RowNumber=@count+1
END TRY
BEGIN CATCH
    SELECT
        ERROR_NUMBER() AS ErrorNumber,
        ERROR_SEVERITY() AS ErrorSeverity,
        ERROR_STATE() AS ErrorState,
        ERROR_PROCEDURE() AS ErrorProcedure,
        ERROR_LINE() AS ErrorLine,
        ERROR_MESSAGE() AS ErrorMessage;
    END CATCH
    SET @count = @count + 1
END
INSERT INTO Balances
VALUES (LTRIM(dbo.udfLastDay (@CurrentMonth))
+ '/' + @CurrentMonth ,N'Tồn quỹ cuối kỳ',
@totalRptAmt, @totalPmtAmt, @balanceAmount,@count+1)
SELECT DueDate, DescriptionInVietnamese,
Receipt, Payment, BalanceAmount
FROM Balances ORDER BY RowNumber ASC
SET NOCOUNT ON
GO

```

Khi thực thi những phát biểu trong ví dụ trên, bạn có thể tìm thấy kết quả trình bày trong ngăn Result gồm hai phần, phần thứ nhất là thông tin của lỗi trông giống như hình 10-23.

| Results     |               |            |                |           |   |
|-------------|---------------|------------|----------------|-----------|---|
| Messages    |               |            |                |           |   |
| ErrorNumber | ErrorSeverity | ErrorState | ErrorProcedure | ErrorLine | ErrorMessage  |
| 1           | 547           | 16         | 0              | NULL      | 51  |
|             |               |            |                |           | The UPDATE statement conflicted with the CHECK c... |
|             |               |            |                |           | The UPDATE statement conflicted with the CHECK c... |
|             |               |            |                |           | The UPDATE statement conflicted with the CHECK c... |

Hình 10-23: Kiểm soát lỗi.

Phần thứ hai là kết quả trình bày tình hình thu và chi trong tháng trông giống như hình 10-24.

## Chương 10: Khai báo biến và phát biểu điều khiển

| DueDate     | Description in Vietnamese            | Rece... | Payment | BalanceAmount |
|-------------|--------------------------------------|---------|---------|---------------|
| 01/10/2007  | Tồn quỹ đầu kỳ                       | 0       | 0       | 35000000      |
| 06 Oct 2007 | Trả tiền mua hàng Ajinomoto Việt Nam | 0       | 5030000 | 29970000      |
| 06 Oct 2007 | Trả tiền mua hàng Suzumi Việt Nam    | 0       | 5132500 | 24837500      |
| 06 Oct 2007 | Trả tiền mua hàng                    | 0       | 7000000 | 17837500      |
| 07 Oct 2007 | Trả tiền mua hàng                    | 0       | 1980000 | 15857500      |
| 07 Oct 2007 | Trả tiền mua hàng Ajinomoto Việt Nam | 0       | 7000000 | 8857500       |
| 07 Oct 2007 | Trả tiền mua hàng                    | 0       | 2000000 | 6857500       |
| 08 Oct 2007 | Trả tiền mua hàng                    | 0       | 2000000 | 4857500       |
| 09 Oct 2007 | Trả tiền mua hàng                    | 0       | 1500000 | 3357500       |
| 09 Oct 2007 | Trả tiền mua hàng                    | 0       | 1500000 | 1857500       |
| 09 Oct 2007 | Trả tiền mua hàng                    | 0       | 1500000 | 357500        |
| 10 Oct 2007 | Trả tiền mua hàng                    | 0       | 3500000 | 0             |
| 10 Oct 2007 | Trả tiền mua hàng                    | 0       | 2500000 | 0             |
| 10 Oct 2007 | Thu tiền bán hàng của khách hàng ... | 5000... | 0       | 500000        |
| 11 Oct 2007 | Thu tiền bán hàng của khách hàng ... | 4500... | 0       | 950000        |
| 11 Oct 2007 | Thu tiền bán hàng của khách hàng ... | 1055... | 0       | 2005000       |
| 11 Oct 2007 | Trả tiền mua hàng                    | 0       | 6177500 | 0             |

Hình 10-24: Tình hình thu và chi trong tháng.

### 3.6. Phát biểu điều khiển WAITFOR

Phát biểu điều khiển WAITFOR ngăn chặn lô, phát biểu SQL hay thủ tục cho đến thời gian đã chỉ định mới thực thi chúng.

WAITFOR

```
{
    DELAY 'time_to_pass'
    | TIME 'time_to_execute'
    | (receive_statement) [ , TIMEOUT timeout ]
}
```

Trong đó, tham số DELAY là thời gian chờ, TIME là thời gian cụ thể hay TIMEOUT là thời gian hết hạn chờ của phát biểu WAITFOR.

Bạn nên dùng phát biểu WAITFOR để thực thi phát biểu SQL, thủ tục nội tại với thời gian chỉ định.

Chẳng hạn, bạn cần thực thi phát biểu DELETE để xóa dữ liệu trong bảng ImportDetailsForBackup vào đúng 9:25 sáng của ngày hiện hành. Trước tiên, bạn kiểm tra dữ liệu trong bảng ImportDetailsForBackup như ví dụ 10-29.

#### Ví dụ 10-29: Khai báo kiểm tra dữ liệu

```
SELECT * FROM ImportDetailsForBackup
GO
```

## Chương 10: Khai báo biến và phát biểu điều khiển

131 M®

Khi thực thi phát biểu SQL trong ví dụ trên, bạn có thể tìm thấy danh sách mẫu tin trong bảng ImportDetailsForBackup trình bày như hình 10-25.

|   | OrdinalNumber | ImportNo  | ProductID | Stock... | Quantity |
|---|---------------|-----------|-----------|----------|----------|
| 1 | 1             | I00000101 | P00001    | ST001    | 500      |
| 2 | 1             | I00000102 | P00001    | ST001    | 400      |
| 3 | 1             | I00000103 | P00001    | ST002    | 400      |

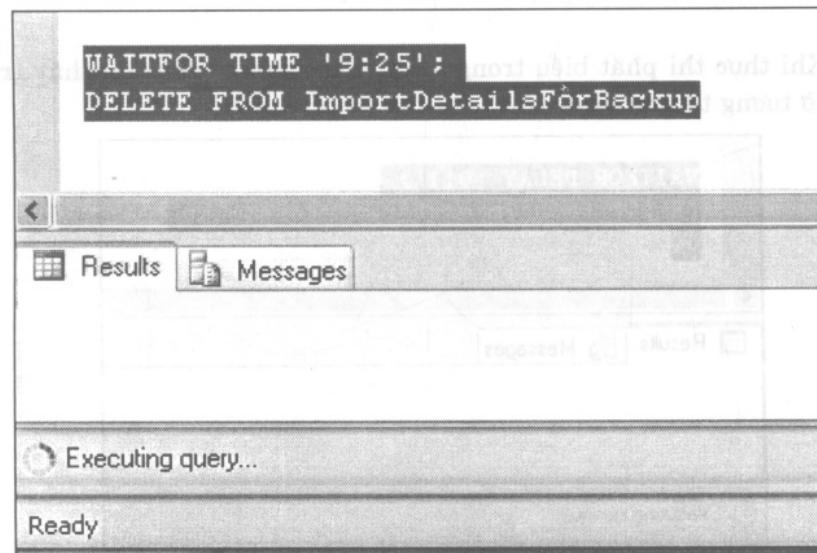
Hình 10-25: Danh sách mẫu tin trong bảng ImportDetailsForBackup.

Kế đến, bạn khai báo phát biểu DELETE với WAITFOR để xóa dữ liệu trong bảng ImportDetailsForBackup như ví dụ 10-30.

**Ví dụ 10-30: Khai báo xóa mẫu tin**

```
WAITFOR TIME '9:25';
DELETE FROM ImportDetailsForBackup
```

Khi thực thi phát biểu DELETE trong ví dụ trên tại thời điểm trước 9:25, bạn có thể tìm thấy trạng thái thực thi đang chờ đến thời gian là 9:25.



Hình 10-26: Chờ thực thi theo thời gian chỉ định.

Vào đúng thời gian 9:25, phát biểu DELETE sẽ được thực thi và ba mẫu tin sẽ bị xóa và kết quả trình bày như hình 10-27.

132

**Chương 10: Khai báo biến và phát biểu điều khiển**

```

WAITFOR TIME '9:25';
DELETE FROM ImportDetailsForBackup;

```

(3 row(s) affected)

Query executed successfully.

Ready

**Hình 10-27: Xóa mẫu tin.**

Trong trường hợp bạn yêu cầu thực thi phát biểu SQL ngay sau khi thời gian (tính theo thời gian) chỉ định kết thúc thì sử dụng từ khóa DELAY như ví dụ 10-31.

**Ví dụ 10-31: Khai báo sử dụng từ khóa DELAY**

```

WAITFOR DELAY '0:1';
DELETE FROM ImportDetailsForBackup
GO

```

Khi thực thi phát biểu trong ví dụ trên, bạn có thể tìm thấy trạng thái chờ tương tự như hình 10-28.

```

WAITFOR DELAY '0:1';
DELETE FROM ImportDetailsForBackup
GO

```

Executing query...

**Hình 10-28: Trạng thái chờ.**

Ngay sau 1 phút trôi qua thì phát biểu DELETE sẽ được thực thi, kết quả trình bày như hình 10-29.

**Chương 10: Khai báo biến và phát biểu điều khiển**

133

```

WAITFOR DELAY '0:1';
DELETE FROM ImportDetailsForBackup
GO

```

Messages

(18 row(s) affected)

Query executed successfully.

**Hình 10-29: Thực thi phát biểu DELETE.**

Chú ý: Bạn có thể tạo dữ liệu mẫu trong bảng ImportDetailsForBackup bằng cách thực thi phát biểu INSERT và SELECT từ bảng ImportDetails như ví dụ 10-32.

**Ví dụ 10-32: Khai báo tạo dữ liệu mẫu**

```

INSERT INTO ImportDetailsForBackup
SELECT * FROM ImportDetails
GO

```

**3.7. Phát biểu điều khiển GOTO**

SQL Server 2005 giới thiệu phát biểu điều khiển GOTO LABEL cho phép bạn yêu cầu trình thực thi bỏ qua các khai báo sau phát biểu GOTO để nhảy đến nhãn định nghĩa trước.

Define the label:

```

label : Alter the execution:
        GOTO label

```

Chẳng hạn, bạn tính tồn quỹ đến ngày chỉ định là cuối ngày 08 tháng 10 năm 2007 thì khai báo như ví dụ 10-33.

**Ví dụ 10-33: Khai báo tính tồn quỹ**

```

-- Khai báo biến ứng với ngày chỉ định
DECLARE @specifyDate CHAR(11)
DECLARE @balanceDate CHAR(11)
SET @specifyDate = '09 Oct 2007'

-- Khai báo biến ứng với tình hình thu, chi và tồn quỹ
DECLARE @count int
DECLARE @rptAmount int, @totalRptAmt int
DECLARE @pmtAmount int, @totalPmtAmt int
DECLARE @balanceAmount int
SET @balanceAmount = 0

```

**M<sup>®</sup> 134****Chương 10: Khai báo biến và phát biểu điều khiển**

```

SET @totalRptAmt = 0
SET @totalPmtAmt = 0
SET @count = 1
SELECT @balanceAmount= BalanceAmount
FROM Balances
WHERE RowNumber=@count

-- Khai báo tính cân đối thu chi
WHILE(exists(SELECT * FROM Balances WHERE
RowNumber=@count+1))
BEGIN
    SELECT
        @balanceDate = DueDate,
        @rptAmount = Receipt,
        @pmtAmount = Payment
    FROM Balances

    WHERE RowNumber=@count +1

    -- Khai báo phát biểu điều khiển GOTO
    IF @specifyDate = @balanceDate
        GOTO Final
    SET @balanceAmount = @balanceAmount + @rptAmount -
    @pmtAmount
    SET @totalRptAmt = @totalRptAmt + @rptAmount
    SET @totalPmtAmt = @totalPmtAmt + @pmtAmount

    -- Khai báo cập nhật cân đối
    UPDATE Balances SET BalanceAmount = @balanceAmount
    WHERE RowNumber=@count+1
    SET @count = @count + 1
    END

    -- Khai báo nhãn Final
Final:
    SELECT DueDate, DescriptionInVietnamese,
    Receipt, Payment, BalanceAmount, RowNumber
    INTO #Balances
    FROM Balances
    WHERE RowNumber<@count+1
    ORDER BY RowNumber ASC
    INSERT INTO #Balances
    SELECT @specifyDate , N'Tồn quỹ đến ngày',
    SUM(Receipt), SUM(Payment), @balanceAmount,@count+2
    FROM #Balances
    SELECT DueDate, DescriptionInVietnamese,
    Receipt, Payment, BalanceAmount
    FROM #Balances
    order by RowNumber
    GO
    drop TABLE #Balances
    GO

```

**Chương 10: Khai báo biến và phát biểu điều khiển**

135 M®

Khi thực thi phát biểu SQL trong ví dụ trên, bạn có thể tìm thấy danh sách mẫu tin ứng với tình hình thu và chi lần tồn quỹ trước ngày 9 tháng 10 năm 2007 trình bày như hình 10-30.

|   | DueDate     | DescriptionInVietnamese              | Rece... | Payment  | BalanceAmount |
|---|-------------|--------------------------------------|---------|----------|---------------|
| 1 | 01/10/2007  | Tồn quỹ đầu kỳ                       | 0       | 0        | 75000000      |
| 2 | 06 Oct 2007 | Trả tiền mua hàng Ajinomoro Việt Nam | 0       | 5030000  | 69970000      |
| 3 | 06 Oct 2007 | Trả tiền mua hàng Suzumi Việt Nam    | 0       | 5132500  | 64837500      |
| 4 | 06 Oct 2007 | Trả tiền mua hàng                    | 0       | 7000000  | 57837500      |
| 5 | 07 Oct 2007 | Trả tiền mua hàng                    | 0       | 1980000  | 55857500      |
| 6 | 07 Oct 2007 | Trả tiền mua hàng Ajinomoro Việt Nam | 0       | 7000000  | 48857500      |
| 7 | 07 Oct 2007 | Trả tiền mua hàng                    | 0       | 2000000  | 46857500      |
| 8 | 08 Oct 2007 | Trả tiền mua hàng                    | 0       | 2000000  | 44857500      |
| 9 | 09 Oct 2007 | Tồn quỹ đến ngày                     | 0       | 30142... | 44857500      |

**Hình 10-30: Tình hình tồn quỹ.**

Chú ý: Dữ liệu trong bảng Balances đã được tính toán trong những ví dụ trước.

## 4. HÀM CASE

Hàm CASE cho phép bạn xét điều kiện và trả về một trong nhiều giá trị cho trước, hàm CASE có hai định dạng.

Định dạng thứ nhất là dùng hàm CASE để so sánh biểu thức với biểu thức hay giá trị.

```
CASE input_expression
    WHEN when_expression THEN result_expression
    [ ...n ]
    [
    ELSE else_result_expression
    ]
END
```

Chẳng hạn, bạn có thể liệt kê danh sách khách hàng thuộc tỉnh thành chỉ định, nếu tỉnh thành là rỗng thì bạn liệt kê danh sách tất cả khách hàng trong bảng Customers.

Bạn khai báo mệnh đề WHERE với cột ProvinceId để liệt kê danh sách khách hàng có cột ProvinceId bằng với giá trị truyền vào của biến như ví dụ 10-34.

**Ví dụ 10-34: Khai báo lọc danh sách khách hàng**

```
DECLARE @ProvinceId CHAR (3)
SET @ProvinceId='HCM'
```

**Chương 10:** Khai báo biến và phát biểu điều khiển

```

SELECT
    CustomerId, CompanyNameInVietnamese
FROM Customers
WHERE ProvinceId = @ProvinceId
GO
  
```

Khi thực thi phát biểu trên, bạn có thể tìm thấy danh sách khách hàng có mã tỉnh thành là HCM như hình 10-31.

| CustomerId | CompanyNameInVietnamese                       |
|------------|---|
| 1 A0001    | Công ty Trách Nhiệm Hữu Hạn Microsoft Vietnam |
| 2 A0002    | Công ty Trách Nhiệm Hữu Hạn IBM Vietnam       |
| 3 A0005    | Công ty Cổ phần Suzumi Vietnam                |
| 4 A0007    | Công ty Đa quốc gia UFC&A                     |
| 5 A0008    | Công ty Cổ phần ReruitVietnam                 |
| 6 A0010    | Công ty Trách Nhiệm Hữu Hạn Hot Getways       |

**Hình 10-31:** Khách hàng có mã là HCM.

Trong trường hợp bạn gán giá trị cho biến @ProvinceId có giá trị là rỗng thì trình bày như ví dụ 10-35.

**Ví dụ 10-35: Khai báo lọc danh sách khách hàng**

```

DECLARE @ProvinceId CHAR (3)
SET @ProvinceId= ''
SELECT
    CustomerId, CompanyNameInVietnamese
FROM Customers
WHERE ProvinceId = @ProvinceId
GO
  
```

Nếu thực thi phát biểu trên, bạn sẽ không tìm thấy khách hàng có mã tỉnh thành là rỗng như hình 10-32.

| CustomerId | CompanyNameInVietnamese |
|------------|-------------------------|
|------------|-------------------------|

**Hình 10-32:** Không tồn tại khách hàng có mã là rỗng.

Để có thể cho phép người sử dụng liệt kê tất cả khách hàng trong bảng Customers khi biến @ProvinceId có giá trị là rỗng, bạn khai báo ví dụ trên như ví dụ 10-36.

**Chương 10: Khai báo biến và phát biểu điều khiển**

137

**Ví dụ 10-36: Khai báo lọc khách hàng**

```

DECLARE @ProvinceId CHAR(3)
SET @ProvinceId='HCM'
SELECT
    CustomerId, CompanyNameInVietnamese
FROM Customers
WHERE ProvinceId =
CASE @ProvinceId
    WHEN '' THEN ProvinceId
    ELSE @ProvinceId
END
GO

```

Khi thực thi phát biểu trên, bạn có thể tìm thấy danh sách khách hàng có mã tỉnh thành là HCM như hình 10-33.

| CustomerId | CompanyNameInVietnamese                       |
|------------|---|
| 1          | Công ty Trách Nhiệm Hữu Hạn Microsoft Vietnam |
| 2          | Công ty Trách Nhiệm Hữu Hạn IBM Vietnam       |
| 3          | Công ty Cổ phần Suzumi Vietnam                |
| 4          | Công ty Đa quốc gia UFCA                      |
| 5          | Công ty Cổ phần ReruitVietnam                 |
| 6          | Công ty Trách Nhiệm Hữu Hạn Hot Getways       |

Query executed successfully.

**Hình 10-33:** Danh sách khách hàng có mã tỉnh thành là HCM.

Tuy nhiên, bạn khai báo để gán giá trị cho biến @ProvinceId là rỗng như ví dụ 10-37.

**Ví dụ 10-37: Khai báo gán giá trị rỗng cho biến @ProvinceId**

```

DECLARE @ProvinceId CHAR(3)
SET @ProvinceId=''
SELECT
    CustomerId, CompanyNameInVietnamese
FROM Customers
WHERE ProvinceId =
CASE @ProvinceId
    WHEN '' THEN ProvinceId
    ELSE @ProvinceId
END
GO

```

Khi thực thi phát biểu trên, bạn có thể tìm thấy danh sách khách hàng như hình 10-34.

M® 138

**Chương 10: Khai báo biến và phát biểu điều khiển**

| CustomerId | CompanyNameInVietnamese                       |
|------------|---|
| 1 A0001    | Công ty Trách Nhiệm Hữu Hạn Microsoft Vietnam |
| 2 A0002    | Công ty Trách Nhiệm Hữu Hạn IBM Vietnam       |
| 3 A0003    | Công ty Trách Nhiệm Hữu Hạn Kodaka Vietnam    |
| 4 A0004    | Công ty Trách Nhiệm Hữu Hạn E-Google Vietnam  |
| 5 A0005    | Công ty Cổ phần Suzumi Vietnam                |
| 6 A0006    | Tập đoàn UCIA USA                             |
| 7 A0007    | Công ty Đa quốc gia UFCA                      |
| 8 A0008    | Công ty Cổ phần FeruitVietnam,                |
| 9 A0009    | Trung tâm giáo dục Vietnam                    |
| 10 A0010   | Công ty Trách Nhiệm Hữu Hạn Hot Getways       |

**Hình 10-34: Danh sách khách hàng.**

Với cách sử dụng hàm CASE như trên, bạn có thể áp dụng cách khai báo này trong trang tìm kiếm, nếu người sử dụng có chọn điều kiện để tìm kiếm thì mẫu tin sẽ lọc theo giá trị chọn, trong trường hợp họ không chọn thì chúng ta sẽ không xem xét đến cột liên quan.

Dạng thứ hai là sử dụng hàm CASE để kiểm tra biểu thức luận lý.  
CASE

```
WHEN Boolean_expression THEN result_expression
[ ...n ]
[
ELSE else_result_expression
]
END
```

Chẳng hạn, bạn có thể thay đổi cách sử dụng hàm CASE trong ví dụ trên thành biểu thức luận lý như ví dụ 10-38.

**Ví dụ 10-38: Khai báo sử dụng hàm CASE dạng thứ hai**

```
DECLARE @ProvinceId CHAR(3)
SET @ProvinceId = ''
SELECT
    CustomerId, CompanyNameInVietnamese
FROM Provinces P, Customers C
WHERE P.ProvinceId = C.ProvinceId
AND C.ProvinceId =
CASE WHEN @ProvinceId = ''
    THEN C.ProvinceId
    ELSE @ProvinceId
END
GO
```

**Chương 10: Khai báo biến và phát biểu điều khiển**

139

Khi thực thi phát biểu SELECT với hàm CASE trong ví dụ trên, nếu giá trị của biến @ProvinceId là rỗng thì kết quả trình bày như hình 10-35.

| CustomerId | CompanyNameInVietnamese                       |
|------------|---|
| 1          | Công ty Trách Nhiệm Hữu Hạn Microsoft Vietnam |
| 2          | Công ty Trách Nhiệm Hữu Hạn IBM Vietnam       |
| 3          | Công ty Trách Nhiệm Hữu Hạn Kodaka Vietnam    |
| 4          | Công ty Trách Nhiệm Hữu Hạn E-Google Vietnam  |
| 5          | Công ty Cổ phần Suzumi Vietnam                |
| 6          | Tập đoàn UCIA USA                             |
| 7          | Công ty Đa quốc gia UFCA                      |
| 8          | Công ty Cổ phần ReruitVietnam                 |
| 9          | Trung tâm giáo dục Vietnam                    |
| 10         | Công ty Trách Nhiệm Hữu Hạn Hot Getways       |

Hình 10-35: Danh sách khách hàng.

Trong trường hợp bạn gán giá trị cho biến @ProvinceId là HAN thì kết quả trình bày như hình 10-36.

```

DECLARE @ProvinceId CHAR(3)
SET @ProvinceId='HAN'
SELECT
    CustomerId, CompanyNameInVietnamese
FROM Provinces P, Customers C
WHERE P.ProvinceId = C.ProvinceId
AND C.ProvinceId =
CASE WHEN @ProvinceId = ''
    THEN C.ProvinceId
    ELSE @ProvinceId |
END
GO

```

| CustomerId | CompanyNameInVietnamese                      |
|------------|--|
| 1          | Công ty Trách Nhiệm Hữu Hạn Kodaka Vietnam   |
| 2          | Công ty Trách Nhiệm Hữu Hạn E-Google Vietnam |

Hình 10-36: Danh sách khách hàng có mã tỉnh thành là HAN.

Bạn có thể khai báo biểu thức phức tạp hơn như ví dụ 10-39.

**Ví dụ 10-39: Khai báo hàm CASE**

WITH [Sales Quantity Range]

AS

(

140

**Chương 10: Khai báo biến và phát biểu điều khiển**

```

SELECT D.ProductId, ProductNameInVietnamese,
       SUM(Quantity) AS TotalQuantity
  FROM Products P, SalesInvoiceDetails D
 WHERE P.ProductId = D.ProductId
 GROUP BY D.ProductId, ProductNameInVietnamese
)
SELECT ProductId, ProductNameInVietnamese,
       TotalQuantity, 'Sales Quantity Range' =
CASE
    WHEN TotalQuantity < 100 THEN 'Low sales'
    WHEN TotalQuantity >= 100 and TotalQuantity < 300 THEN 'Normal sales'
    WHEN TotalQuantity >= 300 and TotalQuantity <= 700 THEN 'Good sales'
    ELSE 'Best sales'
END
FROM [Sales Quantity Range]
ORDER BY TotalQuantity DESC ;
GO

```

Khi thực thi phát biểu SELECT với hàm CASE trong ví dụ trên, kết quả trình bày như hình 10-37.

| ProductId | ProductNameInVietnamese              | TotalQuantity | Sales Quantity Range |
|-----------|--------------------------------------|---------------|----------------------|
| 1 P00002  | Túi xách dùng cho học sinh nữ        | 750           | Best sales           |
| 2 P00003  | Túi xách dùng cho học sinh nam       | 625           | Good sales           |
| 3 P00001  | Túi xách                             | 515           | Good sales           |
| 4 P00004  | Túi áo mưa                           | 176           | Normal sales         |
| 5 P00005  | Túi xách dùng cho Máy tính           | 130           | Normal sales         |
| 6 P00006  | Túi xách dùng cho Điện thoại di động | 90            | Low sales            |

Hình 10-37: Hàm CASE phức tạp.

Chú ý: Chúng ta sẽ tìm hiểu cách sử dụng hàm CASE trước khi khai báo thủ tục nội tại trong chương kế tiếp.

## 5. KẾT CHƯƠNG

Chúng ta vừa tìm hiểu cách khai báo biến kiểu dữ liệu TABLE, phát biểu điều khiển và cách sử dụng hàm CASE.

Trong chương kế tiếp, chúng ta sẽ tập trung tìm hiểu cách khai báo và sử dụng thủ tục nội tại từ đơn giản đến phức tạp.

## Chương 11:

# KHÁM PHÁ THỦ TỤC NỘI TẠI

### Tóm tắt chương 11

Trong chương này chúng ta sẽ tìm hiểu cách khai báo và sử dụng thủ tục nội tại (Stored Procedure) trong ứng dụng cơ sở dữ liệu SQL Server 2005.

Ngoài ra, chúng ta cũng tìm hiểu cách khai báo và truyền giá trị cho tham số vào thủ tục nội tại cùng với hình thức mượn quyền của người sử dụng khác để thực thi thủ tục nội tại.

#### Các vấn đề chính sẽ được đề cập:

- ✓ Giới thiệu thủ tục nội tại.
- ✓ Thủ tục nội tại với tham số.
- ✓ Mượn quyền thực thi thủ tục nội tại.
- ✓ Thủ tục nội tại với giá trị trả về.
- ✓ Thủ tục nội tại để thêm, xóa, cập nhật dữ liệu.
- ✓ Thủ tục nội tại để truy vấn dữ liệu.

## 1. GIỚI THIỆU THỦ TỤC NỘI TẠI

Thủ tục nội tại (Stored procedure) trong Microsoft SQL Server tương tự như thủ tục hay hàm trong các ngôn ngữ lập trình khác. Nó chứa khai báo và phát biểu T-SQL, có thể chấp nhận tham số truyền từ bên ngoài vào và trả về tập giá trị.

Khi gọi thủ tục nội tại, bạn có thể trả về giá trị hay thông tin về lý do phát sinh lỗi.

Lưu ý: Bạn có thể tìm thấy các ví dụ chứa trong tập tin có tên CreateProcedureStatement.sql, ProcedureUseForInsertAnalUpdate và ProcedureUseForSelect.sql.

Tuy phát biểu CREATE PROCEDURE cho phép bạn tạo thủ tục nội tại với nhiều loại phát biểu SQL nhưng không bao gồm các phát biểu CREATE như sau: CREATE AGGREGATE, CREATE RULE, CREATE DEFAULT, CREATE SCHEMA, CREATE hay ALTER FUNCTION, CREATE hay ALTER TRIGGER, CREATE hay ALTER PROCEDURE, CREATE hay ALTER VIEW, SET PARSEONLY, SET SHOWPLAN\_ALL, SET SHOWPLAN\_TEXT, SET SHOWPLAN\_XML và USE DATABASE\_NAME.

Chú ý: Nếu bảng dữ liệu tạm được tạo ra trong thủ tục nội tại thì chúng chỉ có tầm vực sử dụng trong nội bộ thủ tục nội tại đó.

Nếu khai báo và sử dụng thủ tục nội tại trong SQL Server thay vì sử dụng phát biểu SQL, bạn có thể có các lợi ích như sau:

- ✓ Thủ tục nội tại sẽ được đăng ký và lưu trong cơ sở dữ liệu (dung lượng lớn nhất cho phép là 128 megabytes (MB)), bạn chỉ viết một lần và gọi sử dụng nhiều lần khi cần.
- ✓ Bạn có thể truyền tham số (số lượng tham số lớn nhất cho phép là 2100) để có thể thực thi phát biểu SQL và kết quả trả về tùy biến hơn sử dụng đối tượng View.
- ✓ Sử dụng tham số như một phần xử lý dấu nháy đơn để chống tấn công bằng kỹ thuật SQL Injection.
- ✓ Sử dụng tham số như một phần xử lý chuỗi Unicode. Nếu bạn sử dụng phát biểu SQL thì phải chèn ký tự N đầu giá trị dạng Unicode.
- ✓ Cấp quyền sử dụng trên thủ tục nội tại cho từng nhóm người sử dụng hay người sử dụng cụ thể.
- ✓ Khi khai báo và sử dụng thủ tục nội tại, bạn có thể tránh được quá tải trên đường truyền, do chúng được thực thi trên phia trình chủ.

### 1.1. Các loại thủ tục nội tại

Có 3 nhóm thủ tục nội tại giới thiệu trong phiên bản SQL Server 2005, nhóm thứ nhất là do người sử dụng tạo ra. Nó bao gồm hai loại, loại thủ tục nội tại được tạo ra bởi người sử dụng và lưu trong cơ sở dữ liệu, chúng

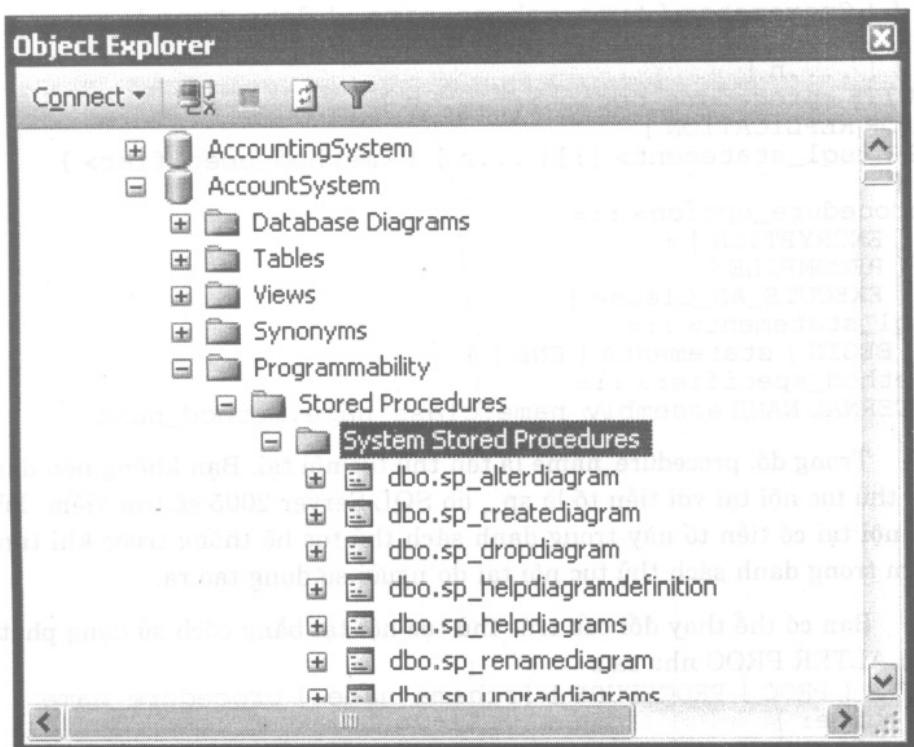
**Chương 11: Khám phá thủ tục nội tại**

143

chứa phát biểu SQL và khai báo bằng T-SQL. Loại thứ hai được khai báo và tạo ra bằng ngôn ngữ lập trình .NET, chúng ta sẽ tìm hiểu quan hệ giữa SQL Server và ngôn ngữ lập trình C# trong tập tiếp theo.

Nhóm thứ hai là thủ tục nội tại hệ thống mà các chức năng quản trị cơ sở dữ liệu thường dùng. Các thủ tục thuộc nhóm này chứa trong cơ sở dữ liệu Resource. Đây là cơ sở dữ liệu chỉ đọc chứa các đối tượng cơ sở dữ liệu và được ánh xạ qua các cơ sở dữ liệu là sys. Bạn có thể triệu gọi các thành phần trong cơ sở dữ liệu Resource bằng cách sử dụng sys.objects, sys.tables, ...

Bạn có thể tìm thấy danh sách thủ tục nội tại hệ thống trong ngăn System Stored Procedures như hình 11-1.



**Hình 11-1: Danh sách thủ tục nội tại hệ thống.**

Chú ý: Bạn có thể tìm thấy tập tin MSSQLSystemResource.mdf cơ sở dữ liệu Resource lưu mặc định trong thư mục x:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\Data\.

Thủ tục nội tại trong cơ sở dữ liệu Resource luôn có tên với tiền tố là sp\_. Do đó bạn không nên đặt tên thủ tục nội do mình tạo ra với tiền tố sp\_.

Thay vào đó, bạn nên đặt tên với cú pháp tương tự như sau: udfspInsert, SPInsert, SPINSCustomers...

Nhóm thứ ba là thủ tục nội tại hệ thống mở rộng. Loại này cũng được lưu trong cơ sở dữ liệu Resource nhưng có tên bắt đầu với `xp_`. Chúng ta sẽ tìm hiểu các thủ tục nội tại này trong tập *SQL Server 2005: Lập trình nâng cao*.

## 1.2. Cú pháp tạo thủ tục nội tại

Để khai báo thủ tục nội tại, bạn có thể sử dụng cú pháp như sau:

```
CREATE { PROC | PROCEDURE } [schema_name.] procedure_name
[ ; number ]
[ { @parameter [ type_schema_name. ] data_type }
  [ VARYING ] [ = default ] [ [ OUT [ PUT ]
  ] [ ,...n ]
[ WITH <procedure_option> [ ,...n ]
[ FOR REPLICATION ]
AS { <sql_statement> [ ; ] [ ...n ] | <methodSpecifier> }
[ ; ]
<procedure_option> ::==
[ ENCRYPTION ]
[ RECOMPILE ]
[ EXECUTE_AS_Clause ]
<sql_statement> ::=
( [ BEGIN ] statements [ END ] )
<methodSpecifier> ::=
EXTERNAL NAME assembly_name.class_name.method_name
```

Trong đó, `procedure_name` là tên thủ tục nội tại. Bạn không nên đặt tên thủ tục nội tại với tiền tố là `sp_`, do SQL Server 2005 sẽ tìm kiếm thủ tục nội tại có tiền tố này trong danh sách thủ tục hệ thống trước khi tìm kiếm trong danh sách thủ tục nội tại do người sử dụng tạo ra.

Bạn có thể thay đổi cấu trúc thủ tục nội tại bằng cách sử dụng phát biểu `ALTER PROC` như sau:

```
ALTER { PROC | PROCEDURE } [schema_name.] procedure_name
[ ; number ]
[ { @parameter [ type_schema_name. ] data_type }
  [ VARYING ] [ = default ] [ [ OUT [ PUT ]
  ] [ ,...n ]
[ WITH <procedure_option> [ ,...n ] ]
[ FOR REPLICATION ]
AS
{ <sql_statement> [ ...n ] | <methodSpecifier> }
<procedure_option> ::==
[ ENCRYPTION ]
[ RECOMPILE ]
```

**Chương 11: Khám phá thủ tục nội tại**

145

```
[ EXECUTE_AS_Clause ]
<sql_statement> ::= 
{ [ BEGIN ] statements [ END ] }
<methodSpecifier> ::= 
EXTERNAL NAME
assembly_name.class_name.method_name
```

Tương tự như vậy, bạn có thể xóa thủ tục nội tại bằng cách sử dụng phát biểu DROP PROC như sau:

```
DROP { PROC | PROCEDURE } { [ schema_name. ] procedure }
[ , . . . n ]
```

Chú ý: Nếu không cần truyền tham số từ bên ngoài vào và các khai báo biến như đã trình bày trong chương trước, bạn có thể khai báo và sử dụng đối tượng VIEW thay vì thủ tục nội tại.

### 1.3. Thủ tục đơn giản

Phát biểu CREATE PROC hay CREATE PROCEDURE cho phép bạn tạo thủ tục nội tại trong cơ sở dữ liệu với phát biểu điều khiển *if*, *while*, *with*, *case* và phát biểu SQL dạng truy vấn hay hành động khác.

Trong khi đối tượng View cho phép kết hợp, khai báo biến thức, trích lọc, sắp xếp dữ liệu và không thể truyền giá trị từ bên ngoài để tạo ra tập dữ liệu đúng với yêu cầu của người sử dụng thì thủ tục nội tại cho phép bạn truyền giá trị thông qua tham số.

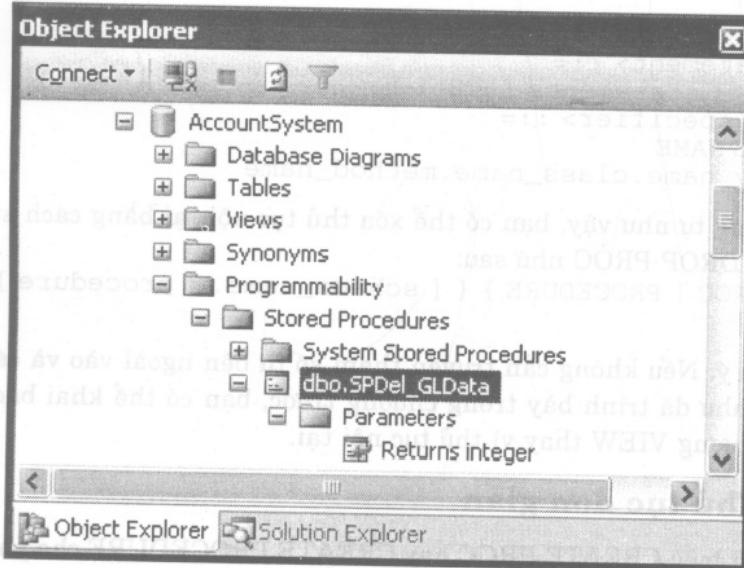
Tương tự như đối tượng View, thủ tục nội tại cũng đã được biên dịch và thường trú trong bộ nhớ. Bạn nên sử dụng thủ tục để thực hiện các thao tác trên cơ sở dữ liệu thay vì sử dụng phát biểu SQL trực tiếp.

Chẳng hạn, bạn khai báo thủ tục nội tại có tên SPDel\_GLData có cấu trúc như ví dụ 11-1.

**Ví dụ 11-1: Khai báo thủ tục nội tại SPDel\_GLData**

```
CREATE PROC SPDel_GLData
AS
    Delete from Receipts
    Delete from Payments
```

Bạn có thể tìm thấy tên thủ tục nội tại SPDel\_GLData xuất hiện trong ngăn Stored Procedures sau khi tạo thành công như hình 11-2.



**Hình 11-2: Thủ tục đơn giản.**

Chú ý: Bạn có thể thay đổi thủ tục nội tại trên bằng cách khai báo tương tự như ví dụ 11-1-1.

**Ví dụ 11-1-1: Khai báo thay đổi cấu trúc thủ tục nội tại**

```
ALTER PROC SPDel_GLData
AS
    Delete from Receipts
    Delete from Payments
    Delete from Balances
GO
```

Ngoài ra, bạn có thể xóa thủ tục SPDel\_GLData khỏi cơ sở dữ liệu bằng cách khai báo như ví dụ 11-1-2.

**Ví dụ 11-1-2: Khai báo xóa thủ tục nội tại**

```
DROP PROC SPDel_GLData
GO
```

#### 1.4. Thủ tục nội tại cùng tên (nhóm thủ tục)

Tham số number là số nguyên chỉ định những thủ tục nội tại cùng tên thành một nhóm, khi bạn sử dụng phát biểu DROP PROC để loại bỏ tên của thủ tục nội tại thì mọi thủ tục nội tại cùng tên khác nhau về number sẽ bị xóa.

**Chương 11: Khám phá thủ tục nội tại**147 

Để tìm hiểu về tham số này, trước tiên bạn khai báo thủ tục nội tại có tên SPDel\_ARData có tham số number là 1, dùng để xóa dữ liệu trong bảng SalesInvoiceBatchs, SalesInvoices, SalesInvoiceDetails như ví dụ 11-2.

**Ví dụ 11-2: Khai báo thủ tục có tham số number**

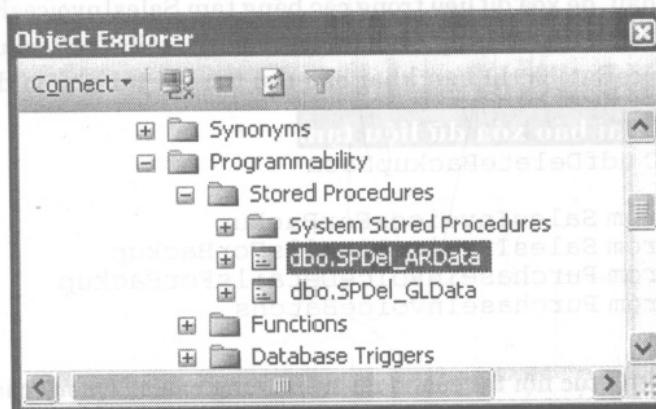
```
CREATE PROC SPDel_ARData;1
AS
    Delete from SalesOrderBatchs
    Delete from SalesOrders
    Delete from SalesOrderDetails
GO
```

Kế đến, bạn khai báo thủ tục nội tại thứ hai cùng tên nhưng có tham số number là 2 như ví dụ 11-3.

**Ví dụ 11-3: Khai báo thủ tục có tham số number**

```
CREATE PROC SPDel_ARData;2
AS
    Delete from MonthlyAccountReceivable
    Delete from CloseAccountReceivable
GO
```

Thực thi hai phát biểu tạo thủ tục trên, bạn có thể tìm thấy hai thủ tục này xuất hiện dưới một tên trong cơ sở dữ liệu như hình 11-3.



**Hình 11-3: Hai thủ tục xuất hiện cùng tên.**

Chú ý: Nếu bạn kích hoạt thủ tục nội tại SPDel\_ARData để thay đổi, cấu trúc của thủ tục này sẽ xuất hiện như ví dụ 11-4.

**Ví dụ 11-4: Cấu trúc thủ tục nội tại SPDel\_ARData**

```
set ANSI_NULLS ON
set QUOTED_IDENTIFIER ON
go
```

**M® 148****Chương 11: Khám phá thủ tục nội tại**

```

ALTER PROC [dbo].[SPDel_ARData]
AS
    Delete from SalesOrderBatchs
    Delete from SalesOrders
    Delete from SalesOrderDetails

ALTER PROC [dbo].[SPDel_ARData];2
AS
    Delete from MonthlyAccountReceivable
    Delete from CloseAccountReceivable

```

**Chú ý:** Nếu bạn sử dụng phát biểu DROP PROC như ví dụ 11-5 thì hai thủ tục nội tại SPDel\_ARData sẽ bị xóa.

**Ví dụ 11-5: Khai báo xóa thủ tục SPDel\_ARData**

```

DROP PROC SPDel_ARData
GO

```

**1.5. Thực thi thủ tục**

Bạn có thể sử dụng phát biểu Transact-SQL là EXECUTE hay EXEC để gọi thủ tục nội tại. Tuy nhiên, bạn không cần chỉ định một trong hai phát biểu EXEC hay EXECUTE nếu sử dụng cửa sổ Query của MS.

Chẳng hạn, để xóa dữ liệu trong các bảng tạm SalesInvoicesForBackup, SalesInvoiceDetailsForBackup, PurchaseInvoiceDetailsForBackup, PurchaseInvoiceBatchs thì bạn khai báo thủ tục nội tại như ví dụ 11-6.

**Ví dụ 11-6: Khai báo xóa dữ liệu tạm**

```

CREATE PROC udfDeleteBackupData
AS
    Delete from SalesInvoicesForBackup
    Delete from SalesInvoiceDetailsForBackup
    Delete from PurchaseInvoiceDetailsForBackup
    Delete from PurchaseInvoiceBatchs
GO

```

Nếu gọi thủ tục nội tại của ví dụ trên trong cửa sổ Query của MS, bạn có thể khai báo như ví dụ 11-7.

**Ví dụ 11-7: Khai báo gọi thủ tục nội tại trong MS**

```

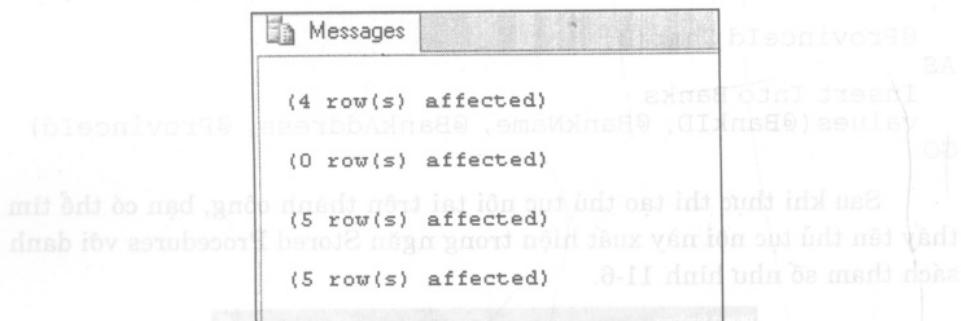
udfDeleteBackupData
GO

```

Khi thực thi thủ tục trên, bạn có thể tìm thấy kết quả trình bày như hình 11-4.

**Chương 11: Khám phá thủ tục nội tại**

149

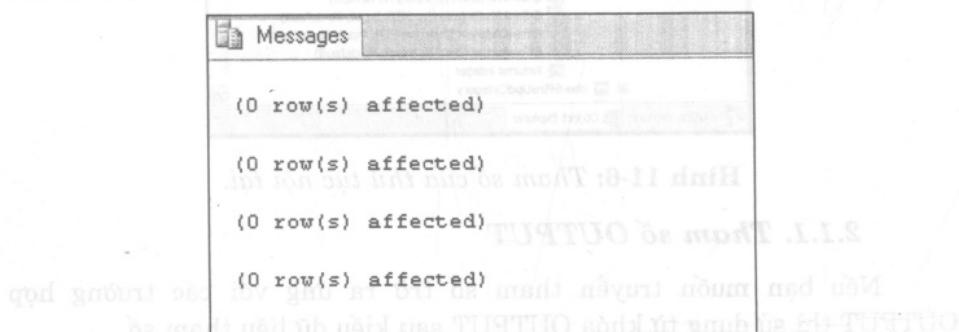
**Hình 11-4:** Thực thi thủ tục nội tại trong MS.

Chú ý: Bạn cũng có thể sử dụng phát biểu EXECUTE hay EXEC bằng cách khai báo như ví dụ 11-8.

**Ví dụ 11-8: Khai báo gọi thủ tục nội tại trong MS**

```
EXEC udfDeleteBackupData
GO
```

Khi thực thi thủ tục trên, bạn có thể tìm thấy kết quả trình bày như hình 11-5.

**Hình 11-5:** Thực thi thủ tục nội tại trong MS.

## 2. THỦ TỤC NỘI TẠI VỚI THAM SỐ

Khi muốn truyền giá trị bên ngoài vào cho thủ tục nội tại, bạn có thể sử dụng tham số với cấu trúc là @parameter. Bạn có thể khai báo danh sách tham số cùng với kiểu dữ liệu tương ứng như ví dụ 11-9.

**Ví dụ 11-9: Khai báo thủ tục có tham số**

```
CREATE PROC SPInsBanks
    @BankID char(3),
    @BankName nvarchar(50),
    @BankAddress nvarchar(50),
```

M® 150

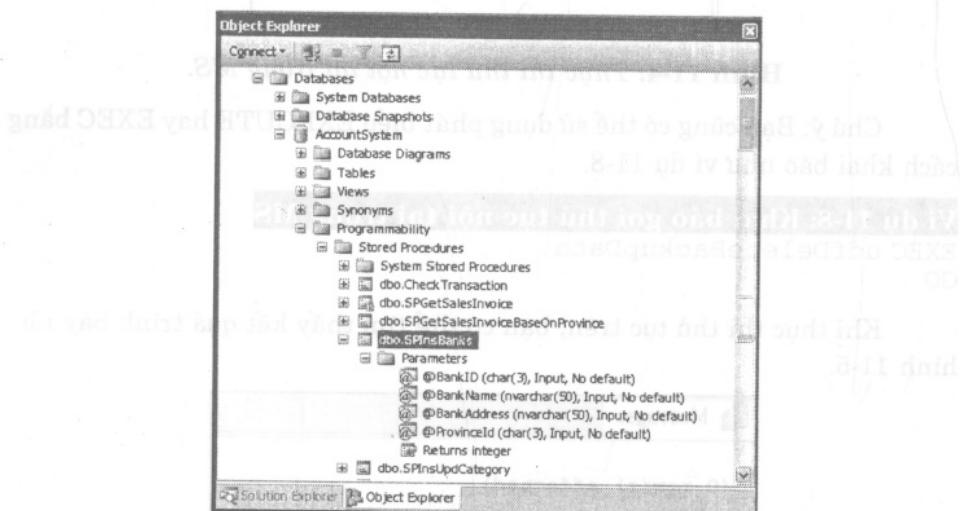
**Chương 11: Khám phá thủ tục nội tại**

```

@ProvinceId Char (3)
AS
Insert Into Banks
values(@BankID, @BankName, @BankAddress, @ProvinceId)
GO

```

Sau khi thực thi tạo thủ tục nội tại trên thành công, bạn có thể tìm thấy tên thủ tục nội này xuất hiện trong ngăn Stored Procedures với danh sách tham số như hình 11-6.



**Hình 11-6: Tham số của thủ tục nội tại.**

### 2.1.1. Tham số OUTPUT

Nếu bạn muốn truyền tham số trả ra ứng với các trường hợp OUTPUT thì sử dụng từ khóa OUTPUT sau kiểu dữ liệu tham số.

Giả sử, bạn muốn lấy ra số lượng của một sản phẩm có trong kho từ bảng Product thì khai báo như ví dụ 11-10.

#### Ví dụ 11-10: Khai báo tham số output

```

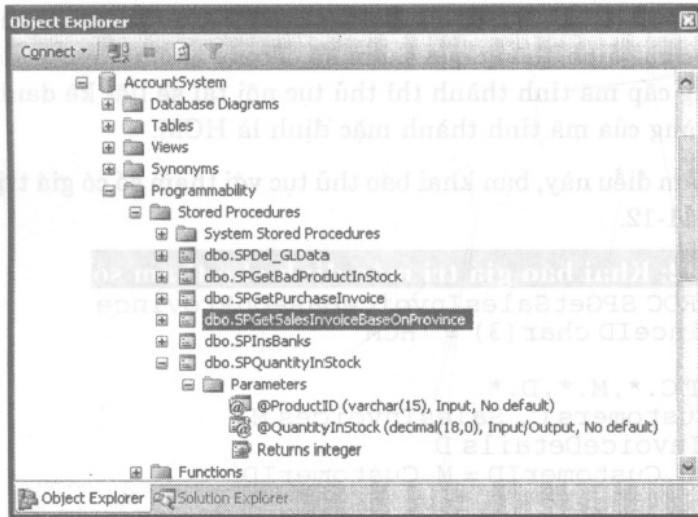
CREATE PROC SPQuantityInStock
    @ProductID varchar(15),
    @QuantityInStock Decimal OUTPUT
AS
    select @QuantityInStock=@QuantityInStock+
        GoodProductInStock from ProductInStocks
        where ProductID = @ProductID
GO

```

Sau khi tạo thành công thủ tục nội tại có tham số là OUTPUT của ví dụ trên, bạn có thể tìm thấy thủ tục nội tại này như hình 11-7.

**Chương 11: Khám phá thủ tục nội tại**

151

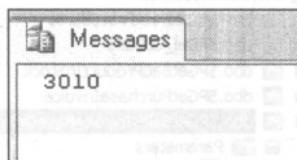
**Hình 11-7: Tham số OUTPUT.**

Khi gọi thủ tục nội tại có tham số là OUTPUT, bạn phải chỉ định tham số với từ khóa này tương tự như ví dụ 11-11.

**Ví dụ 11-11: Khai báo sử dụng tham số OUTPUT**

```
declare @a decimal
set @a=10
exec SPQuantityInStock 'P00001', @a output
print @a
```

Nếu bạn thực thi phát biểu trong ví dụ trên, giá trị của biến @a sẽ trình bày như hình 11-8.

**Hình 11-8: Gọi thủ tục nội tại với tham số OUTPUT.****2.1.2. Giá trị mặc định cho tham số**

Bạn có thể khai báo giá trị mặc định cho tham số của thủ tục bằng cách khai báo giá trị ứng với kiểu dữ liệu của tham số với cú pháp tham khảo như sau:

```
CREATE PROCEDURE_NAME
    @ParameterName dataType [= value] [,]
AS
    SQL_Statements
GO
```

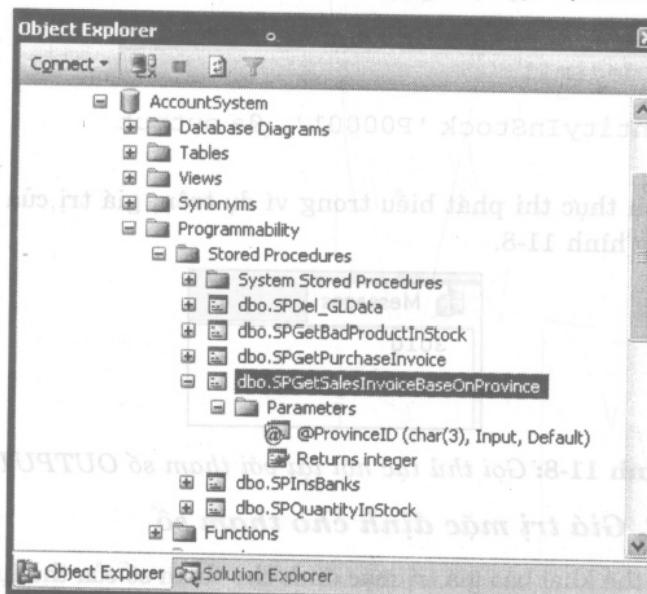
Chẳng hạn, bạn muốn liệt kê danh sách hóa đơn bán hàng của tỉnh thành được chỉ định thông qua tham số. Trong trường hợp người sử dụng không cung cấp mã tỉnh thành thì thủ tục nội tại sẽ liệt kê danh sách hóa đơn bán hàng của mã tỉnh thành mặc định là HCM.

Để làm điều này, bạn khai báo thủ tục với tham số có giá trị mặc định như ví dụ 11-12.

#### Ví dụ 11-12: Khai báo giá trị mặc định cho tham số

```
CREATE PROC SPGetSalesInvoiceBaseOnProvince
    @ProvinceID char(3) = 'HCM'
AS
    SELECT C.* , M.* , D.*
        FROM Customers C, SalesInvoices M,
        SalesInvoiceDetails D
    Where C.CustomerID = M.CustomerID
    and M.InvoiceNo = D.InvoiceNo
    And ProvinceID = @ProvinceID
GO
```

Tương tự như trên, sau khi tạo thủ tục thành công, bạn có thể tìm thấy tên thủ tục này cùng với tham số có giá trị mặc định như hình 11-9.



**Hình 11-9:** Tham số có giá trị mặc định.

Khi gọi thủ tục nội tại có tên SPGetSalesInvoiceBaseOnProvince, nếu bạn không chỉ định tham số thì thủ tục nội tại sẽ thực thi với giá trị HCM cho tham số @ProvinceID như ví dụ 11-12-1.

**Chương 11: Khám phá thủ tục nội tại**

153

**Ví dụ 11-12-1: Khai báo gọi thủ tục nội tại không giá trị cho tham số  
SPGetSalesInvoiceBaseOnProvince**

Ví dụ, trong trường hợp này kết quả sẽ trình bày như hình 11-9-1.

| CustomerID | CompanyNameInVietnamese                      | CompanyNameInSecondLanguage |
|------------|--|-----------------------------|
| 1 A0001    | Công ty Trách Nhiệm Hữu Hạn Macosoft Vietnam | Macosoft Vietnam Co., Ltd.  |
| 2 A0002    | Công ty Trách Nhiệm Hữu Hạn IBN Vietnam      | IBN Vietnam Co., Ltd.       |
| 3 A0001    | Công ty Trách Nhiệm Hữu Hạn Macosoft Vietnam | Macosoft Vietnam Co., Ltd.  |
| 4 A0005    | Công ty Cổ phần Suzumi Vietnam               | Suzumi Vietnam Corp.        |
| 5 A0007    | Công ty Đầu quốc gia UFCA                    | UFCA Corp.                  |
| 6 A0008    | Công ty Cổ phần Rerul Vietnam                | Rerul Vietnam Corp.         |
| 7 A0001    | Công ty Trách Nhiệm Hữu Hạn Macosoft Vietnam | Macosoft Vietnam Co., Ltd.  |
| 8 A0002    | Công ty Trách Nhiệm Hữu Hạn IBN Vietnam      | IBN Vietnam Co., Ltd.       |
| 9 A0001    | Công ty Trách Nhiệm Hữu Hạn Macosoft Vietnam | Macosoft Vietnam Co., Ltd.  |
| 10 A0005   | Công ty Cổ phần Suzumi Vietnam               | Suzumi Vietnam Corp.        |
| 11 A0001   | Công ty Trách Nhiệm Hữu Hạn Macosoft Vietnam | Macosoft Vietnam Co., Ltd.  |
| 12 A0002   | Công ty Trách Nhiệm Hữu Hạn IBN Vietnam      | IBN Vietnam Co., Ltd.       |
| 13 A0001   | Công ty Trách Nhiệm Hữu Hạn Macosoft Vietnam | Macosoft Vietnam Co., Ltd.  |
| 14 A0007   | Công ty Đầu quốc gia UFCA                    | UFCA Corp.                  |
| 15 A0008   | Công ty Cổ phần Rerul Vietnam                | Rerul Vietnam Corp.         |
| 16 A0001   | Công ty Trách Nhiệm Hữu Hạn Macosoft Vietnam | Macosoft Vietnam Co., Ltd.  |
| 17 A0002   | Công ty Trách Nhiệm Hữu Hạn IBN Vietnam      | IBN Vietnam Co., Ltd.       |
| 18 A0001   | Công ty Trách Nhiệm Hữu Hạn Macosoft Vietnam | Macosoft Vietnam Co., Ltd.  |
| 19 A0005   | Công ty Cổ phần Suzumi Vietnam               | Suzumi Vietnam Corp.        |
| 20 A0008   | Công ty Cổ phần Rerul Vietnam                | Rerul Vietnam Corp.         |
| 21 A0001   | Công ty Trách Nhiệm Hữu Hạn Macosoft Vietnam | Macosoft Vietnam Co., Ltd.  |
| 22 A0005   | Công ty Cổ phần Suzumi Vietnam               | Suzumi Vietnam Corp.        |
| 23 A0001   | Công ty Trách Nhiệm Hữu Hạn Macosoft Vietnam | Macosoft Vietnam Co., Ltd.  |
| 24 A0005   | Công ty Cổ phần Suzumi Vietnam               | Suzumi Vietnam Corp.        |

**Hình 11-9-1: Thực thi thủ tục nội tại.**

Nếu bạn chỉ định giá trị cho tham số thì thủ tục nội tại sẽ thực thi ứng với giá trị truyền vào là HAN như ví dụ 11-11-2.

**Ví dụ 11-12-2: Khai báo gọi thủ tục nội tại có giá trị cho tham số  
SPGetSalesInvoiceBaseOnProvince 'HAN'**

Đối với trường hợp truyền giá trị cho tham số thì kết quả sẽ trình bày như hình 11-9-2.

| CustomerID | CompanyNameInVietnamese                      | CompanyNameInSecondLanguage |
|------------|--|-----------------------------|
| 1 A0003    | Công ty Trách Nhiệm Hữu Hạn Kodaka Vietnam   | Kodaka Vietnam Co., Ltd.    |
| 2 A0004    | Công ty Trách Nhiệm Hữu Hạn E-Google Vietnam | E-Google Vietnam Co., Ltd.  |
| 3 A0003    | Công ty Trách Nhiệm Hữu Hạn Kodaka Vietnam   | Kodaka Vietnam Co., Ltd.    |
| 4 A0004    | Công ty Trách Nhiệm Hữu Hạn E-Google Vietnam | E-Google Vietnam Co., Ltd.  |

**Hình 11-9-2: Tham số có giá trị.**

## 2.2. Gọi thủ tục trong thủ tục

Bạn có thể gọi thủ tục nội tại trong thủ tục nội tại bằng lệnh EXEC hay EXECUTE.

Chẳng hạn, bạn khai báo gọi thủ tục nội tại vừa tạo trong ví dụ trên (SPGetSalesInvoiceBaseOnProvince) như ví dụ 11-13.

### Ví dụ 11-13: Khai báo gọi thủ tục nội tại

```
CREATE PROC SPCallISP
AS
    EXECUTE SPGetSalesInvoiceBaseOnProvince
    EXEC SPGetSalesInvoiceBaseOnProvince 'HAN'
GO
```

Khi thực thi phát biểu CREATE PROC trong ví dụ trên, thủ tục nội tại SPCallISP được tạo ra, bạn có thể gọi thủ tục nội tại này bằng cách khai báo như ví dụ 11-13-1.

### Ví dụ 11-13-1: Khai báo gọi thủ tục nội tại SPCallSP

```
SPCallSP
GO
```

Kết quả sẽ trình bày như hình 11-10.

| CustomerID | CompanyNameInVietnamese                       | CompanyNameInSecondLanguage |
|------------|---|-----------------------------|
| 1 A0001    | Công ty Trách Nhiệm Hữu Hạn Microsoft Vietnam | Microsoft Vietnam Co., Ltd. |
| 2 A0002    | Công ty Trách Nhiệm Hữu Hạn IBM Vietnam       | IBM Vietnam Co., Ltd.       |
| 3 A0001    | Công ty Trách Nhiệm Hữu Hạn Microsoft Vietnam | Microsoft Vietnam Co., Ltd. |
| 4 A0005    | Công ty Cổ phần Suzumi Vietnam                | Suzumi Vietnam Corp.        |
| 5 A0007    | Công ty Đa quốc gia UFCA                      | UFCA Corp.                  |
| 6 A0008    | Công ty Cổ phần ReruitVietnam                 | ReruitVietnam Corp.         |
| 7 A0001    | Công ty Trách Nhiệm Hữu Hạn Microsoft Vietnam | Microsoft Vietnam Co., Ltd. |
| 8 A0002    | Công ty Trách Nhiệm Hữu Hạn IBM Vietnam       | IBM Vietnam Co., Ltd.       |

| CustomerID | CompanyNameInVietnamese                      | CompanyNameInSecondLanguage |
|------------|--|-----------------------------|
| 1 A0003    | Công ty Trách Nhiệm Hữu Hạn Kodaka Vietnam   | Kodaka Vietnam Co., Ltd.    |
| 2 A0004    | Công ty Trách Nhiệm Hữu Hạn E-Google Viet... | E-Google Vietnam Co., Ltd.  |
| 3 A0003    | Công ty Trách Nhiệm Hữu Hạn Kodaka Vietnam   | Kodaka Vietnam Co., Ltd.    |
| 4 A0004    | Công ty Trách Nhiệm Hữu Hạn E-Google Viet... | E-Google Vietnam Co., Ltd.  |

Hình 11-10: Gọi thủ tục nội tại.

Nếu thủ tục bị gọi chưa tồn tại trong cơ sở dữ liệu, bạn có thể tạo được thủ tục nội tại dùng để gọi nhưng SQL Server sẽ gửi cảnh báo thủ tục nội tại bị gọi chưa tồn tại.

**Chương 11: Khám phá thủ tục nội tại**

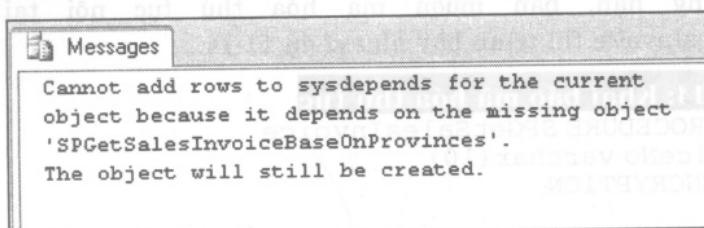
155 M®

Chẳng hạn, bạn khai báo thủ tục nội tại để gọi thủ tục nội tại chưa tồn tại như ví dụ 11-13-2.

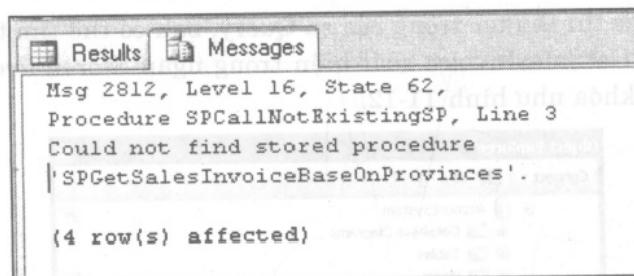
**Ví dụ 11-13-2: Khai báo gọi thủ tục nội tại chưa tồn tại**

```
CREATE PROC SPCallNotExistSP
AS
    EXECUTE SPGetSalesInvoiceBaseOnProvinces
    EXEC SPGetSalesInvoiceBaseOnProvince 'HAN'
GO
```

Khi thực thi phát biểu CREATE PROC trong ví dụ trên, bạn có thể nhận được cảnh báo như hình 11-11.

**Hình 11-11: Cảnh báo.**

Nếu bạn gọi thủ tục SPCallNotExistingSP, lập tức lỗi sẽ phát sinh tương tự như hình 11-11-1.

**Hình 11-11-1: Lỗi phát sinh khi gọi SPCallNotExistingSP.**

Tuy nhiên, thủ tục nội tại thứ hai vẫn được gọi và kết quả trả về như hình 11-11-2.

| CustomerID | CompanyNameInVietnamese                      | CompanyNameInSecondLanguage |
|------------|--|-----------------------------|
| 1 A0003    | Công ty Trách Nhiệm Hữu Hạn Kodaka Vietnam   | Kodaka Vietnam Co., Ltd.    |
| 2 A0004    | Công ty Trách Nhiệm Hữu Hạn E-Google Vietnam | E-Google Vietnam Co., Ltd.  |
| 3 A0003    | Công ty Trách Nhiệm Hữu Hạn Kodaka Vietnam   | Kodaka Vietnam Co., Ltd.    |
| 4 A0004    | Công ty Trách Nhiệm Hữu Hạn E-Google Vietnam | E-Google Vietnam Co., Ltd.  |

**Hình 11-11-2: Kết quả của thủ tục nội tại thứ hai.**

Chú ý: Bạn có thể tìm hiểu thêm cách gọi thủ tục nội tại trong phần trình bày thủ tục nội tại có giá trị trả về.

### 2.3. Mã hóa thủ tục nội tại

Tương tự như đối tượng View, thủ tục nội tại cũng có thể khai báo với từ khóa WITH với ba tùy chọn như sau:

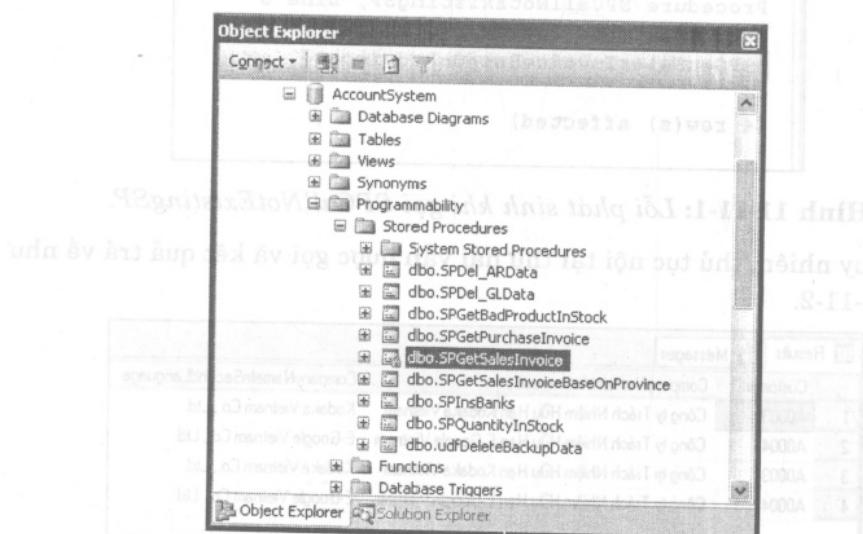
```
<procedure_option> ::=  
    [ ENCRYPTION ]  
    [ RECOMPILE ]  
    [ EXECUTE_AS_Clause ]
```

Chẳng hạn, bạn muốn mã hóa thủ tục nội tại có tên SPGetSalesInvoice thì trình bày như ví dụ 11-14.

#### Ví dụ 11-14: Khai báo mã hóa thủ tục

```
CREATE PROCEDURE SPGetSalesInvoice  
    @InvoiceNo varchar(10)  
    WITH ENCRYPTION  
AS  
    SELECT M.* , D.*  
    FROM SalesInvoices M, SalesInvoiceDetails D  
    Where M.InvoiceNo = D.InvoiceNo  
    And M.InvoiceNo = @InvoiceNo  
    GO
```

Khi thực thi thủ tục trong cửa sổ Query, bạn có thể tìm thấy tên thủ tục có tên SPGetSalesInvoice xuất hiện trong ngăn Stored Procedures với biểu tượng ổ khóa như hình 11-12.



Hình 11-12: Mã hóa thủ tục nội tại



```
sp_helptext SPGetSalesInvoice
GO
Messages
The text for object 'SPGetSalesInvoice' is encrypted.
```

Hình 11-13: Thủ tục nội tại đã bị mã hóa.

### 3. MUỢN QUYỀN THỰC THI THỦ TỤC

Bạn có thể muộn quyền thực thi của người sử dụng khác để có thể triệu gọi thủ tục nội tại bằng cách chỉ định tài khoản của người sử dụng bằng từ khóa WITH với từ khóa EXECUTE AS.

Khi sử dụng EXECUTE AS, bạn có thể chỉ định một trong 4 định danh CALLER | SELF | OWNER | 'user\_name'.

Trong đó, CALLER là người sử dụng đang gọi đến nó, OWNER chỉ cho phép chủ nhân triệu gọi thủ tục, SELF và user\_name là chỉ định tài khoản khai báo sử dụng trong SQL Server 2005 (bao gồm tài khoản của hệ điều hành).

Chẳng hạn, bạn tạo thủ tục nội tại và chỉ cho phép thủ tục đó chạy theo đặc quyền của tài khoản tạo ra nó thì khai báo như ví dụ 11-15.

#### Ví dụ 11-15: Khai báo mệnh đề EXECUTE AS

```
CREATE PROCEDURE SPGetPurchaseInvoice
    @InvoiceNo varchar(10)
    WITH EXECUTE AS OWNER
AS
    SELECT M.* , D.*
    FROM PurchaseInvoices M,
        PurchaseInvoiceDetails D
```

**M<sup>®</sup> 158****Chương 11: Khám phá thủ tục nội tại**

```
Where M.InvoiceNo = D.InvoiceNo
And M.InvoiceNo = @InvoiceNo
GO
```

Trong trường hợp, bạn tạo thủ tục nội tại và chỉ cho phép thủ tục đó chạy theo đặc quyền của tài khoản nào đó thì khai báo chỉ định tài khoản người sử dụng như ví dụ 11-16.

**Ví dụ 11-16: Khai báo mệnh đề EXECUTE AS**

```
CREATE PROCEDURE SPGetPurchaseInvoiceDetails
@InvoiceNo varchar(10)
WITH EXECUTE AS 'khangdbuser'
AS
SELECT *
FROM PurchaseInvoiceDetails
Where InvoiceNo = @InvoiceNo
GO
```

**4. THỦ TỤC VỚI GIÁ TRỊ TRẢ VỀ**

Thủ tục nội tại cho phép bạn khai báo trả về giá trị là số nguyên bằng mệnh đề Return.

Chẳng hạn, bạn khai báo thủ tục nội tại với giá trị trả về là số nguyên có tên SPGetQuantityInStock như ví dụ 11-17.

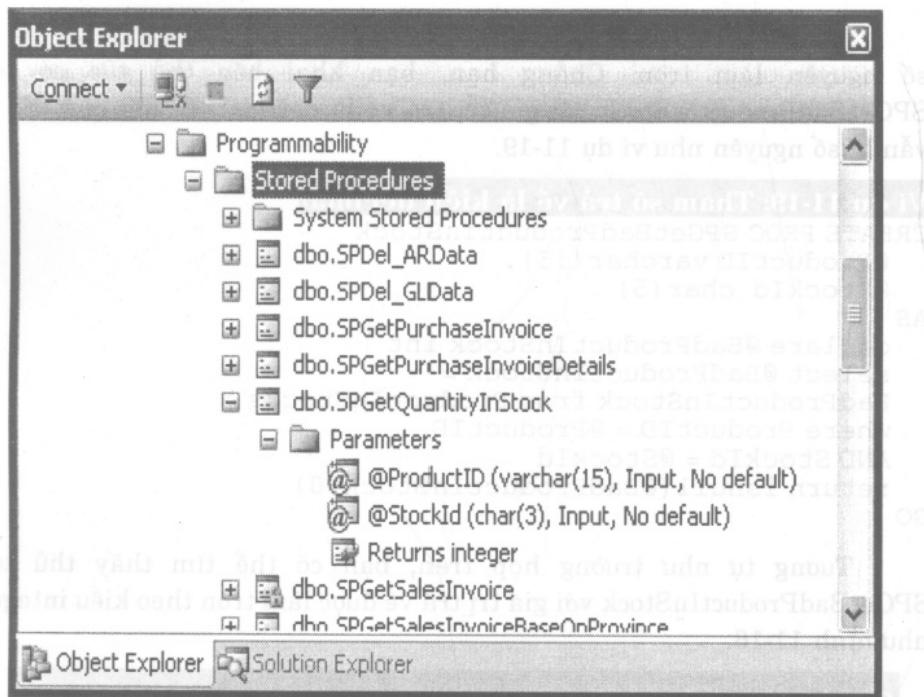
**Ví dụ 11-17: Khai báo mệnh đề Return**

```
CREATE PROC SPGetQuantityInStock
@ProductID varchar(15),
@StockId char(5)
AS
declare @QuantityInStock int
select @QuantityInStock =
GoodProductInStock from ProductInStocks
where ProductID = @ProductID
AND StockId = @StockId
return isnull(@QuantityInStock, 0)
GO
```

Bạn có thể tìm thấy thủ tục SPGetQuantityInStock với giá trị trả về có kiểu integer như hình 11-14.

**Chương 11: Khám phá thủ tục nội tại**

159

**Hình 11-14:** Thủ tục nội tại có giá trị trả về.

Giả sử, bạn khai báo gọi thủ tục nội tại SPGetQuantityInStock như ví dụ 11-18.

**Ví dụ 11-18: Khai báo gọi thủ tục SPGetQuantityInStock**

```
declare @QuantityInStock float
exec @QuantityInStock = SPGetQuantityInStock 'P00001',
'ST001'
select 'P00001' As ProductId, 'ST001' as StockId,
@QuantityInStock As QuantityInStock
Go
```

Khi gọi các phát biểu trong ví dụ trên, kết quả sẽ trình bày như hình 11-15.

|   | ProductId | Stoc... | QuantityInStock |
|---|-----------|---------|-----------------|
| 1 | P00001    | ST001   | 1000            |

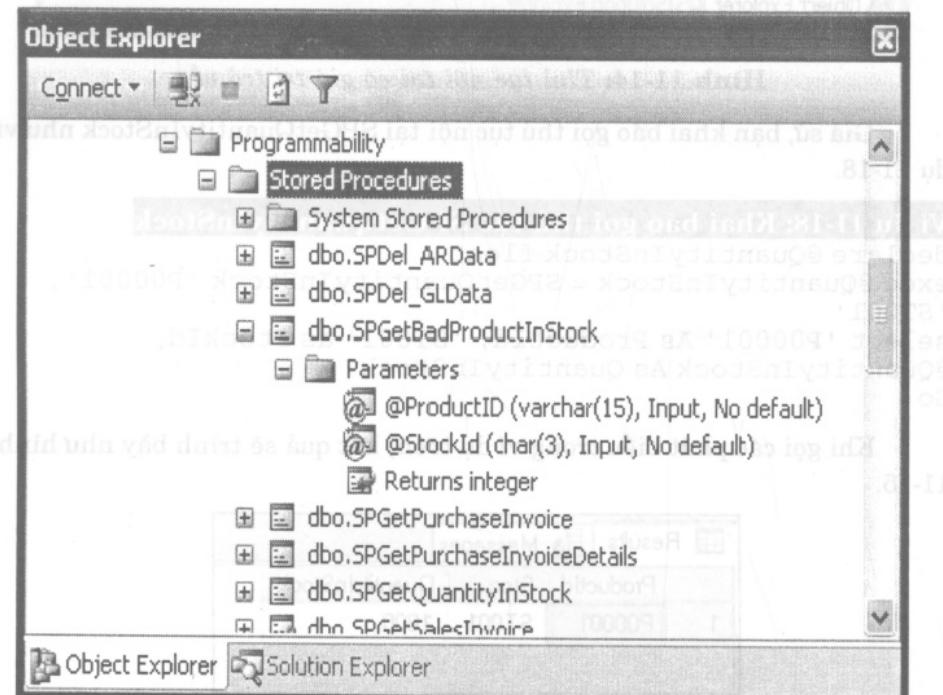
**Hình 11-15:** Thủ tục có giá trị trả về.

Nếu giá trị trả về là số double, float, decimal thì kết quả trả về vẫn là số nguyên làm tròn. Chẳng hạn, bạn khai báo thủ tục có tên SPGetBadProductInStock với giá trị trả về là decimal thì kết quả trả về vẫn là số nguyên như ví dụ 11-19.

#### Ví dụ 11-19: Tham số trả về là kiểu decimal

```
CREATE PROC SPGetBadProductInStock
    @ProductID varchar(15),
    @StockId char(5)
AS
    declare @BadProductInStock int
    select @BadProductInStock =
        BadProductInStock from ProductInStocks
    where ProductID = @ProductID
    AND StockId = @StockId
    return isnull(@BadProductInStock, 0)
GO
```

Tương tự như trường hợp trên, bạn có thể tìm thấy thủ tục SPGetBadProductInStock với giá trị trả về được làm tròn theo kiểu integer như hình 11-16.



Hình 11-16: Thủ tục nội tại có giá trị trả về.

**Chương 11: Khám phá thủ tục nội tại**

161 M®

Để lấy giá trị trả về của thủ tục, bạn sử dụng lệnh EXEC ưng với biến nấm giữ giá trị lấy được. Chẳng hạn, bạn gọi thủ tục SPGetBadProductInStock như ví dụ 11-20.

**Ví dụ 11-20: Khai báo gọi thủ tục có giá trị trả về**

```
declare @BadQuantityInStock int
exec @BadQuantityInStock = SPGetBadProductInStock
    'P00001', 'ST001'
select 'P00001' As ProductId, 'ST001' as StockId,
@BadQuantityInStock As QuantityInStock
GO
```

Nếu thực thi phát biểu trong ví dụ trên, bạn có thể tìm thấy kết quả trình bày như hình 11-17.

| ProductId | Stoc... | QuantityInStock |   |
|-----------|---------|-----------------|---|
| 1         | P00001  | ST001           | 0 |

**Hình 11-17:** Thực thi thủ tục nội tại có tham số.

Tuy nhiên, bạn có thể sử dụng lệnh EXECUTE ưng với biến nấm giữ giá trị lấy được bằng cách gọi thủ tục SPGetBadProductInStock như ví dụ 11-21.

**Ví dụ 11-21: Khai báo gọi thủ tục có giá trị trả về**

```
declare @BadQuantityInStock int
execute @BadQuantityInStock =
SPGetBadProductInStock 'P00002', 'ST001'
select 'P00002' As ProductId,
'ST001' as StockId,
@BadQuantityInStock As QuantityInStock
GO
```

Nếu thực thi phát biểu trong ví dụ trên, bạn có thể tìm thấy kết quả trình bày như hình 11-18.

| ProductId | Stoc... | QuantityInStock |   |
|-----------|---------|-----------------|---|
| 1         | P00002  | ST001           | 0 |

**Hình 11-18:** Thực thi thủ tục nội tại có tham số.

## 5. THỦ TỤC NỘI TẠI ĐỂ THÊM, XÓA, CẬP NHẬT DỮ LIỆU

Để thêm dữ liệu vào bảng, bạn có thể sử dụng phát biểu SQL dạng INSERT với cú pháp đã giới thiệu trong chương 7 của tập “SQL Server 2005 - Lập trình T-SQL”.

Khi làm việc với ngôn ngữ lập trình .NET, bạn có thể sử dụng đối tượng ADO.NET để có thể thực thi phát biểu SQL như đối tượng SqlCommand, SqlDataAdapter.

Chẳng hạn, bạn có thể khai báo và khởi tạo đối tượng SqlCommand để thực thi phát biểu SQL dạng INSERT của SQL Server 2005 như ví dụ 11-22.

### Ví dụ 11-22: Khai báo thực thi phát biểu SQL bằng ADO.NET

```
public class SqlServerTransaction
{
    public int ExecuteNonQuery(string commandText,
    CommandType commandType)
    {
        int effectedRows = 0;
        try
        {
            sqlCommand = new SqlCommand();
            sqlCommand.CommandText = commandText;
            sqlCommand.Connection = sqlConnection;
            sqlCommand.CommandType = commandType;
            if (sqlConnection.State != ConnectionState.Open)
                sqlConnection.Open();
            effectedRows = sqlCommand.ExecuteNonQuery();
            sqlCommand.Dispose();
        }
        catch (Exception ex)
        {
            throw new Exception(ex.Message);
        }
        return effectedRows;
    }
}
```

Trong đó, commandText là tham số nắm giữ phát biểu SQL dạng hành động, commandType là kiểu câu truy vấn dạng phát biểu SQL hay thủ tục nội tại và biến sqlConnection là đối tượng kết nối cơ sở dữ liệu.

**Chương 11: Khám phá thủ tục nội tại**

163

Khi bạn gọi phương thức `ExecuteNonQuery`, bạn cần truyền phát biểu SQL và loại câu truy vấn là  `CommandType.Text`. Giả sử, bạn khai báo để thêm mẫu tin vào bảng Countries như ví dụ 11-22-1.

**Ví dụ 11-22-1: Khai báo thực thi phát biểu INSERT**

```
void Execute()
{
    SqlServerTransaction sqlST = new
        SqlServerTransaction(connectionString);
    string commandText =
        "INSERT INTO Countries(CountryId, CountryName)
        VALUES('BAR', 'Barazil')";
    int i = sqlST.ExecuteNonQuery(commandText,
        CommandType.Text);
}
```

Biến `i` sẽ trả về giá trị là 1 nếu phát biểu `INSERT` thực thi thành công. Trong ví dụ trên, chúng ta gán giá trị cụ thể trong phần khai báo `VALUES`.

Nếu bạn lấy giá trị từ giao diện người dùng, bạn có thể khai báo lại ví dụ trên với hai điều khiển `TextBox` (`txtId` và `txtName`) như ví dụ 11-22-2.

**Ví dụ 11-22-2: Khai báo lấy giá trị từ điều khiển TextBox**

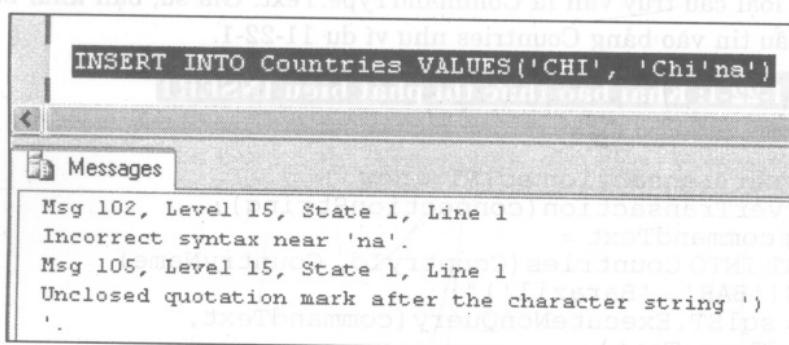
```
void Execute()
{
    SqlServerTransaction sqlST = new
        SqlServerTransaction(connectionString);
    string commandText =
        "INSERT INTO Countries(CountryId, CountryName) " +
        "VALUES('" + txtId.Text + "', '" + txtName.Text + "')";
    int i = sqlST.ExecuteNonQuery(commandText,
        CommandType.Text);
}
```

Đối với ví dụ trên, khi truyền phát biểu SQL đã có giá trị cụ thể từ hai điều khiển `TextBox` (`txtId` là CHI và `txtName` là China), bạn sẽ nhận được phát biểu SQL ngay trước khi thực thi, tương tự như sau:  
`INSERT INTO Countries(CountryId, CountryName)`  
`VALUES('CHI', 'China')`

Phát biểu SQL dạng `INSERT` trên là phát biểu có cú pháp hợp lệ, trong trường hợp người sử dụng nhập mã hay tên của quốc gia có dấu nháy đơn, lỗi sẽ phát sinh do cú pháp sai, tương tự như sau:  
`Msg 105, Level 15, State 1, Line 1`  
`Unclosed quotation mark after the character string '`

Ví dụ, người sử dụng nhập chuỗi Chi'na, như vậy phát biểu SQL sẽ là.  
`INSERT INTO Countries(CountryId, CountryName)`  
`VALUES('CHI', 'Chi'na')`

Khi thực thi phát biểu này, bạn sẽ nhận được lỗi phát sinh do SQL tạo ra như hình 11-19.



**Hình 11-19: Lỗi phát sinh từ SQL Server.**

Giả sử chúng ta chấp nhận giá trị của người sử dụng nhập là hợp lệ, để tránh lỗi này bạn cần thay thế một dấu nháy đơn thành hai dấu nháy đơn liên tiếp.

```
INSERT INTO Countries(CountryId, CountryName)
VALUES ('CHI', 'Chi'na')
```

Như vậy, để có được cú pháp hợp lệ như trên, bạn phải sử dụng phương thức Replace trong ngôn ngữ lập trình C# để thay thế như ví dụ 11-22-3:

#### Ví dụ 11-22-3: Khai báo lấy giá trị từ điều khiển TextBox

```
void Execute()
{
    SqlConnection connection = new SqlConnection(connectionString);
    SqlCommand command = new SqlCommand("INSERT INTO Countries(CountryId, CountryName) " +
                                         "VALUES ('" + txtId.Text.Replace("'", "''") + "','" + txtName.Text.Replace("'", "''") + "')");
    connection.Open();
    command.ExecuteNonQuery();
}
```

Nếu giao diện người sử dụng có N điều khiển dùng cho mục đích nhập liệu, bạn cần sử dụng phương thức Replace nhiều lần, do đó cách này không hiệu quả.

Để chấp nhận dấu nháy đơn mà không cần xử lý như trên, đối tượng ADO.NET giới thiệu đối tượng SqlParameter và thuộc tính Parameters (Collection) của đối tượng SqlCommand để cho phép bạn truyền phát biểu SQL với tham số.

**Chương 11: Khám phá thủ tục nội tại**165 

Nếu bạn sử dụng đối tượng SqlParameter và SqlCommand, phát biểu INSERT trên sẽ viết lại như sau:

```
INSERT INTO Countries(CountryId, CountryName)
VALUES (@Id, @Name)
```

Và bạn sẽ khai báo phương thức ExecuteNonQuery để sử dụng thuộc tính Parameters của SqlCommand như ví dụ 11-23.

**Ví dụ 11-23: Khai báo phương thức ExecuteNonQuery**

```
public class SqlServerTransaction
{
    public int ExecuteNonQuery(string commandText,
    CommandType commandType,
    string[] parameterCollection,
    string[] valueCollection)
    {
        int effectedRows = 0;
        try
        {
            sqlCommand = new SqlCommand();
            sqlCommand.CommandText = commandText;
            sqlCommand.Connection = sqlConnection;
            if (sqlConnection.State != ConnectionState.Open)
                sqlConnection.Open();
            sqlCommand.CommandType = commandType;
            // Khai báo truyền tham số
            for (int i = 0;
                i < parameterCollection.Length; i++)
            {
                sqlCommand.Parameters.AddWithValue(
                    parameterCollection[i],
                    valueCollection[i]);
            }
            effectedRows =
                sqlCommand.ExecuteNonQuery();
            sqlCommand.Dispose();
        }
        catch (Exception ex)
        {
            throw new Exception(ex.Message);
        }
        return effectedRows;
    }
}
```

Khi lấy giá trị từ giao diện người dùng, bạn có thể khai báo lấy giá trị trên với hai điều khiển TextBox và tham số như ví dụ 11-24.

**Ví dụ 11-24: Khai báo lấy giá trị từ điều khiển TextBox**

```
void Execute()
```

**A\* 166****Chương 11: Khám phá thủ tục nội tại**

```
{
SqlServerTransaction sqlST = new
SqlServerTransaction(connectionString);
string commandText =
"INSERT INTO Countries(CountryId, CountryName) " +
" VALUES (@Id, @Name)";
int i = sqlST.ExecuteNonQuery(commandText,
CommandType.Text,
new string[2]{ "@Id", "@Name" },
new string[2]{ txtId.Text, txtName.Text });
}
```

Lưu ý: Để tìm hiểu chi tiết về đối tượng ADO.NET, bạn có thể tìm đọc bộ sách “C# 2005” do nhà sách Minh Khai phát hành.

Sau cùng, ví dụ trên chỉ mới giải quyết được việc xử lý dấu nháy đơn do người sử dụng nhập từ giao diện người dùng, khi truyền phát biểu INSERT trên, SQL Server sẽ kiểm tra cú pháp trước khi chúng được thực thi.

Để không cần kiểm tra cú pháp của phát biểu SQL trước khi thực thi, bạn cần khai báo thủ tục nội tại cho mọi trường hợp thêm, xóa, cập nhật.

### **5.1. Thủ tục nội tại để thêm dữ liệu**

Như đã giới thiệu trong phần trên, để thêm mẫu tin vào bảng dữ liệu, bạn sử dụng phát biểu SQL dạng INSERT trong thủ tục nội tại.

Chẳng hạn, bạn khai báo thủ tục nội tại SPInsUpdCountry để thêm mẫu tin vào bảng Countries tương tự như ví dụ 11-25.

**Ví dụ 11-25: Khai báo thủ tục nội tại thêm mẫu tin**

```
CREATE PROC SPInsUpdCountry
@CountryId CHAR(3),
@CountryName NVARCHAR(30)
AS
INSERT INTO Countries(CountryId, CountryName)
VALUES (@CountryId, @CountryName)
```

Khi thực thi thủ tục SPInsUpdCountry trong cửa sổ Query, bạn khai báo tương tự như ví dụ 11-25-1.

**Ví dụ 11-25-1: Khai báo gọi thủ tục nội tại SPInsUpdCountry**

```
SPInsUpdCountry 'CHI', N'China'
GO
```

Tuy nhiên, trong trường hợp gọi thủ tục SPInsUpdCountry từ ngôn ngữ lập trình C#, bạn khai báo như ví dụ 11-25-2.

**Chương 11: Khám phá thủ tục nội tại**167 **Ví dụ 11-25-2: Khai báo gọi thủ tục nội tại SPInsUpdCountry từ C#**

```

void Execute()
{
    SqlServerTransaction sqlST = new
    SqlServerTransaction(connectionString);

    // Khai báo tên thủ tục
    string commandText = "SPInsUpdCountry";

    // Khai báo chỉ định loại truy vấn là thủ tục

    int i = sqlST.ExecuteNonQuery(commandText,
        CommandType.StoredProcedure,
        new string[2] { "@Id", "@Name" },
        new string[2] { txtId.Text, txtName.Text });
}

```

Nếu có nhu cầu lấy ra giá trị từ thủ tục nội tại thì bạn có thể sử dụng tham số OUTPUT hay phát biểu RETURN.

Đối với một số trường hợp, khi thêm dữ liệu chúng ta cần lấy giá trị nào đó. Ví dụ như sau khi thêm mới mẫu tin, chúng ta lấy ra số tự động tăng vừa phát sinh hay giá trị thay đổi sau khi thực thi thủ tục, bằng cách này bạn có thể sử dụng khai báo thủ tục nội tại với giá trị trả về hay tham số OUTPUT.

### **5.1.1. Thủ tục nội tại để thêm mẫu tin với tham số OUTPUT**

Giả sử, khi thêm mẫu tin vào bảng Customers, cột CustomerId sẽ có chuỗi gồm 5 ký tự tăng dần. Điều này có nghĩa là giá trị tại cột CustomerId của mẫu tin cuối cùng là A0009 thì mẫu tin vừa mới thêm vào sẽ có giá trị tại cột CustomerId là A0010.

Để làm điều này, trước tiên bạn khai báo hàm để lấy ra tổng số mẫu tin đang có trong bảng như ví dụ 11-26.

**Ví dụ 11-26: Khai báo hàm trả về số mẫu tin**

```

CREATE FUNCTION udfTotalRows
(
    @ObjectName nvarchar(50)
)
RETURNS INT
WITH EXECUTE AS CALLER
AS
BEGIN
    DECLARE @NumberOfRow INT

```

```

#O SELECT @NumberOfRow = rows
    FROM sys.partitions
    WHERE object_id =
        (
            SELECT object_id
            FROM sys.tables
            WHERE name = @ObjectName
        )
    RETURN @NumberOfRow
END;
GO

```

Lưu ý: Để lấy ra tổng số mẫu tin đang có trong một bảng dữ liệu, bạn sử dụng phát biểu SELECT tương tự như ví dụ 11-26-1.

#### Ví dụ 11-26-1: Tổng số mẫu tin

```

SELECT rows FROM sys.partitions
WHERE object_id = (SELECT object_id from sys.tables where
name = 'Customers')
GO

```

Khi bạn thực thi phát biểu SELECT trên, bạn có thể tìm thấy cột rows có giá trị là 9 như hình 11-20.

| ROWS |
|------|
| 9    |

Hình 11-20: Số mẫu tin.

Bạn có thể kiểm tra hàm udfTotalRows và khai báo như ví dụ 11-26-2.

#### Ví dụ 11-26-2: Khai báo gọi hàm udfTotalRows

```

SELECT dbo.udfTotalRows( 'Customers' ) AS TotalRows
GO

```

Khi thực thi phát biểu SELECT trong ví dụ trên, bạn có thể tìm thấy kết quả trình bày như hình 11-21.

**Chương 11: Khám phá thủ tục nội tại**

169

| Results   |   | Messages |  |
|-----------|---|----------|--|
| TotalRows |   |          |  |
| 1         | 9 |          |  |

**Hình 11-21: Gọi hàm udfTotalRows.**

Kế đến, bạn khai báo biểu thức bảng để tìm xem giá trị của mẫu tin cuối cùng tại cột CustomerId và giá trị sẽ được thêm vào tương tự như ví dụ 11-26-3.

**Ví dụ 11-26-3: Khai báo tính giá trị của cột CustomerId**

```
DECLARE @Rows INT
SET @Rows = dbo.udfTotalRows('Customers');
WITH LastItem
AS
(
    SELECT CustomerId, ROW_NUMBER()
    OVER(ORDER BY CustomerId) AS RowNumber
    FROM Customers
)
SELECT CustomerId, 'A' + RIGHT('0000' +
LTRIM(CAST(RIGHT(CustomerId, 4) AS INT) + 1), 4)
AS NextValueForCustomerId
FROM LastItem WHERE RowNumber = @Rows
GO
```

Khi thực thi phát biểu SELECT của ví dụ trên, bạn sẽ tìm thấy giá trị tại cột CustomerId của hàng cuối cùng và giá trị kế tiếp như hình 11-22.

| Results |            | Messages |                        |
|---------|------------|----------|------------------------|
|         | CustomerId |          | NextValueForCustomerId |
| 1       | A0009      |          | A0010                  |

**Hình 11-22: Giá trị kế tiếp của cột CustomerId.**

Chú ý: Bạn có thể tìm thấy danh sách 9 mẫu tin đang có trong bảng Customers bằng cách sử dụng phát biểu SELECT như hình 11-23.

M® 170

Chương 11: Khám phá thủ tục nội tại

The screenshot shows a SQL query window with the following content:

```

MYSOLUTION.A...QLQuery4.sql* MYSOLUTION.A...dureForAR.sql
SELECT * FROM Customers
GO
  
```

The results grid displays the following data:

|   | CustomerID | CompanyNameInVietnamese                       |
|---|------------|---|
| 1 | A0001      | Công ty Trách Nhiệm Hữu Hạn Microsoft Vietnam |
| 2 | A0002      | Công ty Trách Nhiệm Hữu Hạn IBM Vietnam       |
| 3 | A0003      | Công ty Trách Nhiệm Hữu Hạn Kodaka Vietnam    |
| 4 | A0004      | Công ty Trách Nhiệm Hữu Hạn E-Google Vietnam  |
| 5 | A0005      | Công ty Cổ phần Suzumi Vietnam                |
| 6 | A0006      | Tập đoàn UCIA USA                             |
| 7 | A0007      | Công ty Đa quốc gia UFCA                      |
| 8 | A0008      | Công ty Cổ phần ReruitVietnam                 |
| 9 | A0009      | Trung tâm giáo dục Vietnam                    |

Hình 11-23: Danh sách mẫu tin trong bảng Customers.

Sau đó, bạn khai báo thủ tục nội tại để thêm mẫu tin vào bảng Customers có giá trị tự động tăng cho cột CustomerId bằng cách sử dụng tham số dạng OUTPUT như ví dụ 11-26-4.

#### Ví dụ 11-26-4: Khai báo tham số OUTPUT với thủ tục nội tại

```

CREATE PROC SPInsUpdCustomer
    @CustomerId CHAR(5) OUTPUT,
    @CompanyNameInVietnamese NVARCHAR(50),
    @CompanyNameInSecondLanguage VARCHAR(50),
    @ContactName NVARCHAR(50),
    @ContactTitle NVARCHAR(50),
    @Address NVARCHAR(100),
    @ProvinceID CHAR(3),
    @Telephone VARCHAR(20),
    @FaxNumber VARCHAR(10),
    @CustomerTypeID CHAR(3),
    @EmailAddress VARCHAR(50),
    @MaxDebt DECIMAL(18,2)
AS
    -- Trường hợp thêm mới
    if (@CustomerId = '') INSERT INTO Customers
        (CompanyNameInVietnamese, CompanyNameInSecondLanguage, ContactName, ContactTitle, Address, ProvinceID, Telephone, FaxNumber, CustomerTypeID, EmailAddress, MaxDebt)
        VALUES (@CompanyNameInVietnamese, @CompanyNameInSecondLanguage, @ContactName, @ContactTitle, @Address, @ProvinceID, @Telephone, @FaxNumber, @CustomerTypeID, @EmailAddress, @MaxDebt)
    -- Trường hợp cập nhật
    else UPDATE Customers
        SET CompanyNameInVietnamese = @CompanyNameInVietnamese, CompanyNameInSecondLanguage = @CompanyNameInSecondLanguage, ContactName = @ContactName, ContactTitle = @ContactTitle, Address = @Address, ProvinceID = @ProvinceID, Telephone = @Telephone, FaxNumber = @FaxNumber, CustomerTypeID = @CustomerTypeID, EmailAddress = @EmailAddress, MaxDebt = @MaxDebt
        WHERE CustomerID = @CustomerId
  
```

**Chương 11: Khám phá thủ tục nội tại**

171

```

BEGIN
    DECLARE @Rows INT
    -- Lấy ra số mẫu tin có trong bảng Customers
    SET @Rows = dbo.udfTotalRows('Customers');
    -- Lấy ra mẫu tin sắp xếp theo số thứ tự
    WITH LastItem
    AS
    (
        SELECT CustomerId,
        ROW_NUMBER()
        OVER(ORDER BY CustomerId) AS RowNumber
        FROM Customers
    )
    -- Lấy ra số mẫu tin cuối cùng trong bảng Customers và tính
    -- mã số kế tiếp
    SELECT @CustomerId= 'A' + RIGHT('0000' +
    LTRIM(CAST(RIGHT(CustomerId, 4) AS INT) + 1), 4)
    FROM LastItem WHERE RowNumber = @Rows
    -- Thêm mẫu tin vào bảng Customers

    INSERT INTO Customers
    (
        CustomerId, CompanyNameInVietnamese,
        CompanyNameInSecondLanguage,
        ContactName, ContactTitle,
        Address, ProvinceID, Telephone,
        FaxNumber, CustomerTypeID,
        EmailAddress, MaxDebt
    )
    VALUES
    (
        @CustomerId, @CompanyNameInVietnamese,
        @CompanyNameInSecondLanguage ,
        @ContactName, @ContactTitle,
        @Address, @ProvinceID, @Telephone,
        @FaxNumber, @CustomerTypeID,
        @EmailAddress, @MaxDebt
    )
END
-- Cập nhật dữ liệu bảng Customers
else
    UPDATE Customers
    SET CompanyNameInVietnamese =
    @CompanyNameInVietnamese,
    CompanyNameInSecondLanguage =
    @CompanyNameInSecondLanguage,
    ContactName = @ContactName,
    ContactTitle = @ContactTitle,

```

**Chương 11: Khám phá thủ tục nội tại**

```

Address = @Address, ProvinceID=@ProvinceID,
Telephone = @Telephone, FaxNumber =@FaxNumber,
CustomerTypeID = @CustomerTypeID,
EmailAddress = @EmailAddress,
MaxDebt = @MaxDebt
WHERE CustomerId = @CustomerId
GO

```

Chú ý: Thủ tục trên sử dụng cho trường hợp thêm mới mẫu tin và cập nhật mẫu tin đang tồn tại dựa vào tham số @CustomerId.

Để sử dụng thủ tục nội tại trên, bạn có thể khai báo biến CustomerID và khởi tạo giá trị là rỗng nếu muốn thêm mới, rồi sau đó sử dụng phát biểu EXEC để thực thi thủ tục nội tại SPInsUpdCustomer như ví dụ 11-26-5.

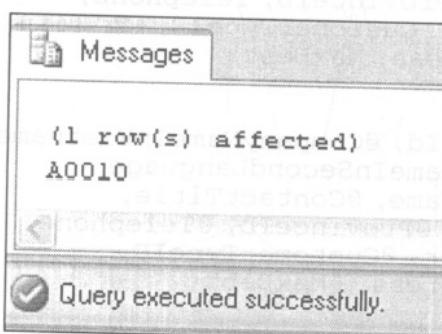
**Ví dụ 11-26-5: Khai báo gọi thủ tục nội tại SPInsUpdCustomer**

```

DECLARE @CustomerId CHAR (5)
SET @CustomerId = ''
EXEC SPInsUpdCustomer @CustomerId OUTPUT,
N'Công ty Trách Nhiệm Hữu Hạn Hot Getways',
'Hot Getways Company', 'Mr Hoo', 'Manager',
'1 No Name', 'HCM', '9090909', '90909090',
'PRI', 'sales@HotGetways.com.vn', '150000000.00'
PRINT @CustomerId
GO

```

Khi thực thi đoạn chương trình trong ví dụ trên, bạn có thể tìm thấy kết quả trình bày như hình 11-24.



**Hình 11-24: Gọi thủ tục nội tại SPInsUpdCustomer.**

Để kiểm tra mẫu tin vừa thêm mới vào bảng Customers có giá trị là A0010 như hình 11-24, bạn có thể thực thi phát biểu SELECT như hình 11-25.

## **Chương 11: Khám phá thủ tục nội tại**

173

|    | CustomerID | CompanyNameInVietnamese                       |
|----|------------|---|
| 1  | A0001      | Công ty Trách Nhiệm Hữu Hạn Microsoft Vietnam |
| 2  | A0002      | Công ty Trách Nhiệm Hữu Hạn IBM Vietnam       |
| 3  | A0003      | Công ty Trách Nhiệm Hữu Hạn Kodaka Vietnam    |
| 4  | A0004      | Công ty Trách Nhiệm Hữu Hạn E-Google Vietnam  |
| 5  | A0005      | Công ty Cổ phần Suzumi Vietnam                |
| 6  | A0006      | Tập đoàn UCIA USA                             |
| 7  | A0007      | Công ty Đa quốc gia UFCA                      |
| 8  | A0008      | Công ty Cổ phần ReruitVietnam                 |
| 9  | A0009      | Trung tâm giáo dục Vietnam                    |
| 10 | A0010      | Công ty Trách Nhiệm Hữu Hạn Hot Getways       |

**Hình 11-25:** Thêm mới mẫu tin thành công.

### **5.1.2. Thủ tục nội tại để thêm mẩu tin với phương thức RETURN**

Tương tự như trường hợp thêm mới mẫu tin vào bảng Customers, bạn có thể khai báo thủ tục nội tại để thêm mẫu tin vào bảng Categories và sử dụng phát biểu RETURN để trả về được tạo ra bằng cách tính tự động như ví dụ 11-27.

#### **Ví dụ 11-27: Khai báo thủ tục thêm mới mẫu tin**

```
CREATE PROC SPInsUpdCategory
    @CategoryId INT,
    @CategoryNameInVietnamese NVARCHAR(50),
    @CategoryNameInSecondLanguage VARCHAR(50),
    @ShowCategoryNameInVietnamese BIT,
    @Discontinued BIT
AS
DECLARE @NewCategoryId INT
if(@CategoryId = 0)
    BEGIN
        INSERT INTO Categories
        (
            CategoryNameInVietnamese,
            CategoryNameInSecondLanguage,
            ShowCategoryNameInVietnamese,
            Discontinued
        )
        VALUES
        (
            @CategoryNameInVietnamese,
```

M® 174

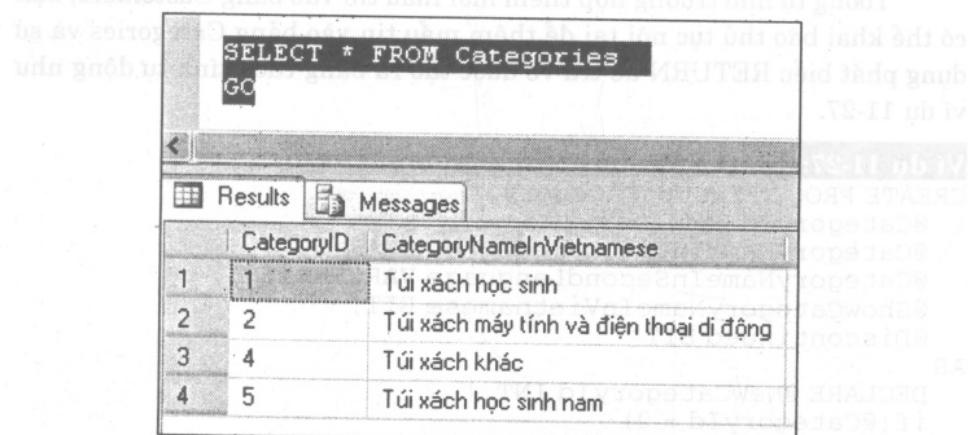
**Chương 11: Khám phá thủ tục nội tại**

```

@CategoryNameInSecondLanguage,
@ShowCategoryNameInVietnamese,
@Discontinued
)
SET @NewCategoryId = @@IDENTITY
END
else
BEGIN
    UPDATE Categories
    SET CategoryNameInVietnamese =
        @CategoryNameInVietnamese,
        CategoryNameInSecondLanguage =
        @CategoryNameInSecondLanguage,
        ShowCategoryNameInVietnamese =
        @ShowCategoryNameInVietnamese,
        Discontinued = @Discontinued
    WHERE CategoryId = @CategoryId
    SET @NewCategoryId = @CategoryId
END
RETURN @NewCategoryId
GO

```

Chẳng hạn, hiện tại chúng ta có danh sách mẫu tin trong bảng Categories như hình 11-26.



The screenshot shows a SQL query window with the following content:

```

SELECT * FROM Categories
GO

```

Below the query window is a results grid titled "Results". The grid displays the following data:

|   | CategoryId | CategoryNameInVietnamese                |
|---|------------|---|
| 1 | 1          | Túi xách học sinh                       |
| 2 | 2          | Túi xách máy tính và điện thoại di động |
| 3 | 4          | Túi xách khác                           |
| 4 | 5          | Túi xách học sinh nam                   |

**Hình 11-26: Danh sách mẫu tin trong bảng Categories.**

Để thêm mẫu tin vào bảng Categories, bạn khai báo gọi thủ tục nội tại SPInsUpdCategory như ví dụ 11-27-1.

**Ví dụ 11-27-1: Khai báo gọi thủ tục nội tại SPInsUpdCategory**

```

DECLARE @CategoryId INT
SET @CategoryId = 0
EXEC @CategoryId = SPInsUpdCategory @CategoryId,
N'Túi xách điện thoại di động',

```

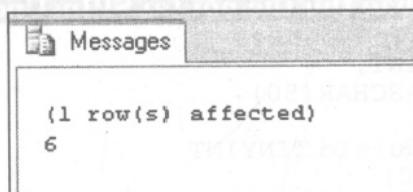
**Chương 11: Khám phá thủ tục nội tại**

175



```
'Mobile Bag', 1, 0
PRINT @CategoryID
GO
```

Nếu thực thi đoạn chương trình trong ví dụ trên, bạn có thể tìm thấy kết quả trình bày như hình 11-27.



**Hình 11-27:** Gọi thủ tục nội tại `SPInsUpdCategory`.

Để kiểm tra mẫu tin vừa thêm mới vào bảng Categories có giá trị là 6 như hình 11-27, bạn có thể thực thi phát biểu SELECT như hình 11-28.

| SELECT * FROM Categories |   |
|--------------------------|---|
| GO                       |   |
|                          |   |
| Results                  | Messages                                |
| CategoryId               | CategoryNameInVietnamese                |
| 1                        | Túi xách học sinh                       |
| 2                        | Túi xách máy tính và điện thoại di động |
| 3                        | Túi xách khác                           |
| 4                        | Túi xách học sinh nam                   |
| 5                        | Túi xách điện thoại di động             |

**Hình 11-28:** Thêm mới mẫu tin thành công.

Chú ý: Cột CategoryId trong bảng Categories có kiểu dữ liệu là INT và là số tự động tăng, đó là lý tại sao chúng ta sử dụng hàm @@IDENTITY.

Tuy nhiên, nếu cột khóa chính có giá trị là số nguyên và không phải là số tự động tăng thì bạn có thể khai báo khác so với những ví dụ trên.

Chẳng hạn, chúng ta có bảng Modules với cấu trúc như ví dụ 11-27-2.

**Ví dụ 11-27-2: Khai báo tạo bảng Modules**

```
CREATE TABLE Modules
(
    ModuleID      TINYINT NOT NULL PRIMARY KEY,
    moduleName NVARCHAR (50) NOT NULL
)
```

**M® 176****Chương 11: Khám phá thủ tục nội tại**

GO

Kế đến, bạn khai báo thủ tục nội tại SPIInsUpdModule để thêm mẫu tin vào bảng Modules với giá trị thêm vào cột ModuleId là số tự động tăng và khai báo trả về như ví dụ 11-27-3.

**Ví dụ 11-27-3: Khai báo thủ tục nội tại SPIInsUpdModule**

```
CREATE PROC SPIInsUpdModule
    @ModuleId TINYINT,
    @ModuleName NVARCHAR (50)
AS
    DECLARE @NewModuleId TINYINT
    if(@ModuleId = 0)
        BEGIN
            WITH LastItem
            AS
            (
                SELECT ModuleId, ROW_NUMBER()
                OVER (ORDER BY ModuleId) AS RowNumber
                FROM Modules
            )
            SELECT @NewModuleId = ModuleId + 1
            FROM LastItem
            WHERE RowNumber =
                dbo.udfTotalRows('Modules')
            IF @NewModuleId IS NULL
                SET @NewModuleId = 1
            INSERT INTO Modules(ModuleId, ModuleName)
            VALUES (@NewModuleId, @ModuleName)
        END
    else
        BEGIN
            UPDATE Modules
            SET ModuleName = @ModuleName
            WHERE ModuleId = @ModuleId
            SET @NewModuleId = @ModuleId
        END
    RETURN @NewModuleId
GO
```

Sau đó, bạn khai báo để gọi thủ tục nội tại SPIInsUpdModule như ví dụ 11-27-4.

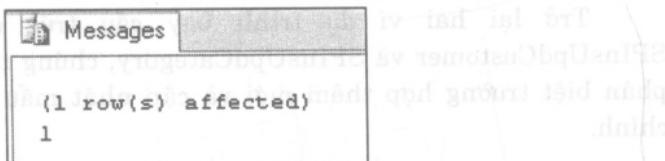
**Ví dụ 11-27-4: Khai báo gọi thủ tục nội tại SPIInsUpdModule**

```
DECLARE @ModuleId TINYINT
SET @ModuleId = 0
EXEC @ModuleId = SPIInsUpdModule @ModuleId,
N'Inventory Control'
PRINT @ModuleId
GO
```

## Chương 11: Khám phá thủ tục nội tại

177 M®

Giả sử, bảng Modules chưa tồn tại mẫu tin, bạn thực thi khai báo trong ví dụ trên thì kết quả trả về như hình 11-29.



**Hình 11-29:** Gọi thủ tục nội tại SPInsUpdModule.

Bạn có thể kiểm tra danh sách mẫu tin trong bảng Modules bằng cách thực thi phát biểu SELECT như hình 11-30.

| SELECT * FROM Modules   |                   |            |   |                   |
|---|-------------------|------------|---|-------------------|
| GO  |                   |            |   |                   |
| Results   |                   |            |   |                   |
| Messages  |                   |            |   |                   |
| <table border="1"> <thead> <tr> <th>ModuleID</th> <th>ModuleName</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>Inventory Control</td> </tr> </tbody> </table> | ModuleID          | ModuleName | 1 | Inventory Control |
| ModuleID  | ModuleName        |            |   |                   |
| 1   | Inventory Control |            |   |                   |

**Hình 11-30:** Danh sách mẫu tin trong bảng Modules.

Chú ý: Trong những ví dụ trên, chúng ta khai báo gọi thủ tục nội tại trong cửa sổ Query của SQL Server 2005. Nếu bạn sử dụng ngôn ngữ lập trình C# hay Visual Basic 2005, khai báo gọi thủ tục nội tại với tham số OUTPUT và giá trị trả về sẽ khác.

Để tham khảo chi tiết về cách gọi thủ tục nội tại như những trường hợp vừa trình bày ở trên, bạn tìm đọc tập 4 thuộc bộ sách “C# 2005” do nhà sách Minh Khai phát hành.

## 5.2. Thủ tục nội tại để cập nhật dữ liệu

Do thông tin cung cấp cho trường hợp thêm mới và cập nhật hầu như giống nhau, chỉ khác nhau đối với giá trị của khóa chính. Khi thêm mới thì giá trị cho cột khóa chính phải được nhập hay tạo tự động, trong khi đó giá trị này đã có sẵn nếu bạn thực hiện tác vụ cập nhật.

Bằng cách dựa vào giá trị của tham số ứng với cột khóa chính, bạn có thể phân biệt trường hợp thêm mới khi giá trị này đang có giá trị là rỗng hay 0 ứng với kiểu dữ liệu là chuỗi hay số.

Ngoài ra, bạn cũng sử dụng phát biểu SQL dạng INSERT hay UPDATE cho trường hợp thêm mới hay cập nhật.

Trở lại hai ví dụ trình bày cấu trúc của thủ tục nội tại SPInsUpdCustomer và SPInsUpdCategory, chúng ta có thể nhận thấy sự phân biệt trường hợp thêm mới và cập nhật mẫu tin dựa vào cột khóa chính.

Chẳng hạn, trường hợp thêm mới mẫu tin vào bảng Customers, bạn đưa vào tham số @CustomerId có kiểu CHAR như sau:

```
IF(@CustomerId = '')
BEGIN
    --Trường hợp thêm mới mẫu tin vào bảng Customers
    END
ELSE
BEGIN
    --Trường hợp cập nhật mẫu tin
    END
```

Tuy nhiên, trong trường hợp bảng Categories với cột khóa CategoryId có kiểu dữ liệu là INT, bạn có thể khai báo tương tự như sau:

```
if(@CategoryId = 0)
BEGIN
    --Trường hợp thêm mới mẫu tin vào bảng Categories
    END
ELSE
BEGIN
    --Trường hợp cập nhật mẫu tin
    END
```

### 5.3. Thủ tục nội tại để xóa dữ liệu

Khác với hai trường hợp thêm và cập nhật dữ liệu, khi xóa dữ liệu của bảng chúng cũng phân ra hai trường hợp là xóa tất cả hay chỉ xóa một mẫu tin dựa vào cột khóa chính.

Chẳng hạn, để xóa dữ liệu trong bảng Suppliers, bạn khai báo như ví dụ 11-28.

#### Ví dụ 11-28: Khai báo thủ tục nội tại xóa dữ liệu

```
CREATE PROC SPDeleteSupplier
    @SupplierId CHAR(5)
AS
    if(@SupplierId = '')
        DELETE FROM Suppliers
    else
        DELETE FROM Suppliers
```

**Chương 11: Khám phá thủ tục nội tại**

179

```
WHERE SupplierId = @SupplierId
GO
```

Khi xóa một mẫu tin ứng với mã, bạn có thể khai báo gọi thủ tục nội tại SPDeleteSupplier như ví dụ 11-28-1.

**Ví dụ 11-28-1: Khai báo gọi thủ tục nội tại SPDeleteSupplier**

```
SPDeleteSupplier 'S0001'
GO
```

Trong trường hợp xóa tất cả mẫu tin trong bảng Suppliers, bạn có thể khai báo gọi thủ tục nội tại SPDeleteSupplier như ví dụ 11-28-2.

**Ví dụ 11-28-2: Khai báo gọi thủ tục nội tại SPDeleteSupplier**

```
SPDeleteSupplier ''
GO
```

Do nhiều bảng dữ liệu có cột khóa chính là kiểu CHAR với chiều dài là 5, bạn có thể viết chung thủ tục nội tại dùng để xóa cho các bảng dữ liệu này.

Chẳng hạn, bạn có thể khai báo dùng để xóa mẫu tin trong bảng dữ liệu có tên Customers, Suppliers, Stocks.

**Ví dụ 11-28-3: Khai báo xóa mẫu tin**

```
CREATE PROC SPDelete
    @ItemId CHAR(5),
    @TableIndex TINYINT
AS
    IF @TableIndex = 1
        DELETE FROM Customers
        WHERE CustomerId = @ItemId
    IF @TableIndex = 2
        DELETE FROM Suppliers
        WHERE SupplierId = @ItemId
    IF @TableIndex = 3
        DELETE FROM Stocks
        WHERE StockId = @ItemId
GO
```

Nếu gọi thủ tục nội tại SPDelete thì bạn khai báo tương tự như ví dụ 11-28-4.

**Ví dụ 11-28-4: Khai báo xóa mẫu tin**

```
SPDelete 'A0010', 1
GO
SPDelete 'S0006', 2
GO
SPDelete 'ST001', 3
GO
```

Lưu ý: Trong trường hợp gọi thủ tục SPDeleteSupplier từ ngôn ngữ lập trình C#, bạn khai báo như ví dụ 11-28-5.

**Ví dụ 11-28-5: Khai báo gọi thủ tục nội tại SPDeleteSupplier từ C#**

```
void Execute()
{
    SqlServerTransaction sqlST = new
    SqlServerTransaction(connectionString);

    // Khai báo tên thủ tục
    string commandText = " SPDeleteSupplier";

    // Khai báo chỉ định loại truy vấn là thủ tục
    int i = sqlST.ExecuteNonQuery(commandText,
        CommandType.StoredProcedure,
        new string[1]{ "@Id" },
        new string[1]{ txtId.Text });
}
```

## 6. THỦ TỤC NỘI TẠI ĐỂ TRUY VẤN DỮ LIỆU

Khi truy vấn dữ liệu từ bảng, chúng ta thường có ba nhu cầu cụ thể. Nhu cầu thứ nhất là liệt kê mẫu tin có giá trị tại khóa chính bằng với giá trị truyền vào, cách lấy giá trị này được dùng khi cần lấy ra mẫu tin ứng với tác vụ trình bày chi tiết hay kích hoạt để cập nhật.

Để thực hiện nhu cầu này, bạn có thể khai báo thủ tục nội tại có tên SPViewSalesInvoices với tham số là khóa chính ứng với InvoiceNo như ví dụ 11-29.

**Ví dụ 11-29: Khai báo liệt kê mẫu tin bảng SPViewSalesInvoices/**

```
CREATE PROC SPViewSalesInvoices
    @InvoiceNo VARCHAR(10)
AS
    SELECT * FROM SalesInvoices
    WHERE InvoiceNo = @InvoiceNo
GO
```

Để gọi thủ tục nội tại SPViewSalesInvoices trong ví dụ trên, bạn khai báo như ví 11-29-1.

**Ví dụ 11-29-1: Khai báo gọi thủ tục nội tại SPViewSalesInvoices**

```
SPViewSalesInvoices 'SI00000003'
GO
```

Nếu thực thi ví dụ trên, bạn có thể tìm thấy kết quả trình bày như hình 11-31.

## Chương 11: Khám phá thủ tục nội tại

181 M®

| InvoiceNo | InvoiceBatch... | DueDate | InvoiceType...      | CustomerID |       |
|-----------|-----------------|---------|---------------------|------------|-------|
| 1         | SI00000003      | SB1001  | 2007-10-12 00:00:00 | SI1        | A0003 |

**Hình 11-31:** Lấy một mẫu tin.

Giả sử, bạn gọi thủ tục nội tại SPViewSalesInvoices trong ví dụ trên với giá trị truyền vào tham số là rỗng, bạn khai báo như ví 11-29-2.

**Ví dụ 11-29-2: Khai báo gọi thủ tục nội tại SPViewSalesInvoices**  
SPViewSalesInvoices  
GO

Nếu thực thi ví dụ trên, bạn có thể tìm thấy kết quả trình bày như hình 11-32.

| InvoiceNo | InvoiceBatch... | DueD... | InvoiceType... | CustomerID |
|-----------|-----------------|---------|----------------|------------|
|           |                 |         |                |            |

**Hình 11-32:** Lấy một mẫu tin.

Trong trường hợp có nhu cầu thứ hai là lấy ra tất cả mẫu tin của bảng PurchaseInvoices, bạn có thể khai báo thủ tục nội tại SPViewPurchaseInvoices như ví dụ 11-30.

**Ví dụ 11-30: Khai báo thủ tục nội tại SPViewPurchaseInvoices**

```
CREATE PROC SPViewPurchaseInvoices
    @InvoiceNo VARCHAR(10)
AS
    IF @InvoiceNo = ''
        SELECT * FROM PurchaseInvoices
    ELSE
        SELECT * FROM PurchaseInvoices
        WHERE InvoiceNo = @InvoiceNo
GO
```

Sau đó, bạn có thể khai báo gọi thủ tục nội tại SPViewPurchaseInvoices với tham số là PI00000002 như ví dụ 11-30-1.

**Ví dụ 11-30-1: Khai báo gọi thủ tục nội tại SPViewPurchaseInvoices**

```
SPViewPurchaseInvoices 'PI00000002'
GO
```

M® 182

**Chương 11: Khám phá thủ tục nội tại**

Nếu thực thi ví dụ trên, bạn có thể tìm thấy kết quả trình bày như hình 11-33.

|   | InvoiceNo  | InvoiceBatch... | DueDate             | InvoiceType... | SupplierID |
|---|------------|-----------------|---------------------|----------------|------------|
| 1 | P100000002 | PBI001          | 2007-10-01 00:00:00 | PB1            | S0002      |

**Hình 11-33: Liệt kê một mẫu tin.**

Trong trường hợp bạn muốn liệt kê tất cả mẫu tin trong bảng thì khai báo gọi thủ tục nội tại SPViewPurchaseInvoices như ví dụ 11-30-2.

**Ví dụ 11-30-2: Khai báo gọi thủ tục nội tại SPViewPurchaseInvoices**  
SPViewPurchaseInvoices  
GO

Nếu thực thi ví dụ trên, bạn có thể tìm thấy kết quả trình bày như hình 11-34.

|   | InvoiceNo  | InvoiceBatch... | DueDate             | InvoiceType... | SupplierID |
|---|------------|-----------------|---------------------|----------------|------------|
| 1 | P100000001 | PBI001          | 2007-10-01 00:00:00 | PB1            | S0001      |
| 2 | P100000002 | PBI001          | 2007-10-01 00:00:00 | PB1            | S0002      |
| 3 | P100000003 | PBI002          | 2007-10-02 00:00:00 | PB1            | S0003      |
| 4 | P100000004 | PBI002          | 2007-10-02 00:00:00 | PB1            | S0001      |
| 5 | P100000005 | PBI002          | 2007-10-02 00:00:00 | PB1            | S0004      |
| 6 | P100000006 | PBI003          | 2007-10-03 00:00:00 | PB1            | S0002      |
| 7 | P100000007 | PBI003          | 2007-10-03 00:00:00 | PB1            | S0003      |
| 8 | P100000008 | PBI004          | 2007-10-04 00:00:00 | PB1            | S0004      |
| 9 | P100000009 | PBI004          | 2007-10-04 00:00:00 | PB1            | S0001      |

**Hình 11-34: Liệt kê tất cả mẫu tin.**

Chú ý: Bạn có thể khai báo giá trị mặc định cho tham số InvoiceNo như ví dụ 11-30-3.

**Ví dụ 11-30-3: Khai báo thủ tục nội tại SPViewPurchaseInvoices**

```
CREATE PROC SPViewPurchaseInvoices
```

```
    @InvoiceNo VARCHAR(10) = ''
```

```
AS
```

```
    IF @InvoiceNo = ''
```

```
        SELECT * FROM PurchaseInvoices
```

**Chương 11: Khám phá thủ tục nội tại**183 

```

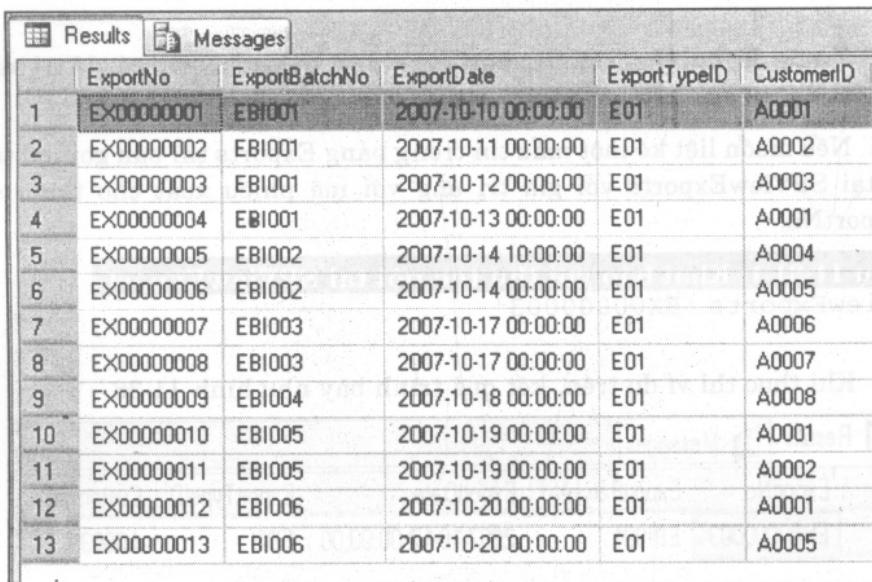
ELSE
    SELECT * FROM PurchaseInvoices
    WHERE InvoiceNo = @InvoiceNo
GO

```

Sau đó, bạn có thể khai báo gọi thủ tục nội tại SPViewPurchaseInvoices không tham số như ví dụ 11-30-4.

**Ví dụ 11-30-4: Khai báo gọi thủ tục nội tại SPViewPurchaseInvoices**  
`SPViewPurchaseInvoices`  
`GO`

Nhu cầu thứ ba là liệt kê danh sách mẫu tin trong bảng dữ liệu dựa trên khóa chính, khóa ngoại khác. Chẳng hạn, chúng ta có danh sách mẫu tin trong bảng Exports như hình 11-35.



|    | ExportNo   | ExportBatchNo | ExportDate          | ExportTypeID | CustomerID |
|----|------------|---------------|---------------------|--------------|------------|
| 1  | EX00000001 | E01001        | 2007-10-10 00:00:00 | E01          | A0001      |
| 2  | EX00000002 | E01001        | 2007-10-11 00:00:00 | E01          | A0002      |
| 3  | EX00000003 | E01001        | 2007-10-12 00:00:00 | E01          | A0003      |
| 4  | EX00000004 | E01001        | 2007-10-13 00:00:00 | E01          | A0001      |
| 5  | EX00000005 | E01002        | 2007-10-14 10:00:00 | E01          | A0004      |
| 6  | EX00000006 | E01002        | 2007-10-14 00:00:00 | E01          | A0005      |
| 7  | EX00000007 | E01003        | 2007-10-17 00:00:00 | E01          | A0006      |
| 8  | EX00000008 | E01003        | 2007-10-17 00:00:00 | E01          | A0007      |
| 9  | EX00000009 | E01004        | 2007-10-18 00:00:00 | E01          | A0008      |
| 10 | EX00000010 | E01005        | 2007-10-19 00:00:00 | E01          | A0001      |
| 11 | EX00000011 | E01005        | 2007-10-19 00:00:00 | E01          | A0002      |
| 12 | EX00000012 | E01006        | 2007-10-20 00:00:00 | E01          | A0001      |
| 13 | EX00000013 | E01006        | 2007-10-20 00:00:00 | E01          | A0005      |

**Hình 11-35: Danh sách mẫu tin trong bảng Exports.**

Trở lại hai ví dụ ứng với nhu cầu thứ nhất và thứ hai vừa trình bày ở trên, bạn có thể khai báo thủ tục nội tại SPViewExports cho nhu cầu thứ ba như ví dụ 11-31.

**Ví dụ 11-31: Khai báo thủ tục nội tại SPViewExports**

```

CREATE PROC SPViewExports
    @ExportNo VARCHAR(10),
    @ExportTypeID CHAR(3) = '',
    @CustomerID CHAR(5) = ''
AS
    IF @ExportNo = ''

```

M® 184

**Chương 11: Khám phá thủ tục nội tại**

```

SELECT * FROM Exports
WHERE ExportTypeId = CASE @ExportTypeId
    WHEN '' THEN ExportTypeId
    ELSE @ExportTypeId END
    AND CustomerId = CASE @CustomerId
        WHEN '' THEN CustomerId
        ELSE @CustomerId END
    ELSE
        SELECT * FROM Exports
        WHERE ExportNo = @ExportNo
        AND ExportTypeId = CASE @ExportTypeId
            WHEN '' THEN ExportTypeId
            ELSE @ExportTypeId END
            AND CustomerId = CASE @CustomerId
                WHEN '' THEN CustomerId
                ELSE @CustomerId END
GO

```

Trong đó, hai tham số @ExportTypeId và @CustomerId có giá trị mặc định là ''.

Nếu muốn liệt kê một mẫu tin trong bảng Exports thì bạn gọi thủ tục nội tại SPViewExports với giá trị ứng với mã phiếu xuất cho tham số @ExportNo.

**Ví dụ 11-31-1: Khai báo gọi thủ tục nội tại SPViewExports**

SPViewExports 'EX00000003'

GO

Khi thực thi ví dụ trên, kết quả trình bày như hình 11-36.

|   | ExportNo   | ExportBatchNo | ExportDate          | ExportTypeID | CustomerId |
|---|------------|---------------|---------------------|--------------|------------|
| 1 | EX00000003 | E01001        | 2007-10-12 00:00:00 | E01          | A0003      |

**Hình 11-36: Liệt kê một mẫu tin.**

Trong trường hợp liệt kê tất cả mẫu tin trong bảng Exports thì bạn gọi thủ tục nội tại SPViewExports với giá trị ứng với mã phiếu xuất cho tham số @ExportNo là rỗng.

**Ví dụ 11-31-2: Khai báo gọi thủ tục nội tại SPViewExports**

SPViewExports ''

GO

Khi thực thi ví dụ trên, kết quả trình bày như hình 11-37.

## **Chương 11: Khám phá thủ tục nội tại**

185

|    | ExportNo   | ExportBatchNo | ExportDate          | ExportTypeID | CustomerID |
|----|------------|---------------|---------------------|--------------|------------|
| 1  | EX00000001 | EBI001        | 2007-10-10 00:00:00 | E01          | A0001      |
| 2  | EX00000002 | EBI001        | 2007-10-11 00:00:00 | E01          | A0002      |
| 3  | EX00000003 | EBI001        | 2007-10-12 00:00:00 | E01          | A0003      |
| 4  | EX00000004 | EBI001        | 2007-10-13 00:00:00 | E01          | A0001      |
| 5  | EX00000005 | EBI002        | 2007-10-14 10:00:00 | E01          | A0004      |
| 6  | EX00000006 | EBI002        | 2007-10-14 00:00:00 | E01          | A0005      |
| 7  | EX00000007 | EBI003        | 2007-10-17 00:00:00 | E01          | A0006      |
| 8  | EX00000008 | EBI003        | 2007-10-17 00:00:00 | E01          | A0007      |
| 9  | EX00000009 | EBI004        | 2007-10-18 00:00:00 | E01          | A0008      |
| 10 | EX00000010 | EBI005        | 2007-10-19 00:00:00 | E01          | A0001      |
| 11 | EX00000011 | EBI005        | 2007-10-19 00:00:00 | E01          | A0002      |
| 12 | EX00000012 | EBI006        | 2007-10-20 00:00:00 | E01          | A0001      |
| 13 | EX00000013 | EBI006        | 2007-10-20 00:00:00 | E01          | A0005      |

**Hình 11-37:** *Liệt kê nhiều mẫu tin.*

Nếu người sử dụng có nhu cầu liệt kê danh sách mẫu tin trong bảng Exports dựa vào các giá trị của khóa ngoại thì bạn có thể khai báo như ví dụ 11-31-3.

#### Ví dụ 11-31-3: Khai báo gửi thủ tục nội tại SPViewExports

```
SPViewExports '' , 'E01' , 'A0001'  
GO
```

Khi thực thi ví dụ trên, kết quả trình bày như hình 11-38.

|   | ExportNo   | ExportBatchNo | ExportDate          | ExportTypeID | CustomerID |
|---|------------|---------------|---------------------|--------------|------------|
| 1 | EX00000001 | EBI001        | 2007-10-10 00:00:00 | E01          | A0001      |
| 2 | EX00000004 | EBI001        | 2007-10-13 00:00:00 | E01          | A0001      |
| 3 | EX00000010 | EBI005        | 2007-10-19 00:00:00 | E01          | A0001      |
| 4 | EX00000012 | EBI006        | 2007-10-20 00:00:00 | E01          | A0001      |

Hình 11-38: *Liệt kê mẫu tin.*

Tóm lại, bạn có thể dựa vào giá trị mặc định của tham số để chia ra nhiều trường hợp khi truy vấn dữ liệu.

Để có thể tìm kiếm danh sách thủ tục nội tại trong cơ sở dữ liệu, bạn có thể sử dụng phát biểu SELECT với bảng dữ liệu sys.procedures như ví dụ 11-32.

#### Ví dụ 11-32: Khai báo liệt kê danh sách thủ tục nội tại

```
SELECT * FROM sys.procedures
GO
```

Khi thực thi phát biểu SELECT trong ví dụ trên, bạn có thể tìm thấy danh sách thủ tục nội tại xuất hiện như hình 11-39.

|    | name                     | object_id | principal_id | schema_id |
|----|--------------------------|-----------|--------------|-----------|
| 1  | sp_helpdiagrams          | 5575058   | NULL         | 1         |
| 2  | sp_helpdiagramdefinition | 21575115  | NULL         | 1         |
| 3  | sp_creatediagram         | 37575172  | NULL         | 1         |
| 4  | sp_renamediagram         | 53575229  | NULL         | 1         |
| 5  | SPInsUpdCategory         | 64719283  | NULL         | 1         |
| 6  | sp_alterdiagram          | 69575286  | NULL         | 1         |
| 7  | SPInsUpdModule           | 80719340  | NULL         | 1         |
| 8  | sp_dropdiagram           | 85575343  | NULL         | 1         |
| 9  | SPDelete                 | 128719511 | NULL         | 1         |
| 10 | SPViewSalesInvoices      | 144719568 | NULL         | 1         |
| 11 | SPViewPurchaseInvoices   | 160719625 | NULL         | 1         |
| 12 | SPViewExports            | 176719682 | NULL         | 1         |
| 13 | SPDel_GLData             | 571149080 | NULL         | 1         |
| 14 | SPInsBanks               | 811149935 | NULL         | 1         |
| 15 | SPGetPurchaseInvoice     | 907150277 | NULL         | 1         |

Hình 11-39: Danh sách thủ tục nội tại.

Tuy nhiên, bạn cũng có thể liệt kê danh sách thủ tục nội tại trong cơ sở dữ liệu bằng cách sử dụng phát biểu SELECT với bảng sysobjects tương tự như ví dụ 11-33.

**Ví dụ 11-33: Khai báo liệt kê danh sách thủ tục nội tại**

```
SELECT * FROM sysobjects
```

```
WHERE xtype='p'
```

```
GO
```

Khi thực thi phát biểu SELECT trong ví dụ trên, bạn có thể tìm thấy danh sách thủ tục nội tại xuất hiện như hình 11-40.

|    | name                            | id         | xtype | uid | info | status | base_s |
|----|---------------------------------|------------|-------|-----|------|--------|--------|
| 1  | sp_helpdiagrams                 | 5575058    | P     | 1   | 0    | 0      | 0      |
| 2  | sp_helpdiagramdefinition        | 21575115   | P     | 1   | 0    | 0      | 0      |
| 3  | sp_creatediagram                | 37575172   | P     | 1   | 0    | 0      | 0      |
| 4  | sp_renamediagram                | 53575229   | P     | 1   | 0    | 0      | 0      |
| 5  | SPInsUpdCategory                | 64719283   | P     | 1   | 0    | 0      | 0      |
| 6  | sp_alterdiagram                 | 69575286   | P     | 1   | 0    | 0      | 0      |
| 7  | SPInsUpdModule                  | 80719340   | P     | 1   | 0    | 0      | 0      |
| 8  | sp_dropdiagram                  | 85575343   | P     | 1   | 0    | 0      | 0      |
| 9  | SPDelete                        | 128719511  | P     | 1   | 0    | 0      | 0      |
| 10 | SPViewSalesInvoices             | 144719568  | P     | 1   | 0    | 0      | 0      |
| 11 | SPViewPurchaseInvoices          | 160719625  | P     | 1   | 0    | 0      | 0      |
| 12 | SPViewExports                   | 176719682  | P     | 1   | 0    | 0      | 0      |
| 13 | SPDel_GLData                    | 571149080  | P     | 1   | 0    | 0      | 0      |
| 14 | SPInsBanks                      | 811149935  | P     | 1   | 0    | 0      | 0      |
| 15 | SPGetPurchaseInvoice            | 907150277  | P     | 1   | 0    | 0      | 0      |
| 16 | SPGetSalesInvoiceBaseOnProvince | 1371151930 | P     | 1   | 0    | 0      | 0      |
| 17 | udfDeleteBackupData             | 1748201278 | P     | 1   | 0    | 0      | 0      |
| 18 | SPDel_ARData                    | 1764201335 | P     | 1   | 0    | 0      | 0      |
| 19 | SPGetSalesInvoice               | 1780201392 | P     | 1   | 0    | 0      | 0      |
| 20 | SPGetPurchaseInvoiceDetails     | 1796201449 | P     | 1   | 0    | 0      | 0      |
| 21 | SPGetQuantityInStock            | 1860201677 | P     | 1   | 0    | 0      | 0      |
| 22 | SPGetBadProductInStock          | 1876201734 | P     | 1   | 0    | 0      | 0      |

**Hình 11-40: Danh sách thủ tục nội tại.**

## 7. KẾT CHƯƠNG

Trong chương này, chúng ta đã tập trung tìm hiểu cách tạo thủ tục nội tại từ cú pháp đơn giản đến phức tạp cho 4 hành động thêm, xóa, cập nhật và truy vấn dữ liệu.

Dựa vào kiểu dữ liệu và chiều dài dữ liệu của cột khóa chính, bạn có thể khai báo một thủ tục dùng chung cho nhiều bảng dữ liệu.

Tương tự như vậy, bạn cũng có thể sử dụng giá trị mặc định của tham số để khai báo thủ tục nội tại cho phép người sử dụng lọc dữ liệu theo nhiều tiêu chí.

Bạn sẽ tiếp tục tìm hiểu một dạng khác của thủ tục nội tại là Trigger.

## *Chương 12:*

# KHÁM PHÁ TRIGGER

## Tóm tắt chương 12

Trong chương trước, chúng ta tìm hiểu về cách lập trình thủ tục và định nghĩa cùng với sử dụng thủ tục nội tại trong SQL Server 2005. Một dạng thủ tục nội tại tiếp tục được giới thiệu trong chương này là Trigger.

Trong chương này, chúng ta tìm hiểu hai loại Trigger chính. Loại thứ nhất dùng để kiểm soát dữ liệu thay đổi được gọi là DML Triggers. Trong khi đó loại thứ hai được dùng để kiểm soát sự thay đổi cấu trúc cơ sở dữ liệu được gọi là DDL Triggers.

Mặc dù, bạn không thể truyền tham số như đã thực hiện điều này cho thủ tục nội tại, Trigger cho phép bạn thực thi tự động phát biểu SQL cùng với các tập lệnh khác dựa trên hành động chính làm thay đổi dữ liệu.

Các vấn đề chính sẽ được đề cập:

- ✓ Giới thiệu Trigger.
  - ✓ DML Trigger.
  - ✓ DDL Trigger.
  - ✓ So sánh DML Trigger và Constraint.

## **1. GIỚI THIỆU THỦ TỤC NỘI TAI**

Để tạo hai cơ chế chính nhằm ràng buộc trọn vẹn cơ sở dữ liệu, SQL Server cung cấp hai quy tắc là Constraint (Bạn đã tìm hiểu cách khai báo Constraint trong các chương trước) và Trigger.

Trigger là thủ tục nội tại đặc biệt mà nó sẽ được gọi một cách tự động khi có biến cố được thực thi trong cơ sở dữ liệu.

**Chú ý:** Bạn có thể tìm thấy các ví dụ trình bày của chương này nằm trong tập tin có tên DDLTriggers.sql và DMLTriggers.sql.

Microsoft SQL Server 2005 giới thiệu hai loại Trigger, loại thứ nhất dùng để kiểm soát dữ liệu thay đổi được gọi là DML Triggers, trong khi đó loại thứ hai được dùng để kiểm soát sự thay đổi cấu trúc cơ sở dữ liệu được gọi là DDL Triggers.

Trong SQL Server 2000, Microsoft đã giới thiệu Trigger dạng thứ nhất, Trigger thứ hai là loại mới trong SQL Server 2005. Những Trigger thuộc loại này đều có liên quan đến biến cố làm thay đổi cấu trúc cơ sở dữ liệu hay trên Server, chúng ta sẽ tìm hiểu chi tiết trong những phần kế tiếp.

Ngoài cách sử dụng MS để định nghĩa Trigger cho cơ sở dữ liệu, bạn cũng có thể sử dụng ngôn ngữ lập trình .NET để khai báo và khởi tạo đối tượng SMO (SQL Server Management Object), sau đó định nghĩa các đối tượng cơ sở dữ liệu.

Mặc dù nó là một loại đặc biệt của thủ tục nội tại, nhưng khi bạn định nghĩa loại Trigger nào đi nữa thì chúng cũng không thể giao tiếp với người sử dụng thông qua tham số input, output. Bạn không thể tác động trực tiếp đến nó thực thi mà nó được thực thi tự động dựa vào sự thay đổi trong cơ sở dữ liệu.

Trigger là một phần của chuyển tác (Transaction) mà nó cần kiểm soát, bởi vì một chuyển tác sẽ không được xem xét kết thúc cho đến khi kết thúc Trigger đang chạy.

Nếu khai báo ROLLBACK TRAN ngay trong đoạn mã của Trigger, bạn thực sự hủy bỏ sự hoạt động của Trigger cũng như hoạt động của chuyển tác đến nơi mà Trigger trực thuộc. Cơ chế hủy sẽ hủy bỏ tất cả quá trình đã thực thi từ khai báo phát biểu BEGIN TRAN.

Chúng ta sẽ tập trung tìm hiểu Trigger loại AFTER. Đại diện cho loại này là DML bao gồm Trigger dùng cho 3 hành động ứng với 3 phát biểu INSERT, UPDATE, DELETE hay nhóm DDL bao gồm Trigger dùng cho hành động tạo, thay đổi hay xóa đối tượng cơ sở dữ liệu ứng với 3 phát biểu CREATE, ALTER, DROP; loại thứ hai là INSTEAD OF (ứng với BEFORE); loại thứ ba là CLR Trigger, bạn có thể phát triển Triggers bằng ngôn ngữ lập trình .NET (dựa trên SMO).

Do DML Trigger là đối tượng cơ sở dữ liệu, nên tên của DML Trigger do người sử dụng đặt phải là duy nhất trong cơ sở dữ liệu hiện hành.



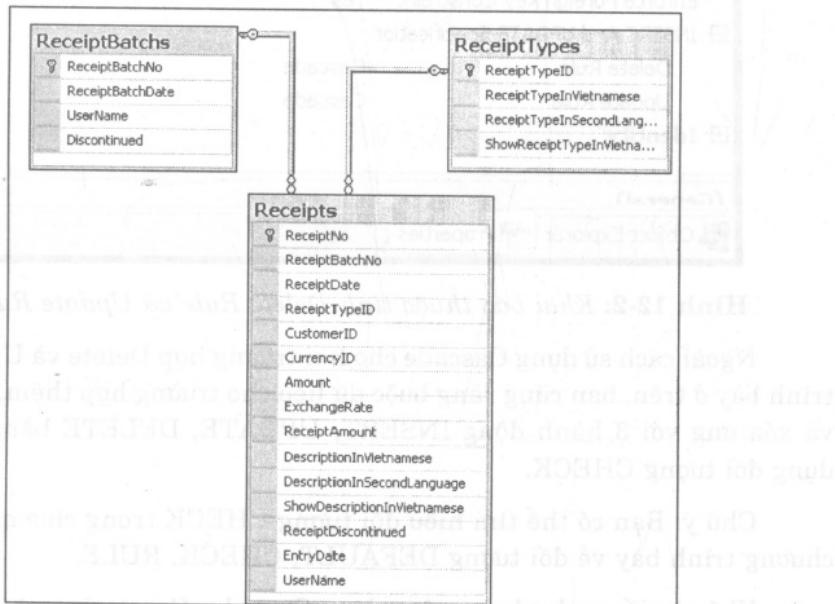
Ngoài ra, quyền để tạo DML Trigger trong cơ sở dữ liệu mặc định là chủ nhân của cơ sở dữ liệu.

**Chú ý:** SQL Server 2005 không hỗ trợ Trigger cho hành động truy vấn ứng với phát biểu SELECT.

## 2. DML TRIGGER

### 2.1. Tại sao sử dụng DML Trigger

Để kiểm soát sự thay đổi dữ liệu trong các bảng dữ liệu có quan hệ ràng buộc trong cơ sở dữ liệu, bạn có thể sử dụng khái niệm cascade giữa hai bảng. Bạn có thể tìm thấy Delete Rule và Update Rule khi tạo ràng buộc giữa hai bảng dữ liệu, để tạo ràng buộc cho hai trường hợp xóa và cập nhật mẫu tin trong bảng cha sẽ có ảnh hưởng đến những mẫu tin có liên quan trong bảng con như hình 12-1.



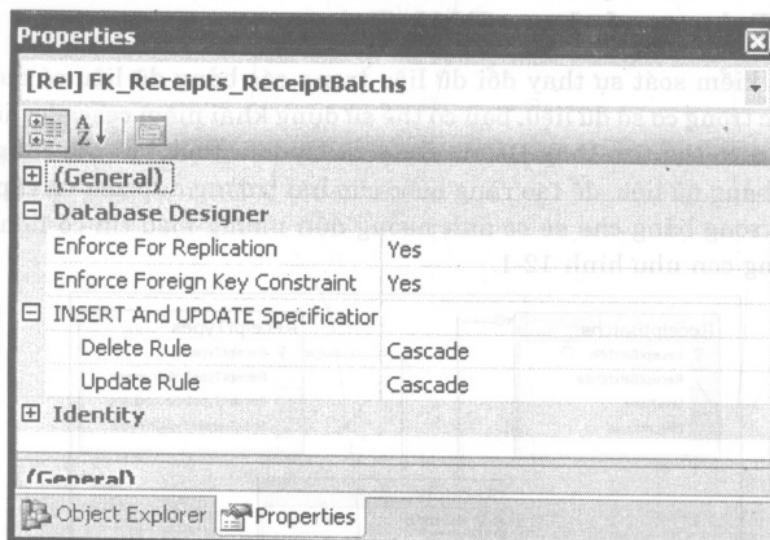
Hình 12-1: Khai báo thuộc tính Cascade.

**Chú ý:** Để tìm hiểu thêm cách tạo quan hệ, bạn có thể tham khảo chương 4 trong tập “SQL Server 2005 - Lập trình T-SQL”.

Để mỗi khi mẫu tin trong bảng cha bị xóa thì những mẫu tin trong bảng con có mã khóa ngoại bằng với khóa chính của mẫu tin đang xóa sẽ bị xóa, bạn cần khai báo Cascade cho trường hợp Delete.

Tương tự như vậy, mỗi khi giá trị trong cột khóa chính của mẫu tin trong bảng cha bị thay đổi thì những mẫu tin trong bảng con có khóa ngoại bằng với khóa chính của mẫu tin đang thay đổi cũng thay đổi theo, bạn cần khai báo Cascade cho trường hợp Update.

Dể làm điều này, trong khi tạo quan hệ giữa hai bảng hay sau khi tạo, bạn có thể kích hoạt cửa sổ thuộc tính, rồi sau đó chọn hai thuộc tính Delete Rule và Update Rule ứng với trường hợp Delete và Update như hình 12-2



**Hình 12-2:** Khai báo thuộc tính Delete Rule và Update Rule

Ngoài cách sử dụng Cascade cho hai trường hợp Delete và Update vừa trình bày ở trên, bạn cũng ràng buộc dữ liệu cho trường hợp thêm, cập nhật và xóa ứng với 3 hành động INSERT, UPDATE, DELETE bằng cách sử dụng đối tượng CHECK.

Chú ý: Bạn có thể tìm hiểu đối tượng CHECK trong chương kế tiếp, chương trình bày về đối tượng DEFAULT CHECK RULE.

Không giống như hai trường hợp Cascade, Constraint và đối tượng CHECK; Trigger loại DML có thể tham chiếu đến các cột khác thay vì cột khóa chính và khóa ngoại trong những bảng dữ liệu mà nó không có quan hệ.

Bạn cũng có thể khai báo cùng loại DML Trigger (INSERT, UPDATE, DELETE) trên cùng một bảng dữ liệu.

Như vậy, nếu bạn chỉ ràng buộc dữ liệu giữa hai bảng thì sử dụng Constraint dựa vào quan hệ và Check dựa trên dữ liệu của cột dữ liệu.

Tuy nhiên, lợi ích của DML Trigger là bạn có thể sử dụng phát biểu SELECT của bảng này để so sánh dữ liệu thêm hay cập nhật vào bảng dữ liệu để thực thi hành động với mục đích nào đó.

Sử dụng DML Trigger, bạn có thể biết được trạng thái của dữ liệu trong bảng trước và sau khi thay đổi để thực thi hành động dựa trên tác động thay đổi đó.

**Chú ý:** Do Trigger là một phần của bảng dữ liệu, nếu bạn xóa bảng dữ liệu thì những Trigger khai báo cho bảng dữ liệu đó sẽ bị xóa theo.

## 2.2. Cấu trúc của Trigger

Để khai báo tạo đối tượng Trigger loại DML, bạn sử dụng cấu trúc như sau:

```
CREATE TRIGGER [ schema_name . ]trigger_name
ON { table | view }
[ WITH <dml_trigger_option> [ ,...n ] ]
{ FOR | AFTER | INSTEAD OF }
{ [ INSERT ] [ , ] [ UPDATE ] [ , ] [ DELETE ] }
[ WITH APPEND ]
[ NOT FOR REPLICATION ]
AS { sql_statement [ ; ] [ ...n ] | EXTERNAL NAME <method
specifier [ ; ] > }
<dml_trigger_option> ::= 
    [ ENCRYPTION ]
    [ EXECUTE AS Clause ]
<methodSpecifier> ::= 
    assembly_name.class_name.method_name
```

Do cú pháp của Trigger cũng khá phức tạp, chúng chỉ tham khảo một vài tham số và sau đó sẽ tìm hiểu thêm qua ví dụ.

- ✓ Tham số trigger\_name là tên của Trigger.
- ✓ Từ khóa ON chỉ định Trigger này sử dụng cho bảng hay đối tượng View nào trong cơ sở dữ liệu {table | view }.
- ✓ WITH chỉ cho phép bạn chỉ định thuộc tính khác cho Trigger, ví dụ như mã hóa nội dung của Trigger bạn sử dụng phát biểu WITH ENCRYPTION.
- ✓ Để chỉ định tác động khiến cho Trigger được thực thi, bạn khai báo từ khóa INSERT, UPDATE, DELETE.
- ✓ sql\_statement là điều kiện và hành động của Trigger, bạn có thể sử dụng phát biểu SQL, phát biểu điều khiển, trong đó có hai đối tượng

ứng với bảng dữ liệu chứa danh sách`mẫu tin bị xóa hay danh sách mẫu tin được thêm mới và cập nhật là DELETED và INSERTED

Chú ý: Hai bảng dữ liệu **DELETED** và **INSERTED** đã được giới thiệu trong phiên bản SQL Server 6.5.

Bảng DELETED sẽ nắm giữ mẫu tin bị xóa mỗi khi người sử dụng chỉ thị thực hiện lệnh xóa bằng phát biểu DELETE hay xóa mẫu tin bằng giao diện trực quan.

Bảng INSERTED sẽ nắm giữ mẫu tin thêm vào mỗi khi người sử dụng chỉ thị lệnh thêm, cập nhật bằng phát biểu INSERT hoặc UPDATE để thêm hay cập nhật dữ liệu từ giao diện trực quan.

Để thay đổi cấu trúc của Trigger đang tồn tại, bạn có thể sử dụng phát biểu ALTER TRIGGER với cú pháp tương tự như sau:

```
ALTER TRIGGER trigger_name
ON table | view
[ WITH <dml_trigger_option> [ ,...n ] ]
( FOR | AFTER | INSTEAD OF )
{ [ DELETE ] [ , ] [ INSERT ] [ , ] [ UPDATE ] }
[ NOT FOR REPLICATION ]
AS { sql_statement [ ; ] [ ...n ] | EXTERNAL NAME <method
specifier> [ ; ] }
<dml_trigger_option> ::==
[ ENCRYPTION ]
[ <EXECUTE AS Clause> ]
<methodSpecifier> ::==
assembly_name.class_name.method_name
```

**Chú ý:** Bạn cũng có thể sử dụng thủ tục hệ thống có tên sp\_rename để thay đổi tên DML Trigger với tên mới.

```
sp_rename [ @objname = ] 'object_name' , [ @newname = ]  
'new_name'  
[ , [ @objtype = ] 'object_type' ]
```

Tương tự như các đối tượng cơ sở dữ liệu khác trong SQL Server 2005, bạn cũng có thể xóa đối tượng Trigger bằng cách sử dụng phát biểu DROP TRIGGER bằng cú pháp như sau:

```
DROP TRIGGER schema_name.trigger_name [ ,...n ] [ ; ]
```

Chẳng hạn, bạn có thể khai báo để tạo đối tượng Trigger ứng với hành động làm thay đổi dữ liệu trên bảng Customers như ví dụ 12-1.

Ví dụ 12-1: Khai báo Trigger cho bảng Customer

```
CREATE TRIGGER reminder1  
ON Customers  
AFTER INSERT, UPDATE
```

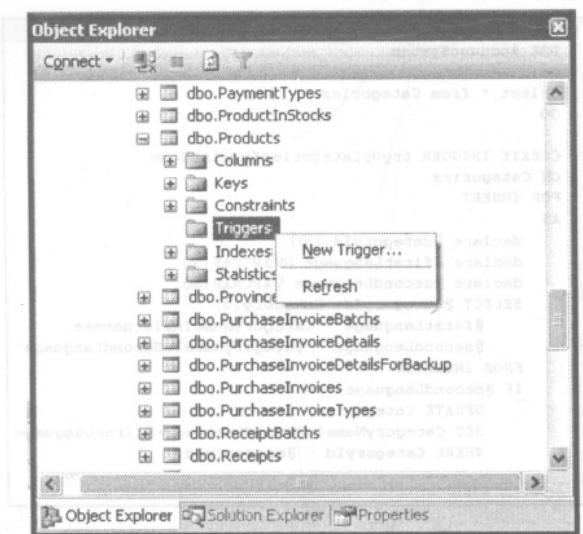
## Chương 12: Khám phá trigger

195 M®

```
AS RAISERROR ('Notify Customer Relations', 16, 10)
GO
```

**2.2.1. Tạo Trigger bằng giao diện MS**

Để tạo Trigger bằng giao diện, bạn bắt đầu từ tên của bảng dữ liệu  
| Triggers | R-Click |, cửa sổ xuất hiện như hình 12-3.

**Hình 12-3: Tạo Trigger bằng MS.**

Sau khi chọn vào New Trigger, bạn có thể tìm thấy cửa sổ dùng để soạn thảo cấu trúc Trigger trông giống như hình 12-4.

```
MY SOLUTION\A...OLQuery2.sql* [MY SOLUTION\Ac...LTriggers.sql*] [Summary]
=====
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
=====
-- Author: <Author, Name>
-- Create date: <Create Date, >
-- Description: <Description, >
=====
CREATE TRIGGER [Schema_Name].[sysname]
  Schema_Name.<Trigger_Name>, sysname, Trigger_Name>
ON [Schema_Name].[sysname],
  Schema_Name.<Table_Name>, sysname, Table_Name>
AFTER <Data_Modification_Statements,
  , INSERT, DELETE, UPDATE>
AS
BEGIN
  -- SET NOCOUNT ON added to prevent extra result sets from
  -- interfering with SELECT statements.
  SET NOCOUNT ON;

  -- Insert statements for trigger here
END
GO
```

**Hình 12-4: Cửa sổ Query mặc định tạo Trigger.**

### **2.2.2. Tạo Trigger bằng phát biểu CREATE từ cửa sổ Query**

Bạn có thể chọn vào nút New Query để mở cửa sổ dùng định nghĩa phát biểu SQL, thủ tục nội tại rồi bạn tự khai báo phát biểu CREATE TRIGGER tương tự như hình 12-5.

```
mysolutionAc...LTriggers.sql* [Summary]
USE AccountSystem
GO
Select * from Categories
GO

CREATE TRIGGER trgUpCategoriesEnglishName
ON Categories
FOR INSERT
AS
    declare @categoryId int;
    declare @firstLanguage NVARCHAR(50);
    declare @secondLanguage VARCHAR(50);
    SELECT @categoryId=CategoryId,
        @firstLanguage = CategoryNameInVietnamese,
        @secondLanguage = CategoryNameInSecondLanguage
    FROM INSERTED
    IF @secondLanguage = ''
        UPDATE Categories
        SET CategoryNameInSecondLanguage=@firstLanguage
        WHERE CategoryId = @categoryId
GO
```

**Hình 12-5:** Mở cửa sổ Query.

### 2.3. Trigger cho hành động thêm mẫu tin

Để kiểm soát việc thêm mẫu tin vào bảng, bạn có thể khai báo Trigger cho hành động INSERT.

Chẳng hạn, khi người sử dụng nhập mới mẫu tin vào bảng Categories, nếu cột dữ liệu có tên CategoryNameInSecondLanguage có giá trị là rỗng thì hệ thống tự động gán giá trị của cột này bằng giá trị của cột CategoryNameInVietnamese.

Để làm điều này, bạn khai báo cấu trúc Trigger ứng với từ khóa AFTER INSERT, tương tự như ví dụ 12-2.

#### Ví dụ 12-2: Khai báo Trigger với hành động INSERT

```
CREATE TRIGGER trgInsCategoriesEnglishName
ON Categories
AFTER INSERT
AS
    declare @categoryId int
    declare @firstLanguage NVARCHAR(50)
    declare @secondLanguage VARCHAR(50)
```

## Chương 12: Khám phá trigger

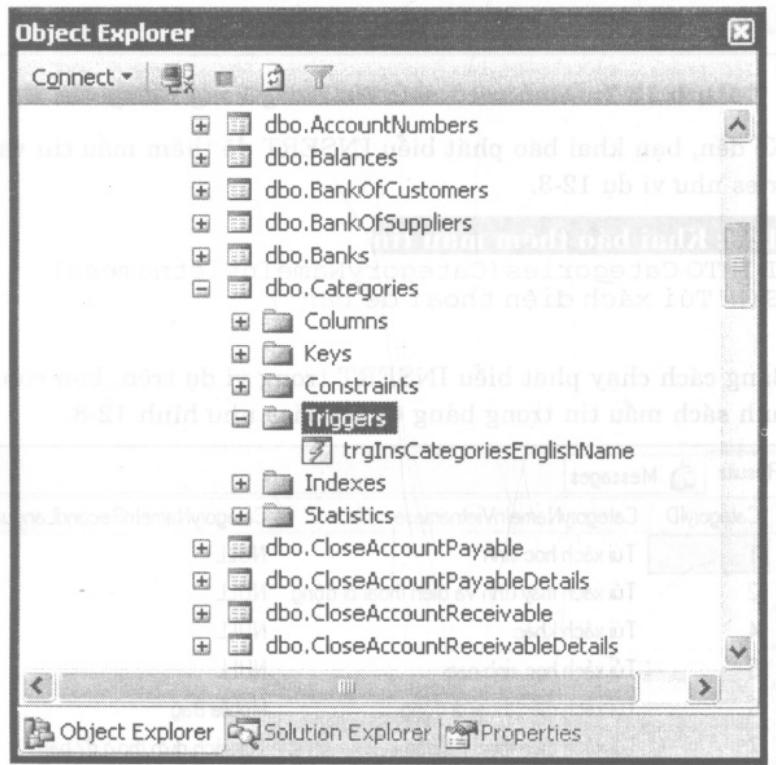
197



```

SELECT @categoryId= CategoryId,
       @firstLanguage = CategoryNameInVietnamese,
       @secondLanguage = CategoryNameInSecondLanguage
  FROM INSERTED
 WHERE @secondLanguage = '' or @secondLanguage is null
    UPDATE Categories
      SET CategoryNameInSecondLanguage = @firstLanguage
     WHERE CategoryId = @categoryId
GO
  
```

Sau khi thực thi phát biểu CREATE TRIGGER trong ví dụ trên, bạn có thể tìm thấy tên Trigger này trong ngăn Triggers như hình 12-6.



**Hình 12-6:** Danh sách Trigger của bảng dữ liệu.

Để kiểm tra tính thi hành của Trigger này, trước tiên chúng ta thử kiểm tra mẫu tin trong bảng Categories bằng phát biểu SELECT như hình 12-7.

| Select * from Categories |            |   |                              |
|--------------------------|------------|---|------------------------------|
|                          | CategoryID | CategoryNameInVietnamese                | CategoryNameInSecondLanguage |
| 1                        | 1          | Túi xách học sinh                       | NULL                         |
| 2                        | 2          | Túi xách máy tính và điện thoại di động | NULL                         |
| 3                        | 4          | Túi xách khác                           | NULL                         |
| 4                        | 5          | Túi xách học sinh nam                   | NULL                         |
| 5                        | 6          | Túi xách điện thoại di động             | Mobile Bag                   |

**Hình 12-7: Danh sách mẫu tin trong bảng Categories.**

Kế đến, bạn khai báo phát biểu INSERT để thêm mẫu tin vào bảng Categories như ví du 12-3.

#### **Ví dụ 12-3: Khai báo thêm mẫu tin**

```
INSERT INTO Categories(CategoryNameInVietnamese)
VALUES(N'Túi xách điện thoại để bàn')
GO
```

Bằng cách chạy phát biểu INSERT trong ví dụ trên, bạn có thể tìm thấy danh sách mẫu tin trong bảng Categories như hình 12-8

| CategoryID | CategoryNameInVietnamese                | CategoryNameInSecondLanguage |
|------------|---|------------------------------|
| 1          | Túi xách học sinh                       | NULL                         |
| 2          | Túi xách máy tính và điện thoại di động | NULL                         |
| 3          | Túi xách khác                           | NULL                         |
| 4          | Túi xách học sinh nam                   | NULL                         |
| 5          | Túi xách điện thoại di động             | Mobile Bag                   |
| 6          | Túi xách điện thoại để bàn              | Túi xách để?n tho?i d? bàn   |

**Hình 12-8:** Trigger đã thực thi.

Lưu ý: Giá trị trong cột CategoryNameInVietnamese không phải là kiểu chứa dữ liệu Unicode.

Trong trường hợp bạn khai báo kiểu dữ liệu cho cột là NVARCHAR thì kết quả trình bày như hình 12.9.

## Chương 12: Khám phá trigger

199



| CategoryID | CategoryNameInVietnamese                | CategoryNameInSecondLanguage |
|------------|---|------------------------------|
| 1          | Túi xách học sinh                       | NULL                         |
| 2          | Túi xách máy tính và điện thoại di động | NULL                         |
| 3          | Túi xách khác                           | NULL                         |
| 4          | Túi xách học sinh nam                   | NULL                         |
| 5          | Túi xách điện thoại di động             | Mobile Bag                   |
| 6          | Túi xách điện thoại để bàn              | Túi xách điện thoại để bàn   |

Hình 12-9: Cập nhật dữ liệu bằng Trigger.

## 2.4. Trigger cho hành động cập nhật mẩu tin

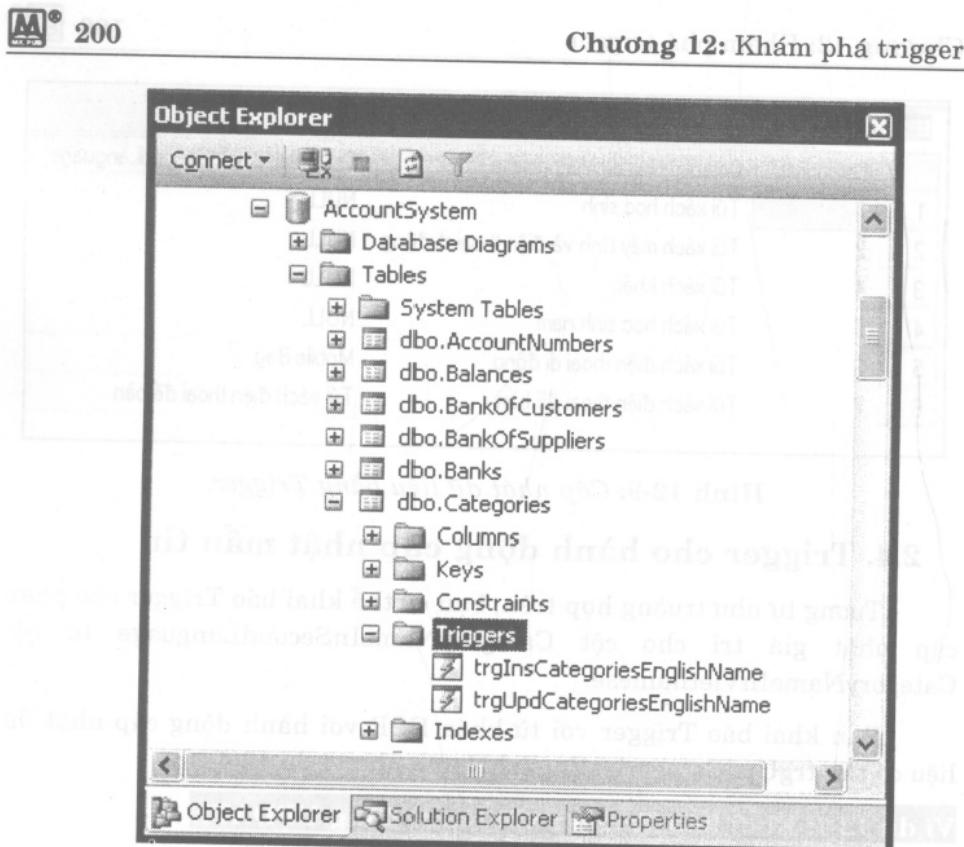
Tương tự như trường hợp trên, bạn có thể khai báo Trigger cho phép cập nhật giá trị cho cột CategoryNameInSecondLanguage từ cột CategoryNameInVietnamese.

Bạn khai báo Trigger với từ khóa FOR với hành động cập nhật dữ liệu có tên trgUpdCategoriesEnglishName như ví dụ 12-4.

### Ví dụ 12-4: Khai báo Trigger với hành động UPDATE

```
CREATE TRIGGER trgUpdCategoriesEnglishName
ON Categories
FOR UPDATE
AS
declare @categoryId int
set @categoryId= 0
declare @firstLanguage NVARCHAR(50)
declare @secondLanguage VARCHAR(50)
SELECT @categoryId= CategoryId,
       @firstLanguage = CategoryNameInVietnamese,
       @secondLanguage = CategoryNameInSecondLanguage
FROM INSERTED
IF @secondLanguage = '' or @secondLanguage is null
    UPDATE Categories
        SET CategoryNameInSecondLanguage=@firstLanguage
        WHERE CategoryId = @categoryId
GO
```

Sau khi thực thi phát biểu CREATE TRIGGER trong ví dụ trên, bạn có thể tìm thấy Trigger này xuất hiện trong ngăn Triggers của cơ sở dữ liệu AccountSystem như hình 12-10.



**Hình 12-10:** Tạo Trigger thành công.

Giả sử chúng ta có dữ liệu trong bảng Categories trình bày như hình 12-11.

```
Select * from Categories  
GO
```

Results Messages

|   | CategoryID | CategoryNameInVietnamese                | CategoryNameInSecondLanguage |
|---|------------|---|------------------------------|
| 1 | 1          | Túi xách học sinh                       | NULL                         |
| 2 | 2          | Túi xách máy tính và điện thoại di động | NULL                         |
| 3 | 4          | Túi xách khác                           | NULL                         |
| 4 | 5          | Túi xách học sinh nam                   | NULL                         |
| 5 | 6          | Túi xách điện thoại di động             | Mobile Bag                   |
| 6 | 7          | Túi xách điện thoại để bàn              | Túi xách điện thoại để bàn   |

**Hình 12-11:** Danh sách mẫu tin trong bảng Categories.

**Chương 12: Khám phá trigger**

201

Để kiểm tra xem Trigger vừa tạo có thực thi hay không, bạn sử dụng phát biểu UPDATE với cấu trúc như ví dụ 12-5.

**Ví dụ 12-5: Khai báo phát biểu UPDATE**

```
UPDATE Categories
SET CategoryNameInVietnamese = N'Túi xách khác'
WHERE CategoryId = 4
GO
```

Sau khi thực thi phát biểu UPDATE này, kết quả trong bảng dữ liệu Categories sẽ giống như hình 12-12.

The screenshot shows a SQL query window with the following content:

```
Select * from Categories
GO
```

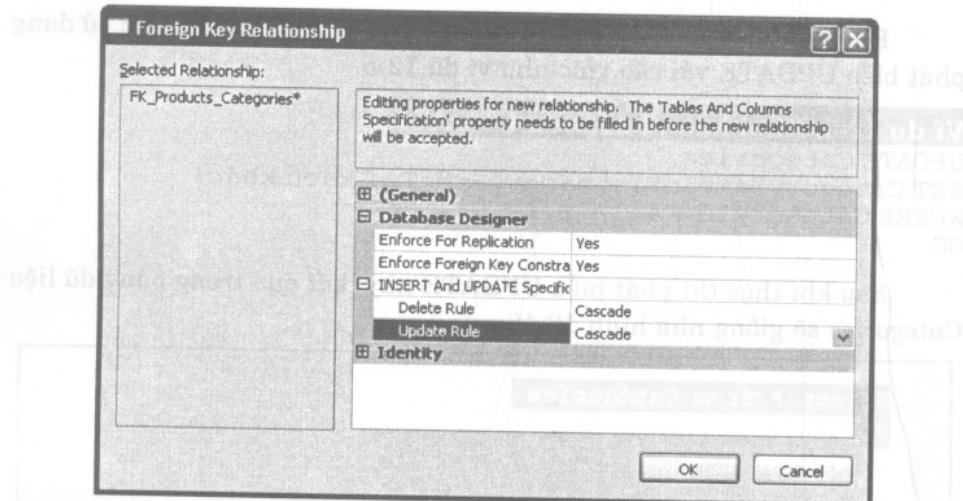
Below the query window is a results grid titled "Results". The grid displays the following data:

| CategoryID | CategoryNameInVietnamese                | CategoryNameInSecondLanguage |
|------------|---|------------------------------|
| 1          | Túi xách học sinh                       | NULL                         |
| 2          | Túi xách máy tính và điện thoại di động | NULL                         |
| 3          | Túi xách khác                           | Túi xách khác                |
| 4          | Túi xách học sinh nam                   | NULL                         |
| 5          | Túi xách điện thoại di động             | Mobile Bag                   |
| 6          | Túi xách điện thoại để bàn              | Túi xách điện thoại để bàn   |

**Hình 12-12: Kiểm tra dữ liệu cập nhật.**

### 2.5. Trigger cho hành động xóa mẫu tin

Trong khi tạo quan hệ giữa bảng Categories và Products, bạn phải khai báo thuộc tính Delete Rule và Update Rule là Cascade, nhằm bảo đảm sự thay đổi dữ liệu trong bảng Categories sẽ thay đổi dữ liệu trong bảng Products.



**Hình 12-13:** Khai báo thuộc tính Delete Rule và Update Rule.

Tuy nhiên, nếu bạn muốn lưu giữ lại mẫu tin vừa xóa trong bảng dữ liệu nào đó, bạn có thể sử dụng đặc điểm của Trigger cho hành động Delete.

Chẳng hạn, chúng ta có danh sách mẫu tin trong bảng SalesPrices như hình 12-14.

```

SELECT * from SalesPrices
GO

```

The Results grid displays the following data:

|   | DateOfPrice         | ProductID | Price | CurrencyID | PriceDiscontinued | UserName |
|---|---------------------|-----------|-------|------------|-------------------|----------|
| 1 | 2007-10-01 00:00:00 | P00001    | 10000 | VND        | 0                 | khang    |
| 2 | 2007-10-01 00:00:00 | P00002    | 10100 | VND        | 0                 | lan      |
| 3 | 2007-10-01 00:00:00 | P00003    | 10100 | VND        | 0                 | ngan     |
| 4 | 2007-10-01 00:00:00 | P00004    | 15000 | VND        | 0                 | hai      |
| 5 | 2007-10-01 00:00:00 | P00005    | 13500 | VND        | 0                 | hoa      |
| 6 | 2007-10-01 00:00:00 | P00006    | 14500 | VND        | 0                 | minh     |
| 7 | 2007-10-01 00:00:00 | P00007    | 16500 | VND        | 0                 | huy      |

**Hình 12-14:** Danh sách mẫu tin trong bảng SalesPrices.

Danh sách trong bảng dữ liệu có tên DeletedSalesPrices trình bày như hình 12-15.

```
Select * from DeletedSalesPrices  
GO
```

Hình 12-15: Danh sách mẫu tin trong bảng DeletedSalesPrices.

Để lưu mẫu tin bị xóa trong bảng SalesPrices vào bảng DeletedSalesPrices mỗi khi người sử dụng thực thi thủ tục nội tại hay phát biểu SQL để xóa mẫu tin nào đó, bạn khai báo Trigger với từ khóa DELETE như ví dụ 12-6.

#### **Ví dụ 12-6: Khai báo xóa mẩu tin**

```
CREATE TRIGGER trgDeleteSalesPrices
ON SalesPrices
AFTER DELETE
AS
    INSERT INTO DeletedSalesPrices
        SELECT * FROM DELETED
GO
```

Chẳng hạn, bạn xóa mẫu tin có mã sản phẩm là P00007 bằng cách khai báo phát biểu Delete như ví dụ 12-7.

#### Ví dụ 12-7: Khai báo xóa sản phẩm P00007

```
Delete from SalesPrices  
Where ProductId = 'P00007'  
And DateOfPrice = CAST('2007-10-01 00:00:00' AS  
SmallDateTime)  
GO
```

Khi thực thi phát biểu Delete trong ví dụ trên, bạn có thể tìm thấy mẫu tin vừa bị xóa trong bảng DeletedSalesPrices như hình 12-16.

```
Select * from DeletedSalesPrices  
GO
```

Results Messages

|   | DateOfPrice         | ProductID | Price | CurrencyID | PriceDiscontinued | UserName |
|---|---------------------|-----------|-------|------------|-------------------|----------|
| 1 | 2007-10-01 00:00:00 | P00007    | 16500 | VND        | 0                 | huy      |

**Hình 12-16:** Mẫu tin bi xóa lưu trong bảng DeletedSalesPrices.

**Chú ý:** Trong trường hợp muốn lưu thêm ngày tháng mẫu tin bị xóa, bạn có thể khai báo thêm cột dữ liệu là DeleteDate rồi sau đó khai báo phát biểu INSERT trong Trigger có chỉ định cột dữ liệu này.

Chẳng hạn, bạn thêm cột dữ liệu là DeletedDate bằng cách sử dụng phát biểu ALTER TABLE như ví dụ 12-8.

**Ví dụ 12-8: Khai báo thêm cột dữ liệu DeletedDate**

```
ALTER TABLE DeletedSalesPrices
ADD DeletedDate SMALLDATETIME
DEFAULT GETDATE()
WITH VALUES
GO
```

**Chú ý:** Mệnh đề WITH VALUES chỉ định hệ thống điền giá trị trong cột vừa mới thêm vào của những mẫu tin đã tồn tại sẽ có giá trị mặc định là ngày hiện hành.

Kế đến, bạn sử dụng phát biểu ALTER TRIGGER để thay đổi cấu trúc của trgDeleteSalesPrices như ví dụ 12-9.

**Ví dụ 12-9: Khai báo thay đổi trgDeleteSalesPrices**

```
ALTER TRIGGER trgDeleteSalesPrices
ON SalesPrices
AFTER DELETE
AS
    INSERT INTO DeletedSalesPrices
        SELECT *, GETDATE()
        FROM DELETED
GO
```

Bạn cũng có thể chỉ định cột dữ liệu trong bảng bằng cách khai báo tương tự như ví dụ 12-10.

**Ví dụ 12-10: Khai báo thay đổi trgDeleteSalesPrices**

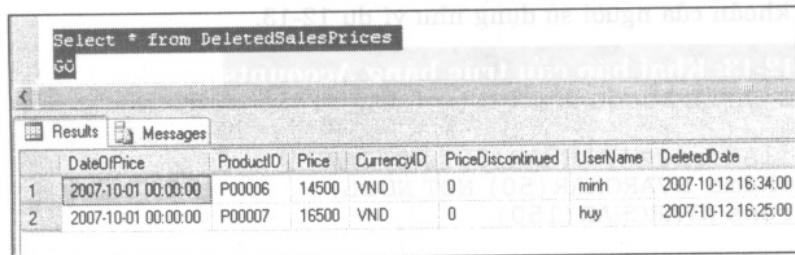
```
ALTER TRIGGER trgDeleteSalesPrices
ON SalesPrices
AFTER DELETE
AS
    INSERT INTO DeletedSalesPrices
        (DateOfPrice, ProductId, Price, CurrencyId,
        PriceDiscontinued, UserName, DeletedDate)
        SELECT DateOfPrice, ProductId, Price, CurrencyId,
        PriceDiscontinued, UserName, GETDATE()
        FROM DELETED
GO
```

Sau khi thực thi phát biểu ALTER TRIGGER của một trong hai ví dụ trên, bạn có thể xóa mẫu tin trong bảng SalesPrices bằng phát biểu Delete như ví dụ 12-11.

#### **Ví dụ 12-11: Khai báo xóa sản phẩm có mã là P00006**

```
Delete from SalesPrices
Where ProductId = 'P00006'
And DateOfPrice = CAST('2007-10-01 00:00:00' AS
SmallDateTime)
GO
```

Nếu thực thi phát biểu Delete trong ví dụ trên, bạn có thể tìm thấy mẫu tin này xuất hiện trong bảng DeletedSalesPrices như hình 12-17.



The screenshot shows a SQL query window with the following content:

```
Select * from DeletedSalesPrices
GO
```

The results pane displays the following data:

|   | DateOfPrice         | ProductId | Price | CurrencyID | PriceDiscontinued | UserName | DeletedDate         |
|---|---------------------|-----------|-------|------------|-------------------|----------|---------------------|
| 1 | 2007-10-01 00:00:00 | P00006    | 14500 | VND        | 0                 | minh     | 2007-10-12 16:34:00 |
| 2 | 2007-10-01 00:00:00 | P00007    | 16500 | VND        | 0                 | huy      | 2007-10-12 16:25:00 |

**Hình 12-17: Lưu mẫu tin bị xóa.**

Chú ý: So sánh hình 12-7 và 12-16, nếu bạn không sử dụng mệnh đề WITH VALUES thì giá trị trong cột mới là DeletedDate sẽ là NULL.

Trong những ví dụ trên, chúng ta phân ra ba trường hợp thêm, cập nhật và xóa dữ liệu trong bảng để thực hiện tác vụ chỉ định Delete, Insert, Update phù hợp. Tuy nhiên, bạn có thể khai báo một Trigger dùng chung cho hai hay ba trường hợp thêm, cập nhật hay xóa dữ liệu.

Chẳng hạn, bạn có thể khai báo Trigger dùng chung cho trường hợp thêm và cập nhật mẫu tin trong bảng Categories như ví dụ 12-12.

#### **Ví dụ 12-12: Khai báo Trigger cho trường hợp thêm và cập nhật**

```
CREATE TRIGGER trgInsAndUpdateCategoriesEnglishName
ON Categories
AFTER INSERT, UPDATE
AS
declare @categoryId int
set @categoryId= 0
declare @firstLanguage NVARCHAR(50)
declare @secondLanguage VARCHAR(50)
SELECT @categoryId= CategoryId,
       @firstLanguage = CategoryNameInVietnamese,
       @secondLanguage = CategoryNameInSecondLanguage
FROM INSERTED
```

**M® 206****Chương 12: Khám phá trigger**

```
IF @secondLanguage = '' or @secondLanguage is null
    UPDATE Categories
        SET CategoryNameInSecondLanguage=@firstLanguage
    WHERE CategoryId = @categoryId
GO
```

Trong những ví dụ trên, chúng ta khai báo Trigger cho bảng dữ liệu, bây giờ chúng ta sang phần Trigger cho đối tượng View. Chẳng hạn, bạn cũng có thể khai báo Trigger để kiểm soát tài khoản của người sử dụng ứng với trường hợp người sử dụng đăng ký tài khoản trong bảng Accounts thông qua vwAccount.

Để làm điều này, trước tiên chúng ta tạo bảng Accounts dùng để lưu trữ tài khoản của người sử dụng như ví dụ 12-13.

**Ví dụ 12-13: Khai báo cấu trúc bảng Accounts**

```
CREATE TABLE Accounts
(
    EmailAddress VARCHAR(50) NOT NULL,
    FullName NVARCHAR(50) NOT NULL,
    Address NVARCHAR(150)
)
GO
```

Kế đến, mỗi khi người sử dụng thay đổi thông tin trong bảng Accounts thì bạn khai báo bảng dữ liệu có tên EmailsToSend để lưu thông tin của Email sẽ gửi đến Email của họ như ví dụ 12-14.

**Ví dụ 12-14: Khai báo tạo bảng dữ liệu EmailsToSend**

```
CREATE TABLE EmailsToSend
(
    EmailAddressToReceipt VARCHAR(50) NOT NULL,
    Subject NVARCHAR(50) NOT NULL,
    Body NVARCHAR(max)
)
GO
```

Tiếp theo, bạn khai báo đối tượng View cho bảng Accounts tương tự như ví dụ 12-15.

**Ví dụ 12-15: Khai báo đối tượng View**

```
CREATE VIEW vwAccounts
AS
SELECT * FROM Accounts
GO
```

Sau đó, bạn tiếp tục khai báo Trigger nhằm kiểm soát hành động thêm tài khoản trong bảng Accounts thông qua đối tượng vwAccounts như ví dụ 12-16.

**Ví dụ 12-16: Khai báo Trigger cho đối tượng vwAccounts**

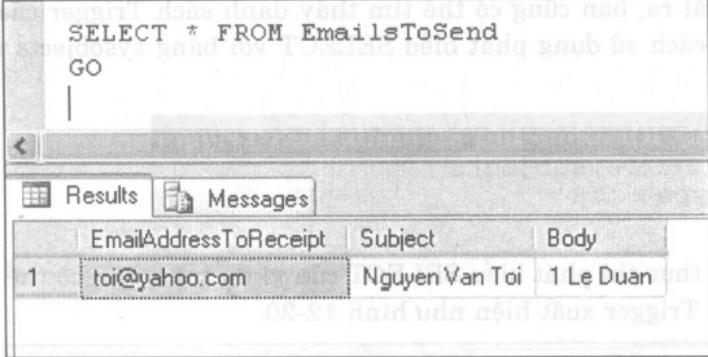
```
CREATE TRIGGER trgEmails
ON vwAccounts
INSTEAD OF INSERT
AS
    INSERT INTO EmailsToSend
        SELECT EmailAddress,
        FullName, Address
    FROM INSERTED
GO
```

Để kiểm tra Trigger có kích hoạt hay không, bạn khai báo phát biểu INSERT để thêm mẫu tin vào bảng Accounts thông qua vwAccounts như ví dụ 12-17.

**Ví dụ 12-17: Khai báo thêm mẫu tin vào vwAccounts**

```
INSERT INTO vwAccounts
VALUES ('toi@yahoo.com', 'Nguyen Van Toi', '1 Le Duan')
GO
```

Khi thực thi phát biểu INSERT của ví dụ trên, bạn có thể tìm thấy mẫu tin xuất hiện trong bảng bằng cách khai báo phát biểu SELECT như hình 12-18.



The screenshot shows a SQL query window with the following content:

```
SELECT * FROM EmailsToSend
GO
```

Below the query window is a results grid titled "Results". The grid has three columns: "EmailAddressToReceipt", "Subject", and "Body". There is one row of data:

| EmailAddressToReceipt | Subject        | Body      |
|-----------------------|----------------|-----------|
| toi@yahoo.com         | Nguyen Van Toi | 1 Le Duan |

**Hình 12-18: Trigger trong đối tượng View.**

Đây là những ví dụ dùng để tham khảo các khai báo Trigger cho cả ba hành động thêm, cập nhật và xóa bảng Account có thể không có mục đích sử dụng trong cơ sở dữ liệu kế toán.

Tuy nhiên, bạn có thể tìm hiểu cách xây dựng Trigger cho những bảng dữ liệu xuất nhập trong phần xây dựng ứng dụng.

Chú ý: Cũng như các đối tượng cơ sở dữ liệu khác, bạn cũng có thể tìm thấy danh sách Trigger của cơ sở dữ liệu bằng cách sử dụng phát biểu SELECT với bảng sys.triggers như ví dụ 12-18.

#### **Ví dụ 12-18: Khai báo liệt kê danh sách Trigger**

```
SELECT * FROM sys.triggers  
GO
```

Nếu thực thi phát biểu SELECT của ví dụ trên, bạn có thể tìm thấy danh sách Trigger xuất hiện như hình 12-19.

|   | name                        | object_id  | parent_class | parent_class_desc |
|---|-----------------------------|------------|--------------|-------------------|
| 1 | trgInsCategoriesEnglishName | 1728725211 | 1            | OBJECT_OR_COLUMN  |
| 2 | trgUpdCategoriesEnglishName | 1744725268 | 1            | OBJECT_OR_COLUMN  |
| 3 | trgDeleteSalesPrices        | 1872725724 | 1            | OBJECT_OR_COLUMN  |
| 4 | trgEmails                   | 1984726123 | 1            | OBJECT_OR_COLUMN  |

**Hình 12-19:** Danh sách Trigger.

Ngoài ra, bạn cũng có thể tìm thấy danh sách Trigger của cơ sở dữ liệu bằng cách sử dụng phát biểu SELECT với bảng sysobjects như ví dụ 12-19.

#### Ví dụ 12-19: Khai báo liệt kê danh sách Trigger

```
SELECT * FROM sysobjects  
WHERE xtype = 'TR'  
GO
```

Nếu thực thi phát biểu SELECT của ví dụ trên, bạn có thể tìm thấy danh sách Trigger xuất hiện như hình 12-20.

|   | name                        | id         | xtype | uid | info | status |
|---|-----------------------------|------------|-------|-----|------|--------|
| 1 | trgInsCategoriesEnglishName | 1728725211 | TR    | 1   | 0    | 0      |
| 2 | trgUpdCategoriesEnglishName | 1744725268 | TR    | 1   | 0    | 0      |
| 3 | trgDeleteSalesPrices        | 1872725724 | TR    | 1   | 0    | 0      |
| 4 | trgEmails                   | 1984726123 | TR    | 1   | 0    | 0      |

**Hình 12-20:** Danh sách Trigger.

### 3. DDL TRIGGER

DDL Trigger cũng có khái niệm giống như Trigger mà chúng ta vừa tìm hiểu trong phần trên. Tuy nhiên, cấu trúc và tầm vực của DDL Trigger không giống như DML Trigger là bị kích hoạt khi có sự thay đổi dữ liệu từ tác động của 3 phát biểu UPDATE, INSERT, DELETE trên đối tượng Table hay View; mà nó sẽ được kích hoạt khi người sử dụng làm thay đổi cấu trúc cơ sở dữ liệu hay đối tượng cơ sở dữ liệu bằng các phát biểu SQL thuộc DDL.

Chúng ta đã tham khảo các phát biểu dùng để tạo, thay đổi hay xóa cơ sở dữ liệu cũng như đối tượng cơ sở dữ liệu là CREATE, ALTER, DROP, GRANT, DENY, REVOKE hay UPDATE STATISTICS. Nếu như DML Trigger dùng để kiểm soát dữ liệu chứa trong Table hay View thì DDL Trigger có thể được sử dụng cho chức năng quản trị cơ sở dữ liệu.

#### 3.1. Tại sao sử dụng DDL Trigger

Sau khi giai đoạn thiết kế cơ sở dữ liệu hoàn tất, để kiểm soát mọi sự thay đổi cấu trúc của cơ sở dữ liệu, chúng ta cần sử dụng loại Trigger này.

Như vậy, mục đích của việc khai báo và sử dụng DDL Trigger là ngăn ngừa sự thay đổi cấu trúc cơ sở dữ liệu. Ngoài ra, bạn cần ghi lại những hành động làm thay đổi cấu trúc cơ sở dữ liệu thì sử dụng loại Trigger này.

Chú ý: DDL Trigger sẽ được kích hoạt sau khi phát biểu SQL dạng DDL thực thi. DDL Trigger không thể sử dụng như INSTEAD OF Trigger.

#### 3.2. Cấu trúc của Trigger

Để khai báo tạo đối tượng Trigger loại DDL, bạn sử dụng cấu trúc như sau:

```
CREATE TRIGGER trigger_name
ON { ALL SERVER | DATABASE }
[ WITH <ddl_trigger_option> [ ,...n ] ]
{ FOR | AFTER } { event_type | event_group } [ ,...n ]
AS { sql_statement [ ; ] { ...n } |
EXTERNAL NAME <method specifier> { ; } }
<ddl_trigger_option> ::= 
    [ ENCRYPTION ]
    [ EXECUTE AS Clause ]
<methodSpecifier> ::= 
    assembly_name.class_name.method_name
```

Tương tự như DML Trigger, do cú pháp của DDL Trigger cũng khá phức tạp, chúng ta chỉ trình bày chi tiết qua từng ví dụ.

- ✓ Tham số trigger\_name là tên của Trigger.
  - ✓ Từ khóa ON chỉ định Trigger này sử dụng cho mọi cơ sở dữ liệu trên Server (ALL SERVER) hay chỉ cơ sở dữ liệu hiện hành (DATABASE).
  - ✓ WITH cho phép bạn chỉ định thuộc tính khác cho Trigger, ví dụ như mã hóa nội dung của Trigger bạn sử dụng phát biểu WITH ENCRYPTION.
  - ✓ Để chỉ định hành động Trigger sẽ được thực thi, bạn khai báo từ khóa CREATE, ALTER, DROP, GRANT, DENY, REVOKE, UPDATE STATISTICS.
  - ✓ sql\_statement là khai báo cho điều kiện và hành động của Trigger, bạn có thể sử dụng phát biểu SQL, phát biểu điều khiển

**Chú ý:** Sau đây là danh sách biến cố mà DDL Trigger sẽ được kích hoạt khi các hành động này được gọi trong cơ sở dữ liệu hiện hành.

|                           |                           |
|---------------------------|---------------------------|
| Create_Application_Role   | Alter_Application_Role    |
| Drop_Application_Role     | Create_Assembly           |
| Alter_Assembly            | Drop_Assembly             |
| Create_Certificate        | Alter_Certificate         |
| Drop_Certificate          | Create_Contract           |
| Drop_Contract             | Grant_Database            |
| Deny_Database             | Revoke_Database           |
| Create_Event_Notification | Drop_Event_Notification   |
| Create_Function           | Alter_Function            |
| Drop_Function             | Create_Index              |
| Alter_Index               | Drop_Index                |
| Create_Message_Type       | Alter_Message_Type        |
| Drop_Message_Type         | Create_Partition_Function |
| Alter_Partition_Function  | Drop_Partition_Function   |
| Create_Partition_Scheme   | Alter_Partition_Scheme    |

**Chương 12: Khám phá trigger**211 

|                       |                  |
|-----------------------|------------------|
| Drop_Partition_Scheme | Create_Procedure |
| Alter_Procedure       | Drop_Procedure   |
| Create_Queue          | Alter_Queue      |
| Create_Role           | Alter_Role       |
| Drop_Role             | Create_Route     |
| Alter_Route           | Drop_Route       |
| Create_Schema         | Alter_Schema     |
| Drop_Schema           | Create_Service   |
| Alter_Service         | Drop_Service     |
| Create_Statistics     | Drop_Statistics  |
| Update_Statistics     | Create_Synonym   |
| Drop_Synonym          | Create_Table     |
| Alter_Table           | Drop_Table       |
| Create_Trigger        | Alter_Trigger    |
| Drop_Trigger          | Drop_User        |
| Create_Type           | Drop_Type        |
| Create_User           | Alter_User       |
| Create_View           | Alter_View       |
| Drop_View             | Drop_Queue       |

Đối với trường hợp Server, chúng ta có thể sử dụng các biến cố như sau cho DDL Trigger:

|                            |                 |
|----------------------------|-----------------|
| Alter_Authorization_Server | Drop_Login      |
| Create_Database            | Alter_Database  |
| Drop_Database              | Create_Endpoint |
| Drop_Endpoint              | Create_Login    |
| Alter_Login                | Grant_Server    |
| Deny_Server                | Revoke_Server   |

Tương tự như trong trường hợp trình bày DML Trigger, để thay đổi cấu trúc của DDL Trigger đang tồn tại, bạn có thể sử dụng phát biểu ALTER TRIGGER với cú pháp tương tự như sau:

```
ALTER TRIGGER trigger_name
ON { DATABASE | ALL SERVER }
[ WITH <ddl_trigger_option> [ ,...n ] ]
{ FOR | AFTER } { event_type [ ,...n ] | event_group }
AS { sql_statement [ ; ] | EXTERNAL NAME <method specifier>
[ ; ] }
<ddl_trigger_option> ::= 
    [ ENCRYPTION ]
    [ <EXECUTE AS Clause> ]
<method specifier> ::=
    assembly_name.class_name.method_name
```

Ngoài ra, bạn cũng có thể xóa đối tượng DDL Trigger bằng cách sử dụng phát biểu DROP TRIGGER bằng cú pháp như sau:

```
DROP TRIGGER trigger_name [ ,...n ]
ON { DATABASE | ALL SERVER }
[ ; ]
```

Chẳng hạn, bạn có thể khai báo để tạo đối tượng DDL Trigger ứng với hành động làm thay đổi cấu trúc bảng trong cơ sở dữ liệu hiện hành như ví dụ 12-20.

#### **Ví dụ 12-20: Khai báo tao DDL Trigger**

```
IF EXISTS (SELECT * FROM sys.triggers
    WHERE name = 'DDL_Trigger_On_Database')
DROP TRIGGER DDL_Trigger_On_Database
ON DATABASE
GO
CREATE TRIGGER DDL_Trigger_On_Database
ON DATABASE
FOR DDL_TABLE_VIEW_EVENTS
AS
    PRINT 'DDL_Trigger_On_Database'
GO
```

Sau đó, bạn có thể xóa DDL Trigger trên bằng cách khai báo như ví dụ 12-21.

#### **Ví dụ 12-21: Khai báo xóa DDL Trigger**

```
DROP TRIGGER DDL_Trigger_On_Database
ON DATABASE
GO
```

Đối với trường hợp tạo DDL Trigger trên Server, bạn có thể khai báo tương tự như ví dụ 12-22.

**Chương 12: Khám phá trigger**

213

**Ví dụ 12-22: Khai báo tạo DDL Trigger trên Server**

```

IF EXISTS (SELECT * FROM sys.server_triggers
    WHERE name = 'DDL_Trigger_On_Server')
DROP TRIGGER DDL_Trigger_On_Server
ON ALL SERVER
GO
CREATE TRIGGER DDL_Trigger_On_Server
ON ALL SERVER
FOR DDL_LOGIN_EVENTS
AS
    PRINT 'DDL_Trigger_On_Server'
GO

```

Tương tự như trường hợp xóa DDL Trigger trên Server, bạn có thể xóa DDL Trigger của Server bằng cách khai báo như ví dụ 12-23.

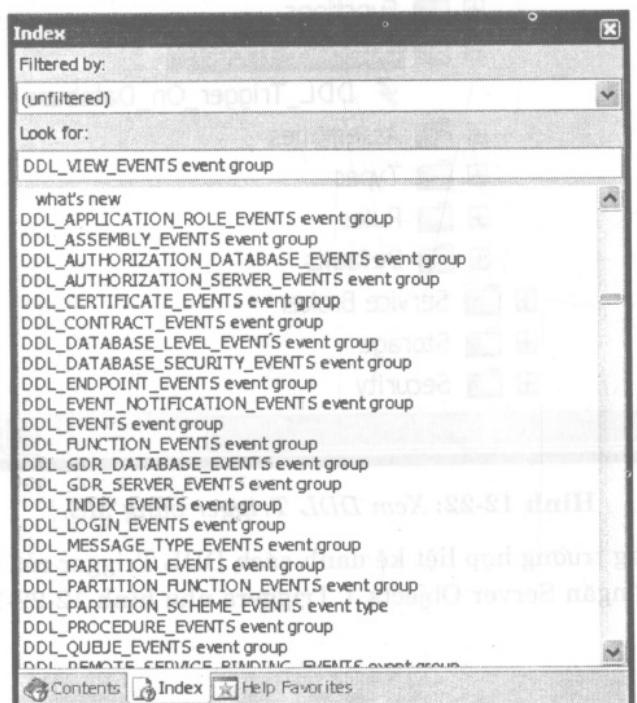
**Ví dụ 12-23: Khai báo xóa DDL Trigger**

```

DROP TRIGGER DDL_Trigger_On_Server
ON ALL SERVER
GO

```

Chú ý: Bạn có thể tìm thấy danh sách các biến cố trong cửa sổ tìm kiếm chỉ mục (Index) của SQL Server 2005 như hình 12-21.

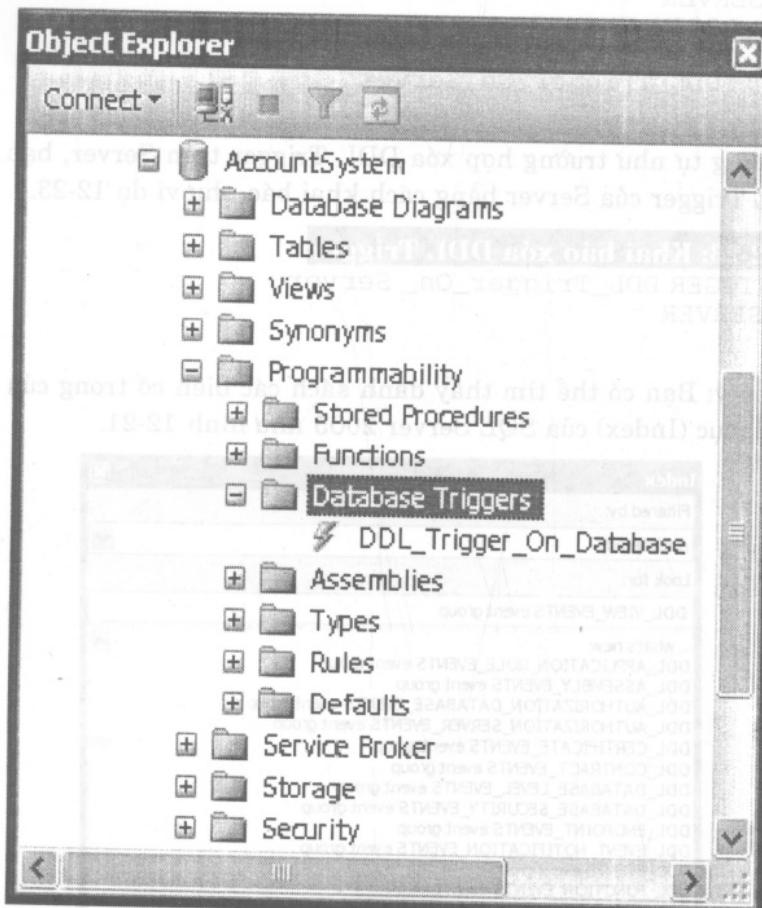


**Hình 12-21:** Danh sách biến cố khai báo trong DDL Trigger.

### 3.3. Danh sách DDL Trigger

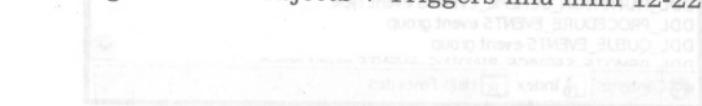
#### 3.3.1. Xem DDL Trigger bằng giao diện MS

Để xem DDL Trigger của cơ sở dữ liệu hiện hành bằng giao diện, bạn bắt đầu từ ngăn Database | AccountSystem | Programmability | Database Triggers như hình 12-22.



Hình 12-22: Xem DDL Trigger bằng MS.

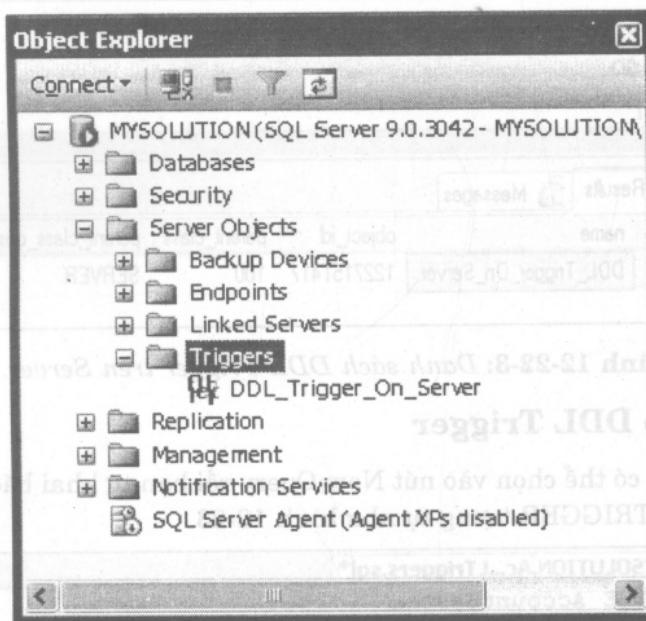
Trong trường hợp liệt kê danh sách DDL Trigger của Server, bạn bắt đầu từ ngăn Server Objects | Triggers như hình 12-22-1.



Hình 12-22-1: Danh sách trigger của cơ sở dữ liệu DML trigger

**Chương 12: Khám phá trigger**

215 M®

**Hình 12-22-1: DDL Trigger trên Server.****3.3.2. Xem danh sách DDL Trigger bằng phát biểu SQL**

Bạn có thể liệt kê danh sách DDL Trigger của cơ sở dữ liệu hiện hành bằng phát biểu SELECT như hình 12-22-2.

```
SELECT * FROM sys.triggers
GO
```

|   | name            | object_id | parent_class | parent_class_desc |
|---|-----------------|-----------|--------------|-------------------|
| 1 | DatabaseChanged | 656721392 | 0            | DATABASE          |

**Hình 12-22-2: Danh sách DDL Trigger trên cơ sở dữ liệu.**

Bạn có thể liệt kê danh sách DDL Trigger của Server hiện hành bằng phát biểu SELECT như hình 12-22-3.

|   | name                  | object_id  | parent_class | parent_class_desc |
|---|-----------------------|------------|--------------|-------------------|
| 1 | DDL_Trigger_On_Server | 1227151417 | 100          | SERVER            |

Hình 12-22-3: Danh sách DDL Trigger trên Server.

### 3.4. Tạo DDL Trigger

Bạn có thể chọn vào nút New Query rồi bạn tự khai báo phát biểu CREATE TRIGGER tương tự như hình 12-23.

```

USE AccountSystem
GO

IF EXISTS (SELECT * FROM sys.triggers
WHERE name = 'DDL_Trigger_On_Database')
DROP TRIGGER DDL_Trigger_On_Database
ON DATABASE
GO
CREATE TRIGGER DDL_Trigger_On_Database
ON DATABASE
FOR DDL_TABLE_VIEW_EVENTS
AS
    PRINT 'DDL_Trigger_On_Database'
GO
DROP TRIGGER DDL_Trigger_On_Database
ON DATABASE
GO

```

Hình 12-23: Mở cửa sổ Query để định nghĩa DDL Trigger.

### 3.5. Hàm EVENTDATA

Mọi biến cố khai báo trong DDL Trigger để kiểm soát cơ sở dữ liệu hiện hành hay Server đều cung cấp một số thuộc tính lưu trữ theo định dạng XML mà bạn có thể tìm thấy bằng cách gọi hàm EVENTDATA.

**Chương 12: Khám phá trigger**217 

Trong đó, hàm EVENTDATA trả về thông tin của biến cố xảy ra trên Server hay cơ sở dữ liệu hiện hành. Hàm này sẽ được gọi khi biến cố bị kích hoạt và thông tin được trả về ứng với biến cố cụ thể khi khai báo DDL Trigger.

Ngoài ra, hàm EVENTDATA trả về dữ liệu theo định dạng XML. Dữ liệu này được gửi đến phía trình khách theo dạng Unicode với chuẩn 2 bytes.

Ví dụ, khi bạn sử dụng phát biểu ALTER TABLE, biến ALTER\_TABLE sẽ trả về dữ liệu cấu trúc như sau:

```
<EVENT_INSTANCE>
  <EventType>type</EventType>
  <PostTime>date-time</PostTime>
  <SPID>spid</SPID>
  <ServerName>name</ServerName>
  <LoginName>name</LoginName>
  <UserName>name</UserName>
  <DatabaseName>name</DatabaseName>
  <SchemaName>name</SchemaName>
  <ObjectName>name</ObjectName>
  <ObjectType>type</ObjectType>
  <TSQLCommand>command</TSQLCommand>
</EVENT_INSTANCE>
```

Chẳng hạn, bạn khai báo DDL Trigger để kiểm soát mọi hành động làm thay đổi cấu trúc cơ sở dữ liệu hiện hành bằng cách trình bày thông tin của hàm EVENTDATA như ví dụ 12-24.

**Ví dụ 12-24: Khai báo tạo DDL Trigger với hàm EVENTDATA**

```
IF EXISTS (SELECT * FROM sys.triggers
           WHERE name = 'EVENTDATA_OF_DDL_TRIGGER_ON_DATABASE')
DROP TRIGGER EVENTDATA_OF_DDL_TRIGGER_ON_DATABASE
ON DATABASE
GO
CREATE TRIGGER EVENTDATA_OF_DDL_TRIGGER_ON_DATABASE
ON DATABASE
FOR DDL_DATABASE_LEVEL_EVENTS
AS
DECLARE @data xml
SET @data = EVENTDATA()
DECLARE @EventType nvarchar(100)
DECLARE @TSQLCommand nvarchar(2000)
SET @EventType =
@data.value('(/EVENT_INSTANCE/EventType)[1]',
'nvarchar(100)')
SET @TSQLCommand =
@data.value('(/EVENT_INSTANCE/TSQLCommand)[1]',
'nvarchar(2000)');
```

```

PRINT @EventType
PRINT @TSQLCommand
GO

```

Sau đó, bạn khai báo tạo rồi xóa bảng dữ liệu có tên TestEVENTDATA như ví dụ 12-25.

#### **Ví dụ 12-25: Khai báo kiểm tra hàm EVENTDATA**

```

CREATE TABLE TestEventData (a int, b int)
GO
DROP TABLE TestEventData ;
GO

```

Khi thực thi hai phát biểu trong ví dụ trên, bạn có thể tìm thấy kết quả trình bày như hình 12-24.

```

Messages

(1 row(s) affected)
CREATE_TABLE
CREATE TABLE TestEventData (a int, b int)

(1 row(s) affected)
DROP_TABLE
DROP TABLE TestEventData ;

Query executed successfully | MYSOLUTION (9.0 SP2) | MYSO

```

Hình 12-24: Hàm EVENTDATA.

### 3.6. Trigger kiểm soát cơ sở dữ liệu hiện hành

Để kiểm soát sự thay đổi cấu trúc cơ sở dữ liệu, trước tiên bạn tạo bảng dữ liệu để lưu lại thông tin về thời gian, người sử dụng, loại phát biểu SQL và nội dung phát biểu SQL như ví dụ 12-26.

#### **Ví dụ 12-26: Khai báo tạo bảng dữ liệu**

```

CREATE TABLE DatabaseObjectLog
(
    ActionTime smalldatetime DEFAULT GETDATE(),
    DB_User nvarchar(50),
    UserEvent nvarchar(100),
    TSQLStatement nvarchar(2000)
);
GO

```

Kể đến, bạn khai báo DDL Trigger trên cơ sở dữ liệu hiện hành có tên DatabaseChanged để kiểm soát mọi thay đổi của cơ sở dữ liệu dựa vào biến có DDL\_DATABASE\_LEVEL\_EVENTS như ví dụ 12-27.

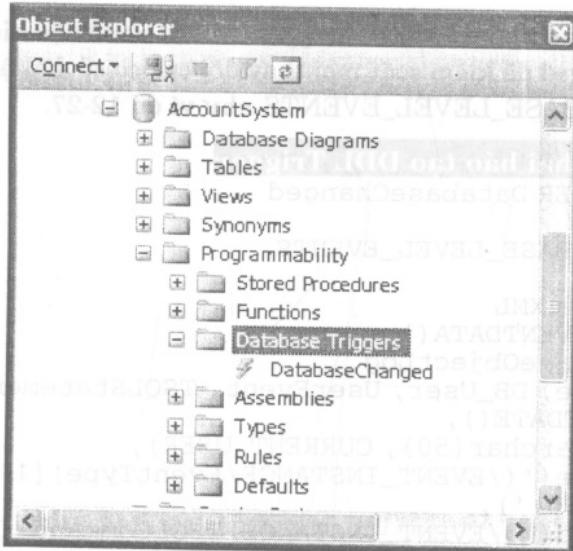
**Ví dụ 12-27: Khai báo tạo DDL Trigger**

```
CREATE TRIGGER DatabaseChanged
ON DATABASE
FOR DDL_DATABASE_LEVEL_EVENTS
AS
DECLARE @data XML
SET @data = EVENTDATA()
INSERT DatabaseObjectLog
(ActionTime, DB_User, UserEvent, TSQLStatement)
VALUES (GETDATE(),
CONVERT(nvarchar(50), CURRENT_USER),
@data.value('(/EVENT_INSTANCE/EventType)[1]',
'nvarchar(100)'), 
@data.value('(/EVENT_INSTANCE/TSQLCommand)[1]',
'nvarchar(2000)') );
GO
```

**Chú ý:** Biến cố DDL\_DATABASE\_LEVEL\_EVENTS là biến cố dạng cơ sở, trong đó nó bao hàm một số các biến cố con ứng với tầm kiểm soát đối tượng cơ sở dữ liệu như sau:

- DDL\_TRIGGER\_EVENTS: Dùng để kiểm soát sự thay đổi đối tượng Trigger trong cơ sở dữ liệu hiện hành.
  - DDL\_FUNCTION\_EVENTS: Dùng để kiểm soát sự thay đổi đối tượng Function trong cơ sở dữ liệu hiện hành.
  - DDL PROCEDURE EVENTS: Dùng để kiểm soát sự thay đổi đối tượng Stored Procedure trong cơ sở dữ liệu hiện hành.
  - DDL\_TABLE\_VIEW\_EVENTS: Dùng để kiểm soát sự thay đổi đối tượng Table và View trong cơ sở dữ liệu hiện hành.
  - DDL\_TYPE\_EVENTS: Dùng để kiểm soát sự thay đổi đối tượng Type trong cơ sở dữ liệu hiện hành.

Sau khi thực thi phát biểu CREATE TRIGGER trong ví dụ trên, bạn có thể tìm thấy tên Trigger này trong ngăn Database Triggers như hình 12-25.



Hình 12-25: Danh sách Trigger của cơ sở dữ liệu.

Để kiểm tra tính thi hành của Trigger này, trước tiên chúng ta thử kiểm tra mẫu tin trong bảng DatabaseObjectLog bằng phát biểu SELECT như hình 12-25-1.

```
SELECT * FROM DatabaseObjectLog;
GO
```

The Results pane shows the following output:

| ActionTime              | DB_User | UserEvent | TSQLStatement                         |
|-------------------------|---------|-----------|---------------------------------------|
| 2023-10-10 10:30:00.000 | sa      | 1         | CREATE TABLE TestTable (a int, b int) |
| 2023-10-10 10:30:00.000 | sa      | 1         | DROP TABLE TestTable ;                |

Hình 12-25-1: Danh sách mẫu tin trong bảng DatabaseObjectLog.

Kế đến, bạn khai báo phát biểu CREATE TABLE và DROP TABLE để tạo rồi xóa bảng TestTable như ví dụ 12-28.

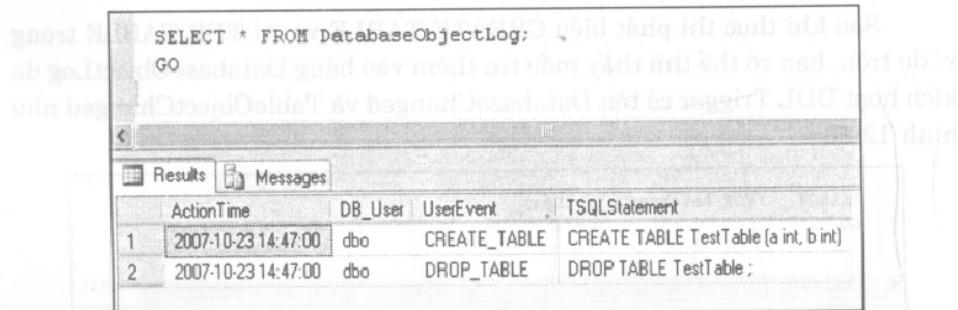
#### Ví dụ 12-28: Khai báo tạo và xóa bảng TestTable

```
CREATE TABLE TestTable (a int, b int)
GO
DROP TABLE TestTable ;
```

Bằng cách chạy phát biểu SELECT với bảng DatabaseObjectLog, bạn có thể tìm thấy danh sách mẫu tin trình bày như hình 12-25-2.

**Chương 12: Khám phá trigger**

221 M®



The screenshot shows a SQL Server Management Studio window with two tabs: 'Results' and 'Messages'. The 'Results' tab displays a table with four columns: ActionTime, DB\_User, UserEvent, and TSQLStatement. There are two rows of data:

| ActionTime          | DB_User | UserEvent    | TSQLStatement                         |
|---------------------|---------|--------------|---------------------------------------|
| 2007-10-23 14:47:00 | dbo     | CREATE_TABLE | CREATE TABLE TestTable (a int, b int) |
| 2007-10-23 14:47:00 | dbo     | DROP_TABLE   | DROP TABLE TestTable;                 |

**Hình 12-25-2: DDL Trigger đã thực thi.****3.6.1. Trigger cho hành động thay đổi cấu trúc Table**

Nếu bạn có nhu cầu kiểm soát sự thay đổi về cấu trúc của đối tượng cơ sở dữ liệu thì có thể khai báo DDL Trigger như ví dụ 12-29.

**Ví dụ 12-29: Khai báo tạo DDL Trigger**

```
CREATE TRIGGER TableObjectChanged
ON DATABASE
FOR DDL_TABLE_EVENTS
AS
DECLARE @data XML
SET @data = EVENTDATA()
INSERT DatabaseObjectLog
(ActionTime, DB_User, UserEvent, TSQLStatement)
VALUES (GETDATE(),
CONVERT(nvarchar(50), CURRENT_USER),
@data.value('/EVENT_INSTANCE/EventType')[1],
@data.value('/EVENT_INSTANCE/TSQLCommand')[1],
@data.value('/EVENT_INSTANCE/TSQLCommand')[1]);
```

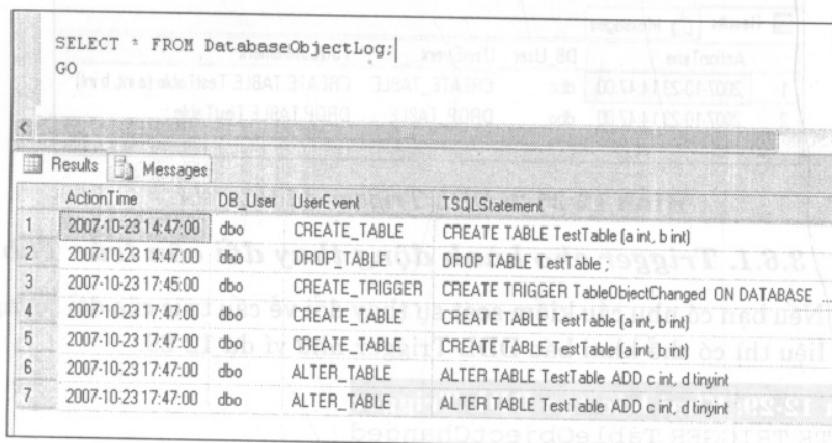
**Chú ý:** Mỗi khi bạn đã quyết định tạo các DML Trigger cho từng loại đối tượng cơ sở dữ liệu thì không nên khai báo DDL Trigger ứng với biến cờ DDL\_DATABASE\_LEVEL\_EVENTS.

Chẳng hạn, bạn khai báo để tạo bảng dữ liệu có tên TestTable bằng phát biểu CREATE TABLE rồi ngay sau đó sử dụng phát biểu ALTER TABLE để thay đổi cấu trúc của bảng này như ví dụ 12-30.

**Ví dụ 12-30: Khai báo tạo và thay đổi bảng dữ liệu**

```
CREATE TABLE TestTable (a int, b int)
GO
ALTER TABLE TestTable
ADD c int, d tinyint
GO
```

Sau khi thực thi phát biểu CREATE TABLE và ALTER TABLE trong ví dụ trên, bạn có thể tìm thấy mẫu tin thêm vào bảng DatabaseObjectLog do kích hoạt DDL Trigger có tên DatabaseChanged và TableObjectChanged như hình 12-26.



| ActionTime          | DB_User | UserEvent      | TSQLStatement                                     |
|---------------------|---------|----------------|---|
| 2007-10-23 14:47:00 | dbo     | CREATE_TABLE   | CREATE TABLE TestTable (a int, b int)             |
| 2007-10-23 14:47:00 | dbo     | DROP_TABLE     | DROP TABLE TestTable;                             |
| 2007-10-23 17:45:00 | dbo     | CREATE_TRIGGER | CREATE TRIGGER TableObjectChanged ON DATABASE ... |
| 2007-10-23 17:47:00 | dbo     | CREATE_TABLE   | CREATE TABLE TestTable (a int, b int)             |
| 2007-10-23 17:47:00 | dbo     | CREATE_TABLE   | CREATE TABLE TestTable (a int, b int)             |
| 2007-10-23 17:47:00 | dbo     | ALTER_TABLE    | ALTER TABLE TestTable ADD c int, d tinyint        |
| 2007-10-23 17:47:00 | dbo     | ALTER_TABLE    | ALTER TABLE TestTable ADD c int, d tinyint        |

Hình 12-26: Tạo DDL Trigger cho Table.

### 3.6.2. Trigger cho hành động thay đổi cấu trúc View

Tương tự như trường hợp kiểm soát đối tượng View, bạn có thể khai báo DDL Trigger như ví dụ 12-31.

#### Ví dụ 12-31: Khai báo tạo DDL Trigger cho đối tượng View

```
CREATE TRIGGER ViewObjectChanged
ON DATABASE
FOR DDL_VIEW_EVENTS
AS
DECLARE @data XML
SET @data = EVENTDATA()
INSERT DatabaseObjectLog
(ActionTime, DB_User, UserEvent, TSQLStatement)
VALUES (GETDATE(),
CONVERT(nvarchar(50), CURRENT_USER),
@data.value('/EVENT_INSTANCE/EventType')[1],
nvarchar(100)),
@data.value('/EVENT_INSTANCE/TSQLCommand')[1],
nvarchar(2000));
GO
```

Chẳng hạn, bạn khai báo để tạo bảng dữ liệu có tên TestTable bằng phát biểu CREATE VIEW rồi ngay sau đó sử dụng phát biểu ALTER VIEW để thay đổi cấu trúc của View ứng với bảng TestTable như ví dụ 12-32.

#### Ví dụ 12-32: Khai báo tao và thay đổi đối tượng View

### --Tạo cấu trúc View<sup>2</sup>

```
CREATE VIEW vwTestTable
```

AS

```
SELECT * FROM TestTable
```

GO

--Thay đổi cấu trúc View

```
ALTER VIEW vwTestTable
```

AS

```
SELECT * FROM TestTable
```

WHERE a=10

GO

Sau khi thực thi phát biểu CREATE VIEW và ALTER VIEW trong ví dụ trên, bạn có thể tìm thấy mẫu tin thêm vào bảng DatabaseObjectLog do kích hoạt DDL Trigger có tên ViewObjectChanged như hình 12-27.

| ActionTime          | DB_User | UserEvent   | TSQLStatement                                    |
|---------------------|---------|-------------|--|
| 2007-10-23 18:09:00 | dbo     | CREATE_VIEW | CREATE VIEW vwTestTable AS SELECT * FROM Test... |
| 2007-10-23 18:09:00 | dbo     | ALTER_VIEW  | ALTER VIEW vwTestTable AS SELECT * FROM Test...  |

**Hình 12-27:** Tao DDL Trigger cho View.

Lưu ý: Một số mẫu tin đã có trong bảng DatabaseObjectLog đã bị xóa trước đó, chính vì vậy hiện tại chúng chỉ chứa hai mẫu tin do chúng ta tạo và thay đổi cấu trúc View trong ví dụ 12-30.

Ngoài cách sử dụng DDL Trigger để kiểm soát thay đổi đối tượng cơ sở dữ liệu như Table hay View, bạn có thể sử dụng các biến cố đang cha (Parent).

Ví dụ, bạn có thể sử dụng biến cố DDL\_TABLE\_VIEW\_EVENTS thay vì sử dụng hai biến cố DDL\_TABLE\_EVENTS ứng với TABLE và DDL VIEW EVENTS ứng với VIEW.

### 3.7. Tao Trigger để kiểm soát Login User trên Server

Tương tự như trường khai báo DDL Trigger để kiểm soát thay đổi cấu trúc cơ sở dữ liệu hiện hành, bạn cũng có thể tạo các DDL Trigger để kiểm soát Server.

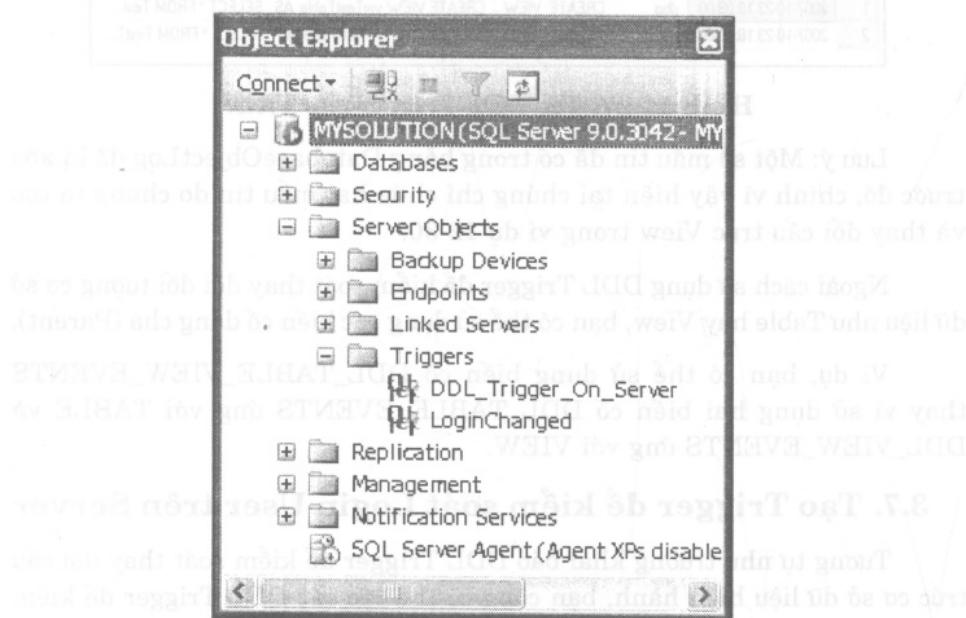
Chẳng hạn, bạn khai báo DDL Trigger để kiểm soát tài khoản đăng nhập SQL Server 2005 như ví dụ 12-33.

#### Ví dụ 12-33: Khai báo tạo DDL Trigger trên Server

```
CREATE TRIGGER LoginChanged
ON ALL SERVER
FOR DDL_LOGIN_EVENTS
AS
DECLARE @data XML
SET @data = EVENTDATA()
INSERT AccountSystem.dbo.DatabaseObjectLog
(ActionTime, DB_User, UserEvent, TSQLStatement)
VALUES (GETDATE(),
CONVERT(nvarchar(50), SYSTEM_USER),
@data.value('/EVENT_INSTANCE/EventType')[1],
nvarchar(100)),
@data.value('/EVENT_INSTANCE/TSQLCommand')[1],
nvarchar(2000));
GO
```

Chú ý: Do bảng DatabaseObjectLog thuộc cơ sở dữ liệu AccountSystem, nên bạn cần chỉ định AccountSystem.dbo trong phát biểu Insert.

Sau khi thực thi ví dụ trên, bạn có thể tìm thấy DDL Trigger vừa tạo ra trong ngăn Server Objects | Triggers như hình 12-28.



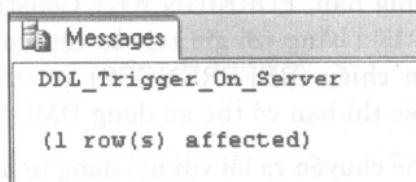
Hình 12-28: DDL Trigger dùng để quản lý Login User.

Để kiểm tra xem DDL Trigger trên có kích hoạt hay không, bạn có thể tạo Login User bằng phát biểu CREATE LOGIN như ví dụ 12-34.

**Ví dụ 12-34: Khai báo tao Login User**

```
CREATE LOGIN KhangPham
WITH PASSWORD = '11111111';
GO
```

Khi thực thi phát biểu CREATE LOGIN trong ví dụ trên, bạn sẽ tìm thấy kết quả như hình 12-29.



**Hình 12-29: Tạo Login User.**

Trong đó, chuỗi DDL\_Trigger\_On\_Server do DDL Trigger có tên DDL\_Trigger\_On\_Server tạo ra.

Bạn có thể tìm thấy mẫu tin mới thêm vào bảng DatabaseObjectLog bằng cách sử dụng phát biểu SELECT và kết quả trình bày như hình 12-30.

| ActionTime          | DB_User                   | UserEvent    |
|---------------------|---------------------------|--------------|
| 2007-10-23 18:09:00 | dbo                       | CREATE_VIEW  |
| 2007-10-23 18:09:00 | dbo                       | ALTER_VIEW   |
| 2007-10-24 08:29:00 | MYSOLUTION\Pham Huu Khang | CREATE_LOGIN |

**Hình 12-30: DDL Trigger theo dõi Login User.**

Chú ý: Bạn phải sử dụng hàm SYSTEM\_USER để lấy tài khoản đăng nhập vào SQL Server thay vì CURRENT\_USER như đã sử dụng trong các biến cố của cơ sở dữ liệu hiện hành.

## 4. SO SÁNH DML TRIGGER VÀ CONSTRAINT

Constraint và DML Trigger đều có lợi ích nhất định do nó tạo nên ứng với trường hợp đặc biệt. Lợi ích chính của DML Trigger là nó có thể chứa đựng mã T-SQL để xử lý quá trình logic phức tạp, do đó DML Trigger có thể hỗ trợ tất cả những chức năng của Constraint.

Tuy nhiên, trong những trường hợp cụ thể thì DML Trigger không thể làm tốt hơn Constraint. Chẳng hạn, trong ràng buộc thực thể thì được ràng buộc ở mức thấp nhất bằng cách sử dụng PRIMARY KEY và UNIQUE Constraint.

Tương tự như vậy, ràng buộc miền phải sử dụng CHECK Constraint và ràng buộc tham chiếu thì sử dụng FOREIGN KEY Constraint.

DML Trigger rất mạnh khi những yêu cầu ràng buộc không thể sử dụng Constraint. Chẳng hạn, FOREIGN KEY Constraint có thể kiểm tra giá trị của một cột dữ liệu bằng với giá trị của cột trong bảng khác trừ khi nó có khai báo tham chiếu (REFERENCES) với thuộc tính Cascading. Trong trường hợp khác thì bạn có thể sử dụng DML Trigger.

Constraint có thể chuyển ra lỗi với nội dung từ hệ thống lỗi chuẩn của SQL Server, nếu bạn có nhu cầu thay đổi nội dung chuỗi lỗi theo ngôn ngữ địa phương thì phải sử dụng DML Trigger thay vì Constraint.

Chú ý: Nếu Constraint đã tồn tại trong bảng dữ liệu, nó sẽ được kiểm tra sau khi INSTEAD OF Trigger kích hoạt và trước khi AFTER Trigger thực thi. Trong trường hợp Constraint xung đột thì INSTEAD OF Trigger sẽ hủy bỏ và AFTER Trigger không kích hoạt.

## 5. KẾT CHƯƠNG

Trong chương này, chúng ta đã tập trung tìm hiểu cách tạo DML Trigger để kiểm soát mọi sự thay đổi dữ liệu trong Table hay View bằng cách hành động của người sử dụng ứng với ba phát biểu INSERT, DELETE và UPDATE.

Tương tự như vậy, bạn cũng tham khảo chi tiết các khai báo và sử dụng DDL Trigger dùng để kiểm soát sự thay đổi cấu trúc cơ sở dữ liệu hiện hành và những sự thay đổi khác trong SQL Server 2005.

Bạn sẽ tiếp tục tìm hiểu cách khai báo và sử dụng ba đối tượng RULE, DEFAULT và TYPE.

## Chương 13:

# ĐỐI TƯỢNG DEFAULT, RULE, TYPE

### Tóm tắt chương 13

Trong chương trước bạn đã tìm hiểu cách khai báo và sử dụng thủ tục nội tại, trong chương này chúng ta tiếp tục tìm hiểu đối tượng DEFAULT và RULE.

Ngoài ra, trong chương này chúng ta cũng tìm hiểu cách sử dụng TYPE khi khai báo kiểu dữ liệu cho cột trong khi thiết kế đối tượng TABLE.

#### Các vấn đề chính sẽ được đề cập:

- ✓ Đối tượng DEFAULT.
- ✓ Đối tượng RULE.
- ✓ Đối tượng TYPE.

## 1. ĐỐI TƯỢNG DEFAULT

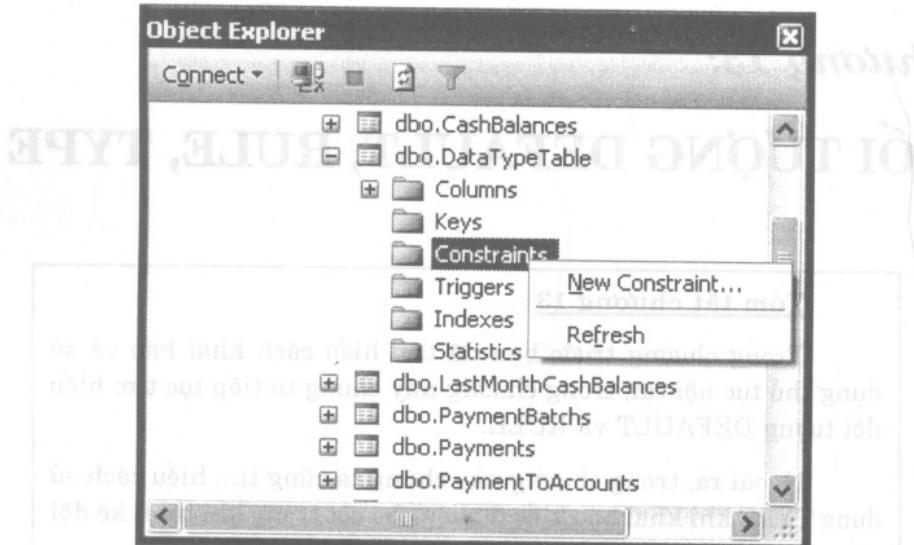
Trong khi bạn thiết lập bảng dữ liệu, một số cột dữ liệu có thể cần khai báo giá trị mặc định nào đó nhằm bảo đảm dữ liệu do người sử dụng nhập vào đúng như mong đợi, chúng ta cần xây dựng quy tắc cho giá trị mặc định này bằng cách sử dụng đối tượng DEFAULT trong SQL Server 2005.

Chú ý: Bạn có thể tìm thấy các ví dụ trình bày của chương này nằm trong tập tin có tên DefaultAndRule.sql.

Lưu ý: Để tạo giá trị mặc định cho cột dữ liệu, bạn có thể sử dụng từ khóa DEFAULT trong phát biểu CREATE TABLE.

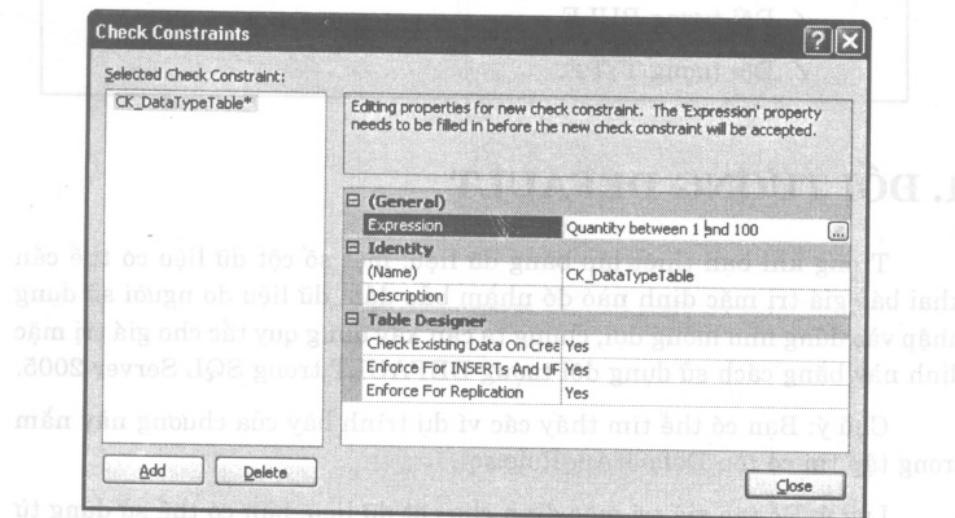
Ví dụ, bạn có thể khai báo ràng buộc trong khi thiết kế cột dữ liệu bằng cách chọn vào ngăn Constraints (hoặc chọn vào cột dữ liệu rồi R-Click | Check Constraints) rồi R-Click, cửa sổ xuất hiện như hình 13-1.

## Chương 13: Đối tượng DEFAULT, RULE, TYPE



Hình 13-1: Tạo mới Constraints.

Bằng cách chọn New Constraint, cửa sổ kế tiếp xuất hiện và bạn có thể khai báo tương tự như hình 13-2.



Hình 13-2: Khai báo Constraint.

Tuy nhiên, ngoài việc chỉ định giá trị mặc định và ràng buộc bằng cách khai báo cột dữ liệu vào bảng trong lúc thiết kế, bạn có thể định nghĩa giá trị mặc định để có thể dùng chung cho nhiều trường hợp ứng với các cột dữ liệu khác nhau.

**Chương 13: Đối tượng DEFAULT, RULE, TYPE**

229 M®

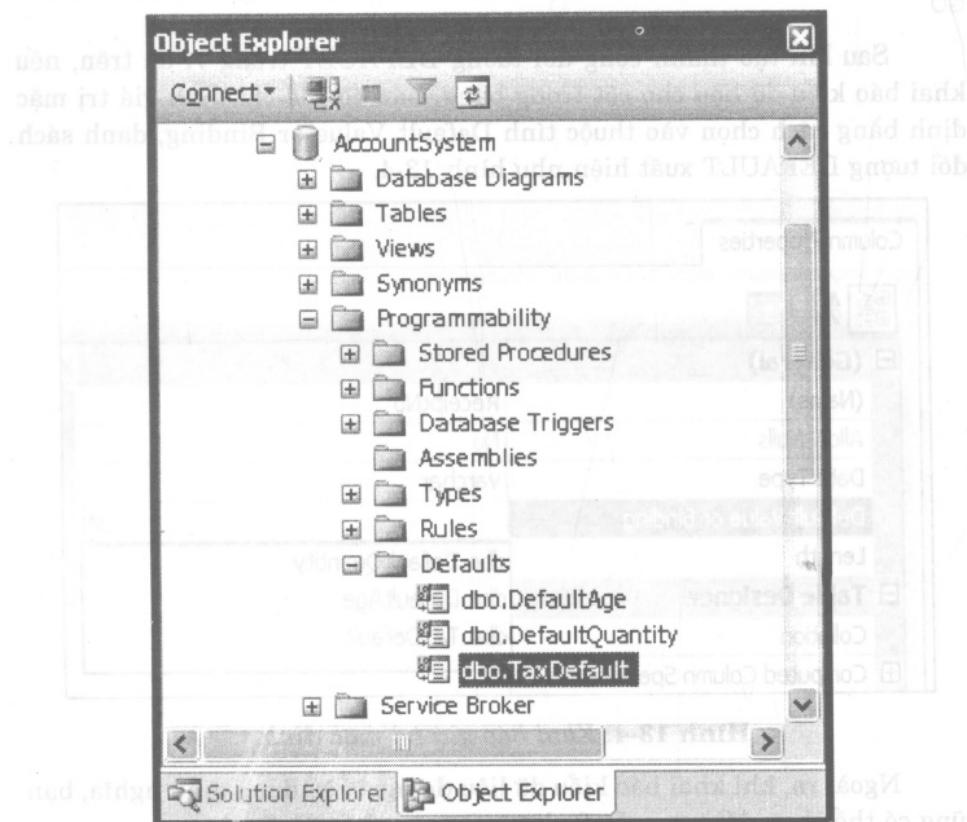
```
CREATE DEFAULT [ schema_name . ] default_name
AS constant_expression [ ; ]
```

Để làm điều này, bạn khai báo tạo đối tượng DEFAULT có giá trị là 10% ứng với 0.1 bằng cách sử dụng phát biểu CREATE DEFAULT như ví dụ 13-1.

**Ví dụ 13-1: Khai báo tạo đối tượng Default**

```
CREATE DEFAULT TaxDefault AS 0.1
GO
```

Sau khi thực thi phát biểu trong ví dụ trên, bạn có thể tìm thấy đối tượng DEFAULT xuất hiện trong ngăn Defaults như hình 13-3.



**Hình 13-3: Danh sách đối tượng DEFAULT.**

Một đối tượng DEFAULT có thể chứa đựng hằng (giá trị cụ thể), hàm (SESSION\_USER, CURRENT\_USER, GETDATE(), ...) hay giá trị NULL. Tuy nhiên, hằng trong đối tượng DEFAULT không cho phép tham chiếu đến cột dữ liệu trong bảng hay đối tượng VIEW khác.

Chú ý: Một cột không cho phép NULL (NOT NULL) và cũng không khai báo giá trị mặc định thì Database Engine sẽ trả về lỗi nếu cột đó không có dữ liệu mỗi khi thêm mới mẫu tin vào bảng.

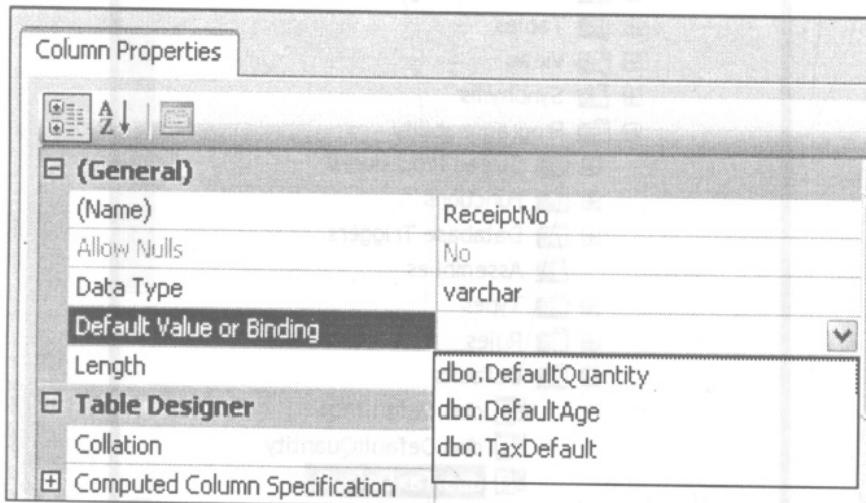
Ngoài ra, đối tượng DEFAULT không thể tạo trên cột dữ liệu có kiểu dữ liệu là timestamp hay thuộc tính IDENTITY (số tự động).

Giả sử, chúng ta đã tạo đối tượng Default với tên gọi là DefaultQuantity có giá trị mặc định là 1.

#### Ví dụ 13-2: Khai báo đối tượng DEFAULT

```
CREATE DEFAULT DefaultQuantity AS 1
GO
```

Sau khi tạo thành công đối tượng DEFAULT trong ví dụ trên, nếu khai báo kiểu dữ liệu cho cột trong bảng, bạn có thể chỉ định giá trị mặc định bằng cách chọn vào thuộc tính Default Value or Binding, danh sách đối tượng DEFAULT xuất hiện như hình 13-4.



Hình 13-4: Khai báo giá trị mặc định.

Ngoài ra, khi khai báo kiểu dữ liệu do người sử dụng định nghĩa, bạn cũng có thể chọn đối tượng Default này trong phần Default.

Tương tự như các đối tượng khác trong cơ sở dữ liệu, bạn có thể xóa đối tượng DEFAULT trong cơ sở dữ liệu bằng cách sử dụng phát biểu DROP DEFAULT như sau:

```
DROP DEFAULT { [ schema_name . ] default_name } [ ,...n ]
[ ; ]
GO
```

**Chú ý:** Một khi có nhu cầu thay đổi cấu trúc của đối tượng DEFAULT thì bạn xóa đối tượng DEFAULT đang tồn tại và tạo mới trở lại, bởi vì SQL Server 2005 không cung cấp phát biểu ALTER DEFAULT như các phát biểu hành động khác.

Tóm lại, mục đích chính của đối tượng DEFAULT là cho phép khai báo đối tượng chứa đựng giá trị mặc định mà có thể áp dụng cho nhiều cột dữ liệu trong bảng trong góc nhìn thiết kế.

## 2. ĐỐI TƯỢNG RULE

RULE có chức năng tương tự như ràng buộc CHECK. Tuy nhiên, Microsoft khuyên bạn nên sử dụng ràng buộc CHECK thay vì RULE, đó là chuẩn để kiểm soát giá trị trong cột dữ liệu của bảng.

Ngoài ra, ràng buộc CHECK cũng khai báo ngắn gọn hơn khai báo RULE do bạn có thể áp dụng cho cột dữ liệu chỉ một RULE trong khi có thể áp dụng nhiều ràng buộc CHECK.

Bạn sử dụng ràng buộc CHECK như một phần khai báo trong phát biểu CREATE TABLE, trong khi đó RULE tạo ra để áp dụng cho từng đối tượng và áp dụng cho từng cột dữ liệu.

Trong khi đối tượng DEFAULT cho phép bạn tạo giá trị mặc định dùng chung cho nhiều trường hợp của nhiều cột dữ liệu khác nhau khi thiết kế thì đối tượng RULE cho phép bạn định nghĩa quy tắc so sánh dùng chung cho nhiều cột dữ liệu trong gốc nhìn thiết kế.

```
chung cho phép cột dữ liệu trong gốc minh thiết  
CREATE RULE [ schema_name . ] rule_name  
AS condition_expression  
[ ; ]
```

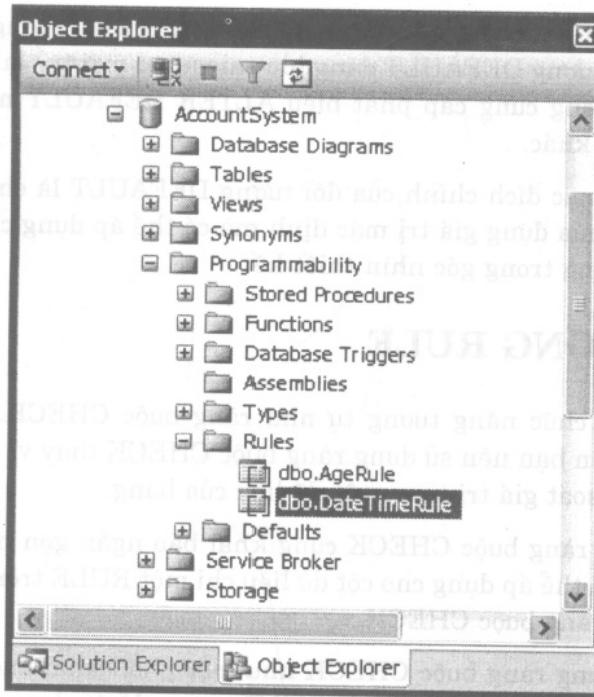
Để có thể sử dụng chung quy tắc so sánh thời gian cho nhiều trường hợp khác, bạn nên tạo đối tượng Rule như ví dụ 13-3.

### Ví dụ 13-3: Khai báo Rule

```
CREATE RULE DateTimeRule  
AS  
    @DateTimeRule >=getdate()  
Go
```

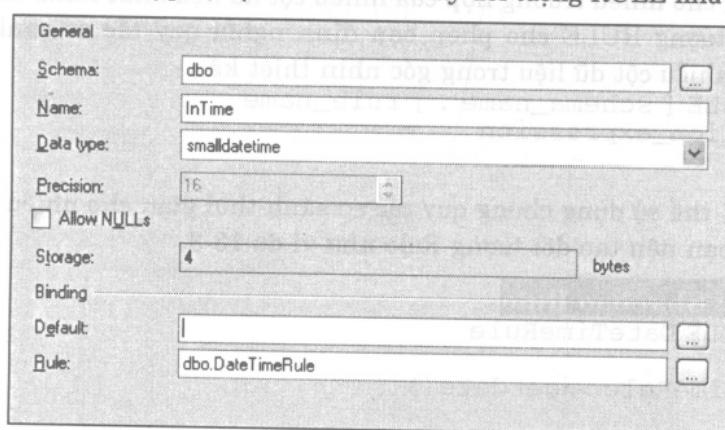
Sau khi thực thi phát biểu CREATE RULE trong ví dụ trên, bạn có thể tìm thấy đối tượng DateTimeRule xuất hiện trong ngăn RULES như hình 13-5.

### Chương 13: Đối tượng DEFAULT, RULE, TYPE



**Hình 13-5: Danh sách RULE.**

Mỗi khi khai báo kiểu dữ liệu do người sử dụng định nghĩa trong phần User-Defined Data Types, bạn có thể tìm thấy đối tượng RULE như hình 13-6.



**Hình 13-6: Sử dụng Rule.**

Tóm lại, khi bạn khai báo kiểu dữ liệu dựa trên kiểu dữ liệu của SQL Server 2005, bạn có thể sử dụng đối tượng RULE này để giới hạn giá trị thỏa với điều kiện khai báo trong RULE.

### 3. ĐỐI TƯỢNG TYPE

SQL Server 2005, cung cấp nhiều loại kiểu dữ liệu cho phép bạn khai báo cột trong bảng chứa dữ liệu tương ứng với thế giới thực. Tuy nhiên, nếu bạn muốn tự định nghĩa ra kiểu dữ liệu dựa trên những kiểu dữ liệu của SQL Server 2005 đang có những ràng buộc hay điều kiện khác nhằm đáp ứng nhu cầu thực tế thì sử dụng TYPE.

Để tạo đối tượng TYPE, bạn sử dụng cú pháp như sau:

```
CREATE TYPE [ schema_name. ] type_name
{
    FROM base_type
    [ ( precision [ , scale ] ) ]
    [ NULL | NOT NULL ]
    | EXTERNAL NAME assembly_name [ .class_name ]
} [ ; ]
```

Trong trường hợp xóa đối tượng TYPE, bạn có thể sử dụng cú pháp như sau:

```
DROP TYPE [ schema_name. ] type_name [ ; ]
```

Chẳng hạn, bạn tạo ra kiểu dữ liệu dựa trên kiểu dữ liệu TINYINT nhưng có giá trị trong khoảng từ 10 đến 100 bằng phát biểu CREATE TYPE như ví dụ 13-4.

#### Ví dụ 13-4: Khai báo RULE

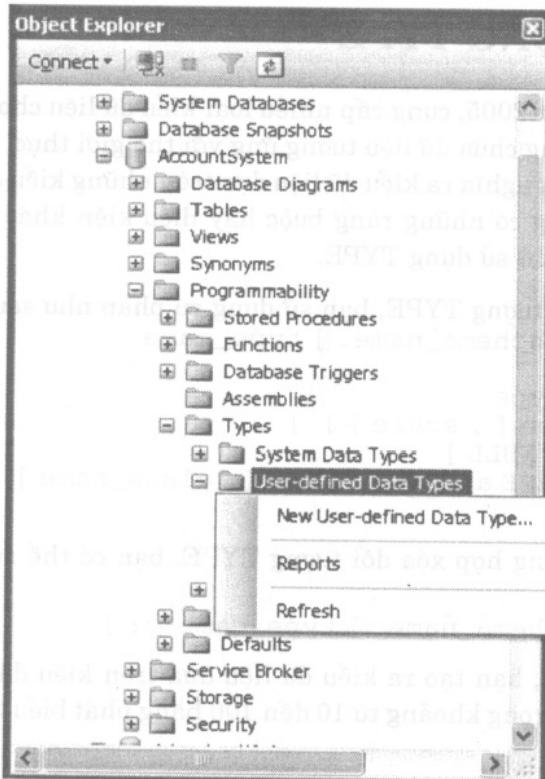
```
CREATE TYPE [dbo]. [AgeType]
FROM [tinyint] NOT NULL
GO
```

Ngoài ra, bạn cũng có thể tạo đối tượng TYPE bằng MS thay vì sử dụng phát biểu CREATE. Chẳng hạn, để tạo đối tượng TYPE ứng với kiểu thời gian áp dụng cho các nghiệp vụ kế toán có ngày phát sinh nhỏ hơn hay bằng ngày hiện hành, bạn khai báo đối tượng RULE để giới hạn ngày phát sinh nhỏ hơn hay bằng ngày hiện hành ứng với cú pháp như ví dụ 13-5.

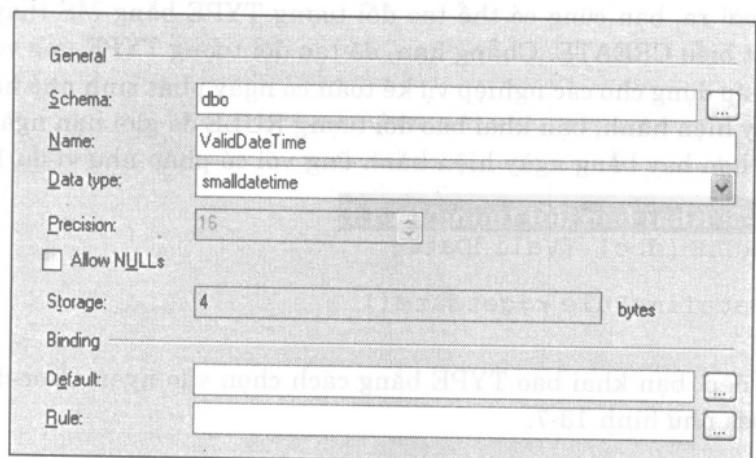
#### Ví dụ 13-5: Khai báo đối tượng RULE

```
CREATE RULE [dbo]. [ValidDate]
AS
@ValidDateTimeRule <=getdate()
GO
```

Ké đến, bạn khai báo TYPE bằng cách chọn vào ngăn User-defined Data Types như hình 13-7.

**Chương 13: Đối tượng DEFAULT, RULE, TYPE****Hình 13-7: Khai báo đối tượng TYPE.**

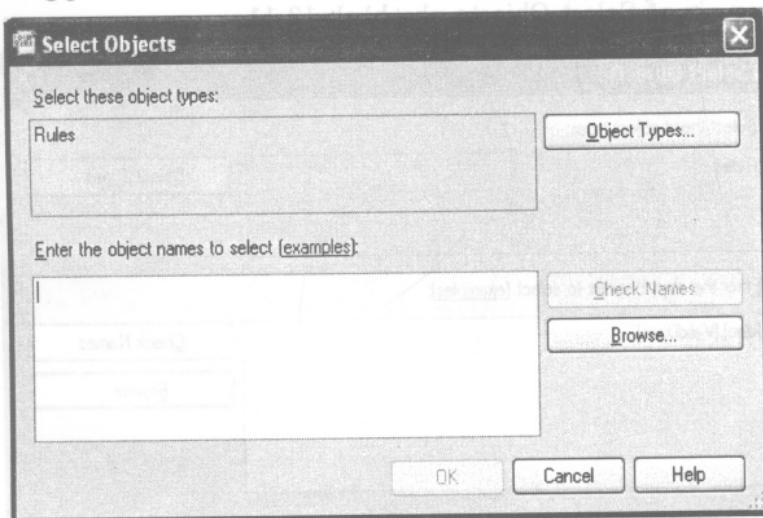
Chọn vào New User-defined Data Type, cửa sổ xuất hiện rồi bạn đặt tên và chọn kiểu dữ liệu có sẵn như hình 13-8.

**Hình 13-8: Tạo mới TYPE.**

## Chương 13: Đối tượng DEFAULT, RULE, TYPE

235 ®

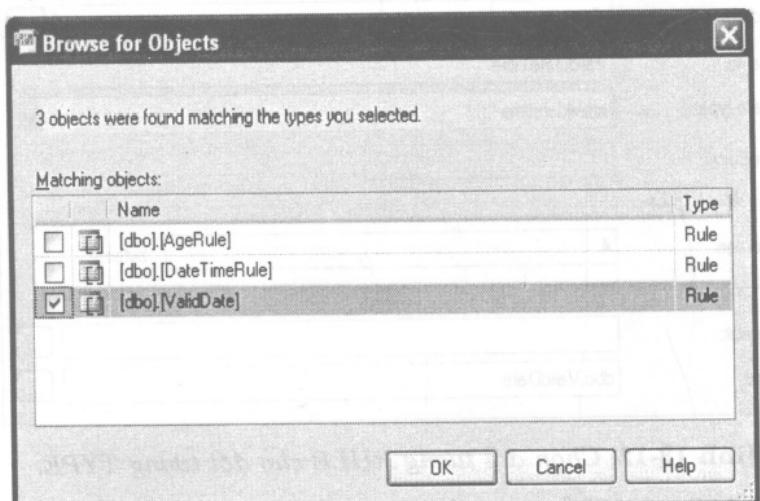
Trong phần Rule, bạn chọn vào nút ..., cửa sổ xuất hiện như hình 13-9.



**Hình 13-9:** Chọn đối tượng RULE.

Nếu bạn nhớ tên thì có thể gõ vào phần “Enter the object names to select (examples)”. Tuy nhiên, bạn cũng có thể chọn vào nút Browse để chọn RULE đang có trong cơ sở dữ liệu rồi chọn đối tượng RULE cần áp dụng cho đối tượng TYPE như hình 13-10.

**Chú ý:** Bạn có thể chọn nhiều đối tượng RULE nhưng chỉ một RULE được áp dụng cho đối tượng TYPE.

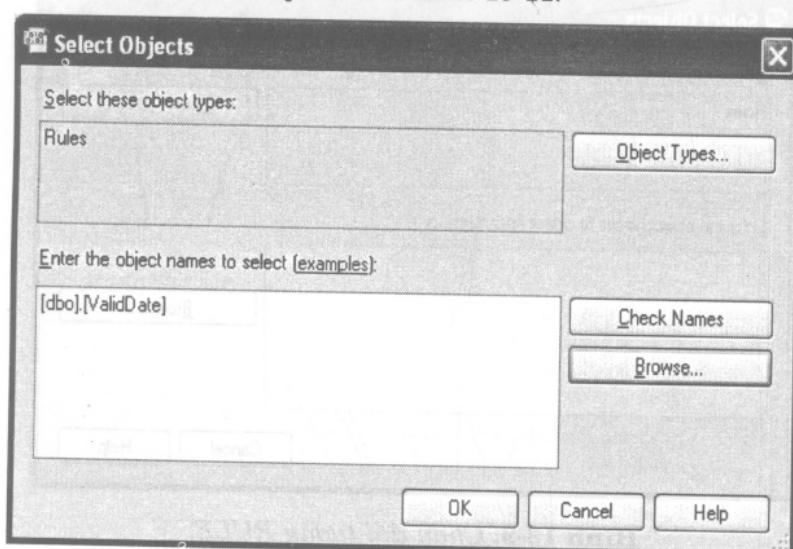


**Hình 13-10:** Danh sách đối tượng RULE.

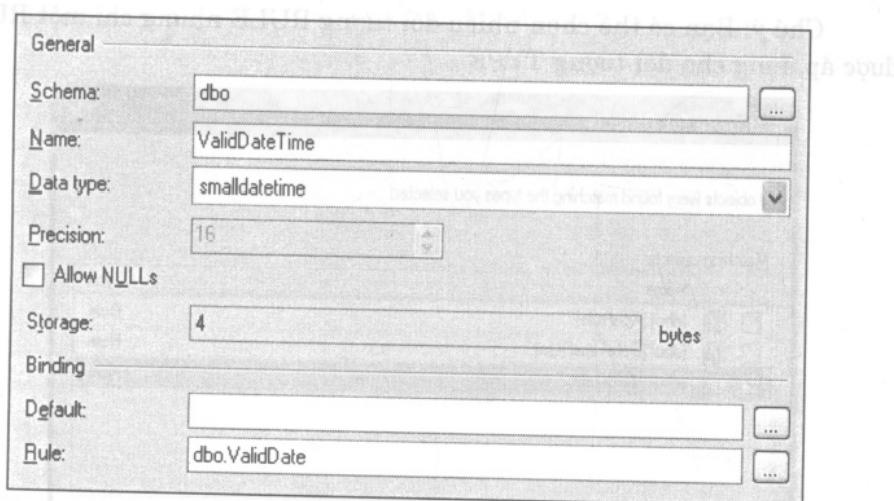
M® 236

**Chương 13: Đối tượng DEFAULT, RULE, TYPE**

Nhấn nút OK, bạn có thể tìm thấy đối tượng RULE được chọn xuất hiện trong cửa sổ Select Objects như hình 13-11.

**Hình 13-11: Chọn đối tượng RULE.**

Sau khi nhấn nút OK, bạn có thể tìm thấy danh sách đối tượng RULE khai báo trong phần Rule của cửa sổ New User-defined Data Type như hình 13-12.

**Hình 13-12: Chọn đối tượng RULE cho đối tượng TYPE.**

Chú ý: Bạn có thể tìm thấy phát biểu CREATE TYPE ứng với đối tượng TYPE có tên ValidDateTime như ví dụ 13-6.

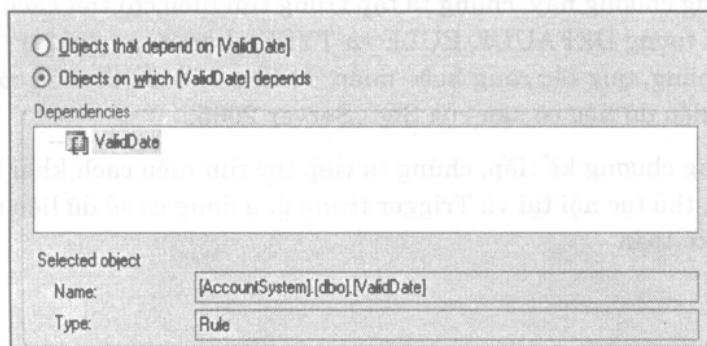
**Chương 13: Đối tượng DEFAULT, RULE, TYPE**

237 M®

**Ví dụ 13-6: Cú pháp tạo đối tượng TYPE**

```
USE [AccountSystem]
GO
/***** Object: UserDefinedDataType [dbo].[ValidDateTime]
[dbo].[ValidDateTime]
Script Date: 10/28/2007 16:37:09 *****/
CREATE TYPE [dbo].[ValidDateTime] 
FROM [smalldatetime] NOT NULL
```

Bạn không tìm thấy đối tượng RULE áp dụng cho đối tượng TYPE này, chính vì vậy bạn có thể tìm kiếm các đối tượng khác có liên quan bằng cách chọn vào tên đối tượng TYPE là ValidDateTime và R-Click rồi chọn vào View Dependencies, cửa sổ Object Dependencies xuất hiện như hình 13-13.



**Hình 13-13:** Các đối tượng liên quan đến đối tượng TYPE.

Để sử dụng đối tượng TYPE vừa tạo trên, bạn có thể trở lại khung nhìn thiết kế đối tượng TABLE. Khi chọn kiểu dữ liệu trong danh sách thả xuống, bạn có thể tìm thấy TYPE có tên ValidDateTime như hình 13-14.

| Column Name              | Data Type                   | Allow Nulls                         |
|--------------------------|-----------------------------|-------------------------------------|
| ExportNo                 | varchar(10)                 | <input type="checkbox"/>            |
| ExportBatchNo            | varchar(10)                 | <input checked="" type="checkbox"/> |
| ExportDate               | smalldatetime               | <input checked="" type="checkbox"/> |
| ExportTypeID             | varbinary(50)               | <input type="checkbox"/>            |
| CustomerID               | varbinary(MAX)              | <input type="checkbox"/>            |
| DescriptionInVietname... | varchar(50)                 | <input type="checkbox"/>            |
| DescriptionInSecondL...  | varchar(MAX)                | <input type="checkbox"/>            |
| ShowDescriptionInViet... | xml                         | <input checked="" type="checkbox"/> |
| ExportDiscontinued       | AgeType:tinyint             | <input checked="" type="checkbox"/> |
| EntryDate                | InTime:smalldatetime        | <input checked="" type="checkbox"/> |
| UserName                 | ValidDateTime:smalldatetime | <input checked="" type="checkbox"/> |
|                          | smalldatetime               | <input type="checkbox"/>            |
|                          | varchar(10)                 | <input type="checkbox"/>            |

**Hình 13-14:** Sử dụng đối tượng TYPE.

**Chú ý:** Nếu bạn sử dụng phát biểu CREATE TABLE thì khai báo kiểu dữ liệu là ValidDateTime tương tự như ví dụ 13-7.

**Ví dụ 13-7: Khai báo sử dụng kiểu đối tượng TYPE**

## CREATE TABLE TypeObject

```
(          PRIMARY KEY  
    ColumnWithTYPE ValidDateTime  
)  
GO
```

#### 4. KẾT CHƯƠNG

Trong chương này, chúng ta tập trung tìm hiểu chi tiết cách tạo và sử dụng 3 đối tượng DEFAULT, RULE và TYPE nhằm tạo ra giá trị mặc định sử dụng chung, quy tắc ràng buộc miền dữ liệu và kiểu dữ liệu có giới hạn dựa trên kiểu dữ liệu có sẵn của SQL Server 2005.

Trong chương kế tiếp, chúng ta tiếp tục tìm hiểu cách khai báo và sử dụng hàm, thủ tục nội tại và Trigger trong ứng dụng cơ sở dữ liệu đính kèm dùng cho kế toán.

**Ứng dụng:** Sử dụng thủ tục nội tại trong ứng dụng

239 M<sup>®</sup>

### **Ứng dụng:**

# SỬ DỤNG THỦ TỤC NỘI TẠI TRONG ỨNG DỤNG

## Tóm tắt phần ứng dụng

Trong phần ứng dụng của tập: Lập trình T\_SQL, chúng ta đã tìm hiểu cách sử dụng phát biểu SQL để thực hiện các thao tác nhằm kết xuất dữ liệu theo yêu cầu của phần kế toán thu và chi, các khoản phải thu, các khoản phải chi và tình hình tồn kho.

Do hầu hết giao tiếp từ ứng dụng được viết bằng ngôn ngữ lập trình .NET, bạn cần khai báo hoàn toàn bằng thủ tục nội tại cho 4 hành động chính là truy vấn, thêm, cập nhật và xóa thay vì sử dụng phát biểu SQL. Chính vì vậy, trong phần này chúng ta tiếp tục sử dụng hàm, thủ tục nội tại và Trigger để tiếp tục thực hiện các thao tác nhằm đáp ứng cho nhu cầu báo cáo kế toán.

Các vấn đề chính sẽ được đề cập:

- ✓ Phần kế toán công nợ phải thu.
  - ✓ Phần kế toán công nợ phải trả.
  - ✓ Phần kế toán tổng hợp.
  - ✓ Phần kế toán xuất nhập tồn.

## 1. PHẦN KẾ TOÁN CÔNG NỢ PHẢI THU

Khi làm việc với chức năng các khoản phải thu, chúng ta có các bảng dữ liệu liên quan là Customers, SalesInvoiceBatchs, SalesInvoiceTypes, SalesInvoices và SalesInvoiceDetails.

Chú ý: Bạn có thể tìm thấy các ví dụ trình bày của chương này nằm trong tập tin có tên `StoredProcedureForAR.sql`.

M® 240

**Ứng dụng:** Sử dụng thủ tục nội tại trong ứng dụng

### 1.1. Bảng Customers

Đối với hành động truy vấn dữ liệu, chúng ta cần khai báo thủ tục nội tại như ví dụ UD-1.

#### Ví dụ UD-1: Khai báo thủ tục nội tại truy vấn

```
CREATE PROC udsViewCustomers
```

```
    @CustomerId CHAR(5)
```

```
AS
```

```
    SELECT * FROM Customers
```

```
    WHERE
```

```
        CustomerId = CASE @CustomerId
```

```
            WHEN '' THEN CustomerId
```

```
            ELSE @CustomerId END
```

```
GO
```

Khi thực thi thủ tục nội tại trên để liệt kê danh sách khách hàng, bạn cần sử dụng cú pháp như ví dụ UD-1-1.

#### Ví dụ UD-1-1: Khai báo gọi thủ tục nội tại

```
udsViewCustomers
```

```
GO
```

Khi thực thi thủ tục trên, bạn có thể tìm thấy danh sách mẫu tin như hình UD-1.

| CustomerID | CompanyNameInVietnamese                       | CompanyNameInSecondLanguage |
|------------|---|-----------------------------|
| 1 A0001    | Công ty Trách Nhiệm Hữu Hạn Macrosoft Vietnam | Macrosoft Vietnam Co., Ltd. |
| 2 A0002    | Công ty Trách Nhiệm Hữu Hạn IBM Vietnam       | IBM Vietnam Co., Ltd.       |
| 3 A0003    | Công ty Trách Nhiệm Hữu Hạn Kodaka Vietnam    | Kodaka Vietnam Co., Ltd.    |
| 4 A0004    | Công ty Trách Nhiệm Hữu Hạn E-Google Vietnam  | E-Google Vietnam Co., Ltd.  |
| 5 A0005    | Công ty Cổ phần Suzumi Vietnam                | Suzumi Vietnam Corp.        |
| 6 A0006    | Tập đoàn UCIA USA                             | UCIA USA                    |
| 7 A0007    | Công ty Đa quốc gia UFCIA                     | UFCIA Corp.                 |
| 8 A0008    | Công ty Cổ phần RecruitVietnam                | RecruitVietnam Corp.        |
| 9 A0009    | Trung tâm giáo dục Vietnam                    | Vietnam Education Center    |
| 10 A0010   | Công ty Trách Nhiệm Hữu Hạn Hot Getways       | Hot Getways Comnay          |

**Hình UD-1: Danh sách khách hàng.**

Trong trường hợp muốn liệt kê một mẫu tin ứng với một khách hàng được chọn, bạn khai báo gọi thủ tục nội tại như ví dụ UD-1-2.

#### Ví dụ UD-1-2: Khai báo liệt kê một khách hàng

```
udsViewCustomers @CustomerId = 'A0002'
```

```
GO
```

**Ứng dụng:** Sử dụng thủ tục nội tại trong ứng dụng

241

Khi thực thi thủ tục trên, bạn có thể tìm thấy thông tin của khách hàng như hình UD-1-1.

| Results    |   | Messages                    |  |
|------------|---|-----------------------------|--|
| CustomerID | CompanyNameInVietnamese                 | CompanyNameInSecondLanguage |  |
| 1 A002     | Công ty Trách Nhiệm Hữu Hạn IBN Vietnam | IBN Vietnam Co., Ltd.       |  |

**Hình UD-1-1: Thông tin một khách hàng.**

Ứng với trường hợp xóa mẫu tin trong bảng Customers, bạn có thể khai báo thủ tục có tên udsDeleteCustomers như ví dụ UD-1-3.

**Ví dụ UD-1-3: Khai báo xóa mẫu tin**

```
CREATE PROC udsDeleteCustomers
    @CustomerId CHAR(5)
AS
    DELETE FROM Customers
    WHERE
        CustomerId = CASE @CustomerId
            WHEN '' THEN CustomerId
            ELSE @CustomerId END
GO
```

Bạn có thể gọi thủ tục nội tại có tên udsDeleteCustomers với hai trường hợp, trường xóa tất cả mẫu tin trong bảng bạn khai báo như ví dụ UD-1-4.

**Ví dụ UD-1-4: Khai báo xóa tất cả mẫu tin**

```
udsDeleteCustomers ''
GO
```

Trong trường hợp xóa một mẫu tin, bạn khai báo gọi thủ tục udsDeleteCustomers như ví dụ UD-1-5.

**Ví dụ UD-1-5: Khai báo xóa một mẫu tin**

```
udsDeleteCustomers 'A0010'
GO
```

Đối với trường hợp thêm mới hay cập nhật, bạn có thể khai báo thủ tục nội tại dùng chung cho hai trường hợp này dựa vào mã khách hàng như ví dụ UD-1-6.

**Ví dụ UD-1-6: Khai báo thêm và cập nhật**

```
CREATE PROC udsInsUpdCustomers
    @Flag BIT,
    @CustomerID char(5),
    @CompanyNameInVietnamese nvarchar(50),
    @CompanyNameInSecondLanguage varchar(50),
```

**M<sup>®</sup> 242****Ứng dụng: Sử dụng thủ tục nội tại trong ứng dụng**

```

@ContactName nvarchar(50) ,
@ContactTitle nvarchar(50) ,
@Address nvarchar(100) ,
@ProvinceID char(3) ,
@Telephone varchar(20) ,
@FaxNumber varchar(10) ,
@CustomerTypeID char(3) ,
@EmailAddress varchar(50) ,
@DueDate smalldatetime ,
@Discontinued bit ,
@MaxDebt decimal(18, 2)
AS
    -- Trường hợp thêm mới
    IF @Flag = 0
        INSERT INTO Customers
        VALUES (@CustomerID, @CompanyNameInVietnamese,
        @CompanyNameInSecondLanguage, 1, @ContactName,
        @ContactTitle, @Address, @ProvinceID,
        @Telephone, @FaxNumber, @CustomerTypeID,
        @EmailAddress, @DueDate, 0, @MaxDebt)
    ELSE
        UPDATE Customers
        SET CompanyNameInVietnamese =
        @CompanyNameInVietnamese,
        CompanyNameInSecondLanguage =
        @CompanyNameInSecondLanguage,
        ContactName = @ContactName,
        ContactTitle = @ContactTitle,
        Address = @Address,
        ProvinceID= @ProvinceID,
        Telephone = @Telephone,
        FaxNumber = @FaxNumber,
        CustomerTypeID = @CustomerTypeID,
        EmailAddress = @EmailAddress,
        Discontinued = @Discontinued,
        MaxDebt = @MaxDebt
        WHERE CustomerID = @CustomerID
GO

```

**1.2. Bảng SalesInvoiceTypes**

Tương tự như bảng Customers, đối với hành động truy vấn dữ liệu cho bảng SalesInvoiceTypes, chúng ta cần khai báo thủ tục nội tại như ví dụ UD-2.

**Ví dụ UD-2: Khai báo thủ tục nội tại truy vấn**

```

CREATE PROC udsViewSalesInvoiceTypes
    @InvoiceTypeId CHAR(3)
AS
    SELECT * FROM SalesInvoiceTypes

```

**Ứng dụng:** Sử dụng thủ tục nội tại trong ứng dụng

243

```

WHERE
InvoiceTypeID = CASE @InvoiceTypeID
    WHEN '' THEN InvoiceTypeID
    ELSE @InvoiceTypeID END
GO

```

Nếu thực thi thủ tục nội tại trên để liệt kê danh sách loại hóa đơn bán hàng, bạn cần sử dụng cú pháp như ví dụ UD-2-1.

**Ví dụ UD-2-1: Khai báo gọi thủ tục nội tại**

```

udsViewSalesInvoiceTypes ''
GO

```

Khi thực thi thủ tục trên, bạn có thể tìm thấy danh sách mẫu tin như hình UD-2.

| InvoiceTypeID | InvoiceTypeNameInVietnamese        |
|---------------|------------------------------------|
| 1             | Hoá đơn bán hàng                   |
| 2             | Hoá đơn bán hàng là Bán thành phẩm |
| 3             | Hoá đơn bán hàng khuyến mãi        |

**Hình UD-2: Loại hóa đơn bán hàng.**

Đối với trường hợp muốn liệt kê một mẫu tin ứng với một mã loại hóa đơn bán hàng được chọn, bạn khai báo gọi thủ tục nội tại như ví dụ UD-2-2.

**Ví dụ UD-2-2: Khai báo liệt kê một loại hóa đơn bán hàng**

```

udsViewSalesInvoiceTypes 'SI2'
GO

```

Khi thực thi thủ tục trên, bạn có thể tìm thấy thông tin của loại hóa đơn bán hàng như hình UD-2-1.

| InvoiceTypeID | InvoiceTypeNameInVietnamese        |
|---------------|------------------------------------|
| 1             | Hoá đơn bán hàng là Bán thành phẩm |

**Hình UD-2-1: Thông tin một mã loại hóa đơn bán hàng.**

Ứng với trường hợp xóa mẫu tin trong bảng SalesInvoiceTypes, bạn có thể khai báo thủ tục có tên udsDeleteSalesInvoiceTypes như ví dụ UD-2-3.

**M® 244****Ứng dụng: Sử dụng thủ tục nội tại trong ứng dụng****Ví dụ UD-2-3: Khai báo xóa mẫu tin**

```
CREATE PROC udsDeleteSalesInvoiceTypes
    @InvoiceTypeId CHAR(3)
AS
    DELETE FROM SalesInvoiceTypes
    WHERE
        InvoiceTypeId = CASE @InvoiceTypeId
            WHEN '' THEN InvoiceTypeId
            ELSE @InvoiceTypeId END
GO
```

Bạn có thể gọi thủ tục nội tại có tên udsDeleteSalesInvoiceTypes với hai trường hợp, trường xóa tất cả mẫu tin trong bảng bạn khai báo như ví dụ UD-2-4.

**Ví dụ UD-2-4: Khai báo xóa tất cả mẫu tin**

```
udsDeleteSalesInvoiceTypes ''
GO
```

Nếu muốn xóa một mẫu tin, bạn khai báo gọi thủ tục udsDeleteSalesInvoiceTypes như ví dụ UD-2-5.

**Ví dụ UD-2-5: Khai báo xóa một mẫu tin**

```
udsDeleteSalesInvoiceTypes 'A0010'
GO
```

Tuy nhiên, khi thêm mới hay cập nhật, bạn có thể khai báo thủ tục nội tại dùng chung cho hai trường hợp này dựa vào mã loại hóa đơn bán hàng như ví dụ UD-2-6.

**Ví dụ UD-2-6: Khai báo thêm và cập nhật**

```
CREATE PROC udsInsUpdSalesInvoiceTypes
    @Flag BIT,
    @InvoiceTypeId char(3),
    @InvoiceTypeNameInVietnamese nvarchar(50),
    @InvoiceTypeNameInSecondLanguage varchar(50)
AS
    IF @Flag = 0
        INSERT INTO SalesInvoiceTypes
        VALUES (@InvoiceTypeId,
                @InvoiceTypeNameInVietnamese,
                @InvoiceTypeNameInSecondLanguage, 1)
    ELSE
        UPDATE SalesInvoiceTypes
        SET InvoiceTypeNameInVietnamese =
            @InvoiceTypeNameInVietnamese,
            InvoiceTypeNameInSecondLanguage =
            @InvoiceTypeNameInSecondLanguage
            WHERE InvoiceTypeId = @InvoiceTypeId
GO
```

**Ứng dụng:** Sử dụng thủ tục nội tại trong ứng dụng

245

### 1.3. Bảng SalesInvoiceBatchs

Khi làm việc với bảng SalesInvoiceBatchs, đối với hành động truy vấn dữ liệu trong bảng SalesInvoiceBatchs, chúng ta cần khai báo thủ tục nội tại như ví dụ UD-3.

#### Ví dụ UD-3: Khai báo thủ tục nội tại truy vấn

```
CREATE PROC udsViewSalesInvoiceBatchs
    @InvoiceBatchNo VARCHAR(10)
AS
    SELECT * FROM SalesInvoiceBatchs
    WHERE
        InvoiceBatchNo = CASE @InvoiceBatchNo
            WHEN '' THEN InvoiceBatchNo
            ELSE @InvoiceBatchNo END
GO
```

Khi thực thi thủ tục nội tại trên để liệt kê danh sách lô hóa đơn bán hàng, bạn cần sử dụng cú pháp như ví dụ UD-3-1.

#### Ví dụ UD-3-1: Khai báo gọi thủ tục nội tại

```
udsViewSalesInvoiceBatchs 'SBI002'
GO
```

Khi thực thi thủ tục trên, bạn có thể tìm thấy danh sách mẫu tin như hình UD-3.

|   | InvoiceBatchNo | InvoiceBatchDate    | BatchDiscontinued |
|---|----------------|---------------------|-------------------|
| 1 | SBI001         | 2007-10-10 00:00:00 | 1                 |
| 2 | SBI002         | 2007-10-14 00:00:00 | 1                 |
| 3 | SBI003         | 2007-10-17 00:00:00 | 1                 |
| 4 | SBI004         | 2007-10-18 00:00:00 | 0                 |
| 5 | SBI005         | 2007-10-19 00:00:00 | 0                 |
| 6 | SBI006         | 2007-10-20 00:00:00 | 0                 |
| 7 | SBI007         | 2007-10-21 00:00:00 | 0                 |

**Hình UD-3: Danh sách lô hóa đơn bán hàng.**

Trong trường hợp muốn liệt kê một mẫu tin ứng với một lô hóa đơn bán hàng được chọn, bạn khai báo gọi thủ tục nội tại như ví dụ UD-3-2.

#### Ví dụ UD-3-2: Khai báo liệt kê một lô hóa đơn bán hàng

```
udsViewSalesInvoiceBatchs 'SBI002'
GO
```

**M® 246****Ứng dụng:** Sử dụng thủ tục nội tại trong ứng dụng

Khi thực thi thủ tục trên, bạn có thể tìm thấy thông tin của lô hóa đơn bán hàng như hình UD-3-1.

|   | InvoiceBatchNo | InvoiceBatchDate    | BatchDiscontinued |
|---|----------------|---------------------|-------------------|
| 1 | SBI002         | 2007-10-14 00:00:00 | 1                 |

**Hình UD-3-1: Thông tin một lô hóa đơn bán hàng.**

Trong trường hợp xóa mẫu tin trong bảng SalesInvoiceBatchs, bạn có thể khai báo thủ tục có tên udsDeleteSalesInvoiceBatchs như ví dụ UD-3-3.

**Ví dụ UD-3-3: Khai báo xóa mẫu tin**

```
CREATE PROC udsDeleteSalesInvoiceBatchs
    @InvoiceBatchNo VARCHAR(10)
AS
    DELETE FROM SalesInvoiceBatchs
    WHERE
        InvoiceBatchNo = CASE @InvoiceBatchNo
            WHEN '' THEN InvoiceBatchNo
            ELSE @InvoiceBatchNo END
GO
```

Sau khi tạo thủ tục nội tại trong ví dụ trên, bạn có thể gọi thủ tục nội tại này với hai trường hợp, nếu xóa tất cả mẫu tin trong bảng bạn khai báo như ví dụ UD-3-4.

**Ví dụ UD-3-4: Khai báo xóa tất cả mẫu tin**

```
udsDeleteSalesInvoiceBatchs ''
GO
```

Trong trường hợp xóa một mẫu tin, bạn khai báo gọi thủ tục udsDeleteSalesInvoiceBatchs như ví dụ UD-3-5.

**Ví dụ UD-3-5: Khai báo xóa một mẫu tin**

```
udsDeleteSalesInvoiceBatchs 'SBI002'
GO
```

Đối với trường hợp thêm mới hay cập nhật dữ liệu, bạn có thể khai báo thủ tục nội tại dùng chung cho hai trường hợp này dựa vào mã lô hóa đơn bán hàng như ví dụ UD-3-6.

**Ví dụ UD-3-6: Khai báo thêm và cập nhật**

```
CREATE PROC udsInsUpdSalesInvoiceBatchs
    @Flag BIT,
    @InvoiceBatchNo VARCHAR(10),
    @InvoiceBatchDate SMALLDATETIME,
```

**Ứng dụng:** Sử dụng thủ tục nội tại trong ứng dụng

247

```
    @BatchDiscontinued BIT
    .
AS
    IF @Flag = 0
        INSERT INTO SalesInvoiceBatchs
        VALUES (@InvoiceBatchNo, @InvoiceBatchDate,
                1, GETDATE(), CURRENT_USER)
    ELSE
        UPDATE SalesInvoiceBatchs
        SET InvoiceBatchDate = @InvoiceBatchDate,
            BatchDiscontinued = @BatchDiscontinued,
            UserName = CURRENT_USER
        WHERE InvoiceBatchNo = @InvoiceBatchNo
GO
```

#### **1.4. Bảng SalesInvoices**

Tương tự như các trường hợp trên, khi truy vấn dữ liệu trong bảng SalesInvoices, bạn cần khai báo thủ tục nội tại như ví dụ UD-4.

#### Ví dụ UD-4: Khai báo thủ tục nội tại truy vấn

```
CREATE PROC udsViewSalesInvoices
    @InvoiceNo VARCHAR(10)
AS
    SELECT * FROM SalesInvoices
    WHERE
        InvoiceNo = CASE @InvoiceNo
            WHEN '' THEN InvoiceNo
            ELSE @InvoiceNo END
GO
```

Nếu thực thi thủ tục nội tại trên để liệt kê danh sách hóa đơn bán hàng, ban cần sử dụng phát biểu gọi thủ tục nội tại như ví dụ UD-4-1.

#### **Ví dụ UD-4-1: Khai báo gọi thủ tục nội tại**

```
udsViewSalesInvoices ''  
GO
```

Nếu thực thi thủ tục trên, bạn có thể tìm thấy danh sách mẫu tin ứng với danh sách hóa đơn bán hàng như hình UD-4.

Tuy nhiên, trong một vài trường hợp người sử dụng muốn liệt kê một mẫu tin ứng với một hóa đơn bán hàng để trình bày hay cập nhật, bạn khai báo gọi thủ tục nội tại như ví dụ UD-4-2.

#### **Ví dụ UD-4-2: Khai báo liệt kê một hóa đơn bán hàng**

```
udsViewSalesInvoices 'SI00000003'  
GO
```

Khi thực thi thủ tục trên, bạn có thể tìm thấy thông tin của hóa đơn bán hàng như hình UD-4-1.

M® 248

**Ứng dụng:** Sử dụng thủ tục nội tại trong ứng dụng

|    | InvoiceNo  | InvoiceBatch... | DueDate             | InvoiceType... | CustomerID |
|----|------------|-----------------|---------------------|----------------|------------|
| 1  | SI00000001 | SBI001          | 2007-10-10 00:00:00 | SI1            | A0001      |
| 2  | SI00000002 | SBI001          | 2007-10-11 00:00:00 | SI1            | A0002      |
| 3  | SI00000003 | SBI001          | 2007-10-12 00:00:00 | SI1            | A0003      |
| 4  | SI00000004 | SBI001          | 2007-10-13 00:00:00 | SI1            | A0001      |
| 5  | SI00000005 | SBI002          | 2007-10-14 10:00:00 | SI1            | A0004      |
| 6  | SI00000006 | SBI002          | 2007-10-14 00:00:00 | SI1            | A0005      |
| 7  | SI00000007 | SBI003          | 2007-10-17 00:00:00 | SI1            | A0006      |
| 8  | SI00000008 | SBI003          | 2007-10-17 00:00:00 | SI1            | A0007      |
| 9  | SI00000009 | SBI004          | 2007-10-18 00:00:00 | SI1            | A0008      |
| 10 | SI00000010 | SBI005          | 2007-10-19 00:00:00 | SI1            | A0001      |
| 11 | SI00000011 | SBI005          | 2007-10-19 00:00:00 | SI1            | A0002      |
| 12 | SI00000012 | SBI006          | 2007-10-20 00:00:00 | SI1            | A0001      |
| 13 | SI00000013 | SBI006          | 2007-10-20 00:00:00 | SI1            | A0005      |

**Hình UD-4: Danh sách hóa đơn bán hàng.**

|   | InvoiceNo  | InvoiceBatch... | DueDate             | InvoiceType... | CustomerID |
|---|------------|-----------------|---------------------|----------------|------------|
| 1 | SI00000003 | SBI001          | 2007-10-12 00:00:00 | SI1            | A0003      |

**Hình UD-4-1: Thông tin một hóa đơn bán hàng.**

Ngoài cách liệt kê toàn bộ hay một hóa đơn bán hàng, người sử dụng thường tìm kiếm hóa đơn bán hàng theo mã khách hàng thì bạn khai báo thủ tục như ví dụ UD-4-3.

**Ví dụ UD-4-3: Khai báo liệt kê danh sách hóa đơn bán hàng theo mã khách hàng**

```
CREATE PROC udsViewSalesInvoicesByCustomer
    @InvoiceNo VARCHAR(10),
    @CustomerId CHAR(5)
AS
    SELECT * FROM SalesInvoices
    WHERE
        InvoiceNo = CASE @InvoiceNo
        WHEN '' THEN InvoiceNo
        ELSE @InvoiceNo END
        AND CustomerId = CASE @CustomerId
        WHEN '' THEN CustomerId
        ELSE @CustomerId END
GO
```

Bạn cũng có thể gọi thủ tục trên với mã khách hàng là A002 như ví dụ UD-4-4.

**Ứng dụng:** Sử dụng thủ tục nội tại trong ứng dụng

249

### Ví dụ UD-4-4: Khai báo liệt kê danh sách hóa đơn bán hàng cho

```
khách hàng A0002
GO
CREATE PROC udsViewSalesInvoicesByCustomer
    @CustomerID VARCHAR(10)
AS
SELECT * FROM SalesInvoices
    WHERE CustomerID = @CustomerID
GO
```

Nếu thực thi phát biểu trong ví dụ trên, bạn có thể tìm thấy danh sách mẫu tin trình bày như hình UD-4-2.

|   | InvoiceNo  | InvoiceBatch... | DueDate             | InvoiceType... | CustomerID |
|---|------------|-----------------|---------------------|----------------|------------|
| 1 | SI00000002 | SBI001          | 2007-10-11 00:00:00 | SI1            | A0002      |
| 2 | SI00000011 | SBI005          | 2007-10-19 00:00:00 | SI1            | A0002      |

**Hình UD-4-2: Hóa đơn mua hàng của khách hàng.**

Do chúng ta quản lý hóa đơn bán hàng theo lô, bạn cần khai báo thủ tục nội tại để liệt kê danh sách hóa đơn bán hàng theo từng lô như ví dụ UD-4-5.

### Ví dụ UD-4-5: Khai báo liệt kê hóa đơn bán hàng theo lô

```
CREATE PROC udsViewSalesInvoicesByBatch
    @InvoiceBatchNo VARCHAR(10)
AS
SELECT * FROM SalesInvoices
    WHERE InvoiceBatchNo = @InvoiceBatchNo
GO
```

Nếu gọi thủ tục nội tại trong ví dụ trên, bạn chỉ có thể chọn một chọn lựa là cung cấp mã số lô hóa đơn bán hàng như ví dụ UD-4-6.

### Ví dụ UD-4-6: Khai báo gọi thủ tục nội tại

```
udsViewSalesInvoicesByBatch
udsViewSalesInvoicesByBatch 'SBI001'
GO
```

Khi thực thi ví dụ trên, bạn có thể tìm thấy danh sách hóa đơn bán hàng trình bày như hình UD-4-3.

|   | InvoiceNo  | InvoiceBatch... | DueDate             | InvoiceType... | CustomerID |
|---|------------|-----------------|---------------------|----------------|------------|
| 1 | SI00000001 | SBI001          | 2007-10-10 00:00:00 | SI1            | A0001      |
| 2 | SI00000002 | SBI001          | 2007-10-11 00:00:00 | SI1            | A0002      |
| 3 | SI00000003 | SBI001          | 2007-10-12 00:00:00 | SI1            | A0003      |
| 4 | SI00000004 | SBI001          | 2007-10-13 00:00:00 | SI1            | A0001      |

**Hình UD-4-3: Danh sách hóa đơn bán hàng theo lô.**

M 250

**Ứng dụng:** Sử dụng thủ tục nội tại trong ứng dụng

Ứng với trường hợp xóa mẫu tin trong bảng SalesInvoices, bạn có thể khai báo thủ tục có tên udsDeleteSalesInvoices như ví dụ UD-4-7.

#### Ví dụ UP-4-7: Khai báo xóa mảng

```
CREATE PROC udsDeleteSalesInvoices
    @InvoiceNo VARCHAR(10)
AS
    DELETE FROM SalesInvoices
    WHERE
        InvoiceNo = @InvoiceNo
        WHEN '' THEN InvoiceNo
        ELSE @InvoiceNo END
GO
```

Bạn có thể gọi thủ tục nội tại có tên udsDeleteSalesInvoices với hai trường hợp, trường xóa tất cả mẫu tin trong bảng bạn khai báo như ví dụ UD-4-8.

#### Ví dụ UD-4-8: Khai báo xóa tất cả mảng tip

```
udsDeleteSalesInvoices ''  
GO
```

Đối với trường hợp xóa một mẫu tin, bạn khai báo gọi thủ tục `udsDeleteSalesInvoices` như ví dụ UD-4-9.

#### Ví dụ UD-4-9: Khai báo xóa một mảng tip

```
udsDeleteSalesInvoices 'SI00000015'  
GO
```

Ngoài ra, bạn cũng khai báo thủ tục nội tại dùng cho trường hợp thêm mới hay cập nhật thông tin hóa đơn bán hàng như ví dụ UD-4-10.

#### Ví dụ UD-4-10: Khai báo thêm và cập nhật

```
CREATE PROC udsInsUpdSalesInvoices
    @Flag BIT,
    @InvoiceNo VARCHAR(10),
    @InvoiceBatchNo VARCHAR(10),
    @DueDate SMALLDATETIME,
    @InvoiceTypeId CHAR(3),
    @CustomerId CHAR(5),
    @CurrencyId CHAR(3),
    @ExchangeRate DECIMAL (18, 0),
    @DescriptionInVietnamese NVARCHAR(150),
    @DescriptionInSecondLanguage NVARCHAR(150),
    @InvoiceDiscontinued BIT
AS
    IF @Flag = 0
        INSERT INTO SalesInvoices
        VALUES(@InvoiceNo, @InvoiceBatchNo,
               @DueDate, @InvoiceTypeId, @CustomerId,
```

**Ứng dụng:** Sử dụng thủ tục nội tại trong ứng dụng

251

```

@CurrencyId, @ExchangeRate,
@DescriptionInVietnamese,
@DescriptionInSecondLanguage,
1, 1, GETDATE(), CURRENT_USER)
ELSE
    UPDATE SalesInvoices
    SET DueDate = @DueDate,
        InvoiceTypeId = @InvoiceTypeId,
        CustomerId = @CustomerId,
        CurrencyId = @CurrencyId,
        ExchangeRate = @ExchangeRate,
        DescriptionInVietnamese = @DescriptionInVietnamese,
        DescriptionInSecondLanguage =
        @DescriptionInSecondLanguage,
        InvoiceDiscontinued = @InvoiceDiscontinued,
        UserName = CURRENT_USER
    WHERE InvoiceNo = @InvoiceNo
GO

```

**1.5. Bảng SalesInvoiceDetails**

Tương tự như các trường hợp bảng hóa đơn bán hàng (SalesInvoices), khi truy vấn dữ liệu trong bảng SalesInvoiceDetails, bạn cần khai báo thủ tục nội tại như ví dụ UD-5.

**Ví dụ UD-5: Khai báo thủ tục nội tại truy vấn**

```

CREATE PROC udsViewSalesInvoiceDetails
    @InvoiceNo VARCHAR(10)
AS
    SELECT * FROM SalesInvoiceDetails
    WHERE InvoiceNo = @InvoiceNo
GO

```

Khi thực thi thủ tục nội tại trên để liệt kê danh sách chi tiết hóa đơn bán hàng, bạn cần sử dụng phát biểu gọi thủ tục nội tại như ví dụ UD-5-1.

**Ví dụ UD-5-1: Khai báo gọi thủ tục nội tại**

```

udsViewSalesInvoiceDetails 'SI00000004'
GO

```

Nếu thực thi thủ tục trên, bạn có thể tìm thấy danh sách mẫu tin ứng với danh sách hóa đơn bán hàng như hình UD-5.

| OrdinalNumber | InvoiceNo  | ProductID | Quantity | Price | Discount | VATRate |
|---------------|------------|-----------|----------|-------|----------|---------|
| 1             | SI00000004 | P00001    | 150      | 15000 | 0        | 10      |
| 2             | SI00000004 | P00004    | 120      | 12000 | 0        | 10      |

Q MYSOLUTION [9.0 SP2] MYSOLUTION\Pham Huu Khang (54) AccountSystem

**Hình UD-5: Danh sách chi tiết hóa đơn bán hàng.**

252

**Ứng dụng:** Sử dụng thủ tục nội tại trong ứng dụng

Tuy nhiên, trong trường hợp người sử dụng muốn liệt kê một mẫu tin ứng với một hóa đơn bán hàng bao gồm thông tin chi tiết, bạn khai báo thủ tục nội tại như ví dụ UD-5-2.

**Ví dụ UD-5-2: Khai báo liệt kê một hóa đơn bán hàng**

```
CREATE PROC udsViewSalesInvoiceDetail
    @InvoiceNo VARCHAR(10),
    @ProductId VARCHAR(10),
    @OrdinalNumber tinyint
AS
    SELECT * FROM SalesInvoiceDetails
    WHERE InvoiceNo = @InvoiceNo
    AND ProductId = @ProductId
    AND OrdinalNumber = @OrdinalNumber
GO
```

Để liệt kê tất cả thông tin của hóa đơn bán hàng, bạn cần khai báo gọi thủ tục nội tại vừa tạo trong ví dụ trên như ví dụ UD-5-3.

**Ví dụ UD-5-3: Khai báo gọi thủ tục nội tại udsViewSalesInvoice**

```
udsViewSalesInvoiceDetail
'SI00000001', 'P00001', 1
GO
```

Khi thực thi thủ tục trên với 3 tham số ứng với một mẫu tin trong chi tiết hóa đơn bán hàng, bạn có thể tìm thấy thông tin chi tiết như hình UD-5-1.

| OrdinalNumber | InvoiceNo  | ProductID | Quantity | Price | Discount | VATRate |
|---------------|------------|-----------|----------|-------|----------|---------|
| 1             | SI00000001 | P00001    | 100      | 10000 | 50000    | 10      |

**Hình UD-5-1: Thông tin chi tiết của một hóa đơn bán hàng.**

**Chú ý:** Mỗi khi xóa mẫu tin trong bảng SalesInvoices thì tất cả mẫu tin liên quan nằm trong bảng SalesInvoiceDetails sẽ tự động xóa theo.

Tuy nhiên, ứng với trường hợp xóa một mẫu tin trong bảng SalesInvoiceDetails, bạn có thể khai báo thủ tục có tên udsDeleteSalesInvoiceDetails như ví dụ UD-5-4.

**Ví dụ UD-5-4: Khai báo xóa một mẫu tin**

```
CREATE PROC udsDeleteSalesInvoiceDetail
    @InvoiceNo VARCHAR(10),
    @ProductId VARCHAR(10),
    @OrdinalNumber tinyint
AS
```

**Ứng dụng:** Sử dụng thủ tục nội tại trong ứng dụng

253

```
DELETE FROM SalesInvoiceDetails
WHERE InvoiceNo = @InvoiceNo
AND ProductId = @ProductId
AND OrdinalNumber = @OrdinalNumber
GO
```

Bạn có thể gọi thủ tục nội tại có tên udsDeleteSalesInvoiceDetail như ví dụ UD-5-5.

**Ví dụ UD-5-5: Khai báo xóa một mẫu tin**

```
udsDeleteSalesInvoiceDetail 'SI00000001', 'P00001', 1
GO
```

Tương tự như các bảng khác, bạn cũng khai báo thủ tục nội tại dùng cho trường hợp thêm mới hay cập nhật thông tin chi tiết hóa đơn bán hàng như ví dụ UD-5-6.

**Ví dụ UD-5-6: Khai báo thêm và cập nhật**

```
CREATE PROC udsInsUpdSalesInvoiceDetails
    @Flag BIT,
    @InvoiceNo VARCHAR(10),
    @ProductId VARCHAR(10),
    @OrdinalNumber TINYINT,
    @Quantity DECIMAL (18, 0),
    @Price DECIMAL (18, 0),
    @Discount DECIMAL (18, 0),
    @VATRate DECIMAL (18, 0)
AS
    IF @Flag = 0
        INSERT INTO SalesInvoiceDetails
        VALUES (@OrdinalNumber, @InvoiceNo,
                @ProductId, @Quantity, @Price,
                @Discount, @VATRate)
    ELSE
        UPDATE SalesInvoiceDetails
        SET Quantity = @Quantity,
            Price = @Price,
            Discount = @Discount,
            VATRate = @VATRate
        WHERE InvoiceNo = @InvoiceNo
        AND ProductId = @ProductId
        AND OrdinalNumber = @OrdinalNumber
GO
```

**1.6. Thủ tục hỗ trợ làm báo cáo**

Với yêu cầu về thống kê thông tin bán hàng, bạn có thể khai báo một số thủ tục nội tại nhằm cung cấp dữ liệu tương ứng với các yêu cầu cụ thể.

M® 254

**Ứng dụng:** Sử dụng thủ tục nội tại trong ứng dụng

Chẳng hạn, bạn khai báo thủ tục nội tại cho phép liệt kê doanh thu bán hàng theo ngày như ví dụ UD-5-7.

**Ví dụ UD-5-7: Báo cáo doanh thu theo ngày**

```
CREATE PROC udsSalesByDate
AS
SELECT Convert(char(10), DueDate, 103) AS SalesDate,
SUM(Quantity*Price*(1+VATRate/100)-Discount) As
SalesAmount
FROM SalesInvoices S, SalesInvoiceDetails D
WHERE S.InvoiceNo = D.InvoiceNo
GROUP BY Convert(char(10), DueDate, 103)
GO
```

Bạn có thể gọi thủ tục trên và kết quả trình bày như hình UD-5-2.

|   | SalesDate  | SalesAmount    |
|---|------------|----------------|
| 1 | 10/10/2007 | 3200000.000000 |
| 2 | 11/10/2007 | 3200000.000000 |
| 3 | 12/10/2007 | 4300000.000000 |
| 4 | 13/10/2007 | 4059000.000000 |
| 5 | 14/10/2007 | 7938750.000000 |
| 6 | 17/10/2007 | 3164500.000000 |
| 7 | 18/10/2007 | 726250.000000  |
| 8 | 19/10/2007 | 2052250.000000 |
| 9 | 20/10/2007 | 426400.000000  |

**Hình UD-5-2: Doanh thu bán hàng theo ngày.**

Tương tự như trên, bạn có thể khai báo thủ tục nội tại để liệt kê doanh thu theo tuần trong tháng hiện hành như ví dụ UD-5-8.

**Ví dụ UD-5-8: Báo cáo doanh thu theo tuần**

```
CREATE PROC udsSalesByWeek
AS
SELECT DATEPART(ww, DueDate) AS SalesWeek,
```

**Ứng dụng:** Sử dụng thủ tục nội tại trong ứng dụng

255

```
SUM(Quantity*Price*(1+VATRate/100)-Discount) As
SalesAmount
FROM SalesInvoices S, SalesInvoiceDetails D
WHERE S.InvoiceNo = D.InvoiceNo
GROUP BY DATEPART(ww, DueDate)
GO
```

Bạn có thể gọi thủ tục trên và kết quả trình bày như hình UD-5-3.

|   | SalesWeek | SalesAmount     |
|---|-----------|-----------------|
| 1 | 41        | 14759000.000000 |
| 2 | 42        | 14308150.000000 |

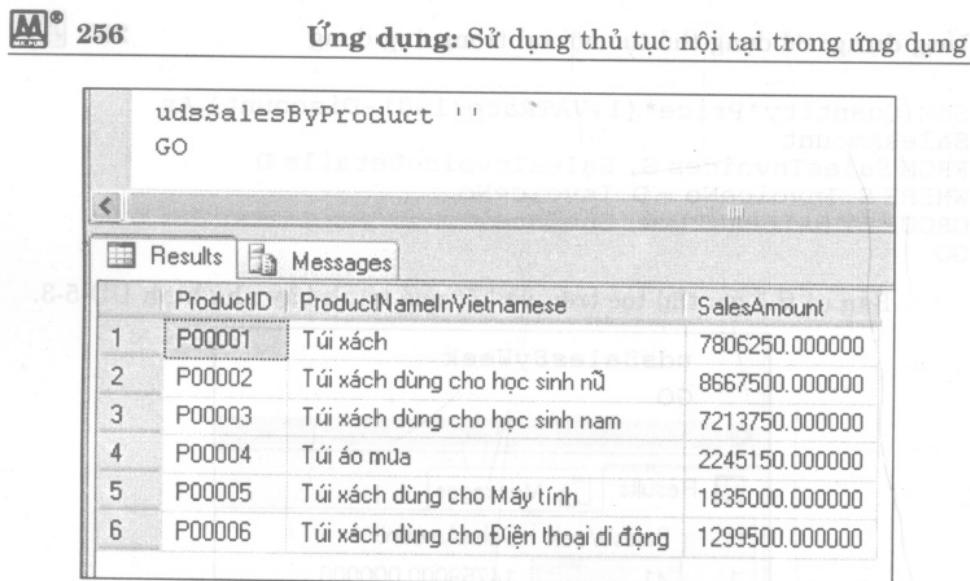
**Hình UD-5-3: Doanh thu bán hàng theo tuần.**

Bạn cũng có thể khai báo thủ tục nội tại cho phép người sử dụng báo cáo doanh số bán hàng theo sản phẩm như ví dụ UD-5-9.

**Ví dụ UD-5-9: Báo cáo doanh thu theo sản phẩm**

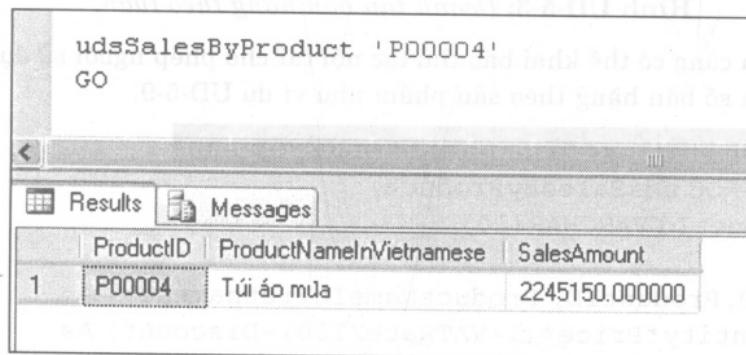
```
CREATE PROC udsSalesByProduct
    @ProductId VARCHAR(10)
AS
SELECT P.ProductID, ProductNameInVietnamese,
SUM(Quantity*Price*(1+VATRate/100)-Discount) As
SalesAmount
FROM Products P, SalesInvoiceDetails D
WHERE P.ProductID = D.ProductID
AND D.ProductId = CASE @ProductId
    WHEN '' THEN D.ProductId
    ELSE @ProductId END
GROUP BY P.ProductID, ProductNameInVietnamese
GO
```

Đối với trường hợp thủ tục trên, chúng ta có thể gọi nó bằng hai cách, cách thứ nhất là cung cấp mã sản phẩm là rỗng thì danh sách sản phẩm trình bày như hình UD-5-4.



**Hình UD-5-4:** Doanh thu của sản phẩm.

Nếu bạn muốn liệt kê doanh số bán hàng của một sản phẩm thì khai báo như hình UD-5-5.



**Hình UD-5-5:** Doanh số bán hàng của một sản phẩm.

Trong trường hợp liệt kê doanh thu theo khách hàng, bạn có thể khai báo thủ tục nội tại như ví dụ UD-5-10.

**Ví dụ UD-5-10: Khai báo doanh thu theo khách hàng**

**Viết UB-3-10: Khai báo doanh thu**

```
CREATE PROC ussSalesByCustomerID CHAR(5)
```

35

```
AS  
SELECT C.CustomerID, CompanyNameInVietnamese,  
SUM(Quantity*Price*(1+VATRate/100)-Discount) As  
SalesAmount  
FROM Customers C INNER JOIN SalesInvoices S  
ON C.CustomerID = S.CustomerID  
INNER JOIN SalesInvoiceDetails D
```

**Ứng dụng:** Sử dụng thủ tục nội tại trong ứng dụng

257

```

ON S.InvoiceNo = D.InvoiceNo
AND C.CustomerId = CASE @CustomerId
    WHEN '' THEN C.CustomerId
    ELSE @CustomerId END
GROUP BY C.CustomerID, CompanyNameInVietnamese
GO

```

Nếu bạn gọi thủ tục trên với mã khách hàng là rỗng thì danh sách khách hàng cùng với doanh thu của họ trình bày như hình UD-5-6.

| CustomerID | CompanyNameInVietnamese                       | SalesAmount    |
|------------|---|----------------|
| A0001      | Công ty Trách Nhiệm Hữu Hạn Microsoft Vietnam | 8976250.000000 |
| A0002      | Công ty Trách Nhiệm Hữu Hạn IBM Vietnam       | 3832500.000000 |
| A0003      | Công ty Trách Nhiệm Hữu Hạn Kodaka Vietnam    | 4300000.000000 |
| A0004      | Công ty Trách Nhiệm Hữu Hạn E-Google Vietnam  | 5670000.000000 |
| A0005      | Công ty Cổ phần Suzumi Vietnam                | 2397650.000000 |
| A0006      | Tập đoàn UCIA USA                             | 2185000.000000 |
| A0007      | Công ty Đa quốc gia UFCA                      | 979500.000000  |
| A0008      | Công ty Cổ phần ReruitVietnam                 | 726250.000000  |

**Hình UD-5-6:** Doanh thu theo khách hàng.

Trong trường hợp liệt kê doanh số bán hàng cho một khách hàng, bạn có thể khai báo tương tự như hình UD-5-7.

| CustomerID | CompanyNameInVietnamese                      | SalesAmount    |
|------------|--|----------------|
| A0004      | Công ty Trách Nhiệm Hữu Hạn E-Google Vietnam | 5670000.000000 |

**Hình UD-5-7:** Doanh số bán hàng cho một khách hàng.

Bạn cũng có thể liệt kê doanh thu bán hàng theo tỉnh thành bằng cách khai báo như ví dụ UD-5-11.

M® 258

**Ứng dụng:** Sử dụng thủ tục nội tại trong ứng dụng**Ví dụ UD-5-11: Khai báo doanh thu theo tỉnh thành**

```

CREATE PROC udsSalesByProvince
    @ProvinceID CHAR(3)
AS
SELECT C.ProvinceID, P.ProductID,
ProductNameInVietnamese,
SUM(Quantity*Price*(1+VATRate/100)-Discount) AS
SalesAmount
FROM Customers C, SalesInvoices S,
Products P, SalesInvoiceDetails D
WHERE P.ProductID = D.ProductID
AND C.CustomerID = S.CustomerID
AND S.InvoiceNo = D.InvoiceNo
AND C.ProvinceID = CASE @ProvinceID
    WHEN '' THEN ProvinceId
    ELSE @ProvinceId END
GROUP BY C.ProvinceID, P.ProductID,
ProductNameInVietnamese
ORDER BY SalesAmount DESC, ProductID ASC
GO

```

Nếu liệt kê doanh thu bán hàng của tất cả tỉnh thành thì bạn khai báo gọi thủ tục nội tại trong ví dụ trên như hình UD-5-8.

The screenshot shows the SQL Server Management Studio interface. In the top query window, the stored procedure `udsSalesByProvince` is executed with the command `GO`. In the bottom results window, the output is displayed in a table format:

| ProvinceID | ProductID | ProductNameInVietnamese              | SalesAmount    |
|------------|-----------|--------------------------------------|----------------|
| 1 HCM      | P00002    | Túi xách dùng cho học sinh nữ        | 7617500.000000 |
| 2 HAN      | P00003    | Túi xách dùng cho học sinh nam       | 4845000.000000 |
| 3 HAN      | P00001    | Túi xách                             | 4075000.000000 |
| 4 HCM      | P00001    | Túi xách                             | 3731250.000000 |
| 5 HCM      | P00003    | Túi xách dùng cho học sinh nam       | 2368750.000000 |
| 6 HCM      | P00004    | Túi áo mưa                           | 2245150.000000 |
| 7 DNA      | P00005    | Túi xách dùng cho Máy tính           | 1796250.000000 |
| 8 HAN      | P00002    | Túi xách dùng cho học sinh nữ        | 1050000.000000 |
| 9 HCM      | P00006    | Túi xách dùng cho Điện thoại di động | 910750.000000  |
| 10 DNA     | P00006    | Túi xách dùng cho Điện thoại di động | 388750.000000  |
| 11 HCM     | P00005    | Túi xách dùng cho Máy tính           | 38750.000000   |

**Hình UD-5-8: Doanh thu theo tỉnh thành..**

Trong trường hợp chỉ định một tỉnh thành thì bạn khai báo tương tự như hình UD-5-9.

**Ứng dụng:** Sử dụng thủ tục nội tại trong ứng dụng

259



The screenshot shows the execution of a stored procedure named 'udsSalesByProvince' with parameter 'HCM'. The results are displayed in a table with columns: ProvinceID, ProductID, ProductNameInVietnamese, and SalesAmount. The data is as follows:

|   | ProvinceID | ProductID | ProductNameInVietnamese              | SalesAmount    |
|---|------------|-----------|--------------------------------------|----------------|
| 1 | HCM        | P00002    | Túi xách dùng cho học sinh nữ        | 7617500.000000 |
| 2 | HCM        | P00001    | Túi xách                             | 3731250.000000 |
| 3 | HCM        | P00003    | Túi xách dùng cho học sinh nam       | 2368750.000000 |
| 4 | HCM        | P00004    | Túi áo mưa                           | 2245150.000000 |
| 5 | HCM        | P00006    | Túi xách dùng cho Điện thoại di động | 910750.000000  |
| 6 | HCM        | P00005    | Túi xách dùng cho Máy tính           | 38750.000000   |

**Hình UD-5-9:** Doanh thu của sản phẩm theo một tỉnh thành.

Tương tự như trong chương ứng dụng của tập: *Lập trình T\_SQL*, bạn có thể khai báo thủ tục nội tại cho phép liệt kê N sản phẩm có doanh số bán hàng lớn nhất theo tỉnh thành.

Để làm điều này, bạn khai báo thủ tục nội tại có cấu trúc như ví dụ UD-5-12.

#### Ví dụ UD-5-12: Doanh số của N sản phẩm bán chạy nhất

```
CREATE PROC udsTopSalesOfProducts
    @TopNumber tinyint,
    @ProvinceId CHAR(3)
AS
SELECT TOP(@TopNumber) WITH TIES
P.ProductID, ProductNameInVietnamese,
    SUM(Quantity*Price*(1+VATRate/100)-Discount) As
SalesAmount
FROM Customers C, SalesInvoices S,
Products P, SalesInvoiceDetails D
WHERE P.ProductID = D.ProductID
AND C.CustomerID = S.CustomerID
AND S.InvoiceNo = D.InvoiceNo
AND C.ProvinceID = CASE @ProvinceId
    WHEN '' THEN C.ProvinceID
    ELSE @ProvinceId END
GROUP BY P.ProductID, ProductNameInVietnamese
ORDER BY SalesAmount DESC, ProductID ASC
GO
```

Do thủ tục trên yêu cầu hai tham số, chính vì vậy bạn có thể gọi nó bằng cách cung cấp giá trị tương tự như hình UD-5-10.

260

**Ứng dụng:** Sử dụng thủ tục nội tại trong ứng dụng

| ProductID | ProductNameInVietnamese        | SalesAmount    |
|-----------|--------------------------------|----------------|
| 1 P00002  | Túi xách dùng cho học sinh nữ  | 8667500.000000 |
| 2 P00001  | Túi xách                       | 7806250.000000 |
| 3 P00003  | Túi xách dùng cho học sinh nam | 7213750.000000 |
| 4 P00004  | Túi áo mưa                     | 2245150.000000 |
| 5 P00005  | Túi xách dùng cho Máy tính     | 1835000.000000 |

**Hình UD-5-10:** Doanh số của 5 sản phẩm bán chạy nhất.

Tương tự như trên, nếu bạn muốn liệt kê 3 sản phẩm bán chạy nhất tại tỉnh thành có mã là HCM thì bạn khai báo tương tự như hình UD-5-11.

| ProductID | ProductNameInVietnamese        | SalesAmount    |
|-----------|--------------------------------|----------------|
| 1 P00002  | Túi xách dùng cho học sinh nữ  | 7617500.000000 |
| 2 P00001  | Túi xách                       | 3731250.000000 |
| 3 P00003  | Túi xách dùng cho học sinh nam | 2368750.000000 |

**Hình UD-5-11:** Doanh số của 3 sản phẩm bán chạy nhất tại HCM.

Trong trường hợp bạn muốn liệt kê N khách hàng mua hàng có số tiền lớn nhất thì khai báo như ví dụ UD-5-13.

**Ví dụ UD-5-13: Khai báo doanh thu cao nhất theo khách hàng**

```
CREATE PROC udsTopSalesOfCustomers
```

```
    @TopNumber tinyint,
    @ProvinceId CHAR(3)
```

```
AS
```

```
SELECT TOP (@TopNumber)
```

```
    C.CustomerID, CompanyNameInVietnamese,
```

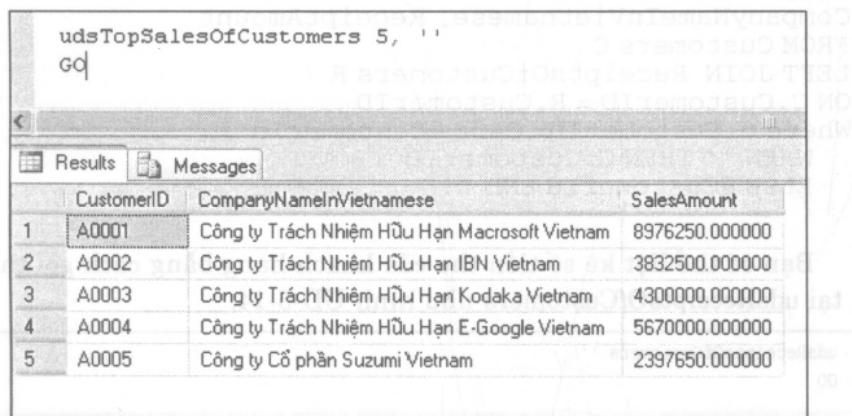
**Ứng dụng:** Sử dụng thủ tục nội tại trong ứng dụng261 

```

SUM(Quantity*Price*(1+VATRate/100)-Discount) As SalesAmount
FROM Customers C LEFT JOIN SalesInvoices S
ON C.CustomerID = S.CustomerID
LEFT JOIN SalesInvoiceDetails D
ON S.InvoiceNo = D.InvoiceNo
AND ProvinceId = CASE @ProvinceId
    WHEN '' THEN ProvinceId
    ELSE @ProvinceId END
GROUP BY C.CustomerID, CompanyNameInVietnamese
Order By C.CustomerID
GO

```

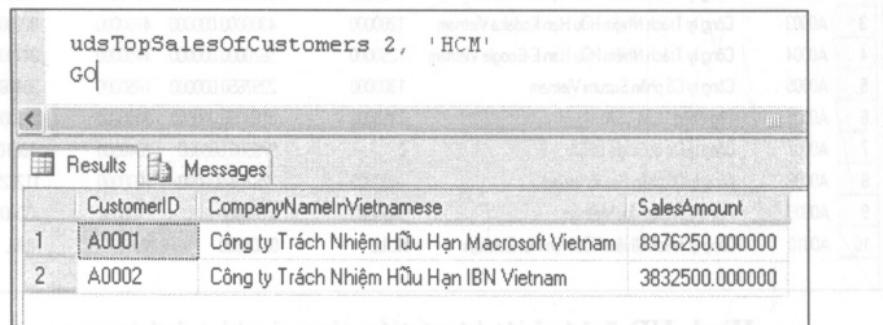
Để gọi thủ tục trong ví dụ trên, bạn cũng có thể chia ra hai trường hợp, trường hợp thứ nhất là cung cấp giá trị cho tham số @ProvinceId là rỗng như hình UD-5-12.



| CustomerID | CompanyNameInVietnamese                       | SalesAmount    |
|------------|---|----------------|
| 1 A0001    | Công ty Trách Nhiệm Hữu Hạn Microsoft Vietnam | 8976250.000000 |
| 2 A0002    | Công ty Trách Nhiệm Hữu Hạn IBM Vietnam       | 3832500.000000 |
| 3 A0003    | Công ty Trách Nhiệm Hữu Hạn Kodaka Vietnam    | 4300000.000000 |
| 4 A0004    | Công ty Trách Nhiệm Hữu Hạn E-Google Vietnam  | 5670000.000000 |
| 5 A0005    | Công ty Cổ phần Suzumi Vietnam                | 2397650.000000 |

**Hình UD-5-12: 5 khách hàng có doanh thu lớn nhất.**

Nếu muốn liệt kê danh sách 2 khách hàng có doanh thu lớn nhất tại tỉnh thành có mã là HCM thì khai báo như hình UD-5-13.



| CustomerID | CompanyNameInVietnamese                       | SalesAmount    |
|------------|---|----------------|
| 1 A0001    | Công ty Trách Nhiệm Hữu Hạn Microsoft Vietnam | 8976250.000000 |
| 2 A0002    | Công ty Trách Nhiệm Hữu Hạn IBM Vietnam       | 3832500.000000 |

**Hình UD-5-13: 2 khách hàng có doanh thu lớn nhất tại HCM.**

M® 262

**Ứng dụng:** Sử dụng thủ tục nội tại trong ứng dụng

Đối với trường hợp phiếu thu, bạn cũng khai báo thủ tục nội tại cho phép người sử dụng liệt kê tổng tiền thu của khách hàng như ví dụ UD-5-14.

**Ví dụ UD-5-14: Khai báo tổng thu của khách hàng**

```
CREATE PROC udsReceiptsOfCustomers
    @CustomerId CHAR(5)
AS
WITH ReceiptsOfCustomers
AS
(
    SELECT CustomerID,
    ISNULL(SUM(ReceiptAmount), 0) AS ReceiptAmount
    FROM Receipts
    GROUP BY CustomerID
)
SELECT C.CustomerID,
    CompanyNameInVietnamese, ReceiptAmount
    FROM Customers C
    LEFT JOIN ReceiptsOfCustomers R
    ON C.CustomerID = R.CustomerID
    WHERE C.CustomerID = CASE @CustomerId
        WHEN '' THEN C.CustomerID
        ELSE @CustomerId END
GO
```

Bạn có thể liệt kê số tiền thu của khách hàng bằng cách gọi thủ tục nội tại udsReceiptsOfCustomers như hình UD-5-14.

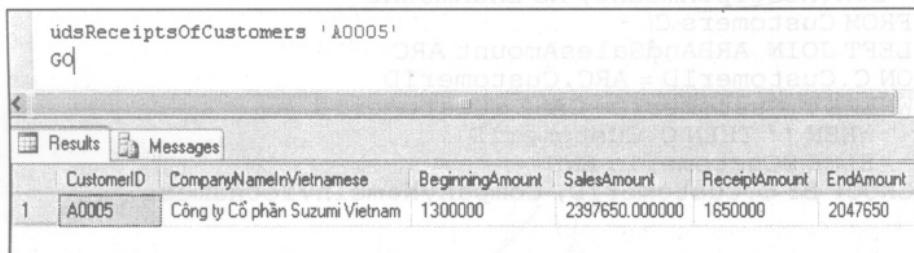
The screenshot shows the SQL Server Management Studio interface. In the top-left pane, there is a code editor window containing the T-SQL script for the stored procedure. Below it, the 'Results' tab is selected, showing the output of the query. The results are presented in a table with the following data:

|    | CustomerID | CompanyNameInVietnamese                       | BeginningAmount | SalesAmount    | ReceiptAmount | EndAmount |
|----|------------|---|-----------------|----------------|---------------|-----------|
| 1  | A0001      | Công ty Trách Nhiệm Hữu Hạn Microsoft Vietnam | 1000000         | 8976250.000000 | 6255000       | 3721250   |
| 2  | A0002      | Công ty Trách Nhiệm Hữu Hạn IBM Vietnam       | 1100000         | 3832500.000000 | 1650000       | 3282500   |
| 3  | A0003      | Công ty Trách Nhiệm Hữu Hạn Kodaka Vietnam    | 1200000         | 4300000.000000 | 4700000       | 800000    |
| 4  | A0004      | Công ty Trách Nhiệm Hữu Hạn E-Google Vietnam  | 1250000         | 5670000.000000 | 4450000       | 2470000   |
| 5  | A0005      | Công ty Cổ phần Suzuki Vietnam                | 1300000         | 2397650.000000 | 1650000       | 2047650   |
| 6  | A0006      | Tập đoàn UCIA USA                             | 1350000         | 2185000.000000 | 3000000       | 535000    |
| 7  | A0007      | Công ty Đa quốc gia UFCA                      | 0               | 979500.000000  | 2040500       | -1061000  |
| 8  | A0008      | Công ty Cổ phần ReruitVietnam                 | 1400000         | 726250.000000  | 1000000       | 1126250   |
| 9  | A0009      | Trung tâm giáo dục Vietnam                    | 1450000         | 0.000000       | 1100000       | 350000    |
| 10 | A0010      | Công ty Trách Nhiệm Hữu Hạn Hot Getways       | NULL            | 0.000000       | NULL          | NULL      |

**Hình UD-5-14: Liệt kê số tiền thu của khách hàng.**

**Ứng dụng:** Sử dụng thủ tục nội tại trong ứng dụng263 

Tuy nhiên, bạn có thể liệt kê số tiền thu của một khách hàng bằng cách gọi thủ tục nội tại udsReceiptsOfCustomers như hình UD-5-15.



| udsReceiptsOfCustomers 'A0005' |                                |                 |                |               |           |
|--------------------------------|--------------------------------|-----------------|----------------|---------------|-----------|
| GO                             |                                |                 |                |               |           |
| Results                        |                                | Messages        |                |               |           |
| CustomerID                     | CompanyNameInVietnamese        | BeginningAmount | SalesAmount    | ReceiptAmount | EndAmount |
| 1 A0005                        | Công ty Cổ phần Suzumi Vietnam | 1300000         | 2397650.000000 | 1650000       | 2047650   |

**Hình UD-5-15: Liệt kê số tiền thu của khách hàng A0005.**

Chú ý: Bạn có thể tham khảo chi tiết về tình hình thu và chi trong những phần kế tiếp.

Sau cùng, bạn phải khai báo thủ tục nội tại cho phép liệt kê công nợ của khách hàng của tháng hiện hành như ví dụ UD-5-15.

#### Ví dụ UD-5-15: Khai báo tình hình công nợ thu khách hàng

```
CREATE PROC udsAR
    @MonthYear CHAR(7),
    @CustomerId CHAR(5)
AS
WITH ARBAndSalesAmount
AS
(
    SELECT CustomerID, EndAmount As BeginningAmount,
    0 As SalesAmount, 0 As ReceiptAmount
    FROM CloseAccountReceivableDetails
    WHERE CloseMonth = @MonthYear
    UNION ALL
    SELECT CustomerID, 0 ,
    SUM(Quantity*Price*(1+VATRate/100)-Discount) As
SalesAmount, 0
    FROM SalesInvoices S, SalesInvoiceDetails D
    WHERE S.InvoiceNo = D.InvoiceNo
    GROUP BY CustomerID
    UNION ALL
    SELECT CustomerID, 0, 0,
    ISNULL(SUM(ReceiptAmount),0) As ReceiptAmount
    FROM Receipts
    GROUP BY CustomerID
)
SELECT C.CustomerID, CompanyNameInVietnamese,
    SUM(BeginningAmount) AS BeginningAmount,
    SUM(SalesAmount) AS SalesAmount,
```

M® 264

**Ứng dụng:** Sử dụng thủ tục nội tại trong ứng dụng

```

SUM(ReceiptAmount) AS ReceiptAmount,
SUM(BeginningAmount) + SUM(SalesAmount)
- SUM(ReceiptAmount) AS EndAmount
FROM Customers C
LEFT JOIN ARBAndSalesAmount ARC
ON C.CustomerID = ARC.CustomerID
WHERE C.CustomerID= CASE @CustomerId
    WHEN '' THEN C.CustomerID
    ELSE @CustomerId END
GROUP BY C.CustomerID, CompanyNameInVietnamese
GO

```

Chú ý: Đối tượng View có tên ARBAndSalesAmount đã được tạo trong tập: *Lập trình T\_SQL*.

Để liệt kê công nợ thu của khách hàng, bạn có thể gọi thủ tục nội tại udsAR như hình UD-5-16.

| CustomerID | CompanyNameInVietnamese  | BeginningAmount | SalesAmount    | ReceiptAmount | EndAmount |
|------------|--|-----------------|----------------|---------------|-----------|
| A0001      | Công ty Trách Nhiệm Hữu H...<br>...tập trung vào các sản phẩm công nghệ thông tin    | 1000000         | 8976250.000000 | 6255000       | 3721250   |
| A0002      | Công ty Trách Nhiệm Hữu H...<br>...tập trung vào các sản phẩm công nghệ thông tin    | 1100000         | 3832500.000000 | 1650000       | 3282500   |
| A0003      | Công ty Trách Nhiệm Hữu H...<br>...tập trung vào các sản phẩm công nghệ thông tin    | 1200000         | 4300000.000000 | 4700000       | 800000    |
| A0004      | Công ty Trách Nhiệm Hữu H...<br>...tập trung vào các sản phẩm công nghệ thông tin    | 1250000         | 5670000.000000 | 4450000       | 2470000   |
| A0005      | Công ty Cổ phần Suzumi Vietn...<br>...tập trung vào các sản phẩm công nghệ thông tin | 1300000         | 2397650.000000 | 1650000       | 2047650   |
| A0006      | Tập đoàn UCIA USA  | 1350000         | 2185000.000000 | 3000000       | 535000    |
| A0007      | Công ty Đa quốc gia UFCA   | 0               | 979500.000000  | 2040500       | -1061000  |
| A0008      | Công ty Cổ phần ReruitVietnam  | 1400000         | 726250.000000  | 1000000       | 1126250   |
| A0009      | Trung tâm giáo dục Vietnam   | 1450000         | 0.000000       | 1100000       | 350000    |
| A0010      | Công ty Trách Nhiệm Hữu H...<br>...tập trung vào các sản phẩm công nghệ thông tin    | NULL            | NULL           | NULL          | NULL      |

**Hình UD-5-16:** Công nợ phải thu của khách hàng.

Tuy nhiên, khi liệt kê công nợ thu của một khách hàng, bạn có thể gọi thủ tục nội tại udsAR tương tự như hình UD-5-17.

| CustomerID | CompanyNameInVi...<br>...tập trung vào các sản phẩm công nghệ thông tin   | BeginningAmount | SalesAmount    | ReceiptAmount | EndAmount |
|------------|---|-----------------|----------------|---------------|-----------|
| A0005      | Công ty Cổ phần S...<br>...tập trung vào các sản phẩm công nghệ thông tin | 1300000         | 2397650.000000 | 1650000       | 2047650   |

**Hình UD-5-17:** Công nợ phải thu của khách hàng A0005.

Do yêu cầu của hệ thống, cuối tháng người sử dụng cần thực hiện chức năng đóng sổ, chính vì vậy bạn cần khai báo thủ tục để chuyển thông tin công nợ phải thu của khách hàng vào hai bảng CloseAccountReceivable và CloseAccountReceivableDetails nhằm cung cấp thông tin công nợ thu đầu kỳ của khách hàng cho tháng kế tiếp.

Ngoài ra, nếu sau khi đóng sổ thì bạn phải xóa tất cả dữ liệu của các bảng liên quan như: Receipts, SalesInvoiceBatchs, SalesInvoices, SalesInvoiceDetails dựa vào cờ ứng với tham số có tên @DeleteData.

**Chú ý:** Nếu bạn cho phép người sử dụng thực hiện việc đóng sổ tạm thời thì truyền giá trị 0 cho tham số @DeleteData.

Để làm điều này, trước tiên bạn khai báo thủ tục nội tại có tên udsCloseMonthForAR với cấu trúc như ví dụ UD-5-16.

#### Ví dụ UD-5-16: Khai báo đóng sổ phần công nợ phải thu

```
CREATE PROC udsCloseMonthForAR
    @CurrentMonth CHAR(7),
    @DeleteData BIT
AS
    -- Tháng trước đó
    DECLARE @MonthYear CHAR(7)
    SET @MonthYear = dbo.udfPreviousMonth(@CurrentMonth)
    -- Xóa dữ liệu nếu trước đó có tính tạm thời

    DELETE FROM CloseAccountReceivable
    WHERE CloseMonth = @CurrentMonth;
    -- Thêm dữ liệu vào bảng CloseAccountReceivable

    INSERT INTO CloseAccountReceivable
    VALUES (@CurrentMonth, GETDATE(), CURRENT_USER);
    -- Thêm dữ liệu vào bảng CloseAccountReceivableDetails
    WITH ARBAndSalesAmount
    AS
    (
        SELECT CustomerID, EndAmount As BeginningAmount,
        0 As SalesAmount, 0 As ReceiptAmount
        FROM CloseAccountReceivableDetails
        WHERE CloseMonth = @MonthYear
        UNION ALL
        SELECT CustomerID, 0,
        SUM(Quantity*Price*(1+VATRate/100)-Discount) As
        SalesAmount, 0
        FROM SalesInvoices S, SalesInvoiceDetails D
        WHERE S.InvoiceNo = D.InvoiceNo
        GROUP BY CustomerID
```

**M® 266****Ứng dụng: Sử dụng thủ tục nội tại trong ứng dụng**

```

UNION ALL
SELECT CustomerID, 0, 0,
       ISNULL(SUM(ReceiptAmount), 0) AS ReceiptAmount
  FROM Receipts
 GROUP BY CustomerID
)

```

-- Thêm dữ liệu vào bảng CloseAccountReceivableDetails

```

INSERT INTO CloseAccountReceivableDetails
SELECT C.CustomerID, @CurrentMonth,
       SUM(BeginningAmount) AS BeginningAmount,
       SUM(SalesAmount) AS SalesAmount,
       SUM(ReceiptAmount) AS ReceiptAmount,
       SUM(BeginningAmount) + SUM(SalesAmount)
     - SUM(ReceiptAmount) AS EndAmount
  FROM Customers C
 LEFT JOIN ARBAndSalesAmount ARC
    ON C.CustomerID = ARC.CustomerID
 GROUP BY C.CustomerID;

```

-- Xóa dữ liệu trong các bảng liên quan

```

IF @DeleteData = 1
BEGIN
    PRINT 'Delete all data';
    --DELETE FROM Receipts;
    --DELETE FROM SalesInvoiceBatchs;
    --DELETE FROM SalesInvoices;
    --DELETE FROM SalesInvoiceDetails;
END
GO

```

**Chú ý:** Nếu bạn thực thi 4 phát biểu DELETE ứng với trường hợp tham số @DeleteData có giá trị là 1 để xóa dữ liệu trong bảng liên quan là Receipts, SalesInvoiceBatchs, SalesInvoices, SalesInvoiceDetails thì dữ liệu thêm vào sẽ tính toán cho tháng kế tiếp.

Trong trường hợp cho phép người sử dụng tạm thời tính lại công nợ phải thu của một khách hàng sau khi đã thực hiện việc đóng sổ tạm thời thì khai báo như ví dụ UD-5-17.

**Ví dụ UD-5-17: Tính lại công nợ phải thu của một khách hàng**

```

CREATE PROC udsCloseMonthForAROfCustomer
    @CurrentMonth CHAR(7),
    @CustomerId CHAR(5)
AS
DECLARE @MonthYear CHAR(7)
SET @MonthYear = dbo.udfPreviousMonth(@CurrentMonth)
IF @CustomerId != ''
BEGIN

```

Ứng dụng: Sử dụng thủ tục nội tại trong ứng dụng

267

```

DELETE FROM CloseAccountReceivableDetails
WHERE CloseMonth = @CurrentMonth
AND CustomerId = @CustomerId;
WITH ARBAndSalesAmount
AS
(
    SELECT CustomerId, EndAmount AS BeginningAmount,
    0 AS SalesAmount, 0 AS ReceiptAmount
    FROM CloseAccountReceivableDetails
    WHERE CloseMonth = @MonthYear
    AND CustomerId = @CustomerId
    UNION ALL
    SELECT CustomerId, 0 ,
    SUM(Quantity*Price*(1+VATRate/100)-Discount)
    AS SalesAmount, 0
    FROM SalesInvoices S, SalesInvoiceDetails D
    WHERE S.InvoiceNo = D.InvoiceNo
    AND CustomerId = @CustomerId
    GROUP BY CustomerId
    UNION ALL
    SELECT CustomerId, 0, 0,
    ISNULL(SUM(ReceiptAmount),0) AS ReceiptAmount
    FROM Receipts
    WHERE CustomerId = @CustomerId
    GROUP BY CustomerId
)
INSERT INTO CloseAccountReceivableDetails
SELECT C.CustomerId, @CurrentMonth,
    SUM(BeginningAmount) AS BeginningAmount,
    SUM(SalesAmount) AS SalesAmount,
    SUM(ReceiptAmount) AS ReceiptAmount,
    SUM(BeginningAmount) + SUM(SalesAmount)
    - SUM(ReceiptAmount) AS EndAmount
    FROM Customers C
    LEFT JOIN ARBAndSalesAmount ARC
    ON C.CustomerId = ARC.CustomerId
    WHERE C.CustomerId = @CustomerId
    GROUP BY C.CustomerId;
END
GO

```

Chú ý: Dựa vào cách khai báo thủ tục nội tại ứng với các bảng dữ liệu vừa trình bày ở trên, bạn có thể tự thiết kế một số thủ tục nội tại khác để giải quyết vấn đề đặc thù nào đó tùy ý.

## 2. PHẦN KẾ TOÁN CÔNG NỢ PHẢI TRẢ

Tương tự như những chức năng của phần các khoản phải thu, khi làm việc với phần các khoản phải chi, chúng ta có các bảng dữ liệu liên quan là

**Ứng dụng:** Sử dụng thủ tục nội tại trong ứng dụng

Suppliers, PurchaseInvoiceBatchs, PurchaseInvoiceTypes, PurchaseInvoices và PurchaseInvoiceDetails.

Chú ý: Bạn có thể tìm thấy các ví dụ trình bày của chương này nằm trong tập tin có tên StoredProcedureForAP.sql.

## 2.1. Bảng Suppliers

Khi truy vấn dữ liệu trong bảng Suppliers, bạn cần khai báo thủ tục nội tại như ví dụ UD-6.

### Ví dụ UD-6: Khai báo thủ tục nội tại truy vấn bảng Suppliers

```
CREATE PROC udsViewSuppliers
    @SupplierId CHAR(5)
AS
    SELECT * FROM Suppliers
    WHERE 1=1
    AND SupplierId = CASE @SupplierId
        WHEN '' THEN SupplierId
        ELSE @SupplierId END
GO
```

Khi liệt kê danh sách nhà cung cấp bằng việc thực thi thủ tục nội tại udsViewSuppliers, bạn cần sử dụng cú pháp như ví dụ UD-6-1.

### Ví dụ UD-6-1: Khai báo gọi thủ tục nội tại

```
udsViewSuppliers ''
```

Khi thực thi thủ tục trên, bạn có thể tìm thấy danh sách mẫu tin ứng với nhà cung cấp như hình UD-6.

| SupplierID | CompanyNameInVietnamese                        |
|------------|--|
| 1 S0001    | Công ty Trách Nhiệm Hữu Hạn Ajinomoto Việt Nam |
| 2 S0002    | Công ty Trách Nhiệm Hữu Hạn Suzumi Việt Nam    |
| 3 S0003    | Công ty Cổ phần Yamaka Việt Nam                |
| 4 S0004    | Công ty Trách Nhiệm Hữu Hạn Microsoft Vietnam  |
| 5 S0005    | Công ty Trách Nhiệm Hữu Hạn Delle Vietnam      |

**Hình UD-6: Danh sách nhà cung cấp.**

Trong trường hợp muốn liệt kê một mẫu tin ứng với một nhà cung cấp được chọn, bạn khai báo gọi thủ tục nội tại như ví dụ UD-6-2.

**Ứng dụng:** Sử dụng thủ tục nội tại trong ứng dụng

269

**Ví dụ UD-6-2: Khai báo liệt kê một nhà cung cấp**  
 udsViewSuppliers 'S0003'  
 GO

Khi thực thi thủ tục trên, bạn có thể tìm thấy thông tin của nhà cung cấp có mã S0003 như hình UD-6-1.

| SupplierID | CompanyNameInVietnamese         | CompanyNameInSecondLanguage |
|------------|---------------------------------|-----------------------------|
| 1<br>S0003 | Công ty Cổ phần Yamaka Việt Nam | Vietnam Yamaka              |

**Hình UD-6-1: Thông tin một nhà cung cấp.**

Ứng với trường hợp xóa mẫu tin trong bảng Suppliers, bạn có thể khai báo thủ tục có tên udsDeleteSuppliers như ví dụ UD-3.

**Ví dụ UD-6-3: Khai báo xóa mẫu tin**

```
CREATE PROC udsDeleteSuppliers
  @SupplierId CHAR(5)
AS
  DELETE FROM Suppliers
  WHERE
    SupplierId = CASE @SupplierId
      WHEN '' THEN SupplierId
      ELSE @SupplierId END
GO
```

Bạn có thể gọi thủ tục nội tại có tên udsDeleteSuppliers với hai trường hợp, trường xóa tất cả mẫu tin trong bảng Suppliers thì bạn khai báo như ví dụ UD-6-4.

**Ví dụ UD-6-4: Khai báo xóa tất cả mẫu tin**

```
udsDeleteSuppliers ''
GO
```

Trong trường hợp xóa một mẫu tin trong bảng Suppliers, bạn khai báo gọi thủ tục udsDeleteSuppliers như ví dụ UD-6-5.

**Ví dụ UD-6-5: Khai báo xóa một mẫu tin**

```
udsDeleteSuppliers 'S0010'
GO
```

Nếu cho phép người sử dụng thêm mới hay cập nhật thông tin nhà cung cấp, bạn có thể khai báo thủ tục nội tại dùng chung cho hai trường hợp này dựa và mã nhà cung cấp như ví dụ UD-6-6.

**AA® 270****Ứng dụng:** Sử dụng thủ tục nội tại trong ứng dụng**Ví dụ UD-6-6: Khai báo thêm và cập nhật thông tin nhà cung cấp**

```

CREATE PROC udsInsUpdSuppliers
    @Flag BIT,
    @SupplierID char(5),
    @CompanyNameInVietnamese nvarchar(50),
    @CompanyNameInSecondLanguage varchar(50),
    @ContactName nvarchar(50),
    @ContactTitle nvarchar(50),
    @Address nvarchar(100),
    @ProvinceID char(3),
    @Telephone varchar(20),
    @FaxNumber varchar(10),
    @SupplierTypeID char(3),
    @EmailAddress varchar(50),
    @DueDate smalldatetime,
    @Discontinued bit
AS
    IF @Flag = 0
        INSERT INTO Suppliers
        VALUES (@SupplierID, @CompanyNameInVietnamese,
                @CompanyNameInSecondLanguage, 1, @ContactName,
                @ContactTitle, @Address, @ProvinceID,
                @Telephone, @FaxNumber, @SupplierTypeID,
                @EmailAddress, @DueDate, 0)
    ELSE
        UPDATE Suppliers
        SET CompanyNameInVietnamese =
            @CompanyNameInVietnamese,
            CompanyNameInSecondLanguage =
            @CompanyNameInSecondLanguage,
            ContactName = @ContactName,
            ContactTitle = @ContactTitle,
            Address = @Address,
            ProvinceID = @ProvinceID,
            Telephone = @Telephone,
            FaxNumber = @FaxNumber,
            SupplierTypeID = @SupplierTypeID,
            EmailAddress = @EmailAddress,
            Discontinued = @Discontinued
        WHERE SupplierID = @SupplierID
GO

```

**2.2. Bảng PurchaseInvoiceTypes**

Tương tự như bảng SalesInvoiceTypes, đối với hành động truy vấn dữ liệu cho bảng PurchaseInvoiceTypes, chúng ta cần khai báo thủ tục nội tại như ví dụ UD-7.

**Ứng dụng:** Sử dụng thủ tục nội tại trong ứng dụng

271

**Ví dụ UD-7: Khai báo thủ tục nội tại truy vấn**

```
CREATE PROC udsViewPurchaseInvoiceTypes
    @InvoiceTypeId CHAR(3)
AS
    SELECT * FROM PurchaseInvoiceTypes
    WHERE
        InvoiceTypeId = CASE @InvoiceTypeId
            WHEN '' THEN InvoiceTypeId
            ELSE @InvoiceTypeId END
GO
```

Khi thực thi thủ tục nội tại trên để liệt kê danh sách loại hóa đơn bán hàng, bạn cần sử dụng cú pháp như ví dụ UD-7-1.

**Ví dụ UD-7-1: Khai báo gọi thủ tục nội tại**

```
udsViewPurchaseInvoiceTypes
GO
```

Khi thực thi thủ tục trên, bạn có thể tìm thấy danh sách mẫu tin như hình UD-7.

| InvoiceTypeID | InvoiceTypeName                    |
|---------------|------------------------------------|
| 1             | PI1                                |
| 2             | Hoá đơn mua hàng là Bán thành phẩm |
| 3             | Hoá đơn mua hàng khuyến mãi        |

**Hình UD-7: Loại hóa đơn mua hàng.**

Đối với trường hợp muốn liệt kê một mẫu tin ứng với một mã loại hóa đơn mua hàng được chọn, bạn khai báo gọi thủ tục nội tại như ví dụ UD-7-2.

**Ví dụ UD-7-2: Khai báo liệt kê một loại hóa đơn mua hàng**

```
udsViewPurchaseInvoiceTypes 'PI3'
GO
```

Khi thực thi thủ tục trên, bạn có thể tìm thấy thông tin của loại hóa đơn mua hàng như hình UD-7-1.

| InvoiceTypeID | InvoiceTypeName             |
|---------------|-----------------------------|
| 1             | Hoá đơn mua hàng khuyến mãi |

**Hình UD-7-1: Thông tin một mã loại hóa đơn mua hàng.**

M 272

**Ứng dụng:** Sử dụng thủ tục nội tai trong ứng dụng

Trong trường hợp xóa mẫu tin trong bảng PurchaseInvoiceTypes, bạn có thể khai báo thủ tục có tên udsDeletePurchaseInvoiceTypes như ví dụ UD-7-3.

#### Ví dụ UD-7-3: Khai báo xóa mảng tin

```
CREATE PROC udsDeletePurchaseInvoiceTypes
    @InvoiceTypeId CHAR(3)
AS
    DELETE FROM PurchaseInvoiceTypes
    WHERE
        InvoiceTypeId = CASE @InvoiceTypeId
            WHEN '' THEN InvoiceTypeId
            ELSE @InvoiceTypeId END
GO
```

Bạn có thể xóa loại hóa đơn mua hàng bằng cách gọi thủ tục nội tại có tên udsDeletePurchaseInvoiceTypes với hai trường hợp, đối với trường hợp xóa tất cả mẫu tin trong bảng thì bạn khai báo như ví dụ UD-7-4.

#### **Ví dụ UD-7-4: Khai báo xóa tất cả mảng tin**

```
udsDeletePurchaseInvoiceTypes
GO
```

Nếu muốn xóa một mẫu tin, bạn khai báo gọi thủ tục `udsDeletePurchaseInvoiceTypes` như ví du UD-7-5.

#### **Ví dụ UD-7-5: Khai báo xóa một mảng**

```
udsDeletePurchaseInvoiceTypes 'P0010'  
GO
```

Ngoài ra, tương tự như các bảng khác, khi thêm mới hay cập nhật thông tin loại hóa đơn mua hàng, bạn có thể khai báo thủ tục nội tại dùng chung cho hai trường hợp này dựa và mã loại hóa đơn mua hàng như ví dụ UD-7-6.

#### Ví dụ UD-7-6: Khai báo thêm và cập nhật

```

CREATE PROC udsInsUpdPurchaseInvoiceTypes
    @Flag BIT,
    @InvoiceTypeId char(3),
    @InvoiceTypeNameInVietnamese nvarchar(50),
    @InvoiceTypeNameInSecondLanguage varchar(50)
AS
    IF @Flag = 0
        INSERT INTO PurchaseInvoiceTypes
            VALUES (@InvoiceTypeId,
                    @InvoiceTypeNameInVietnamese,
                    @InvoiceTypeNameInSecondLanguage, 1)
    ELSE
        UPDATE PurchaseInvoiceTypes

```

**Ứng dụng:** Sử dụng thủ tục nội tại trong ứng dụng

273

```

SET InvoiceTypeNameInVietnamese =
@InvoiceTypeNameInVietnamese,
InvoiceTypeNameInSecondLanguage =
@InvoiceTypeNameInSecondLanguage
WHERE InvoiceTypeId = @InvoiceTypeId
GO

```

**2.3. Bảng PurchaseInvoiceBatchs**

Khi truy vấn dữ liệu trong bảng PurchaseInvoiceBatchs, chúng ta cần khai báo thủ tục nội tại như ví dụ UD-8.

**Ví dụ UD-8: Khai báo thủ tục nội tại truy vấn**

```

CREATE PROC udsViewPurchaseInvoiceBatchs
    @InvoiceBatchNo VARCHAR(10)
AS
    SELECT * FROM PurchaseInvoiceBatchs
    WHERE
        InvoiceBatchNo = CASE @InvoiceBatchNo
            WHEN '' THEN InvoiceBatchNo
            ELSE @InvoiceBatchNo END
GO

```

Khi thực thi thủ tục nội tại trên để liệt kê danh sách lô hóa đơn mua hàng, bạn cần sử dụng cú pháp như ví dụ UD-8-1.

**Ví dụ UD-8-1: Khai báo gọi thủ tục nội tại**

```

udsViewPurchaseInvoiceBatchs ''
GO

```

Khi thực thi thủ tục trên, bạn có thể tìm thấy danh sách mẫu tin trong bảng PurchaseInvoiceBatchs như hình UD-8.

|   | InvoiceBatchNo | InvoiceBatchDate    | BatchSize | EntryDate           | UserName |
|---|----------------|---------------------|-----------|---------------------|----------|
| 1 | PBI001         | 2007-10-01 00:00:00 | 0         | 2007-10-31 09:09:00 | khang    |
| 2 | PBI002         | 2007-10-02 00:00:00 | 0         | 2007-10-31 09:09:00 | khang    |
| 3 | PBI003         | 2007-10-03 00:00:00 | 0         | 2007-10-31 09:09:00 | khang    |
| 4 | PBI004         | 2007-10-04 00:00:00 | 0         | 2007-10-31 09:09:00 | khang    |

**Hình UD-8: Danh sách lô hóa đơn mua hàng.**

Nếu liệt kê một mẫu tin ứng với một lô hóa đơn mua hàng được chọn, bạn khai báo gọi thủ tục nội tại như ví dụ UD-8-2.

**Ví dụ UD-8-2: Khai báo liệt kê một lô hóa đơn mua hàng**

udsViewPurchaseInvoiceBatches ISBN0024

3  
GO

Khi thực thi thủ tục trên, bạn có thể tìm thấy thông tin của lô hóa đơn mua hàng như hình IUD-8-1

|   | InvoiceBatchNo | InvoiceBatchDate    | BatchDiscontinued | EntryDate           | UserName |
|---|----------------|---------------------|-------------------|---------------------|----------|
| 1 | PB1001         | 2007-10-01 00:00:00 | 0                 | 2007-10-31 09:09:00 | khang    |

**Hình UD-8-1:** Thông tin một lô hóa đơn mua hàng.

Đối với trường hợp xóa mẫu tin trong bảng PurchaseInvoiceBatchs, bạn có thể khai báo thủ tục có tên udsDeletePurchaseInvoiceBatchs như ví dụ UD-8-3.

#### Ví dụ UD-8-3: Khai báo xóa mảng tip

```
CREATE PROC udsDeletePurchaseInvoiceBatches
    @InvoiceBatchNo VARCHAR(10)
AS
    DELETE FROM PurchaseInvoiceBatches
    WHERE
        InvoiceBatchNo = CASE @InvoiceBatchNo
            WHEN '' THEN InvoiceBatchNo
            ELSE @InvoiceBatchNo END
GO
```

Sau khi tạo thủ tục nội tại udsDeletePurchaseInvoiceBatchs, bạn có thể gọi thủ tục nội tại này với hai trường hợp, nếu xóa tất cả mẫu tin trong bảng ban khai báo như ví du UD-8-4.

#### Ví dụ UD-8-4: Khai báo xóa fast cả mảng tin

```
udsDeletePurchaseInvoiceBatchs GO
```

Nếu xóa một mẫu tin, bạn khai báo gọi thủ tục `udsDeletePurchaseInvoiceBatchs` như ví dụ UD-8-5.

#### **Ví dụ UD-8-5: Khai báo xóa một mảng tin**

```
udsDeletePurchaseInvoiceBatchs 'PBI002 '
GO
```

Trong trường hợp thêm mới hay cập nhật dữ liệu vào bảng PurchaseInvoiceBatchs, bạn có thể khai báo thủ tục nội tại dùng chung cho hai trường hợp này dựa và mã lô hóa đơn mua hàng như ví dụ UD-8-6.

**Ứng dụng: Sử dụng thủ tục nội tại trong ứng dụng**

275

**Ví dụ UD-8-6: Khai báo thêm và cập nhật**

```

CREATE PROC udsInsUpdPurchaseInvoiceBatchs
    @Flag BIT,
    @InvoiceBatchNo VARCHAR(10),
    @InvoiceBatchDate SMALLDATETIME,
    @BatchDiscontinued BIT
AS
    IF @Flag = 0
        INSERT INTO PurchaseInvoiceBatchs
        VALUES (@InvoiceBatchNo, @InvoiceBatchDate,
                1, GETDATE(), CURRENT_USER)
    ELSE
        UPDATE PurchaseInvoiceBatchs
        SET InvoiceBatchDate = @InvoiceBatchDate,
            BatchDiscontinued = @BatchDiscontinued,
            UserName = CURRENT_USER
        WHERE InvoiceBatchNo = @InvoiceBatchNo
GO

```

**2.4. Bảng PurchaseInvoices**

Tương tự như bảng SalesInvoices, khi truy vấn dữ liệu trong bảng PurchaseInvoices, bạn cần khai báo thủ tục nội tại như ví dụ UD-9.

**Ví dụ UD-9: Khai báo thủ tục nội tại truy vấn**

```

CREATE PROC udsViewPurchaseInvoices
    @InvoiceNo VARCHAR(10)
AS
    SELECT * FROM PurchaseInvoices
    WHERE
        InvoiceNo = CASE @InvoiceNo
            WHEN '' THEN InvoiceNo
            ELSE @InvoiceNo END
GO

```

Để liệt kê danh sách hóa đơn bán hàng, bạn thực thi thủ tục nội tại trên với phát biểu gọi thủ tục nội tại như ví dụ UD-9-1.

**Ví dụ UD-9-1: Khai báo gọi thủ tục nội tại**

```

udsViewPurchaseInvoices ''
GO

```

Nếu thực thi thủ tục trên, bạn có thể tìm thấy danh sách mẫu tin ứng với danh sách hóa đơn mua hàng như hình UD-9.



**Ứng dụng:** Sử dụng thủ tục nội tại trong ứng dụng

|   | InvoiceNo  | InvoiceBatch... | DueDate             | InvoiceType... | SupplierID |
|---|------------|-----------------|---------------------|----------------|------------|
| 1 | P100000001 | PBI001          | 2007-10-01 00:00:00 | PB1            | S0001      |
| 2 | P100000002 | PBI001          | 2007-10-01 00:00:00 | PB1            | S0002      |
| 3 | P100000003 | PBI002          | 2007-10-02 00:00:00 | PB1            | S0003      |
| 4 | P100000004 | PBI002          | 2007-10-02 00:00:00 | PB1            | S0001      |
| 5 | P100000005 | PBI002          | 2007-10-02 00:00:00 | PB1            | S0004      |
| 6 | P100000006 | PBI003          | 2007-10-03 00:00:00 | PB1            | S0002      |
| 7 | P100000007 | PBI003          | 2007-10-03 00:00:00 | PB1            | S0003      |
| 8 | P100000008 | PBI004          | 2007-10-04 00:00:00 | PB1            | S0004      |
| 9 | P100000009 | PBI004          | 2007-10-04 00:00:00 | PB1            | S0001      |

**Hình UD-9: Danh sách hóa đơn mua hàng.**

Tuy nhiên, trong một vài trường hợp người sử dụng muốn liệt kê một mẫu tin ứng với một hóa đơn mua hàng với mục đích trình bày hay cập nhật, bạn khai báo gọi thủ tục nội tại như ví dụ UD-9-2.

#### Ví dụ UD-9-2: Khai báo liệt kê một hóa đơn mua hàng

```
udsViewSalesInvoices 'PI00000002'  
GO
```

Khi thực thi thủ tục trên, bạn có thể tìm thấy thông tin của hóa đơn mua hàng như hình UD-9-1

| Results |            | Messages        |                     |                |            |
|---------|------------|-----------------|---------------------|----------------|------------|
|         | InvoiceNo  | InvoiceBatch... | DueDate             | InvoiceType... | SupplierID |
| 1       | P100000002 | PBI001          | 2007-10-01 00:00:00 | PB1            | S0002      |

**Hình UD-9-1:** Thông tin một hóa đơn mua hàng.

Nếu cho phép người sử dụng tìm kiếm hóa đơn mua hàng theo mã nhà cung cấp, ban có thể khai báo thủ tục như ví dụ UD-9-3.

**Ví dụ UD-9-3: Khai báo liệt kê danh sách hóa đơn mua hàng theo mã nhà cung cấp**

```
ma nha cung cap id uud gaud sum rob sed does dash by
CREATE PROC udsViewPurchaseInvoicesBySupplier
    @InvoiceNo VARCHAR(10),
    @SupplierId CHAR(5)
AS
    SELECT * FROM PurchaseInvoices
    WHERE
```

**Ứng dụng:** Sử dụng thủ tục nội tại trong ứng dụng277 

```

InvoiceNo = CASE @InvoiceNo
    WHEN '' THEN InvoiceNo
    ELSE @InvoiceNo END
    AND SupplierId = CASE @SupplierId
        WHEN '' THEN SupplierId
        ELSE @SupplierId END
GO

```

Bạn có thể gọi thủ tục trên với mã số hóa đơn và nhà cung cấp là rỗng như ví dụ UD-9-4.

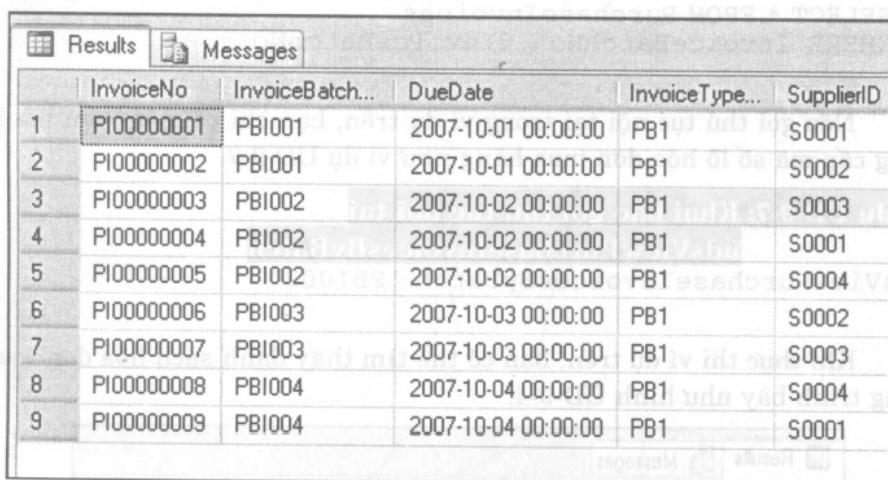
**Ví dụ UD-9-4: Khai báo liệt kê danh sách hóa đơn mua hàng của tất cả nhà cung cấp**

```

udsViewPurchaseInvoicesBySupplier '', ''
GO

```

Nếu thực thi phát biểu trong ví dụ trên, bạn có thể tìm thấy danh sách mẫu tin trình bày như hình UD-9-2.



|   | InvoiceNo  | InvoiceBatch... | DueDate             | InvoiceType... | SupplierID |
|---|------------|-----------------|---------------------|----------------|------------|
| 1 | P100000001 | PBI001          | 2007-10-01 00:00:00 | PB1            | S0001      |
| 2 | P100000002 | PBI001          | 2007-10-01 00:00:00 | PB1            | S0002      |
| 3 | P100000003 | PBI002          | 2007-10-02 00:00:00 | PB1            | S0003      |
| 4 | P100000004 | PBI002          | 2007-10-02 00:00:00 | PB1            | S0001      |
| 5 | P100000005 | PBI002          | 2007-10-02 00:00:00 | PB1            | S0004      |
| 6 | P100000006 | PBI003          | 2007-10-03 00:00:00 | PB1            | S0002      |
| 7 | P100000007 | PBI003          | 2007-10-03 00:00:00 | PB1            | S0003      |
| 8 | P100000008 | PBI004          | 2007-10-04 00:00:00 | PB1            | S0004      |
| 9 | P100000009 | PBI004          | 2007-10-04 00:00:00 | PB1            | S0001      |

**Hình UD-9-2: Danh sách nhà cung cấp và hóa đơn mua hàng.**

Ngoài ra, bạn cũng có thể gọi thủ tục trên với mã nhà cung cấp là S0006 như ví dụ UD-9-5.

**Ví dụ UD-9-5: Khai báo liệt kê danh sách hóa đơn mua hàng cho một nhà cung cấp S0004**

```

udsViewPurchaseInvoicesBySupplier '', 'S0004'
GO

```

Nếu thực thi phát biểu trong ví dụ trên, bạn có thể tìm thấy danh sách mẫu tin trình bày như hình UD-9-3.

|   | InvoiceNo | InvoiceBatch... | DueDate             | InvoiceType... | SupplierID |
|---|-----------|-----------------|---------------------|----------------|------------|
| 1 | P10000005 | PBI002          | 2007-10-02 00:00:00 | PB1            | S0004      |
| 2 | P10000008 | PBI004          | 2007-10-04 00:00:00 | PB1            | S0004      |

Hình UD-9-3: Hóa đơn bán hàng của nhà cung cấp.

Tương tự như phần quản lý hóa đơn bán hàng, chúng ta quản lý hóa đơn mua hàng theo lô, bạn cần khai báo thủ tục nội tại để liệt kê danh sách hóa đơn mua hàng theo từng lô như ví dụ UD-9-6.

Ví dụ UD-9-6: Khai báo liệt kê hóa đơn mua hàng theo lô

**Ví dụ 3B-3-6. Khai báo hết khép kín cho đơn mua hàng**

```
CREATE PROC uspViewPurchaseInvoice  
@InvoiceBatchNo VARCHAR(10)
```

5

```
SELECT * FROM PurchaseInvoices
```

```
SELECT * FROM PurchaseInvoices  
WHERE InvoiceBatchNo = @InvoiceBatchNo
```

60

Nếu gọi thủ tục nội tại trong ví dụ trên, bạn chỉ có một chọn lựa là cung cấp mã số lô hóa đơn mua hàng như ví dụ UD-9-7.

#### Ví dụ UD-9-7: Khai báo gõ thủ tục nội tai

#### **ViewsPurchaseInvoicesByBatch**

`udsViewPurchaseInvoicesByBatch` 'PBT002'

ua  
co

Khi thực thi ví dụ trên, bạn có thể tìm thấy danh sách hóa đơn mua hàng trình bày như hình UD-9-4.

|   | InvoiceNo  | InvoiceBatch... | DueDate             | InvoiceType... | SupplierID |
|---|------------|-----------------|---------------------|----------------|------------|
| 1 | P100000003 | PBI002          | 2007-10-02 00:00:00 | PB1            | S0003      |
| 2 | P100000004 | PBI002          | 2007-10-02 00:00:00 | PB1            | S0001      |
| 3 | P100000005 | PBI002          | 2007-10-02 00:00:00 | PB1            | S0004      |

Hình UD-9-4: Danh sách hóa đơn mua hàng theo lô

Ứng với trường hợp xóa mẫu tin trong bảng PurchaseInvoices, bạn có thể khai báo thủ tục có tên udsDeletePurchaseInvoices, như ví dụ UID-9-8.

#### **Ví dụ JID-9-8: Khai báo xóa mẫu tin**

Viết CD-9-8: Khai báo xóa mẫu tin  
CREATE PROC uspDeletePurchaseInvoice

```
CREATE PROC ussDeletePur  
@InvoiceNo VARCHAR(10)
```

**Ứng dụng:** Sử dụng thủ tục nội tại trong ứng dụng

279 

```
AS  
    DELETE FROM PurchaseInvoices  
    WHERE  
        InvoiceNo = CASE @InvoiceNo  
            WHEN '' THEN InvoiceNo  
            ELSE @InvoiceNo END  
GO
```

Bạn có thể gọi thủ tục nội tại có tên udsDeletePurchaseInvoices với hai trường hợp, trường hợp xóa tất cả mẫu tin trong bảng bạn khai báo như ví dụ UD-9-9.

#### **Ví dụ UD-9-9: Khai báo xóa tất cả mảng tin**

```
udsDeletePurchaseInvoices ''  
GO
```

Đối với trường hợp xóa một mẫu tin, bạn khai báo gọi thủ tục `udsDeletePurchaseInvoices` như ví dụ UD-9-10.

**Ví dụ UD-9-10: Khai báo xóa một mảng tin**

```
udsDeletePurchaseInvoices 'PI00000015'  
GO
```

Ngoài ra, bạn cũng có thể khai báo thủ tục nội tại dùng cho trường hợp thêm mới hay cập nhật thông tin hóa đơn mua hàng như ví dụ UD-9-11.

#### Ví dụ UD-9-11: Khai báo thêm và cập nhật

```
CREATE PROC udsInsUpdPurchaseInvoices
    @Flag BIT,
    @InvoiceNo VARCHAR(10),
    @InvoiceBatchNo VARCHAR(10),
    @DueDate SMALLDATETIME,
    @InvoiceTypeId CHAR(3),
    @SupplierId CHAR(5),
    @CurrencyId CHAR(3),
    @ExchangeRate DECIMAL (18, 0),
    @DescriptionInVietnamese NVARCHAR(150),
    @DescriptionInSecondLanguage NVARCHAR(150),
    @InvoiceDiscontinued BIT
```

```
AS
IF @Flag = 0
    INSERT INTO PurchaseInvoices
    VALUES (@InvoiceNo, @InvoiceBatchNo,
            @DueDate, @InvoiceTypeId, @SupplierId,
            @CurrencyId, @ExchangeRate,
            @DescriptionInVietnamese,
            @DescriptionInSecondLanguage,
            1, 1, GETDATE(), CURRENT_USER)
ELSE
    UPDATE PurchaseInvoices
```

M® 280

**Ứng dụng:** Sử dụng thủ tục nội tại trong ứng dụng

```

SET DueDate = @DueDate,
InvoiceTypeId = @InvoiceTypeId,
SupplierId = @SupplierId,
CurrencyId = @CurrencyId,
ExchangeRate = @ExchangeRate,
DescriptionInVietnamese = @DescriptionInVietnamese,
DescriptionInSecondLanguage =
@DescriptionInSecondLanguage,
InvoiceDiscontinued = @InvoiceDiscontinued,
UserName = CURRENT_USER
WHERE InvoiceNo = @InvoiceNo
GO

```

**2.5. Bảng PurchaseInvoiceDetails**

Tương tự như các trường hợp bảng hóa đơn mua hàng (PurchaseInvoices), khi truy vấn dữ liệu trong bảng PurchaseInvoiceDetails, bạn cần khai báo thủ tục nội tại như ví dụ UD-10.

**Ví dụ UD-10: Khai báo thủ tục nội tại truy vấn**

```

CREATE PROC udsViewPurchaseInvoiceDetails
    @InvoiceNo VARCHAR(10)
AS
    SELECT * FROM PurchaseInvoiceDetails
    WHERE InvoiceNo = @InvoiceNo
GO

```

Khi thực thi thủ tục nội tại trên, để liệt kê danh sách chi tiết hóa đơn mua hàng, bạn cần sử dụng phát biểu gọi thủ tục nội tại như ví dụ UD-10-1.

**Ví dụ UD-10-1: Khai báo gọi thủ tục nội tại**

```

udsViewPurchaseInvoiceDetails 'PI00000002'
GO

```

Nếu thực thi thủ tục trên, bạn có thể tìm thấy danh sách mẫu tin ứng với danh sách hóa đơn mua hàng như hình UD-10.

|   | OrdinalNumber | InvoiceNo  | ProductID | Quantity | Price | Discount | VATRate |
|---|---------------|------------|-----------|----------|-------|----------|---------|
| 1 | 1             | PI00000002 | P00001    | 200      | 8500  | 0        | 10      |
| 2 | 2             | PI00000002 | P00002    | 250      | 8500  | 0        | 10      |
| 3 | 3             | PI00000002 | P00003    | 150      | 8500  | 0        | 10      |
| 4 | 4             | PI00000002 | P00004    | 50       | 9500  | 0        | 10      |

**Hình UD-10: Danh sách chi tiết hóa đơn mua hàng.**

**Ứng dụng:** Sử dụng thủ tục nội tại trong ứng dụng

281  MITSUBISHI

Tuy nhiên, trong trường hợp người sử dụng muốn liệt kê một mẫu tin ứng với một hóa đơn mua hàng bao gồm thông tin chi tiết, bạn khai báo thủ tục nội tại như ví dụ UD-10-2.

**Ví dụ UD-10-2: Khai báo liệt kê một hóa đơn mua hàng**

```
CREATE PROC udsViewPurchaseInvoiceDetail
    @InvoiceNo VARCHAR(10),
    @ProductId VARCHAR(10),
    @OrdinalNumber tinyint
AS
    SELECT * FROM PurchaseInvoiceDetails
    WHERE InvoiceNo = @InvoiceNo
        AND ProductId = @ProductId
        AND OrdinalNumber = @OrdinalNumber
```

Để liệt kê tất cả thông tin của hóa đơn mua hàng, bạn cần khai báo  
goi thủ tục nội tai vừa tao trong ví du trên như ví dụ UD-10-3.

Ví dụ UD-10-3: Khai báo gói thủ tục nội tại udsViewPurchaseInvoice

```
udsViewPurchaseInvoiceDetail  
'PI00000002', 'P00001', 1  
GO
```

Khi thực thi thủ tục trên với 3 tham số ứng với một mẫu tin trong chi tiết hóa đơn mua hàng, bạn có thể tìm thấy thông tin chi tiết như hình UD-10-1.

| Order Details |               |            |           |          |       |          |         |
|---------------|---------------|------------|-----------|----------|-------|----------|---------|
|               | OrdinalNumber | InvoiceNo  | ProductID | Quantity | Price | Discount | VATRate |
| 1             | 1             | P100000002 | P00001    | 200      | 8500  | 0        | 10      |

**Hình UD-10-1:** Thông tin chi tiết của một hóa đơn mua hàng.

Lưu ý: Mỗi khi xóa mẫu tin trong bảng PurchaseInvoices thì tất cả mẫu tin liên quan nằm trong bảng PurchaseInvoiceDetails sẽ tự động xóa theo, do chúng ta cài đặt Cascade khi thiết lập quan hệ giữa chúng.

Tuy nhiên, để cho phép người sử dụng xóa một mẫu tin trong bảng PurchaseInvoiceDetails, bạn có thể khai báo thủ tục có tên udsDeletePurchaseInvoiceDetails như ví dụ UD-10-4.

#### Ví dụ UD-10-4: Khai báo xóa một mảng tin

```
CREATE PROC udsDeletePurchaseInvoiceDetail
    @InvoiceNo VARCHAR(10),
    @ProductId VARCHAR(10).
```

282

**Ứng dụng:** Sử dụng thủ tục nội tại trong ứng dụng

```
@OrdinalNumber tinyint  
AS  
    DELETE FROM PurchaseInvoiceDetails  
    WHERE InvoiceNo = @InvoiceNo  
        AND ProductId = @ProductId  
        AND OrdinalNumber = @OrdinalNumber  
GO
```

Bạn có thể gọi thủ tục nội tại có tên udsDeletePurchaseInvoiceDetail như ví dụ UD-10-5.

#### Ví dụ UD-10-5: Khai báo xóa một mảng tin

```
udsDeletePurchaseInvoiceDetail  
'PI00000001', 'P00001', 1  
GO
```

Bạn cũng khai báo thủ tục nội dung cho trường hợp thêm mới hay cập nhật thông tin chi tiết hóa đơn mua hàng như ví dụ UD-10-6

#### Ví dụ UD-10-6: Khai báo thêm và cập nhật

```
CREATE PROC udsInsUpdPurchaseInvoiceDetails
    @Flag BIT,
    @InvoiceNo VARCHAR(10),
    @ProductId VARCHAR(10),
    @OrdinalNumber TINYINT,
    @Quantity DECIMAL (18, 0),
    @Price DECIMAL (18, 0),
    @Discount DECIMAL (18, 0),
    @VATRate DECIMAL (18, 0)
AS
    IF @Flag = 0
        INSERT INTO PurchaseInvoiceDetails
        VALUES (@OrdinalNumber, @InvoiceNo,
                @ProductId, @Quantity, @Price,
                @Discount, @VATRate)
    ELSE
        UPDATE PurchaseInvoiceDetails
        SET Quantity = @Quantity,
            Price = @Price,
            Discount = @Discount,
            VATRate = @VATRate
        WHERE InvoiceNo = @InvoiceNo
        AND ProductId = @ProductId
        AND OrdinalNumber = @OrdinalNumber
GO
```

**Ứng dụng:** Sử dụng thủ tục nội tại trong ứng dụng

283 ®

#### **2.6. Thủ tục hỗ trợ làm báo cáo**

Theo yêu cầu về thống kê thông tin mua hàng, bạn có thể khai báo một số thủ tục nội tại nhằm cung cấp dữ liệu tương ứng với các tiêu chí liệt kê theo thời gian, nhà cung cấp hay sản phẩm.

Chẳng hạn, bạn khai báo thủ tục nội tại cho phép liệt kê doanh thu mua hàng theo ngày như ví du UD-11.

#### Ví dụ UD-11: Báo cáo chi phí mua hàng theo ngày

```
CREATE PROC udsPurchaseByDate
AS
SELECT Convert(char(10), DueDate, 103) AS SalesDate,
SUM(Quantity*Price*(1+VATRate/100)-Discount) AS PurchaseAmount
FROM PurchaseInvoices S, PurchaseInvoiceDetails D
WHERE S.InvoiceNo = D.InvoiceNo
GROUP BY Convert(char(10), DueDate, 103)
GO
```

Bạn có thể gọi thủ tục trên và kết quả là danh sách chi phí mua hàng trình bày như hình UD-11.

| udsPurchaseByDate |            |
|-------------------|------------|
| GO                |            |
|                   | Results    |
| 1                 | 01/10/2007 |
| 2                 | 02/10/2007 |
| 3                 | 03/10/2007 |
| 4                 | 04/10/2007 |

**Hình UD-11:** Chi phí mua hàng theo ngày.

Tương tự như trên, bạn có thể khai báo thủ tục nội tại để liệt kê chi phí mua hàng theo tuần trong tháng hiện hành như ví dụ UD-11-1.

#### **Ví dụ UD-11-1: Báo cáo chi phí mua hàng theo tuần**

CREATE PROCEDURE PurchaseByWeek

CR  
AS

```
AS  
SELECT DATEPART(ww, DueDate) AS SalesWeek,
```

