



Ks. NGUYỄN NAM THUẬN - LỮ ĐỨC HÀO



Quản lý cơ sở dữ liệu **Với** **Microsoft** **SQL Server 2005**

- CÁC TÍNH NĂNG QUẢN TRỊ CƠ SỞ DỮ LIỆU
- CÁC CÔNG CỤ QUẢN TRỊ VÀ PHÁT TRIỂN CƠ SỞ DỮ LIỆU
- CÁC CƠ CẤU QUẢN LÝ MỚI
- CÁC TÍNH NĂNG PHÁT TRIỂN CƠ SỞ DỮ LIỆU
- CÁC TÍNH NĂNG KINH DOANH THÔNG MINH

KS : NGUYỄN NAM THUẬN
LỮ ĐỨC HÀO

Quản lý cơ sở dữ liệu với

Microsoft

SQL Server 2005

- Các tính năng quản trị cơ sở dữ liệu
- Các công cụ quản trị và phát triển cơ sở dữ liệu
- Các cơ cấu quản lý mới
- Các tính năng phát triển cơ sở dữ liệu
- Các tính năng kinh doanh thông minh

NHÀ XUẤT BẢN GIAO THÔNG VẬN TẢI

LỜI NÓI ĐẦU

SQL Server 2005 là một ấn bản có nhiều tính năng, cung cấp một loạt công cụ mới cho nhà quản trị cơ sở dữ liệu cũng như cho nhà phát triển cơ sở dữ liệu. Cuốn sách "*Quản lý cơ sở dữ liệu với Microsoft SQL Server 2005*" được biên soạn nhằm giúp những người quản lý IT, các nhà quản trị cơ sở dữ liệu, nhà phát triển cơ sở dữ liệu, những người tích hợp các hệ thống và các chuyên viên tư vấn theo kịp các tính mới quan trọng nhất trong SQL Server 2005. Cuốn sách này cũng giúp bạn hiểu rõ SQL Server 2005 khác với các ấn bản trước ở những điểm nào và đưa ra những thông tin mà bạn có thể sử dụng để đánh giá lợi ích của việc chọn phiên bản mới này.

Sách gồm 3 phần:

Phần I: Các tính năng quản trị cơ sở dữ liệu. Trong phần này, bạn sẽ tìm hiểu những tính năng quan trọng nhất đối với người quản trị cơ sở dữ liệu. Bạn sẽ được giới thiệu về SQL Workbench mới, đem lại cho người quản trị một kinh nghiệm quản lý hoàn toàn mới. Ngoài ra, phần này cũng đề cập đến các tính năng bảo vệ, tính khả dụng và khả năng phục hồi mà Microsoft đã đưa vào SQL Server 2005.

Phần II: Các tính năng phát triển cơ sở dữ liệu. Phần này trình bày các tính năng mới trong lĩnh vực phát triển cơ sở dữ liệu như các tính năng lập trình, các dịch vụ thông báo, SQL Server Service Broker và sự hợp nhất XML.

Phần III: Các tính năng kinh doanh thông minh. Phần này đề cập đến lĩnh vực lớn và quan trọng trong các cải tiến của Microsoft SQL Server 2005. Với SQL Server 2005, tính năng SQL Service Reporting Services đã được đưa vào như là một phần hợp nhất của bộ SQL Server cơ sở, các tính năng định dạng và triển khai báo cáo của Reporting Service cũng được bao hàm. Ngoài ra còn có nhiều cải tiến khác như sự cải tiến của bộ máy Analysis Services, sự quản lý và sự phát triển.

Với nội dung nêu trên và cách bố cục theo từng chương và đề mục kèm hình ảnh minh họa, chúng tôi hy vọng cuốn sách này sẽ là một tài liệu tham khảo thật hữu ích cho quý bạn đọc. Rất mong đón nhận sự đóng góp ý kiến xây dựng từ quý bạn đọc về những sai sót khó tránh khỏi trong quá trình biên soạn. Xin chân thành cảm ơn.

Tác giả.

PHẦN I

CÁC TÍNH NĂNG QUẢN TRỊ CƠ SỞ DỮ LIỆU

CHƯƠNG I

Các tính năng của bộ máy lưu trữ và kiến trúc cơ sở dữ liệu

Dựa vào kiểu kiến trúc cơ bản được thiết lập với SQL Server 7, Microsoft đã tiếp tục thực hiện các phần cải tiến mới trong bộ máy SQL Server. Theo định luật Moore, phần cứng đã liên tục tăng gấp đôi về dung lượng cứ khoảng 18 tháng một lần, trong khi cùng lúc giá phần cứng lại liên tục giảm. Tuy nhiên, trong khi phần cứng ngày càng rẻ hơn thì chi phí mà mọi người bỏ ra lại tăng lên. Trong ấn bản SQL Server 2005, Microsoft đã đưa ra những cải tiến cho phép SQL Server tận dụng các thế hệ mới nhất của phần cứng hiệu suất cao, cung cấp cho nó khả năng đáp ứng nhu cầu cao nhất của công ty hay doanh nghiệp. Cùng lúc, công ty đã thêm vào các tính năng làm cho SQL Server dễ quản lý hơn và cũng đưa ra những tính năng mới nhằm cho phép bạn tận dụng tối đa SQL Server vào công việc tổ chức của bạn. Trong chương này, bạn sẽ được giới thiệu một số cải tiến quan trọng nhất về kiến trúc mà Microsoft đã thực hiện cho SQL Server 2005, mục đích là để bạn thấy rõ những tính năng này có thể nâng cao hiệu suất và độ tin cậy của môi trường cơ sở dữ liệu như thế nào.

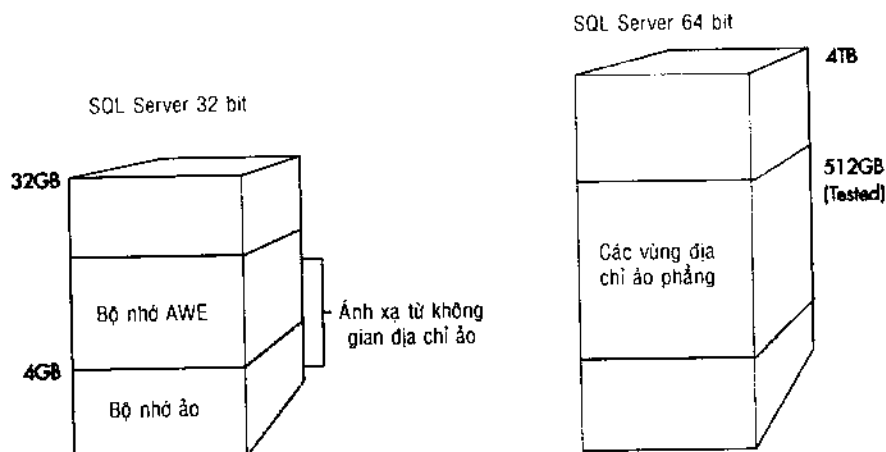
Hỗ trợ phần cứng mới

Một trong những tính năng mới cho phép SQL Server 2005, một bộ máy cơ sở dữ liệu (database engine) đạt được các mức thực thi ngang bằng các cơ sở dữ liệu UNIX lớn nhất là sự hỗ trợ phần cứng cải tiến của nó. Bộ nhớ luôn là một trong những yếu tố quan trọng nhất đóng góp vào hiệu suất của cơ sở dữ liệu, và các cơ sở dữ liệu UNIX lớn đầy các mức hiệu suất TPC-C không nhóm cụm lên hàng đầu nhờ dùng một nền hệ điều hành và phần cứng 64 bit. Khi một nền 64 bit tương đương trở nên có sẵn cho SQL Server ở dạng bộ xử lý Intel Itanium2 64 bit và sự hỗ trợ 64 bit gốc của Windows Server 2003, SQL Server lập tức nhảy lên đứng đầu trong các chuẩn TPC-C. SQL Server 2005 thừa kế sự hỗ

trợ 64 bit gốc từ Windows Server 2003 cùng với khả năng hỗ trợ Non-Uniform Memory Architecture (NUMA). Được xây dựng trên nền Windows Server 2003, SQL Server 2005 cung cấp sự hỗ trợ 64 bit cho cả kiến trúc Itanium IA-64 và kiến trúc AMD x64.

Sự hỗ trợ 64 bit gốc

Khi chạy trên Windows Server 2003 dành cho các hệ thống Itanium 64 bit, SQL Server 2005 hỗ trợ các bộ xử lý Itanium và Itanium2 của Intel. Khi chạy trên SQL Server 2005 dành cho các hệ thống mở rộng 64 bit (64-bit Extended Systems), SQL Server 2005 hỗ trợ các bộ xử lý Opteron và Athlon 64 bit của AMD cũng như Xeon with Intel Extended Memory 64 Technology (EMT 64) của Intel. SQL Server 2005 hoàn toàn hỗ trợ cả hai nền phần cứng 32 bit và 64 bit cho mọi dịch vụ lớn của nó, bao gồm SQL Server Engine, Analysis Services, SQL Agent và Reporting Services. Ưu điểm thật sự khi chuyển một cơ sở dữ liệu sang nền 64 bit không phải là công suất xử lý nhanh hơn mà ưu điểm thật sự nằm ở bộ nhớ có thể ghi địa chỉ gia tăng đáng kể. Hình 1.1 minh họa sự so sánh giữa bộ nhớ có thể ghi địa chỉ tối đa cho SQL Server 32 bit và phiên bản 64 bit của SQL Server 2005.



Hình 1.1 So sánh sự ghi địa chỉ bộ nhớ 32 bit và 64 bit

Kiểu kiến trúc 32 bit gốc có giới hạn tối đa là 4GB bộ nhớ có thể ghi địa chỉ. Trong Windows, giới hạn 4GB này được chia đều giữa hệ điều hành và các ứng dụng. Nói cách khác, 2GB dành cho hệ điều hành Windows và 2GB còn lại dành cho các ứng dụng. Nhờ vào sự hỗ trợ Advanced Windowing Extensions (AWE) trong phiên bản 32 bit của Windows, phiên bản 32 bit của SQL Server có thể ghi địa chỉ tối đa 32GB RAM. Tuy sự gia tăng này là thiết thực, nhưng vẫn còn chi phí

phân trang để đạt đến các trang bộ nhớ thích hợp. Sự thực thì 64 bit gốc đã hầu như loại bỏ sự ràng buộc bộ nhớ bằng cách tăng bộ nhớ có thể ghi địa chỉ tối đa lên 32TB. Hiện tại không có hệ thống sản xuất nào hỗ trợ bất kỳ nơi nào gần lượng RAM vật lý này. Lượng RAM vật lý tối đa mà trên đó SQL Server đã được thử nghiệm vào lúc ấn hành là 512GB. Với ấn bản sắp tới của service pack dành cho Windows Server 2003, bộ nhớ được hỗ trợ tối đa hy vọng sẽ lên đến 1TB.

Sự hỗ trợ NUMA

Một tính năng mới khác có ảnh hưởng mạnh đến hệ điều hành Windows Server 2003 là sự hỗ trợ kiểu kiến trúc bộ nhớ không đồng bộ (Non-Uniform Memory Architecture, gọi tắt là NUMA). NUMA là một kiến trúc hệ thống được sử dụng bởi một số nhà sản xuất hệ thống như IBM và Unisys, nó quản lý việc sử dụng bộ nhớ và CPU trong các hệ thống có nhiều bộ xử lý một cách hiệu quả hơn kiến trúc SMP tiêu biểu. Trong các hệ thống SMP chuẩn, khi tốc độ và số lượng bộ xử lý tăng lên, sự cạnh tranh giữa các bộ xử lý để truy cập vào bộ nhớ tạo ra sự tranh giành bus. Điều này dẫn đến sự trì hoãn trong việc xử lý và gây cản trở cho khả năng mở rộng của một hệ thống. Kết quả là các hệ thống SMP không mở rộng theo tuyến tính khi số lượng bộ xử lý tăng lên. Kiểu kiến trúc NUMA được thiết kế để giải quyết vấn đề này trong các hệ thống SMP, trong đó các bộ xử lý có thể truy cập dữ liệu trong RAM nhanh hơn khả năng mà bus hệ thống và RAM cung cấp. Kiến trúc NUMA nhóm CPU và bộ nhớ vào các vỏ cục bộ của nhiều bộ xử lý. Các vỏ này được nối nhau qua một bus mở rộng vốn chuyển sự lưu thông dữ liệu qua vỏ (pods). Kiểu sắp xếp vỏ này đã giải quyết vấn đề tranh giành bằng cách giới hạn số lượng CPU cạnh tranh để truy cập vào bộ nhớ. Để nhận ra lợi ích lớn nhất từ kiểu kiến trúc này, cả hệ điều hành lẫn các trình ứng dụng phải được thiết kế nhằm giảm thiểu sự lưu thông dữ liệu qua vỏ và tăng tối đa sự truy cập bộ nhớ trong vỏ nhanh hơn. Nếu hệ điều hành và ứng dụng được thiết kế chính xác, kiến trúc NUMA cho phép mở rộng gần như tuyến tính khi nhiều bộ xử lý hơn được thêm vào. Windows Server 2003 và SQL Server 2005 hợp nhất các cải tiến về kiểu kiến trúc để tăng mức độ định vị các luồng và bộ nhớ trong cùng vỏ.

Sự hỗ trợ siêu phân luồng

SQL Server 2005 cũng tận dụng siêu phân luồng (hyper-threading), một phần nhờ vào sự hỗ trợ mới cho siêu phân luồng mà Microsoft đã thêm vào SQL Server 2003. Siêu phân luồng là một công nghệ CPU do Intel phát triển tạo ra hai bộ xử lý lô gíc cho mỗi bộ xử lý vật lý trong một hệ thống. Mỗi bộ xử lý lô gíc có khả năng thực thi đồng thời các luồng (thread) riêng biệt. Mục tiêu của siêu phân luồng là cung cấp sự

tiêu thụ tài nguyên tốt hơn cho các ứng dụng đa luồng hay nhiều ứng dụng đang chạy trên một máy. Trong khi siêu phân luồng cung cấp tiềm năng gia tăng công suất trên một server (máy chủ) thì các bộ xử lý lại tranh giành các tài nguyên hệ thống chẳng hạn như dữ liệu trong cache của bộ xử lý. Sự tranh giành tài nguyên này ngăn chặn một CPU siêu phân luồng thực thi cũng như một hệ thống tương đương sử dụng hai CPU vật lý.

SQL Server 2005 tận dụng hai lợi ích chính từ sự hỗ trợ siêu phân luồng của Windows Server 2003. Trước tiên, không giống như Windows 2000, Windows Server 2003 chỉ tính các bộ xử lý vật lý cho các mục đích đăng ký. Do đó, một bộ xử lý đơn sử dụng công nghệ siêu phân luồng (hyper-threading) được đăng ký đúng là một bộ xử lý đơn hơn là một bộ xử lý kép, như nó đã từng được như vậy trong Windows 2000. Kế tiếp, Windows Server 2003 cung cấp lịch trình luồng cải tiến cho các hệ thống được siêu phân luồng. Các thay đổi này dẫn đến sự thực thi tốt hơn cho các ứng dụng đa luồng như SQL Server.

Bộ máy SQL Server

Mục đầu tiên của chương này đã cho bạn thấy bức tranh lớn hơn của sự hỗ trợ phần cứng mới trong SQL Server 2005, mục kế tiếp này sẽ đi sâu vào một số cải tiến quan trọng nhất mà Microsoft đã thực hiện cho chính bộ máy SQL Server.

Sự hợp nhất .NET Framework

Cải tiến bộ máy SQL Server có ý nghĩa nhất trong SQL Server 2005 là sự hợp nhất của Microsoft .NET Framework. Sự hợp nhất của .NET Framework với SQL Server đã mở rộng tính năng của SQL Server 2005 bằng cách cho phép phát triển các thủ tục lưu trữ, các hàm do người dùng ấn định, các bộ kích khởi, các hàm gộp và các loại được chỉ định bởi người dùng nhờ sử dụng bất kỳ ngôn ngữ .NET, chẳng hạn như VB.NET, C#, hay J#. Sự hợp nhất của .NET CLR với SQL Server 2005 không chỉ ở bề ngoài, vì bộ máy cơ sở dữ liệu SQL Server còn làm chủ CLR trong tiến trình. Bạn sẽ tìm hiểu thêm về sự hợp nhất của .NET Framework với SQL Server 2005 trong chương 4.

Sự hỗ trợ nhiều instance cải tiến

Một cải tiến quan trọng khác trong Enterprise Edition của SQL Server 2005 là sự hỗ trợ lên đến 50 instance (thể hiện). Con số này đã tăng từ tối đa 16 instance được hỗ trợ trong SQL Server 2000. Đây là một cải tiến đặc biệt có ý nghĩa để kiểm soát các nhà cung cấp, những người thuê bao nhiều dịch vụ SQL Server như là một phần trong việc cung cấp dịch vụ Web của họ.

Các kiểu dữ liệu mới

SQL Server 2005 cũng hỗ trợ nhiều kiểu dữ liệu mới. Trong khi sự hợp nhất của .NET Framework cho phép hỗ trợ các kiểu do người dùng ấn định, SQL Server 2005 cũng cung cấp hai kiểu dữ liệu gốc mới khác: các kiểu dữ liệu varbinary (max) và XML. Kiểu dữ liệu varbinary (max) cung cấp một phương pháp mới để sử dụng LOBs với SQL Server. Không giống như các kiểu dữ liệu Image và text cũ hơn, kiểu dữ liệu varbinary (max) mới có thể được sử dụng như là một biến và bằng lập trình nó có thể được xem như là các kiểu dữ liệu nhỏ hơn, cho phép thích ứng với những tình huống sử dụng dễ dàng hơn và nhất quán hơn.

Kiểu dữ liệu XML mới, dựa vào kiểu dữ liệu varbinary (max), cho phép bạn chứa các tài liệu XML trong cơ sở dữ liệu (database). Tuy nhiên, không giống như kiểu dữ liệu varbinary (max), kiểu dữ liệu XML mới được thiết kế cho dữ liệu XML và hỗ trợ sự xác nhận lược đồ của các tài liệu XML. Bạn có thể tìm hiểu thêm về các kiểu dữ liệu mới được hỗ trợ với SQL Server 2005 ở chương 7.

Database Snapshots và Mirroring

Database Mirroring giúp tránh lỗi cơ sở dữ liệu bằng cách cho SQL Server 2005 một khả năng phòng hờ cơ sở dữ liệu khẩn. Database Mirroring là một công nghệ khả dụng cấp cơ sở dữ liệu hoạt động với mọi phần cứng chuẩn được hỗ trợ bởi SQL Server. Không cần bất kỳ sự lưu trữ chia sẻ giữa server chính và server được tạo đối xứng gương (mirrored server) và cũng không có bất kỳ giới hạn nào về khoảng cách. Database Mirroring hoạt động bằng cách gửi các log (sổ nhật ký) giao dịch giữa server chính và server phản chiếu, về cơ bản làm cho tính năng Database Mirroring mới trở thành một ứng dụng chuyển log thời gian thực. Database Mirroring có thể được cài đặt với một cơ sở dữ liệu đơn hoặc nó có thể được cài đặt cho nhiều cơ sở dữ liệu.

Database Snapshots cung cấp một snapshot (ảnh chụp nhanh) chỉ đọc của một cơ sở dữ liệu tại một thời điểm nhất định. Database Snapshots thích hợp nhất cho việc tạo các bản sao của một cơ sở dữ liệu để báo cáo hay cho việc tạo các bản sao dự phòng của một cơ sở dữ liệu mà bạn có thể sử dụng để cuộn một cơ sở dữ liệu sản xuất trở lại một trạng thái trước. Database Snapshots có thể được kết hợp với Database Mirroring để tạo ra một server báo cáo dựa vào dữ liệu nằm trên server được phản chiếu (hay tạo đối xứng gương). Dữ liệu trên server được phản chiếu không thể được truy cập trực tiếp bởi vì server đang phản chiếu luôn ở chế độ phục hồi. Tuy nhiên, bạn có thể tạo một Database Snapshots trên cơ sở dữ liệu được phản chiếu và sau đó truy cập khung xem cơ sở dữ liệu để báo cáo. Bạn có thể tìm hiểu thêm về các khung xem cơ sở dữ liệu và sự phản chiếu (mirroring) ở chương 3.

Sự hỗ trợ HTTP gốc

Một trong những cải tiến quan trọng khác cho bộ máy cơ sở dữ liệu SQL Server là sự bổ sung phần hỗ trợ HTTP gốc vào chính bộ máy. Khả năng của SQL Server trong việc xử lý các yêu cầu HTTP đến cho phép SQL Server cung cấp sự thực thi câu lệnh SQL và sự viện dẫn thủ tục lưu trữ qua giao thức SOAP. Điều này có nghĩa là SQL Server 2005 có thể xử lý các yêu cầu dịch vụ Web đến mà không cần sự hiện diện của IIS hay một web server khác. Sự hỗ trợ HTTP mới đã cho SQL Server có các khả năng lắng nghe HTTP gốc, bao gồm khả năng hỗ trợ các điểm cuối HTTP đang chỉ định URL, cổng và các yêu cầu sẽ được hỗ trợ. SQL Server cũng có thể xuất bản các dịch vụ Web ở dạng ngôn ngữ mô tả dịch vụ Web (Web Services Description Language, gọi tắt là WSDL) cho các điểm cuối. Sự hỗ trợ HTTP của SQL Server là tuân theo các chuẩn, hỗ trợ SOAP 1.0 và 1.2 cũng như WSDL 1.1. Tính năng hỗ trợ HTTP gốc mới này cũng hỗ trợ cả sự xác thực Windows và SQL Server cũng như SSL. Để tính năng mới này tương thích hơn với sự lập trình giữa tầng, các thủ tục lưu trữ có thể trả về các tập hợp kết quả ở dạng một ADO.NET DataSet.

Các event server và bộ kích khởi DDL

Các tính năng Server Event và DDL Triggers mới của SQL Server 2005 cho phép bạn phản hồi bằng lập trình các thay đổi trong hệ thống. Tuy cả hai tính năng mới này có thể thực hiện các chức năng tương tự nhưng chúng được thực thi hoàn toàn khác nhau. Giống như các bộ kích khởi (trigger) DML chuẩn, các bộ kích khởi DDL là những event đồng bộ thực thi các thủ tục lưu trữ. Bạn sẽ tìm hiểu thêm về các bộ kích khởi DDL ở chương 4.

Ngược lại, các event server thì không đồng bộ. Trong mô hình event server, server gửi một event (sự kiện) đến một SQL Server Broker Service, và sau đó một người dùng có thể truy xuất event đó một cách độc lập. Bản thân event được ghi ở dạng dữ liệu XML. Không có cách nào để cuộn lại một event, và một event có thể bị bỏ qua nếu không có người dùng nào truy xuất nó. Khi event xảy ra, event hệ thống được kích khởi, thông báo cho bạn về event đó hay tùy ý thực thi một thường trình mã. Ví dụ sau đây minh họa cú pháp dùng để cài đặt một thông báo event:

```
CREATE EVENT NOTIFICATION MyDDLEvents
```

```
ON SERVER FOR DDL_STATEMENTS TO SERVICE MyDDL_log
```

Ví dụ này tạo ra một event mới, đặt tên cho thông báo event là MyDDLEvents và đính kèm event với câu lệnh DDL. Mệnh đề TO SERVICE chỉ định SQL Broker Service tên MyDDL_log sẽ là người

nhận của các event. Bạn sẽ tìm hiểu thêm về SQL Server Broker ở chương 6.

Các cải tiến file dữ liệu cơ sở dữ liệu

SQL Server 2005 bây giờ hỗ trợ khả năng thay đổi đường dẫn của các file log và dữ liệu của một cơ sở dữ liệu nhờ dùng lệnh ALTERDATABASE. SQL Server 2000 đã cung cấp khả năng di chuyển các file cho cơ sở dữ liệu tempdb, nhưng điều này chưa được phép với bất kỳ cơ sở dữ liệu khác. Như bạn có thể mong đợi, SQL Server 2005 hỗ trợ sự di chuyển các file chỉ ở dạng một hoạt động ngoại tuyến. Ví dụ sau đây minh họa cú pháp câu lệnh ALTERDATABASE mới:

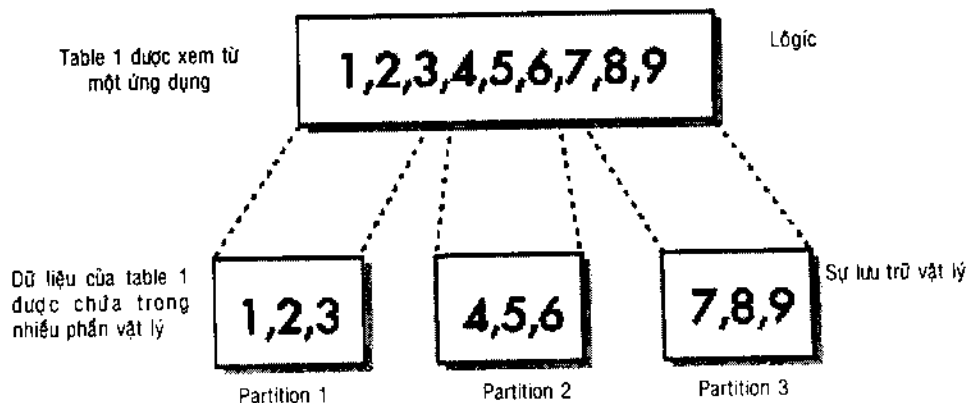
```
ALTER DATABASE <database_name>
```

```
MODIFY FILE(name=<'data_file_name'>, filename=<'new path'>)
```

Sự chia phần dữ liệu

Một cải tiến mới khác trong SQL Server 2005 là khả năng thực hiện sự chia phần dữ liệu. Sự chia phần dữ liệu (data partitioning) cho phép bạn phân chia một đối tượng cơ sở dữ liệu đơn chẳng hạn như table hay index ra thành nhiều phần. Tính năng data partitioning mới này giúp dễ dàng quản lý các table (bảng biểu) và index (chỉ mục) quá lớn. Sự chia phần là rõ ràng đối với các ứng dụng, vốn nhìn thấy chỉ đối tượng cơ sở dữ liệu và không nhận biết nhiều phần trong sự lưu trữ nền tảng, được quản lý bởi SQL Server. Các phần (partition) có thể được tạo và được thả mà không làm ảnh hưởng đến tính có sẵn của chính đối tượng cơ sở dữ liệu. Về cơ bản, sự chia phần cho phép bạn phân chia bộ lưu trữ dữ liệu (data store) nền tảng ra thành nhiều đối tượng trong khi vẫn trình bày một khung xem thống nhất của đối tượng và tất cả các phần của nó với một ứng dụng. Hình 1.2 minh họa tổng quan về sự chia phần.

Sự chia phần dữ liệu



Hình 1.2 Sự chia phần dữ liệu

SQL Server 2005 hỗ trợ sự chia phần dữ liệu (data partitioning) cho các table, index và các khung xem được tạo chỉ mục. Hàng là đơn vị chia phần cơ bản. Các phần (partition) có thể được tạo theo những giá trị được tìm thấy trong các cột trong một hàng. Đây được gọi là sự chia phần ngang (horizontal partition). Chẳng hạn, một table có thể được phân chia theo ngày tháng, trong đó một phần khác được tạo cho mỗi năm. Loại phân chia theo ngày tháng này cho phép bạn thực hiện một loại xử lý của số trượt ngày tháng, trong đó bạn có thể thả phần có chứa dữ liệu từ năm trước mà không làm ảnh hưởng đến sự truy cập vào dữ liệu chứa trong phần của năm hiện tại.

Sự chia phần dữ liệu cung cấp hai lợi ích quan trọng cho các cơ sở dữ liệu thật lớn (VLDBs). Các phần dữ liệu (data partition) giúp dễ dàng quản lý dữ liệu, cho phép bạn lưu chọn lọc chỉ những phần được chỉ định. Chẳng hạn, trong trường hợp của một table lớn được phân chia theo ngày tháng, bạn có thể muốn lưu dự phòng chỉ năm hiện tại, mà không lưu phần của năm trước. Một ưu điểm khác là trong các hệ thống có nhiều bộ xử lý, bạn có thể dành riêng một CPU cho việc xử lý phần riêng của nó nhằm cải tiến công suất.

Có hai bước cơ bản để thực thi sự chia phần dữ liệu. Trước tiên, bạn cần xác định chính xác cách bạn muốn chia phần một đối tượng đã cho. Thứ hai, bạn cần gán mỗi phần cho một vị trí lưu trữ vật lý. Các phần khác nhau có thể được gán hết cho một nhóm file hoặc các phần khác nhau có thể được ánh xạ sang nhiều nhóm file.

Ví dụ sau đây minh họa cú pháp để tạo một hàm chia phần đơn giản và phương pháp chia phần một table nhờ dùng một Range partition:

```
CREATE PARTITION FUNCTION MyPF
(int) AS RANGE LEFT FOR VALUES (50, 100)

GO

CREATE PARTITION SCHEME MyPS
AS PARTITION MyPF TO (FileGroup1)

GO

CREATE TABLE MyTable (col1 int, col2 varchar(50))

ON MyPS(col1)

GO
```

Đầu tiên tạo ra một hàm chia phần được gọi là MyPF. (int) cho thấy sự chia phần sẽ được thực hiện trên một cột được xác định nhờ

dùng kiểu dữ liệu int. Từ khóa RANGE chỉ định sự chia phần Range (Range partitioning) sẽ được sử dụng. Từ khóa LEFT kiểm soát phần nào sẽ nhận được các giá trị đường biên. Trị của LEFT biểu thị bất kỳ hàng nào có một trị trùng với biên chia phần sẽ được chuyển đến phần nằm ngay phía bên trái. Mệnh đề VALUES được sử dụng để xác định các điểm biên của các phần. Cần lưu ý rằng các trị này là những điểm biên chứ không phải chính các phần. Điều này sẽ thật sự dẫn đến việc tạo ba phần: Phần đầu chứa các trị âm đến 50; phần thứ hai chứa các trị từ 51 đến 100; phần thứ ba chứa tất cả các trị từ 101 trở lên.

Dòng thứ hai tạo một phương pháp chia phần được gọi là MyPS. Mệnh đề AS PARTITION được sử dụng để chỉ định hàm chia phần mà phương pháp này sẽ sử dụng. Ví dụ này dùng hàm chia phần MyPF. Mệnh đề TO xác định nhóm file hay các nhóm file sẽ chứa các phần (partition). Ví dụ này sử dụng một nhóm file đơn, được gọi là FileGroup1. Kế tiếp, phương pháp chia phần cần được đính kèm với table sẽ được chia phần. Ví dụ này minh họa cú pháp CREATE TABLE mở rộng vốn cho phép chia phần table. Phần đầu tiên của câu lệnh CREATE TABLE không thay đổi. Nó chỉ định tên table, My Table trong ví dụ này, và các cột của table. Table đơn giản này sử dụng hai cột, được gọi là col1 và col2. Từ khóa ON mới sau đó được sử dụng để chỉ định phương pháp chia phần. Ví dụ này sử dụng phương pháp chia phần MyPS vừa được tạo. Và cột chứa dữ liệu khóa của phần chia được đặt trong các dấu ngoặc đơn. Ví dụ này sử dụng cột col1 cho khóa chia phần. Cột này là một kiểu dữ liệu int, kiểu này phải trùng với kiểu dữ liệu được chỉ định trong hàm chia phần (partition function).

Có một vài hạn chế về các loại cột có thể dùng cho khóa chia phần (partitioning key). Các hạn chế này rất giống với các hạn chế của các cột dùng trong một index. Các kiểu dữ liệu text, ntext và image không được sử dụng. Tương tự, các cột timestamp (dấu thời gian) cũng bị hạn chế. Chỉ các kiểu dữ liệu "T-SQL" gốc mới có thể được sử dụng. Bạn không thể dùng một kiểu dữ liệu do người dùng ấn định làm một khóa chia phần. Tuy nhiên, bạn có thể sử dụng kiểu dữ liệu varchar(max) mới. Ngoài ra còn có một giới hạn là 1000 phần chia trên mỗi table và tất cả các phần chia phải tồn tại trên một nút đơn.

Các cải tiến index

Có nhiều cải tiến mới cho các index (chỉ mục) trong SQL Server 2005. Trước tiên, sự tạo lại một index nhóm cụm không còn bắt buộc phải tạo lại tất cả các index chưa được nhóm cụm nữa. Trong SQL Server 2000, khi bạn tạo lại một index được nhóm cụm, mọi index chưa được nhóm cụm liên quan cũng được tạo lại. Điều đó bây giờ không còn nữa, vì SQL Server 2005 vẫn giữ nguyên các index không được nhóm cụm trong khi tạo lại các index được nhóm cụm.

Kế tiếp, có một tính năng included columns mới cho phép bạn thêm các cột không phải khóa vào một index. Tính năng mới này cho phép bao hàm nhiều mẫu truy vấn hơn vào index, do đó cải tiến sự thực thi của các mẫu truy vấn bằng cách giảm thiểu nhu cầu bộ máy SQL Server đến table nền tảng để hoàn chỉnh mẫu truy vấn. Thay vào đó, bộ máy có thể đáp ứng các yêu cầu truy vấn bằng cách sử dụng chỉ dữ liệu trong index bao hàm. Một trong những khía cạnh thật sự hay của tính năng included columns mới là các cột vốn không phải là thành phần của khóa sẽ không được đưa vào cỡ tối đa của index, vốn vẫn là 900 byte.

Một cải tiến index mới khác mà Microsoft đã thêm vào SQL Server 2005 là khả năng vô hiệu hóa một index. Việc vô hiệu hóa một index làm cho index đó không được bảo lưu bởi bộ máy SQL Server đồng thời ngăn chặn việc sử dụng index đó. Khi một index bị vô hiệu hóa, SQL Server hủy cấp phát vùng lưu trữ mà index này sử dụng nhưng vẫn giữ lại metadata của index. Trước khi một index bị vô hiệu hóa có thể được kích hoạt trở lại, nó cần được tạo lại nhờ dùng lệnh ALTER INDEX.

Các hoạt động index trực tuyến

Các phiên bản trước của SQL Server không cho phép bất kỳ sự cập nhật nào vào một index trong khi index đó đang được tạo lại. Bạn cần chờ đợi cho đến khi tiến trình tạo lại hoàn tất và table có thể được cập nhật lại. Tính năng online index operations (các hoạt động index trực tuyến) mới của SQL Server 2005 cho phép các ứng dụng truy cập index cũng như thực hiện các hoạt động cập nhật, chèn và xóa trên một table trong khi hoạt động tạo lại index đang chạy. Bạn có thể tìm hiểu thêm về các hoạt động index trực tuyến của SQL Server 2005 ở chương 3.

Các cải tiến danh mục hệ thống và metadata

Trong SQL Server 2000 và các phiên bản trước, danh mục hệ thống và metadata được lưu trữ ở dạng thành phần của mọi cơ sở dữ liệu trong cơ sở dữ liệu chính. Với SQL Server 2005, điều này đã thay đổi và metadata bây giờ nằm trong cơ sở dữ liệu tài nguyên mà hệ thống lưu trữ ở dạng một đối tượng sys. SQL Server 2005 không còn cho phép truy cập trực tiếp vào các table hệ thống nữa. Sự thay đổi này đem lại sự an ninh tốt hơn và nâng cấp hệ thống nhanh hơn bằng cách củng cố metadata của hệ thống. Catalog metadata được bảo đảm an toàn nhờ sử dụng các bộ lọc cấp độ hàng.

Metadata mới hoàn toàn tương thích ngược miễn là bạn đừng sử dụng các table hệ thống chưa được tạo tài liệu mà Microsoft đã nhiều lần cảnh báo mọi người đừng sử dụng. Systems metadata trong SQL Server 2005 được thể hiện qua một tập hợp các khung xem danh mục (catalog views). Các khung xem danh mục cũng như các khung xem

ANSI INFORMATION_SCHEMA, các hàm đặc tính và các hàm cài sẵn thay thế nhu cầu sử dụng các table hệ thống như bạn có thể đã dùng trong SQL Server 2000. Tổng cộng có hơn 250 khung xem danh mục trong SQL Server 2005 và chúng có thể được xem từ sơ đồ sys của mọi cơ sở dữ liệu người dùng, Bạn có thể tìm thấy các khung xem hệ thống mới bằng cách sử dụng Microsoft SQL Server Manager Studio để mở Object Browser và sau đó định hướng đến nút Databases > <database> > Views > System Views. Bạn cũng có thể mở một cửa sổ truy vấn mới và nhập mẫu truy vấn sau:

```
select * from sys.system_views
```

Multiple Active Results Sets (MARS)

Các phiên bản trước của SQL Server bị giới hạn trong một tập hợp kết quả hoạt động trên mỗi nối kết. SQL Server 2005 bây giờ có khả năng hỗ trợ nhiều tập hợp kết quả hoạt động trên một nối kết đơn. Tính năng mới này cho phép bạn mở một nối kết đơn với cơ sở dữ liệu, thực thi một mẫu truy vấn và xử lý một số kết quả, sau đó bắt đầu một mẫu truy vấn khác và xử lý kết quả của nó. Các ứng dụng của bạn có thể tự do tới lui giữa nhiều tập hợp kết quả đang mở. Các ví dụ minh họa sử dụng các tính năng MARS mới được trình bày ở chương 4.

Tải dữ liệu khối

SQL Server 2005 cung cấp một số cải tiến tuyệt vời cũng như tăng hiệu suất trong tải dữ liệu khối. Tiến trình tải dữ liệu khối (bulk data loading) bây giờ sử dụng một dạng file dựa vào XML vốn cung cấp mọi tính năng được tìm thấy trong các phiên bản trước của file dạng BCP (Bulk Copy Programs) và nhiều hơn nữa. Ngoài ra, dạng XML làm cho file dạng BCP dễ đọc và dễ hiểu hơn. Về tính tương thích ngược với các ứng dụng hiện tại, file dạng BCP cũ vẫn có thể được sử dụng.

Tiến trình tải dữ liệu khối của SQL Server 2005 bây giờ có hỗ trợ sự ghi nhận ký các hàng bị lỗi. Điều này cho phép tiến trình tải dữ liệu khối tiếp tục diễn ra ngay cả khi gặp phải các hàng hay dữ liệu không hợp lệ. Các hàng được định dạng không chính xác được ghi vào một file lỗi cùng với một phần mô tả về tình trạng lỗi. Các hàng vi phạm các ràng buộc được gởi lại một table lỗi cùng với tình trạng lỗi cụ thể của chúng.

Tìm kiếm Full-Text

Sự hỗ trợ tìm kiếm Full-Text cũng đã được cải tiến trong SQL Server 2005. Các phiên bản trước của SQL Server yêu cầu sử dụng các thủ tục lưu trữ để tạo các danh mục tìm kiếm Full-Text. Với SQL Server 2005, nhiều câu lệnh DDL mới đã được giới thiệu nhằm cho phép bạn làm

việc với các tính năng tìm kiếm Full-Text của SQL Server. Chẳng hạn, hai câu lệnh DDL tìm kiếm Full-Text T-SQL mới là: CREATE FULLTEXT CATALOG và CREATE FULLTEXT INDEX.

Các cải tiến khác cho tính năng tìm kiếm Full-Text trong SQL Server 2005 bao gồm khả năng sao lưu dự phòng và phục hồi các danh mục tìm kiếm Full-Text và các index dữ liệu cùng với cơ sở dữ liệu của bạn. Tương tự, các danh mục Full-Text và các index có thể được đính kèm và hủy đính kèm với các cơ sở dữ liệu tương ứng của chúng. Một cải tiến hấp dẫn khác trong sự hỗ trợ tìm kiếm Full-Text của SQL Server 2005 là khả năng sử dụng một từ điển đồng nghĩa (thesaurus) để tìm các từ đồng nghĩa của các từ tìm kiếm.

Các cải tiến bộ xử lý truy vấn T-SQL

Có nhiều cải tiến cho bộ xử lý truy vấn trong SQL Server 2005, bao gồm Common Tables Expressions (CTE), một mệnh đề TOP cải tiến, một câu lệnh WAITFOR cải tiến, và một mệnh đề OUTPUT mới cho các câu lệnh DML. Các ví dụ về việc sử dụng các phần cải tiến này trong T-SQL được trình bày ở chương 4.

Sự bảo mật

Sự bảo mật mà Microsoft tạo cho SQL Server chủ yếu tập trung vào việc làm cho sản phẩm an toàn hơn và mạnh hơn từ khâu thiết kế cho đến khâu triển khai. Khi thiết kế các phần cải tiến bảo mật cho SQL Server 2005, Microsoft đã tuân theo một số nguyên tắc bảo mật cơ bản. Trước tiên, Microsoft muốn làm cho hệ thống an toàn ngay từ trong hộp bằng cách đẩy mạnh mọi xác lập cài đặt mặc định theo hướng tạo ra một môi trường an toàn. Trong khi nó để lại các tùy chọn mở cho người dùng chọn các xác lập ít an toàn hơn, việc chọn các tùy chọn này đòi hỏi sự lựa chọn tinh vi. Kế tiếp, Microsoft tuân theo nguyên tắc của các đặc quyền tối thiểu trong thiết kế hệ thống. Hệ thống được thiết kế sao cho một cá nhân chỉ có các đặc quyền bắt buộc để thực hiện một chức năng đã cho và không gì khác. Sau cùng, Microsoft muốn giảm bớt vùng bề mặt bộc lộ tiềm năng bằng cách đưa ra khả năng cài đặt chỉ những thành phần cần thiết.

Mọi tính năng bảo mật mới trong SQL Server 2005 chịu sự tác động sâu sắc bởi những thứ mà Microsoft phát hiện trong lúc thiết lập hệ thống bảo mật vào đầu năm 2002 và đã được đưa vào kiểu thiết kế và thực thi SQL Server 2005. Một số cải tiến tính năng bảo mật chính trong SQL Server 2005 mà bạn sẽ đọc trong mục này bao gồm sự phân biệt user với schema, ngữ cảnh thực thi thủ tục lưu trữ mới, sự điều khiển các permission, sự tuân theo chính sách password mới, các thay

đối đối với hệ thống bảo mật ở cấp độ hàng và sự bảo mật cải tiến cho các danh mục.

Sự phân biệt User-Schema

Sự thay đổi có ý nghĩa nhất liên quan đến hệ thống bảo mật mới trong SQL Server 2005 là sự phân biệt user-schema. Một user (người dùng), hay có lẽ chính xác hơn là một principal, là bất kỳ thực thể mà dựa vào đó các đối tượng cơ sở dữ liệu được bảo đảm an toàn. Một principal là một user Windows, một user SQL Server, một role (vai trò) hay một role ứng dụng. Với SQL Server 2000, được sở hữu trực tiếp bởi các user, các đối tượng cơ sở dữ liệu và bản thân các user được tìm thấy trong table hệ thống sys_users. Đó là tất cả những thay đổi với SQL Server 2005. Bây giờ các đối tượng cơ sở dữ liệu được sở hữu bởi các schema. Các user không còn trực tiếp sở hữu trực tiếp các đối tượng cơ sở dữ liệu nữa mà sở hữu các schema. Với SQL Server 2005, các user và các principal bảo mật khác có thể được tìm thấy trong khung xem sys.database_principals mới. Danh sách schema của SQL Server 2005 có thể được tìm thấy trong khung xem sys.schemas mới.

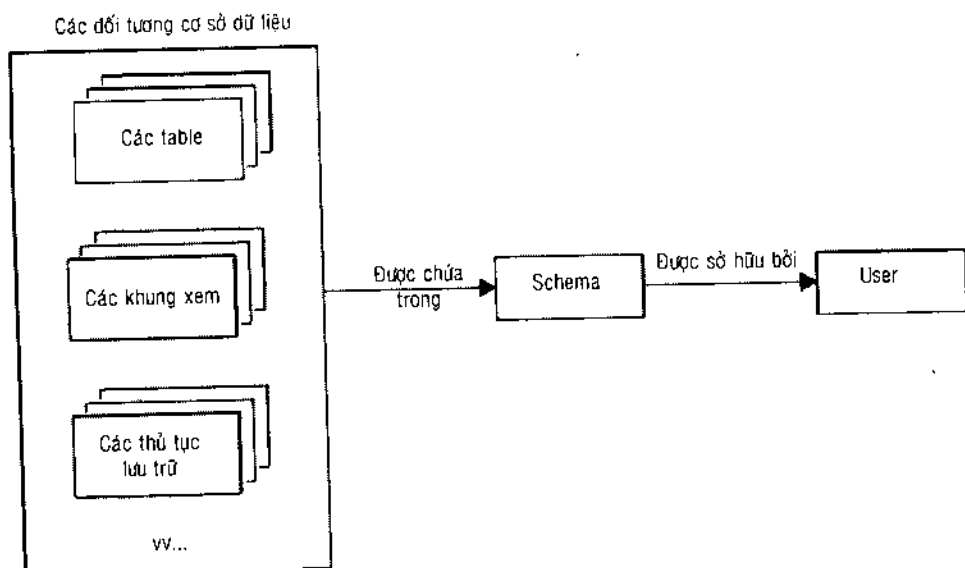
Một schema là một đối tượng chứa của các đối tượng. Schema được nhận biết bởi phần thứ ba trong cú pháp đặt tên đối tượng gồm bốn phần được sử dụng bởi SQL Server. Ví dụ sau đây minh họa cú pháp đặt tên của SQL Server 2005, trong đó mỗi phần của tên tăng dần về tính cụ thể.

Server_name.Database_name . Schema_name . Object_name

Trong tất cả các phiên bản trước của SQL Server, tên schema và tên owner (người sở hữu) về cơ bản là cùng một thứ. Với SQL Server 2005, owner đã được tách riêng với schema. Khi SQL Server 2000 và các ấn bản trước phân giải các tên đối tượng, trước tiên SQL Server tìm Database_name.User_name.Object_name và nếu không tìm được, nó sẽ tìm Database_name.dbo.Object_name.

Lý do chính đối với sự tách biệt giữa user và schema này trong SQL Server 2005 là giải quyết vấn đề của nhu cầu thay đổi quyền sở hữu (ownership) của nhiều đối tượng cơ sở dữ liệu nếu một user đã cho (còn được gọi là người sở hữu đối tượng cũ) rời khỏi tổ chức. Ngoài ra, hoạt động thay đổi quyền sở hữu của một đối tượng cơ sở dữ liệu cũng dẫn đến sự thay đổi tên. Chẳng hạn, nếu người sở hữu Table 1 trong cơ sở dữ liệu MyDB được thay đổi từ UserA sang UserB thì tên của Table 1 sẽ thay đổi từ MyDB.UserA.Table 1 sang MyDB.UserB.Table 1. Để giúp ngăn chặn vấn đề này, nhiều tổ chức đã chọn chuẩn cho phép mọi đối tượng cơ sở dữ liệu được sở hữu bởi dbo, nhưng không có gì trong server ép buộc việc thực hiện này.

Sự thực thi khái niệm của một database schema trong SQL Server 2005 đã đưa ra một mức trừu tượng trong chuỗi quyền sở hữu đối tượng cơ sở dữ liệu. Hình 1.3 minh họa chuỗi quyền sở hữu đối tượng cơ sở dữ liệu của SQL Server 2005.



Hình 1.3 Chuỗi quyền sở hữu đối tượng của SQL Server 2005

Với SQL Server 2005, các đối tượng cơ sở dữ liệu được chứa trong các schema và đến lượt các schema này được sở hữu bởi các user. Mức trừu tượng mới này làm cho việc thay đổi quyền sở hữu đối tượng cơ sở dữ liệu (database object ownership) trở nên dễ quản lý hơn nhiều. Việc thả một user sở hữu các đối tượng cơ sở dữ liệu trong SQL Server 2005 có nghĩa là DBA bây giờ cần thay đổi chỉ quyền sở hữu của schema và không cần thay đổi quyền sở hữu của tất cả các đối tượng cơ sở dữ liệu riêng lẻ. Điều này giúp giảm đáng kể số lượng đối tượng mà DBA cần liên hệ để thay đổi quyền sở hữu của các đối tượng trong một cơ sở dữ liệu. Để thay đổi quyền sở hữu của tất cả các đối tượng trong một cơ sở dữ liệu SQL Server 2005, bạn chỉ cần thay đổi người sở hữu schema, sau đó bạn có thể thả user cũ. Sự thay đổi người sở hữu đối tượng cơ sở dữ liệu không làm thay đổi tên của một đối tượng bởi vì tên schema không thay đổi, chỉ quyền sở hữu của nó là thay đổi.

Như bạn có thể mong đợi, đối tượng schema mới làm thay đổi cách mà SQL Server thực hiện sự phân giải tên đối tượng cơ sở dữ liệu. Mỗi user bây giờ có một schema mặc định kết hợp với user đó và SQL Server 2005 trước tiên sẽ tìm một tên đối tượng chưa được định tính nhờ dùng schema mặc định của user. Nếu điều đó không thành công, SQL Server sẽ tìm đối tượng nhờ dùng tên schema của dbo. Chẳng hạn,

nếu UserA có một schema mặc định là MySchema1 và user đó thực hiện một mẫu truy vấn tìm kiếm Table1 thì server trước tiên sẽ cố phân giải tên nhờ dùng MySchema1.Table1 rồi sau đó trở lại dbo.Table1.

Cũng như các cơ sở dữ liệu SQL Server 2000 có thể chứa nhiều user và role (vai trò), các cơ sở dữ liệu SQL Server 2005 có thể chứa nhiều schema. Mỗi schema có một nguyên tắc sở hữu, thường là một user hay một role. Đối với các mục đích phân giải tên, mỗi user có một schema mặc định. Các đối tượng cơ sở dữ liệu thật sự sau đó được chứa trong một schema. Để tạo các đối tượng cơ sở dữ liệu mới bên trong một schema, bạn phải có sự cho phép CREATE dành cho chính đối tượng cơ sở dữ liệu và sự cho phép ALTER hay CONTROL cho schema. Sự tạo chuỗi quyền sở hữu (Ownership chaining) vẫn dựa vào các owner thật sự, không dựa vào schema.

SQL Server 2005 đưa ra nhiều thay đổi DDL để xử lý sự phân biệt user và schema mới, bao gồm câu lệnh CREATE/DROP/ALTER dành cho các đối tượng USER, ROLE, và SCHEMA. Danh sách sau đây minh họa cách tạo và gán một schema cơ sở dữ liệu:

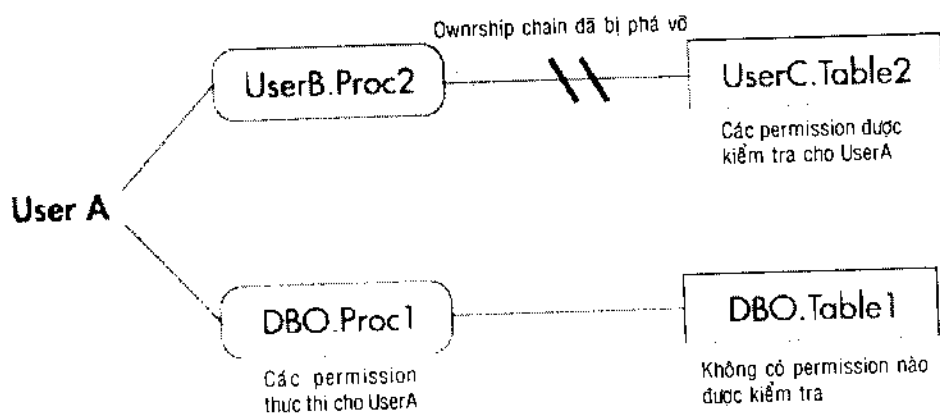
```
/* Create a login */  
  
CREATE LOGIN UserA WITH PASSWORD = 'ABC123#$'  
GO  
  
/* Create a user for that login – the schema doesn't need to exist*/  
  
CREATE USER UserA FOR LOGIN UserA  
    WITH DEFAULT_SCHEMA = MySchema  
GO  
  
/* Create the schema and assign its owner */  
  
CREATE SCHEMA MySchema AUTHORIZATION UserA  
GO  
  
/* Create a Table in the new schema */  
  
CREATE TABLE MySchema.Table1 (col1 char (20))  
GO
```

Dòng đầu tiên trong danh sách này tạo ra một login mới được gọi là UserA và xác lập một password cho login đó. Dòng kế tiếp tạo ra một user mới được gọi là UserA và xác lập schema mặc định cho UserA sang MySchema. Schema thật sự không cần tồn tại cùng lúc nó được chỉ định trong câu lệnh CREATE USER. Nếu bạn không chỉ định một schema mặc định khi bạn tạo một user mới thì schema mặc định sẽ được xác lập sang dbo. Kế tiếp, câu lệnh CREATE SCHEMA được sử dụng để tạo một schema mới được gọi là MySchema. Mệnh đề AUTHORIZATION

xác lập người sở hữu của schema là UserA. Sau cùng, một table tên Table 1 được tạo trong schema tên MySchema. Người sở hữu cho MySchema và các đối tượng của nó chẳng hạn như Table 1 là UserA.

Ngữ cảnh thực thi thủ tục lưu trữ

Trong khi Microsoft đề cập đến một tính năng mới như là một ngữ cảnh thực thi thủ tục lưu trữ, nó thật sự áp dụng cho các module hơn là chỉ các thủ tục lưu trữ (stored procedures). Một module có thể là một thủ tục lưu trữ, một hàm hay một hợp ngữ. Việc xác lập ngữ cảnh thực thi cho một module làm cho tất cả các câu lệnh chứa trong module đó được kiểm tra với các phần cho phép (permissions) dựa vào ngữ cảnh thực thi đã được chỉ định. Nói cách khác, bằng cách xác lập ngữ cảnh thực thi của một module đã cho, bạn làm cho tất cả các câu lệnh chứa trong module đó được thực thi nhờ dùng quyền của user mà bạn chỉ định thay vì trình gọi thực sự của module. Tính năng mới này cho phép bạn có được những ưu điểm tương tự như những ưu điểm đã được nhận biết qua chuỗi quyền sở hữu của SQL Server 2000, nhưng nó linh hoạt hơn vì nó không có cùng hạn chế. Chẳng hạn, không giống như sự tạo chuỗi quyền sở hữu (ownership chaining) của SQL Server 2005, vốn không cho phép bạn thay đổi ngữ cảnh thực thi cho SQL động, ngữ cảnh thực thi của SQL Server 2005 áp dụng cho SQL động cũng như cho SQL tĩnh. Để hiểu rõ hơn về điều này, hãy xem minh họa ownership chaining của SQL Server 2000 ở hình 1.4.

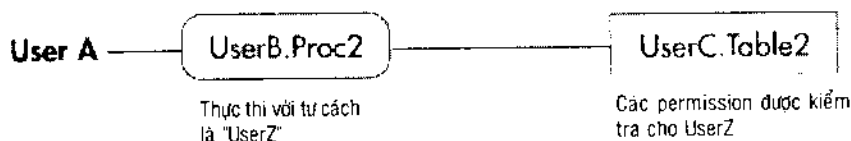


Hình 1.4 Ownership chain của SQL Server 2000

Để UserA thực thi dbo.Proc 1, UserA phải có permission thực thi cho đối tượng đó. Tuy nhiên, khi dbo.Proc 1 truy cập dbo.Table 1, không có permission nào được kiểm tra, bởi vì dbo là owner của cả hai đối tượng. Đây là một ví dụ về một ownership chain nguyên vẹn. Trong tình huống

kế tiếp, để UserA thực thi UserB.Proc2, UserA phải có các permission Execute (thực thi) cho đối tượng đó. Sau đó, khi UserB.Proc2 cố truy cập UserC.Table2, các permission Select từ UserA phải được kiểm tra. Trong trường hợp này, bởi vì UserB.Proc2 và UserC.Table2 có các owner khác nhau nên ownership chain bị phá vỡ.

Sự thực thi của SQL Server 2005 giúp đơn giản tình huống này, như minh họa ở hình 1.5, trong tình huống này, khi UserA cố thực thi UserB.Proc2, SQL Server kiểm tra để bảo đảm UserA có các permission Execute cho UserB.Proc1. Nếu đối tượng UserB.Proc1 được tạo với một ngữ cảnh thực thi qui định thủ tục lưu trữ sẽ được thực thi với tư cách là UserZ thì khi thủ tục lưu trữ UserB.Proc1 cố truy cập UserC.Table1, SQL Server sẽ kiểm tra các permission Select chỉ trên user đã được chỉ định trong ngữ cảnh thực thi, trong trường hợp này là UserZ. Không có permission Select nào là bắt buộc cho UserA, vốn là người gọi thật sự.



Hình 1.5 Ngữ cảnh thực thi của SQL Server 2005

Danh sách sau đây minh họa cách bạn thay đổi ngữ cảnh thực thi của một thủ tục lưu trữ được gọi là MyProc1:

```
ALTER PROC MySchema.Proc1 WITH EXECUTE AS USER UserB
```

Câu lệnh này minh họa mệnh đề WITH EXECUTE mới. Ở đây, thủ tục lưu trữ Proc1 chứa trong MySchema được xác lập để thực thi trong ngữ cảnh của UserB. Bạn phải chỉ định một user name cho nội dung thực thi. Bạn không được chỉ định tên của một role. Các thay đổi trong ngữ cảnh thực thi được lưu trữ trong khung xem sys.sql_modules.

Kiểm soát các permission tốt hơn

SQL Server 2005 cũng cung cấp sự kiểm soát tốt hơn đối với các permission (sự cho phép). Với SQL Server 2005, Microsoft đã bổ sung thêm nhiều permission ở các phạm vi khác nhau. Các phạm vi của các permission có thể được áp dụng là: Server, cơ sở dữ liệu, schema, đối tượng, và principal. Mục đích thiết kế sau các permission cải tiến của SQL Server 2005 là nguyên tắc độc quyền tối thiểu, cho DBA khả năng kiểm soát chính xác những permission nào được gán. Các permission mới không tách khỏi các role cố định hiện tại của SQL Server. Mọi role trước đây vẫn được giữ lại và có thể cùng tồn tại mà không gặp bất kỳ

vấn đề gì. Một tình huống cụ thể cho thấy ý nghĩa của các permission mới này là trường hợp kiểm toán. Trên SQL Server 2000, bạn cần là thành viên của nhóm sysadmins để thực hiện việc kiểm toán. Tuy nhiên, tư cách thành viên trong nhóm này còn cho phép nhiều khả năng khác. Một số permission có sẵn trong SQL Server 2005 cho phép thực hiện các chức năng kiểm toán mà không bắt buộc user phải là thành viên của nhóm sysadmins.

Các trạng thái permission cơ bản của GRANT, DENY, và REVOKE vốn được sử dụng trong các phiên bản trước đây của SQL Server vẫn được áp dụng. Một điểm khác biệt về cách SQL Server 2005 sử dụng các permission là cùng permission có thể được áp dụng ở nhiều phạm vi. Chẳng hạn, nếu bạn áp dụng một permission cho phạm vi cơ sở dữ liệu, nó áp dụng cho mọi đối tượng trong cơ sở dữ liệu đó. Nếu bạn áp dụng một permission cho cấp độ schema, nó cũng áp dụng cho các đối tượng chứa trong schema đó. Ngoại trừ permission DENY, permission phạm vi cao hơn luôn được sử dụng. Tuy nhiên, DENY luôn được ưu tiên ở bất kỳ cấp độ. Bảng 1.1 liệt kê một số permission SQL Server 2005 mới và quan trọng nhất. Các permission server được tìm thấy trong khung xem sys.server_permissions, trong khi các permission cơ sở dữ liệu được tìm thấy trong khung xem sys.database_permissions.

Bảng 1.1 Một số permission mới trong SQL Server 2005

Permission	Mô tả
ALTER	Cho phép thay đổi các thuộc tính của một đối tượng. Cũng cho phép thực thi các câu lệnh CREATE/DROP/ALTER.
ALTER ANY X	Cho phép thay đổi bất kỳ đối tượng thuộc loại X. Chẳng hạn, nếu bạn đã thay table cho X thì điều này sẽ cho phép thay đổi bất kỳ table nào trong cơ sở dữ liệu.
ALTER TRACE	Cho phép thực hiện kiểm toán và chạy Profiler.
CONTROL	CẤP các permission chính.
SELECT	Cho phép truy cập một đối tượng. Bây giờ áp dụng cho schema và các cấp độ cơ sở dữ liệu thay vì cho chỉ cấp độ đối tượng.
EXECUTE	Cho phép thực thi một thủ tục, hợp ngữ hay hàm. Bây giờ áp dụng cho schema và các cấp độ cơ sở dữ liệu thay vì chỉ cấp độ đối tượng.

IMPERSONATE	Cho phép một login hay user đóng vai một user khác.
TAKE OWNERSHIP	Cho phép lấy quyền sở hữu của một đối tượng.
VIEW DEFINITION	Cho phép xem metadata của một đối tượng.

Sự tuân theo chính sách password

Một tính năng bảo mật mới quan trọng khác trong SQL Server 2005 là sự hỗ trợ các chính sách password (mật khẩu). Chính sách mới này tuân theo các chính sách password Windows cục bộ và cho phép bạn thực thi một chính sách bảo mật nhất quán trong toàn doanh nghiệp, không chỉ cho các hệ thống server Windows của bạn mà còn cho các hệ thống cơ sở dữ liệu SQL Server. SQL Server 2005 có khả năng tuân theo các chính sách password strength, password expiration và account lockout. Như bạn mong đợi, chính sách password strength buộc các password phải có một mức phức tạp cho trước. Chính sách password expiration bảo đảm các password hết hạn và phải được xác lập lại tại một thời khoảng cho trước. Chính sách account lockout cho phép một account được khóa sau số lần nỗ lực nhất định của password không đúng. Mọi chính sách password mới này được hỗ trợ trên Windows Server 2003. Tuy nhiên, chỉ chính sách password complexity là được hỗ trợ trên Windows 2000 Server. Theo sự thúc đẩy bảo mật của Microsoft, mọi chính sách được kích hoạt trong phần cài đặt mặc định, chúng còn được cấu hình lại trên cơ sở từng login. SQL Server 2005 chứa các chính sách password mới trong khung xem danh mục sys.sql_logins.

Bảo mật danh mục

Cải tiến cuối cùng liên quan đến hệ thống bảo mật được trình bày trong mục này là bảo mật danh mục mới do SQL Server 2005 cung cấp. Các table hệ thống được sử dụng bởi SQL Server 2000 trong các cơ sở dữ liệu riêng lẻ và trong cơ sở dữ liệu chính bây giờ được thực thi như là các khung xem danh mục (catalog view) trong SQL Server 2005. Metadata của server hiển thị trong các khung xem này được bảo đảm an toàn theo mặc định và có các permission chung. Các khung xem danh mục của SQL Server 2005 sử dụng sự bảo mật ở cấp độ hàng (row-level security), giới hạn sự truy cập vào dữ liệu chứa trong các khung xem đó trong chỉ những đối tượng mà bạn sở hữu hay có các permission cho chúng. Một cách tự nhiên, sa là ngoại lệ của vấn đề này. Tài khoản sa vẫn có sự truy cập vào mọi đối tượng trong server.

Để cho phép một user hay row truy cập metadata, DBA cần sử dụng permission VIEW DEFINITION mới. Permission VIEW DEFINITION có thể được sử dụng để cung cấp cho một user vốn không phải là người sở hữu và cũng không có các permission truy cập đối tượng khả năng xem metadata của đối tượng đó. Permission VIEW DEFINITION có thể được áp dụng ở các phạm vi server, cơ sở dữ liệu, schema và đối tượng.

CHƯƠNG 2

Các công cụ quản trị và phát triển cơ sở dữ liệu

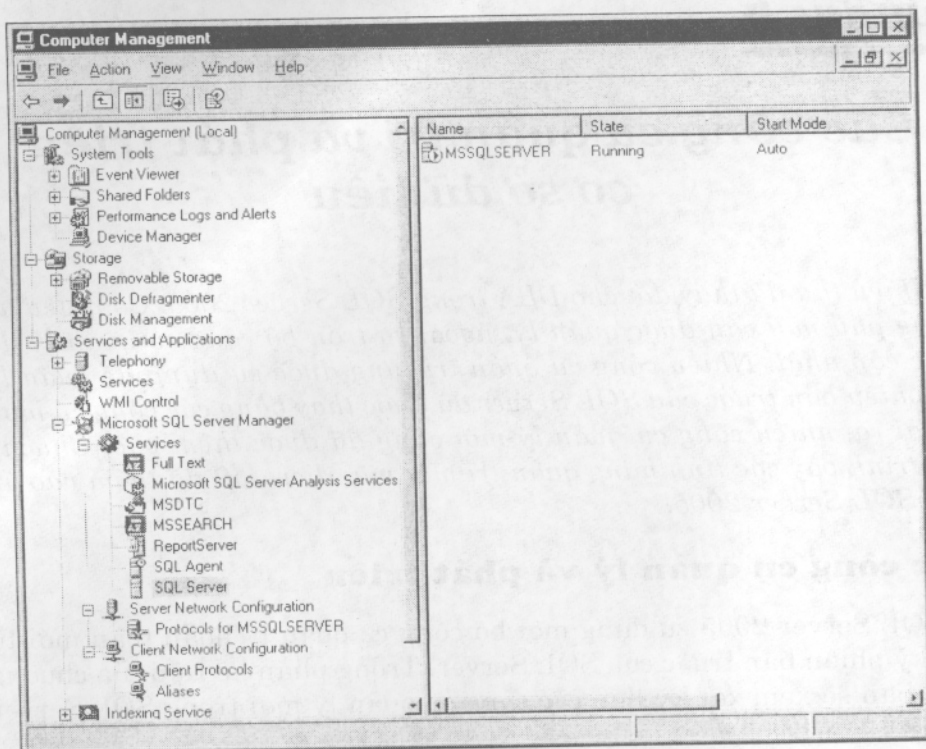
Nhiều thứ đã thay đổi cho DBA trong SQL Server 2005. Có nhiều hệ thống phụ mới cần được quản lý, ngoài ra trọn bộ công cụ quản lý đã được cập nhật. Nhiều công cụ quản trị từng được sử dụng để quản lý các phiên bản trước của SQL Server đã được thay bằng các công cụ mới. Ngoài ra, nhiều công cụ quản lý mới cũng đã được thêm vào. Chương này trình bày các tính năng quản trị mới mà Microsoft đã thêm vào ấn bản SQL Server 2005.

Các công cụ quản lý và phát triển

SQL Server 2005 sử dụng một bộ công cụ quản lý hoàn toàn mới từ bất kỳ phiên bản trước của SQL Server. Trong phần kế tiếp của chương này, bạn sẽ xem xét kỹ hơn các công cụ quản lý mới trong SQL Server 2005.

SQL Computer Manager

Điều đầu tiên mà SQL Server DBA sẽ nhận thấy về bộ công cụ quản lý mới của SQL Server 2005 là không có Server Manager. Trong các phiên bản trước của SQL Server, Server Manager nằm trong System Tray (khay hệ thống). Server Manager cung cấp dạng biểu diễn bằng hình ảnh cho tình trạng của các dịch vụ SQL Server khác nhau, bao gồm dịch vụ SQL Server, dịch vụ SQL Agent, và dịch vụ Distributed Transaction Coordinator. Bạn cũng có thể sử dụng Server Manager để mở và ngưng các dịch vụ đó. Tuy đây là một công cụ tiện lợi nhưng Server Manager không tương thích Windows Logo và đó là lý do chính để Microsoft loại bỏ nó ra khỏi SQL Server 2005. Với SQL Server 2005, bạn có thể xem và điều khiển các dịch vụ SQL Server nhờ dùng SQL Computer Manager mới. SQL Computer Manager là một snap-in MMC có thể được truy cập nhờ dùng tùy chọn My Computer > Manage. Hình 2.1 minh họa SQL Computer Manager mới này.



Hình 2.1 SQL Computer Manager

Bằng cách sử dụng SQL Computer Manager, bạn có thể thấy mọi dịch vụ SQL Server đang chạy trên hệ thống được kê ngay bên dưới mức Services chính. Bạn sử dụng cửa sổ Computer Management để quản lý dịch vụ sau: SQL Server, SQL Agent, ReportServer, MSDTC, Microsoft.SQL Server Analysis và FullText.

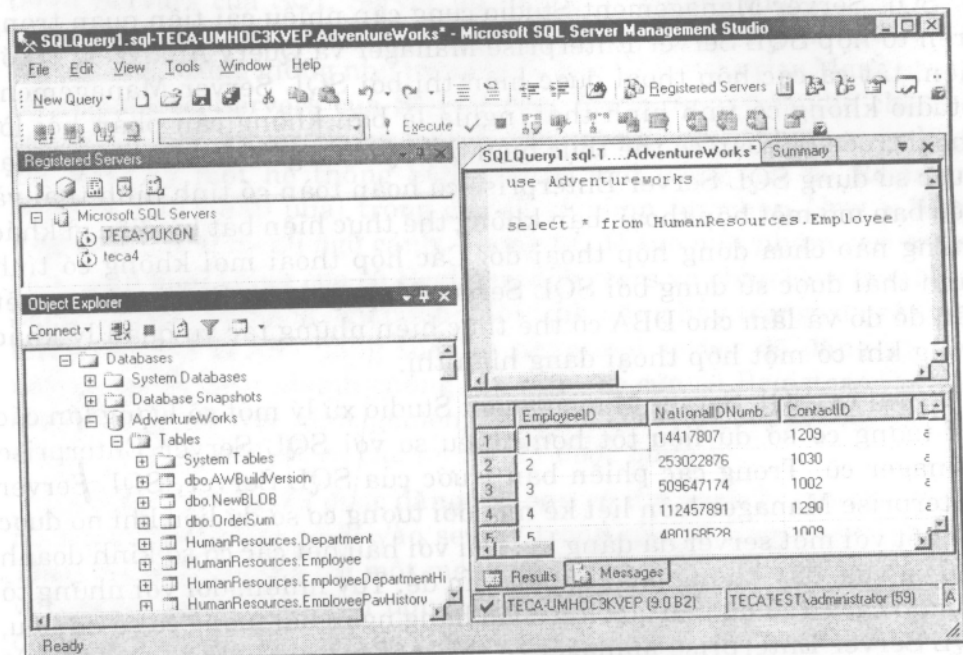
Bạn có thể điều khiển bất kỳ dịch vụ bằng cách trước tiên chọn nó trong khung hình cây ở phía bên trái của màn hình. Khung này hiển thị tình trạng của dịch vụ trong ô details ở bên phải. Để quản lý dịch vụ, bạn nhấp phải vào dịch vụ trong ô details rồi chọn tùy chọn để Start, Stop, Pause, Resume, hay Restart dịch vụ từ menu ngữ cảnh bật lên.

SQL Server Management Studio

Server Manager không phải là công cụ quản lý quen thuộc duy nhất được thay đổi trong SQL Server 2005. SQL Server Enterprise Manager, vốn là công cụ quản lý chính cho SQL Server các phiên bản 7 và 2000, đã được thay bằng SQL Server Management Studio mới. Cũng vậy, Query Analyzer, vốn là công cụ phát triển T-SQL cốt lõi trong SQL Server các phiên bản 7 và 2000, đã được thay bằng SQL Server Management Studio mới. SQL Server 2005 còn cung cấp nhiều công cụ quản

trị khác cho DBA, bao gồm Administrative Console mới, Database Tuning Advisor và một Profiler mới.

SQL Server Management Studio mới là một sự khởi đầu từ những công cụ quản trị đã có trong các ấn bản trước của SQL Server. SQL Server Management Studio mới hợp nhất hầu như mọi tính năng đã từng được cung cấp bởi SQL Server Enterprise và Query Analyzer. SQL Server Management Studio được truy cập bằng cách dùng tùy chọn menu Start > Programs > Microsoft SQL Server > SQL Server Management Studio. Hình 2.2 minh họa SQL Server Management Studio.



Hình 2.2 SQL Server Management Studio

SQL Server Management Studio có thể được sử dụng để quản lý các hệ thống SQL Server 2005 cũng như các hệ thống SQL Server 2000 và SQL Server 7. Nó không thể được sử dụng trên các hệ thống SQL Server 6.5 hoặc cũ hơn. Bạn sử dụng SQL Server 7 hay SQL Server 2000 Enterprise Manager để quản lý một hệ thống SQL Server 2005 mới, nhưng do bởi một số thay đổi về kiến trúc giữa hai ấn bản nên điều này không được hỗ trợ hay đề nghị. Các công cụ quản lý cũ hơn không được truy cập bất kỳ tính năng mới đã được thêm vào SQL Server 2005. Trong khi bạn có thể thực hiện việc duyệt table và cơ sở dữ liệu cơ bản, nhiều hành động khác sẽ dẫn đến các lỗi. SQL Server Management Studio là công cụ được lựa chọn để quản lý các hệ thống kết hợp SQL Server 2005 và SQL Server 2000.

SQL Server Management Studio được tạo nhờ dùng một phiên bản chuyên biệt của Visual Studio 2005 IDE (Interactive Development Environment, môi trường phát triển tương tác). Giống như Visual Studio 2005, SQL Server Management Studio hỗ trợ việc tạo các dự án phần mềm. Nó cho phép bạn viết, hiệu chỉnh, chạy và gỡ rối mã. Nó cũng được hợp nhất với Visual SourceSafe dành cho sự điều khiển phiên bản mã nguồn. Tuy nhiên, không giống như Visual Studio 2005, nó không cho phép bạn biên dịch VB.NET, C#, J#, hay VC++. Thay vào đó, SQL Server Management Studio hoạt động với T-SQL, MDX, MX, và XMLA.

SQL Server Management Studio cung cấp nhiều cải tiến quan trọng trên tổ hợp SQL Server Enterprise Manager và Query Analyzer. Trước tiên, tất cả các hộp thoại được hiển thị bởi SQL Server Management Studio không có tính hình thái, nghĩa là bạn không cần phản hồi lời thoại trước khi bạn có thể thực hiện bất kỳ điều gì khác. Các hộp thoại được sử dụng SQL Server Enterprise cũ hoàn toàn có tính hình thái và nếu bạn mở một hộp thoại, bạn không thể thực hiện bất kỳ điều gì khác chừng nào chưa đóng hộp thoại đó. Các hộp thoại mới không có tính hình thái được sử dụng bởi SQL Server Management Studio giải quyết vấn đề đó và làm cho DBA có thể thực hiện những tác vụ quản lý khác trong khi có một hộp thoại đang hiển thị.

Ngoài ra, SQL Server Management Studio xử lý một số lượng lớn các đối tượng cơ sở dữ liệu tốt hơn nhiều so với SQL Server Enterprise Manager cũ. Trong các phiên bản trước của SQL Server, SQL Server Enterprise Manager luôn liệt kê mọi đối tượng cơ sở dữ liệu khi nó được nối kết với một server đã đăng ký. Đối với hầu hết các cơ sở kinh doanh nhỏ và vừa, đây không phải là một vấn đề. Tuy nhiên, đối với những tổ chức có các cơ sở dữ liệu quá lớn chứa hàng ngàn đối tượng cơ sở dữ liệu, SQL Server Enterprise Manager có thể mất rất nhiều thời gian để liệt kê mọi đối tượng cơ sở dữ liệu và các thuộc tính của chúng - về cơ bản là mô phỏng SQL Server Enterprise Manager không được sử dụng cho đến khi tất cả các đối tượng đã được liệt kê. Với SQL Server 2005, SQL Server Management Studio thông minh hơn nhiều ở cách nó xử lý các đối tượng cơ sở dữ liệu, nó không liệt kê các đối tượng cơ sở dữ liệu cho đến khi người dùng chọn mở rộng một hạng mục cơ sở dữ liệu để hiển thị trong Object Browser. Điều này SQL Server Management Studio cung cấp sự phản hồi tốt hơn nhiều khi nó được sử dụng để quản lý các cơ sở dữ liệu rất lớn.

SQL Server Management Studio phục vụ như là chủ cho đa số các công cụ quản lý và phát triển SQL Server. Cửa sổ Registered Servers cho phép bạn chọn các hệ thống SQL Server mà bạn sẽ quản lý. Cửa sổ Object Explorer cho phép bạn làm việc với các đối tượng cơ sở dữ liệu trong mỗi server. Và cửa sổ Solution Explorer cho phép bạn nhóm mã

nguồn cơ sở dữ liệu thành các tập hợp lô gíc. Bạn có thể tìm hiểu thêm về Registered Servers, Object Explorer, Query Editor, và Solution Explorer trong các mục sau của chương này. Report Designer được trình bày ở chương 8 và DTS Designer được trình bày ở chương 9.

Registered Servers

Rất giống như khi sử dụng SQL Server Enterprise Manager, bạn phải đăng ký các server trong SQL Server Management Studio trước khi bạn có thể sử dụng nó để quản lý chúng. Bạn sử dụng cửa sổ Registered Servers của SQL Server Management Studio (được minh họa ở phần phía trên bên trái trong hình 2.2) để đăng ký các hệ thống SQL Server mới. Cũng giống như trong SQL Server Enterprise Manager, cửa sổ Registered Servers của SQL Server Management Studio cho phép bạn nhóm các server chung lại với nhau thành các nhóm server lô gíc. Để đăng ký một hệ thống SQL Server mới trong cửa sổ Registered Servers, bạn nhấp phải trong cửa sổ và chọn tùy chọn New > Server Registration từ menu ngữ cảnh. Tương tự, để tạo một nhóm server mới, bạn nhấp phải trong cửa sổ Registered Servers và chọn tùy chọn New > Server Group từ menu ngữ cảnh. Một tính năng hay trong cửa sổ Registered Servers là khả năng xuất và nhập mọi server đã đăng ký. Điều này cho phép bạn nhanh chóng tập hợp các cửa sổ Registered Servers của các SQL Server Management Studio khác nhau mà không cần phải đăng ký bằng tay tất cả các server đã được quản lý.

Sau khi một server được đăng ký, bạn có thể quản lý các dịch vụ của nó bằng cách nhấp phải vào server trong cửa sổ Registered Servers. Thao tác này sẽ hiển thị một menu ngữ cảnh vốn cho phép bạn nối kết với server cũng như khởi động, ngưng, tạm ngưng, tiếp tục hay khởi động lại dịch vụ SQL Server. Bạn có thể làm việc với các đối tượng cơ sở dữ liệu của một server bằng cách nhấp đúp vào tên server. Thao tác này sẽ nối kết bạn với server SQL đã đăng ký và tự động mở cửa sổ Object Explorer.

Object Explorer

Cửa sổ SQL Server Management Studio Object Explorer cho phép bạn thực hiện các chức năng quản lý tương tự như khi sử dụng SQL Server Enterprise Manager. Bạn có thể thấy cửa sổ Object Explorer ở góc dưới bên trái trong hình 2.2. Bằng cách sử dụng Object Explorer, bạn có thể

- ❖ Khởi động và ngưng một server
- ❖ Cấu hình các thuộc tính của một server
- ❖ Tạo các cơ sở dữ liệu

- ❖ Gán hay hủy gán cơ sở dữ liệu
- ❖ Tạo các đối tượng cơ sở dữ liệu như các table, các khung xem, hay các thủ tục lưu trữ
- ❖ Tạo các script, tạo đối tượng T-SQL
- ❖ Cài đặt các login server
- ❖ Quản lý các permission đối tượng cơ sở dữ liệu
- ❖ Cấu hình và quản lý sự sao lập
- ❖ Cấu hình và quản lý các server đã liên kết
- ❖ Giám sát hoạt động server
- ❖ Xem các log hệ thống

Để làm việc với các đối tượng cơ sở dữ liệu được hiển thị trong Object Explorer, bạn nhấp phải vào đối tượng bạn muốn trong cây Object Explorer để hiển thị menu ngữ cảnh của đối tượng đó. Menu ngữ cảnh cung cấp một tập hợp tùy chọn duy nhất cho mỗi đối tượng cơ sở dữ liệu. Chẳng hạn, menu ngữ cảnh cơ sở dữ liệu cho phép bạn xóa, thay đổi tên, gán, hủy gán, thu nhỏ, sao lưu dự phòng, và phục hồi các cơ sở dữ liệu, trong khi menu ngữ cảnh table cho phép bạn tạo một table mới, chỉnh sửa một table, mở các phần lệ thuộc trong khung xem table và xác lập các permission.

Một tính năng hấp dẫn khác của Object Explorer mới là khả năng tạo các script một cách dễ dàng để tạo ra bất kỳ đối tượng cơ sở dữ liệu hiển thị trong danh sách Object Explorer. Script Wizard có thể được sử dụng để tạo tài liệu cho schema của một cơ sở dữ liệu, để tạo một bản sao dự phòng của một cơ sở dữ liệu hay để tạo một bản sao thủ của một cơ sở dữ liệu hoặc các đối tượng cơ sở dữ liệu được chọn. Bạn có thể chọn để tạo một script đơn cho tất cả các đối tượng được chọn hay tạo nhiều script, mỗi script dành cho một đối tượng. Các script T-SQL có thể được xuất ra một file, ra clipboard hay ra SQL Server Management Studio Query Editor.

Query Editor

Query Editor là phần thay thế cho Query Analyzer. Nó cho phép bạn viết và chạy các script T-SQL. Bạn có thể nhìn thấy Query Editor ở phần giữa trên của hình 2.2. Bạn khởi động Query Editor từ SQL Server Management Studio bằng cách hoặc chọn tùy chọn New Query trên trang Start hoặc chọn tùy chọn File > New > SQL Server Query > SQL Server Query. Query Editor hỗ trợ sự phát triển các mẫu truy vấn trong T-SQL, MDX, XMLA, và SQL Mobile Queries. Không giống như Query Analyzer, luôn hoạt động trong chế độ đã nối kết, Query Editor mới có

tùy chọn để làm việc ở chế độ nối kết hoặc không nối kết với server. Theo mặc định, nó tự động nối kết với server ngay sau khi bạn chọn tạo một mẫu truy vấn (query) mới.

Giống như bảng đối chiếu Visual Studio 2005 của nó, Query Editor hỗ trợ các từ khóa được tạo mã màu, hiển thị bằng hình ảnh các lỗi cú pháp và cho phép người phát triển vừa chạy vừa gỡ rối mã. Ngoài ra, không giống như Query Analyzer cũ, Query Editor hỗ trợ khái niệm của các dự án (project), trong đó các nhóm file liên quan có thể được nhóm lại với nhau để tạo thành một giải pháp. Query Editor mới cũng cung cấp sự hỗ trợ đầy đủ cho việc điều khiển mã nguồn nhờ dùng Visual SourceSafe. Query Editor còn thêm vào các thành phần đã có trong Query Analyzer. Nó có thể hiển thị các kết quả truy vấn ở dạng lưới hay dạng text, và nó có thể hiển thị bằng hình ảnh các kế hoạch thực thi của một mẫu truy vấn. Cũng có một tùy chọn để lưu các script T-SQL của bạn nhờ dùng sự điều khiển phiên bản SourceSafe cài sẵn. Sự điều khiển phiên bản (Version control) giúp dễ dàng phát triển nhóm bằng cách ngăn chặn nhiều nhà phát triển đồng thời thay đổi cùng một module. Mã nguồn phải được kiểm tra từ nơi lưu trữ mã trước khi nó có thể được chỉnh sửa rồi được kiểm tra trở lại, cho bạn một vị trí trung tâm để chứa mã cơ sở dữ liệu. Sự điều khiển phiên bản kết hợp với các script tạo cơ sở dữ liệu SQL đem lại một phương pháp quý giá cho việc cô lập mã nguồn kết hợp với mỗi ấn bản của schema cơ sở dữ liệu. Điều này cũng có tác dụng như là một cơ sở để so sánh schema của một cơ sở dữ liệu đã triển khai với schema mong đợi đã được lưu nhờ dùng sự điều khiển phiên bản. Ngoài ra, như bạn sẽ thấy trong mục “Các công cụ thực thi” của chương này. Query Editor còn có khả năng biểu diễn bằng đồ họa kế hoạch thực thi của một mẫu truy vấn.

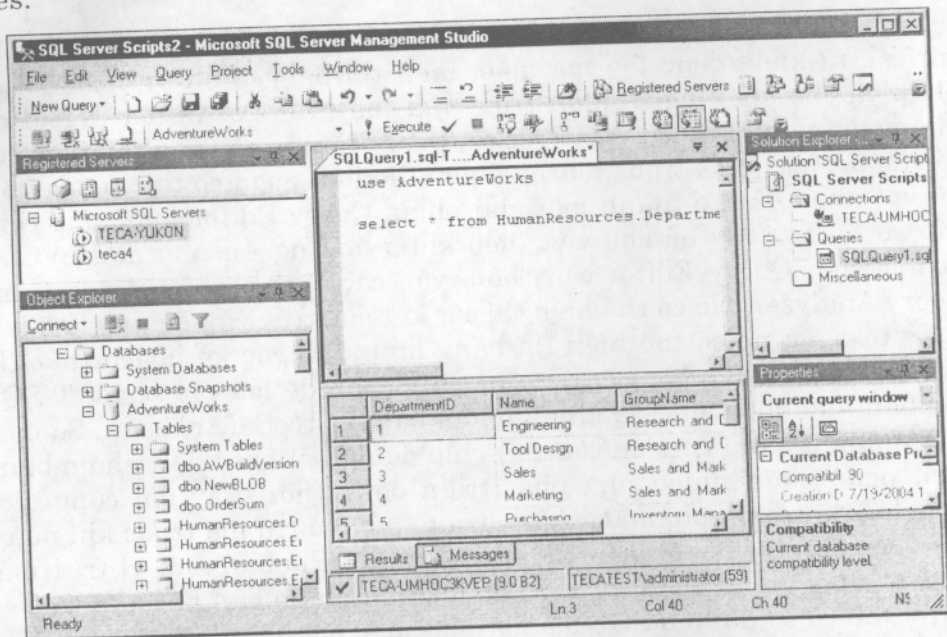
Cửa sổ Results

Cửa sổ Results của SQL Server Management Studio hiển thị kết quả của các mẫu truy vấn đã được thực thi trong Query Editor. Bạn có thể nhìn thấy cửa sổ Results ở phần phía dưới bên phải của hình 2.2. Cửa sổ Results có thể được cài đặt để hiển thị các kết quả truy vấn hoặc ở dạng text hoặc ở dạng lưới.

Solution Explorer

Một công cụ quản lý quan trọng khác được cung cấp như là thành phần của SQL Server Management Studio là Solution Explorer. Bạn có thể nhìn thấy Solution Explorer ở góc phía trên bên phải của hình 2.3. Solution Explorer được sử dụng để cung cấp một khung xem hình cây dạng phân cấp của các dự án và các file khác nhau trong một giải pháp. Mục đầu tiên được liệt kê trong Solution Explorer là tên của giải pháp SQL Server Management Studio. Theo mặc định, tên này là Solution 1, nhưng

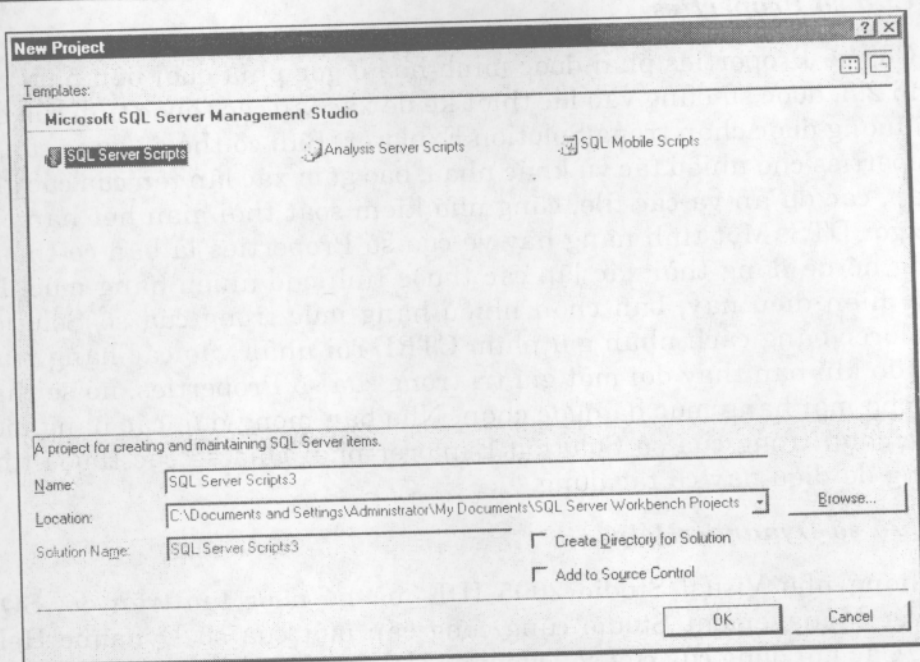
bạn có thể thay đổi tên này sang bất kỳ tên nào mà bạn muốn bằng cách chọn giải pháp và sau đó thay đổi tên của nó trong cửa sổ Properties.



Hình 2.3 SQL Server Management Studio - Solution Explorer

Bên dưới giải pháp (solution), bạn có thể có một hay nhiều mục dự án hoặc một hay nhiều file. Các file được liệt kê trong Solution Explorer có thể được kết hợp với một dự án, hoặc chúng có thể được kết hợp với chính giải pháp SQL Server Management Studio mà không cần dự án trung gian. Các file có thể bao gồm tất cả các kiểu file khác nhau được chỉnh sửa nhờ dùng SQL Server Management Studio, bao gồm T-SQL Queries, Analysis Server Queries, và XMLA Queries.

SQL Server Management Studio Solution Explorer hỗ trợ nhiều loại dự án khác nhau, bao gồm SQL Server Scripts, Analysis Server Scripts, và SQL Module Scripts. Hình 2.4 minh họa hộp thoại New Project của SQL Server Management Studio.



Hình 2.4 SQL Server Management Studio - New Project

SQL Server Scripts

Các dự án SQL Server Scripts được sử dụng để nhóm các nối kết SQL Server và các script T-SQL có liên quan lại với nhau. Một công dụng phổ biến cho loại dự án này là nhóm tất cả các script DDL (Data Definition Language) của cơ sở dữ liệu lại với nhau.

Analysis Server Scripts

Gần như giống với SQL Server Scripts được sử dụng cho các file dựa trên T-SQL, các dự án Analysis Server Scripts dùng để chứa các nối kết Analysis Server cũng như các script MDX và XMLA. Một lần nữa, một cách bạn có thể tận dụng loại dự án này là để cho dự án chứa tất cả các script vốn tạo nên kho lưu trữ dữ liệu của bạn và một dự án có thể chứa các script để tải kho lưu trữ dữ liệu của bạn.

SQL Mobile Scripts

Các dự án SQL Mobile Scripts được sử dụng để nhóm các nối kết và các mẫu truy vấn cho một cơ sở dữ liệu SQL Server CE lại với nhau. Đối với một dự án SQL Server CE, một đối tượng nối kết tiêu biểu cho nối kết với cơ sở dữ liệu CE.

Cửa sổ Properties

Cửa sổ Properties phải được minh họa ở góc phía dưới bên phải của hình 2.3, được sử dụng vào lúc thiết kế để xác lập các thuộc tính của các đối tượng được chọn trong Solution Explorer. Bạn có thể sử dụng cửa sổ Properties cho nhiều tác vụ khác nhau, bao gồm xác lập tên của các giải pháp, các dự án và các file, cũng như kiểm soát thời gian hết hạn cho các gói DTS. Một tính năng hay về cửa sổ Properties là bạn có thể sử dụng nó để đồng thời xác lập các thuộc tính cho nhiều hạng mục. Để thực hiện điều này, bạn chọn nhiều hạng mục trong cửa sổ Solution Explorer bằng cách nhấn giữ phím CTRL rồi nhấp vào các hạng mục. Sau đó khi bạn thay đổi một giá trị trong cửa sổ Properties, nó sẽ thay đổi cho mọi hạng mục đã được chọn. Như bạn mong đợi, các hạng mục được chọn trong cửa sổ Solution Explorer phải chia sẻ các thuộc tính chung để điều này có tác dụng.

Cửa sổ Dynamic Help

Giống như Visual Studio 2005 IDE mà nó được tạo trên đó, SQL Server Management Studio cũng cung cấp một cửa sổ Dynamic Help mới. Các nội dung của cửa sổ Dynamic Help tự động thay đổi theo vị trí của con trỏ hay điểm nhấp hiện hành trong SQL Server Management Studio.

Sự điều khiển nguồn

Sự điều khiển nguồn hợp nhất là một tính năng mới khác được cung cấp bởi SQL Server Management Studio. Sự điều khiển nguồn (source control) giúp dễ dàng phát triển đội hay nhóm bằng cách cung cấp một cơ chế điều khiển trung tâm quản lý sự truy cập vào các file nguồn. Để sử dụng tính năng điều khiển nguồn SQL Server Management Studio, bạn phải có cài đặt Visual SourceSafe, vốn đi kèm với sản phẩm Visual Studio 2005 của Microsoft. Tính năng điều khiển nguồn SQL Server Management Studio cung cấp một tiến trình kiểm tra vào và kiểm tra ra nhằm bảo đảm hai nhà phát triển không đồng thời thay đổi cùng một file nguồn (và có thể xóa sạch các phần chỉnh sửa của nhau). Hệ thống điều khiển nguồn cũng có khả năng lưu trữ các phiên bản trước của mỗi dự án. Bạn có thể thêm các dự án và các file solution vào hệ thống điều khiển nguồn Visual SourceSafe bằng cách sử dụng tùy chọn File > Source Control từ menu SQL Server Management Studio.

Cửa sổ Current Activity

SQL Server Management Studio mới của SQL Server 2005 còn cung cấp một tính năng giám sát tiến trình đã hợp nhất được gọi là cửa sổ Current Activity. Bạn có thể chạy Current Activity Monitor từ Object Explorer bằng cách chọn nút Database > Management > Activity Moni-

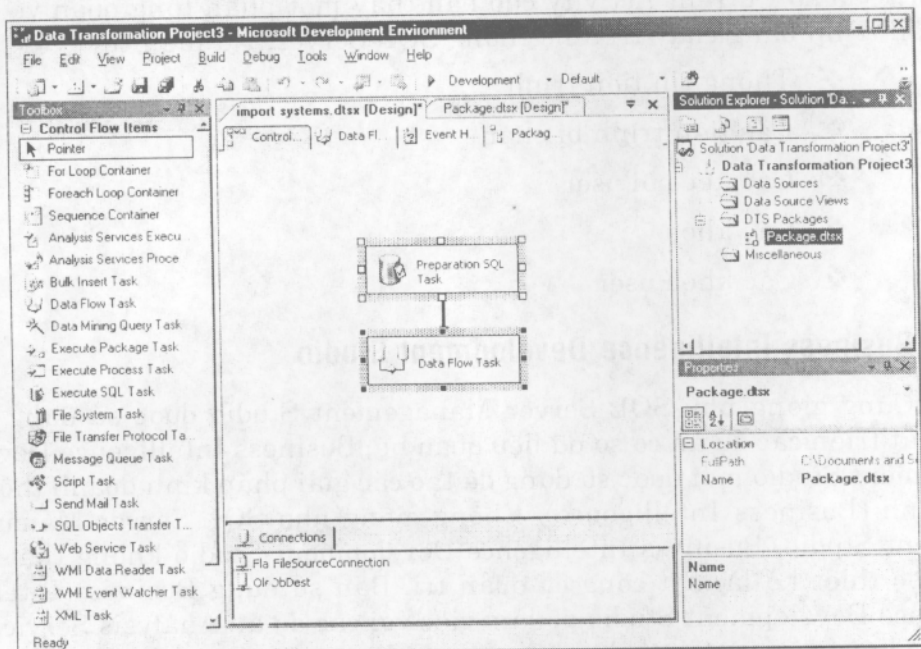
tor. Cửa sổ Current Activity cho bạn thấy một phần tổng quan về các tiến trình đang chạy trên hệ thống SQL Server. Nó hiển thị

- ❖ Thông tin tiến trình
- ❖ Các tiến trình bị chặn
- ❖ Các kết nối user
- ❖ Các khóa
- ❖ Các khóa user

Business Intelligence Development Studio

Cũng giống như SQL Server Management Studio được sử dụng để phát triển các dự án cơ sở dữ liệu quan hệ, Business Intelligence Development Studio mới được sử dụng để tạo các giải pháp kinh doanh thông minh (Business Intelligence). Không giống như SQL Server Management Studio, Business Intelligence Development Studio không thật sự được thiết kế là một công cụ quản trị. Bạn sử dụng Business Intelligence Development Studio để làm việc với các dự án Analysis Services, để phát triển và triển khai các báo cáo Reporting Services và để thiết kế các gói Data Transformation Services.

Vì Business Intelligence Development Studio có một mục đích khác với SQL Server Management Studio nên nó cũng có một diện mạo khác. Ngoài ra, trong khi SQL Server Management Studio tận dụng một chế độ nối kết theo mặc định thì Business Intelligence Development Studio bắt đầu ở một chế độ chưa nối kết. Business Intelligence Development Studio được truy cập nhờ dùng tùy chọn menu Start > Programs > Microsoft SQL Server > Business Intelligence Development Studio. Hình 2.5 minh họa một hình ảnh của Business Intelligence Development Studio.



Hình 2.5 Business Intelligence Development Studio

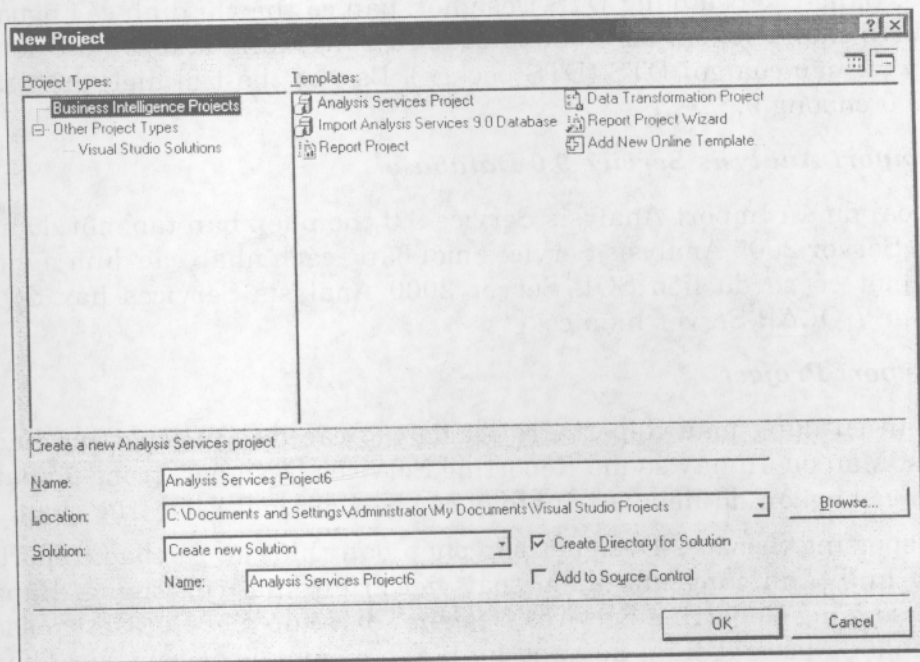
Giống như SQL Server Management Studio, Business Intelligence Development Studio được tạo trên Visual Studio 2005 IDE. Nó cung cấp một môi trường phát triển theo hướng giải pháp trong đó một hay nhiều dự án được chứa trong một giải pháp (solution). Mỗi dự án còn có các loại đối tượng phù hợp cho dự án đó. Chẳng hạn, một dự án Reporting Services sẽ chứa các định nghĩa report, trong khi đó một dự án DTS chứa các đối tượng DTS package (gói DTS). Giống như SQL Server Management Studio, Business Intelligence Development Studio không cho phép bạn biên dịch VB.NET, C#, J#, hay VC++. Thay vào đó Business Intelligence Development Studio được thiết kế để làm việc với các dự án BI như DTS và Reporting Services. Business Intelligence Development Studio cũng được hợp nhất với Visual SourceSafe để điều khiển phiên bản mã nguồn.

Cửa sổ Toolbox

Cửa sổ Toolbox trong Business Intelligence Development Studio hiển thị ở phía bên trái của màn hình trong hình 2.5. Toolbox được sử dụng bởi TDS Designer và Reporting Services Designer để rê và thả các thành phần lên các bề mặt thiết kế tương ứng của chúng.

Solution Explorer

Giống như SQL Server Development Studio, Business Intelligence Development Studio cũng có một cửa sổ Solution Explorer. Bạn có thể thấy Solution Explorer ở góc phía trên bên phải của màn hình minh họa ở hình 2.5. Solution Explorer cung cấp một khung xem hình cây dạng phân cấp của các dự án và các file khác nhau vốn tạo thành một giải pháp Business Intelligence Development Studio. Hạng mục đầu tiên trong cây phân cấp Solution Explorer là tên giải pháp. Bên dưới giải pháp, bạn có thể có một hay nhiều mục và / hoặc nhiều mục. Business Intelligence Development Studio Solution Explorer cung cấp các mẫu dự án sau: Analysis Services Project, Data Transformation Project, Import Analysis Services 9.0 Database, Reports Project, và Reports Project Wizard. Giống như trong SQL Server Management Studio, các giải pháp Business Intelligence Development Studio không bị hạn chế trong một dự án. Bạn có thể tạo các giải pháp gồm nhiều dự án vốn được tạo thành từ bất kỳ loại dự án được hỗ trợ. Hình 2.6 minh họa hộp thoại New Project của Business Intelligence Development Studio.



Hình 2..6 Business Intelligence Development Studio - New Project

Dự án Analysis Services

Các dự án Analysis Services chứa các định nghĩa cho các đối tượng trong một cơ sở dữ liệu Analysis Services. Chúng bao gồm các định nghĩa cho các nguồn dữ liệu, các khung xem nguồn dữ liệu, các hình

khối, các kích thước, các mô hình khai thác, các vai trò và các hợp ngữ. Một trong những công cụ chính trong các dự án Analysis Services là Cube Designer.

Cube Designer là một công cụ trực quan để tạo các hình khối OLAP. Nó được khởi động bằng cách nhấp đúp vào nút Cube nằm ở dưới một dự án Analysis Services hoặc bằng cách nhấp phải vào nút Cube và chọn View Designer. Bạn có thể tìm hiểu thêm thông tin về cube designer ở chương 10.

Data Transformation

Định nghĩa dự án cho các dự án Data Transformation cho phép bạn tạo các đối tượng được sử dụng trong một giải pháp DTS. Các định nghĩa này bao gồm việc tạo các nguồn dữ liệu, các khung xem nguồn dữ liệu và các gói DTS.

DTS Designer, được minh họa ở giữa màn hình trong hình 2.5, cho phép bạn tạo các gói DTS một cách trực quan bằng cách chọn các thành phần DTS từ thanh công cụ rồi rê và thả chúng lên trên bề mặt thiết kế DTS. Bằng cách sử dụng DTS Designer, bạn có thể chỉ định các nguồn dữ liệu, chọn các loại biến đổi sẽ được sử dụng, cũng như cài đặt dòng chảy dữ liệu của gói DTS (DTS package). Bạn có thể tìm hiểu thêm về DTS ở chương 9.

Import Analysis Service 9.0 Database

Loại dự án Import Analysis Service 9.0 cho phép bạn tạo một dự án SQL Server 2005 Analysis Services mới bằng cách nhập các định nghĩa cho một cơ sở dữ liệu SQL Server 2000 Analysis Services hay SQL Server 7 OLAP Server hiện có.

Report Project

Bạn sử dụng mẫu Report Project để tạo các dự án Reporting Services. Mẫu dự án này sẽ mở Reporting Services Designer, trong đó bạn chọn các nguồn dữ liệu và trình bày các báo cáo một cách trực quan.

Reporting Services Designer cho phép bạn thiết kế các báo (report) bằng hình ảnh cũng như kiểm soát việc triển khai của chúng. Bằng cách sử dụng Reporting Services Designer, bạn tạo một Dataset và sau đó sử dụng bề mặt thiết kế bằng hình ảnh cùng với các control từ hộp công cụ Reports Items để trình bày các báo cáo. Các báo cáo này có thể chạy trên một cơ sở dữ liệu SQL Server 2005 cũng như bất kỳ nguồn cơ sở dữ liệu nào tương thích OLE DB và ODBC. Reporting Services và Report Designer được trình bày chi tiết hơn ở chương 8.

Report Project Wizard

Mẫu dự án Report Project Wizard khởi động Report Wizard, vốn hướng dẫn bạn qua một loạt các màn hình dễ sử dụng nhằm cho phép bạn định nghĩa một báo cáo cơ bản. Query Builder hợp nhất được đưa vào Report Wizard để giúp bạn tạo các mẫu truy vấn SQL. Ngay sau khi bạn đã tạo một báo cáo nhờ dùng Report Wizard, bạn có thể trở lại tinh chỉnh nó nhờ dùng Report Designer. Report Wizard cũng được trình bày chi tiết hơn ở chương 8.

Cửa sổ Properties

Cửa sổ Properties phải được minh họa ở góc dưới cùng bên phải trong hình 2.5, được sử dụng vào lúc thiết kế để xác lập các thuộc tính của các đối tượng đã được chọn trong Solution Explorer.

Dynamic Help

Giống như SQL Server Management Studio, Business Intelligence Development Studio cũng có một cửa sổ Dynamic Help mới. Các nội dung trợ giúp trong cửa sổ Dynamic Help thay đổi theo vị trí của con trỏ hay điểm nhấp hiện hành trong Business Intelligence Development Studio.

Sự điều khiển nguồn

Business Intelligence Development Studio cũng cung cấp sự điều khiển nguồn (source control) hoàn toàn hợp nhất nhờ dùng Visual SourceSafe. Sự điều khiển nguồn giúp dễ dàng phát triển đội hay nhóm bằng cách cung cấp một cơ chế điều khiển trung tâm quản lý sự truy cập vào các file nguồn. Để sử dụng tính năng điều khiển nguồn với Business Intelligence Development Studio, bạn cần cài đặt Visual SourceSafe, được phân phối cùng với Visual Studio 2005. Sự điều khiển nguồn cung cấp một tiến trình check-in (kiểm tra vào) và check-out (kiểm tra ra) nhằm bảo đảm hai nhà phát triển không đồng thời thay đổi cùng một file nguồn, nó cũng có thể lưu trữ các phiên bản trước của mỗi dự án. Bạn có thể thêm các giải pháp, các dự án và các file vào hệ thống điều khiển nguồn bằng cách sử dụng tùy chọn File > Source Control từ Business Intelligence Development Studio.

Khi nào cần sử dụng SQL Server Management Studio và Business Intelligence Development Studio

Thoạt nhìn cả hai có vẻ giống nhau nhưng thực chất SQL Server Management Studio và Business Intelligence Development Studio được thiết kế để thực thi các mục đích khác nhau. Cả SQL Server Management Studio lẫn Business Intelligence Development Studio đều được tạo

dựa trên Visual Studio 2005 IDE, và vì SQL Server 2005 đi kèm với cả một server cơ sở dữ liệu quan hệ SQL Server và một server Analysis Services OLAP nên bạn có thể nghĩ rằng SQL Server Management Studio được sử dụng để quản lý SQL Server Management Studio trong khi Business Intelligence Development Studio được sử dụng để quản lý Analysis Services. Thực tế không phải như vậy. SQL Server Management Studio được sử dụng để quản lý cả SQL Server và Analysis Services cũng như để phát triển các mẫu truy vấn T-SQL và MDX. SQL Server Management Studio có thể kết nối với các trường hợp thể hiện của SQL Server, Analysis Services và Reporting Services và Reporting Services. Bạn có thể sử dụng nó để mở và ngưng các dịch vụ đó, cũng như để quản lý chúng nhờ dùng SQL Server Management Studio GUI hay bằng cách chạy các script.

Business Intelligence Development Studio không phải là một công cụ quản trị mà nó được thiết kế nhằm cho phép bạn phát triển và triển khai các giải pháp kinh doanh thông minh. Nó có thể được sử dụng để thiết kế các gói DTS và các báo cáo Reporting Services cũng như tạo và chạy các mẫu truy vấn MXD và XMLA, nhưng nó không được sử dụng để quản lý Analysis Services.

Sqlcmd

Một công cụ quản lý mối quan trọng khác trong SQL Server 2005 là tiện ích Sqlcmd mới. Tiện ích Sqlcmd về cơ bản là một sự thay thế cho các tiện ích osql và isql cũ hơn trong các ấn bản trước của SQL Server. Chương trình isql cũ đã sử dụng thư viện SQL Server DB để kết nối với SQL Server, trong khi chương trình osql sử dụng ODBC. Tiện ích Sqlcmd mới kết nối với SQL Server nhờ dùng OLE DB. Để tương thích ngược, tiện ích osql vẫn được kèm theo SQL Server 2005. Tuy nhiên, isql đã bị loại ra khỏi ấn bản SQL Server 2005. Giống như các công cụ đó, tiện ích Sqlcmd được chạy từ dòng nhắc lệnh đồng thời cho phép bạn nhập và thực thi các câu lệnh T-SQL, các thủ tục lưu trữ và các lô T-SQL.

Một tính năng quan trọng của tiện ích Sqlcmd là ngoài khả năng thực thi các lệnh, nó có thể kết nối với cơ sở dữ liệu bằng một nối kết quản trị chuyên dụng (Dedicated Administrative Connection, gọi tắt là DAC). DAC cho phép bạn kết nối và chạy ở một mức ưu tiên cao hơn bất kỳ tiến trình SQL Server khác, giúp bạn có thể kết thúc bất kỳ tiến trình. Để sử dụng DAC, bạn phải khởi động tiện ích Sqlcmd nhờ dùng lệnh **A**. Bạn có thể tìm hiểu thêm thông tin về khả năng DAC của Sqlcmd ở chương 4. Ngoài lệnh **A**, tiện ích Sqlcmd còn hỗ trợ nhiều lệnh khác. Chẳng hạn, bạn có thể sử dụng lệnh **L** để liệt kê tất cả các hệ thống SQL Server đã đăng ký, như minh họa ở đây:

Sqlcmd L

Một lệnh hữu dụng khác là `p`, lệnh này kết xuất các số liệu thống kê về sự thực thi cho tập hợp kết quả. Bạn có thể có được một danh sách đầy đủ về các lệnh được hỗ trợ bằng cách nhập `Sqlcmd /?` tại dòng nhắc lệnh.

Sqlcmd Scripting

Ngoài việc hỗ trợ sự thực thi các câu lệnh T-SQL chuẩn, tiện ích `Sqlcmd` còn hỗ trợ nhiều phần mở rộng scripting nhằm cho phép bạn đưa sự điều khiển lưu dòng và các biến vào các script của bạn. Bạn có thể định nghĩa các biến tạo script theo cách ngầm nhờ dùng câu lệnh `V`, hoặc bạn có thể xác lập chúng bằng cách dùng lệnh `setvar`. Ví dụ sau đây minh họa cách bạn có thể kết hợp các lệnh `Sqlcmd` với các biến được sử dụng trong các script:

```
sqlcmd S MySQLServer d AdventureWorks v CUSTOMERID="ALFKI" i MyScript.sql
```

Lưu ý tiện ích `Sqlcmd` sử dụng một kết nối đáng tin cậy theo mặc định.

Ví dụ này sử dụng câu lệnh `S` để chỉ định tên hệ thống SQL Server. Câu lệnh `d` chỉ định cơ sở dữ liệu mà nó sẽ được kết nối với. Câu lệnh `v` được sử dụng để định nghĩa một biến có tên là `VENDORNAME` và cung cấp cho biến đó trị của `International`. Câu lệnh `i` được sử dụng để cho tiện ích `Sqlcmd` biết rằng lệnh T-SQL sẽ đến từ file `MyScript.sql`. Bạn có thể xem các nội dung của file `MyScript.sql` trong danh sách sau:

```
select * from Purchasing. Vendor where Name = '$(VENDORNAME)'
```

Ngoài việc cung cấp các biến do người dùng ấn định, Microsoft còn cung cấp một tập hợp các biến đã có sẵn để sử dụng với các script `Sqlcmd`. Bảng 2.1 liệt kê các biến mở rộng mà Microsoft đã thêm vào tiện ích `Sqlcmd` và các câu lệnh được sử dụng để cung cấp các trị cho các biến đó.

Bảng 2.1 Các biến mở rộng `Sqlcmd`

Biến	Câu lệnh
<code>SQLCMDUSER</code>	<code>-U</code>
<code>SQLCMDPASSWORD</code>	<code>-P</code>
<code>SQLCMDSERVER</code>	<code>-S</code>
<code>SQLCMDWORKSTATION</code>	<code>-H</code>
<code>SQLCMDDBNAME</code>	<code>-d</code>
<code>SQLCMDLOGINTIMEOUT</code>	<code>-l</code>

SQLCMDSTATTIMEOUT	-t
SQLCMDHEADERS	-h
SQLCMDCOLSEP	-s
SQLCMDCOLWIDTH	-w
SQLCMDPACKETSIZE	-a
SQLCMDERRORLEVEL	-m

Ngoài việc cung cấp một tập hợp các biến có sẵn, tiện ích `Sqlcmd` còn cung cấp một danh sách các lệnh mở rộng. Bảng 2.2 liệt kê các lệnh mở rộng của các tiện ích `Sqlcmd`.

Bảng 2.2 Các tiện ích mở rộng `Sqlcmd`

Lệnh	Mô tả
[:]GO [count]	Báo hiệu kết thúc một lô (batch) và thực thi các câu lệnh trong cache. Việc thêm một trị đếm tùy ý sẽ thực thi các câu lệnh với số lần đã cho.
[:]RESET	Xóa sạch các câu lệnh.
[:]ED	Bắt đầu lần hiệu chỉnh kế tiếp cho các câu lệnh hiện hành.
[:]!!!	Thực thi các lệnh hệ điều hành.
[:]QUIT	Kết thúc tiện ích <code>Sqlcmd</code> .
[:]EXIT (results)	Sử dụng trị của một tập hợp kết quả làm trị trả về.
[:]r <filename>	Đưa vào các câu lệnh <code>Sqlcmd</code> bổ sung từ file được chỉ định.
:/ServerList	Liệt kê các hệ thống SQL Server đã cấu hình.
:List	Liệt kê các nội dung trong các câu lệnh.
:Error <filename>	Hướng lại kết xuất lỗi sang file đã chỉ định.
:Out	Hướng lại các kết quả truy vấn sang file đã chỉ định.
:Perftrace <filename>	Hướng lại các số liệu thống kê về sự thực thi sang file đã chỉ định.
:Connect [timeout]	Kết nối với một instance SQL Server.
:On Error [exit > retry > ignore]	Chỉ định hành động sẽ được thực hiện khi gặp một lỗi.

:XML [ON > OFF]

Chỉ định các kết quả XML có được kết xuất ở dạng một dòng liên tục hay không.

Danh sách sau đây cho bạn thấy cách sử dụng các lệnh mở rộng của các tiện ích Sqlcmd. Ví dụ này kết nối với server và sau đó sử dụng lệnh: :EXIT để cho ra số đếm hàng từ table customers:

```
C:\>sqlcmd
```

```
1> use adventureworks
```

```
2> :EXIT(select count(*) from sales.customer)
```

```
Changed database context to 'AdventureWorks'.
```

91

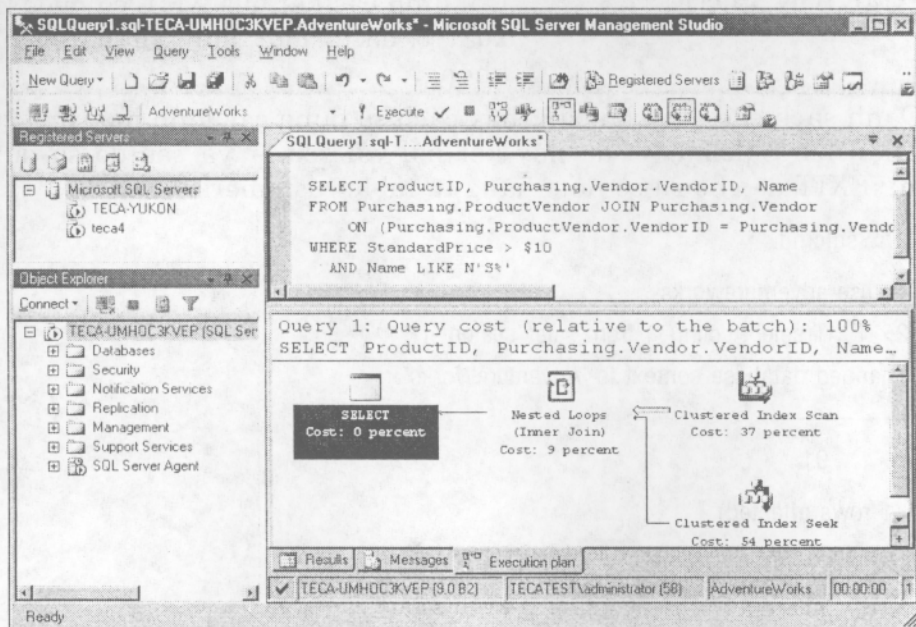
(1 rows affected)

Các công cụ thực thi

Ngoài bộ công cụ quản lý mới, SQL Server 2005 còn cung cấp một bộ công cụ đã cập nhật để phân tích và cải tiến hiệu suất của server. Trong năm năm kể từ ấn bản gần đây nhất của SQL Server, Microsoft đã thêm vào nhiều tính năng cho các công cụ quản lý đi kèm với SQL Server và các công cụ thực thi cũng không ngoại lệ. Trong mục này, bạn sẽ xem xét bốn công cụ quản lý sự thực thi mới và quan trọng nhất trong SQL Server 2005: Showplan đồ họa của SQL Server Management Studio Query Editor, Database Tuning Advisor và Profiler.

SQL Server Management Studio Query Editor: Các kế hoạch thực thi

Giống như Query Analyzer cũ, Query Editor mới có khả năng biểu diễn bằng đồ họa kế hoạch thực thi cho một mẫu truy vấn (query). SQL Server Management Studio Query Editor mới cũng mở rộng khả năng này bằng cách cho phép bạn hiển thị một kế hoạch thực thi ước tính cũng như kế hoạch thực thi thật sự. Hình 2.7 minh họa một kế hoạch thực thi mẫu.



Hình 2.7 Query Editor Showplan

Một số tính năng mới đã được thêm vào Showplan đồ họa của Query Editor bao gồm một bộ biểu tượng mới đã được thiết kế lại. Sự tạo mã màu mới tuân theo một mẫu nhất định. Tất cả các bộ lặp có màu xanh dương, con trỏ có màu vàng và các cấu trúc ngôn ngữ SQL có màu xanh lục. Các trị tham số được sử dụng cho các mẫu truy vấn tham số hóa sẽ được hiển thị trong Showplan. Có thông tin cải tiến cho các mẫu truy vấn được thực thi song song nơi Showplan có thể cho thấy có bao nhiêu hàng được tạo theo từng luồng (thread). Ngoài ra, thông tin Showplan mới cũng sẽ hiển thị khi nào các phần mở rộng do người dùng ấn định (UDX) CLR được sử dụng như là thành phần của một mẫu truy vấn.

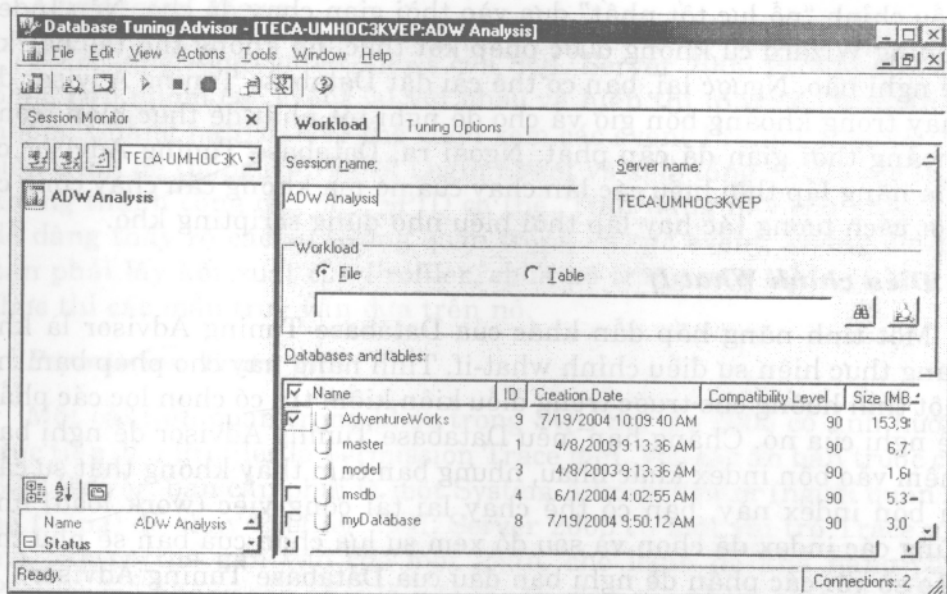
Showplan XML có thể xuất

Không giống như Query analyzer cũ, Query Editor còn có khả năng xuất các kết quả Showplan ở dạng một tài liệu XML. Tài liệu Showplan XML này sau đó được nhập vào bởi bộ phận hỗ trợ của Microsoft hay một cá nhân hỗ trợ kỹ thuật khác để xử lý các vấn đề liên quan đến sự thực thi mẫu truy vấn. Các Showplan XML được tạo là schema đã được hợp chuẩn và dạng Showplan XML mới thì dễ chuyển đổi và có thể email đến một site từ xa để xem lại. Nó có thể được nhập và được lập đồ thị bởi một user từ xa mà không đòi hỏi sự truy cập vào cơ sở dữ liệu nguồn. Việc sử dụng dạng XML cho kết xuất Showplan đem lại một vài ưu điểm vượt trội hơn các dạng text và đồ họa. Nó có thể được xử lý bằng cách

sử dụng bất kỳ công nghệ XML, chẳng hạn như XQuery, XPath, DOM hay SAX. Nếu bạn hỗ trợ các cơ sở dữ liệu di động, một điểm hay khác là SQL Server CE sử dụng cùng dạng này. Một điểm quan trọng cần lưu ý là tuy dạng xuất sang XML mới có sẵn nhưng bạn không nhất thiết cần sử dụng nó. Các dạng Showplan text và đồ họa cũ hơn vẫn được hỗ trợ.

Database Tuning Advisor

Database Tuning Advisor là một tính năng mới khác trong SQL Server 2005, nó cho phép bạn tinh chỉnh hiệu suất thực thi của các ứng dụng cơ sở dữ liệu. Database Tuning Advisor về cơ bản là một phiên bản cập nhật của Index Tuning Wizard vốn đã có trong các ấn bản trước của SQL Server. Tuy nhiên, Database Tuning Advisor mới còn vượt xa hơn là chỉ đưa ra các index mới, và đó là một trong những lý do để nó được đặt tên mới này. Database Tuning Advisor có khả năng hoạt động với các phân chia (partition); nó hỗ trợ sự thực thi theo thời biểu cũng như cải thiện sự phân tích tình huống what-if. Lý do khác là Database Tuning Advisor bây giờ là một ứng dụng đầy đủ tính năng hơn là chỉ một loạt các hộp thoại Wizard. Bạn có thể xem một ví dụ của Database Tuning Advisor ở hình 2.8.



Hình 2.8 Database Tuning Advisor

Đề nghị chia phần

Ngoài đề nghị index (chỉ mục), Database Tuning Advisor mới còn hỗ trợ đầy đủ sự chia phần dữ liệu SQL Server 2005. Database Tuning

Advisor có thể đề nghị sử dụng cả hai phần chia đã canh chỉnh và chưa canh chỉnh (phần đã canh chỉnh sử dụng một index đã được nhóm cụm trong cùng thứ tự với table, trong khi phần chưa canh chỉnh sử dụng một index chưa được nhóm cụm trong một thứ tự khác với table cơ sở.) Ngoài khả năng đề nghị thêm các phần chia (partition), Database Tuning Advisor còn đề nghị loại bỏ các phần chia hiện có.

Tạm ngưng và tiếp tục

Index Tuning Wizard cũ, giống như một ứng dụng wizard tiêu biểu, hoạt động trong chế độ chạy một lần nơi bạn phải qua các màn hình wizard từ đầu đến cuối. Ngược lại, Database Tuning Advisor mới có thể được ngưng và bắt đầu lại với trạng thái đã lưu. Thậm chí bạn có thể tiếp tục trạng thái trên các máy trạm client khác.

Điều chỉnh theo thời biểu

Database Tuning Advisor hỗ trợ hai tình huống khác nhau cho việc điều chỉnh theo thời biểu. Trước tiên, không giống như Index Tuning Wizard cũ, vốn có thể chạy trong những thời hạn lâu dài trong khi gặp các đề nghị index tối ưu, Database Tuning Advisor mới có thể được cài đặt để chạy trong một khoảng thời gian đã định rồi chuyển một đề nghị điều chỉnh “nỗ lực tốt nhất” dựa vào thời gian chạy đã cho. Nếu Index Tuning Wizard cũ không được phép kết thúc, nó không thể tạo bất kỳ đề nghị nào. Ngược lại, bạn có thể cài đặt Database Tuning Advisor để chạy trong khoảng bốn giờ và cho đề nghị tốt nhất để thực hiện trong khoảng thời gian đã cấp phát. Ngoài ra, Database Tuning Advisor có khả năng lập thời biểu các lần chạy của nó mà không cần chạy công cụ một cách tương tác hay lập thời biểu nhờ dùng scripting khó.

Điều chỉnh What-If

Một tính năng hấp dẫn khác của Database Tuning Advisor là khả năng thực hiện sự điều chỉnh what-if. Tính năng này cho phép bạn thử một tình huống cho trước trong điều kiện kiểm tra có chọn lọc các phần đề nghị của nó. Chẳng hạn, nếu Database Tuning Advisor đề nghị bạn thêm vào bốn index khác nhau, nhưng bạn cảm thấy không thật sự cần cả bốn index này, bạn có thể chạy lại tải công việc (work load) nhờ dùng các index đã chọn và sau đó xem sự lựa chọn của bạn sẽ như thế nào so với các phần đề nghị ban đầu của Database Tuning Advisor.

Các cải tiến Profiler

Tuy không phải là một tính năng mới, nhưng Profiler đã được cải tiến trong ấn bản mới. Một trong những thay đổi cơ bản nhất trong Profiler là ở mức độ kiến trúc. Profiler được gắn liền với bộ máy SQL Server và trong các ấn bản trước, khi các phần cập nhật được áp dụng

cho bộ máy SQL Server ảnh hưởng đến thông tin do Profiler sử dụng, một phần cập nhật tương ứng cần được áp dụng cho các hệ thống client dùng để chạy Profiler. Với ấn bản SQL Server 2005, Microsoft đã trích xuất định nghĩa của các event và các cột hợp lệ từ bộ máy SQL Server và chứa chúng trong các file XML. Khi Profiler khởi động lần đầu tiên, nó kiểm tra phiên bản của SQL Server mà nó đang nối kết với. Nếu Profiler nối kết với một phiên bản đã biết của SQL Server thì nó sẽ sử dụng các định nghĩa đã được lưu trong cache của nó. Nếu nó nối kết với một phiên bản đã cập nhật của SQL Server thì nó tải xuống thông tin tạo profile XML mới. Điều này cho phép Profiler hoạt động liên tục với các phiên bản đã cập nhật của SQL Server không cần các phần cập nhật vào client Profiler mỗi lần một phần cập nhật áp dụng cho bộ máy SQL Server.

Tạo profile Analysis Services

Một trong những thay đổi lớn nhất trong Profiler dành cho SQL Server 2005 là khả năng tạo profile các câu lệnh MDX chạy trên Analysis Services. Profiler mới có thể hoạt động trên một instance SQL Server 2005 Analysis Services giống như cách nó đã hoạt động với bộ máy SQL Server quan hệ trong các ấn bản trước.

Aggregated Views

Tính năng Aggregated Views của SQL Server 2005 Profile mới cho phép bạn nhóm các event lại với nhau và hiển thị tổng số cho toàn bộ nhóm. Chẳng hạn, bạn có thể chọn xem các event Profiler theo ID của tiến trình hệ thống (SPID) và Login Event và xem tổng số của các login không thành công của một SPID cụ thể. Tính năng này cho phép bạn dễ dàng thấy rõ các xu hướng quan trọng và các event, không đòi hỏi bạn phải lấy kết xuất của Profiler, chứa nó trong một table và sau đó thực thi các mẫu truy vấn dựa trên nó.

Permission Trace mới

Một cải tiến quan trọng khác trong SQL Server 2005 có ảnh hưởng trực tiếp đến Profiler là permission Trace mới. Với các ấn bản trước của SQL Server, bạn cần phải là một System Administrator (người quản trị hệ thống) để chạy Profiler. SQL Server 2005 có một vai trò Trace mới mà bạn có thể gán vào cho một login, cho login đó khả năng chạy Profiler mà không cần phải là một thành viên của nhóm system administrator. Rõ ràng, bởi vì Profiler cho phép truy cập mọi dữ liệu nên bạn cần phải thật thận trọng khi cấp phép permission Trace.

Trích xuất dữ liệu theo loại event

SQL Server 2005 Profiler mới còn có khả năng trích xuất thông tin

từ kết xuất Profiler theo loại Event. Chẳng hạn, tính năng này cho phép bạn trích xuất mọi câu lệnh T-SQL từ một phiên làm việc (session) Profiler đã cho và sau đó viết các event ở dạng một file .sql có thể được thực thi nhờ dùng SQL Server Management Studio. Profiler cũng cung cấp tính năng này cho các câu lệnh MDX và DMX. Một công dụng thật sự hấp dẫn khác cho tính năng này là khả năng trích xuất thông tin deadlock. Thông tin này sau đó có thể được trình bày trong khung xem đồ họa bằng cách sử dụng Profiler.

Create Trace Wizard

Một tính năng mới khác trong SQL Server 2005 là sự giới thiệu lại của Trace Wizard. Trace Wizard ban đầu được giới thiệu trong SQL Server 7 nhưng sau đó bị loại ra khỏi ấn bản SQL Server 2000. Microsoft đã thêm Create Trace Wizard vào trở lại SQL Server 2005. Về cơ bản, Create Trace Wizard giúp dễ dàng chạy các Trace bằng cách hướng người quản lý qua các tình huống trace (truy vết) thông thường.

Các cơ cấu quản lý mới

Ngoài các công cụ quản lý mới trong SQL Server 2005, cơ cấu quản lý SQL Server cũng đã được đổi mới hoàn toàn. Một cơ cấu quản lý mới dựa vào .NET được gọi là System Management Objects (SMO) có thể dùng để viết các ứng dụng quản lý server tùy ý cho SQL Server 2005. Các API quản lý mới dựa vào .NET cũng đã được thêm vào để quản lý Analysis Services và sự sao lập. Một sự thay đổi có ý nghĩa khác cho bộ công cụ quản trị SQL Server 2005 là sự hỗ trợ cấu hình WMI và các event WMI. Trong mục tiếp theo của chương này, bạn sẽ tìm hiểu về mỗi cơ cấu quản lý mới này trong SQL Server 2005.

SQL Server Management Objects (SMO)

Giống như Distributed Management Objects (DMO), cơ cấu quản lý đối tượng SMO được thiết kế để có thể quản lý bằng lập trình mọi khía cạnh của SQL Server. Tuy nhiên, không giống như DMO, vốn dựa vào COM, SMO mới đã được thực thi như là một thư viện lớp .NET. Điều này có nghĩa là SMO đòi hỏi .NET Framework phải được cài đặt trên các hệ thống dùng để chạy các ứng dụng quản lý SMO. SMO có thể được sử dụng để quản lý các hệ thống SQL Server 7 và SQL Server 2000 cũng như SQL Server 2005.

Với tính tương thích ngược, DMO sẽ tiếp tục được hỗ trợ bởi SQL Server 2005 nhưng nó không được cải tiến để hỗ trợ các tính năng mới trong ấn bản mới này. Chẳng hạn, DMO không thể được sử dụng để quản lý SQL Service Broker, tính năng HTTP Endpoints mới, các kiểu dữ liệu XML và varbinary(max) hay bất kỳ tính năng mới khác mà

Microsoft đã thêm vào ấn bản SQL Server 2005. Nói cách khác, DMO chỉ hỗ trợ những tính năng đã có trong các ấn bản trước của SQL Server. Cơ cấu quản lý đối tượng SMO mới cung cấp hơn 150 lớp mới tiêu biểu cho các tính năng mới trong SQL Server 2005. Để giúp dễ dàng chuyển các ứng dụng DMO cũ hơn sang SMO, cơ cấu quản lý đối tượng do SMO sử dụng gần như dựa vào cùng cơ cấu đối tượng đã được sử dụng bởi DMO. Tuy các cơ cấu đối tượng này không đồng nhất nhưng chúng khá giống nhau. Chẳng hạn, cả SMO và DMO đều sở hữu các đối tượng tiêu biểu cho các server, các dữ liệu, các table, các cột và các đối tượng cơ sở dữ liệu chính khác.

Ngoài khả năng truy cập mọi tính năng mới trong SQL Server 2005, cơ cấu đối tượng SMO mới còn có hai tính năng mới quan trọng khác. Thứ nhất, SMO sử dụng sự thể hiện tối ưu hóa của các đối tượng, nghĩa là nó có thể tạo một instance thể hiện một đối tượng cấp cao như một đối tượng server mà không cần truy xuất mọi thuộc tính tạo nên đối tượng đó. Trong trường hợp của một đối tượng giống như một server, điều này có nghĩa là SMO không cần truy xuất mọi thông tin cơ sở dữ liệu cùng với mọi thông tin đối tượng cơ sở dữ liệu cho tất cả cơ sở dữ liệu trên server khi đối tượng server được tạo. Sự thể hiện đầy đủ bị trì hoãn cho đến khi đối tượng được tham chiếu theo cách tường minh. Sự thể hiện tối ưu hóa (Optimized instantiation) dẫn đến hiệu suất thực thi tốt hơn nhiều - đặc biệt đối với các phần thực thi VLDB, SMO cũng có khả năng thu chụp và tạo lô các nhóm câu lệnh SQL lại với nhau để đạt được hiệu suất tốt hơn. Ngoài ra còn có một khả năng tạo script cải tiến được kích hoạt bởi lớp Scripter mới. Lớp Scripter này có thể phát hiện và tạo các script cho các đối tượng cơ sở dữ liệu và các phần lệ thuộc của chúng.

SMO được thực thi trong các namespace Microsoft SqlServer Management Smo Microsoft SqlServer Management Smo Common, và Microsoft SqlServer Management Smo Agent namespace.

Analysis Management Objects (AMO)

Cũng giống như SMO được sử dụng để quản lý các server SQL Server, Analysis Management Objects (AMO) được sử dụng để quản lý Analysis Services. Trong số những thứ khác, AMO có thể dùng để quản lý các server Analysis Services, các nguồn dữ liệu, các hình khối, các kích thước, các số đo và các mô hình khai thác dữ liệu. Giống như SMO, AMO là một cơ cấu đối tượng dựa vào .NET, nó đòi hỏi phải có CLR để chạy. Về mặt khái niệm, AMO dựa vào thư viện COM-based Decision Support Objects (DSO) cũ hơn. Giống như với DMO, SQL Server 2005 vẫn hỗ trợ DSO cho tính tương thích ngược của các ứng dụng hiện có. AMO tạo ra các thông báo gốc XML vốn có thể được sử dụng để tạo hay

thay đổi các đối tượng Analysis Server. Các thông báo XML này có thể được lưu ở dạng các tài liệu XML. AMO được thực thi trong namespace Microsoft.SqlServer.Management.AnalysisServices.

Replication Management Objects (RMO)

Giống hệt như SMO được thiết kế để kiểm soát mọi khía cạnh cấu hình và quản lý SQL Server, Replication Management Objects (RMO) mới dựa trên .NET được thiết kế để cấu hình và quản lý tiến trình sao lập cơ sở dữ liệu trong SQL Server 2005. Giống như SMO và AMO, các ứng dụng RMO yêu cầu phải có CLR. Cơ cấu đối tượng RMO mới chứa các lớp cho phép quản lý các tiến trình sao chụp, quản trị, đồng bộ hóa và giám sát SQL Server. RMO được thực thi trong namespace Microsoft.SqlServer.Replication.

Windows Management Instrumentation (WMI)

SQL Server 2005 cũng có thể được quản lý nhờ dùng Windows Management Instrumentation (WMI) API. WMI là chuẩn quản lý sản phẩm chéo của Microsoft nhưng trước ấn bản SQL Server 2005, WMI đã quản lý SQL Server không hiệu quả bằng DMO hay SQL Server Enterprise Manager. Các khả năng quản lý WMI của SQL Server 2005 đã được mở rộng với một WMI Configuration Provider mới và các event WMI mới.

WMI Configuration Provider

WMI Configuration Provider của SQL Server 2005 cho phép một ứng dụng quản lý từ xa kết nối với SQL Server và thực hiện các chức năng quản lý. WMI configuration API được thiết kế để thực hiện chức năng ở dạng chưa kết nối. Provider WMI được thực thi trong namespace Microsoft.SqlServer.Smo.Wmi.

Các event WMI

Các event WMI mới của SQL Server 2005 cho phép SQL Server được giám sát bởi Microsoft Operations Manager (MOM) và các ứng dụng quản lý tuân theo WMI khác của nhóm thứ ba. SQL Server 2005 có thể đưa ra các event WMI cho hầu như mọi hoạt động có thể được theo dõi nhờ dùng Profiler. Chẳng hạn, các event được tạo cho các hoạt động create database, drop database, alter database, và mọi hoạt động khác có thể được theo dõi. Cũng giống như khi sử dụng Trace, một số chi phí phát sinh khi bạn sử dụng các event WMI, nhưng chúng cung cấp thông tin quản lý rất phong phú. Các event WMI có thể được thiết lập bởi SQL Server, Analysis Services, DTS, và Reporting Services.

CHƯƠNG 3

Tính khả dụng và các tính năng phục hồi

Ấn bản SQL Server 2005 đã tiếp tục cải tiến tính khả dụng của SQL Server bằng cách tăng các khả năng nhóm cụm (cluster) của nó và đưa ra các tùy chọn cơ sở dữ liệu mới bảo đảm tính khả dụng liên tục và cải tiến các khả năng phục hồi cơ sở dữ liệu. Tính khả dụng cao tương đối dễ đạt được trên một quy mô nhỏ nhưng khó dần theo phép số mũ khi kích cỡ của hệ thống tăng lên. SQL Server 2005 cung cấp nhiều tính năng mới cải tiến tính khả dụng cơ sở dữ liệu và khả năng phục hồi bằng việc giải quyết các trở ngại chính vốn gây trở ngại cho sự khả dụng của cơ sở dữ liệu cấp doanh nghiệp. Một số yếu tố cản trở sự khả dụng của cơ sở dữ liệu trong các phiên bản trước của SQL Server bao gồm thời gian bị trì hoãn, nhu cầu có sự truy cập cơ sở dữ liệu độc nhất cho các hoạt động bảo trì đã chọn lọc, và thỉnh thoảng gặp khó khăn trong việc làm cho server phản hồi lúc CPU đang được sử dụng nhiều, chẳng hạn như khi mã của một người dùng nào đó đi vào một vòng lặp vô hạn. Một yếu tố khác có tác dụng ngược đến sự khả dụng của cơ sở dữ liệu là nâng cấp phần cứng. Tuy nhật ký phần cứng là hoạt động có trong kế hoạch nhưng chúng thường đòi hỏi hệ thống ngưng hoạt động để thực hiện việc nâng cấp. SQL Server 2005 có một khả năng mới được thiết kế để giảm thời gian ngưng hoạt động mà một trong các tiến trình nâng cấp phần cứng phổ biến nhất yêu cầu. Trong chương này, bạn sẽ xem xét về tính khả dụng mới này và các tùy chọn phục hồi trong SQL Server 2005 để bạn có thể hiểu các tính năng này được sử dụng như thế nào trong việc thực thi SQL Server 2005 trong một môi trường cơ sở dữ liệu sản xuất có tính khả dụng và khả năng phục hồi cao.

Sự bảo vệ để tránh lỗi cơ sở dữ liệu hay Server

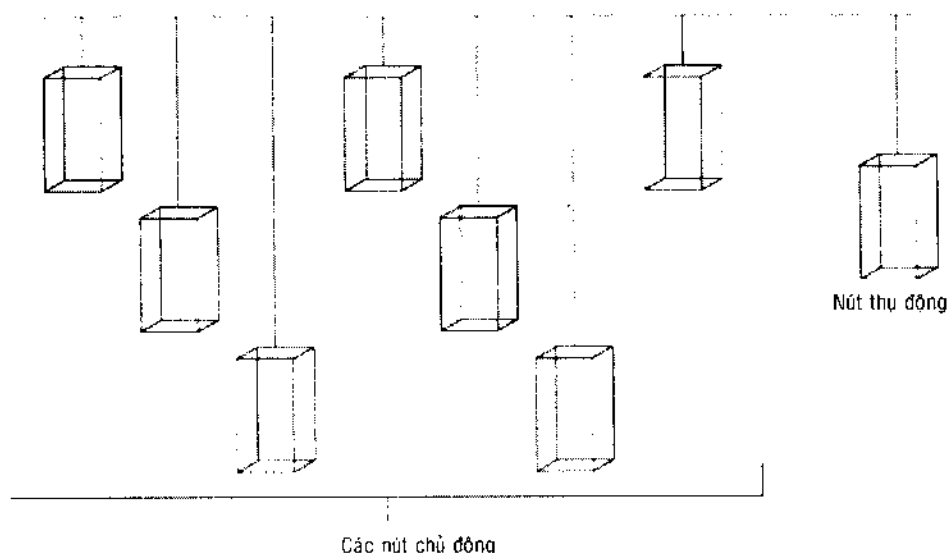
Một sự cố server cơ sở dữ liệu xảy ra hoặc do lỗi phần cứng hoặc do một vấn đề phần mềm mô phỏng server không thể hoạt động trong một khoảng thời gian nào đó. Sự cố server cơ sở dữ liệu cũng có thể do các yếu tố môi trường gây ra như thảm họa. Trong mục này, bạn sẽ tìm hiểu một số tính năng mới quan trọng nhất mà SQL Server 2005 cung cấp để xử lý các sự cố server cơ sở dữ liệu.

Cải tiến sự nhóm cụm Failover

Một ưu điểm chính của tính khả dụng cao là cải thiện đáng kể sự hỗ trợ cho failover clustering (nhóm cụm failover). Nhờ tận dụng sự nhóm cụm cải tiến trong Windows Server 2003, bây giờ SQL Server 2005 có thể được thực thi trên các cụm (cluster) có đến tám nút trên Windows Server 2003 Datacenter Edition. Ngoài ra, SQL Server 2005 còn hỗ trợ sự nhóm cụm bốn nút trên Windows Server 2003 Enterprise Edition và Windows 2000 Datacenter Server. Sự nhóm cụm tối đa hai nút được hỗ trợ trong Windows 2000 Advanced Server.

Với các dịch vụ nhóm cụm Windows, mỗi server trong cụm (cluster) được gọi là một nút (node). Tất cả các nút trong một cluster ở trạng thái giao tiếp liên tục. Nếu một trong các nút trong cluster trở nên không có sẵn, một nút khác sẽ tiếp tục nhiệm vụ của nó và bắt đầu cung cấp các dịch vụ giống như nút đã bị thất bại. Tiến trình này được gọi là failover. Các user đang truy cập cluster được hoán đổi tự động sang nút mới.

Sự nhóm cụm (clustering) được cài đặt bằng hai cách cơ bản: Active-Active clustering (nhóm cụm chủ động - chủ động), trong đó tất cả các nút đều làm việc, hoặc Active-Passive (chủ động - thụ động), trong đó một trong các nút tạm nghỉ cho đến khi nút chủ động bị thất bại. Windows Server 2003 cũng hỗ trợ các cấu hình N+1 (N hoạt động với 1 rảnh), vốn cung cấp một tình huống nhóm cụm rất linh hoạt và hiệu quả về mặt chi phí để kích hoạt những ứng dụng có tính khả dụng cao. Chẳng hạn, với một cluster tám nút trong một cấu hình N+1, bạn có bảy trong tám nút được cài đặt là có sẵn và cung cấp các dịch vụ khác nhau trong khi nút thứ tám là một nút thụ động (passive) có thể có các dịch vụ của bất kỳ một trong bảy nút chủ động (active). Hình 3.1 minh họa một cluster tám nút, trong đó có bảy nút chủ động và một nút đang chờ hoạt động nếu có bất kỳ nút nào trong số bảy nút chủ động bị thất bại.



Hình 3.1: Sự hỗ trợ cluster tám nút

Một số cải tiến về sự nhóm cụm trong SQL Server 2005 bao gồm sự hỗ trợ thiết đặt cluster tự động. Ngoài ra, tất cả các dịch vụ khác trong SQL Server 2005 đều nhận biết cluster, bao gồm:

- ❖ Database Engine
- ❖ Analysis services
- ❖ Reporting services
- ❖ Notification service
- ❖ SQL Server Agent
- ❖ Full-Text Search
- ❖ Service Broker
- ❖ SQLiMail

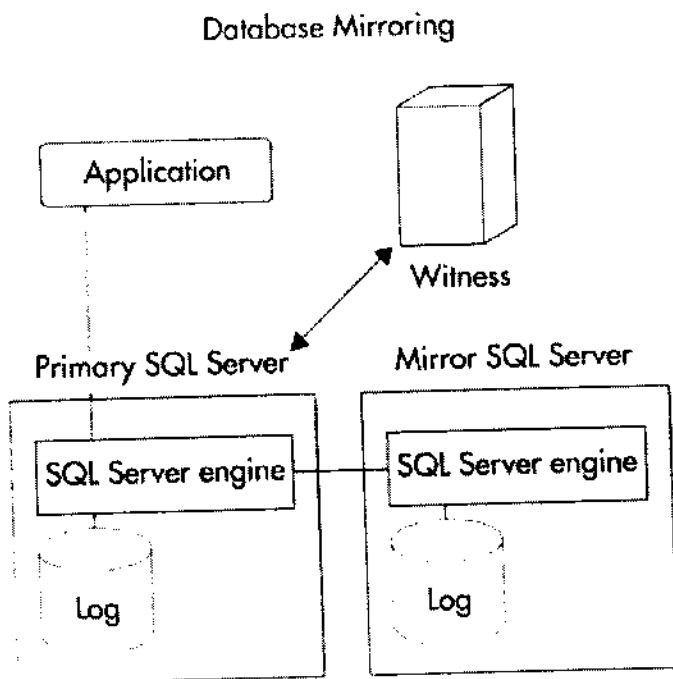
Ngoài ra, tất cả các công cụ quản lý chính trong SQL Server 2005 cũng nhận biết cluster, bao gồm:

- ❖ SQL Server Management Studio
- ❖ Service Control Manager
- ❖ SQL Profiler
- ❖ SQL Query Analyzer

Bạn có thể tìm hiểu thêm về các công cụ quản lý SQL Server 2005 ở chương 1.

Database Mirroring

Có lẽ tính năng mới quan trọng nhất trong vùng khả dụng là sự hỗ trợ Database Mirroring của SQL Server 2005. Tính năng Database Mirroring (phản chiếu hay tạo đối xứng gương cơ sở dữ liệu) giúp tránh các lỗi cơ sở dữ liệu hay server bằng cách cho SQL Server 2005 một khả năng dự phòng khẩn (instant standby capability). Database Mirroring về cơ bản cung cấp failover ở cấp cơ sở dữ liệu. Trong trường hợp cơ sở dữ liệu chính bị hỏng, gương phản chiếu cơ sở dữ liệu (database mirror) cho phép một cơ sở dữ liệu phòng hờ thứ hai có sẵn trong khoảng 2-3 giây. Database Mirroring không làm thất thoát dữ liệu và cơ sở dữ liệu được phản chiếu sẽ luôn cập nhật với giao tác hiện hành đang được xử lý trên server cơ sở dữ liệu chính. Việc chạy Database Mirroring không ảnh hưởng nhiều đến công suất giao tác. Sự hỗ trợ failover có thể được cài đặt để thực hiện tự động hay bằng tay. Chế độ manual failover thì tốt cho việc kiểm tra, nhưng bạn sẽ muốn các cơ sở dữ liệu sản xuất của bạn có khả năng failover tự động. Database Mirroring hoạt động với mọi thiết bị phần cứng chuẩn hỗ trợ SQL Server - không cần các hệ thống đặc biệt, server chính và server được phản chiếu không cần đồng nhất. Ngoài ra, không giống như trong các giải pháp nhóm cụm với tính khả dụng cao, không cần bất kỳ sự lưu trữ chia sẻ giữa server chính (primary server) và server được phản chiếu (mirrored server). Bạn có thể xem một phần tổng quan về cách hoạt động của tính năng Database Mirroring mới ở hình 3.2.



Hình 3.2: Database Mirroring

Database Mirroring được thực thi nhờ dùng ba hệ thống: server chính, server phản chiếu, và vật chứng. Đừng nhầm lẫn giữa các tên này. Server chính (primary server) chỉ là tên của hệ thống hiện đang cung cấp các dịch vụ cơ sở dữ liệu. Mọi nối kết client đến được thực hiện cho primary server. Công việc của server phản chiếu (mirror server) là duy trì một bản sao cơ sở dữ liệu được phản chiếu của primary server. Tùy thuộc vào cấu hình của server được phản chiếu, primary server và mirror có thể liên tục hoán đổi vai trò cho nhau. Vật chứng (witness) về cơ bản hoạt động như là một bên thứ ba độc lập, giúp xác định hệ thống nào sẽ có vai trò của primary server. Mỗi hệ thống được bầu chọn để làm primary server. Cần phải có hai phiếu bầu để quyết định primary server. Điều này quan trọng bởi vì có khả năng các cuộc giao tiếp giữa primary server và mirror server có thể bị ngắt, trong trường hợp này mỗi hệ thống sẽ tự bầu chọn chính nó là primary server. Vật chứng phải bỏ phiếu quyết định.

Database Mirroring hoạt động bằng cách gửi các log (nhật trình) giao tác giữa primary server và mirroring server. Vì vậy về bản chất, tính năng Database Mirroring mới là một ứng dụng chuyển log thời gian thực. Database Mirroring có thể được cài đặt với một cơ sở dữ liệu hoặc nó có thể được cài đặt cho nhiều cơ sở dữ liệu. Khi một hệ thống client viết một yêu cầu ra primary server, yêu cầu đó thật sự được viết vào log file của primary server trước khi được viết vào file dữ liệu, bởi vì SQL Server sử dụng một log ghi trước (write-ahead log). Kế tiếp, bản ghi giao tác đó được gửi đến mirroring server, ở đó nó được ghi vào log giao tác của mirroring server. Sau khi mirroring server đã viết bản ghi vào log của nó, nó gửi một thư hồi báo cho primary server báo rằng đã nhận được bản ghi, cho cả hai hệ thống cùng dữ liệu trong mỗi log file của chúng. Trong trường hợp của các hoạt động Commit, primary server chờ nhận một thư hồi báo từ mirroring server trước khi gửi sự phản hồi của nó trở lại client, cho nó biết hoạt động đã hoàn tất. Để giữ cho các file dữ liệu cập nhật trên mirroring server, nó cần phải ở một trạng thái phục hồi liên tục, lấy dữ liệu từ log và cập nhật file dữ liệu.

Database Mirroring được khởi chạy bằng cách lấy một bản sao dự phòng của cơ sở dữ liệu mà bạn muốn phản chiếu hay tạo đối xứng gương trên primary server và phục hồi bản sao dự phòng đó trở lại mirroring server. Nói cách khác, cơ sở dữ liệu được phản chiếu phải hiện hữu trên mirror server trước khi tiến trình phản chiếu (mirroring) có thể bắt đầu. Điều này sẽ đặt schema và dữ liệu và cơ sở dữ liệu nền tảng tại chỗ. Tiến trình sao lưu dự phòng (backup) và phục hồi (restore) có thể sử dụng bất kỳ loại phương tiện chuẩn của SQL Server, bao gồm cả băng từ hay đĩa. Kế tiếp, bạn sử dụng lệnh ALTER DATABASE như minh họa ở đây để bắt đầu tiến trình mirroring:

```
ALTER DATABASE <database name> SET PARTNER = '<partner server name>'
```


Lệnh ALTER DATABASE trước tiên phải được chạy trên mirroring server (server đang phản chiếu), hướng nó đến tên của primary SQL Server (SQL Server chính trong mệnh đề SET PARTNER. Sau đó lệnh ALTER DATABASE được chạy lần thứ hai, lần này trên primary server. Khi lệnh ALTER DATABASE được chạy trên primary server, tên của mirroring server được cung cấp trong mệnh đề SET PARTNER. Ngay khi lệnh ALTER DATABASE đã hoàn tất trên primary server, sự phản chiếu hay tạo đối xứng gương cơ sở dữ liệu (database mirroring) diễn ra và primary server sẽ bắt đầu chuyển các log đến mirroring server. Bước kế tiếp trong tiến trình cài đặt Database Mirroring là cài đặt vật chứng (witness). Gần giống như bạn cài đặt primary và mirroring server, bạn cài đặt witness bằng cách chạy lại lệnh ALTER DATABASE trên primary server, như minh họa trong danh sách dưới đây:

```
ALTER DATABASE <database name> SET WITNESS = '<witness server name>'
```

Trong trường hợp này, tên cơ sở dữ liệu sẽ là tên đã được sử dụng trong các lệnh trước. Tuy nhiên, lần này mệnh đề SET WITNESS được sử dụng để chỉ định witness server. Ngay khi witness server đã được cài đặt, Database Mirroring có mọi thông tin mà nó cần để thực hiện tiến trình failover cơ sở dữ liệu.

Về cơ bản, database mirroring cung cấp cho bạn một cơ sở dữ liệu ảo không bị lỗi. Tuy nhiên, đó chưa phải là toàn bộ vấn đề về Database Mirroring. Sự thực thì một cơ sở dữ liệu dự phòng khẩn (instant standby database) làm cho cơ sở dữ liệu đó nhanh chóng có sẵn sau một sự cố, nhưng các kết nối user với cơ sở dữ liệu bị hỏng đó thì sẽ như thế nào? Đó là lúc chúng ta cần đến tính năng mới Transparent Client Redirect.

Transparent Client Redirect

Tính năng Transparent Client Redirect phối hợp rất chặt chẽ với tính năng Database Mirroring mới nhằm cho phép các hệ thống client tự động được gửi lại mirroring server khi primary server trở nên không có sẵn. Tính năng mới này được thực thi trong SQL Server 2005 Microsoft Data Access Components (MDAC) mới và không bắt buộc thay đổi đối với các ứng dụng lớp client hay dữ liệu. Phần giữa MDAC nhận biết cả primary server và mirroring server. MDAC có được tên của mirroring server vào lần kết nối đầu tiên của nó với primary server. Khi kết nối với primary server bị mất, MDAC trước tiên sẽ cố kết nối lại với mirroring server. Nếu kết nối đầu tiên đó không thành công thì MDAC sẽ tự động gửi lại nỗ lực kết nối lần thứ hai của nó cho mirroring server. Cũng giống như với clustering, nếu kết nối bị mất ở giữa một giao tác thì giao tác đó sẽ được cuộn trở lại và phải được thực hiện lại sau khi client kết nối với mirroring server.

Khi nào cần sử dụng Clustering hay Database Mirroring

Cả clustering và database mirroring đều là các giải pháp chuẩn bị cho doanh nghiệp, chúng có khả năng cung cấp một hệ thống dự phòng nếu cơ sở dữ liệu hay server bị hỏng. Chúng có nhiều điểm tương tự. Cả hai đều có khả năng hoạt động không mất dữ liệu, phát hiện lỗi tự động, failover tự động và kết nối lại client. Tuy nhiên, có một số điểm khác biệt quan trọng. Database Mirroring hoạt động ở cấp cơ sở dữ liệu trong khi clustering (nhóm cụm) hoạt động ở cấp server. Database Mirroring có thời gian failover rất nhanh khoảng 2-3 giây trong khi clustering thường có thời gian failover khoảng 30 giây - đôi khi lâu hơn, tùy thuộc vào cấp độ hoạt động của cơ sở dữ liệu và kích cỡ của các cơ sở dữ liệu trên server đã bị hỏng. Database Mirroring cũng bảo vệ để chống các lỗi đĩa tốt hơn bởi vì không có sự lưu trữ đĩa chia sẻ, như có một trong giải pháp clustering. Hầu như không có sự hạn chế khoảng cách đối với Database Mirroring, trong khi clustering có giới hạn khoảng 100 dặm (mile), bởi vì có một hạn chế về thời gian để truyền nhịp đập giữa các nút cluster. Ngoài ra, Database Mirroring là một công nghệ đơn giản hơn và dễ thực thi hơn clustering. Mặt khác, Database Mirroring không được sử dụng cho các cơ sở dữ liệu hệ thống, trong khi clustering bảo vệ các cơ sở dữ liệu hệ thống cũng như các cơ sở dữ liệu người dùng. Về cơ bản, clustering là giải pháp tốt hơn để bảo vệ toàn bộ server, trong khi Database Mirroring là giải pháp tốt hơn để bảo vệ một ứng dụng hay cơ sở dữ liệu quan trọng.

Các cải tiến về tính khả dụng của cơ sở dữ liệu

Mặc dù các tình huống trên giải quyết các vấn đề quan trọng vốn làm giảm tính khả dụng của cơ sở dữ liệu, nhưng có lẽ không có vấn đề nào làm ảnh hưởng đến tính khả dụng và khả năng phục hồi cơ sở dữ liệu hơn bảo trì cơ sở dữ liệu. Bảo trì cơ sở dữ liệu là một công việc hằng ngày bạn không thể tránh. SQL Server 2005 giúp giải quyết những khó khăn trong lĩnh vực quan trọng này bằng việc thêm vào hai tính năng mới: Database Snapshots và sự truy cập phục hồi sớm (early restore access). Trong phần kế tiếp của chương này, chúng ta sẽ xem xét chi tiết hơn mỗi tính năng mới này.

Database Snapshot

Tính năng Database Snapshot cung cấp một snapshot chỉ đọc của một cơ sở dữ liệu tại một thời điểm nhất định. Một Database Snapshot thì phù hợp nhất cho việc tạo các bản sao của một cơ sở dữ liệu để báo cáo hay để tạo một bản sao dự phòng của cơ sở dữ liệu mà bạn sử dụng để phục hồi một cơ sở dữ liệu sản xuất trở lại trạng thái trước của nó. Khi một Database Snapshot được tạo, hệ thống tạo ra một bản sao

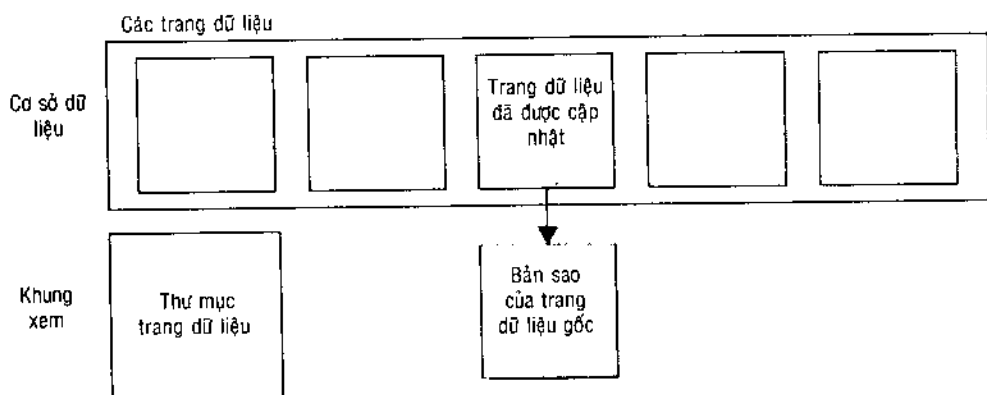
metadata của cơ sở dữ liệu đã chỉ định tại thời điểm cụ thể đó. Bất kỳ sự thay đổi kế tiếp đối với cơ sở dữ liệu gốc không được phản ánh trong Database Snapshot. Các ứng dụng kết nối với một Database Snapshot một cách chính xác như thể nó là một cơ sở dữ liệu khác. Một Database Snapshot có thể được tạo cho bất kỳ cơ sở dữ liệu. Các Database Snapshot được tạo như là phần của câu lệnh CREATE DATABASE DDL. Bạn có thể xem cách tạo một khung xem trên cơ sở dữ liệu mẫu AdventureWorks trong danh sách sau:

```
CREATE DATABASE AdWSnapShot_080104_0700 ON  
( NAME = AdventureWorks_Data,  
  FILENAME = 'C:\Program Files\Microsoft SQL Server  
  \MSSQL.1\MSSQL\DATA\AdventureWorks_data_040422_1800.ss')  
AS SNAPSHOT OF AdventureWorks;
```

Trong ví dụ này, bạn có thể thấy một Database Snapshot tên AdWSnapShot_080104_0700 được tạo trên cơ sở dữ liệu AdventureWorks. Các Database Snapshot được tạo nhờ dùng mệnh đề AS SNAPSHOT OF mà bạn có thể thấy trong danh sách ở trên. Khi bạn tạo một Database Snapshot, bạn phải chỉ định tất cả các file dữ liệu được sử dụng trong cơ sở dữ liệu nguồn. Vì cơ sở dữ liệu AdventureWorks có một file dữ liệu nên chỉ file AdventureWorks_Data là phải được chỉ định trong câu lệnh CREATE DATABASE. Bạn không chỉ định các log file.

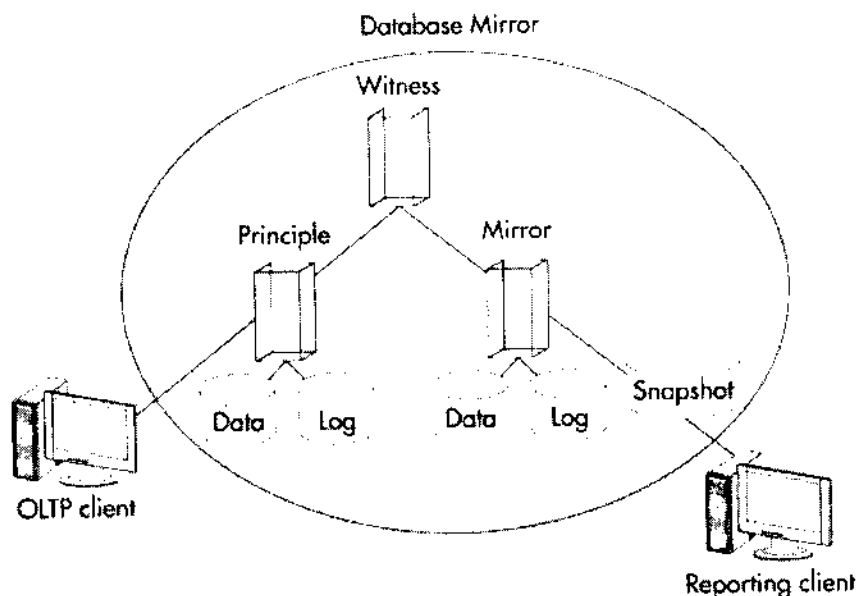
Tạo một Database Snapshot là một hoạt động không tốn kém vì về cơ bản server sử dụng metadata kết hợp với sự phục hồi (recovery) để tạo điểm xem (viewpoint). Bạn cần hiểu rằng các Database Snapshot không phải là một bản sao hoàn chỉnh của một cơ sở dữ liệu. Việc tạo một Snapshot là một hoạt động chỉ metadata (metadata-only). Một Database Snapshot sử dụng các trang dữ liệu giống như cơ sở dữ liệu gốc, vì vậy nó không đòi hỏi nhiều khoảng trống đĩa bổ sung. Các Database Snapshot được tạo nhờ dùng công nghệ copy-on-write, trong đó bất kỳ lúc nào có sự thay đổi đối với một trong các trang dữ liệu của cơ sở dữ liệu nguồn, một bản sao của trang đó sẽ được lưu cho Database Snapshot và sau đó trang được cập nhật sẽ được ghi như bình thường. Khi Database Snapshot được truy cập, nó sử dụng các trang dữ liệu chia sẻ cho đến khi nó đạt tới trang đã thay đổi, và sau đó nó sẽ xem các trang đã được sao chép thay vì các trang có chứa dữ liệu đã cập nhật. Theo cách này, Database Snapshot cần sự lưu trữ cho chỉ những trang đã được thay đổi kể từ lúc Database Snapshot đã được tạo. Hình 3.3 minh họa cách hoạt động của các Database Snapshot.

Các Database Snapshot



Hình 3.3: Các Database Snapshot

Các Database Snapshot có thể được kết hợp với Database Mirroring để tạo một server báo cáo (reporting server) dựa vào dữ liệu có trên mirrored server (hình 3.4). Bình thường, dữ liệu trên mirroring server (server được phản chiếu hay tạo đối xứng gương) luôn ở chế độ phục hồi, vì vậy, nó không thể được truy cập bởi một ứng dụng. Tuy nhiên, bạn có thể tạo một Database Snapshot mà nó dựa vào cơ sở dữ liệu được phản chiếu và Database Snapshot đó có thể được truy cập trong chế độ chỉ đọc (read-only) để báo cáo.



Hình 3.4: Database Mirroring và Database Snapshot tạo một server báo cáo (reporting server)

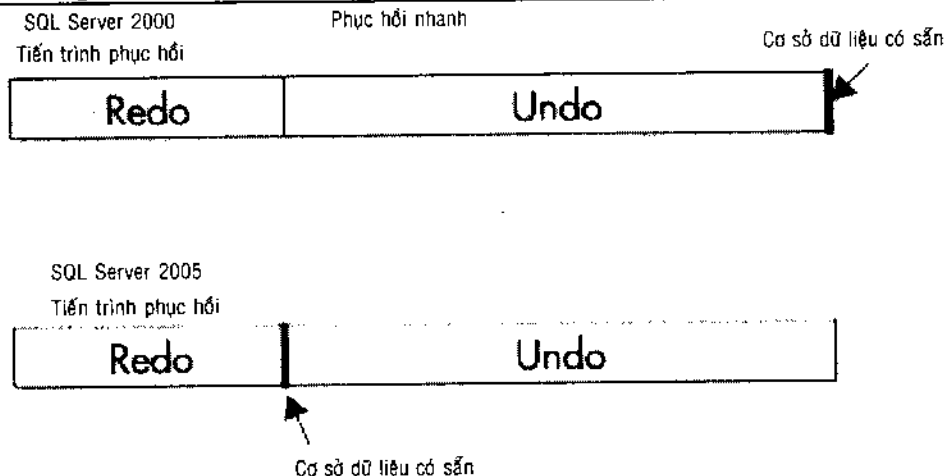
Mặc dù cơ sở dữ liệu trên mirroring server không thể được truy cập trực tiếp bởi vì nó đang ở trạng thái phục hồi, nhưng điều đó không làm ảnh hưởng đến Database Snapshot, vốn truy cập một snapshot của các trang dữ liệu trong cơ sở dữ liệu của mirroring server. Việc tạo một Database Snapshot trên mirroring server cho phép bạn tận dụng tốt hơn chức năng xử lý của mirroring server bằng cách chuyển báo cáo tĩnh (static reporting) sang server đó. Một điều cần xem xét khi sử dụng các Database Snapshot trên một Database Mirror là trong trường hợp failover, các Database Snapshot hiện có (đã được tạo trên mirroring server) vẫn còn nguyên vẹn.

Cần biết rằng các Database Snapshot không phải là một tính năng khả dụng mà là một tính năng failover. Chúng cung cấp nhiều cách hơn để truy cập dữ liệu của bạn, tăng tính khả dụng của dữ liệu. Chúng không phải là một tính năng failover - nếu cơ sở dữ liệu gốc không có sẵn thì Database Snapshot cũng sẽ không có sẵn.

Early Restore Access

Mỗi lần cơ sở dữ liệu SQL Server được khởi động hay một cơ sở dữ liệu được phục hồi, cơ sở dữ liệu đó phải trải qua giai đoạn phục hồi trong đó bất kỳ giao tác nằm bên trong log sẽ được áp dụng cho các file dữ liệu. Giai đoạn phục hồi thường được gọi là redo. Ngay sau khi phần redo của tiến trình phục hồi đã hoàn chỉnh thì mọi hoạt động undo được thực hiện vì bất kỳ giao tác chưa hoàn chỉnh trong log sẽ được cuộn trở lại. Với SQL Server 2000, cơ sở dữ liệu không có sẵn cho đến khi nào đã hoàn tất mọi hoạt động redo và undo. Nếu một cơ sở dữ liệu được đưa xuống trong khi nó hoạt động và cơ sở dữ liệu đó khá bận thì có thể có một khoảng thời gian trễ khá lâu trước khi cơ sở dữ liệu đó lại có sẵn. Vì bạn cần chờ đợi cho đến khi xử lý xong mọi mục nhập trong log.

Early Restore Access mới của SQL Server 2005 cho phép các cơ sở dữ liệu có sẵn sau khi phần redo của tiến trình phục hồi đã hoàn tất. Với SQL Server 2005, khi cơ sở dữ liệu được khởi động lại, mọi giao tác mở trong log sẽ được redo và sau đó cơ sở dữ liệu lập tức có sẵn. Kết quả là cơ sở dữ liệu có sẵn sớm hơn nhiều. Với các giao tác chưa được chuyển đi, SQL Server vẫn giữ các khóa trên các trang dữ liệu do các giao tác đó sử dụng để các giao tác vẫn nhất quán cho dù cơ sở dữ liệu đang được sử dụng. Sau đó SQL Server 2005 bắt đầu tiến trình undo các giao tác này trong khi cơ sở dữ liệu đang hoạt động. Người dùng có thể bắt đầu các hoạt động đọc/ghi vào cơ sở dữ liệu trong suốt giai đoạn undo của tiến trình phục hồi. Tuy nhiên, mọi nỗ lực truy cập dữ liệu mà SQL Server đã khóa trong suốt tiến trình undo sẽ bị chặn lại cho đến khi tiến trình undo mở các khóa trên dữ liệu đó. Hình 3.5 minh họa sự khác nhau về tính khả dụng giữa SQL Server 2000 và SQL Server 2005.



Hình 3.5: Sự truy cập phục hồi sớm (early restore access)

Hành động RESTORE bây giờ xác định phạm vi của tùy chọn redo. Bạn có thể phục hồi toàn bộ, một phần, hay nhóm file của các bản sao dự phòng. Nếu bạn chỉ định bạn muốn phục hồi một nhóm file thì chỉ dữ liệu của nhóm file đó là được thêm vào hoạt động cuộn tới. Nếu bạn chỉ định phục hồi toàn bộ bản sao dự phòng thì mọi dữ liệu trong bản sao dự phòng sẽ được sử dụng cho hoạt động redo.

Online Index Operations

Với các phiên bản trước của SQL Server, khi một index (chỉ mục) đang được tạo lại, bạn không thể thực hiện bất kỳ hoạt động cập nhật trên table cho đến khi kết thúc tiến trình tạo lại index. Tính năng Online Index Operations mới của SQL Server 2005 mở rộng tính khả dụng của SQL Server bằng việc cho phép các ứng dụng cập nhật, chèn và xóa các hàng từ table trong khi index đang được tạo lại. Tính năng online index mới thực hiện phần việc này bằng cách lưu giữ hai bản sao của index: một để các ứng dụng tiếp tục sử dụng và một index tạm thời thứ hai được sử dụng trong khi index đang được tạo lại. Bộ máy SQL Server (SQL Server engine) duy trì cả hai index này với bất kỳ sự thay đổi khi tiến trình tạo lại đang được thực hiện. Sau khi tiến trình này kết thúc, index cũ bị loại bỏ và được thay bằng index mới. Online index rebuild (tạo lại chỉ mục trực tuyến) được hỗ trợ cho các câu lệnh CREATE INDEX, ALTER INDEX REBUILD, ALTER INDEX DISABLE, DROP INDEX, ALTER TABLE ADD CONSTRAINT, và ALTER TABLE DROP CONSTRAINT. Mặc dù SQL Server 2005 bây giờ có hỗ trợ online index rebuild, nhưng nó đòi hỏi phải tăng thêm chi phí, vì bạn vẫn có thể chọn tạo lại các index ngoại tuyến.

Danh sách sau đây minh họa cách sử dụng tính năng tạo index trực tuyến:

```
CREATE INDEX MyIndex ON Person.Contact (LastName) WITH (ONLINE=ON)
```

Ở đây, bạn có thể thấy câu lệnh CREATE INDEX chuẩn để tạo một index có tên là MyIndex trên cột LastName của table PersonContact vốn nằm trong cơ sở dữ liệu AdventureWorks. Mệnh đề ONLINE=ON cho phép index được tạo trực tuyến (online).

Theo dõi trang bị hỏng

Một tính năng mới có liên quan chặt chẽ với khả năng thực hiện các tiến trình phục hồi cấp trang của SQL Server 2005 là theo dõi các trang bị hỏng. Bất kỳ trang bị hỏng sẽ được theo dõi trong một table, và bằng cách sử dụng tính năng fine-grained restore, bạn có thể phục hồi trên cơ sở từng trang trong khi cơ sở dữ liệu vẫn online. Bất kỳ giao tác nào sử dụng dữ liệu từ một trang bị hỏng sẽ được cuộn trở lại. Nếu trang bị hỏng đó vô tình hiển thị trong lúc cuộn lại một giao tác thì cơ sở dữ liệu buộc phải khởi động lại.

Nối kết quản trị chuyên dụng

Một tính năng mới trong SQL Server 2005 giúp cung cấp tính khả dụng tốt hơn là sự nối kết của quản trị viên chuyên dụng mới. Sự nối kết của quản trị viên chuyên dụng cung cấp cho DBA sự truy cập vào server bất kể tải trọng hiện tại của server. Điều này cho phép DBA truy cập vào server và hủy bất kỳ tiến trình còn sót lại. Để bắt đầu công cụ SQLCMD mới ở chế độ quản trị chuyên dụng, bạn sẽ nhập lệnh sau đây:

```
C:\Sqlcmd -A
```

Bộ nhớ Hot-Plug

Một tính năng khả dụng mới khác bắt nguồn từ Windows Server 2003 là bộ nhớ Hot-Plug. Bộ nhớ Hot-Plug cho phép bạn thêm RAM trong khi hệ thống đang chạy. Như bạn mong đợi, tính năng này đòi hỏi sự hỗ trợ từ OS cơ sở và nền phần cứng. Trong trường hợp của SQL Server 2005, điều này có nghĩa là SQL Server phải chạy trên Windows Server 2003 để có thể tận dụng tính năng khả dụng này. Nếu nền hỗ trợ tính năng này thì bạn có thể thêm bộ nhớ vào lúc hệ thống đang chạy và SQL Server 2005 sẽ có thể nhận biết động lượng RAM bổ sung này không cần phải khởi động lại server hay tắt máy. Tuy bộ nhớ Hot-Plug cho phép bạn thêm RAM vào lúc hệ thống đang chạy nhưng nó không cho phép bạn loại bỏ nó.

Cải tiến cấu hình động

Mọi cấu hình trong SQL Server 2005 bây giờ đều động, bao gồm cả CPU và hệ thống I/O. Với SQL Server 2005, bây giờ bạn có thể thay đổi các giá trị này và hiệu ứng sẽ xảy ra ngay tức thời. Không cần phải khởi động lại server nữa. Ngoài ra, các thay đổi cho Address Windowing Extentions (AWE) cũng động. SQL Server 2005 có thể tự động điều chỉnh các thay đổi sang cơ AWE vật lý. Tính năng này được thiết kế để hoạt động kết hợp với khả năng sử dụng bộ nhớ Hot-Plug vào lúc hệ thống đang chạy. Nó đòi hỏi phải có Windows Server 2003.

Lưu dự phòng và phục hồi

SQL Server 2005 cũng có nhiều tính năng mới trong lĩnh vực của công nghệ lưu dự phòng và phục hồi. Tuy nhiên tính năng khả dụng trước đây có liên quan đến một số tính năng lưu dự phòng và phục hồi mới này, nhưng mục này tập trung chủ yếu vào những thay đổi mà Microsoft đã thực hiện để làm cho tiến trình lưu dự phòng và phục hồi trở nên mạnh hơn và dễ thực hiện hơn.

Phục hồi từng phần

Với SQL Server 2000, cơ sở dữ liệu không có sẵn trong lúc phục hồi. Với SQL Server 2005, cơ sở dữ liệu online (trực tuyến) ngay khi nhóm file chính được phục hồi. Chỉ dữ liệu đang được phục hồi là không có sẵn. Nếu một user truy cập dữ liệu được tìm thấy trong nhóm file chính, hoạt động này sẽ thành công mà không có dấu hiệu nào báo cho user biết rằng phần còn lại của cơ sở dữ liệu vẫn đang được phục hồi. Nếu user cố truy cập dữ liệu từ một nhóm file đang được phục hồi hoặc vẫn offline (ngoại tuyến) thì cơ sở dữ liệu sẽ trả về một thông báo cho biết rằng dữ liệu đang offline.

Các cải tiến về độ tin cậy của phương tiện

SQL Server 2005 cũng cung cấp nhiều cải tiến về cách nó xử lý phương tiện. Cụ thể, nó cho phép thực hiện các tiến trình lưu dự phòng trên các thiết bị được phản chiếu, nâng cao tính toàn vẹn checksum, và cho phép tiến trình phục hồi tiếp tục diễn ra ngay cả khi bắt gặp các lỗi.

Media Mirroring

Một trong những tính năng mới bảo đảm độ tin cậy của phương tiện trong SQL Server 2005 là khả năng hỗ trợ media mirroring (phản chiếu hay tạo đối xứng gương phương tiện). Media mirroring cho phép bạn đồng thời thực hiện một tiến trình lưu dự phòng vào nhiều thiết bị.

Chẳng hạn, bạn có thể lưu dự phòng cơ sở dữ liệu của bạn vào cả hai băng 1 (tape1) và băng 2 (tape2) cùng lúc, có hai bản sao giống nhau của bộ bản sao. Phương tiện dự phòng thừa sẽ đem lại một sự bảo vệ quan trọng nhằm giúp công ty của bạn tránh các lỗi phương tiện đồng thời bảo đảm khả năng thực hiện tiến trình phục hồi thành công. Ví dụ sau đây mặc định cách sử dụng tính năng media mirroring mới:

```
BACKUP DATABASE AdventureWorks TO
```

```
TAPE='\\.\tape1'
```

```
MIRROR TO
```

```
TAPE='\\.\tape2'
```

```
WITH FORMAT, MEDIANAME = 'ADWBackup'
```

Ví dụ này cho thấy cơ sở dữ liệu AdventureWorks đang được lưu dự phòng vào tape1 và được phản chiếu sang tape2. Sự thực thi của tiến trình lưu dự phòng này không bị ảnh hưởng bởi việc thêm vào một mirror dự phòng. Bạn có thể dùng đến bốn mirror cho một bản sao dự phòng đơn.

Cải thiện việc xác nhận các bản sao dự phòng

Với SQL Server 200, việc sử dụng lệnh RESTORE VERIFYONLY làm cho SQL Server đọc băng mà không cần phải đi xa hơn vào tiến trình phục hồi. Việc sử dụng lệnh RESTORE VERIFYONLY trong SQL Server 2005 thực hiện mọi thứ ngoại trừ thật sự phục hồi dữ liệu trong phương tiện phục hồi, cho bạn một ý tưởng tốt hơn nhiều về mức độ quan trọng của dữ liệu trong bộ dự phòng.

Tiếp tục với các lỗi phục hồi trước đây

Một cải tiến khác về độ tin cậy của phương tiện là tiến trình phục hồi có thể tiếp tục diễn ra với các lỗi phương tiện trước đây. Các phiên bản trước của SQL Server sẽ hủy một tiến trình phục hồi nếu bắt gặp bất kỳ lỗi nào trong lúc phục hồi. Tuy nhiên, trong những trường hợp mà đây là phiên bản phương tiện duy nhất có sẵn thì nó có thể gây trở ngại rất lớn đến việc phục hồi bất kỳ dữ liệu nào có sẵn. Với tính năng mới này, SQL Server 2005 cho phép tiến trình phục hồi tiếp tục diễn ra càng lâu càng tốt, bỏ qua các lỗi phương tiện mà nó bắt gặp. Sau khi tiến trình phục hồi đã hoàn tất, cơ sở dữ liệu có thể được sửa bằng tay.

Database Page Checksum

Tính năng Database Page Checksum mới cho phép cơ sở dữ liệu phát hiện các lỗi đĩa và I/O vốn không được báo cáo bởi hệ thống phụ của đĩa.

Khi tính năng Database Page Checksum được kích hoạt, SQL Server sẽ tính một trị checksum khi trang được ghi vào đĩa và ghi checksum đó cùng với dữ liệu. Khi trang được đọc, một checksum khác được tính và được so sánh với checksum ban đầu đã ghi cùng với dữ liệu. Nếu chúng khác nhau thì một lỗi được báo cáo. Lệnh sau đây cho thấy cách Database Page có thể được thêm vào một cơ sở dữ liệu hiện có:

```
ALTER DATABASE AdventureWorks
```

```
SET PAGE_VERIFY CHECKSUM
```

Lưu dự phòng Log và cơ sở dữ liệu cùng lúc

Một tính năng mới khác trong lĩnh vực lưu dự phòng SQL Server 2005 là khả năng thực hiện cùng lúc cả hai tiến trình lưu dự phòng log và cơ sở dữ liệu. Với SQL Server 2000, bạn phải chờ đợi lưu dự phòng log cho đến sau khi tiến trình lưu dự phòng cơ sở dữ liệu đã hoàn tất. Với SQL Server 2005, các tiến trình lưu dự phòng cơ sở dữ liệu không còn cản trở các tiến trình lưu dự phòng log nữa. Bạn cũng có thể lưu dự phòng các file và các nhóm file cùng lúc bạn lưu dự phòng log giao tác. Tuy nhiên, bạn nên giới hạn việc lưu dự phòng mỗi lần một file dữ liệu trên mỗi cơ sở dữ liệu.

Lưu dự phòng Full-Text Catalog

Một cải tiến khác trong SQL Server 2005 là khả năng lưu dự phòng Full Text Catalog. Với SQL Server 2000, dữ liệu Full Text Catalog được duy trì bên ngoài SQL Server bởi Microsoft Search Service. Việc lưu dự phòng cơ sở dữ liệu SQL Server chứa toàn dữ liệu text không tự động lưu dự phòng full-text catalog. Điều này dẫn đến khả năng catalog (danh mục) không đồng bộ với dữ liệu full-text đang được phục hồi. SQL Server 2005 xử lý vấn đề này bằng cách cho phép lưu dự phòng tự động mọi full-text catalog ngoài cùng lúc cơ sở dữ liệu được lưu dự phòng. Điều này bảo đảm full-text catalog và dữ liệu vẫn nhất quán giữa các tiến trình lưu dự phòng và phục hồi. Các lệnh DATABASE ATTACH và DEATTACH cũng có một tùy chọn để đưa vào bất kỳ full-text catalog. Các liên kết trong cơ sở dữ liệu chỉ đến các file ngoài được sử dụng. Khi cơ sở dữ liệu được lưu dự phòng, các file ngoài này cũng được lưu dự phòng, bảo đảm sự nhất quán cho cơ sở dữ liệu.

PHẦN II

CÁC TÍNH NĂNG PHÁT TRIỂN CƠ SỞ DỮ LIỆU

CHƯƠNG 4

Các tính năng lập trình

Các tính năng phát triển mới trong SQL Server 2005 là sự tích lũy nỗ lực trong nhiều năm của cả đội ngũ phát triển SQL Server và đội ngũ phát triển .NET Framework. Ý nghĩa nhất trong số các tính năng phát triển mới này là sự hợp nhất của .NET Common Language Runtime (CLR). Sự hợp nhất của CLR đã mang theo một loại các khả năng mới, kể cả khả năng tạo các đối tượng cơ sở dữ liệu bằng cách dùng bất kỳ ngôn ngữ tương thích .NET, bao gồm C#, VB.NET, và Managed C++. Trong chương này, bạn sẽ được giới thiệu về các tính năng hợp nhất .NET CLR mới đó cũng như xem một số ví dụ minh họa cách sử dụng chúng. Kế tiếp, chương này sẽ trình bày một chủ điểm quen thuộc hơn với các nhà phát triển và SQL Server DBA: các tính năng mới trong T-SQL. Sau đó, chương sẽ đưa bạn qua phía client và trình bày một số tính năng phát triển mới trong .NET Framework Data Provider for SQL Server đã cập nhật vốn đi kèm với SQL Server 2005.

Sự hợp nhất Common Language Run-time (CLR)

Chắc chắn tính năng mới có ý nghĩa nhất trong ấn bản SQL Server 2005 là sự hợp nhất của Microsoft .NET Framework. Sự hợp nhất của CLR với SQL Server đã mở rộng khả năng của SQL Server về nhiều phương diện quan trọng. Trong khi T-SQL, ngôn ngữ truy cập và xử lý dữ liệu hiện tại, thì thích hợp cho các hoạt động truy cập dữ liệu hướng tập hợp, nhưng nó cũng có hạn chế. Được thiết kế cách đây hơn một thập niên, T-SQL là một ngôn ngữ theo thủ tục chứ không phải ngôn ngữ hướng đối tượng. Sự hợp nhất của CLR với SQL Server 2005 đã mang theo cùng với nó khả năng tạo các đối tượng cơ sở dữ liệu nhờ dùng các ngôn ngữ hướng đối tượng hiện đại như VB.NET và C#, trong

khi các ngôn ngữ này không có bản chất hướng tập hợp mạnh như T-SQL, chúng hỗ trợ logic phức tạp, có khả năng tính toán tốt hơn, cung cấp sự truy cập dễ dàng hơn vào các tài nguyên bên ngoài, giúp sử dụng lại mã dễ dàng, và có một môi trường phát triển hạng nhất với chức năng mạnh hơn rất nhiều so với Query Analyzer cũ.

Sự hợp nhất của .NET CLR với SQL Server 2005 cho phép phát triển các thủ tục lưu trữ, các hàm do người dùng ấn định, các bộ kích khởi, các hàm gộp, và các kiểu dữ liệu bằng cách sử dụng một trong các ngôn ngữ .NET. Sự hợp nhất của .NET CLR với SQL Server 2005 không chỉ ở bề ngoài. Thật vậy, bộ máy cơ sở dữ liệu SQL Server 2005 làm chủ CLR cả trong tiến trình. Bằng cách sử dụng một tập hợp API, bộ máy SQL Server thực hiện mọi việc quản lý bộ nhớ cho các chương trình CLR được triển khai.

Mã đã quản lý truy cập cơ sở dữ liệu nhờ dùng ADO.NET kết hợp với SQL Server .NET Data Provider mới. Một đối tượng SQL Server mới được gọi là assembly (hợp ngữ), là đơn vị triển khai cho các đối tượng .NET với cơ sở dữ liệu. Để tạo các đối tượng cơ sở dữ liệu CLR, trước tiên bạn phải tạo một DLL nhờ dùng Visual Studio 2005. Sau đó, bạn nhập DLL đó vào SQL Server ở dạng một hợp ngữ. Sau cùng, bạn liên kết hợp ngữ đó với một đối tượng cơ sở dữ liệu chẳng hạn như thủ tục lưu trữ hay bộ kích khởi. Trong mục kế tiếp, bạn sẽ xem xét chi tiết hơn về cách bạn thật sự sử dụng các tính năng CLR mới trong SQL Server 2005.

Các hợp ngữ

Để tạo các đối tượng cơ sở dữ liệu .NET, bạn phải viết mã đã quản lý và biên dịch nó thành một hợp ngữ .NET. Cách phổ biến nhất để làm điều này là sử dụng Visual Studio 2005, sau đó tạo một dự án SQL Server mới và biên dịch nó thành một DLL. Các chi tiết về cách bạn thật sự làm để tạo một dự án mã đã quản lý với Visual Studio 2005 được trình bày trong mục "Các thủ tục lưu trữ .NET" ở sau của chương này.

Sau khi hợp ngữ đã được tạo, bạn có thể tải hợp ngữ đó vào SQL Server nhờ dùng lệnh T-SQL CREATE ASSEMBLY, như bạn có thể thấy ở đây:

```
CREATE ASSEMBLY MyCLRDLL.
```

```
FROM '\\SERVERNAME\CodeLibrary\MyCLRDLL.dll'
```

Lệnh CREATE ASSEMBLY lấy một thông số có chứa đường dẫn đến DLL vốn sẽ được tải vào SQL Server. Đây có thể là một đường dẫn cục bộ, nhưng thông thường hơn nó là một đường dẫn đến một phần chia sẻ

file mạng. Khi CREATE ASSEMBLY được thực thi, DLL được sao chép vào cơ sở dữ liệu chính.

Nếu một hợp ngữ được cập nhật hay trở nên không phù hợp, bạn có thể loại bỏ hợp ngữ đó nhờ dùng lệnh DROP ASSEMBLY như sau:

```
DROP ASSEMBLY MyCLRDL
```

Bởi vì các hợp ngữ tồn tại trong cơ sở dữ liệu nên khi mã nguồn dành cho hợp ngữ đó được chỉnh sửa và hợp ngữ được biên dịch lại, hợp ngữ đó trước tiên phải bị loại ra khỏi cơ sở dữ liệu nhờ dùng lệnh DROP ASSEMBLY và sau đó được tải lại nhờ dùng lệnh CREATE ASSEMBLY trước khi các phần cập nhật được phản ánh trong các đối tượng cơ sở dữ liệu SQL Server.

Bạn có thể sử dụng khung xem sys.assemblies để xem các hợp ngữ đã được thêm vào SQL Server 2005, như minh họa ở đây:

```
SELECT * FROM sys.assemblies
```

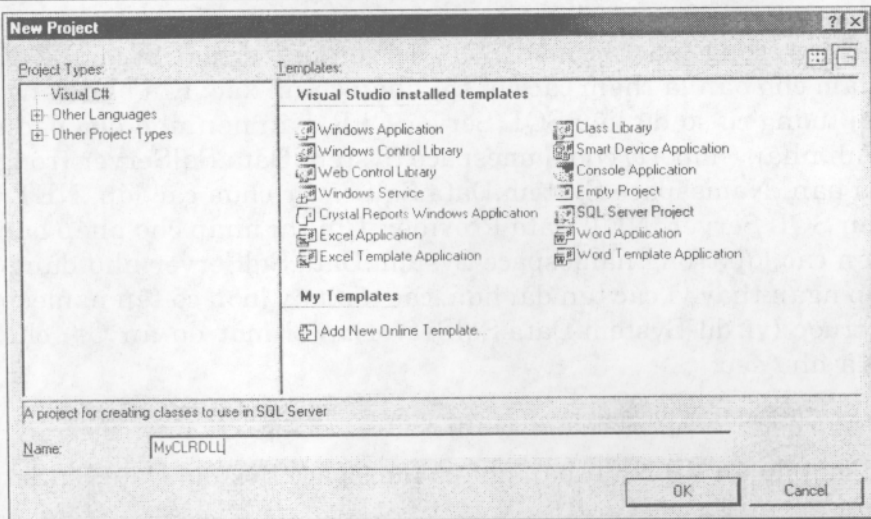
Vì các hợp ngữ được tạo nhờ dùng các file ngoài, nên bạn cũng muốn xem các file đã được sử dụng để tạo các hợp ngữ đó. Bạn có thể làm điều đó nhờ dùng khung xem sys.assembly_files, như minh họa ở đây:

```
SELECT * FROM sys.assembly_files
```

SQL Server .NET Data Provider

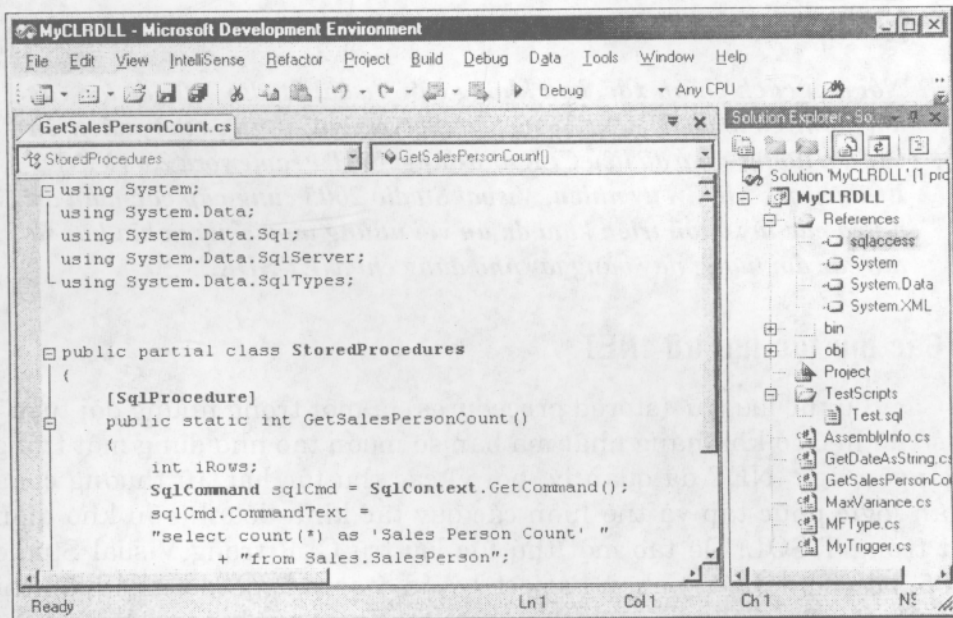
Nếu bạn quen thuộc với ADO.NET, bạn có thể thắc mắc các thủ tục cơ sở dữ liệu CLR mới này sẽ nối kết như thế nào với cơ sở dữ liệu. ADO.NET thực hiện nối kết cơ sở dữ liệu của nó nhờ dùng .NET data provider dựa trên client giống như .NET Framework Data Provider dành cho SQL Server, vốn nối kết với cơ sở dữ liệu SQL Server, hoặc .NET Framework Data Provider dành cho Oracle, vốn nối kết các ứng dụng ADO.NET với các cơ sở dữ liệu Oracle. Tuy điều đó tuyệt vời cho một ứng dụng đã nối mạng, nhưng việc đi qua các thư viện mạng không phải là chế độ nối kết hiệu quả nhất cho mã đang chạy trực tiếp trên server. Để giải quyết vấn đề này, Microsoft đã tạo ra SQL Server .NET Data Provider mới. SQL Server .NET Data Provider thiết lập một nối kết trong bộ nhớ với cơ sở dữ liệu SQL Server.

Một phần tham chiếu đến SQL Server .NET Data Provider mới được thêm tự động vào các ứng dụng Visual Studio 2005 của bạn khi bạn tạo một dự án SQL Server mới. Để tạo một loại dự án SQL Server mới, trước tiên bạn mở Visual Studio 2005 và sau đó chọn tùy chọn File | New Project từ menu. Sau đó, từ hộp thoại New Projects, chọn loại dự án (ví dụ Visual Studio Projects, Visual C# Projects), rồi cuộn qua danh sách các mẫu (template) cho đến khi bạn nhìn thấy mẫu dự án SQL Server như minh họa ở hình 4.1.



Hình 4.1: Tạo một dự án SQL Server mới với Visual Studio 2005

Sau khi bạn chọn mẫu SQL Server Project, hãy đặt tên cho dự án của bạn và nhấn OK để tạo dự án (project). Mọi tham chiếu bắt buộc sẽ tự động được thêm vào dự án SQL Server của bạn. SQL Server .NET Data Provider mới được thêm ở dạng tham chiếu sqlaccess mà bạn có thể thấy nó được bật sáng ở hình 4.2. Ngoài ra, bạn còn thấy phần tham chiếu System.Data, vốn cung cấp sự hỗ trợ cho ADO.NET và các đối tượng hướng dữ liệu của nó chẳng hạn như DataSet và các kiểu dữ liệu SQL Server.



Hình 4.2: Tham chiếu SQL Server .NET Data Provider

Ngoài việc thêm các phần tham chiếu thích hợp, một trong những điểm quan trọng mà các mẫu SQL Server của Visual Studio 2005 tự động làm cho bạn là thêm các chỉ thị nhập chính xác. Khi bạn đang tạo các đối tượng cơ sở dữ liệu SQL Server CLR, bạn nên nhớ đưa vào một câu lệnh nhập (import) cho namespace System.Data.SqlServer trong dự án của bạn. Namespace System.Data.SqlServer chứa các lớp .NET vốn tạo nên SQL Server .NET Data Provider. Chỉ thị nhập cho phép bạn đề cập đến các lớp trong namespace System.Data.SqlServer nhờ dùng các tên lớp ngắn thay vì các tên dài hơn, các tên này luôn có tên namespace đứng trước (ví dụ System.Data.SqlServer). Với một dự án C#, chỉ thị nhập là như sau:

```
Using System.Data.SqlServer;
```

Với một dự án VB.NET, bạn nhập namespace System.Data.SqlServer như sau:

```
Imports System.Data.SqlServer
```

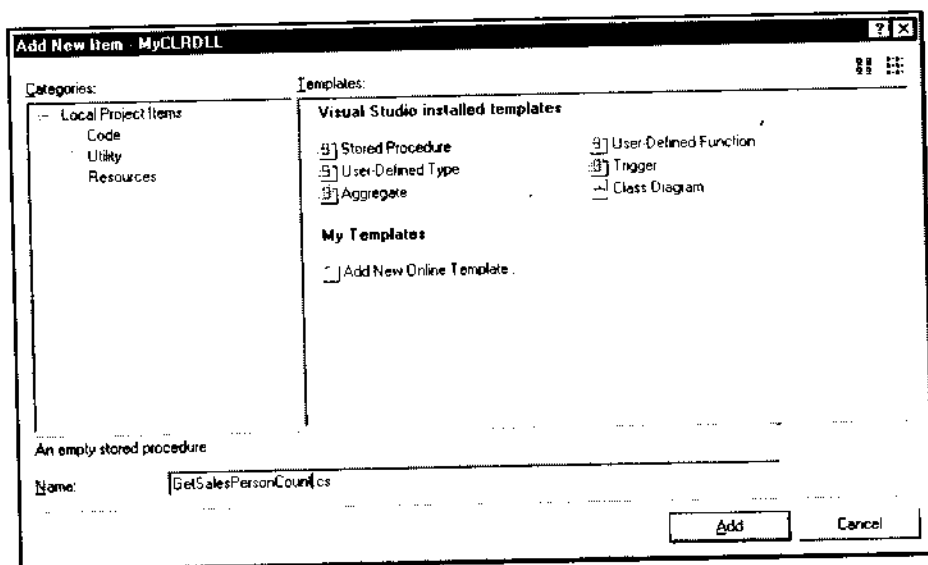
Không giống như các dự án ADO.NET tiêu biểu, trong đó bạn phải mở một nối kết theo cách tường minh đến một instance SQL Server đã đặt tên nhờ dùng phương thức Open của đối tượng nối kết, SQL Server .NET Data Provider thực hiện một nối kết ngầm với hệ thống SQL Server cục bộ, và như bạn sẽ thấy trong các ví dụ dưới đây, không cần tạo một đối tượng nối kết ADO.NET và viện dẫn phương thức Open của nó.



Nói một cách chính xác, bạn không cần Visual Studio 2005 để tạo các đối tượng cơ sở dữ liệu .NET cho SQL Server 2005. Bạn có thể phát triển các đối tượng cơ sở dữ liệu CLR nhờ dùng .NET Framework 2.0 và .NET Framework SDK. Tuy nhiên, Visual Studio 2005 cung cấp các mẫu dự án và các tùy chọn triển khai dự án với những ưu điểm ý nghĩa hơn là tạo các đối tượng này bằng tay nhờ dùng chỉ .NET SDK.

Các thủ tục lưu trữ .NET

Các thủ tục lưu trữ (stored procedures) là một trong những đối tượng cơ sở dữ liệu có khả năng nhất mà bạn sẽ muốn tạo nhờ dùng một trong các ngôn ngữ .NET đã quản lý, bởi vì các thủ tục lưu trữ thường chứa phần logic phức tạp và thể hiện các quy tắc kinh doanh vốn khó diễn đạt trong T-SQL. Để tạo một thủ tục lưu trữ CLR trong Visual Studio 2005, bạn sử dụng tùy chọn Project | Add Stored Procedure để hiển thị hộp thoại Visual Studio installed templates như minh họa ở hình 4.3.



Hình 4.3: Thêm một thủ tục lưu trữ CLR

Từ hộp thoại Add New Item, chọn tùy chọn Stored Procedure từ danh sách các mẫu được hiển thị trong danh sách Templates và sau đó cung cấp tên của thủ tục lưu trữ trong trường Name mà bạn thấy ở cuối màn hình. Ở đây, bạn có thể thấy thủ tục lưu trữ được đặt tên là GetSalesPersonCount. Visual Studio 2005 sẽ thêm một lớp mới vào dự án của bạn cho thủ tục lưu trữ. File lớp được tạo sẽ được đặt tên theo tên của thủ tục lưu trữ và sẽ chứa tất cả các chỉ thị nhập yêu cầu cũng như mã đặt tên thủ tục lưu trữ. Bạn có thể tùy ý điền vào phần mã còn lại để làm cho thủ tục lưu trữ hoạt động. Ví dụ sau đây minh họa mã nguồn được yêu cầu để tạo một thủ tục lưu trữ CLR đơn giản:

```
using System;  
  
using System.Data;  
using System.Data.Sql;  
using System.Data.SqlClient;  
using System.Data.SqlTypes;
```

```
public partial class StoredProcedures  
{  
    [SqlProcedure]  
    public static int GetSalesPersonCount()  
    {
```



```

int iRows;
SqlCommand sqlCommand = SqlContext.GetCommand();
sqlCmd.CommandText =
    "SELECT COUNT(*) AS 'Sales Person Count' "
    + "FROM Sales.SalesPerson";
iRows = (int)sqlCmd.ExecuteScalar();
return iRows;
}
};

```

Điểm quan trọng đầu tiên cần lưu ý trong mã này là chỉ thị nhập namespace `System.Data.SqlServer`. Chỉ thị này cho phép dự án `MyCLRDDL` sử dụng `SQL Server .NET Data Provider` mà không phải lúc nào cũng cần tham chiếu tên đầy đủ. Điểm quan trọng thứ hai cần lưu ý là thuộc tính `[SqlProcedure]` đứng trước tên phương thức; nó cho trình biên dịch biết rằng phương thức sẽ được trình bày như là một thủ tục lưu trữ `SQL Server`. Kế tiếp, bạn có thể nhận thấy tên lớp mặc định cho thủ tục lưu trữ này được xác lập sang `StoredProcedures`. Lớp này chứa một phương thức tĩnh `GetSalesPersonCount` vốn cho ra một kiểu dữ liệu `in`. Đối với `C#`, phương thức phải được định nghĩa là tĩnh (`static`). Với mã `VB.NET`, phương thức cần được định nghĩa là `Shared`. Mã ở đây về cơ bản truy xuất số hàng từ table `Sales.SalesPerson` trong cơ sở dữ liệu `adventureWorks` mẫu. Chú ý rằng bên trong lớp phương thức `Open` không được sử dụng. Đây là một sự khởi đầu cho mã `ADO.NET` dựa trên nền client. `SQL Server .NET Data Provider` tự động mở một nối kết với server cục bộ thay cho bạn.

Sau khi mã nguồn thủ tục lưu trữ `CLR` đã được biên dịch thành một hợp ngữ, bạn có thể thêm hợp ngữ đó vào cơ sở dữ liệu và tạo thủ tục lưu trữ `CLR`. Bạn có thể làm điều này bằng hai cách: Nếu bạn đang phát triển trong `Visual Studio 2005` thì bạn có thể chỉ cần sử dụng tùy chọn `Build | Deploy Solution` để cài đặt thủ tục lưu trữ `CLR` mới trong cơ sở dữ liệu `SQL Server`. Hoặc, bạn có thể thực hiện các bước triển khai bằng tay. Để giúp bạn hiểu rõ cách các đối tượng `CLR` trong cơ sở dữ liệu, mục kế tiếp sẽ trình bày các bước triển khai này.

Sau khi tạo `DLL`, bước kế tiếp là sử dụng `DLL` đó để tạo một đối tượng `SQL Server` mới được gọi là hợp ngữ (`assembly`). Mã sau đây minh họa việc tạo một hợp ngữ cho `MyCLRDDL.DLL`:

```

CREATE ASSEMBLY MyCLRDDL
FROM "\\MyFileShared\\Code Library\\MyCLRDDL.dll"

```

Lệnh CREATE ASSEMBLY sử dụng đối số đầu tiên để đặt tên hợp ngữ. Ở đây, nó đã được đặt tên là MyCLRDLL, giống như tên của .NET DLL thật sự, nhưng không bắt buộc phải sử dụng các tên giống nhau. Đối số sau mệnh đề FROM cho câu lệnh CREATE ASSEMBLY biết nơi để tìm DLL vật lý trên đĩa. Nó có thể nằm trên ổ đĩa cục bộ, hoặc nó có thể là một đường dẫn UNC.

Lưu ý

Vào lúc viết bài này, lần đầu tiên bạn chạy lệnh CREATE ASSEMBLY bằng tay, Microsoft.VisualStudio.DataTools.SqlAttributes.dll cũng cần có trong thư mục chứa .NET DLL mà bạn muốn thêm vào ở dạng một hợp ngữ.

Khi hợp ngữ đã được tạo, DLL được sao chép vào cơ sở dữ liệu SQL Server đích và hợp ngữ được đăng ký. Mã sau đây minh họa việc tạo thủ tục lưu trữ GetSalesPersonCount vốn sử dụng hợp ngữ MyCLRDLL:

```
CREATE PROCEDURE GetSalesPersonCount
AS EXTERNAL NAME
MyCLRDLL.StoredProcedures.GetSalesPersonCount
```

Mệnh đề EXTERNAL NAME là mới với SQL Server 2005. Ở đây, mệnh đề EXTERNAL NAME chỉ định thủ tục lưu trữ GetSalesPersonCount sẽ được tạo nhờ dùng một hợp ngữ .NET. Một hợp ngữ có thể chứa nhiều lớp và phương thức; câu lệnh EXTERNAL NAME sử dụng cú pháp sau đây để nhận biết đúng lớp và phương thức cần dùng từ hợp ngữ:

```
Assembly Name.ClassName.MethodName
```

Trong trường hợp của ví dụ trước, hợp ngữ đã đăng ký được đặt tên là MyCLRDLL. Lớp bên trong hợp ngữ là StoredProcedures và phương thức bên trong lớp sẽ được thực thi là GetSalesPersonCount.

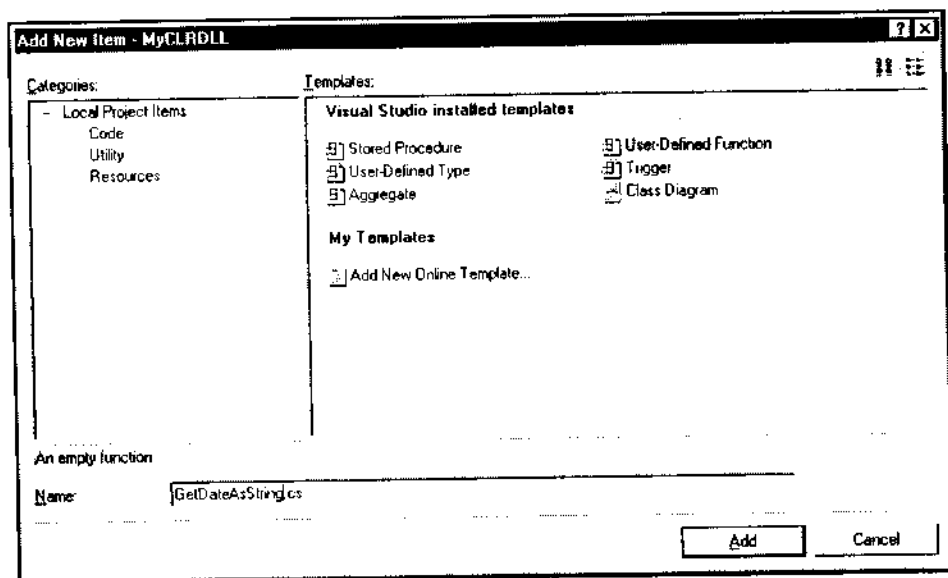
Sau khi thủ tục lưu trữ CLR đã được tạo, nó có thể được gọi chính xác như bất kỳ thủ tục lưu trữ T-SQL, như minh họa trong ví dụ dưới đây:

```
DECLARE @mycount INT
EXEC @mycount = GetSalesPersonCount
PRINT @mycount
```

Các hàm .NET do người dùng ấn định

Tạo các hàm .NET do người dùng ấn định là một tính năng mới khác có được nhờ sự hợp nhất của .NET CLR. Các hàm do người dùng ấn định (user-defined functions, gọi tắt là UDF) vốn cho ra các kiểu vô hướng phải cho ra một kiểu dữ liệu .NET có thể được chuyển đổi ngầm sang một kiểu dữ liệu SQL Server. Các hàm vô hướng được viết với .NET Framework có thể vượt trội T-SQL trong những tình huống nhất định bởi vì không giống như các hàm T-SQL, các hàm .NET được tạo nhờ dùng mã đã biên dịch. Các hàm do người dùng ấn định cũng có thể cho ra kiểu dữ liệu table, trong trường hợp này hàm phải cho ra một tập hợp kết quả.

Để thêm một UDF nhờ dùng Visual Studio 2005, bạn có thể sử dụng tùy chọn menu **Project | Add User-Defined Function** được minh họa trong hình 4.4.



Hình 4.4: Thêm một hàm do người dùng ấn định CLR

Bạn có thể thêm hàm này vào dự án hiện có như minh họa ở đây (thêm nó vào MyCLRDL mẫu đã được tạo ở ví dụ trước), hoặc tạo một dự án SQL Server mới. Ví dụ sau đây minh họa một UDF đơn giản với tên `GetDataAsString` vốn thực hiện một phép chuyển đổi cơ bản từ dạng ngày tháng sang dạng chuỗi:

```
using System;  
using System.Data.Sql;  
using System.Data.SqlTypes;
```

```
public partial class UserDefinedFunctions
{
    [SqlFunction]
    public static SqlString GetDateAsString()
    {
        DateTime CurrentDate = new DateTime();
        return CurrentDate.ToString();
    }
};
```

Ở đây, lưu ý rằng namespace `System.Data.SqlServer` là không cần thiết, vì hàm đặc trưng này không thực hiện bất kỳ sự truy cập dữ liệu. Kế tiếp, bạn có thể thấy theo mặc định, Visual Studio 2005 tạo ra lớp `UserDefinedFunctions` để chứa mọi phương thức mà hợp ngữ này sẽ trình bày ở dạng các UDF. Bạn cũng có thể nhận thấy thuộc tính `[SqlFunction]` được sử dụng để nhận biết phương thức `GetDateAsString` là một UDF. Mã ở đây chỉ chuyển đổi ngày tháng hệ thống sang một kiểu dữ liệu chuỗi.

Để tạo hàm, hợp ngữ trước tiên phải được tạo như bạn thấy trong ví dụ thủ tục lưu trữ. Nếu bạn đang sử dụng Visual Studio 2005, bạn chỉ cần chọn tùy chọn `Build | Deploy Solution`. Nếu bạn đang thực hiện điều này bằng tay và nó được đưa vào một hợp ngữ với các đối tượng CLR khác, trước tiên bạn cần loại bỏ các đối tượng đó, loại bỏ hợp ngữ, rồi tạo lại hợp ngữ và các đối tượng. Vì phương thức này đã được thêm vào `MyCLRDLL` vốn đã được sử dụng trong một hợp ngữ và một thủ tục lưu trữ, câu lệnh `DROP PROCEDURE` và câu lệnh `DROP ASSEMBLY` trước tiên phải được chạy để xóa các đối tượng cơ sở dữ liệu phụ thuộc trước khi `.NET DLL` đã cập nhật có thể được tải lại vào một hợp ngữ. Sau khi đã xóa các đối tượng hiện tại, câu lệnh `CREATE ASSEMBLY` sau đây có thể được chạy để tải lại DLL mới:

```
CREATE ASSEMBLY MyCLRDLL
FROM '\\MyFileShare\\Code Library\\MyCLRDLL.dll'
```

Bây giờ, câu lệnh `CREATE FUNCTION` được sử dụng để tạo một hàm SQL Server mới vốn thực thi phương thức thích hợp trong hợp ngữ. Danh sách sau đây minh họa cách câu lệnh `CREATE FUNCTION` có thể tạo một hàm `.NET` do người dùng ấn định:

```
CREATE FUNCTION GetDateAsString()
RETURNS nvarchar(256)
```

EXTERNAL NAME

MyCLRDLL.UserDefinedFunctions.GetDateAsString

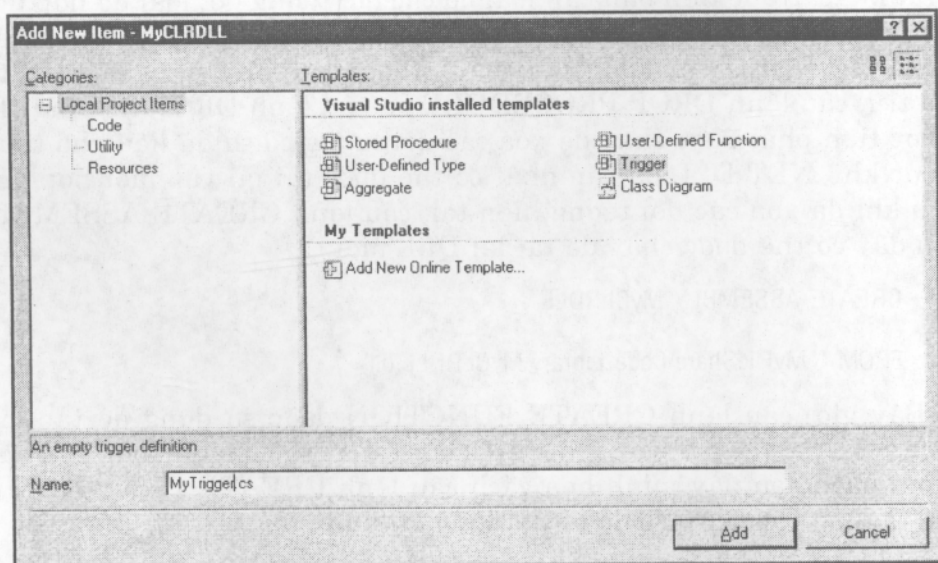
Với các hàm do người dùng ấn định, câu lệnh CREATE FUNCTION đã được mở rộng với mệnh đề EXTERNAL NAME, về cơ bản mệnh đề này liên kết tên hàm do người dùng ấn định với phương thức thích hợp trong hợp ngữ .NET. Trong ví dụ này, hàm GetDateAsString đang sử dụng hợp ngữ MyCLRDLL, trong hợp ngữ đó, nó đang dùng lớp UserDefinedFunctions và phương thức GetDateAsString bên trong lớp đó.

Sau khi hàm đã được tạo, nó có thể được gọi như một hàm SQL Server bình thường. Bạn có thể xem cách thực thi hàm GetDateAsString trong ví dụ dưới đây:

```
SELECT dbo.GetDateAsString ()
```

Các bộ kích khởi .NET

Ngoài các thủ tục lưu trữ và các hàm do người dùng ấn định, các khả năng hợp nhất .NET mới trong SQL Server 2005 còn cho phép tạo các bộ kích khởi .NET do người dùng ấn định. Để thêm một bộ kích khởi do người dùng ấn định (user-defined trigger, gọi tắt là UDT) nhờ dùng Visual Studio 2005, bạn có thể sử dụng tùy chọn menu Project | Add Trigger, như minh họa ở hình 4.5.



Hình 4.5: Thêm một bộ kích khởi CLR

Giống như với các đối tượng cơ sở dữ liệu CLR khác, bạn chọn tùy chọn Trigger từ danh sách các mẫu (template) rồi cung cấp tên của bộ kích khởi trong dòng nhắc. Visual Studio 2005 sẽ tạo một stub file mà bạn có thể thêm mã của bạn vào. Danh sách mã trong ví dụ sau đây minh họa một bộ kích khởi CLR đơn giản được gọi là MyTrigger:

```
using System;
using System.Data;
using System.Data.Sql;
using System.Data.SqlClient;
using System.Data.SqlTypes;

public partial class Triggers
{
    // Enter existing table or view for the target
    // and uncomment the attribute line
    [SqlTrigger (Name="MyTrigger",
        Target="Person.ContactType", Event="FOR INSERT")]
    public static void MyTrigger()
    {
        SqlTriggerContext oTriggerContext =
            SqlContext.GetTriggerContext();
        SqlPipe sPipe = SqlContext.GetPipe();
        SqlCommand sqlCmd = SqlContext.GetCommand();
        if (oTriggerContext.TriggerAction == TriggerAction.Insert)
        {
            sqlCmd.CommandText = "SELECT * FROM inserted";
            sPipe.Execute(sqlCmd);
        }
    }
}
```

Giống như các ví dụ khác, stub file chứa các chỉ thị nhập thích hợp cũng như tạo một lớp (class), trong trường hợp này là lớp Triggers, và một phương thức với thuộc tính thích hợp của nó. Ví dụ mã này sử dụng hai đối tượng ADO.NET mới: đối tượng SqlTriggerContext và đối tượng SqlPipe. Đối tượng SqlTriggerContext cung cấp thông tin về hành động kích khởi và các cột chịu tác động. Đối với SqlTriggerContext luôn được thể hiện bởi đối tượng SqlContext. Nói chung, đối tượng SqlContext

cung cấp thông tin về ngữ cảnh của người gọi. Cụ thể, trong trường hợp này, đối tượng `SqlContext` cho phép mã truy cập table ảo vốn đã được tạo trong lúc thực thi bộ kích khởi (trigger). Table ảo này chứa dữ liệu vốn làm cho trigger hoạt động.

Kế tiếp, một đối tượng `SqlPipe` được tạo. Đối tượng `SqlPipe` tiêu biểu cho một ống dẫn truyền thông tin giữa CLR và mã đang gọi. Ở đây, đối tượng `SqlPipe` cho phép trigger mở rộng giao tiếp với người gọi bên ngoài. Sau đó đối tượng `SqlContext` được sử dụng để xác định hành động trigger có phải là một thao tác chèn hay không. Nếu đúng như vậy thì các nội dung của table trigger ảo được truy xuất và được gửi đến người gọi nhờ dùng phương thức `Execute` của đối tượng `SqlPipe`.

Ngay khi mã đã được tạo, bạn có thể triển khai nó ra cơ sở dữ liệu dữ liệu nhờ dùng tùy chọn `Visual Studio 2005 Build & Deploy solution` hoặc loại bỏ và tạo lại bằng tay hợp ngữ cũng như bất kỳ đối tượng lệ thuộc như bạn đã thấy trong các ví dụ thủ tục lưu trữ và UDF trước đây của chương này. Để tạo một trigger CLR bằng tay, bạn có thể dùng câu lệnh `CREATE TRIGGER` mà bạn sẽ thấy trong ví dụ kế tiếp. Mã sau đây minh họa cách tạo trigger mở rộng trên table `Person.ContactType` trong cơ sở dữ liệu `AdventureWorks`:

```
CREATE TRIGGER MyTrigger
ON Person.ContactType
FOR INSERT
AS EXTERNAL NAME
MyCLRDLL.Triggers.MyTrigger
```

Giống như trong các ví dụ .NET khác, trigger mở rộng được tạo nhờ dùng câu lệnh `CREATE TRIGGER`. Câu lệnh `CREATE TRIGGER` đã được mở rộng với mệnh đề `AS EXTERNAL NAME`, mệnh đề này liên kết trigger với một phương thức trong một hợp ngữ. Ở đây, mệnh đề `EXTERNAL NAME` chỉ đến hợp ngữ `MyCLRDLL`. Bên trong lớp `Triggers` của namespace đó, phương thức `MyTrigger` có chứa mã sẽ được thực thi khi trigger mở rộng này được kích khởi.

Trigger .NET sẽ được kích khởi cho mọi thao tác chèn được thực hiện trên table `Job`. Chẳng hạn, câu lệnh `INSERT` sau đây sẽ thêm một hàng vào table `Person.ContactType`, vốn sẽ làm cho trigger .NET kích khởi:

```
INSERT INTO Person.ContactType VALUES(102, 'The Big Boss',
```

```
'2004-07-20 00:00:00.000')
```

Trigger mẫu, `MyTrigger`, thực hiện một câu lệnh `select` (chọn) trên giá trị hàng được chèn vào. Sau đó, nó sử dụng đối tượng `SqlPipe` để gửi

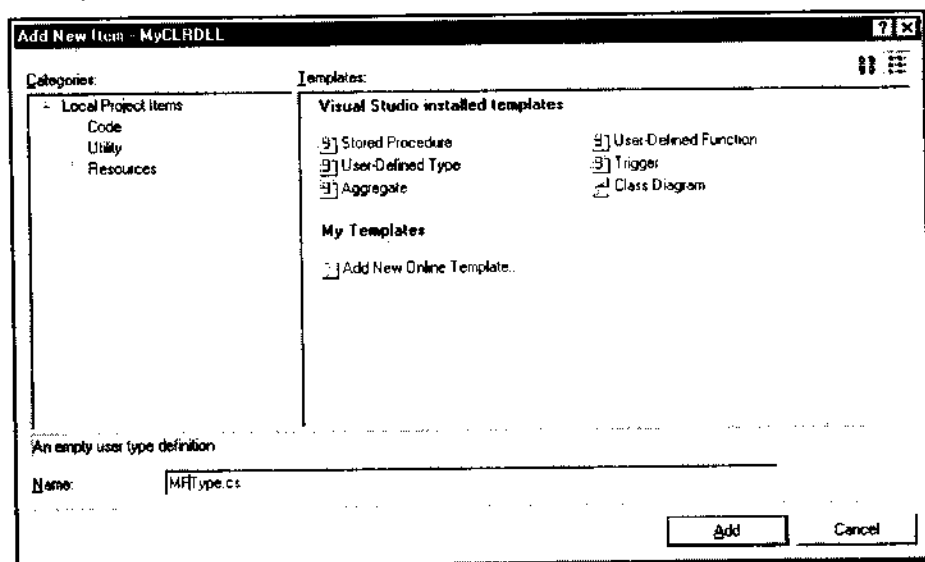
kết quả trở lại người gọi. Trong ví dụ ở đây, trigger sẽ gọi các nội dung của các giá trị hàng được chèn vào trở lại người gọi.

Các kiểu dữ liệu CLR do người dùng ấn định

Một tính năng mới quan trọng khác trong SQL Server 2005 vốn có được nhờ sự hợp nhất của .NET CLR là khả năng tạo các kiểu dữ liệu do người dùng ấn định (UDT) thật sự. Nhờ dùng UDT, bạn có thể mở rộng các kiểu dữ liệu thô do SQL Server cung cấp và thêm các kiểu dữ liệu đã được chuyên biệt hóa vào ứng dụng hay môi trường của bạn.

Trong ví dụ sau đây, bạn sẽ xem cách tạo một UDT tiêu biểu cho một mã gender (giống): M cho male (name) hoặc F cho female (nữ). Tuy bạn có thể chứa mã này trong một trường ký tự một byte chuẩn, nhưng việc sử dụng một UDT bảo đảm trường sẽ chấp nhận chỉ hai giá trị này mà không cần bổ sung gì thêm cho các trigger, các ràng buộc, hay các kỹ thuật hợp chuẩn khác.

Nếu bạn đang sử dụng Visual Studio 2005, cách tốt nhất để tạo một UDT là dùng các mẫu SQL Server. Để tạo một UDT mới, bạn nhấp phải vào dự án của bạn trong Visual Studio 2005 và chọn Add | Add Class từ menu ngữ cảnh. Thao tác này sẽ hiển thị hộp thoại Add New Item như minh họa ở hình 4.6.



Hình 4.6: Tạo một .NET SQL Server UDT

Chọn User-Defined Type từ danh sách các mẫu SQL Server. Nhập tên mà bạn muốn gán cho lớp rồi nhấp Open để nhờ Visual Studio tạo một stub file cho UDT. Stub file thực thi bốn phương thức mà SQL

Server 2005 yêu cầu cho mọi UDT. Các phương thức này là chung để thực thi các yêu cầu hợp đồng SQL Server UDT - bạn có thể tùy ý thêm mã để làm cho UDT thực hiện những hành động ý nghĩa. Bảng 4.1 liệt kê bốn phương thức UDT bắt buộc.

Bảng 4.1: Các phương thức UDT bắt buộc

Phương thức	Mô tả
IsNullabe	Phương thức bắt buộc này được sử dụng để chỉ rõ đối tượng có null hay không. SQL Server 2005 yêu cầu mọi UDT để thực thi khả năng null, vì vậy phương thức này phải luôn cho ra true.
Parse	Phương thức bắt buộc này chấp nhận một thông số chuỗi và chứa nó ở dạng một UDT.
ToString	Phương thức bắt buộc này chuyển đổi các nội dung của UDT thành một chuỗi.
Default constructor	Phương thức bắt buộc này tạo một instance mới của UDT.

Bạn có thể xem lớp MFTYPE đã hoàn chỉnh dùng để thực thi một UDT cho các mã M (male) và F (female) trong danh sách sau:

```
using System;
using System.Data.Sql;
using System.Data.SqlTypes;

[Serializable]
[SqlUserDefinedType(Format.SerializedDataWithMetadata,
MaxByteSize=512)]
public class MFTYPE: INullable
{
    string m_value;
    public override string ToString()
    {
        string s = "null";
        if (m_value != null)
        {
            s = m_value.ToString();
            return s;
        }
    }
}
```

```
        else return m_value.ToString();
    }

    public bool IsNull
    {
        get
        {
            if (m_value == null)
                return true;
            else return false;
        }
    }

    public static MFTYPE Null
    {
        get
        {
            MFTYPE h = new MFTYPE();
            return h;
        }
    }

    public static MFTYPE Parse(SqlString s)
    {
        if (s.IsNull || s.Value.ToLower() == "null")
            return Null;
        MFTYPE u = new MFTYPE();
        u.Value = s.ToString();
        return u;
    }

    // Create a Value Property
    public SqlString Value
    {
        get
        {
            return (m_value);
        }
    }
}
```

```

    }
    set
    {
        if (value == "M" || value == "F")
        {
            m_value = value.ToString();
        }
        else
        {
            throw new ArgumentException
                ("MFTYPE data type must be M or F");
        }
    }
}
}

```

Mục đầu tiên trong mã này về cơ bản là một mẫu (template) được yêu cầu bởi tất cả các kiểu dữ liệu cho người dùng ấn định. Các thuộc tính của lớp phải có khả năng chuỗi hóa, lớp phải thực thi giao diện INullable, và tên lớp được xác lập sang tên của UDT. Bạn có thể tùy ý thêm giao diện IComparable. Trong ví dụ này, MFTYPE là tên lớp. Một biến chuỗi tên m_value được khai báo để chứa các nội dung của mã. Kế tiếp, bạn có thể thấy phương thức ToString bắt buộc. Phương thức ToString kiểm tra để xem các nội dung của biến m_value có null hay không. Nếu có thì chuỗi "null" được trả về. Nếu không, phương thức ToString của m_value trả về trị chuỗi của các nội dung.

Mục kế tiếp trong mã định nghĩa thuộc tính IsNull. Phương thức ghép của thuộc tính này kiểm tra các nội dung của biến m_value và trả về trị true nếu m_value là null. Nếu không, phương thức get trả về trị false. Kế tiếp, bạn có thể thấy phương thức Null, phương thức này được tạo bởi mẫu để thực thi yêu cầu của UDT cho khả năng null.

Phương thức Parse chấp nhận một đối số chuỗi, nó chứa đối số này trong thuộc tính Value của đối tượng. Bạn có thể xem định nghĩa cho thuộc tính Value ở phía dưới trong mã. Phương thức Parse phải được khai báo là static (tĩnh), hoặc nếu bạn đang sử dụng VB.NET, nó phải là một thuộc tính Shared.

Thuộc tính Value là của phần thực thi này. Trong ví dụ ở đây, thuộc tính Value được sử dụng để chứa và truy xuất trị của UDT. Nó cũng chịu trách nhiệm hiệu chỉnh các trị chấp nhận được. Trong phương thức set, bạn có thể nhận thấy chỉ các trị của M hay F là được chấp nhận. Việc cố

dùng bất kỳ trị khác sẽ làm tung ra một biệt lệ nhằm thông báo cho người gọi biết rằng “kiểu dữ liệu MFTYPE phải là M hay F”.

Rất giống như hàm hay thủ tục lưu trữ CLR, sau khi mã đã hoàn chỉnh, nó được biên dịch thành một DLL. DLL đó sau đó sẽ được nhập ở dạng một hợp ngữ SQL Server nhờ dùng câu lệnh CREATE ASSEMBLY hoặc tùy chọn Visual Studio 2005 Deploy Solution. (Tùy chọn Visual Studio 2005 Deploy Solution tạo cả hợp ngữ và UDT.)

Để thêm UDT bằng tay vào một cơ sở dữ liệu, bạn có thể dùng câu lệnh CREATE TYPE tương tự như sau:

```
CREATE TYPE MFTYPE EXTERNAL NAME MyCLRDLL.MFTYPE
```

Giống như khi tạo các đối tượng cơ sở dữ liệu .NET khác, từ khóa EXTERNAL NAME được sử dụng để chỉ định hợp ngữ và namespace cho UDT. Trong trường hợp này, vì bản thân UDT được thực thi như là một lớp nên không có tên phương thức nào được yêu cầu. Trị của MyCLRDLL nhận biết hợp ngữ, và trị của MFTYPE nhận biết lớp của UDT. Để xem các UDT đã được tạo trong một cơ sở dữ liệu, bạn có thể truy vấn khung xem Sys.Types, như minh họa ở đây:

```
SELECT * FROM sys.Types
```

Ngay khi UDT được tạo, bạn có thể sử dụng nó trong T_SQL giống như các kiểu dữ liệu ghi chú của SQL Server. Tuy nhiên, vì các UDT có chứa các phương thức và thuộc tính, nên sẽ có những điểm khác biệt. Ví dụ sau đây minh họa cách MFTYPE UDT có thể được sử dụng như là một biến và cách truy cập thuộc tính Value của nó:

```
DECLARE @mf MFTYPE
```

```
SET @mf.Value='N'
```

```
PRINT @mf.Value
```

Trong danh sách này, biến UDT được khai báo nhờ dùng câu lệnh T_SQL DECLARE chuẩn. Bạn có thể truy cập các thành viên của UDT bằng cách đặt ký hiệu (.) ở phía trước chúng. Trong danh sách này, câu lệnh SET được sử dụng cố gán trị của N cho thuộc tính Value của UDT. Bởi vì N không phải là một trị hợp lệ nên lỗi sau đây được tạo ra:

```
.Net SqlClient Data Provider: Msg 6522, Level 16, State 1, Line 2
```

```
A CLR error occurred during execution of 'MFTYPE':
```

```
System.ArgumentException: MFTYPE data type must be M or F
```

```
at MFTYPE.set_Value(SqlString value)
```

Cũng giống như các UDT có thể dùng làm các biến, chúng cũng có thể được sử dụng để tạo các cột. Danh sách sau đây minh họa việc tạo một table sử dụng MFTYPE UDT:

```
CREATE TABLE MyContacts
```

```
(ContactID int,
```

```
FirstName varchar(25),
```

```
LastName varchar(25),
```

```
Gender MFTYPE)
```

Trong khi tạo các cột với kiểu UDT giống như khi sử dụng một kiểu dữ liệu gốc, việc gán các trị cho UDT thì hơi khác với gán cột chuẩn. Các UDT phức tạp có thể chứa nhiều trị. Trong trường hợp này, vì UDT dùng một trị đơn giản nên bạn có thể gán các trị cho nó chính xác như bạn làm với bất kỳ kiểu dữ liệu cài sẵn. Ví dụ này minh họa cách chèn một hàng trong table MyContacts mẫu vốn có chứa MFTYPE UDT:

```
INSERT INTO MyContacts VALUES (1, 'Michael', 'Otey', 'M')
```

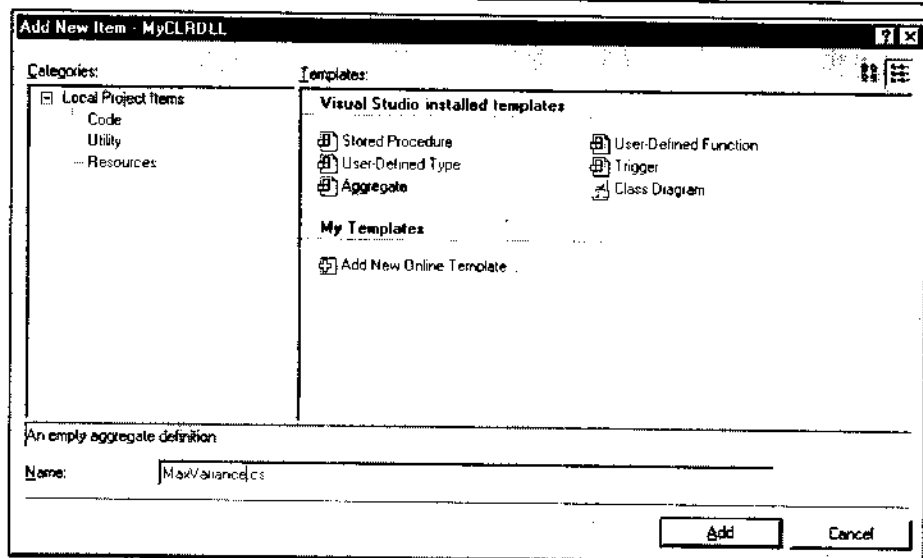
Để truy xuất nội dung của UDT nhờ dùng câu lệnh SELECT, bạn cần dùng ký hiệu UDT.Member, như minh họa ở đây khi tham chiếu một cột UDT.

```
SELECT ContactID, LastName, Gender, Value FROM MyContacts
```

Các hàm gộp CLR do người dùng ấn định

Hàm gộp do người dùng ấn định (user-defined aggregate, gọi tắt là UDAGG) là một loại đối tượng cơ sở dữ liệu .NET mới khác được giới thiệu trong SQL Server 2005. Về cơ bản, một hàm gộp do người dùng ấn định là một hàm mở rộng cho phép bạn gộp các trị qua một nhóm trong lúc xử lý một mẫu truy vấn (query). SQL Server luôn cung cấp một tập hợp hàm gộp cơ bản như MIN, MAX và SUM mà bạn có thể sử dụng trên một mẫu truy vấn. Các hàm gộp do người dùng ấn định cho phép bạn mở rộng nhóm hàm gộp cơ bản này bằng các hàng của riêng bạn. Giống như các hàm gộp gốc, các hàm gộp do người dùng ấn định cho phép bạn thực hiện các phép tính trên một tập hợp các trị và trả về một trị duy nhất. Khi bạn tạo một UDAGG, bạn cung cấp logic sẽ thực hiện phép gộp. Trong mục này, bạn sẽ xem cách tạo một UDAGG đơn giản vốn tính trị trung bình cho một tập hợp số.

Để tạo một UDAGG mới nhờ dùng Visual Studio 2005, bạn chọn tùy chọn Project | Add Aggregate từ menu để hiển thị hộp thoại Add New Item như bạn thấy ở hình 4.7.



Hình 4.7: Thêm một hàm gộp do người dùng ấn định

Chọn Aggregate từ danh sách các mẫu SQL Server rồi nhập tên cho lớp và nhấp Open. Visual Studio sẽ tạo một stub file cho lớp Aggregate. Giống như một UDT, stub file của lớp Aggregate thực thi bốn phương thức mà SQL Server 2005 yêu cầu cho tất cả các hàm gộp do người dùng ấn định. Bảng 4.2 liệt kê bốn phương thức bắt buộc cho tất cả UDAGG.

Bảng 4.2: Các phương thức gộp bắt buộc

Phương thức	Mô tả
Init	Phương thức bắt buộc này khởi tạo đối tượng. Nó được viện dẫn một lần cho mỗi phép gộp.
Accumulate	Phương thức bắt buộc này được viện dẫn một lần cho mỗi phần tử trong tập hợp đang được gộp.
Merge	Phương thức bắt buộc này được viện dẫn khi server thực thi mẫu truy vấn nhờ dùng phép song song. Phương thức này được sử dụng để trộn dữ liệu từ các instance (thể hiện) song song khác nhau lại với nhau.
Terminate	Phương thức bắt buộc này cho ra kết quả của phép gộp. Nó được viện dẫn một lần sau khi mọi phần tử đã được xử lý xong.

Bạn có thể thấy lớp MaxVariance mẫu dùng để thực thi một hàm gộp MaxVar do người dùng ấn định trong danh sách sau:

```
[Serializable]
[SqlUserDefinedAggregate(Format.SerializedDataWithMetadata, MaxByteSize = 512)]
public class MaxVariance
{
    int m_LowValue;
    int m_HighValue;

    public void Init()
    {
        m_LowValue = 999999999;
        m_HighValue = -999999999;
    }
    public void Accumulate(SqlInt32 Value)
    {
        if (Value > m_HighValue) m_HighValue = (int)Value;
        if (Value < m_LowValue) m_LowValue = (int)Value;
    }
    public void Merge (MaxVariance Group)
    {
        if (Group.GetHighValue() > m_HighValue)
            m_HighValue = Group.GetHighValue();
        if (Group.GetLowValue() < m_LowValue)
            m_LowValue = Group.GetLowValue();
    }
    public SqlInt32 Terminate ()
    {
        return m_HighValue - m_LowValue;
    }
    // Helper methods
    public int GetLowValue()
    {
        return m_LowValue;
    }
    public int GetHighValue()
```

```

    return m_HighValue;
}
}

```

Ở đầu lớp này, bạn thấy thuộc tính chuỗi hóa được yêu cầu bởi các lớp UDAGG. Kế tiếp, hai biến được khai báo để chứa các trị nhỏ nhất và lớn nhất mà hàm gộp bắt gặp. Sau đó, trong phương thức Init hai biến được gán các trị cao và thấp, bảo đảm chúng sẽ được gán các trị từ danh sách. Trong khi phương thức Init được gọi chỉ một lần thì phương thức Accumulate được gọi một lần cho mỗi hàng trong tập hợp kết quả (result set). Trong ví dụ này, phương thức Accumulate so sánh trị đến với các trị được chứa trong các biến m_HighValue và m_LowValue. Nếu trị đến cao hơn trị cao hiện tại, nó được chứa trong biến m_HighValue. Nếu trị này thấp hơn trị của m_LowValue thì nó được chứa trong m_LowValue. Mặt khác, không có hành động nào được thực hiện bởi phương thức Accumulate cho UDAGG này.

Lưu ý

Do các UDAGG được chuỗi hóa nên bạn cần chú ý tổng yêu cầu lưu trữ của UDAGG. UDAGG được chuỗi hóa theo mỗi lần viện dẫn của phương thức Accumulate và nó không được vượt quá kích cỡ cột tối đa 8000 byte.

Phương thức Merge được sử dụng khi UDAGG được xử lý song song, thường đây không phải là trường hợp cho hầu hết mọi mẫu truy vấn. Nếu Merge được gọi, nhiệm vụ của nó là nhập các trị gộp hiện tại từ instance song song. Bạn có thể thấy ở đây nó thực hiện điều đó nhờ dùng hai phương thức helper vốn xuất các trị trong các biến m_HighValue và m_LowValue. Các trị này được so sánh với các trị thoát, và nếu chúng cao hơn hay thấp hơn, chúng sẽ thay các trị hiện tại trong m_HighValue và m_LowValue.

Phương thức Terminate được gọi một lần sau khi mọi kết quả đã được xử lý xong. Với ví dụ này, phương thức Terminate chỉ lấy giá trị cao nhất trừ giá trị thấp nhất được tìm thấy và trả sai số về người gọi.

Sau khi biên dịch lớp .NET thành một DLL, bạn có thể nhập DLL ở dạng một hợp ngữ SQL Server nhờ dùng tùy chọn Visual Studio 2005 Deploy Solution hoặc sử dụng bằng tay câu lệnh CREATE ASSEMBLY và các câu lệnh CREATE AGGGRATE. Câu lệnh CREATE AGGREGATE được minh họa dưới đây:

```
CREATE AGGREGATE MaxVariance(@MyInt int)
```


RETURNS int

EXTERNAL NAME MyCLRDLL.MaxVariance

Ví dụ này cho thấy việc tạo một hàm gộp tên MaxVariance. Hàm gộp (aggregate) này có thể được dùng với các kiểu dữ liệu số nguyên và nó cho ra một trị nguyên. Trong mệnh đề EXTERNAL NAME, bạn có thể nhận thấy mã cho UDAGG này được tìm thấy trong lớp MaxVariance của hợp ngữ MyCLRDLL.

Bạn có thể sử dụng UDAGG cũng giống như các hàm gộp cài sẵn trong SQL Server. Một điểm khác biệt nhỏ là UDAGG cần có tên schema đứng trước nhằm cho phép hệ thống định vị nó. Dòng sau đây minh họa việc sử dụng MaxVariance UDAGG:

```
SELECT dbo.MaxVariance (MinQty) FROM Sales.SpecialOffer
```

Kết quả của UDAGG này sẽ là sai số giữa các trị cao và thấp được tìm thấy trong cột Sales.SpecialOffer.

Sự bảo mật đối tượng cơ sở dữ liệu .NET

Không có phần thảo luận nào về các tính năng CLR mới là hoàn chỉnh nếu như không đề cập đến các vấn đề bảo mật liên quan đến việc sử dụng các hợp ngữ .NET và SQL Server CLR. Không giống như T-SQL, vốn không có bất kỳ tiện ích gốc nào để tham chiếu các tài nguyên bên ngoài cơ sở dữ liệu, các hợp ngữ .NET hoàn toàn có khả năng xử lý cả hai tài nguyên hệ thống và mạng. Do đó, sự bảo đảm an toàn cho chúng là một khía cạnh quan trọng trong việc phát triển chúng. Với SQL Server 2005, Microsoft đã hợp nhất mô hình bảo mật SQL Server dựa vào user với mô hình bảo mật CLR dựa vào các permission (sự cho phép). Theo mô hình bảo mật SQL Server, các user có thể truy cập chỉ các đối tượng cơ sở dữ liệu - bao gồm các đối tượng được tạo từ các hợp ngữ .NET - mà họ được phép. Hệ thống bảo mật CLR mở rộng khả năng này bằng cách cung cấp sự điều khiển loại tài nguyên hệ thống có thể được truy cập bởi mã .NET đang chạy trên server. Các permission của hệ thống bảo mật CLR được chỉ định vào lúc hợp ngữ được tạo bằng cách dùng mệnh đề WITH PERMISSION SET của câu lệnh CREATE ASSEMBLY. Bảng 4.3 tóm tắt các tùy chọn cho các permission của hệ thống bảo mật cơ sở dữ liệu CLR vốn có thể được áp dụng cho các đối tượng cơ sở dữ liệu SQL Server.

Bảng 4.3: Các tùy chọn bảo mật đối tượng cơ sở dữ liệu CLR

Sự bảo mật CLR	Được phép truy cập bên ngoài	Các cuộc gọi đến mã chưa quản lý
SAFE	Không được truy cập bên ngoài	Không có cuộc gọi nào đến mã chưa quản lý
EXTERNAL_ACCESS	Sự truy cập ngoài được cho phép qua các API quản lý	Không có cuộc gọi nào đến mã chưa quản lý
UNSAFE	Được phép truy cập ngoài	Cho phép các cuộc gọi đến mã chưa quản lý

Việc dùng SAFE sẽ hạn chế mọi sự truy cập bên ngoài. EXTERNAL_ACCESS permission cho phép truy cập một số tài nguyên bên ngoài nhờ dùng các API đã quản lý. SQL Server đóng vai người gọi để truy cập các tài nguyên bên ngoài. Bạn phải có EXTERNAL_ACCESS permission mới để tạo các đối tượng với xác lập permission này. UNSAFE permission cơ bản là một loại permission bất cứ cái gì đến. Tất cả các tài nguyên hệ thống có thể được truy cập và các cuộc gọi đến cả hai mã đã quản lý và chưa quản lý là được phép. Chỉ những người quản lý hệ thống là có thể tạo các đối tượng với UNSAFE permission.

Khi nào cần sử dụng các đối tượng cơ sở dữ liệu CLR

Các đối tượng cơ sở dữ liệu được tạo nhờ dùng CLR thích hợp nhất cho việc thay thế các thủ tục lưu trữ mở rộng vốn đòi hỏi sự truy cập vào các tài nguyên hệ thống bên ngoài, cho việc tạo các đối tượng cơ sở dữ liệu yêu cầu logic phức tạp, hay cho việc tạo các đối tượng cơ sở dữ liệu có khả năng được truyền giữa cơ sở dữ liệu và lớp data tier của một ứng dụng. Đối với sự truy cập dữ liệu thô và cập nhật các hàm thì chúng không thích hợp bằng T-SQL.

Các cải tiến của T-SQL

Với tất cả những tính năng mới liên quan đến .NET, bạn có thể thắc mắc Microsoft có dự định loại bỏ sự hỗ trợ T-SQL hay không, nhưng chắc chắn điều đó không xảy ra. T-SQL vẫn là ngôn ngữ thích hợp nhất để sử dụng cho sự truy cập dữ liệu thô, và như bạn có thể đã nhận thấy, cú pháp T-SQL đã được sử dụng trong các ví dụ trước, vì vậy chắc chắn T-SQL sẽ vẫn được hỗ trợ trong thời gian sắp tới. Thật vậy, Microsoft đã đưa ra nhiều cải tiến quan trọng cho T-SQL mà bạn sẽ tìm hiểu trong mục này.

Các cải tiến về TOP

Trong SQL Server 2000, bạn buộc phải dùng một trị hằng kết hợp với một mệnh đề tốt. Nói cách khác, bạn có thể chọn TOP 5 hay các hàng TOP 10, trong đó trị của 5 hay 10 là một hằng số. Với SQL Server 2005, hàm TOP bây giờ cho phép dùng một biểu thức kết hợp với mệnh đề TOP. Một biểu thức có thể là bất kỳ biểu thức T-SQL được phép, bao gồm cả một biến hay một mẫu truy vấn con vô hướng. Mệnh đề TOP cũng được hỗ trợ trong các câu lệnh INSERT, UPDATE và DELETE. Chúng làm cho mệnh đề TOP có sự linh hoạt tốt hơn nhiều so với trước đây. Ví dụ sau đây minh họa việc sử dụng mệnh đề TOP mới:

```
USE AdventureWorks
```

```
DECLARE @MyTop INT
```

```
SET @MyTop = 15
```

```
SELECT TOP (@MyTop) CustomerID, SalesPerson FROM Sales.Customer
```

Common Table Expressions (CTE)

Một tính năng T-SQL mới khác là hỗ trợ common table expressions (CTEs). CTE giống như một khung xem (view); tuy nhiên, chúng được nhúng trong một mẫu truy vấn. Lý do chính khiến Microsoft đưa CTE vào SQL Server 2005 là cung cấp một cơ cấu để xử lý các mẫu truy vấn đệ quy. Phép đệ quy có được nhờ dữ kiện một CTE được phép ám chỉ chính nó. Để tránh khả năng làm tràn ngập hệ thống bằng một mẫu truy vấn đệ quy có cấu trúc kém, SQL Server thực thi một giới hạn trong toàn server trên mức tối đa được phép của phép đệ quy, với số mức tối đa mặc định là 100. Một CTE được thực thi như là thành phần của từ khóa WITH và có thể được sử dụng với các câu lệnh SELECT, INSERT, UPDATE và DELETE. Để thực thi các mẫu truy vấn đệ quy nhờ dùng CTE mới, bạn phải sử dụng một cú pháp đặc biệt, như minh họa trong mã đơn giản dưới đây. Ví dụ này thực hiện một mẫu truy vấn đệ quy đơn giản nhờ dùng table HumanResources.Employee trong cơ sở dữ liệu AdventureWorks mẫu:

```
USE AdventureWorks
```

```
WITH EmployeeChart(EmployeeID, ManagerID, Title)
```

```
AS
```

```
(SELECT EmployeeID, ManagerID, Title
```

```
FROM HumanResources.Employee
```

```
WHERE EmployeeID = 3
```

```
UNION ALL
```

```

SELECT L2.EmployeeID, L2. ManagerID, L2.Title
FROM HumanResources.Employee AS L2
JOIN EmployeeChart
ON L2.ManagerID = EmployeeChart.EmployeeID

SELECT * from EmployeeChart

```

Để sử dụng một CTE, trước tiên bạn viết một mệnh đề **WITH** mà bạn dùng để đặt tên CTE và chỉ định các cột để kết buộc vào một câu lệnh **SELECT**. Phải có một dấu chấm phẩy ở trước từ khóa **WITH** nếu đó không phải là câu lệnh đầu tiên trong một lô. Câu lệnh **SELECT** đầu tiên được gọi là thành viên neo (anchor member), và nó không được ám chỉ chính nó. Trong trường hợp này, nó truy xuất các cột EmployeeID, ManagerID và Title từ table AdventureWorks Employee. Câu lệnh **SELECT** thứ hai tham chiếu CTE và được gọi là thành viên đệ quy (recursive member). Trong trường hợp này, nó truy xuất cùng các cột và được nối với thành viên neo trên cột ManagerID. Bạn có thể xem kết quả của CTE này trong danh sách dưới đây:

EmployeeID	ManagerID	Title
3	12	Engineering Manager
4	3	Senior Tool Designer
9	3	Design Engineer
11	3	Design Engineer
158	3	Research and Development Manager
263	3	Senior Tool Designer
267	3	Senior Design Engineer
270	3	Design Engineer
5	263	Tool Designer
265	263	Tool Designer
79	158	Research and Development Engineer
114	158	Research and Development Engineer
217	158	Research and Development Manager

(13 row(s) affected)

PIVOT và UNPIVOT

Sự bổ sung các toán tử quan hệ **PIVOT** và **UNPIVOT** là một tính năng mới khác được tìm thấy trong T-SQL của SQL Server 2005. Các toán tử **PIVOT** và **UNPIVOT** hữu dụng nhất cho các tình huống OLAP trong đó bạn đang xử lý dữ liệu bảng thay vì dữ liệu quan hệ. Toán tử **PIVOT** chuyển đổi một tập hợp hàng thành các cột. Toán tử **UNPIVOT** đảo ngược toán tử **PIVOT**, chuyển đổi các cột được tạo pivot trở lại thành các hàng. Tuy nhiên, phụ thuộc vào tình huống, phép **UNPIVOT** có thể không đảo ngược chính xác phép **PIVOT**. Tình huống này xảy ra

bởi vì phép PIVOT thường được cài đặt sao cho nó sẽ loại bỏ những trị nhất định. Nếu một trị bị loại bỏ trong phép PIVOT thì rõ ràng nó không thể được unpivot. Do đó, toán tử UNPIVOT không luôn cho ra một ảnh phản chiếu chính xác của điều kiện pivot ban đầu.

Bạn có thể thấy cách hoạt động của toán tử PIVOT trong các danh sách sau đây. Giả sử có một thao tác chọn đơn giản trên một file OrderSum, bạn có thể thấy một tập hợp đơn đặt hàng cho một CustomerID là 1, trong đó mỗi đơn đặt hàng có một năm liên quan.

OrderId	CustomerID	OrderYear
-----	-----	-----
100	1	2000
101	1	2000
102	1	2000
103	1	2001
104	1	2001
105	1	2002
106	1	2003
107	1	2004

(8 row(s) affected)

Bằng cách sử dụng toán tử PIVOT mới của SQL Server 2005, bạn có thể chuyển đổi tập hợp kết quả này, vốn liệt kê mỗi năm theo chiều dọc, thành một tập hợp kết quả liệt kê các năm theo chiều ngang cho mỗi khách hàng và tính tổng số đơn đặt hàng cho mỗi năm. Phép PIVOT mẫu được minh họa trong danh sách dưới đây:

```
SELECT * FROM OrderSum
    PIVOT (COUNT(OrderID)
        FOR OrderYear IN([2000], [2001], [2002], [2003], [2004]))
    AS P
    WHERE CustomerID = 1
```

Ở đây, phép PIVOT được sử dụng với câu lệnh SELECT để tạo ra một tập hợp kết quả mới. Trị đầu tiên của toán tử pivot xác định trị sẽ được đặt trong cột pivot. Trong ví dụ này, phép gộp COUNT(OrderID) tính tổng số đơn đặt hàng cho mỗi trị pivot. Từ khóa FOR xác định cột mà các trị của nó sẽ được pivot. Ví dụ này minh họa phép pivot đang được thực hiện trên cột OrderYear. Các trị được xác định bởi danh sách

từ khóa IN là các trị từ cột đã được pivot mà sẽ được dùng làm các tiêu đề cột. Bạn có thể xem tập hợp kết quả đã được pivot trong danh sách sau đây:

CustomerID	2000	2001	2002	2003	2004
1	3	2	1	1	1

Warning: Null value is eliminated by an aggregate or other SET operation.

(1 row(s) affected)

Các bộ kích khởi DDL

Các phiên bản trước của SQL Server chỉ cho phép sử dụng các bộ kích khởi (trigger) với các event thao tác dữ liệu như chèn hay cập nhật một hàng. SQL Server 2005 mở rộng khả năng này bằng cách cho phép đặt các trigger trên các event DDL chẳng hạn như tạo và loại bỏ các đối tượng cơ sở dữ liệu như các table, các khung xem, các thủ tục và các login. Các trigger DDL có thể được kết hợp với các câu lệnh CREATE, ALTER, và DROP. Điều này cho phép DBA đặt ra các hạn chế trên loại thao tác DDL có thể được thực hiện trong một cơ sở dữ liệu đã cho, hoặc bạn có thể dùng các trigger này để gửi các thông báo về những thay đổi schema quan trọng đã xảy ra trong cơ sở dữ liệu. Ví dụ sau đây minh họa cách thêm một trigger DDL có tên là NoTableUpdate vào các câu lệnh DROP Table và ALTER Table DDL:

```
CREATE TRIGGER NoTableUpdate
ON DATABASE FOR DROP_TABLE, ALTER_TABLE
AS
PRINT 'DROP TABLE and ALTER TABLE statement are not allowed'
ROLLBACK
```

Ở đây, bạn có thể thấy cách trigger DDL mới được sử dụng như thế nào để hạn chế việc sử dụng các câu lệnh DROP TABLE và ALTER TABLE. Nếu một câu lệnh ALTER TABLE hay DROP TABLE được ấn hành thì trigger NoTableUpdate sẽ cho ra một thông báo lỗi và cuộn trở lại thao tác DDL. Việc cố ấn hành một câu lệnh ALTER TABLE trong cơ sở dữ liệu có chứa trigger NoTableUpdate được minh họa ở đây:

```
DROP TABLE and ALTER TABLE statement are not allowed
.Net SqlClient Data Provider: Msg 3609, Level 16, State 2, Line 1
Transaction ended in trigger. Batch has been aborted.
```

Để thực hiện sự thay đổi cho các table trong một cơ sở dữ liệu sau khi đã cố định trigger này, trước tiên bạn cần loại bỏ trigger DDL.

Kết xuất DML

Một tính năng T-SQL mới khác trong SQL Server 2005 là khả năng cho ra kết xuất từ các câu lệnh T-SQL INSERT, UPDATE và DELETE DML. Mệnh đề Output mới cho ra dữ liệu đã sửa đổi. Chẳng hạn, câu lệnh DELETE sau đây loại bỏ tất cả các hàng ra khỏi table OrderSum:

```
DECLARE @MyOrderSumTVar TABLE(
    OrderID int,
    CustomerID int,
    OrderYear int);
DELETE FROM OrderSum
OUTPUT DELETED.* INTO @MyOrderSumTVar
SELECT * FROM @MyOrderSumTVar
```

Lưu ý table mẫu OrderSum đã được tạo trong ví dụ trước đây của chương này.

Ở đây mệnh đề OUTPUT DELETED.* chỉ định tất cả các hàng đã bị xóa sẽ được kết xuất. Với các phiên bản trước của SQL Server, bạn cũng sẽ thấy số hàng chịu ảnh hưởng bởi câu lệnh này. Bạn có thể thấy kết quả của mệnh đề T-SQL DML Output mới ở đây:

```
(8 row(s) affected)
OrderID      CustomerID  OrderYear
-----
100          1          2000
101          1          2000
102          1          2000
103          1          2001
104          1          2001
105          1          2002
106          1          2003
107          1          2004
```

```
(8 row(s) affected)
```

WAITFOR

Một tính năng T-SQL mới khác trong SQL Server 2005 là cải tiến sự hỗ trợ cho lệnh WAITFOR. Trong các phiên bản trước đây của SQL

Chương 4 - Các tính năng lập trình

Server, lệnh WAITFOR có thể chờ đợi trong chỉ một khoảng thời gian đã xác định trước. Với SQL Server 2005, lệnh WAITFOR có khả năng chờ đợi kết quả của một câu lệnh RECEIVE. Lý do chính đằng sau tính năng này là hỗ trợ lập trình T-SQL dễ dàng cho các khả năng lập hàng đợi mới được cung cấp bởi hệ thống phụ SQL Service Broker. (Bạn có thể tìm hiểu thêm về SQL Service Broker ở chương 6). Danh sách sau đây minh họa cách sử dụng lệnh WAITFOR mới kết hợp với một câu lệnh RECEIVE:

```
WAITFOR (RECEIVE TOP (1) * FROM dbo.ServiceBrokerQueue)
```

Kiểu dữ liệu varchar(max) mới

Kiểu dữ liệu varchar(max) mới cung cấp một phương cách khác cho kiểu dữ liệu text/image. Kiểu dữ liệu varchar(max) mới là một phần mở rộng của các kiểu dữ liệu varchar, nvarchar và varbinary. Giống như các kiểu dữ liệu text, ntext, và image, kiểu dữ liệu varchar(max) hỗ trợ lên đến 2GB dữ liệu. Tuy nhiên, không giống như các kiểu dữ liệu text, ntext, và image hiện có, varchar(max) có thể chứa cả hai dữ liệu ký tự và nhị phân. Cũng vậy, nó không hỗ trợ cho các text pointer.

Microsoft đã giới thiệu kiểu dữ liệu varchar(max) mới nhằm làm cho việc thao tác các kiểu dữ liệu lớn này giống như thao tác dữ liệu chuỗi chuẩn. Điều này cho phép bạn sử dụng cùng mô hình lập trình cho dữ liệu varchar(max) mà bạn dùng để thao tác dữ liệu varchar chuẩn. Mọi hàm chuỗi hoạt động trên các kiểu dữ liệu varchar(max) và các hàm SUBSTRING có thể được sử dụng để đọc các cụm dữ liệu. Ngoài ra, câu lệnh T-SQL UPDATE đã được cải tiến để hỗ trợ việc cập nhật các cụm bên trong một kiểu dữ liệu varchar(max). Bạn có thể tạo một cột nhờ dùng kiểu dữ liệu varchar(max) mới, như minh họa ở đây:

```
CREATE TABLE NewBLOB
```

```
(
```

```
    DataID INT IDENTITY NOT NULL,
```

```
    BLOBData VARCHAR(MAX) NOT NULL
```

```
)
```

Xử lý hủy giao tác

Một cải tiến quan trọng khác được thể hiện bởi T-SQL trong SQL Server 2005 là xử lý hủy giao tác (transaction abort handling). Với SQL Server 2005, một mô hình Try/Catch mới đã được thêm vào giao tác (transaction). Cấu trúc Try/Catch mới cho phép bắt các lỗi hủy giao tác

mà không làm mất ngữ cảnh giao tác. Với SQL Server 2000, tuy bạn có thể hủy một giao tác nhưng không có cách nào để duy trì ngữ cảnh của giao tác sao cho bạn có thể phục hồi hoàn chỉnh giao tác đã bị hủy. Mô hình Try/Catch mới của SQL Server 2005 cho phép bạn duy trì ngữ cảnh hoàn chỉnh của giao tác đã bị hủy, cho bạn sự lựa chọn để tạo lại giao tác. Danh sách mã sau đây minh họa cấu trúc T-SQL Try/Catch cơ bản:

```
BEGIN TRY
```

```
    <SQL Statements>
```

```
END TRY
```

```
BEGIN CATCH TRAN_ABORT
```

```
    <SQL Statements>
```

```
END CATCH
```

Giao tác được nỗ lực trong khối Try. Nếu RAISERROR với câu lệnh TRAN_ABORT được ấn hành trong khối Try, sự điều khiển được chuyển sang khối Catch. Bên trong khối Catch, biến @@error có thể được lượng giá để xác định điều kiện lỗi.

Khi nào cần sử dụng các đối tượng cơ sở dữ liệu T-SQL

Các đối tượng cơ sở dữ liệu T-SQL không đang được đưa ra trong SQL Server 2005. Thật vậy, T-SQL vẫn là sự lựa chọn tốt nhất cho những đối tượng cần thực hiện các hoạt động truy cập dữ liệu thô. Chẳng hạn, nếu bạn có các thủ tục lưu trữ mà chức năng chính của chúng là chèn, cập nhật, hay xóa các hàng dữ liệu, các đối tượng này sẽ được phát triển nhờ dùng T-SQL hơn là một trong các ngôn ngữ .NET.

Các cải tiến của ADO.NET

Ngoài các tính năng CLR và T-SQL mới, ấn bản SQL Server 2005 còn cung cấp một số cải tiến mở rộng ở phía client bằng việc kết hợp một phần cập nhật vào ADO.NET. Như bạn sẽ thấy, nhiều tính năng mà Microsoft đã thêm vào ADO.NET bộc lộ một số khả năng mới đã được thêm vào bộ máy cơ sở dữ liệu SQL Server. Ngoài các tính năng mới được liệt kê trong các mục dưới đây, ADO.NET mới còn hỗ trợ T-SQL varchar(max) và các kiểu dữ liệu XML.

Hỗ trợ con trỏ Server nhờ dùng SqlResultSet

Một trong những tính năng mới quan trọng nhất trong ADO.NET là

sự hỗ trợ các con trỏ phía server. Đây là một trong những nơi mà ADO.NET trước đây còn thiếu những tính năng được tìm thấy trong Com-based ADO. Các phiên bản trước của ADO.NET chỉ hỗ trợ các con trỏ phía client sao cho nền client phải thực hiện việc duy trì tập hợp kết quả (result set). Với các con trỏ phía server, bạn có thể chuyển công việc đó sang server. Microsoft đã bổ sung tính năng mới này chủ yếu để hỗ trợ SQL Server .NET Data Provider mới trong tiến trình, vốn chạy trên server. Microsoft bổ sung tính năng này cho các yêu cầu phía server trong tiến trình bởi vì trên server cần có sự cuộn động qua các kết quả của tập hợp kết quả và ít có sự tương tác từ người dùng. Các con trỏ phía server có thể giữ trạng thái, vốn làm giảm khả năng mở rộng và tăng nhu cầu đi vòng đến server. Tuy nhiên, do bởi các vấn đề mở rộng nên Microsoft đã thêm tính năng này vào chỉ namespace System.Data.SqlServer phía server. Sự hỗ trợ con trỏ phía server không phải là thành phần của namespace System.Data.SqlClient phía client.

Với ADO.NET 2.0, đối tượng `SqlResultSet` mới bọc lộ các con trỏ phía server cho ứng dụng của bạn. Các con trỏ này vừa có khả năng cập nhật vừa có khả năng cuộn động. Các con trỏ phía server mới được thể hiện bởi phương thức `ExecuteResultSet` mới trong đối tượng `System.Data.SqlServer SqlCommand`. Ví dụ sau đây minh họa việc sử dụng đối tượng ADO.NET `SqlResultSet`:

```
using System;

using System.Data;
using System.Data.Sql;
using System.Data.SqlServer;
using System.Data.SqlTypes;

public partial class StoredProcedures
{
    [SqlProcedure]
    public static void GetProductName()
    {
        SqlPipe myPipe = SqlContext.GetPipe();
        myPipe.Send("GetProductName: Opening server cursor");
        SqlCommand cmd = SqlContext.GetCommand();
        cmd.CommandType = CommandType.Text;
        cmd.CommandText = "SELECT Name FROM Production.Product WHERE MakeFlag
= 1";
```

```

SqlResultSet resultset = cmd.ExecuteNonQuery
(ResultSetOptions.Scrollable |
ResultSetOptions.Updatable);

```

```

if (resultset.HasRows == true)
{
    while (resultset.Read())
    {
        myPipe.Send(resultset.GetString(0));
        // You could optionally update with
        //resultset.Update();
        // or scroll back using
        //resultset.ReadLast();
    }
    resultset.Close();
}
myPipe.Send("GetProductName: Server cursor closed");
}
};

```

Ở đây, bạn có thể thấy thủ tục lưu trữ CLR mới được gọi là `GetProductName`. Trước tiên, đối tượng `SqlPipe` được sử dụng để gửi một thông báo tiến độ cho client. Sau đó, đối tượng `SqlCommand` mới được thể hiện sẽ nhận nội dung của cột `Name` trên table `Production.Product` của cơ sở dữ liệu mẫu `AdventureWorks`. Kế tiếp, một thể hiện (instance) của đối tượng `SqlResultSet` được tạo. Gần giống như `SqlDataReader`, đối tượng `SqlResultSet` được thể hiện nhờ dùng đối tượng `SqlCommand`. Trong trường hợp này, một con trỏ có thể cuộn và cập nhật được mở nhờ dùng phương thức `ExecuteResultSet`. Sau khi con trỏ đã được mở, ứng dụng của bạn có thể cuộn tới, cuộn lui, và thực hiện các hoạt động cập nhật. Vì các con trỏ phía server duy trì trạng thái và chiếm các tài nguyên server trong khoảng thời gian chúng được mở nên rất cần đóng chúng khi chúng không còn cần thiết nữa.

Sự hỗ trợ không đồng bộ

Một tính năng khác đã có trong ADO nhưng lại thiếu trong các ấn bản đầu tiên của ADO.NET là sự hỗ trợ các mẫu truy vấn không đồng bộ. Các mẫu truy vấn không đồng bộ (asynchronous queries) cho các ứng dụng client khả năng đệ trình các mẫu truy vấn mà không làm cản

trở tự tương tác của người dùng. Ở tầng giữa của các ứng dụng, sự hỗ trợ không đồng bộ ADO.NET mới cung cấp khả năng cho các ứng dụng sử dụng để ẩn hành nhiều yêu cầu cơ sở dữ liệu trên các luồng khác nhau mà không làm cản trở các luồng (thread). Sự hỗ trợ không đồng bộ này cũng có tác dụng với các phiên bản trước của SQL Server, bao gồm SQL Server 7 và 2000. Với SQL Server 2005, ADO.NET cung cấp sự hỗ trợ không đồng bộ cho cả việc mở một nối kết và thực thi các lệnh. Sự thực thi thì giống như các hoạt động không đồng bộ khác trong .NET Framework. Hoạt động không đồng bộ được bắt đầu nhờ dùng phương thức `BEGINxxx` của đối tượng và được kết thúc nhờ dùng phương thức `ENDxxx`. Đối tượng `IAsyncResult` được sử dụng để kiểm tra tình trạng hoàn tất của lệnh.

```
SqlConnection cn = new SqlConnection
("SERVER=TECA-YUKON;INTEGRATED SECURITY=True;"
 + "DATABASE=AdventureWorks;async=True");
SqlCommand cmd = new SqlCommand("SELECT * FROM Production.Product", cn);
cmd.CommandType = CommandType.Text;
try
{
    cn.Open();
    IAsyncResult myResult = cmd.BeginExecuteReader();
    while (!myResult.IsCompleted)
    {
        // Perform other code actions
    }
    // Process the contents of the reader
    SqlDataReader rdr = cmd.EndExecuteReader(myResult);

    // Close the reader
    rdr.Close();
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message);
}
finally
{
}
```

```
cn.Close();
```

```
}
```

Sự hỗ trợ không đồng bộ của ADO.NET 2.0 được thực thi trong các namespace client như namespace System.Data.SqlClient. Điểm quan trọng đầu tiên cần lưu ý trong ví dụ này là chuỗi nối kết. Để thực thi sự hỗ trợ không đồng bộ, chuỗi nối kết phải chứa các từ khóa `ASYNC=true`. Kế tiếp, chú ý đối tượng `IAsyncResult` bên trong khối `Try`. Phương thức `BeginExecuteReader` của đối tượng `SqlCommand` được sử dụng để bắt đầu một mẫu truy vấn không đồng bộ vốn cho ra tất cả các hàng trong table `Production.Product`. Sự điều khiển được đưa trở lại ứng dụng ngay sau khi câu lệnh được thực thi. Ứng dụng không cần chờ đợi mẫu truy vấn này kết thúc. Kế tiếp, một vòng lặp `While` được sử dụng để kiểm tra tình trạng của đối tượng `IAsyncResult`. Khi lệnh không đồng bộ hoàn tất, thuộc tính `IsCompleted` được xác lập sang `true`. Vào lúc này, vòng lặp `While` hoàn tất và lệnh `EndExecuteReader` được sử dụng để gán mẫu truy vấn không đồng bộ cho một `SqlDataReader` để xử lý.

Multiple Active Result Sets (MARS)

Khả năng tận dụng Multiple Active Result Sets (MARS) mới của SQL Server 2005 là một cải tiến khác trong phiên bản ADO.NET mới. Trong các phiên bản trước của ADO.NET và SQL Server, bạn bị giới hạn trong một tập hợp kết quả hoạt động trên mỗi nối kết. Và tuy COM-based ADO và OLE DB có một tính năng cho phép ứng dụng xử lý nhiều tập hợp kết quả nhưng thực chất tính năng đó đang tạo ra các kết nối mới nhân danh bạn để xử lý các lệnh bổ sung. Tính năng MARS mới trong ADO.NET tận dụng khả năng có nhiều lệnh hoạt động trên một kết nối đơn của SQL Server 2005. Trong mô hình này, bạn có thể mở một kết nối với cơ sở dữ liệu, mở lệnh đầu tiên và xử lý một số kết quả, mở lệnh thứ hai và xử lý kết quả, rồi trở lại lệnh đầu tiên và xử lý nhiều kết quả hơn. Bạn có thể tự do chuyển đổi tới lui giữa các lệnh hoạt động khác nhau. Không có sự cản trở nào giữa các lệnh và cả hai lệnh chia sẻ một kết nối đơn với cơ sở dữ liệu. Tính năng này đem lại một hiệu suất lớn và đạt được khả năng mở rộng cho các ứng dụng ADO.NET 2.0. Vì tính năng dựa vào một cơ sở dữ liệu SQL Server 2005 nên nó được sử dụng chỉ với các cơ sở dữ liệu SQL Server 2005 và không có tác dụng với các phiên bản trước của SQL Server. Ví dụ sau đây minh họa việc sử dụng MARS:

```
SqlConnection cn = new SqlConnection
    ("SERVER=TECA-YUKON;INTEGRATED SECURITY=True;"
    + "DATABASE=AdventureWorks");
SqlCommand cmd1 =
```

```
new SqlCommand("SELECT * FROM HumanResources.Department", cn);
cmd1.CommandType = CommandType.Text;
try
{
    cn.Open();
    SqlDataReader rdr = cmd1.ExecuteReader();
    while (rdr.Read())
    {
        if (rdr["Name"].ToString() == "Production")
        {
            SqlCommand cmd2 = new SqlCommand
                ("SELECT * FROM HumanResources.Employee "
                + "WHERE DepartmentID = 7", cn);
            cmd2.CommandType = CommandType.Text;
            SqlDataReader rdr2 = cmd2.ExecuteReader();
            while (rdr2.Read())
            {
                // Process results
            }
            rdr2.Close();
        }
    }
    rdr.Close();
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message);
}
finally
{
    cn.Close();
}
```

Trong ví dụ này, bạn có thể nhận thấy cả cmd1 và cmd2 chia sẻ cùng đối tượng SqlConnection, được đặt tên là cn. Đối tượng cmd1 được sử dụng để mở một luồng SqlDataReader vốn đọc tất cả các hàng từ table HumanResources.Department. Khi Department Production được tìm thấy,

đối tượng SqlCommand thứ hai, được đặt tên là cmd2, được sử dụng để đọc các nội dung của table HumanResources.Employee. Điểm quan trọng cần lưu ý là SqlCommand, được gọi là cmd2, có thể thực thi nhờ dùng đối tượng SqlConnection hoạt động vốn cũng đang phục vụ đối tượng cmd1.

Sự phân trang

Hỗ trợ phân trang là một tính năng mới khác trong phiên bản SQL Server 2005 của ADO.NET. Sự phân trang (paging) luôn là một lĩnh vực khó làm việc trong các ứng dụng client, và ADO.NET mới cung cấp sự hỗ trợ cơ bản cho việc phân trang bằng cách cho phép ứng dụng chọn và kết buộc với một dãy hàng từ một tập hợp kết quả. Với khả năng mở rộng, sự phân trang mới không duy trì bất kỳ trạng thái nào trên server. Tuy nhiên, điều này cũng có nghĩa là có khả năng tư cách thành viên của tập hợp phân trang sẽ thay đổi giữa các lần thực thi. Điều này có nghĩa là tính năng phân trang mới phù hợp nhất với dữ liệu khá ổn định và không thay đổi thường xuyên. Sự hỗ trợ phân trang trong ADO.NET mới là dựa trên thứ tự, và để sử dụng nó, bạn phải chỉ định một hàng bắt đầu trong tập hợp kết quả và số hàng cần đưa vào trang. Các hàng của tập hợp trang được đọc nhờ dùng DataReader chuẩn. Thường con PageProductsTable mẫu dưới đây minh họa việc sử dụng hàm phân trang ADO.NET mới:

```
private SqlDataReader PageProductsTable(int nStartRow, int nPageSize)
```

```
{
    SqlConnection cn = new SqlConnection
        ("SERVER=TECA-YUKON;INTEGRATED SECURITY=True;"
        + "DATABASE=AdventureWorks");
    SqlCommand cmd =
        new SqlCommand("SELECT * FROM Production.Product", cn);
    cmd.CommandType = CommandType.Text;
    cn.Open();
    return cmd.ExecuteReader
        (CommandBehavior.Default, nStartRow, nPageSize);
}
```

Trong ví dụ này, phương thức PageProductsTable nhận ở dạng đối số hai số nguyên vốn xác định vị trí bắt đầu và kích cỡ phân trang của các hàng sẽ được đọc. Nó trả về một đối tượng ADO.NET SqlDataReader. Bên trong thường trình, các đối tượng SqlConnection và SqlCommand được tạo như bình thường. Sau khi đối tượng SqlConnection được mở

phương thức `ExecutePageReader` của đối tượng `SqlCommand` được gọi để truy xuất một trang kết quả. Đối số đầu tiên của phương thức `ExecutePageReader` là bộ liệt kê `CommandBehaviorDefault` vốn cho đối tượng `SqlCommand` biết cách xử lý nối kết khi hoạt động đã hoàn tất. Đối số thứ hai là một số thứ tự cho biết hàng bắt đầu. Đối số thứ ba chỉ định số hàng sẽ được trả về. Bạn có thể dùng các khả năng phân trang. Bạn có thể dùng các khả năng phân trang của ADO.NET 2.0, như bạn thấy trong ví dụ dưới đây:

```
DataTable dt = new DataTable("Products");
```

```
dt.Load(PageProductsTable(10, 10));
```

```
dataGridView1.DataSource = dt;
```

Ở đây, một đối tượng `DataTable` mới được tạo, và sau đó phương thức `Load` được sử dụng để dẫn truyền kết quả của `SqlDataReader` đã được phân trang vào `DataTable`. Sau đó, `DataTable` được kết buộc với một đối tượng `dataGridView`.

Chèn khối

Một cải tiến quan trọng khác trong ADO.NET 2.0 là đối tượng `SqlBulkCopy` mới. Đối tượng `SqlBulkCopy` cung cấp một phương thức thực thi cao để chuyển các đối tượng giữa các cơ sở dữ liệu khác nhau hay các hệ thống SQL Server khác nhau. Ví dụ sau đây minh họa cách sử dụng đối tượng `SqlBulkCopy` mới:

```
// Create source & destination connections
```

```
SqlConnection cnSource = new SqlConnection
```

```
("SERVER=TECA-YUKON;INTEGRATED SECURITY=True;"
```

```
+ "DATABASE=AdventureWorks");
```

```
SqlConnection cnDest = new SqlConnection
```

```
("SERVER=TECA-YUKON;INTEGRATED SECURITY=True;"
```

```
+ "DATABASE=AdventureWorks2");
```

```
cnSource.Open();
```

```
cnDest.Open();
```

```
// Read the source data
```

```
SqlCommand cmd = new SqlCommand
```

```
("SELECT * FROM Sales.SpecialOffer", cnSource);
```

```
SqlDataReader rdr = cmd.ExecuteReader();
```



```
// Create SqlBulkCopy object and write the destination data
SqlBulkCopy bulkData = new SqlBulkCopy(cnDest);
bulkData.DestinationTableName = "SpecialOffers";
bulkData.WriteToServer(rdr);
```

```
bulkData.Close();
cnSource.Close();
cnDest.Close();
```

Trong ví dụ này, hai đối tượng nối kết được tạo chỉ đến các cơ sở dữ liệu khác nhau trong cùng hệ thống. Đối tượng nối kết đầu tiên dùng cơ sở dữ liệu AdventureWorks và đối tượng thứ hai dùng một bản sao được gọi là AdventureWorks. Cả hai đối tượng nối kết đều được mở và sau đó một SqlDataReader được sử dụng để đọc dữ liệu từ nối kết nguồn. Tiếp, một đối tượng SqlBulkCopy được tạo và được gán cho đối tượng nối kết đích. Phương thức WriteToServer của đối tượng SqlBulkCopy được gọi nhờ dùng SqlDataReader vốn đã được gán cho nối kết nguồn. Phương thức WriteToServer sao chép dữ liệu từ nguồn sang đích. Cần lưu ý rằng đối tượng đích phải hiện hữu trên nối kết đích. Có các phương thức bổ sung trong đối tượng SqlBulkCopy mà bạn có thể dùng để thực hiện phép ánh xạ schema tùy ý giữa các table nguồn và đích.

Mô hình nối kết chung

Một trong những vấn đề mà ADO.NET 1.0 cũng đã gặp phải là nó không nhận biết provider. Nói cách khác, bạn cần sử dụng một provider nhất định để nối kết với một nền cơ sở dữ liệu đích nhất định. Chẳng hạn, SqlConnection có thể nối kết với chỉ các hệ thống SQL Server, không nối kết với các hệ thống Oracle. Cũng vậy, OracleClient có thể nối kết với chỉ các hệ thống Oracle, không nối kết các hệ thống SQL Server. Tuy bạn có thể tạo mã để tải provider chính xác lúc đang chạy nhưng kết quả thường không thanh lịch và chắc chắn nó không tiện lợi. ADO.NET 2.0 giải quyết vấn đề này bằng việc thêm vào một khả năng Provider Factory mới có thể thể hiện provider thích hợp vào lúc chạy. Ví dụ sau đây minh họa cách sử dụng Provider Factory mới:

```
DbDataReader rdr;
DbProviderFactory provider =
    DbProviderFactories.GetFactory("System.Data.SqlClient");

using (DbConnection cn = provider.CreateConnection())
{
```

```
using (DbCommand cmd = provider.CreateCommand())
{
    cmd.CommandText = "SELECT * FROM Production.Location";
    cmd.Connection = cn;
    cn.ConnectionString =
        ("SERVER=TECA-YUKON;INTEGRATED SECURITY=True;"
        + "DATABASE=AdventureWorks");
    cn.Open();
    rdr = cmd.ExecuteReader(CommandBehavior.CloseConnection);
    DataTable dt = new DataTable("Product Locations");
    dt.Load(rdr);
    dataGridView1.DataSource = dt;
}
}
```

Ở đây, bạn có thể thấy phương thức `GetFactory` được sử dụng để tạo một thể hiện của dữ liệu `System.Data.SqlClient` được cung cấp vào lúc chạy. Sau đó, một đối tượng `DbCommand` được sử dụng để thực thi một lệnh nhằm truy xuất nội dung của table `Production.Location`, vốn được chuyển đến `DbDataReader`. Sau cùng, các kết quả của `DbDataReader` được tải đến một `DataTable` vốn được kết buộc với một đối tượng `dataGridView`.

Lưu ý

Bạn cần đưa vào namespace `System.Data.Common` để sử dụng các đối tượng `DbProvider`.

CHƯƠNG 5

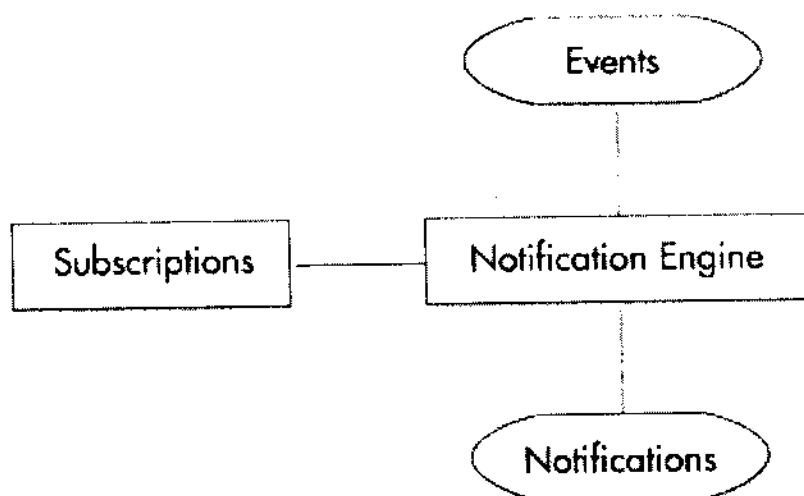
Các dịch vụ thông báo

Các dịch vụ thông báo (Notification Services) được giới thiệu lần đầu tiên vào năm 2002 như là một web download cho SQL Server 2005. Với SQL Server 2005, Notification Services đã được thêm vào như là một hệ thống phụ mới cho sản phẩm cơ sở. Notification Services là một framework cho phép bạn phát triển các ứng dụng gửi thông báo tùy ý vốn có khả năng đẩy thông tin đã tùy biến đến nhiều thuê bao và thiết bị. Notification Services được sử dụng bởi nhiều công ty nổi tiếng nhằm chuyển phát thông tin cá nhân đến khách hàng. Chẳng hạn, MSN mobile dùng Notification Services để gửi tin tức đã cá nhân hóa và những thông tin khác đến nhiều thiết bị di động. Một phần thực thi Notification Services nổi tiếng khác có thể được tìm thấy tại ESPN.com, vốn cung cấp một dịch vụ mà các thuê bao có thể dùng để tìm các tỉ số thể thao. Notification Services của SQL Server 2005 cho phép bạn tạo cùng các loại ứng dụng kiểu thông báo này. Trong khi bạn có thể xây dựng những ứng dụng thông báo từ đầu, Notification Services của SQL Server 2005 cho bạn một sự khởi đầu vững chắc về loại dự án này bằng cách đưa ra một framework mạnh, có khả năng mở rộng và đã kiểm tra mà bạn có thể dùng làm cơ sở cho các ứng dụng thông báo riêng của bạn.

Trong chương này, bạn sẽ tìm hiểu về hệ thống phụ Notification Services mới. Trong phần đầu tiên của chương này, bạn sẽ xem tổng quan về hệ thống phụ mới này. Kế tiếp, bạn học cách xây dựng các ứng dụng Notification Services. Để bắt đầu, bạn sẽ xem qua tiến trình phát triển và sau đó xem một số mã mẫu minh họa cách sử dụng framework Notification Services để tạo các ứng dụng kiểu thông báo.

Tổng quan về Notification Services

Một ứng dụng Notification Services là một lớp phần mềm nằm giữa một nguồn thông tin và người nhận có chủ đích của thông tin đó. Ứng dụng Notification Services phản ánh các event nhất định đã có, đồng thời lọc và truyền thông tin một cách thông minh về những event đó đến nhiều loại thiết bị đích khác nhau nhờ dùng một thời biểu chuyển phát cá nhân hóa. Ứng dụng Notification Services gồm có ba thành phần cơ bản: event (sự kiện), subscriptions (đăng ký, thuê bao), và notifications (thông báo). Hình 5.1 minh họa tổng quan cấp cao của một ứng dụng Notification Services.



Hình 5.1: Tổng quan về Notification Services

Events

Trong một ứng dụng Notification Services, event (sự kiện) chỉ là những điều đang xảy ra mà bạn muốn được thông báo về. Trong trường hợp của ví dụ ESPN.com, một event có thể là một điều gì đó giống như tỷ số sau cùng của một sự kiện thể thao. Với NASDAQ, một event có thể là một giá chứng khoán cho trước tăng lên một mức độ nhất định nào đó. Với Notification Services của SQL Server 2005, các event được chứa ở dạng các hàng trong một table trong cơ sở dữ liệu Notification Services.

Subscriptions

Subscriptions là cách những người dùng của một ứng dụng Notification Services thông báo cho hệ thống biết về loại event mà họ quan tâm. Chẳng hạn, với ví dụ Notification Services về thể thao, một người dùng có thể tạo một subscriptions để lấy tỉ số cuối của các môn Los Angeles Lakers. Trong ví dụ Notification Services về giá chứng khoán, một người dùng có thể tạo một subscription để được thông báo khi nào giá chứng khoán Microsoft tăng quá 50\$ mỗi cổ phần. Notification Services của SQL Server 2005 chứa các subscription, giống như các event, ở dạng các hàng trong một table.

Notifications

Về cơ bản, một notification là một thông báo sẽ được gửi đến người dùng cuối vốn có thông tin về event mà người dùng đã đăng ký. Với ví dụ các dịch vụ thông báo ESPN, đây có thể là một thông báo e-mail cho

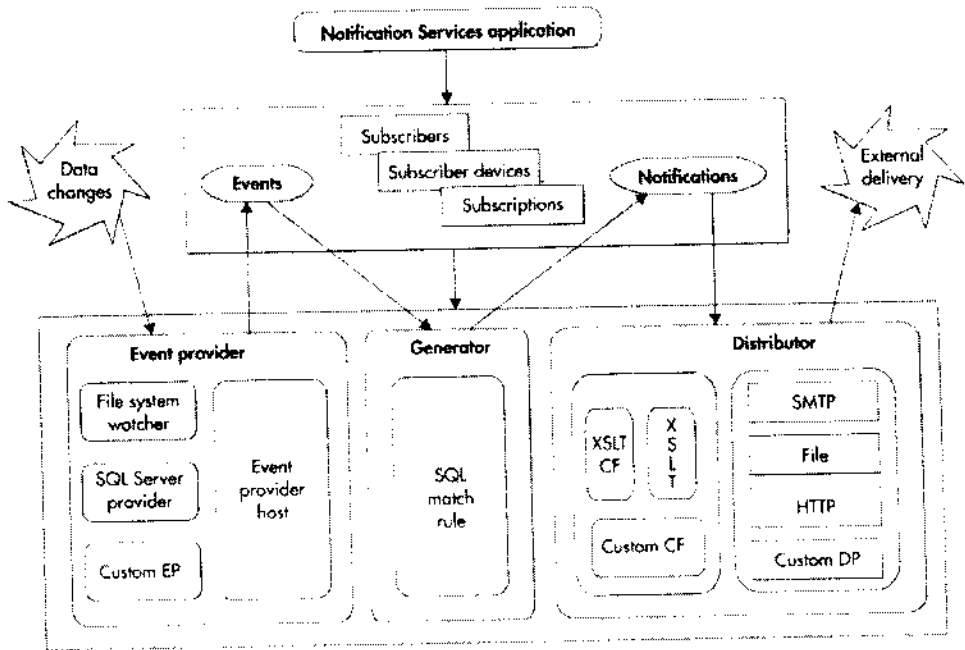
biết rằng đội bạn ưa thích đã đánh bại đối thủ và tỉ số cuối cùng là bao nhiêu. Notifications có thể được chuyển phát ở nhiều dạng khác nhau đến nhiều loại thiết bị đích khác nhau.

Notification Engine

Công việc của ứng dụng Notification Services là giám sát các event ngoài để tìm sự tương kết giữa các event và các subscription đã được đăng ký. Khi một event tương kết với một subscription, bộ máy Notification Services gửi một notification (thông báo) đến người dùng cuối. Khả năng mở rộng của một ứng dụng Notification Services phụ thuộc phần lớn vào việc bộ máy Notification Services tương kết các event (sự kiện) với các subscription (thuê bao) tốt như thế nào. Microsoft đã thiết kế framework Notification Services nền tảng để có thể mở rộng ở cấp độ Internet, nghĩa là với nền thích hợp, Notification Services của SQL Server 2005 có thể mở rộng để xử lý hàng triệu event, subscription, và notification. Để làm điều đó, Notification Services tận dụng bộ máy cơ sở dữ liệu quan hệ cực kỳ hiệu quả của SQL Server 2005 để nối các hàng từ table events với các hàng trong table subscription nhằm tương kết các event với subscription.

Kiểu kiến trúc của Notification Services

Notification Services của SQL Server 2005 là một nền được thiết kế nhằm cho phép bạn phát triển và triển khai các ứng dụng thông báo có khả năng mở rộng cao. Các ứng dụng SQL Server 2005 Notification Services được tạo nhờ dùng T-SQL và XML, chúng được thực thi bằng cách sử dụng .NET Framework hợp nhất đã được cài vào SQL Server 2005. Có cả hai phiên bản 32 bit và 64 bit của nền Notification Services. Bạn có thể xem qua kiểu kiến trúc của SQL Server 2005 Notification Services ở hình 5.2.



Hình 5.2: Tổng quan về kiểu kiến trúc Notification Services

Ở đầu hình 5.2, bạn thấy Notification Services application (ứng dụng Notification Services). Ứng dụng này được viết bởi nhà phát triển; nhiệm vụ chính của nó là thêm thông tin đăng ký vào table subscriptions của Notification Services. Ứng dụng này có thể là một ứng dụng web ASP.NET, một ứng dụng Win32 Windows chuẩn, hay bất kỳ loại ứng dụng có khả năng gọi các API .NET đã được quản lý hoặc trực tiếp hoặc thông qua các lớp .NET COM Interop.

Ở phía bên trái của hình 5.2, bạn thấy phần event của kiểu kiến trúc SQL Server 2005 Notification Services. Các event được thêm vào hệ thống bởi các event provider (bộ cung cấp sự kiện). Các event provider này giám sát các nguồn dữ liệu ngoài đang mong đợi sự thay đổi. SQL Server 2005 có kèm theo hai event provider cài sẵn: một file system watcher (bộ xem hệ thống file), có khả năng phát hiện các thay đổi trong các file hệ điều hành, và SQL Server event provider, giám sát các thay đổi trong một cơ sở dữ liệu SQL Server or Analysis Services. Notification Services cũng có thể được mở rộng để xem các nguồn dữ liệu khác thông qua việc sử dụng các event provider tùy biến. Khi một event provider bắt gặp một sự thay đổi dữ liệu (data change), nó ghi lại event đó trong table events của cơ sở dữ liệu Notification Services.

Bộ tạo (hay bộ máy) Notification Services, được minh họa ở chính giữa hình 5.2, dùng phần logic mà nhà phát triển Notification Services cung cấp để tương kết các event được chứa trong các event provider với

các subscription do những người dùng cuối của ứng dụng nhập vào. Khi một event tương kết với một subscription, bộ tạo (Generator) thêm một hàng có chứa event và thông tin thuê bao vào table notification của cơ sở dữ liệu Notification Services.

Distributor (bộ phân phối), được minh họa ở phía dưới bên phải của hình 5.2, giám sát table notifications để tìm các mục nhập mới. Khi bộ tạo thêm một mục nhập mới vào table notifications, bộ phân phối truy xuất mục nhập đó và định dạng nó để chuyển phát. Theo mặc định, bộ phân phối sẽ định dạng thông báo nhờ dùng một style sheet XML (XSLT) vốn mô phỏng kết xuất theo một cách tương thích với thiết bị xuất đã chỉ định. Ngay khi thông báo đã được định dạng, Distributor sẽ chuyển nó sang provider phân phối để chuyển phát đến người dùng đích. SQL Server 2005 có kèm theo ba provider phân phối cài sẵn: một provider SMTP, một provider File; và một provider HTTP vốn có thể được sử dụng cho SOAP, .NET Alerts, và SMS notifications. Provider SMTP chuyển phát các thông báo qua một e-mail server chẳng hạn như Exchange. Provider File chuyển phát thông báo ở dạng một file hệ điều hành. Provider HTTP chuyển phát các thông báo ở dạng một trang web HTML hoặc dùng một trong các giao thức khác liên quan đến HTTP. Ngoài ra, bạn cũng có thể mở rộng các khả năng chuyển phát của bộ phân phối bằng cách thêm vào một provider phân phối tùy ý.

Phát triển các ứng dụng Notification Services

Trong phần đầu của chương này, bạn đã xem tổng quan về SQL Server 2005 Notification Services mới. Trong mục kế tiếp này, bạn sẽ tìm hiểu sâu hơn về các bước thật sự liên quan đến việc phát triển các ứng dụng SQL Server 2005 Notification Services. Trước tiên, bạn sẽ xem qua tiến trình phát triển và sau đó bắt tay vào tạo một ứng dụng Notification Services đơn giản nhất.

Các bước phát triển

Tiến trình để phát triển các ứng dụng Notification Services bắt đầu với việc định nghĩa schema và các quy tắc điều khiển cách hoạt động của ứng dụng. Kế tiếp, bạn phải biên dịch ứng dụng. Sau đó, bạn cần tạo một giao diện cho phép người dùng thêm các subscription vào ứng dụng. Sau cùng, bạn cần thêm bất kỳ thành phần tùy biến mà ứng dụng có thể cần. Sau đây, chúng ta sẽ xem xét mỗi bước này một cách chi tiết hơn.

Định nghĩa Schema và vạch ra các quy tắc

Nhà phát triển Notification Services sử dụng kết hợp XML và T-SQL để định nghĩa schema và vạch ra các quy tắc của ứng dụng. Khi bạ

định nghĩa schema và vạch ra các quy tắc cho một ứng dụng Notification Services, về cơ bản bạn đang mô tả các event mà ứng dụng sẽ giám sát cũng như các subscription của ứng dụng, các notification (thông báo) của nó, và phần logic sẽ được sử dụng để tương kết các event với các subscription. Các quy tắc và schema của ứng dụng Notification Services được xác định chủ yếu trong hai file: `adf.xml` và `config.xml`. Bạn có thể tạo những file này nhờ dùng một bộ hiệu chỉnh text chuẩn hay một bộ hiệu chỉnh nhận biết XML chẳng hạn như Visual Studio 2005 hay XMLSpy. Các thông tin chi tiết hơn về nội dung cụ thể của các file `adf.xml` và `config.xml` được trình bày ở sau trong chương này.

Biên dịch ứng dụng

Sau khi đã tạo schema và các quy tắc, bước kế tiếp trong tiến trình tạo một ứng dụng Notification Services là biên dịch toàn bộ mã và đăng ký một dịch vụ mà nó sẽ chạy các ứng dụng Notification Services. Để biên dịch ứng dụng, SQL Server 2005 cung cấp tiện ích dòng lệnh `nscontrol`, tiện ích này dùng để tạo, đăng ký và cũng cập nhật ứng dụng Notification Services. Ngoài ra, SQL Server Management Studio cung cấp một hộp thoại cho phép bạn tạo các ứng dụng Notification Services một cách tương tác.

Tạo ứng dụng quản lý sự đăng ký thông tin

Hai bước đầu tiên tạo bộ máy cốt lõi của ứng dụng Notification Services. Tuy nhiên, người dùng vẫn cần một cách nào đó để thêm thông tin đăng ký của riêng họ vào ứng dụng Notification Services. Để cho phép người dùng nhập thông tin đăng ký của họ, ứng dụng Notification Services cần một giao diện quản lý sự đăng ký, thường là một ứng dụng Web hay GUI được tạo nhờ dùng các công nghệ ASP.NET, VB.NET hay C# kết hợp với một nền phát triển như Visual Studio 2005.

Thêm các thành phần tùy biến

Bước cuối cùng trong tiến trình tạo ứng dụng Notification Services là tùy ý thêm vào bất kỳ thành phần mà ứng dụng có thể cần. Các thành phần tùy biến sẽ bao gồm bất kỳ event provider đã yêu cầu, các dạng nội dung, hay các giao thức chuyển phát thông báo vốn không có trong sản phẩm cơ sở SQL Server 2005 Notification Services.

Ứng dụng Notification Services mẫu

Ứng dụng Notification Services mẫu được trình bày trong phần kế tiếp của chương này giám sát giá trị của một cột trong một table SQL Server. Khi giá trị cột đó vượt quá một ngưỡng nhất định, ứng dụng Notification Services mẫu sẽ gửi một thông báo e-mail đến người dùng cuối (end user). Mẫu này là một dạng đơn giản của ví dụ Stock đã được

kèm theo với các chương trình mẫu SQL Server 2000 Notification Services. Để làm cho mọi thứ hoạt động, ứng dụng bắt đầu bằng cách sử dụng provider SQL Server 2005 để kiểm tra một table trong cơ sở dữ liệu. Một event cần được tạo để kiểm tra giá trị của một cột trong table đó, người dùng phải nhập một nội dung đăng ký cho event đó và một quy tắc phải được thêm vào nhằm cho phép SQL Server tương kết các event với các nội dung đăng ký (subscription). Khi một event tương kết với quy tắc event, provider phân phối e-mail sẽ gửi một thông báo dạng XML đến người dùng cuối. Bạn đã xem qua ứng dụng Notification Services mẫu, bây giờ hãy xem cách tạo nó.

Tạo các File định nghĩa và cấu hình ứng dụng

Các ứng dụng Notification Services gồm có hai file chính: một file định nghĩa ứng dụng (application definition file, gọi tắt là ADF) và một file cấu hình ứng dụng (application configuration file, gọi tắt là ACF) - cả hai file XML phải được cài vào cùng với các schema xsd của chúng. Các schema XSD có tác dụng bảo đảm cả hai tài liệu đều sở hữu thành phần và thuộc tính được yêu cầu. File ACF định rõ tên của ứng dụng Notification Services cũng như tên thể hiện (instance) của nó và đường dẫn thư mục. Tên instance về cơ bản là tên của một dịch vụ Windows chạy ứng dụng Notification Services.

File ADF là file cốt lõi dành cho Notification Services; các mục khác nhau của ADF mô tả event, subscription, các quy tắc (rules), và cấu trúc thông báo sẽ được sử dụng bởi ứng dụng Notification Services. Với ấn bản SQL Server 2000, bạn cần làm mọi việc tạo mã XML bằng tay để tạo hai file này. Điều này cũng đúng với các phiên bản beta đầu tiên của SQL Server 2005. Tuy nhiên, ấn bản SQL Server 2005 sau cùng sẽ có các công cụ đồ họa cho phép bạn tạo các file ACF và ADF mà không phải tạo mã XML bằng tay trong một bộ hiệu chỉnh text.

Ngoài hai file chính này vốn định rõ ứng dụng Notification Services và kiểu cấu trúc, một ứng dụng Notification Services cũng cần một ứng dụng quản lý sự đăng ký chịu trách nhiệm thêm thông tin đăng ký vào cơ sở dữ liệu của ứng dụng Notification Services. Trong khi ADF và ACF được tạo chủ yếu nhờ dùng XML và T-SQL, ứng dụng quản lý sự đăng ký (subscription management application) thường được tạo nhờ dùng một ngôn ngữ cấp cao với một giao diện người dùng đồ họa.

••••• Thủ thuật

Cấu trúc giải pháp gồm nhiều dự án của Visual Studio .NET cho phép bạn dễ dàng nhóm cả các file định nghĩa XML và mã quản lý sự đăng ký ở dạng hai dự án vốn là thành phần của một giải pháp chung.

Danh sách sau đây minh họa file ACF, config.xml, được sử dụng trong ứng dụng Notification Services mẫu của chương này:

```
<?xml version="1.0" encoding="utf-8"?>
<NotificationServicesInstance xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://www.microsoft.com/MicrosoftNotificationServices
/ConfigurationFileSchema">
  <InstanceName>NSSampleInstance</InstanceName>
  <SqlServerSystem>tecaayukon</SqlServerSystem>
  <Applications>
    <Application>
      <ApplicationName>NSSample</ApplicationName>
      <BaseDirectoryPath>C:\NSSample</BaseDirectoryPath>
      <ApplicationDefinitionFilePath>ADF.xml
      </ApplicationDefinitionFilePath>
    </Application>
  </Applications>
  <DeliveryChannels>
    <DeliveryChannel>
      <DeliveryChannelName>EmailChannel
      </DeliveryChannelName>
      <ProtocolName>SMTP</ProtocolName>
    </DeliveryChannel>
  </DeliveryChannels>
</NotificationServicesInstance>
```

Bạn có thể thấy ACF là một tài liệu tương đối đơn giản. File này có thể được tạo nhờ dùng bất kỳ bộ hiệu chỉnh text hay bộ hiệu chỉnh nhận biết XML. Các điểm quan trọng nhất cần lưu ý là các thẻ gán SqlServerSystem, InstanceName, ApplicationName, baseDirectoryPath, và ApplicationDefinitionFilePath. Như bạn có thể đoán được, thẻ gán tên SqlServerSystem chứa tên của hệ thống SQL Server sẽ điều khiển các cơ sở dữ liệu Notification Services, thẻ gán InstanceName định rõ tên instance cho ứng dụng và thẻ gán ApplicationName định rõ tên của ứng dụng Notification Services. BaseDirectoryPath cho trình biên dịch biết nơi để tìm file ADF và thẻ gán ApplicationDefinitionFilePath cung cấp tên của tài liệu XML có chứa mã ADF. Ngoài các thành phần cơ bản này, ACF cũng sử dụng thẻ gán DeliveryChannel để định rõ các thông

báo sẽ được chuyển phát như thế nào. Trong ví dụ này, thẻ gán DeliveryChannel dùng giao thức SMTP để chuyển phát các thông báo e-mail.

Định nghĩa Event Schema ADF

Các định nghĩa cốt lõi điều khiển cách hoạt động của một ứng dụng Notification Services được tìm thấy trong ADF. Điều đầu tiên cần được thực hiện để tạo ứng dụng mẫu là tạo schema cho các event. Trong file ADF, thành phần EventClasses chứa mã XML dùng để định nghĩa các event Notification Services. Thành phần EventClasses có thể chứa nhiều định nghĩa event. Mỗi định nghĩa event được mô tả trong một thành phần con EventClass riêng biệt. Đoạn mã sau đây từ ví dụ file adf.xml minh họa mã XML dùng để tạo mẫu này:

```
<?xml version="1.0" encoding="utf-8" ?>
<Application xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://www.microsoft.com/MicrosoftNotificationServices/
  ApplicationDefinitionFileSchema">
  <!-- Describe the Events -->
  <EventClasses>
    <EventClass>
      <EventClassName>DataEvents</EventClassName>
      <Schema>
        <Field>
          <FieldName>DataID</FieldName>
          <FieldType>int</FieldType>
          <FieldTypeMods>not null</FieldTypeMods>
        </Field>
        <Field>
          <FieldName>DataValue</FieldName>
          <FieldType>int</FieldType>
          <FieldTypeMods>null</FieldTypeMods>
        </Field>
      </Schema>
      <IndexSqlSchema>
        <SqlStatement>CREATE INDEX DataEventsIndex
          ON DataEvents ( DataID )</SqlStatement>
      </IndexSqlSchema>
```

```

<Chronicles>
  <Chronicle>
    <ChronicleName>DataEventsTable</ChronicleName>
    <SqlSchema>
      <SqlStatement>
        IF EXISTS(SELECT name FROM dbo.sysobjects
          WHERE name = 'DataEventsTable')
        DROP TABLE dbo.DataEventsTable
        CREATE TABLE DataEventsTable
        (
          [DataID] int,
          [DataValue] int
        )
      </SqlStatement>
    </SqlSchema>
  </Chronicle>
</Chronicles>
</EventClass>
</EventClasses>

```

Tất cả các file ADF phải bắt đầu với các thành phần ứng dụng mà như bạn có thể dự đoán, chúng tiêu biểu cho ứng dụng Notification Services. Các thành phần chính bên trong thành phần định nghĩa ứng dụng là EventClasses, SubscriptionClasses, và NotificationClasses. Danh sách mã ở trên chứa mục EventClasses của ADF. Bởi vì ứng dụng mẫu này sử dụng chỉ một event nên thành phần EventClasses chứa chỉ một thành phần EventClass, được gọi là DataEvents. Một Schema bên trong thành phần EventClass định nghĩa event schema mà ứng dụng Notification Services sẽ giám sát. Trong trường hợp này, hai cột được xác định: cột DataID - một bộ định danh đơn giản, và cột DataValue chứa một giá trị liên quan. Sự giống nhau của hai định nghĩa này với các cột SQL Server là không ngẫu nhiên. Notification Services dùng các định nghĩa này để tạo các cột trong cơ sở dữ liệu Notification Services. Để bảo đảm sự thực thi tốt, thẻ gán IndexSqlSchema được sử dụng để tạo một index (chỉ mục) qua cột DataID.

Mục Chronicles định rõ các table chứa dữ liệu event để các thuê bao sử dụng theo thời biểu. Mục Chronicles có hai thẻ gán bắt buộc: thẻ gán ChronicleName và thẻ gán SqlSchema. Thẻ gán ChronicleName cung cấp tên của table Chronicle và thẻ gán SqlSchema chứa các câu lệnh T-SQL để tạo table. Bởi vì mã này được sử dụng trong cả hai tiến trình

tạo và cập nhật Notification Services nên trước tiên nó phải kiểm tra sự tồn tại của table và loại bỏ nó nếu nó được tìm thấy. Dòng kế tiếp của mã T-SQL sau đó sẽ tạo table.

Định rõ các Provider ADF

Sau khi định nghĩa các event mà ứng dụng sẽ giám sát, bước kế tiếp trong tiến trình tạo ứng dụng Notification Services là định rõ provider nào sẽ chuyển phát các event đó đến ứng dụng. Trong phần mã sau đây từ mục kế tiếp của file adf.xml, bạn có thể thấy phần định rõ event SQL Server được sử dụng để nối kết ứng dụng Notification Services với một SQL Server:

```
<!-- Specify the SQL Server Event Provider -->
<Providers>
  <HostedProvider>
    <ProviderName>SQLData</ProviderName>
    <ClassName>SQLProvider</ClassName>
    <SystemName>tecayukon</SystemName>
    <Schedule>
      <Interval>P0DT00H00M60S</Interval>
    </Schedule>
    <Arguments>
      <Argument>
        <Name>EventsQuery</Name>
        <Value>SELECT DataID, DataValue FROM DataEventsTable</Value>
      </Argument>
      <Argument>
        <Name>EventClassName</Name>
        <Value>DataEvents</Value>
      </Argument>
    </Arguments>
  </HostedProvider>
</Providers>
```

Mục Providers của ADF mô tả các event provider được sử dụng bởi ứng dụng Notification Services. Trong ví dụ này, thành phần HostedProvider định rõ event provider SQL Server. Thành phần ProviderName được sử dụng để gán một tên cho provider và thành phần SystemName cung cấp tên của hệ thống SQL Server mà provider sẽ nối kết với.

Thành phần Schedule định rõ provider sẽ nối kết với hệ thống thường xuyên như thế nào; điều này được điều khiển bởi trị xác định trong thành phần Interval. Trị trong thành phần Interval dùng kiểu dữ liệu thời hạn XML. Phần ODT của trị này tiêu biểu cho một khoảng ngày tháng với một trị là 0. Phần 00HR tiêu biểu cho một khoảng giờ với một trị là 0. Phần 00M tiêu biểu cho một khoảng phút với một trị là 0. Phần 60S tiêu biểu cho một khoảng giây với một trị là 60. Nói cách khác, khoảng này được xác lập tại 60 giây.

Thành phần Arguments cung cấp mẫu truy vấn sẽ được sử dụng để trích dữ liệu từ nguồn event. Trong ví dụ này, các nội dung của cột DataID và DataValue trong DataEventsTable sẽ được truy xuất 60 một lần cho lớp event DataEvents vốn đã được định rõ trong thành phần EventClass trước đây.

Định rõ Schema đăng ký ADF

Ngay khi các event đã được mô tả, bước kế tiếp trong tiến trình tạo file ADF là định rõ các phần đăng ký. Danh sách mã sau đây minh họa phần kế tiếp của file adf.xml mô tả các phần đăng ký (subscription) được sử dụng bởi ứng dụng Notification Services mẫu:

```
<!-- Describe the Subscription -->
<SubscriptionClasses>
  <SubscriptionClass>
    <SubscriptionClassName>DataSubs</SubscriptionClassName>
    <Schema>
      <Field>
        <FieldName>DeviceName</FieldName>
        <FieldType>nvarchar(255)</FieldType>
        <FieldTypeMods>not null</FieldTypeMods>
      </Field>
      <Field>
        <FieldName>SubLocale</FieldName>
        <FieldType>nvarchar(10)</FieldType>
        <FieldTypeMods>not null</FieldTypeMods>
      </Field>
      <Field>
        <FieldName>DataID</FieldName>
        <FieldType>int</FieldType>
        <FieldTypeMods>not null</FieldTypeMods>
      </Field>
```

```

<Field>
  <FieldName>DataTriggerValue</FieldName>
  <FieldType>int</FieldType>
  <FieldTypeMods>not null</FieldTypeMods>
</Field>
</Schema>
<IndexSqlSchema>
  <SqlStatement>CREATE INDEX DataSubIndex ON DataSubs
    ( DataID )</SqlStatement>
</IndexSqlSchema>
<EventRules>
  <EventRule>
    <RuleName>DataSubEventRule</RuleName>
    <Action>
      SELECT dbo.DataNotificationsNotify
        (s.DeviceName, s.SubLocale, e.DataID, e.DataValue)
      FROM DataSubs s JOIN DataEvents e
        ON s.DataID = e.DataID
      LEFT OUTER JOIN DataEventsTable t
        ON s.DataID = t.DataID
      WHERE s.DataTriggerValue &lt;= e.DataValue
        AND (s.DataTriggerValue &gt; t.DataValue
          OR .DataValue IS NULL)

      INSERT INTO DataEventsTable (DataID, DataValue)
      SELECT e.DataID, e.DataValue
        FROM DataEvents e
        WHERE e.DataID NOT IN
          (SELECT DataID from DataEventsTable)

      UPDATE DataEventsTable
        SET DataValue = e.DataValue
        FROM DataEvents e, DataEventsTable t
        WHERE e.DataID = t.DataID
          AND e.DataValue &gt; t.DataValue
    </Action>
  </EventRule>
</EventRules>

```

```

<EventClassName>DataEvents</EventClassName>
</EventRule>
</EventRules>
</SubscriptionClass>
</SubscriptionClasses>

```

Giống như EventClass, mục SubscriptionClasses của tài liệu ADF có thể mô tả nhiều phần đăng ký, trong đó mỗi phần đăng ký được mô tả trong một thành phần SubscriptionClass riêng biệt. Ví dụ này dùng một SubscriptionClass có tên DataSubs. Mục Schema mô tả nội dung đăng ký. Trường DeviceName nhận biết loại thiết bị đích. SubLocale được sử dụng để tùy ý thay đổi ngôn ngữ mà người đăng ký hay thuê bao sẽ sử dụng để nhận thông báo. Các trường DataID và DataTriggerValue nhận dạng event sẽ được đăng ký với và trị dữ liệu sẽ kích khởi một thông báo. Một lần nữa, giống như EventClass, thành phần IndexSqlSchema được sử dụng để tạo một index trên cột DataID để đạt được sự thực thi tốt hơn. Notification Services dùng các phần mô tả này để tạo các cột cơ sở dữ liệu trên SQL Server.

Sau khi đã cài đặt các phần đăng ký, mục kế tiếp của mã trong thành phần EventRules định nghĩa logic mà ứng dụng Notification Services sẽ dùng để tương kết các event với các phần đăng ký (subscription). Trong khi thông tin Event and Subscription được định rõ nhờ dùng XML, các quy tắc event được tạo nhờ dùng mã T-SQL chứa trong thành phần EventRules Action. Trong ví dụ này, điều quan trọng nhất cần lưu ý là table DataSubs được nối với table DataEvents nơi DataValue từ table DataEvents lớn hơn DataTriggerValue nhưng lại thấp hơn một trị mới. Khi thỏa điều kiện nối, một hàng dành cho người dùng đăng ký hay thuê bao (subscriber) sẽ được tạo. Các câu lệnh INSERT và UPDATE được sử dụng để cập nhật table Chronicle khi không tìm thấy một trị hay khi một trị lớn hơn trị chronicle hiện có.

Schema thông báo ADF

Phần kế tiếp của định nghĩa file ADF là cài đặt schema NotificationClasses. NotificationClasses mô tả cách thông tin notification (thông báo) sẽ được chuyển phát. Danh sách mã sau đây từ mục cuối của file adf.xml có chứa định nghĩa NotificationClass. Thành phần NotificationClasses có thể mô tả nhiều loại thông báo mà mỗi loại sẽ được mô tả trong thành phần NotificationClass riêng của nó. Bởi vì ứng dụng mẫu này sử dụng chỉ một loại thông báo, nên mục NotificationClasses chứa một thành phần NotificationClass.

```

<!-- Describes the Notifications -->
<NotificationClasses>

```



```

<NotificationClass>
  <NotificationClassName>DataNotifications</NotificationClassName>
  <Schema>
    <Fields>
      <Field>
        <FieldName>DataID</FieldName>
        <FieldType>int</FieldType>
      </Field>
      <Field>
        <FieldName>DataValue</FieldName>
        <FieldType>int</FieldType>
      </Field>
    </Fields>
  </Schema>
  <!-- Specify the Content Format XSLT -->
  <ContentFormatter>
    <ClassName>XsltFormatter</ClassName>
    <Arguments>
      <Argument>
        <Name>XsltBaseDirectoryPath</Name>
        <Value>C:\NSSample</Value>
      </Argument>
      <Argument>
        <Name>XsltFileName</Name>
        <Value>NSSample.xslt</Value>
      </Argument>
    </Arguments>
  </ContentFormatter>
</NotificationClass>
</NotificationClasses>
<Generator>
  <SystemName>tegayukon</SystemName>
</Generator>
<Distributors>
  <Distributor>
    <SystemName>tegayukon</SystemName>

```

```

</Distributor>
</Distributors>
<ApplicationExecutionSettings>
  <PerformanceQueryInterval>PT5S</PerformanceQueryInterval>
</ApplicationExecutionSettings>

</Application>

```

Trong danh sách này, bạn có thể thấy lớp thông báo được đặt tên là `DataNotification`. Thành phần Schema của các lớp `DataNotification` xác định thông tin sẽ được gửi đến người đăng ký. Ở đây bạn có thể thấy của các trường `DataID` và `DataValue` sẽ được gửi như là thành phần của thông báo.

Thành phần `ContentFormatter` xác định cách thông báo sẽ được định dạng khi nó được gửi đến người đăng ký. Ví dụ này minh họa việc sử dụng `XSLTFormatter` cài sẵn. Thành phần `Arguments` mô tả thư mục mà file `XSLT` được tìm thấy cũng như tên của file. Trong danh sách này, bạn thấy file `XSLT` được nằm trong thư mục `C:\NSSample` và được đặt tên là `NSSample.xslt`.

Các thành phần `Generator`, `Distributor`, và `ApplicationExecutionSettings` chỉ định hệ thống `SQL Server` sẽ được sử dụng để tạo các thông báo, hệ thống sẽ được sử dụng để phân phối các thông báo và khoảng thời gian mà các bộ đếm sự thực thi của hệ thống sẽ được cập nhật,

Định dạng kết xuất thông báo

Trong danh sách trước, bạn thấy thông báo đã được định dạng nhờ dùng style sheet `NSSample.xslt`. Bạn có thể thấy style sheet mẫu đó có định mạo như thế nào trong danh sách dưới đây:

```

<?xml version="1.0" encoding="UTF-8" ?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="notifications">
    <HTML>
    <BODY>
      <xsl:apply-templates />
      <l>This message was generated using
      <BR/>Microsoft SQL Server Notification Services</l><BR/><BR/>
    </BODY>

```

```

</HTML>
</xsl:template>
<xsl:template match="notification">
  <P>The data value of <B><xsl:value-of select="DataID"/></B>
  <BR/>reached the value <B><xsl:value-of select="DataValue"/></B>
</P>
</xsl:template>
</xsl:stylesheet>

```

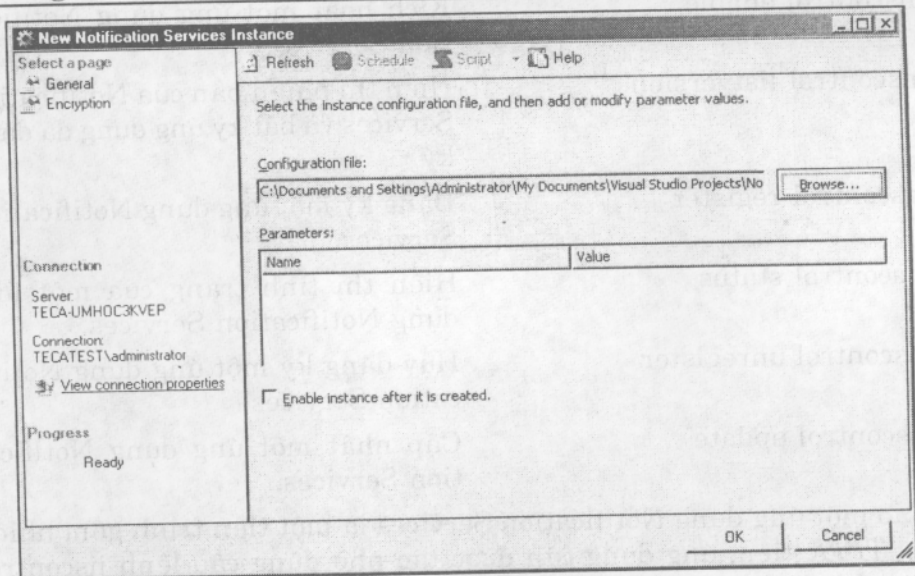
Style sheet dùng để định dạng kết xuất của ứng dụng Notification Services là một style sheet HTML chuẩn. Trong mục template, bạn thấy nơi các trường DataID và DataValue từ NotificationClass được hiển thị trong thông báo.

Tạo ứng dụng Notification Services

Sau khi đã tạo mã ứng dụng XML và T-SQL theo yêu cầu, bạn chuẩn bị tạo ứng dụng Notification Services. Các ứng dụng Notification Services được tạo một cách tương tác nhờ dùng SQL Server Management Studio hoặc chúng có thể được tạo nhờ dùng tiện ích nscontrol.

Tạo các ứng dụng Notification Services bằng cách sử dụng SQL Server Management Studio

Sau khi đã tạo các file config.xml và adf.xml vốn định rõ Notification Services, bạn có thể dùng chúng để tạo ứng dụng Notification Services từ SQL Server Management Studio bằng cách trước tiên mở Object Browser và nhấp phải nút Notification Services. Sau đó, bạn có thể chọn tùy chọn New Notification Services Instance từ menu ngữ cảnh để hiển thị một màn hình như minh họa ở hình 5.3.



Hình 5.3: Hộp thoại New Notification Services

Để tạo một ứng dụng Notification Services mới nhờ dùng hộp thoại New Notification Services, bạn nhấp Browse và định hướng đến thư mục có chứa file config.xml của ứng dụng. Kế tiếp, bạn chọn file config.xml và nhấp OK. Nếu bạn muốn ứng dụng được kích hoạt ngay sau khi nó được tạo, bạn cần đánh dấu kiểm vào hộp Enable Instance After It Is Created.

Tạo các ứng dụng Notification Services bằng cách sử dụng Nscontrol

Nscontrol là một công cụ dòng lệnh dùng để tạo và quản lý các ứng dụng Notification Services. Nscontrol hiểu được số lệnh hành động khác nhau mà bạn có thể dùng để làm việc với các ứng dụng Notification Services. Bảng 5.1 liệt kê các lệnh hành động nscontrol có sẵn.

Bảng 5.1: Các lệnh Nscontrol

Lệnh Nscontrol

nscontrol create

nscontrol delete

nscontrol disable

nscontrol displayargumentkey

Mô tả

Tạo một ứng dụng Notification Services và các cơ sở dữ liệu của nó.

Xóa một ứng dụng Notification Services và các cơ sở dữ liệu của nó.

Tắt một ứng dụng Notification Services.

Hiển thị khóa dùng để mã hóa dữ liệu event.

nscontrol enable	Kích hoạt một ứng dụng Notification Services.
nscontrol listversions	Hiển thị phiên bản của Notification Services và bất kỳ ứng dụng đã đăng ký.
nscontrol register	Đăng ký một ứng dụng Notification Services.
nscontrol status	Hiển thị tình trạng của một ứng dụng Notification Services.
nscontrol unregister	Hủy đăng ký một ứng dụng Notification Services.
nscontrol update	Cập nhật một ứng dụng Notification Services.

Tạo một ứng dụng Notification Services là một tiến trình gồm nhiều bước. Trước tiên, ứng dụng cần được tạo nhờ dùng câu lệnh nscontrol create. Lệnh này tạo cơ sở dữ liệu được sử dụng bởi ứng dụng Notification Services. Kế tiếp, ứng dụng cần được đăng ký nhờ dùng lệnh nscontrol register. Lệnh này tạo dịch vụ sẽ được sử dụng để chạy ứng dụng. Sau đó, ứng dụng cần được kích hoạt nhờ dùng lệnh nscontrol enable. File lô sau đây minh họa trình tự lệnh cần thiết để tạo ứng dụng NSSample Notification Services mẫu:

```

echo off
cls
set NSdir="C:\Program Files\Microsoft SQL Server\90\NotificationServices\9.0.242\bin"
echo =====
echo Beginning NSSampleInstance Creation
echo =====
echo .
echo Create the application databases
%NSdir%\nscontrol create -in config.xml

echo Register the application
%NSdir%\nscontrol register -name NSSampleInstance -service

echo Enable the application
%NSdir%\nscontrol enable -name NSSampleInstance

echo start the NS app as a service

```

`net start NS$NSSampleInstance`

`echo` Display the status of the app

`%NSdir%\nscontrol status -name NSSampleInstance`

The `nscontrol create` command `-in` argument specifies the name of the

Đối số `-in` của lệnh `nscontrol create` chỉ định tên của Notification Services ACF. Trong ví dụ này, ACF được đặt tên là `config.xml`. Việc chạy lệnh `nscontrol create` sẽ tạo ra hai cơ sở dữ liệu trên server, `NSSampleInstanceMain` và `NSSampleInstanceNSSample`, vốn chứa định nghĩa ứng dụng Notification Services và các event dữ liệu.

Lệnh `nscontrol register` dùng đối số `-name` để nhận biết tên instance của ứng dụng Notification Services cần đăng ký. Lệnh chuyển đổi `-service` hướng nó đến đăng ký một dịch vụ được gọi là `NS$NSSampleInstance`.

Lệnh `nscontrol enable` dùng thông số `-name` để nhận biết tên instance của ứng dụng sẽ được kích hoạt.

Ngay sau khi ứng dụng đã được kích hoạt, dịch vụ của nó có thể được mở nhờ dùng lệnh `net start`. Để thử, bạn cũng có thể thực thi ứng dụng `NS$NSSampleInstance` từ dòng nhắc lệnh hoặc hộp thoại Run.

Thêm các thuê bao vào ứng dụng

Những người đăng ký hay các thuê bao (subscribers) được thêm vào các ứng dụng Notification Services nhờ dùng các API mà đã được quản lý mà Microsoft cung cấp với SQL Server 2005 Notification Services. Các API .NET Framework của Microsoft cho phép bạn thêm, cập nhật và xóa các thuê bao cũng như các thiết bị thuê bao và nội dung đăng ký. Trong khi Notification Services API được cung cấp qua các lớp mã đã quản lý, bạn cũng có thể truy cập API từ mã chưa quản lý bằng cách dùng các ứng dụng COM trên nền Win32.

Notification Services API nằm trong `Microsoft.SqlServer.NotificationServices.dll`, vốn phải được thêm vào dự án .NET của bạn ở dạng một tham chiếu. Sau đó, bạn có thể sử dụng các lớp Notification Services để quản lý các phần đăng ký vào các ứng dụng notification services của bạn. Mã sau đây minh họa cách bạn có thể thêm một phần đăng ký nhờ dùng API mà đã được quản lý:

```
using Microsoft.SqlServer.NotificationServices;
```

```
using System.Text;
```

```
public class NSSubscriptions
```

```
{
```

```

private string AddSubscription(string instanceName, string
    applicationName, string subscriptionClassName, string subscriberId)
{
    // Create the Instance object
    NSInstance myNSInstance = new NSInstance(instanceName);

    // Create the Application object
    NSApplication myNSApplication = new NSApplication
        (myNSInstance, applicationName);

    // Create the Subscription object
    Subscription myNSSubscription = new Subscription
        (myNSApplication, subscriptionClassName);
    myNSSubscription.Enabled = true;
    myNSSubscription.SubscriberId = subscriberId;

    // Set the subscription data fields
    myNSSubscription["DeviceName"] = "MyDevice";
    myNSSubscription["SubscriberLocale"] = "USA";
    myNSSubscription["DataID"] = 1;
    myNSSubscription["DataTriggerValue"] = 100;

    // Add the subscription
    string subscriptionId = myNSSubscription.Add();
    return subscriptionId;
}
}

```

Ở đầu danh sách này, bạn có thể thấy một chỉ thị nhập cho namespace `Microsoft.SqlServer.NotificationServices`. Việc sử dụng chỉ thị nhập (import directive) cho phép bạn dùng các lớp trong namespace `NotificationServices` mà không bắt buộc bạn định rõ đầy đủ các tên. Kế tiếp, bên trong phương thức `AddSubscription` bạn thấy mã để thêm phần đăng ký. Trước tiên, một đối tượng thể hiện `Notification Services` được tạo, tiếp theo sau là một đối tượng `Application`. Sau đó, đối tượng `Application` được sử dụng để tạo một đối tượng đăng ký với tên `myNSSubscriptions`. Các thuộc tính của đối tượng `Subscription` được gán các trị và phương thức `Add` của nó được gọi để thật sự thêm phần đăng ký vào cơ sở dữ liệu.

CHƯƠNG 6**SQL Server Service Broker**

SQL Server Service Broker là một hệ thống phụ mới bổ sung thêm sự hỗ trợ bảo đảm Asynchronous queuing (tạo hàng đợi không đồng bộ) vào SQL Server 2005. Asynchronous queuing tăng thêm một kích thước mở rộng vào SQL Server 2005. Asynchronous queuing được tìm thấy trong nhiều ứng dụng khác có khả năng mở rộng cao, bao gồm các hệ thống phụ I/O của hệ điều hành, các web server và cả các hoạt động bên trong của chính bộ máy cơ sở dữ liệu SQL Server. Sự bổ sung asynchronous queuing vào SQL Server 2005 cũng mang lại khả năng xử lý asynchronous queuing cho các ứng dụng cơ sở dữ liệu người dùng cuối (end-user). Asynchronous queuing là một nhân tố quan trọng cho khả năng mở rộng bởi vì nó cho phép một chương trình phản hồi nhiều yêu cầu hơn so với khả năng mà nền có thể xử lý về mặt vật lý. Chẳng hạn, trong trường hợp của một web server nếu 10 ngàn user đồng thời yêu cầu các tài nguyên từ server mà không có asynchronous queuing, web server sẽ bị tràn ngập vì nó cố mở các luồng để xử lý tất cả những yêu cầu đến. Asynchronous queuing cho phép bắt mọi yêu cầu trong một hàng đợi (queue) sao cho thay vì tràn ngập, web server có thể xử lý các mục nhập từ hàng đợi ở mức hiệu quả tối đa. Trong trường hợp của web server, asynchronous queuing cho phép server xử lý hiệu quả nhiều nối kết user hơn. SQL Server Service Broker cho phép bạn cài cùng loại mở rộng này vào các ứng dụng cơ sở dữ liệu của bạn.

Trong chương này, bạn sẽ tìm hiểu về các tính năng mới do SQL Server Service Broker cung cấp. Bạn sẽ xem tổng quan về hệ thống phụ mới và tìm hiểu các thành phần cốt lõi của nó. Sau đó, bạn sẽ xem các điểm cơ bản về cách tạo các ứng dụng SQL Server Service Broker. Ở đây bạn sẽ tìm hiểu metadata của SQL Server Service Broker và mô hình lập trình của nó. Trước tiên, bạn sẽ học cách tạo các loại thông báo và hàng đợi. Sau đó, bạn sẽ xem cách sử dụng các lệnh T-SQL mới vốn cho phép các ứng dụng tạo thông báo thêm các mục nhập vào một hàng đợi cũng như nhận các mục nhập đã được thêm vào một hàng đợi. Sau cùng, bạn sẽ học về một số tính năng quản trị trong hệ thống phụ SQL Server Service Broker.

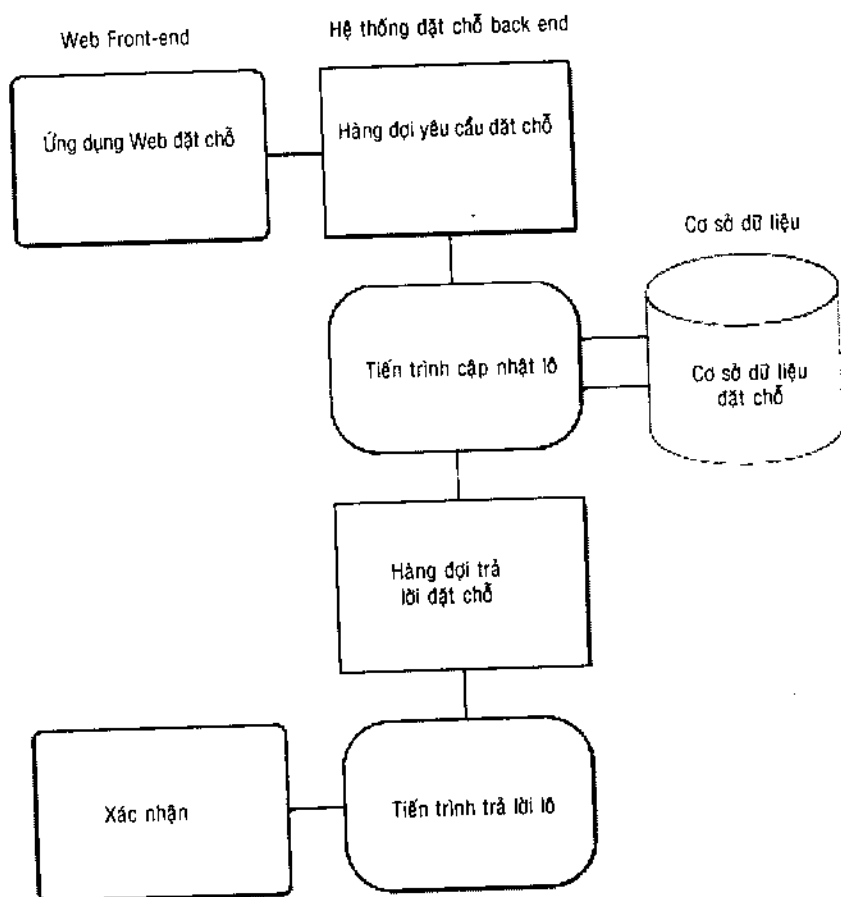
Tổng quan về SQL Server Service Broker

SQL Server Service Broker bổ sung khả năng thực hiện asynchronous queuing (tạo hàng đợi không đồng bộ) vào SQL Server 2005. Khả

năng tạo hàng đợi mới này được cài vào bộ máy SQL Server và hoàn toàn có tính giao tác. Các giao tác (Transaction) có thể hợp nhất các event trong hàng đợi và có thể vừa được gửi đi vừa được cuộn lại. Bạn truy cập SQL Server Service Broker bằng cách sử dụng các câu lệnh SQL Server mới. Các ví dụ về những câu lệnh này sẽ được trình bày trong mục kế tiếp của chương. Ngoài ra, SQL Server Service Broker mới cũng hỗ trợ sự chuyển phát đáng tin cậy của các thông báo đến các hàng đợi từ xa. Điều này có nghĩa là các ứng dụng tạo hàng đợi được thiết lập nhờ dùng SQL Server Service Broker có thể mở rộng qua nhiều hệ thống SQL Server mà vẫn cung cấp sự chuyển phát thông báo bảo đảm - ngay cả đến các hàng đợi từ xa. Các thông báo được gửi qua các hàng đợi có thể rất lớn, đến 2GB. SQL Server Service Broker sẽ lưu ý đến các cơ chế bắt buộc để tách các thông báo lớn ra thành các cụm nhỏ hơn vốn được gửi qua mạng và sau đó tập hợp lại ở đầu kia. Có cả hai phiên bản 32 bit và 64 bit của SQL Server Service Broker.

Thiết kế ứng dụng hàng đợi

Tuy ý tưởng tạo hàng đợi trong các ứng dụng có lẽ còn mới lạ với hầu hết những người thiết kế cơ sở dữ liệu, nhưng các hàng đợi khá phổ biến trong các ứng dụng có khả năng mở rộng cao. Một trong những loại nổi tiếng nhất trong số các loại ứng dụng này là các hệ thống đặt chỗ của hãng hàng không được sử dụng bởi tất cả những hãng hàng không lớn như United, Delta, và American, cũng như bởi các nhà môi giới du lịch chẳng hạn như Expedia và CheapTickets.com. Để hiểu rõ việc tạo hàng đợi được áp dụng như thế nào trong một trong những ứng dụng này, hãy xem hình 6.1, hình này minh họa kiểu thiết kế của một ứng dụng tạo hàng đợi mẫu.



Hình 6.1 Kiểu thiết kế của ứng dụng đã tạo hàng đợi

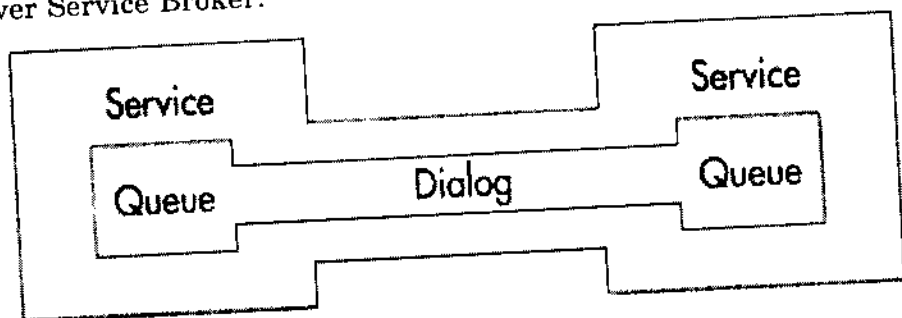
Hình 6.1 trình bày một khung xem tổng quan cấp cao về một hệ thống đặt chỗ (reservation) của hãng hàng không. Ở đây bạn có thể thấy lớp trình bày của ứng dụng được chuyển đến trình duyệt của người dùng cuối bởi một ứng dụng đang chạy trên một web farm. Ứng dụng đó có thể được viết nhờ dùng ASP.NET hay một ngôn ngữ phát triển web khác. Ứng dụng front-end sau đó sẽ tương tác với hệ thống đặt chỗ thật sự mà bình thường nó chạy trên một hệ thống khác.

Bởi vì các ứng dụng như thế này phải hỗ trợ hàng ngàn user đồng thời nên chúng không có đủ khả năng khóa các hàng trong khi một user đợi quyết định về các chi tiết cuối cùng của một chuyến bay hay bắt đầu đặt chỗ, đi ăn trưa, lập kế hoạch cho đến lúc kết thúc. Việc khóa hàng (row locking) trong loại tình huống này sẽ kìm hãm trầm trọng khả năng mở rộng của ứng dụng và cả tính dễ sử dụng của ứng dụng. Việc tạo hàng đợi (queuing) giúp giải quyết vấn đề này bằng cách cho phép

ứng dụng tạo ra một yêu cầu không đồng bộ cho một lần đặt chỗ (reservation), nó gửi yêu cầu đến hệ thống đặt chỗ back-end và lập tức được giải phóng cho tác vụ khác. Không có điểm nào trong tiến trình đặt chỗ có khóa được đặt trên các table cơ sở dữ liệu. Hệ thống đặt chỗ back-end, về cơ bản nó hoạt động trong chế độ lô (batch), sẽ lấy yêu cầu đặt chỗ từ hàng đợi rồi thực hiện việc cập nhật vào cơ sở dữ liệu. Vì tiến trình cập nhật đang được thực hiện trong chế độ lô nên nó xảy ra rất nhanh mà không cần sự tương tác từ người dùng và chỉ cần một lượng thời gian rất nhỏ để khóa các hàng trong khi tiến trình cập nhật đang được thực hiện. Nếu yêu cầu thành công thì sự đặt chỗ của người dùng cuối được xác nhận. Ngược lại, nếu yêu cầu bị từ chối do tất cả các chỗ đã được đặt trước hay vì một lý do nào đó, sự đặt chỗ sẽ không được chấp nhận và user sẽ được thông báo về tình trạng này.

Hộp thoại

Hộp thoại là một thành phần thiết yếu trong SQL Server Service Broker mới của Microsoft. Về cơ bản, hộp thoại (dialog) cung cấp sự truyền thông báo hai chiều giữa hai điểm cuối (endpoint). Các điểm cuối cho các thông báo này có thể là hai ứng dụng đang chạy trên các server hay các instance khác nhau, hoặc chúng có thể là hai ứng dụng đang chạy trên cùng một server. Hình 6.2 minh họa cuộc hộp thoại của SQL Server Service Broker.



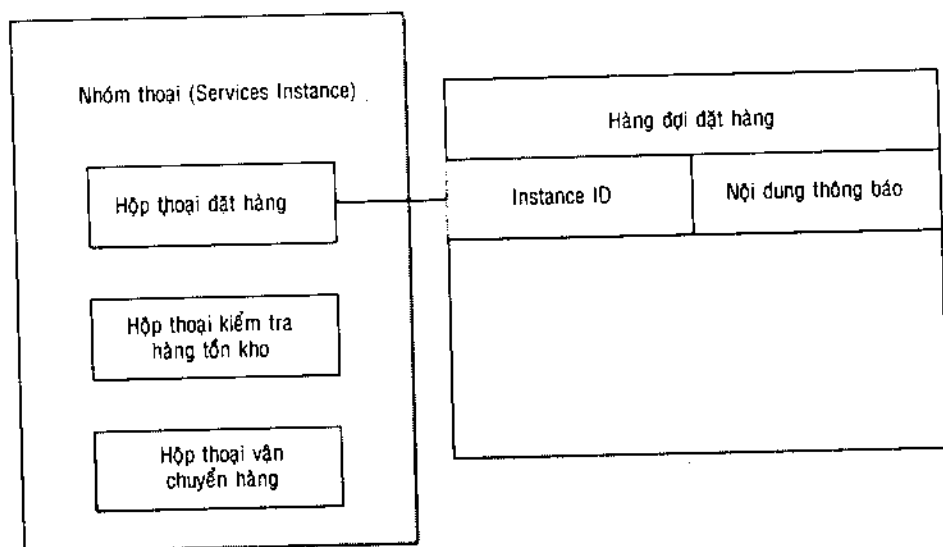
Hình 6.2 Hộp thoại SQL Server Service Broker

Mục đích chính của một cuộc hộp thoại SQL Server Service Broker là cung cấp một chuỗi event theo trình tự. Các hộp thoại SQL Server Service Broker duy trì trình tự event một cách đáng tin cậy qua các server hay các luồng (thread) ngay cả khi có các sự cố mạng, ứng dụng hay server tạm thời làm gián đoạn việc xử lý các event trong hàng đợi. Khi tiến trình xử lý hàng đợi được phục hồi, các event sẽ tiếp tục được xử lý theo trình tự kể từ lúc event cuối cùng trong hàng đợi được xử lý. Các cuộc hộp thoại cho phép các thông báo trong hàng đợi luôn được đọc ở cùng trình tự mà chúng đã được đưa vào hàng đợi. Các cuộc hộp

thoại có thể được cài đặt để xử lý các event hoặc trong chế độ full-duplex hoặc trong chế độ half-duplex.

Nhóm thoại

Một thành phần cốt lõi khác của hệ thống phụ SQL Server Service Broker là nhóm thoại. Một nhóm thoại (conversation group) là một tập hợp các hộp thoại SQL Server Service Broker có liên quan với nhau. Một ứng dụng có thể yêu cầu nhiều cuộc hộp thoại để hoàn tất một tác vụ, và một nhóm thoại cho phép bạn nhóm tất cả các cuộc hộp thoại có liên quan đó lại với nhau một cách lô gíc. Chẳng hạn, một ứng dụng nhập đơn đặt hàng có thể có một cuộc hộp thoại để xử lý các đơn đặt hàng, một cuộc hộp thoại để xử lý danh mục hàng tồn kho, một cuộc hộp thoại để xử lý việc chuyển hàng, và một cuộc hộp thoại khác để xử lý các yêu cầu thanh toán. Tất cả những cuộc hộp thoại có liên quan này có thể được nhóm lại với nhau để tạo thành một nhóm thoại. Trong trường hợp này, nhóm thoại về cơ bản tiêu biểu cho mọi cuộc hộp thoại cho một ứng dụng. Hình 6.3 minh họa mối quan hệ của nhóm thoại và cuộc hộp thoại của SQL Server Service Broker.



Hình 6.3 Nhóm thoại SQL Server Service Broker

Mục đích chính đằng sau nhóm thoại là cung cấp một cơ chế khóa cho những hàng đợi có liên quan. Về mặt này, bạn cần hiểu rằng khóa không được áp dụng cho bất kỳ table cơ sở dữ liệu nền tảng mà khóa do nhóm thoại kích hoạt được áp dụng cho các thực thể trong các hàng đợi. Trong trường hợp của đơn đặt hàng, điều này có nghĩa là không có tiến trình nào có thể đọc các đơn đặt hàng cho bất kỳ hàng đợi trong nhóm

thoại của bạn trong khi ứng dụng đang giữ một khóa trên các mục nhập đó.

Ngoài khóa (locking) nhóm thoại còn cho phép ứng dụng duy trì trạng thái. Duy trì trạng thái luôn là một trong những khía cạnh khó nhất của một ứng dụng không đồng bộ bởi vì có khoảng thời gian trì hoãn khá lâu giữa các lần đến của thông báo. Việc liên tục để mở các luồng hoạt động trong khoảng thời gian đó không phải là một sự lựa chọn bởi vì tất cả các luồng chưa dùng đến sẽ chiếm rất nhiều tài nguyên hệ thống và hạn chế khả năng mở rộng. Nhóm thoại (conversation group) duy trì trạng thái của một thực thể nhờ dùng một table trạng thái. Nhóm thoại sử dụng một instance ID (một GUID) làm khóa cơ sở để nhận biết các mục nhập trong nhóm hàng đợi.

Kích hoạt SQL Server Service Broker

Kích hoạt SQL Server Service Broker là một tính năng duy nhất khác của hệ thống phụ SQL Server Service Broker. Sự kích hoạt (activation) cho phép bạn tạo một thủ tục lưu trữ có liên quan với một hàng đợi nhập liệu đã cho. Mục đích của thủ tục lưu trữ là tự động xử lý các thông báo từ hàng đợi đó. Khi mỗi thông báo mới đến, thủ tục lưu trữ liên quan tự động được thực thi để xử lý các thông báo đến. Nếu thủ tục lưu trữ bắt gặp một lỗi, nó tung ra một ngoại lệ và tự động được tái sử dụng.

Theo định kỳ, SQL Server Service Broker kiểm tra tình trạng của hàng đợi nhập liệu (input queue) để tìm hiểu xem thủ tục lưu trữ có theo kịp các thông báo đến trên hàng đợi nhập liệu hay không. Nếu SQL Server Service Broker xác định có các thông báo đang đợi trên hàng đợi thì nó sẽ tự động mở một instance khác của bộ đọc hàng đợi để xử lý các thông báo bổ sung. Tiến trình tự động mở các bộ đọc hàng đợi bổ sung này có thể tiếp tục cho đến khi đạt tới trị MAX_QUEUE_READERS đã được xác lập sẵn. Cũng vậy, khi SQL Server Service Broker xác định không còn thông báo nào trên hàng đợi, nó sẽ bắt đầu tự động giảm số lượng bộ đọc hàng đợi (queue reader).

Các hàng đợi SQL Server Service Broker không nhất thiết phải được kết hợp với các thủ tục lưu trữ. Các thông báo cần nhiều việc xử lý phức tạp hơn cũng có thể được kết hợp với các thủ tục bên ngoài ở tầng giữa. Vì các tiến trình ở tầng giữa này nằm bên ngoài cơ sở dữ liệu nên chúng cần được kích hoạt theo cách khác. Để kích hoạt tự động các tiến trình bên ngoài, SQL Server Service Broker cũng hỗ trợ việc kích khởi một event SQL Server. Các event này có thể được đăng ký với việc sử dụng WMI (Windows Management Instrumentation). Trong trường hợp này, khi một event vào hàng đợi, event SQL Server được kích khởi và được đọc bởi thuê bao WMI (WMI subscriber), mà đến lượt nó sẽ mở bộ đọc hàng đợi ngoài.

Chuyển thông báo

Giao thức chuyển thông báo SQL Server Service Broker cho phép các thông báo được gửi qua mạng. Việc chuyển thông báo SQL Server Service Broker dựa trên TCP/IP và kiểu kiến trúc chung của hệ thống chuyển thông báo SQL Server Service Broker thì hơi giống kiểu kiến trúc được sử dụng bởi TCP/IP và FTP. Tiến trình chuyển thông báo SQL Server Service Broker được tạo thành từ hai giao thức: giao thức Binary Adjacent Broker, đây là một giao thức cấp thấp như TCP và giao thức Dialog, đây là một giao thức cấp cao hơn như FTP và “cưỡi” trên giao thức Binary Adjacent Broker cấp thấp.

Giao thức Binary Adjacent Broker

Giao thức Binary Adjacent Broker là một giao thức TCP/IP cấp thấp có hiệu quả cao vốn cung cấp tiến trình chuyển thông báo cơ bản. Nó là một giao thức hai hướng và có nhiều thành phần, vì vậy nó có thể xử lý việc chuyển thông báo cho nhiều hộp thoại SQL Server Service Broker. Giao thức Binary Adjacent Broker không bận tâm về trình tự thông báo hay xác nhận sự chuyển phát thông báo. Điều đó hoàn toàn được xử lý bởi giao thức Dialog. Giao thức Binary Adjacent Broker chỉ gửi các thông báo qua mạng nhanh như nó có thể.

Giao thức Dialog

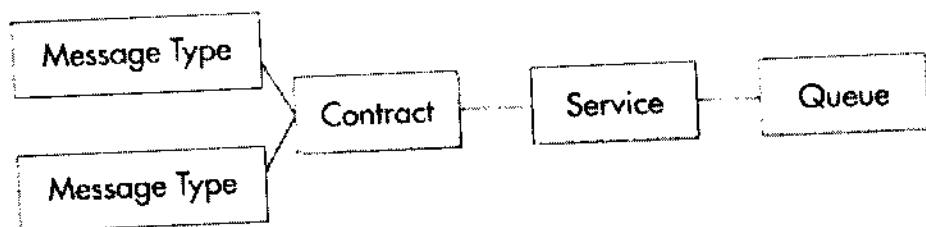
Giao thức Dialog xử lý các cuộc giao tiếp từ đầu này đến đầu kia cho một hộp thoại SQL Server Service Broker. Được thiết kế để cung cấp sự chuyển phát các thông báo một lần theo trình tự (one-time-only in-order), nó xử lý việc gửi các thông báo và xác nhận chúng. Nó cũng cung cấp khả năng xử lý lỗi đối xứng, trong đó cả hai nút cuối được thông báo về bất kỳ lỗi chuyển phát thông báo. Ngoài ra, giao thức Dialog còn chịu trách nhiệm xác thực và mã hóa các thông báo.

Phát triển với SQL Server Service Broker

Như bạn đã thấy trong phần đầu của chương này, SQL Server Service Broker là một hệ thống phụ cho phép phát triển các ứng dụng gửi thông báo theo hướng cơ sở dữ liệu. Trong khi phần đầu của chương này cung cấp cho bạn một khung xem tổng quan về các thành phần chính của hệ thống phụ SQL Server Service Broker và giúp bạn hiểu rõ các chức năng cũng như sự tương tác của các thành phần đó, mục này sẽ trình bày chi tiết hơn về cách bạn phát triển các ứng dụng nhờ dùng SQL Server Service Broker.

Mô hình lập trình

SQL Server Service Broker tận dụng một tập hợp metadata để mô tả các loại thông báo mà một ứng dụng có thể nhận được, các thông báo được gửi đi theo những hướng nào và các thông báo có liên quan với nhau như thế nào. Về cơ bản, metadata của SQL Server Service Broker mô tả cơ cấu gửi thông báo chính của một ứng dụng. Ngoài ra, metadata có các permission kết hợp dùng để bảo toàn ứng dụng gửi thông báo (message application). Hình 6.4 minh họa mô hình metadata của SQL Server Service Broker.



Hình 6.4 Metadata SQL Server Service Broker

Loại thông báo (message type) là đối tượng cơ bản nhất trong metadata của SQL Server Service Broker. Loại thông báo được sử dụng để mô tả các thông báo nào sẽ được dùng. Với các thông báo XML, các loại thông báo có thể có một schema tùy ý kết hợp với thông báo. Nếu có một schema kết hợp, SQL Server Service Broker sẽ bảo đảm thông báo biên dịch với định nghĩa schema. Nếu không có schema nào thì các thông báo được xem như là dữ liệu nhị phân.

Đối tượng chính kế tiếp trong metadata của SQL Server Service Broker là contract (hợp đồng). Các loại thông báo được nhóm thành các contract. Contract mô tả tất cả các thông báo có thể nhận được nhờ dùng một hộp thoại cụ thể nào đó. Chẳng hạn, nếu một hộp thoại yêu cầu và trả lời đặt chỗ hình thành một hệ thống đặt chỗ đơn giản, contract sẽ chỉ định hai loại thông báo đó vừa được nhóm lại với nhau vừa liên hệ với hệ thống đặt chỗ. Về cơ bản, contract chỉ định những loại thông báo nào là được phép bởi một hộp thoại cho trước.

Các contract được nhóm lại với nhau để hình thành một dịch vụ (service), về cơ bản dịch vụ này tiêu biểu cho tất cả các cuộc hộp thoại được yêu cầu để xử lý các thông báo cho dịch vụ đó. Một dịch vụ được kết hợp với một hay nhiều hàng đợi và là đối tượng chính được điều khiển bởi ứng dụng SQL Server Service Broker. Ứng dụng mở một cuộc hộp thoại cho một dịch vụ. Ứng dụng không liên quan đến các chi tiết vật lý về cách dịch vụ được thực thi, vì các chi tiết thực thi vật lý được xác định bởi metadata. Hàng đợi (queue) chứa các thông báo mà một ứng dụng SQL Server Service Broker nhận được.

T-SQL DDL và DML

T-SQL đã được cải tiến với nhiều câu lệnh mới vốn cho phép hợp nhất việc truyền thông báo SQL Server Service Broker với các thủ tục cơ sở dữ liệu thông thường. Bảng 6.1 tóm tắt các câu lệnh T-SQL DDL mới dùng để tạo các đối tượng SQL Server Service Broker.

Bảng 6.1 Các câu lệnh TSQL dùng cho các đối tượng SQL Server Service Broker

T-SQL DDL	Mô tả
CREATE MESSAGE TYPE	Tạo một loại thông báo mới
CREATE CONTRACT	Tạo một contract mới trong một cơ sở dữ liệu
CREATE QUEUE	Tạo một hàng đợi mới trong một cơ sở dữ liệu
CREATE ROUTE	Tạo một lộ trình mới trong một cơ sở dữ liệu
CREATE SERVICE	Tạo một dịch vụ mới trong một cơ sở dữ liệu
ALTER MESSAGE TYPE	Thay đổi một loại thông báo
ALTER CONTRACT	Thay đổi một contract (hợp đồng)
ALTER QUEUE	Thay đổi một hàng đợi
ALTER ROUTE	Thay đổi một lộ trình
ALTER SERVICE	Thay đổi một dịch vụ
DROP MESSAGE TYPE	Xóa một loại thông báo ra khỏi một cơ sở dữ liệu
DROP CONTRACT	Xóa một contract ra khỏi một cơ sở dữ liệu
DROP QUEUE	Xóa một hàng đợi ra khỏi một cơ sở dữ liệu
DROP ROUTE	Xóa một lộ trình ra khỏi một cơ sở dữ liệu
DROP SERVICE	Xóa một dịch vụ ra khỏi một cơ sở dữ liệu

Ngoài các câu lệnh T-SQL DDL mới dùng để tạo các đối tượng SQL Server Service Broker, còn có một nhóm các câu lệnh T-SQL mới hoạt động với các thông báo trong một ứng dụng SQL Server Service Broker. Bảng 6.2 liệt kê các câu lệnh T-SQL DML có liên quan với SQL Server Service Broker mới.

Bảng 6.2 Các câu lệnh T-SQL dành cho các thông báo SQL Server Service Broker

T-SQL DML	Mô tả
BEGIN DIALOG	Mở một đối thoại (dialog) mới
END CONVERSATION	Kết thúc một cuộc nói chuyện do một đối thoại sử dụng
MOVE CONVERSATION	Chuyển một cuộc nói chuyện sang một đối thoại mới
GET SERVICE INSTANCE	Truy xuất một service instance ID từ một hàng đợi
RECEIVE	Nhận một thông báo từ một hàng đợi
SEND	Gửi một thông báo đến một hàng đợi
BEGIN CONVERSATION TIMER	Mở bộ định giờ (timer). Khi bộ định giờ hết hạn, Service Broker đặt một thông báo trên hàng đợi

Ứng dụng SQL Server Service Broker mẫu

Trong phần đầu tiên của mục này, bạn đã xem tổng quan về cách sử dụng SQL Server Service Broker để phát triển các ứng dụng truyền thông báo không đồng bộ. Mục này bây giờ sẽ trình bày chuyên sâu hơn và minh họa cách bạn sử dụng T-SQL để tạo các đối tượng SQL Server Service Broker theo yêu cầu và tiếp tục hướng dẫn cách sử dụng chúng trong một ứng dụng đơn giản. Ứng dụng mẫu là một hệ thống đặt chỗ đơn giản vốn chấp nhận một yêu cầu đặt chỗ đơn giản trên một hàng đợi nhập liệu (input queue) rồi trả lời bằng một thông báo trên một hàng đợi trả lời (response queue).

Tạo các đối tượng SQL Server Service Broker

Mã dùng để tạo các đối tượng SQL Server Service Broker theo yêu cầu được minh họa trong danh sách sau đây:

```
CREATE MESSAGE TYPE ResRequest
    ENCODING varbinary
CREATE MESSAGE TYPE ResResponse
    ENCODING varbinary
```

```
CREATE CONTRACT ResContract
    (ResRequest SENT BY any,
    ResResponse SENT BY any)
```

```
CREATE QUEUE ResRequestQueue WITH ACTIVATION  
(STATUS = ON,  
  PROCEDURE_NAME = ResRequestProc,  
  MAX_QUEUE_READERS = 5,  
  EXECUTE AS SELF)
```

```
CREATE QUEUE ResResponseQueue
```

```
CREATE SERVICE ResRequestService ON QUEUE  
  ResRequestQueue(ResContract)
```

```
CREATE SERVICE ResResponseService ON QUEUE  
  ResResponseQueue(ResContract)
```

Bước đầu tiên để tạo một ứng dụng SQL Server Service Broker là tạo các loại thông báo, mô tả những thông báo nào sẽ được gửi đi. Hai câu lệnh đầu tiên tạo ra hai loại thông báo đơn giản. Thông số đầu tiên được sử dụng để đặt tên loại thông báo, và từ khóa ENCODING cho biết thông báo sẽ có dạng nhị phân hay XML. Trong ví dụ này, cả hai loại thông báo ResRequest và ResResponse đều được tạo ở dạng các thông báo nhị phân, nghĩa là chúng sẽ chấp nhận bất kỳ loại dữ liệu nào.

Kế tiếp, một contract (hợp đồng) được tạo. Contract này mô tả tất cả những thông báo có thể nhận được nhờ dùng một hộp thoại cụ thể. Đối số đầu tiên dùng để đặt tên contract. Mệnh đề SENT BY dùng để chỉ những thông báo nào được kết hợp với contract và nơi xuất phát của những thông báo đó.

Sau đó các hàng đợi phải được tạo. Ví dụ này minh họa việc tạo hai hàng đợi: ResRequestQueue và ResResponseQueue. ResRequestQueue dùng từ khóa ACTIVATION để kích khởi tự động một thủ tục lưu trữ (stored procedure) vốn sẽ đọc các nội dung của hàng đợi. Thủ tục lưu trữ này phải tồn tại vào lúc hàng đợi được tạo; nếu không một lỗi sẽ xảy ra. Thủ tục lưu trữ ResRequest được minh họa ở sau trong mục này. Từ khóa MAX_QUEUE_READERS chỉ định số lượng reader (bộ đọc) tối đa mà SQL Server Service Broker sẽ tự động kích hoạt. Tùy chọn EXECUTE AS cho phép bạn thực thi thủ tục đã được kích hoạt trong một ngữ cảnh user khác.

Sau khi các hàng đợi đã được tạo, bạn có thể hiển thị các nội dung của các hàng đợi bằng cách dùng câu lệnh SELECT chính xác như thể hàng đợi là một table cơ sở dữ liệu chuẩn. Dòng mã sau đây cho thấy cách hiển thị các nội dung của hàng đợi Request:

```
SELECT * FROM ResRequestQueue.
```

Ngay sau khi các hàng đợi được tạo, chúng còn rỗng. Tuy nhiên, việc chạy các câu lệnh SELECT trên hàng đợi là một cách hay để kiểm tra tính năng của các ứng dụng SQL Server Service Broker mà bạn đang phát triển.

Sau khi các hàng đợi đã được tạo xong, bước kế tiếp là tạo một dịch vụ (service). Bạn tạo một dịch vụ nhờ dùng câu lệnh CREATE SERVICE. Thông số đầu tiên đặt tên dịch vụ. Mệnh đề ON QUEUE nhận biết hàng đợi được kết hợp với dịch vụ, và sau đó các contract có liên quan với dịch vụ được liệt kê.

Nếu một trong các dịch vụ nằm trên một hệ thống từ xa, bạn cũng sẽ cần tạo một lộ trình (route). Câu lệnh CREATE ROUTE cung cấp cho SQL Server Service Broker địa chỉ hệ thống nơi tìm thấy dịch vụ từ xa, nó cho SQL Server biết cách chuyển phát thông báo đến hệ thống từ xa.

Gửi các thông báo đến một hàng đợi

Sau khi các đối tượng SQL Server Service Broker cần thiết đã được tạo, bạn chuẩn bị sử dụng chúng trong các ứng dụng tạo hàng đợi. Danh sách mã sau đây minh họa cách bạn có thể thêm một thông báo vào hàng đợi ResRequestQueue:

```
DECLARE @ResRequestDialog UNIQUEIDENTIFIER
```

```
BEGIN TRANSACTION
```

```
BEGIN DIALOG @ResRequestDialog  
FROM SERVICE ResResponseService  
TO SERVICE 'ResRequestService'  
ON CONTRACT ResContract  
WITH LIFETIME = 1000;
```

```
SEND ON CONVERSATION @ResRequestDialog  
MESSAGE TYPE ResRequest (N'My Request Message');
```

```
SEND ON CONVERSATION @ResRequestDialog  
MESSAGE TYPE ResRequest (N'My Request Message2');
```

```
COMMIT TRANSACTION
```

Ở đầu danh sách này, bạn có thể thấy một biến tên ResRequestDialog được tạo; nó chứa một bộ nhận dạng duy nhất vốn sẽ được gán bởi một đối thoại SQL Server Service Broker. Kế tiếp, một giao tác được mở. Bạn nên gói gọn mọi hành động được thực hiện bởi SQL Server Service Broker trong một giao tác (transaction). Điều này cho phép bạn ủy thác và tùy ý cuộn lại bất kỳ thay đổi nào sang hàng đợi. Sau đó câu lệnh BEGIN DIALOG được sử dụng để mở một đối thoại SQL Server Service Broker mới. Khi bạn khai báo một đối thoại, bạn luôn cần chỉ định hai điểm cuối. FROM SERVICE nhận dạng người khởi tạo của các thông báo, trong khi từ khóa TO SERVICE nhận dạng điểm cuối đích. Ở đây, người gửi được đặt tên là ResResponseService và đích được đặt tên là ResRequestService. Tuy ví dụ này dùng các tên đơn giản nhưng Microsoft đề nghị bạn nên dùng một tên URL để nhận dạng riêng từng đối tượng SQL Server Service Broker. Chẳng hạn, để bảo đảm tính độc nhất trong mạng, Microsoft đề nghị dùng các tên như [http://MyCompany.com/MyApp/ResResponseservice]. Từ khóa ON CONTRACT chỉ định contract được sử dụng cho đối thoại. Contract chỉ định những thông báo nào mà đối thoại này có thể gửi hay nhận được.

Sau đó, hai thao tác SEND được thực thi. Các câu lệnh này gửi hai thông báo đến dịch vụ đích, vốn sẽ nhận được các thông báo đó và thêm chúng vào hàng đợi có liên quan với dịch vụ đó. Sau cùng, giao tác được chuyển đi.

Truy xuất các thông báo từ một hàng đợi

Bạn đã biết cách thêm một thông báo vào một hàng đợi, ví dụ kế tiếp sẽ minh họa cách tạo một thủ tục lưu trữ vốn đọc các thông báo từ hàng đợi. Như bạn có thể nhớ từ các ví dụ tạo đối tượng SQL Server Service Broker trước đây, hàng đợi đích, ResRequestQueue phải được tạo bằng sự kích hoạt. Điều này có nghĩa là một thủ tục lưu trữ kết hợp tên ResRequestProc sẽ được khởi động tự động khi một thông báo đến trên hàng đợi. Bạn có thể thấy mã cho thủ tục lưu trữ đó trong danh sách sau:

```
CREATE PROC ResRequestProc
AS
DECLARE @ResRequestDialog UNIQUEIDENTIFIER
DECLARE @message_type_id int
DECLARE @message_body NVARCHAR(1000)
DECLARE @message NVARCHAR(1000)
```

```
while(1=1)
```

```
BEGIN
```

BEGIN TRANSACTION

```
WAITFOR (RECEIVE top(1)
@message_type_id = message_type_id,
@message_body = message_body,
@ResRequestDialog = conversation_handle
FROM ResRequestQueue), TIMEOUT 200;

if (@@ROWCOUNT = 0)
BEGIN
    COMMIT TRANSACTION
    BREAK
END

IF (@message_type_id = 2) — End dialog message
    BEGIN
        PRINT ' Dialog ended '
        + cast(@ResRequestDialog as nvarchar(40))
    END
ELSE
    BEGIN
        BEGIN TRANSACTION
        BEGIN DIALOG @ResRequestDialog
        FROM SERVICE ResRequestService
        TO SERVICE 'ResResponseService
        ON CONTRACT ResContract
        WITH LIFETIME = 1000;

        SELECT @message = 'Received:' + @message_body;

        SEND ON CONVERSATION @ResRequestDialog
        MESSAGE TYPE ResResponse (@message);

        PRINT CONVERT(varchar(30), @message)
        COMMIT TRANSACTION
```

```
END CONVERSATION @ResRequestDialog
```

```
END
```

```
COMMIT TRANSACTION
```

```
END
```

Một biến (variable) sẽ chứa sự nhận dạng đối thoại trả lời được khai báo ở đầu thủ tục lưu trữ này, tiếp sau đó là ba biến dùng để thông tin từ hàng đợi đang được đọc. Sau đó một vòng lặp được khởi tạo để đọc tất cả các mục nhập từ hàng đợi. Bên trong vòng lặp đó, một giao tác được mở và câu lệnh RECEIVE được sử dụng để nhận một thông báo. Trong ví dụ này, mệnh đề TOP(1) được sử dụng để hạn chế thủ tục nhận mỗi lần chỉ một thông báo. Nếu mệnh đề TOP được bỏ qua, bạn có thể nhận mọi thông báo hiện có trên hàng đợi. Câu lệnh RECEIVE tập hợp ba biến. Message_type_id nhận dạng loại thông báo, thường là một thông báo do người dùng ấn định hay một thông báo End Dialog. Biến Ymessage_body chứa các nội dung của thông báo thật sự, trong khi đó biến @ResRequestDialog chứa một handle giúp nhận biết đối thoại đang gửi.

Kế tiếp, tập hợp kết quả được kiểm tra để bảo đảm một thông báo đã thật sự được nhận. Nếu không có hàng nào được nhận thì giao tác cuối được ủy thác và thủ tục kết thúc. Ngược lại, nếu các hàng đã được nhận, messagetypeid được kiểm tra để xem thông báo đó là một thông báo user hay là một thông báo End Dialog. Nếu đó là một thông báo user, các nội dung sẽ được xử lý. Trước tiên, một đối thoại được mở cho ResResponseService. Đối thoại này sẽ được sử dụng để gửi thông báo đã sửa đổi đến ResResponseQueue. Một lần nữa, ResContract được chỉ định, hạn chế những loại thông báo nào sẽ được phép.

Sau khi đối thoại đã được mở, thông báo đã nhận được sửa đổi bằng cách kết chuỗi "Received" với các nội dung của thông báo đã được nhận, sau đó câu lệnh SEND được sử dụng để gửi thông báo đã sửa đổi đến ResResponseQueue. Sau cùng, cuộc đối thoại kết thúc và giao tác được ủy thác.

Quản trị Service Broker

SQL Server Service Broker cung cấp nhiều tùy chọn cấu hình điều khiển cách hoạt động của hệ thống phụ SQL Server Service Broker. Trong mục này, bạn sẽ xem qua một số tùy chọn cấu hình quan trọng nhất, tìm hiểu hệ thống bảo mật SQL Server Service Broker cũng như xem xét một số khung xem hệ thống mà SQL Server 2005 cung cấp nhằm cho phép bạn lấy thông tin về các đối tượng và các ứng dụng SQL Server Service Broker.

Các tùy chọn cấu hình hệ thống

Có nhiều tùy chọn cấu hình hệ thống có thể được xác lập nhờ dùng thủ tục lưu trữ hệ thống `sp_configure` nhằm tác động đến cách hoạt động của hệ thống phụ SQL Server Service Broker. Bảng 6.3 liệt kê các tùy chọn `sp_configure` có sẵn cho SQL Server Service Broker.

Bảng 6.3 Các tùy chọn cho `sp_configure`

Thông số <code>sp_configure</code>	Mô tả
broker tcp listen port	Xác lập cổng mà SQL Server Service Broker dùng cho các kết nối mạng. Trị mặc định là 4022.
broker authentication mode	Xác lập loại xác thực từ xa sẽ được sử dụng cho các nối kết. Trị 1 có nghĩa là không có sự xác thực nào được sử dụng. Trị 2 có nghĩa là sự xác thực được hỗ trợ. Trị 3 có nghĩa là sự xác thực là bắt buộc. Trị mặc định là 3.
broker forwarding size limit	Xác lập khoảng trống đĩa tối đa theo đơn vị MB để chứa các thông báo sẽ được gửi đi. Trị mặc định là 10.
broker message forwarding	Xác lập loại chuyển tiếp thông báo. Trị 1 là không cho phép chuyển tiếp. Trị 2 cho phép chuyển tiếp với domain. Trị 3 cho phép chuyển tiếp ngoài. Trị mặc định là 1.

Bảo mật đối thoại

Khi các cuộc đối thoại (dialog) được tạo, chúng có thể tùy ý được bảo đảm an toàn nhờ dùng mệnh đề `WITH ENCRYPTION`. Khi một đối thoại được tạo nhờ dùng mệnh đề `WITH ENCRYPTION`, một khóa session (phiên làm việc) được tạo để mã hóa các thông báo được gửi đi nhờ dùng đối thoại này. Một điểm quan trọng về sự bảo mật đối thoại là nó một hình thức bảo mật nối tiếp nhau. Nói cách khác, thông báo được mã hóa khi lần đầu tiên nó được gửi từ một đối thoại, và nó không được giải mã cho đến khi nào thông báo đã tới điểm cuối của nó. Các nội dung thông báo vẫn được mã hóa khi thông báo được chuyển tiếp qua bất kỳ chặng trung gian. Để bảo mật đối thoại, SQL Server Service Broker dùng sự xác thực dựa trên certificate (chứng nhận), trong đó certificate của user đang gửi được gửi cùng với thông báo. Do bởi bản chất không đồng bộ của SQL Server Service Broker, thông tin bảo mật được chứa trong các header thông báo và được truy xuất bởi dịch vụ tiếp

nhận khi thông báo được truy xuất. Kiểu thiết kế này cho phép các ứng dụng SQL Server Service Broker tránh nhu cầu thiết lập một nối kết để xác thực các thông báo.

Các khung xem hệ thống

SQL Server 2005 cung cấp nhiều khung xem hệ thống mới nhằm cho phép bạn truy xuất thông tin về các đối tượng SQL Server Service Broker và các tình trạng hiện tại của chúng. Bảng 6.4 liệt kê các khung xem danh mục Service Broker mới.

Bảng 6.4 Các khung xem danh mục Service Broker mới

Khung xem hệ thống	Mô tả
sys.service_message_types	Liệt kê tất cả các loại thông báo đã được tạo. Các loại thông báo hệ thống được liệt kê ở vị trí đầu tiên, trong khi các loại thông báo do người dùng ấn định được liệt kê ở cuối màn hình hiển thị.
sys.service_contracts	Liệt kê tất cả những contract đã được tạo.
sys.service_contract_message_usages	Liệt kê các mối quan hệ giữa các contract và các loại thông báo. Các mối quan hệ có thể là một đối một (one-to-one) hay một đối nhiều (one-to-many).
sys.services	Liệt kê các dịch vụ đã được tạo.
sys.service_contract_usages	Liệt kê các mối quan hệ giữa các contract và dịch vụ. Các mối quan hệ có thể là một đối một hay một đối nhiều.
sys.service_instances	Liệt kê các dịch vụ hoạt động vào thời điểm hiện tại.
sys.conversation_endpoints	Liệt kê các điểm cuối đối thoại đang hoạt động.
sys.routes	Liệt kê các lộ trình đã được tạo.
sys.remote_service_bindings	Liệt kê các mối quan hệ của các dịch vụ và các user sẽ thực thi dịch vụ.
sys.transmission_queue	Liệt kê tất cả những thông báo trong hàng đợi sẽ được gửi đi.
sys.service_queues	Liệt kê các hàng đợi đã được tạo.

CHƯƠNG 7

Sự hợp nhất XML

Tổng quan

XML (eXtensible Markup Language) đã nổi lên như là một trong những công nghệ Internet quan trọng nhất. Kiểu cấu trúc dựa vào text linh động của XML đã làm cho nó được sử dụng cho nhiều tác vụ mạng, bao gồm chuyển dữ liệu / tài liệu, mô phỏng trang web, và cả ở dạng chuyển các cuộc gọi thủ tục từ xa (remote Procedure Calls, gọi tắt là RPC) thông qua SOAP (Simple Object Access Protocol). XML đã thật sự trở thành tiếng nói chung của các ngôn ngữ máy tính.

Lần đầu tiên Microsoft đã thêm sự hỗ trợ XML vào SQL Server 2000, bắt đầu với sự hỗ trợ mệnh đề FOR XML như là thành phần của câu lệnh SELECT và hàm OpenXML. Khi XML tiếp tục phát triển và nhanh chóng được chấp nhận bởi các doanh nghiệp, Microsoft lập tức cung cấp thêm tính năng bổ sung bằng cách đưa ra một loạt các ấn bản web. SQL dành cho XML 1.0 đã bổ sung thêm sự hỗ trợ cho UpdateGrams, Templates, và BulkLoad vào ấn bản nền tảng SQL Server 2000. Hai ấn bản web kế tiếp, SQLXML 2.0 và SQLXML 3.0, cải tiến thêm sản phẩm SQL Server 2000 bằng cách bổ sung sự hỗ trợ cho XML Views và SOAP ngoài nhiều tính năng mới khác. Tuy sự hỗ trợ của SQL Server 2000 cho XML đã đem lại một sự khởi đầu tốt cho việc hợp nhất các tài liệu XML phân cấp với dữ liệu quan hệ của SQL Server, nhưng nó có một số hạn chế. Ngay khi dữ liệu XML đã được chứa trong một cơ sở dữ liệu SQL Server nhờ dùng kiểu dữ liệu Text hay Image thì hầu như không còn điều gì để bạn có thể làm với nó. SQL Server 2000 không thể truy vấn dữ liệu phân cấp vốn tạo thành tài liệu XML mà không cần sử dụng T-SQL phức tạp hay mã phía client.

SQL Server 2005 được xây dựng dựa trên điểm khởi đầu này bằng cách tăng thêm sự hỗ trợ cho nhiều tính năng XML mới. Ở một cấp độ cao, SQL Server 2005 cung cấp một mức lưu trữ thống nhất mới cho XML và dữ liệu quan hệ. SQL Server 2005 bổ sung một kiểu dữ liệu XML mới vốn hỗ trợ cả hai mẫu truy vấn XML gốc cũng như tạo kiểu dữ liệu mạnh bằng cách kết hợp kiểu dữ liệu XML với một XSD (Extensible Schema Definition). Ngoài ra, nó còn cung cấp sự ánh xạ hai hướng giữa dữ liệu quan hệ và dữ liệu XML. Sự hỗ trợ XML còn được hợp nhất rất tốt vào bộ máy cơ sở dữ liệu quan hệ SQL Server 2005, vì nó cung cấp sự hỗ trợ cho các trigger trên XML, sự sao lập dữ liệu XML

và tải khối dữ liệu XML, cũng như sự hỗ trợ cải tiến cho sự truy cập dữ liệu thông qua SOAP và nhiều cải tiến khác. Trong chương này, bạn sẽ được giới thiệu về những tính năng XML mới quan trọng nhất trong SQL Server 2005.

Kiểu dữ liệu XML gốc

Không còn nghi ngờ, cải tiến XML quan trọng nhất mà Microsoft đã thêm vào SQL Server 2005 là sự hỗ trợ cho một kiểu dữ liệu XML gốc mới. Kiểu dữ liệu XML, gọi đơn giản là XML có thể được dùng làm một cột cho một table hay một biến hoặc thông số trong một thủ tục lưu trữ. Nó có thể được sử dụng để chứa cả dữ liệu đã tạo kiểu và chưa tạo kiểu. Nếu dữ liệu chứa trong một cột XML không có schema XSD thì nó được xem là chưa tạo kiểu (untyped). Nếu có một schema XSD kết hợp thì SQL Server 2005 sẽ kiểm tra schema đó để bảo đảm nơi chứa dữ liệu tuân theo định nghĩa schema. Trong mọi trường hợp, SQL Server 2005 kiểm tra dữ liệu vốn được chứa trong kiểu dữ liệu XML để bảo đảm tài liệu XML được hình thành khá tốt. Nếu dữ liệu không được hình thành tốt, SQL Server 2005 sẽ đưa ra một lỗi và dữ liệu sẽ không được chứa. Kiểu dữ liệu XML có thể chấp nhận tối đa 2GB dữ liệu và được chứa như kiểu dữ liệu varbinary(max). Danh sách sau đây minh họa việc tạo một table đơn giản dùng kiểu dữ liệu XML mới cho một trong các cột của nó.

```
CREATE TABLE MyXMLDocs
```

```
(DocID INT PRIMARY KEY IDENTITY,
```

```
MyXmlDoc XML)
```

Điều quan trọng nhất cần lưu ý trong ví dụ này là định nghĩa của cột MyXMLDoc, vốn dùng kiểu dữ liệu XML để chỉ định cột sẽ chứa dữ liệu XML. Bạn có thể chứa dữ liệu XML vào một cột XML nhờ dùng câu lệnh T-SQL INSERT chuẩn. Ví dụ dưới đây cho bạn thấy cách tập hợp một cột XML nhờ dùng một câu lệnh INSERT đơn giản:

```
INSERT INTO MyXmlDocs Values
```

```
('<MyXMLDoc>
```

```
<DocumentID>1</DocumentID>
```

```
<DocumentText>Text</DocumentText>
```

```
</MyXMLDoc>')
```

Lưu ý

Một điểm quan trọng cần lưu ý ở đây là bởi vì dữ liệu XML chưa được tạo kiểu nên bất kỳ tài liệu XML hợp lệ có thể được chèn vào kiểu dữ liệu XML.

Các kiểu dữ liệu XML được tạo kiểu mạnh

Kiểu dữ liệu XML gốc kiểm tra để bảo đảm bất kỳ dữ liệu chứa trong một cột hay biến XML là một tài liệu XML gốc. Trên phương diện riêng của nó, nó không kiểm tra bất kỳ thứ gì ngoài việc đó. Tuy nhiên, Microsoft đã thiết kế kiểu dữ liệu XML để có thể hỗ trợ nhiều sự hợp chuẩn tài liệu phức tạp nhờ dùng một schema XSD. Khi một schema XSD được xác định cho một cột kiểu dữ liệu XML, bộ máy SQL Server sẽ kiểm tra để bảo đảm mọi dữ liệu chứa trong cột XML đều tuân theo định nghĩa nằm trong schema XSD.

Danh sách sau đây minh họa một schema XSD mẫu cho tài liệu XML đơn giản đã được sử dụng trong ví dụ trước:

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified" targetNamespace="MyXMLDocSchema"
  xmlns="MyXMLDocSchema">
  <xs:element name="MyXMLDoc">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="DocumentID" type="xs:string" />
        <xs:element name="DocumentBody" type="xs:string" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Schema XSD này dùng namespace của MyXMLDoc Schema và định nghĩa một tài liệu XML vốn có một phần tử phức tạp được gọi là MyXMLDoc. Phần tử phức tạp MyXMLDoc có chứa hai phần tử đơn giản. Phần tử đơn giản đầu tiên phải được gọi là DocumentID và phần tử đơn giản thứ hai được gọi là DocumentBody. Cả hai phần tử này đều có chứa dữ liệu kiểu chuỗi XML.

Để tạo một cột hay biến XML có kiểu mạnh, trước tiên bạn cần đăng ký schema XSD với SQL Server nhờ dùng câu lệnh CREATE XML SCHEMA T-SQL DDL. Danh sách sau đây minh họa cách bạn kết hợp

câu lệnh CREATE XML SCHEMA COLLECTION với MyXMLDocSchema mẫu để đăng ký schema với cơ sở dữ liệu SQL Server 2005.

```
CREATE XML SCHEMA COLLECTION MyXMLDocSchema AS
N'<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" targetNamespace="http://MyXMLDocSchema">
  <xs:element name="MyXMLDoc">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="DocumentID" type="xs:string" />
        <xs:element name="DocumentBody" type="xs:string" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>'
```

Câu CREATE XML SCHEMA COLLECTION DDL lấy một đối số đơn đặt tên tập hợp. Kế tiếp, sau mệnh đề AS nó mong đợi một schema XSD hợp lệ được đặt trong các dấu trích dẫn đơn. Nếu schema không hợp lệ thì một lỗi sẽ xảy ra khi câu lệnh được thực thi. Câu lệnh CREATE XML SCHEMA COLLECTION là thuộc về cơ sở dữ liệu, và schema được đăng ký có thể được truy cập chỉ trong cơ sở dữ liệu mà schema được đăng ký cho nó.

Ngay sau khi bạn đã đăng ký schema XML với SQL Server 2005, bạn có thể tiếp tục kết hợp các biến XML và các cột với schema đó. Việc làm như vậy bảo đảm bất kỳ tài liệu XML chứa trong các biến hay các cột đó sẽ gắn liền với định nghĩa được cung cấp bởi schema kết hợp. Ví dụ sau đây minh họa cách bạn có thể tạo một table dùng một cột XML đã được tạo kiểu mạnh:

```
CREATE TABLE MyXMLDocs
(DocID INT PRIMARY KEY IDENTITY,
MyXmlDoc XML(MyXMLDocSchema))
```

Ở đây bạn có thể thấy table MyXMLDoc được tạo nhờ dùng câu lệnh CREATE TABLE gần giống như trong ví dụ trước. Tuy nhiên, trong trường hợp này, cột MyXMLDoc được tạo nhờ dùng một đối số chỉ định tên đó của định nghĩa schema XSD đã đăng ký. Nếu bạn xem danh sách trước, bạn có thể thấy schema đã được đăng ký nhờ dùng tên

MyXMLDocSchema. Sau khi cột MyXMLDoc đã được kết hợp với schema vừa được đăng ký, bất kỳ dữ liệu được chèn vào cột này sẽ được tạo kiểu mạnh theo định nghĩa schema và mọi nỗ lực chèn dữ liệu không phù hợp với định nghĩa schema sẽ bị loại ra. Danh sách sau đây minh họa một câu lệnh INSERT có thể thêm dữ liệu vào cột MyXMLDoc được tạo kiểu mạnh:

```
INSERT INTO MyXMLDocs Values
```

```
('<MyXMLDoc xmlns="http://MyXMLDocSchema">
  <DocumentID>1</DocumentID>
  <DocumentBody>"My text"</DocumentBody>
</MyXMLDoc>')
```

Lưu ý

Bởi vì ví dụ này dùng một kiểu dữ liệu XML được tạo kiểu (type), nên dữ liệu phải tuân theo định nghĩa do schema XSD kết hợp cung cấp.

Trong trường hợp này, tài liệu XML phải tham chiếu namespace XML kết hợp http://MyXMLDocSchema. Và tài liệu XML phải chứa một phần tử phức tạp được gọi là MyXMLDoc, đến lượt phần tử này chứa các phần tử DocumentID và DocumentBody. Bộ máy SQL Server sẽ từ chối bất kỳ nỗ lực nào chèn các tài liệu XML khác vào cột MyXMLDoc nếu dữ liệu không phù hợp với schema XSD được cho thì SQL Server sẽ trả về một thông báo lỗi giống như thông báo được minh họa trong danh sách dưới đây:

```
Msg 6965, Level 16, State 1, Line 1
```

```
XML Validation: Invalid content,expected
```

```
element(s):MyXMLDocSchema:DocumentID where element 'MyXMLDocSchema:Do' was
specified
```

Lưu ý

Như bạn có thể thấy từ mối quan hệ phụ thuộc của chúng, nếu bạn gán một schema cho một cột trong một table thì table đó phải được thay đổi hay bị loại ra trước khi định nghĩa schema đó có thể được cập nhật.

Truy xuất một schema XML đã đăng ký

Ngay sau khi bạn nhập một schema nhờ dùng CREATE XML SCHEMA COLLECTION, các thành phần schema được chứa trong metadata của SQL Server. Schema được chứa có thể được liệt kê bằng cách truy vấn khung xem hệ thống sys.xml_namespaces, như bạn có thể thấy trong ví dụ dưới đây:

```
SELECT * FROM sys.xml_namespaces
```

Câu lệnh này sẽ cho ra một tập hợp kết quả với tất cả những schema đã đăng ký trong một cơ sở dữ liệu như minh họa dưới đây:

xml_collection_id	name	xml_namespace_id
1	http://www.w3.org/2001/XMLSchema	1
65540	http://MyXMLDocSchema	1

(2 row(s) affected)

Bạn cũng có thể dùng hàm XML_SCHEMA_NAMESPACE mới để truy xuất schema XML. Mẫu truy vấn dưới đây truy xuất một schema từ cơ sở dữ liệu cho một namespace đã cho:

```
SELECT XML_SCHEMA_NAMESPACE (N 'dbo', N'MyXMLDocSchema')
```

Câu lệnh này sẽ cho ra một tập hợp kết quả với mọi cột sử dụng schema đã đăng ký, như bạn có thể thấy trong danh sách dưới đây:

```
-----
<xsd:schema xmlns:xsd=http://www.w3.org/2001/XMLSchema
targetNamespace="http://MyXMLDocSchema" xmlns:t=http://MyXMLDocSchema
elementFormDefault="qualified"><xsd:elementname="MyXMLDoc">
<xsd:complexType><xsd:complexContent><xsd:restriction base="xsd:any
(1 row(s) affected)
```

Thuộc tính elementname liệt kê những cột dùng kiểu dữ liệu XML đã được tạo kiểu.

Các phương thức kiểu dữ liệu XML

SQL Server 2005 cung cấp nhiều phương thức cài sẵn hoạt động gần giống như các kiểu dữ liệu do người dùng ấn định để làm việc với kiểu dữ liệu XML. Các phương thức này cho phép bạn đi sâu vào nội dung của các tài liệu XML vốn được lưu trữ nhờ dùng kiểu dữ liệu XML. Khả năng lưu trữ dữ liệu XML trong SQL Server có mặt hạn chế riêng của nó. Để thật sự dễ dàng hợp nhất XML chuyên sâu, bạn cũng cần một cách để truy vấn và xử lý dữ liệu vốn được lưu trữ nhờ dùng kiểu dữ liệu

XML mới, và đó chính là điều mà các phương thức kiểu dữ liệu XML sau đây cho phép:

Exists(XQuery, node ref)

Phương thức Exists của kiểu dữ liệu XML cho phép bạn kiểm tra các nội dung của một tài liệu XML để xem sự tồn tại của các phần tử hay thuộc tính nhờ dùng một biểu thức XQuery. (Ngôn ngữ XQuery được trình bày chi tiết hơn trong mục kế tiếp.) Danh sách sau đây minh họa cách sử dụng phương thức Exists của kiểu dữ liệu XML.

```
SELECT * FROM MyXMLDocs
```

```
WHERE MyXmlDoc.exist('declare namespace xd=http://MyXMLDocSchema
```

```
/xd:MyXMLDoc[xd:DocumentID eq "1"]') = 1
```

Thông số đầu tiên của phương thức XML Exists là bắt buộc và nhận một biểu thức XQuery. Thông số thứ hai là tùy ý và chỉ định một phần tham chiếu nút bên trong tài liệu XML. Ở đây XQuery kiểm tra một phần tử DocumentID bằng một trị là 1. Một namespace được khai báo bởi vì cột MyXMLDoc có một schema kết hợp. Phương thức Exists có thể trả về trị là TRUE (1) nếu các biểu thức XQuery trả về một nút. FALSE (0) nếu biểu thức không trả về một nút XML, hoặc NULL nếu instance kiểu dữ liệu XML là NULL. Bạn có thể xem kết quả của phương thức XML Exists ở đây:

```
DocID      MyXmlDoc
```

```
-----
1          <MyXMLDocxmlns="http://MyXMLDocSchema"><DocumentID>1
</DocumentID> <DocumentBody>"My text"</DocumentBody></MyXMLDoc>
(1 row(s) affected)
```

Modify(XML DML)

Phương thức Modify của kiểu dữ liệu XML cho phép bạn chỉnh sửa một tài liệu XML đã được lưu trữ. Bạn có thể dùng phương thức Modify để cập nhật toàn bộ tài liệu XML hoặc để cập chỉ phần được chọn của tài liệu. Bạn có thể xem một ví dụ về cách sử dụng phương thức Modify trong danh sách dưới đây:

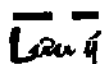
```
UPDATE MyXMLDocs
```

```
SET MyXMLDoc.modify('declare namespace xd=http://MyXMLDocSchema
```

```
replace value of (/xd:MyXMLDoc/xd:DocumentBody)[1] with "My New Body")
```

```
WHERE DocID = 1
```

Phương thức Modify của kiểu dữ liệu XML sử dụng một câu lệnh XML Data Modification Language (XML DML) làm thông số của nó. XML DML là một phần mở rộng Microsoft cho ngôn ngữ XQuery vốn cho phép chỉnh sửa các tài liệu XML. Phương ngữ XQuery hỗ trợ trị Replace của các câu lệnh Insert và Delete XML DML. Trong ví dụ này, vì cột MyXMLDoc XML được tạo kiểu nên câu lệnh XML DML phải chỉ định namespace cho schema. Kế tiếp, bạn có thể thấy nơi mà trị Replace của lệnh XML DML được sử dụng để thay trị của phần tử DocumentBody bằng trị mới "My New Body" cho hàng mà tại đó cột DocID quan hệ bằng một. Trị thay thế của mệnh đề phải nhận biết chỉ một nút duy nhất, nếu không nó sẽ thất bại. Do đó, nút đầu tiên được nhận biết nhờ dùng ký hiệu [1].



Tuy ví dụ này minh họa việc thực hiện một phép thay thế nhưng phương thức Modify cũng hỗ trợ các phép chèn và xóa.

Query(XQuery, node ref)

Phương thức Query của kiểu dữ liệu XML có thể truy xuất toàn bộ nội dung của một tài liệu XML hoặc phần được chọn trong tài liệu XML. Bạn có thể xem một ví dụ minh họa cách sử dụng phương thức Query trong danh sách sau:

```
SELECT DocID, MyXMLDoc.query('declare namespace xd=http://MyXMLDocSchema
    /xd:MyXMLDoc/xd:DocumentBody') AS Body
FROM MyXMLDocs
```

Biểu thức XQuery này trả về các trị từ phần tử DocumentBody của tài liệu XML. Một lần nữa, namespace được chỉ định bởi vì kiểu dữ liệu MyXMLDoc có một schema kết hợp được gọi là MyXMLDocSchema. Trong ví dụ này, bạn có thể thấy cách SQL Server 2005 hợp nhất dữ liệu cột quan hệ với dữ liệu XML dễ dàng như thế nào. Ở đây, DocID xuất phát từ một cột quan hệ, trong khi phần tử DocumentBody được truy vấn từ cột XML. Danh sách sau đây minh họa kết quả của XQuery:

DocID Body

-
- | | |
|---|---|
| 1 | <xd:DocumentBody xmlns:xd="http://MyXMLDocSchema">My New Body</xd:DocumentBody> |
| 2 | <xd:DocumentBody xmlns:xd="http://MyXMLDocSchema">"My 2nd text"</xd:DocumentBody> |

(2 row(s) affected)

Value(XQuery, node ref)

Phương thức Value cho phép trích các trị vô hướng từ một kiểu dữ liệu XML. Bạn có thể xem một ví dụ về cách sử dụng phương thức Value của kiểu dữ liệu XML trong danh sách sau:

```
SELECT MyXMLDoc.value('declare namespace xd=http://MyXMLDocSchema
(/xd:MyXMLDoc/xd:DocumentID)[1]', 'int') AS ID

FROM MyXMLDocs
```

Không giống như các phương thức kiểu dữ liệu XML khác, phương thức XML Value yêu cầu hai thông số. Thông số đầu tiên là một biểu thức XQuery và thông số thứ hai chỉ định kiểu dữ liệu SQL sẽ chứa trị vô hướng được trả về bởi phương thức Value. Ví dụ này cho ra tất cả những trị chứa trong phần tử DocumentID và chuyển đổi chúng sang kiểu dữ liệu int, như minh họa trong kết quả dưới đây:

ID

1

2

(2 row(s) affected)

Sự hỗ trợ XQuery

Trong mục trước, bạn đã thấy cách XQuery được sử dụng trong các phương thức của kiểu dữ liệu XML mới. XQuery được tạo dựa vào ngôn ngữ XPath của W3C (www.w3c.org) để truy vấn dữ liệu XML. XQuery mở rộng ngôn ngữ XPath bằng cách thêm khả năng cập nhật dữ liệu cũng như hỗ trợ việc rảo lập và phân loại kết quả tốt hơn. Vào lúc viết bài này, ngôn ngữ XQuery do SQL Server 2005 sử dụng là phần thực thi đầu tiên dựa vào một bản phác thảo đang hoạt động của chuẩn XQuery được gởi lên W3C, vì vậy có khả năng một số chi tiết thực thi sẽ thay đổi trước khi SQL Server 2005 chính thức được ấn hành. Để tìm hiểu thêm về chuẩn W3C XQuery, bạn có thể xem <http://www.w3c.org/XML/Query>.

Các index XML

Kiểu dữ liệu XML hỗ trợ tối đa 2GB lưu trữ, đây là một dung lượng rất lớn. Kích cỡ của dữ liệu XML và việc sử dụng nó có thể tác động rất lớn đến hiệu suất mà hệ thống đạt được trong khi truy vấn XML. Để cải

thiện hiệu suất của các mẫu truy vấn XML, SQL Server 2005 cung cấp khả năng tạo các index qua các cột có kiểu dữ liệu XML.

Các index XML chính

Để tạo một index XML trên một cột kiểu dữ liệu XML, một khóa chính được nhóm cụm, tồn tại cho table. Ngoài ra, nếu bạn cần thay đổi khóa chính cho table, trước tiên bạn phải xóa index XML. Một index XML bao hàm mọi phần tử trong cột XML và bạn có thể chỉ có một index XML trên mỗi cột. Bởi vì các index XML dùng namespace giống như các index quan hệ SQL Server bình thường nên các index XML không thể có cùng tên với một index hiện có. Các index XML có thể được tạo chỉ trên các kiểu dữ liệu XML trong một table. Chúng không thể được tạo trên các cột trong các khung xem hay trên các biến kiểu dữ liệu XML. Một index XML chính gồm có một dạng biểu diễn nhất quán của dữ liệu trong cột XML. Mã để tạo một index XML chính được minh họa trong danh sách dưới đây:

```
CREATE PRIMARY XML INDEX MyXMLDocsIdx ON MyXMLDocs (MyXMLDoc)
```

Ví dụ này minh họa việc tạo một index XML chính được gọi là MyXMLDocsIdx. Index này được tạo trên cột kiểu dữ liệu XML MyXMLDoc trong table MyXMLDocs. Cũng giống như các index SQL Server bình thường, các index XML có thể được xem bằng cách truy vấn khung xem sys.indexes.

```
SELECT * FROM sys.indexes WHERE name = MyXMLDocsIdx
```

Các index XML phụ

Ngoài index chính, bạn cũng có thể tạo các index XML phụ. Các index phụ được tạo dựa trên một trong những đặc tính tài liệu sau đây:

- ❖ **Path** (đường dẫn) Đường dẫn tài liệu được sử dụng để tạo index.
- ❖ **Value** (trị) Các trị tài liệu được sử dụng để tạo index.
- ❖ **Property** (thuộc tính) Các thuộc tính tài liệu được sử dụng để tạo index.

Các index phụ luôn được phân chia theo cách giống như index XML chính. Danh sách sau đây minh họa việc tạo một index XML đường dẫn phụ:

```
CREATE XML INDEX My2ndXMLDocsIdx ON MyXMLDocs(MyXMLDoc)
```

```
USING XML INDEX MyXMLDocsIdx FOR PATH
```

Các cải tiến FOR XML

Mệnh đề FOR XML lần đầu tiên được giới thiệu cho câu lệnh T-SQL SELECT trong SQL Server 2000. Nó đã được cải tiến trong SQL Server 2005. Một số khả năng mới được tìm thấy trong phần hỗ trợ FOR XML trong SQL Server 2005 là hỗ trợ kiểu dữ liệu XML, các kiểu dữ liệu do người dùng ấn định, kiểu dữ liệu timestamp và sự hỗ trợ cải tiến cho dữ liệu chuỗi. Ngoài ra, các cải tiến FOR XML cũng bao gồm sự hỗ trợ cho một chỉ thị Type mới, các mẫu truy vấn FOR XML xếp lồng, và tạo schema XSD nội dòng.

Chỉ thị Type

Khi các kiểu dữ liệu XML được trả về nhờ dùng chỉ thị Type của các mệnh đề FOR XML, chúng không được chuỗi hóa. Thay vào đó các kết quả được trả về ở dạng một kiểu dữ liệu XML. Bạn có thể xem một ví dụ về cách dùng mệnh đề FOR XML với chỉ thị XML Type ở đây:

```
SELECT DocID, MyXMLDoc FROM MyXMLDocs
```

```
WHERE DocID=1 FOR XML AUTO, TYPE
```

Mẫu truy vấn này trả về cột DocID quan hệ cùng với cột kiểu dữ liệu XML MyXMLDoc. Nó dùng mệnh đề FOR XML AUTO để trả về kết quả ở dạng XML. Chỉ thị Type chỉ định kết quả sẽ được trả về ở dạng một kiểu dữ liệu XML. Bạn có thể thấy kết quả của việc sử dụng chỉ thị Type ở đây:

```
<MyXMLDocs DocID="1">
```

```
<MyXMLDoc>
```

```
<MyXMLDoc xmlns="MyXMLDocSchema">
```

```
<DocumentID>1</DocumentID>
```

```
<DocumentBody>My New Body</DocumentBody>
```

```
</MyXMLDoc>
```

```
</MyXMLDoc>
```

```
</MyXMLDocs>
```

Lưu ý

Chỉ thị Type trả về kiểu dữ liệu XML ở dạng một dòng liên tục. Sự định dạng được thêm vào danh sách trước để làm cho nó dễ đọc hơn.

Các mẫu truy vấn FOR XML xếp lồng

SQL Server 2000 giới hạn việc sử dụng mệnh đề FOR XML ở mức cao nhất của một mẫu truy vấn (query). Các mẫu truy vấn con không thể sử dụng mệnh đề FOR XML. SQL Server 2005 thêm khả năng sử dụng các mẫu truy vấn FOR XML xếp lồng. Các mẫu truy vấn xếp lồng hữu dụng cho việc trả về nhiều hạng mục trong đó có một mối quan hệ bố con (parent-child). Một ví dụ về kiểu quan hệ này có thể là các bảng ghi mục đặt hàng và chi tiết đặt hàng; một ví dụ khác có thể là loại sản phẩm và từng loại nhỏ. Bạn có thể xem một ví dụ về cách sử dụng mệnh đề FOR XML xếp lồng trong danh sách sau:

```
SELECT DocID, MyXMLDoc,
```

```
(SELECT MyXMLDoc
```

```
FROM MyXMLDocs2
```

```
WHERE MyXMLDocs2.DocID = MyXMLDocs.DocID
```

```
FOR XML AUTO, TYPE)
```

```
FROM MyXMLDocs Where DocID = 2 FOR XML AUTO, TYPE
```

Trong ví dụ này, mẫu truy vấn ngoài cùng trên table MyXMLDoc được kết hợp với một mẫu truy vấn con trên table MyXMLDoc2 (đối với ví dụ ở đây, đó là sự sao lập đơn giản của table MyXMLDoc). Điều quan trọng cần lưu ý trong danh sách này là khả năng sử dụng mệnh đề FOR XML trong mẫu truy vấn con của SQL Server 2005. Trong trường hợp này, mẫu truy vấn con đang sử dụng chỉ thị Type để cho ra kết quả ở dạng một kiểu dữ liệu XML gốc. Nếu chỉ thị Type không được sử dụng thì kết quả sẽ được trả về ở dạng một kiểu dữ liệu nvarchar và kiểu dữ liệu XML sẽ được thực thể hóa. Bạn có thể xem kết quả của mẫu truy vấn FOR XML xếp lồng trong danh sách minh họa dưới đây:

```
<MyXMLDocs DocID="2">
```

```
<MyXMLDoc>
```

```
<MyXMLDoc xmlns="MyXMLDocSchema">
```

```
<DocumentID>1</DocumentID>
```

```
<DocumentBody>&quot;My text&quot;</DocumentBody>
```

```
</MyXMLDoc>
```

```
</MyXMLDoc>
```

```
</MyXMLDocs>
```



```

<MyXMLDoc xmlns="http://MyXMLDocSchema">
  <DocumentID>1</DocumentID>
  <DocumentBody>My New Body</DocumentBody>
</MyXMLDoc>
</MyXMLDoc>
</MyXMLDocs>

```

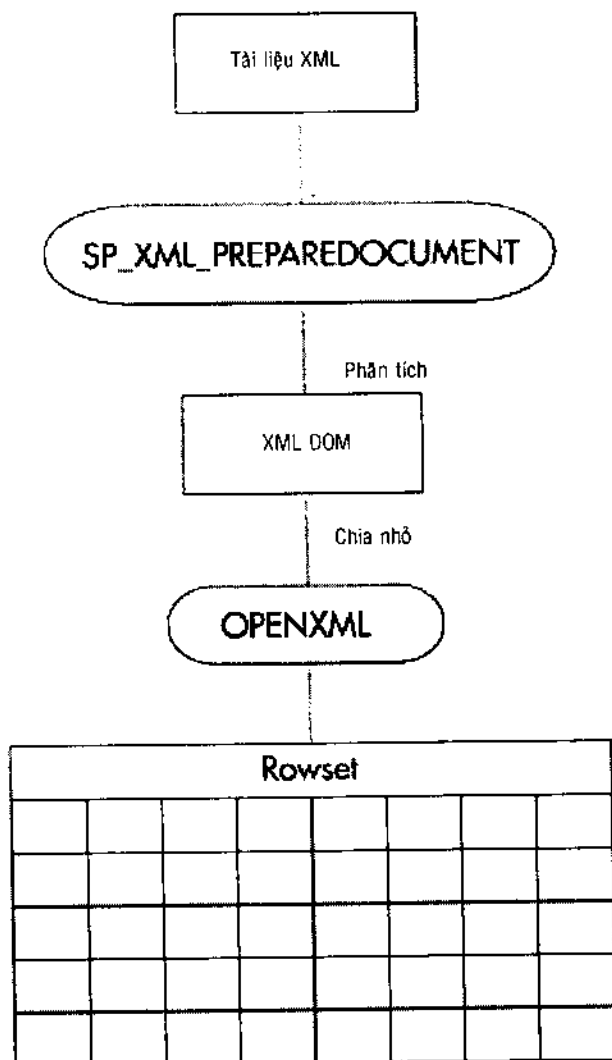
Chỉ thị XML SCHEMA có thể cho ra nhiều schema, nhưng nó luôn cho ra ít nhất là hai: một schema được trả về cho namespace `SqlTypes` và một schema thứ hai được trả về mô tả kết quả của mẫu truy vấn FOR XML. Trong danh sách trước, bạn có thể thấy phần mô tả schema của cột kiểu dữ liệu XML bắt đầu tại: `<xsd:element name="MyXMLDocs">`. Kế tiếp, các kết quả XML được minh họa tại dòng bắt đầu với `<MyXMLDocs xmlns="urn:schemas-microsoft-com:sql:SqlRowSet">`.

Lưu ý

Bạn có thể tạo một schema XDR (XML Data Reduced) bằng cách dùng chỉ thị `XMLDATA` kết hợp với mệnh đề FOR XML.

Các cải tiến OPENXML

Mệnh đề FOR XML thì kết hợp cho việc truy xuất XML từ cơ sở dữ liệu SQL Server 2005. Về cơ bản, mệnh đề FOR XML tạo ra một tài liệu XML từ dữ liệu quan hệ. Từ khóa OPENXML là đối chiếu của mệnh đề FOR XML. Hàm OPENXML cung cấp một rowset quan hệ qua một tài liệu XML. Để sử dụng tính năng OPENXML của SQL Server, trước tiên bạn phải gọi thủ tục lưu trữ `sp_xml_preparedocument`, thủ tục này phân tích tài liệu XML nhờ dùng XML Document Object Model (DOM) và trả về một handle cho OPENXML. Sau đó, OPENXML cung cấp một khung xem rowset của tài liệu XML đã được phân tích. Khi bạn làm việc xong với tài liệu, bạn gọi thủ tục lưu trữ `sp_xml_removedocument` để giải phóng các tài nguyên hệ thống bị chiếm bởi OPENXML và XML DOM. Hình 7.1 minh họa tổng quan về tiến trình này.



Hình 7.1 Tổng quan về OPENXML

Với SQL Server 2005, sự hỗ trợ OPENXML đã được mở rộng để trợ thêm kiểu dữ liệu XML mới và kiểu dữ liệu do người dùng ấn định. Ví dụ sau đây minh họa cách bạn có thể dùng OPENXML kết hợp với một mệnh đề WITH và kiểu dữ liệu XML mới:

```

DECLARE @hdocument int
DECLARE @doc varchar(1000)
SET @doc = '<MyXMLDoc>
    <DocumentID>1</DocumentID>
    <DocumentBody>"OPENXML Example"</DocumentBody>
  
```

Ở đầu danh sách này, bạn có thể thấy nơi khai báo hai biến. Biến @hdocument sẽ được sử dụng để chứa handle tài liệu XML được trả về bởi thủ tục lưu trữ sp_xml_preparedocument, trong khi biến @doc sẽ chứa chính tài liệu XML mẫu. Kế tiếp, thủ tục lưu trữ sp_xml_preparedocument được thực thi và được chuyển hai biến. Thủ tục lưu trữ sp_xml_preparedocument dùng XML DOM để phân tích tài liệu XML rồi trả về một handle cho tài liệu đã phân tích trong biến @hdocument. Handle tài liệu đó sau đó được chuyển từ khóa OPENXML trong câu lệnh SELECT.

Thông số đầu tiên do OPENXML sử dụng là handle tài liệu chứa trong biến @hdocument. Thông số thứ hai là một mẫu XPath chỉ định các mức trong tài liệu XML sẽ tạo rowset quan hệ. Thông số thứ ba chỉ định kiểu ánh xạ XML sang quan hệ sẽ được thực hiện. Trị 2 cho biết ánh xạ tập trung vào phần tử sẽ được sử dụng. Trị 1 cho biết ánh xạ tập trung vào thuộc tính sẽ được thực hiện. Mệnh đề WITH cung cấp dạng của rowset được trả về. Trong ví dụ này, mệnh đề WITH chỉ định rowset được trả về sẽ gồm có hai cột varchar được gọi là DocumentID và DocumentBody. Trong khi ví dụ này minh họa các tên rowset trùng với các phần tử XML nhưng điều đó không bắt buộc. Sau cùng, thủ tục lưu trữ sp_xml_removedocument được thực thi để giải phóng các tài nguyên hệ thống.

Câu lệnh SELECT này dùng tính năng OPENXML sẽ cho ra một rowset gồm có các trị phần tử từ tài liệu XML. Bạn có thể xem kết quả của việc dùng OPENXML trong danh sách sau đây:

DocumentID DocumentBody

1 "OPENXML Example"

(1 row(s) affected)

XML Bulk Load

Một trong những điều đầu tiên mà có lẽ bạn sẽ muốn thực hiện để tận dụng kiểu dữ liệu XML mới là tải các tài liệu XML của bạn vào các

cột XML từ đĩa. Tính năng XML bulk load mới giúp thực hiện công việc này một cách dễ dàng. Tính năng này cung cấp một cơ chế tốc độ cao để tải các tài liệu XML vào các cột SQL Server. Bạn có thể xem một ví dụ về cách dùng tính năng XML bulk load trong danh sách sau:

```
INSERT into MyXMLDocs(MyXMLDoc) SELECT * FROM OPENROWSET
```

```
(Bulk 'c:\temp\MyXMLDoc.xml', SINGLE_CLOB) as x
```

Trong ví dụ, câu lệnh INSERT được sử dụng để chèn kết quả của câu lệnh SELECT * FROM OPENROWSET và cột MyXMLDoc trong table MyXMLDocs. Hàm OPENROWSET sử dụng Bulk rowset provider cho OPENROWSET để đọc dữ liệu vào từ file C:\temp\MyXMLDoc.xml. Bạn có thể xem nội dung của file MyXMLDoc.xml trong danh sách sau:

```
<MyXMLDoc xmlns="http://MyXMLDocSchema">
```

```
<DocumentID>3</DocumentID>
```

```
<DocumentBody>"The Third Body"</DocumentBody>
```

```
</MyXMLDoc>
```

Nếu bạn thực thi lệnh này từ SQL Server Management Studio, bạn cần nhớ rằng nó sẽ được thực thi trên hệ thống SQL Server và do đó các phần tham chiếu file và đường dẫn phải được tìm thấy trên hệ thống server cục bộ. Đối số SINGLE_CLOB chỉ định dữ liệu từ file sẽ được chèn vào một hàng duy nhất. Nếu bạn bỏ qua đối số SINGLE_CLOB thì dữ liệu từ file có thể được chèn vào nhiều hàng. Theo mặc định, Bulk provider dành cho hàm OPENROWSET sẽ phân chia các hàng trên ký tự Carriage Return (trả về đầu dòng), đây là ký tự phân cách hàng mặc định. Cách khác, bạn cũng có thể chỉ định các dấu phân cách trường và hàng nhờ dùng các đối số tùy ý FIELDTERMINATOR và ROWTERMINATOR của hàm OPENROWSET.

Sự truy cập HTTP SOAP gốc

Một tính năng mới liên quan đến XML trong SQL Server 2005 là sự hỗ trợ HTTP SOAP gốc. Tính năng mới này cho phép SQL Server trả lời trực tiếp các yêu cầu HTTP/SOAP vốn được ấn hành bởi các dịch vụ web mà không cần yêu cầu một hệ thống IIS đóng vai trò trung gian. Với sự hỗ trợ HTTP SOAP gốc, bạn có thể tạo các dịch vụ web có khả năng thực thi các lô T-SQL, các thủ tục lưu trữ và các hàm vô hướng do người dùng ấn định. Để bảo đảm mức độ bảo mật cao, sự truy cập HTTP gốc được tắt theo mặc định. Tuy nhiên, bạn có thể kích hoạt sự hỗ trợ HTTP bằng cách trước tiên tạo một điểm cuối HTTP. Bạn có thể xem

một ví dụ về mã để tạo một điểm cuối (endpoint HTTP) trong danh sách sau:

```
CREATE ENDPOINT MyHTTPEndpoint
STATE = STARTED
AS HTTP(
    PATH = '/sql',
    AUTHENTICATION = (INTEGRATED ),
    PORTS = ( CLEAR ),
    SITE = 'server'
)
FOR SOAP (
    WEBMETHOD 'http://tempUri.org/'. 'GetProductName'
        (name='AdventureWorks.dbo.GetProductName',
        schema=STANDARD ),
    BATCHES = ENABLED,
    WSDL = DEFAULT,
    DATABASE = 'AdventureWorks',
    NAMESPACE = 'http://AdventureWorks/Products'
)
```

Ví dụ này minh họa việc tạo một điểm cuối HTTP được gọi là MyHTTPEndPoint cho thủ tục lưu trữ GetProductName trong cơ sở dữ liệu AdventureWorks mẫu. Ngay sau khi điểm cuối HTTP được tạo, nó có thể được truy cập qua một yêu cầu SOAP do một ứng dụng đưa ra. Bạn có thể dùng các câu lệnh ALTER ENDPOINT và DROP ENDPOINT DDL để quản lý các điểm cuối HTTP của SQL Server. Các điểm cuối HTTP mới cũng có thể cung cấp sự mã hóa dòng dữ liệu nhờ dùng SSL. Sự hỗ trợ HTTP của SQL Server đã được trình bày ở chương 2.

Lệnh sau đây minh họa cách liệt kê các điểm cuối HTTP đã được tạo:

```
select * from sys.http_endpoints
```

Các cải tiến XML cho server phân tích

Trong khi đa số các cải tiến XML trong SQL Server 2005 đã được thực thi cho bộ máy cơ sở dữ liệu quan hệ, Analysis Services cũng đã nhận được các tính năng quan trọng liên quan đến XML, quan trọng nhất là XML mới dành cho Analysis Services còn được gọi là XMLA. Trong mục này, bạn sẽ tìm hiểu thêm về XML mới dành cho tính năng Analysis Services.

XML dành cho Analysis Services

XML dành cho Analysis Services là một API cung cấp sự truy cập dữ liệu vào các nguồn dữ liệu Analysis Services trên web. XML dành cho Analysis Services được tạo mẫu theo OLE DB ở chỗ nó sẽ cung cấp một mô hình truy cập dữ liệu phổ quát cho bất kỳ nguồn dữ liệu đa hướng. Tuy nhiên, không giống như OLE DB dựa vào COM, XML dành cho Analysis Services được tạo dựa trên giao thức SOAP gốc XML. Cũng không giống như OLE DB, vốn được tạo với ý định mô hình client / server, XML dành cho Analysis Services được tối ưu hóa để sử dụng trên web.

XML dành cho Analysis Services cung cấp hai phương thức có thể được truy cập rộng rãi: phương thức Discover và phương thức Execute. Phương thức Discover lấy thông tin về một nguồn dữ liệu. Phương thức Discover có thể liệt kê thông tin về các nguồn dữ liệu có sẵn, các provider nguồn dữ liệu, và metadata có sẵn. Phương thức Execute cho phép một ứng dụng chạy các lệnh dựa trên XML dành cho các nguồn dữ liệu Analysis Services.

Các khung xem danh mục XML hệ thống

SQL Server 2005 chứa thông tin về XML vốn được sử dụng trong server trong nhiều khung xem hệ thống mới. Bảng 7.1 mô tả tóm tắt các khung xem XML hệ thống mới.

Bảng 7.1 Các khung xem danh mục XML của SQL Server 2005

Khung xem danh mục XML	Mô tả
sys.xml_attributes	Khung xem này cung cấp một hàng cho mỗi thuộc tính XML được lưu trữ.
sys.xml_components	Khung xem này cung cấp một hàng cho mỗi thành phần của một schema XML.
sys.xml_component_ placements	Khung xem này cung cấp một hàng cho mỗi vị trí của một thành phần XML.
sys.xml_elements	Khung xem này cung cấp một hàng cho mỗi thành phần XML vốn là một phần tử XML.
sys.xml_facets	Khung xem này cung cấp một hàng cho mỗi khía cạnh thuộc kiểu XML.
sys.xml_model_groups	Khung xem này cung cấp một danh sách tất cả những thành phần XML

vốn là thành phần của một Model Group.

`sys.xml_namespaces`

Khung xem này cung cấp một hàng cho mỗi namespace được xác định bởi XSD.

`sys.xml_types`

Khung xem này cung cấp một hàng cho mỗi thành phần XML vốn là một kiểu XML.

`sys.xml_wildcards`

Khung xem này cung cấp một hàng cho mỗi thuộc tính XML hay phần tử thay thế.

`sys.xml_wildcard_
namespaces`

Khung xem này cung cấp một hàng cho mỗi namespace thay thế XML.

PHẦN III

CÁC TÍNH NĂNG KINH DOANH THÔNG MINH

CHƯƠNG 8

Các dịch vụ báo cáo

Một trong những cải tiến quan trọng nhất trong SQL Server 2005 là hệ thống phụ Reporting Services mới. Reporting Services (các dịch vụ báo cáo) được giới thiệu lần đầu tiên như là một phần bổ sung vào SQL Server 2000 và che lấp một khoảng hở quan trọng trong dòng sản phẩm SQL Server. Kể từ lúc nó được giới thiệu, SQL Server luôn là một trong những nền cơ sở dữ liệu quan hệ dễ thực thi nhất và điều này đã làm cho nó trở nên phổ biến với các phần thực thi cấp bộ phận cũng như một nền cơ sở dữ liệu cho các doanh nghiệp cỡ vừa và nhỏ. Tuy nhiên, ngay sau khi SQL Server được cài đặt, nếu bạn muốn bắt đầu truy xuất thông tin từ cơ sở dữ liệu bằng cách tạo các báo cáo (report) thì bạn cần thoát ra và dùng một sản phẩm khác có các tính năng báo cáo cài sẵn để xuất dữ liệu đó ở dạng một report có thể xem được hay được in. SQL Server không có công cụ cài sẵn nào có khả năng tạo các report - kết xuất phổ biến và được mong đợi nhất từ một hệ thống cơ sở dữ liệu quan hệ. Để lấy một số report đầu tiên từ SQL Server, nhiều công ty đã bắt đầu dùng các công cụ tạo report desktop như Microsoft Access, nhưng sau cùng đa số các công ty cỡ vừa và lớn hơn sẽ chọn những sản phẩm tạo report mạnh hơn của hãng thứ ba, chẳng hạn như Business Object's Crystal Reports.

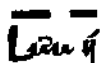
Sự ra đời của Reporting Services với SQL Server 2000 đã thay đổi tất cả những điều đó. Reporting Services cung cấp một giải pháp tạo report vượt xa các khả năng của những giải pháp đơn giản như Access. Reporting Services là một hệ thống tạo report cấp doanh nghiệp, nó không chỉ cung cấp khả năng thiết kế report bằng đồ họa mà còn cho phép bạn triển khai những report đó một cách an toàn qua doanh nghiệp được mô phỏng ở nhiều dạng khác nhau bao gồm cả các report HTML dựa trên web, các report client trên nền Windows, và các report được mô phỏng cho những thiết bị di động. SQL Server 2005 Reporting Services tiếp tục phát triển thành chuẩn mới cho việc lập báo cáo doanh nghiệp SQL

Server. Trong phần đầu tiên của chương này, bạn sẽ xem tổng quan về kiểu kiến trúc và các thành phần hệ thống do Reporting Services của SQL Server 2005 sử dụng. Sau đó trong phần thứ hai của chương, bạn sẽ xem xét cách thiết kế và triển khai các report nhờ dùng Reporting Services.

Kiểu kiến trúc Reporting Services

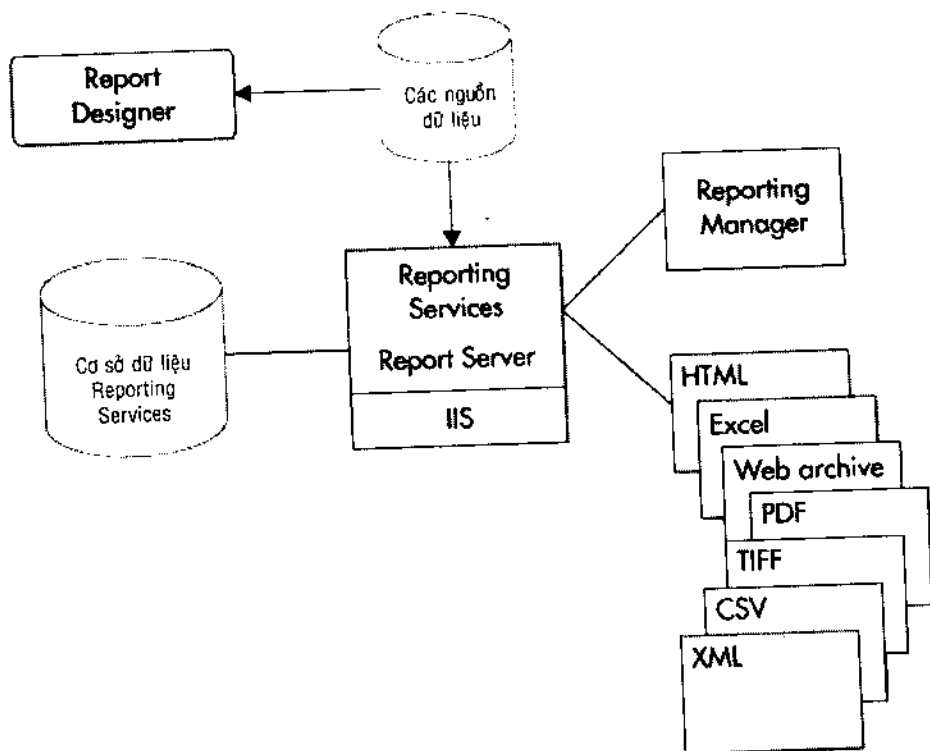
Reporting Services của SQL Server 2005 không chỉ là một công cụ thiết kế báo cáo (report). Nó là một nền tạo report không những cho phép tạo các report mà còn chứa các định nghĩa report, cung cấp sự truy cập an toàn vào các report, mô phỏng các report ở nhiều dạng kết xuất khác nhau, lập thời biểu chuyển phát report nhờ dùng một cơ chế đẩy hay kéo và cho phép triển khai các report đó.

Reporting Services là một thành phần được cài đặt tùy ý vốn đòi hỏi phải có IIS (Internet Information Services) để được cài đặt. Nếu IIS không có trên hệ thống đang chạy phần cài đặt, tùy chọn để cài đặt Reporting Services sẽ không có trên các hộp thoại cài đặt của SQL Server 2005. Trong khi Reporting Services có thể được cài đặt trên hệ thống server giống như bộ máy cơ sở dữ liệu SQL Server, nhưng để cải thiện khả năng mở rộng thì việc cài đặt Reporting Services trên một server riêng biệt thường tốt hơn.



Reporting Services được đăng ký như là thành phần của SQL Server 2005 và không đòi hỏi bất kỳ sự đăng ký riêng để sử dụng trên một hệ thống đơn. Tuy nhiên, nó yêu cầu đăng ký thêm nếu bạn thực thi nó trên một hệ thống riêng biệt.

SQL Server 2005 Reporting Services không chỉ là một ứng dụng đơn, nó còn là một hệ thống phụ chạy trên server vốn được thiết kế để cho phép tạo, quản lý và thực thi các report qua doanh nghiệp. Reporting Services có thể được sử dụng để tạo các report dạng bảng, dạng đồ họa và dạng tự do vốn có thể được xem và quản lý nhờ dùng một kết nối dựa trên web. Những báo cáo này có thể chạy trên một cơ sở dữ liệu SQL Server 2005 cũng như bất kỳ nguồn cơ sở dữ liệu nào tương thích OLE DB hay ODBC. Hình 8.1 minh họa tổng quan về kiểu kiến trúc Reporting Services.



Hình 8.1 Tổng quan về Reporting Services

Như bạn có thể thấy ở hình 8.1, Reporting Services được tạo nhờ dùng Report Designer. Reporting Services Report Designer có một bề mặt thiết kế hoàn toàn bằng đồ họa và chương trình giao diện cho phép bạn thiết kế và kiểm tra các report một cách tương tác. Không giống như Report Designer của chương trình Microsoft Access quen thuộc, Reporting Services Report Designer hoàn toàn không bị tạo dải, nó có một mức độ linh hoạt lớn hơn nhiều. Một Report Designer được tạo dải làm hạn chế những vùng nhất định của Report Designer vì những mục đích đã được ấn định trước; Chẳng hạn, phần trên cùng của trang luôn là tiêu đề, phần giữa của trang là nội dung báo cáo và phần dưới cùng dành cho tiêu đề cuối trang (footer). Trong khi nhiều report tuân theo kiểu này, bạn không có sự linh hoạt để làm bất kỳ điều gì khác. Kiểu trình bày không tạo dải của Reporting Services Report Designer cho bạn nhiều tự do hơn; chẳng hạn bạn có thể dễ dàng trình bày report của mình ở bốn góc phần tư của trang, mỗi phần dùng kiểu định dạng và dữ liệu riêng. Reporting Services Report Designer sẽ được trình bày chi tiết hơn trong phần thứ hai của chương này.

Sau khi một report được thiết kế nhờ dùng Report Designer, các định nghĩa report được chứa trong cơ sở dữ liệu Report Server. Các report Reporting Services được lưu trữ ở một dạng dữ liệu dựa vào XML mới được gọi là Report Design Layout (RDL). Theo mặc định, các định nghĩa RDL này được chứa trong cơ sở dữ liệu Report Server của SQL Server 2005. Ngoài các đặc tính RDL của report, cơ sở dữ liệu Report Server còn chứa thông tin về sự bảo mật và đích đến của một report.

Reporting Services Report Server, thành phần cốt lõi trong kiến trúc Reporting Services, là một ứng dụng ASP.NET chạy trên Windows Internet Information Server (IIS). Trách nhiệm chính của Report Server là xử lý các file định nghĩa report đã lưu trữ rồi mô phỏng các report ở dạng thích hợp cho cơ chế chuyển tải đã chỉ định. Ngoài các report chuẩn được in ra, Reporting Services Report Server còn mô phỏng các report ở các dạng kết xuất khác nhau sau đây:

- ❖ HTML
- ❖ Excel
- ❖ Web archive
- ❖ PDF
- ❖ TIFF
- ❖ CSV
- ❖ XML

Reporting Services được quản lý nhờ dùng Report Manager. Report Manager là một ứng dụng dựa trên web cho phép DBA hay người quản lý báo cáo kiểm soát sự bảo mật cũng như toàn bộ thuộc tính quản lý của các report được tạo nhờ dùng Reporting Services. Điều này bao gồm việc chỉ định những người nào có thể tạo và thay đổi một report cũng như những người nào có thể chạy một report đã cho. Report Manager cũng có thể cài đặt cả hai thời biểu chuyển tải ở dạng đẩy và kéo cho các report Reporting Services.

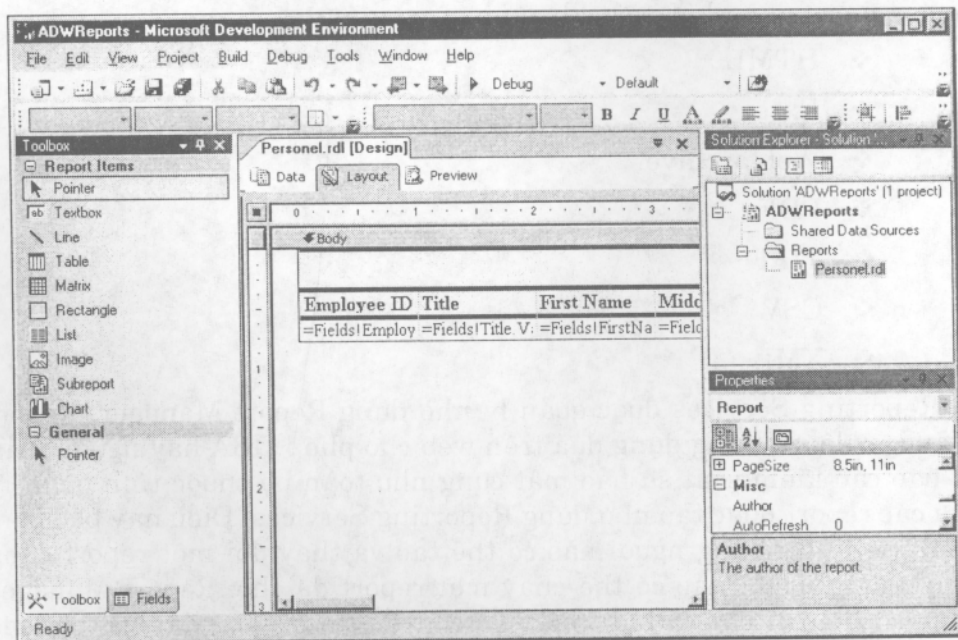
Trong mục kế tiếp, bạn sẽ xem xét chi tiết hơn về tính năng của mỗi thành phần này.

Các thành phần Reporting Services

Hệ thống Reporting Services mới của SQL Server 2005 gồm có nhiều thành phần liên quan với nhau. Trong mục này, bạn sẽ xem xét chi tiết hơn về mỗi thành phần này, bắt đầu với thành phần dễ thấy nhất, đó là Report Designer.

Report Designer

Reporting Services Report Designer cho phép bạn thiết kế các report bằng hình ảnh cũng như kiểm soát việc triển khai chúng. Không giống như phần bổ sung Reporting Services cho SQL Server 2000, vốn đòi hỏi Visual Studio để thiết kế các report, Reporting Services của SQL Server 2005 cung cấp Report Designer ở dạng thành phần của Business Intelligence Development Studio. Để khởi động Report Designer, trước tiên bạn mở Business Intelligence Studio của SQL Server 2005 rồi chọn File > New > Project > Report Project để tạo một dự án Reporting Services mới. Sau đó, để mở Report Designer, bạn chọn tùy chọn Project > Add New Item > Report. Hình 8.2 minh họa một ví dụ về Reporting Services Report Designer.



Hình 8.2 Reporting Services Report Designer

Toolbox

Cửa sổ Toolbox trong Reporting Services Report Designer được minh họa ở phía bên trái của màn hình trong hình 8.2. Toolbox được sử dụng để rê và thả các thành phần lên trên các bề mặt thiết kế tương ứng của chúng.

Các mục điều khiển report

Report Designer Toolbox có chứa các mục điều khiển (control) sau đây:

- ❖ **Textbox** Control Textbox cho phép bạn hiển thị dữ liệu text trên report của bạn. Textbox có thể được đặt bất kỳ nơi nào trên report và chứa dữ liệu cột, các nhãn và các trường đã được tính.
- ❖ **Line** Control Line cho phép bạn vẽ một đường thẳng trên kiểu trình bày report.
- ❖ **Table** Control Table cho phép bạn kết buộc một table với kiểu trình bày report.
- ❖ **Matrix** Bạn có thể sử dụng control Matrix để hiển thị một lưới trên kiểu trình bày report. Bạn có thể kết buộc control Matrix với dataset của report.
- ❖ **Rectangle** Control Rectangle được sử dụng chủ yếu như là một đối tượng chứa (container) cho những thành phần report khác. Nó cũng có thể được dùng làm thành phần đồ họa bởi control Line.
- ❖ **List** Control List cho phép bạn đặt một danh sách trên kiểu trình bày report. Danh sách này có thể được kết buộc vào các trường trong dataset của bạn.
- ❖ **Image** Control Image cho phép bạn kết buộc các ảnh nhị phân vào kiểu trình bày report. Các dạng được hỗ trợ là: bmp, .jpg, .gif, và png.
- ❖ **Subreport** Control Subreport được sử dụng để liên kết một mục của report với một report khác đã được ấn định trước. Subreport có thể là một report độc lập hoặc có thể được thiết kế để chạy trong một report khác.
- ❖ **Chart** Control Chart vẽ một biểu đồ trên kiểu trình bày report. Control Chart có thể được kết buộc với dataset của report và hỗ trợ nhiều loại biểu đồ khác nhau, bao gồm biểu đồ dạng cột, biểu đồ dạng thanh, biểu đồ hình bánh, biểu đồ bọt, biểu đồ diện tích, biểu đồ radar, biểu đồ chứng khoán, biểu đồ cực...

Design Surface

Ở chính giữa màn hình được minh họa ở hình 8.2, bạn có thể thấy Design Surface (bề mặt thiết kế) của Report Designer. Design Surface của Report Designer là nơi bạn tạo kiểu trình bày của report. Design

Surface có ba tab: Data, Layout, và Preview. Để tạo một report, trước tiên bạn phải xác định một dataset và bạn có thể làm điều đó nhờ dùng tab Data. Dataset được tạo bằng cách chỉ định nối kết cơ sở dữ liệu hoặc bằng cách nhập chuỗi nối kết thích hợp bằng tay hoặc bằng cách điền vào các dòng nhắc hiển thị trong hộp thoại Data Link. Ngay sau khi đã thiết lập nối kết, bạn định nghĩa mẫu truy vấn (query) của bạn hoặc bằng cách nhập câu lệnh SQL hoặc bằng cách dùng Query Designer dạng đồ họa. Query Designer cho phép bạn tạo một mẫu truy vấn SQL một cách trực quan bằng cách rê và thả các table rồi chọn các cột mong muốn, sau đó xác định bằng hình ảnh bất kỳ tiêu chuẩn nối và chọn hàng theo yêu cầu. Ngay sau khi mẫu truy vấn đã được thiết kế, một dạng biểu diễn bộ tạo mẫu truy vấn bằng đồ họa của mẫu truy vấn đó sẽ được hiển thị trên tab Data.

Tab Layout là nơi bạn thiết kế report của mình. Bạn thiết kế report bằng cách rê và thả các mục từ Toolbox lên trên ô Layout rồi di chuyển và định lại kích cỡ chúng. Như bạn có thể thấy ở hình 8.2, ô Layout hiển thị tất cả những control Reporting Services đã được thêm vào report cũng như các phần kết buộc của chúng.

Tab Preview cho phép bạn xem trước diện mạo của report đã được mô phỏng. Khi bạn nhấp tab Preview, Report Designer sẽ thực thi report và hiển thị nó trong tab Preview. Preview không cho phép bạn thực hiện sự thay đổi về cách thể hiện report. Để thay đổi report, bạn cần dùng tab Layout.

Solution Explorer

Bạn có thể thấy Solution Explorer của Report Designer được minh họa ở góc phía trên bên phải của hình 8.2. Solution Explorer cung cấp một khung xem cây phân cấp của các dự án khác nhau và các file được đưa vào một giải pháp Business Intelligence Development Studio. Mục trên cùng trong cây phân cấp Solution Explorer là tên giải pháp. Ở hình 8.2, bạn có thể thấy giải pháp được gọi là ADWReports. Bên dưới giải pháp, bạn có thể có một hay nhiều dự án (project). Ở hình 8.2, bạn có thể thấy trong ví dụ này giải pháp có một dự án Reporting Services được gọi là report, và bên dưới dự án đó là một report Personel.rdl. Mỗi dự án Reporting Services có thể chứa nhiều report.

Properties

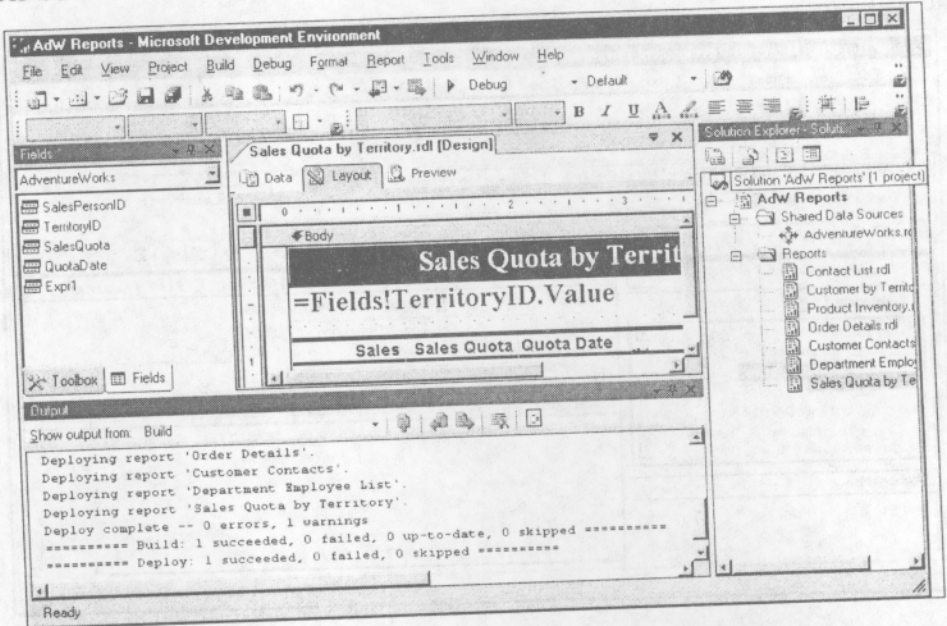
Cửa sổ Properties của Report Designer được minh họa ở góc dưới cùng bên phải của hình 8.2. Cửa sổ Properties có thể được sử dụng để xác lập các thuộc tính của các mục trình bày report vào lúc thiết kế.

Fields

Report Designer cũng cung cấp một cửa sổ Fields hiển thị nội dung

Chương 8 - Các dịch vụ báo cáo

của dataset đang được sử dụng. Bạn có thể rê và thả các trường được liệt kê trong cửa sổ Fields lên trên bề mặt thiết kế của Report Designer để đưa chúng vào report. Hình 8.3 minh họa cửa sổ Fields ở góc phía trên bên trái.



Hình 8.3 Cửa sổ Report Designer Fields

Bạn có thể luân phiên thay đổi giữa cửa sổ Fields và Toolbox bằng cách nhấp vào tab thích hợp nằm ở cuối cửa sổ Fields.

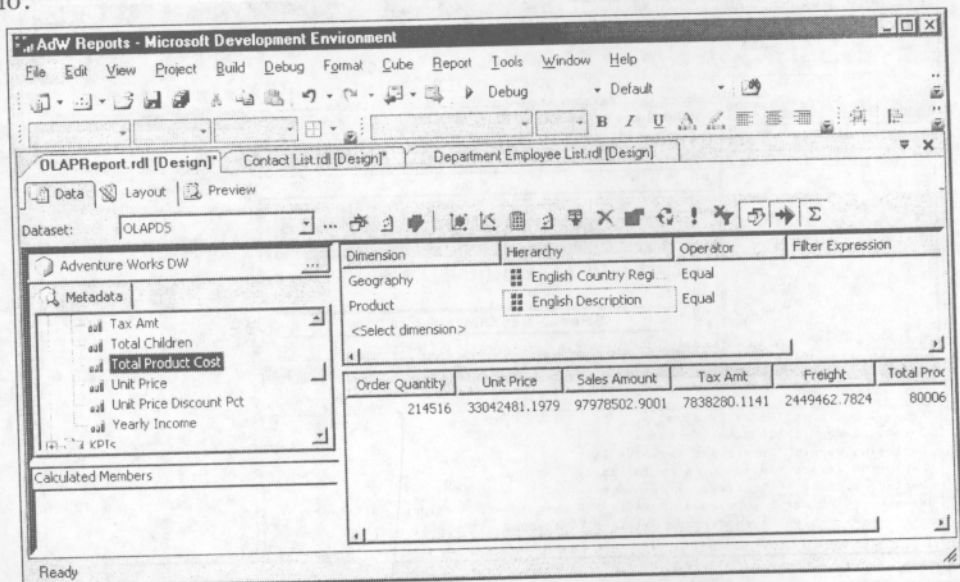
Output

Report Designer cũng cung cấp một cửa sổ Output vốn hiển thị kết quả của tiến trình tạo và triển khai các report. Sau khi một report đã được thiết kế, nó phải được tạo rồi được triển khai trước khi nó có thể được sử dụng. Tiến trình Build tạo một hợp ngữ .NET, trong khi tiến trình Deploy lấy hợp ngữ đó và cài đặt nó trong cơ sở dữ liệu ReportServer. Kết quả của những tiến trình này được hiển thị trong cửa sổ Outout mà bạn có thể thấy ở cuối hình 8.3.

OLAP Report Designer

Gần như giống Reporting Services Report Designer vốn cho phép bạn thiết kế các report cho các cơ sở dữ liệu quan hệ, OLAP Report Designer cho phép bạn thiết kế các report cho các cơ sở dữ liệu Analysis Services. Tùy thuộc vào nguồn dữ liệu mà bạn chọn, Report Designer sẽ tự động mở ở dạng Report Designer quan hệ vừa được mô tả hoặc ở dạng OLAP Report Designer. Nếu nguồn dữ liệu bạn chọn là một nguồn

dữ liệu ODBC, OLE DB hay Native SQL Client thì Report Designer quan hệ được minh họa ở hình 8.3 sẽ được mở. Nếu nguồn dữ liệu bạn chọn là một nguồn dữ liệu Microsoft SQL Server dành cho Analysis Services, OLAP Report Designer mới được minh họa ở hình 8.4 sẽ được mở.



Hình 8.4 OLAP Report Designer

OLAP Report Designer được tối ưu hóa cho việc tạo các report dựa vào các khối Analysis Services. Thay vì hiển thị các table và các cột designer cho phép bạn tạo khối metadata mà bạn có thể dùng để tạo các report của bạn.

Metadata

Ở phía bên trái của hình 8.4, bạn có thể thấy cửa sổ Metadata của OLAP Report Designer. Cửa sổ Metadata liệt kê mọi thuộc tính của khối. Ở đầu danh sách này, bạn sẽ tìm thấy các số đo của khối, sau đó là các KPI và tất cả những kích thước khác. Bạn có thể rê và thả các trường trong cửa sổ Fields lên trên các ô Dimension và Measure.

Ô Dimensions

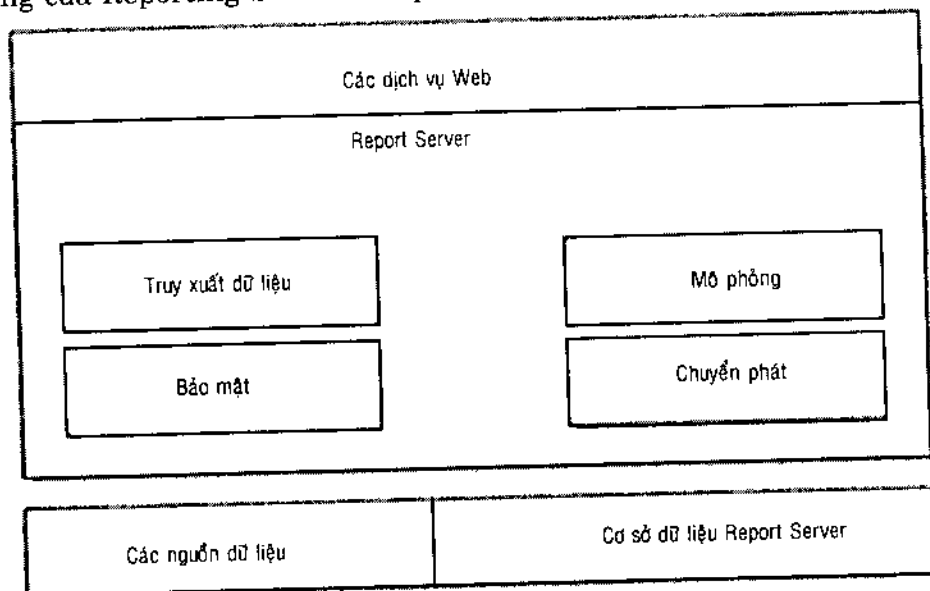
Được minh họa ở phần phía trên bên phải của hình 8.4, ô Dimensions được sử dụng để chỉ định tất cả những kích thước cần dùng trong report. Để thêm một kích thước vào ô Dimensions, bạn chỉ cần rê kích thước mong muốn cho cửa sổ Metadata và thả nó lên trên ô Dimensions.

Ô Measure

Ô Measure của OLAP Report Designer được minh họa ở phần dưới cùng của hình 8.4. Bạn sử dụng ô Measure gần giống như bạn sử dụng ô Dimensions, bằng cách rê và thả các số đo từ cửa sổ Metadata vào ô Measure. OLAP Report Designer tự động tạo các header ngang qua đỉnh của ô Measure và lấp đầy phần còn lại của ô bằng các trị dữ liệu thích hợp từ khối.

Report Server

Report Server là bộ máy mô phỏng và phân phối chính cho Reporting Services. Report Server dùng các file Report Definition Layout (RDL) được tạo bởi Report Designer và mô phỏng report ở dạng kết xuất mong muốn. Ngoài việc mô phỏng các report, Report Server còn xử lý sự bảo mật và phân phối các report. Hình 8.5 minh họa các tổng quan về tính năng của Reporting Services Report Server.



Hình 8.5: Tổng quan về Report Server

Report Server là một ứng dụng dựa trên ASP.NET và trưng bày một điểm cuối HTTP SOAP dùng để quản lý và truy cập server. Report Server là trung tâm của SQL Server 2005 Reporting Services và xử lý mọi tác vụ tạo cũng như phân phối report thiết yếu. Khi một người dùng yêu cầu một report hay một report được triển khai cho một người dùng cuối, Report Server kiểm tra các thuộc tính bảo mật của report để đảm bảo người dùng được phép truy cập report cũng như các đối tượng cơ sở dữ liệu được sử dụng trong report. Nếu người dùng có các permission

yêu cầu thì Report Server sẽ truy xuất định nghĩa report từ cơ sở dữ liệu ReportServer và mô phỏng report theo dạng đã chỉ định trong RDL. Khi report được mô phỏng, Report Server sẽ truy cập mọi nguồn dữ liệu được yêu cầu, truy xuất dữ liệu và tạo report. Ngay sau khi report đã được tạo xong, Report Server xử lý việc phân phối report đến tất cả những đích chuyển phát đã được xác định trước. Report Server lưu trữ các kết quả đã được truy xuất ở một dạng trung gian trong khoảng thời gian đã định. Sự lưu trữ này là một tính năng tuyệt vời cho khả năng mở rộng, vì nó cho phép lặp lại những yêu cầu để report được xử lý một cách nhanh chóng. Khi các report được lưu trữ, tất cả những bước truy xuất và mô phỏng dữ liệu theo yêu cầu đã được hoàn tất và report chỉ cần phân phối report đã lưu trữ đến người dùng cuối.

Report Delivery

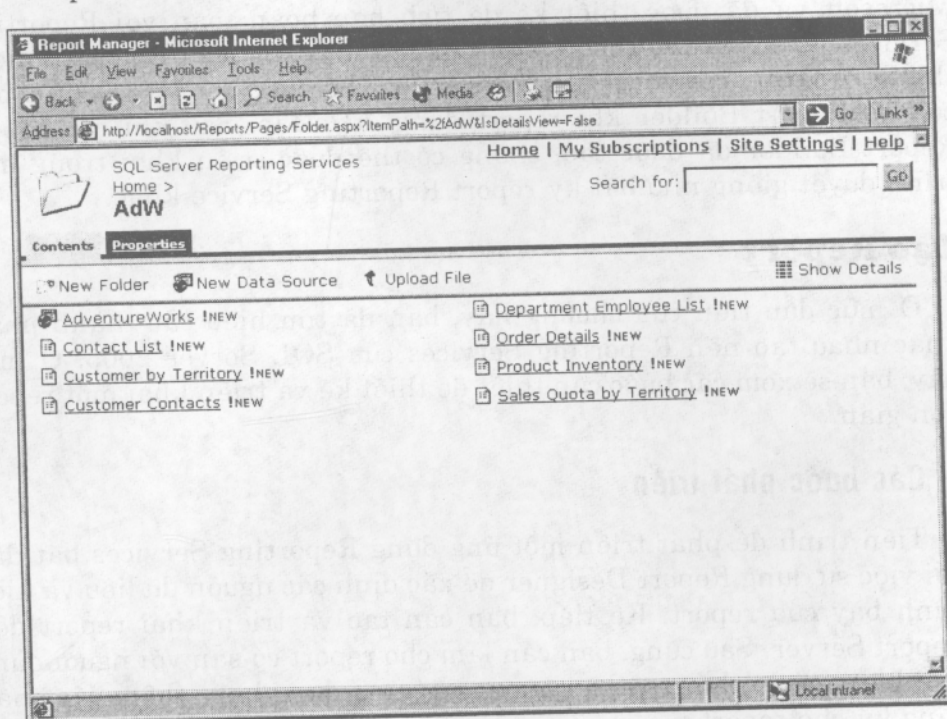
Một trong những điểm chính của tiến trình tạo giải pháp báo cáo là xác định cách chuyển các report đến người dùng cuối. Report Server chịu trách nhiệm chuyển report và hỗ trợ các report kiểu kéo theo yêu cầu cũng như các tùy chọn chuyển kiểu đẩy và dựa vào sự đăng ký.

- ❖ **Chuyển theo yêu cầu Reporting Services** cung cấp hai loại chuyển report theo yêu cầu: truy cập report dựa vào URL và truy cập report dựa vào các dịch vụ web. Với sự truy cập URL, người dùng cuối nhập trực tiếp một URL vào trình duyệt hoặc nhấp vào một liên kết để truy cập Report Server. Bất kỳ thông số yêu cầu được chuyển đi ở dạng thành phần của URL. Report Server sẽ mô phỏng report ở dạng HTML trong trình duyệt của người dùng. Người dùng cuối cũng có thể truy cập report bằng cách đưa ra các yêu cầu SOAP cho Reporting Services Report Server. Trong tình huống này, người dùng thường sẽ chọn một tùy chọn từ ứng dụng của mình vốn thật sự gửi yêu cầu SOAP. Một ưu điểm của phương pháp này là SOAP cung cấp các phương thức cho phép ứng dụng phát hiện bất kỳ thông số report được yêu cầu. Như bạn sẽ thấy trong mục kế tiếp, sự truy cập Reporting Services SOAP không chỉ giới hạn trong việc chuyển các report mà nó còn cung cấp sự truy cập đầy đủ vào các chức năng quản lý Reporting Services.
- ❖ **Chuyển theo sự đăng ký** Ngoài tùy chọn chuyển report kiểu kéo, Reporting Services còn hỗ trợ việc chuyển kiểu đẩy theo sự đăng ký. Trái với chuyển report kiểu kéo. Trong đó người dùng bắt đầu tiến trình tạo report, với tùy chọn chuyển report kiểu đẩy dựa vào sự đăng ký, bộ máy Report Server chuyển các report đến người dùng cuối theo thời biểu đã định. Các phần đăng ký có thể dựa vào thời gian hay được

điều khiển bởi dữ liệu. Chẳng hạn, bạn có thể cấu hình Report Server để chuyển một report đã cho hay một tập hợp report vào cuối ngày làm việc, cuối tuần hay cuối tháng. Để chuyển các report theo thời biểu, Reporting Services phải sử dụng SQL Agent. Các phần đăng ký được điều khiển bởi dữ liệu thì khác với sự đăng ký thông thường ở chỗ các phần đăng ký được điều khiển bởi dữ liệu (data-driven subscriptions) lấy thông tin đăng ký vào lúc chạy. Các phần đăng ký được điều khiển bởi dữ liệu được thiết kế để xử lý những tình huống mà trong đó người nhận report có thể thay đổi giữa các lần chạy khác nhau của report. Cả hai cơ chế chuyển report theo yêu cầu và dựa vào sự đăng ký chia sẻ các tùy chọn xuất report như nhau.

Report Manager

Report Manager là công cụ chính để quản lý các giải pháp tạo report Reporting Services. Report Manager là một ứng dụng ASP.NET dựa trên web và được truy cập bằng cách hướng trình duyệt web của bạn đến `http://<servername>/Reports`. Hình 8.6 minh họa Report Manager.



Hình 8.6: Reporting Services Report Manager

Nếu bạn đã sử dụng Sharepoint thì bạn chắc chắn sẽ nhận thấy một điểm tương tự nổi bật giữa Reporting Services Report Manager và một site Sharepoint. Cả hai có một diện mạo rất giống nhau. Report Manager cho phép bạn xem và quản lý tất cả những report đã được triển khai đến Report Server. Bạn có thể hiệu chỉnh các nguồn dữ liệu và các chuỗi nối kết được sử dụng bởi các report và chỉnh sửa nhiều đặc tính report. Ngoài ra, Report Manager cho phép Reporting Services Administrator cài đặt sự bảo mật và đăng ký cho những report được truy cập nhờ dùng Reporting Services.

Report Builder

Để nâng cao kinh nghiệm tạo report cho người dùng cuối, Reporting Services cũng đưa vào Report Builder vốn được tạo dựa trên công nghệ ActiveViews. Microsoft đã lấy công nghệ Report Builder vào năm 2004 từ công ty ActiveViews. Sự đưa vào Report Builder đã cho SQL Server 2005 một công cụ tạo report và mẫu truy vấn dành cho người dùng cuối dựa trên web. Report Builder trực tiếp giải quyết vấn đề về các yêu cầu report bằng cách cung cấp một công cụ cho phép người dùng cuối tạo các report của riêng mình. Report Builder dựa vào .NET framework của Microsoft và đã được thiết kế để tích hợp hoàn toàn với Reporting Services. Công cụ tạo report dành cho người dùng cuối Report Builder thì dễ sử dụng; các chi tiết của report có thể được xem trực tiếp trong cửa sổ Report Builder khi report đang được tạo. Sau khi các report Report Builder đã được tạo, chúng có thể được triển khai trong một trình duyệt giống như bất kỳ report Reporting Service khác.

Tạo Report

Ở mục đầu tiên của chương này, bạn đã tìm hiểu các thành phần khác nhau tạo nên Reporting Services của SQL Server 2005. Ở mục này, bạn sẽ xem các bước cần thiết để thiết kế và triển khai một report đơn giản.

Các bước phát triển

Tiến trình để phát triển một ứng dụng Reporting Services bắt đầu với việc sử dụng Report Designer để xác định các nguồn dữ liệu và kiểu trình bày của report. Kế tiếp, bạn cần tạo và triển khai report đến Report Server. Sau cùng, bạn cần làm cho report có sẵn với người dùng cuối bằng cách nhúng report trong một ứng dụng hoặc thêm các phần đăng ký cho report.

Thiết kế giải pháp báo cáo

Khi tạo các giải pháp báo cáo nhờ dùng Reporting Services, trước

tiên bạn chọn một dataset xác định dữ liệu sẽ được dùng trong báo cáo (report), sau đó bạn trình bày các trường dữ liệu riêng lẻ trên report. Để xử lý các report chứng khoán được trình bày ở dạng bảng hay ma trận. Microsoft đã cung cấp một Report Design Wizard hướng dẫn bạn qua các bước tạo một nguồn dữ liệu và trình bày report.

Tạo và triển khai giải pháp báo cáo

Ngay sau khi report đã được thiết kế, bạn cần tạo report rồi triển khai nó ra Report Server. Việc tạo report sẽ tạo ra một hợp ngữ .NET vốn chạy report. Việc triển khai report về cơ bản sẽ lấy hợp ngữ này và sao chép nó sang Reporting Services Report Server. Trong khi bạn có thể thực hiện điều này bằng tay, Report Designer có các tùy chọn cài sẵn để vừa tạo vừa triển khai các report ra Report Server.

Làm cho Report có sẵn với người dùng cuối

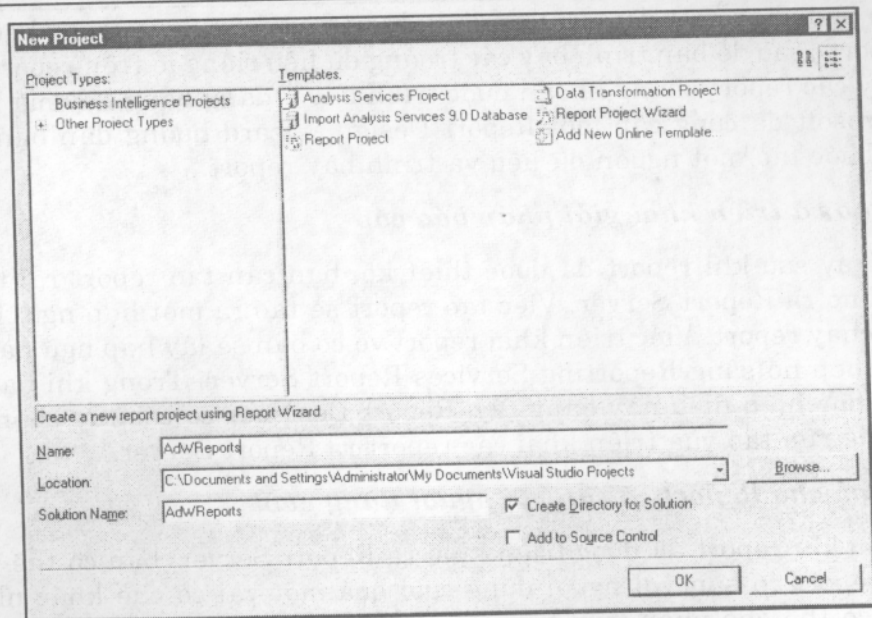
Sau khi report đã được triển khai ra Report Server, bạn có thể làm cho report có sẵn với người dùng cuối qua một vài cơ chế khác nhau. Bạn có thể cho phép truy cập vào các report bằng cách nhúng chúng trong một trình ứng dụng, thông qua các URL của chúng, hoặc bằng cách tạo một phần đăng ký để đẩy report đến người dùng cuối. Các phần đăng ký report có thể được cài đặt để được chuyển vào một thời điểm nhất định, hoặc chúng có thể được điều khiển bởi dữ liệu.

Bạn đã xem phần tổng quan về tiến trình phát triển Reporting Services, mục kế tiếp sẽ đưa bạn qua các bước phát triển và triển khai một report đơn giản nhờ dùng Reporting Services của SQL Server 2005.

Tạo một Report Reporting Services

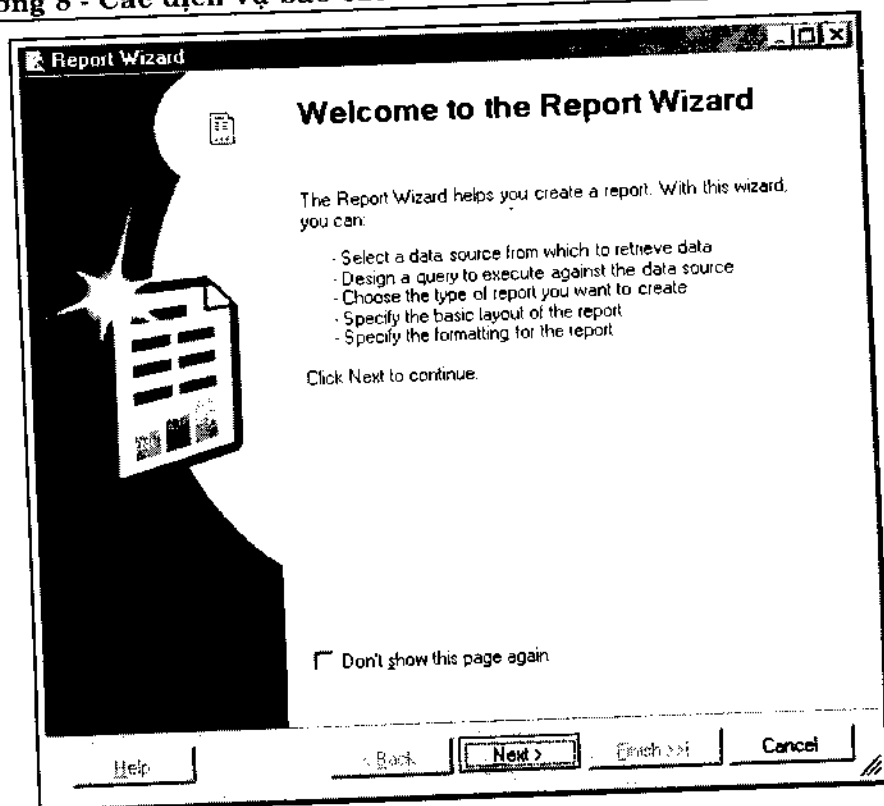
Bạn có thể bắt đầu sử dụng Report Designer bằng cách khởi động Report Wizard và dùng nó để tạo report đầu tiên của bạn hoặc bằng cách bắt đầu với một bề mặt thiết kế trống rồi thêm các thành phần định nghĩa report của riêng bạn vào. Trong mỗi trường hợp, xác định một dataset là điều đầu tiên bạn cần thực hiện để tạo một report. Trong ví dụ này, bạn sẽ thấy cách để tạo nhanh một report nhờ dùng Report Wizard.

Để tạo một ứng dụng Reporting Services, trước tiên bạn mở Business Intelligence Development Studio rồi chọn tùy chọn File > New Project để hiển thị hộp thoại New Project như minh họa ở hình 8.7.



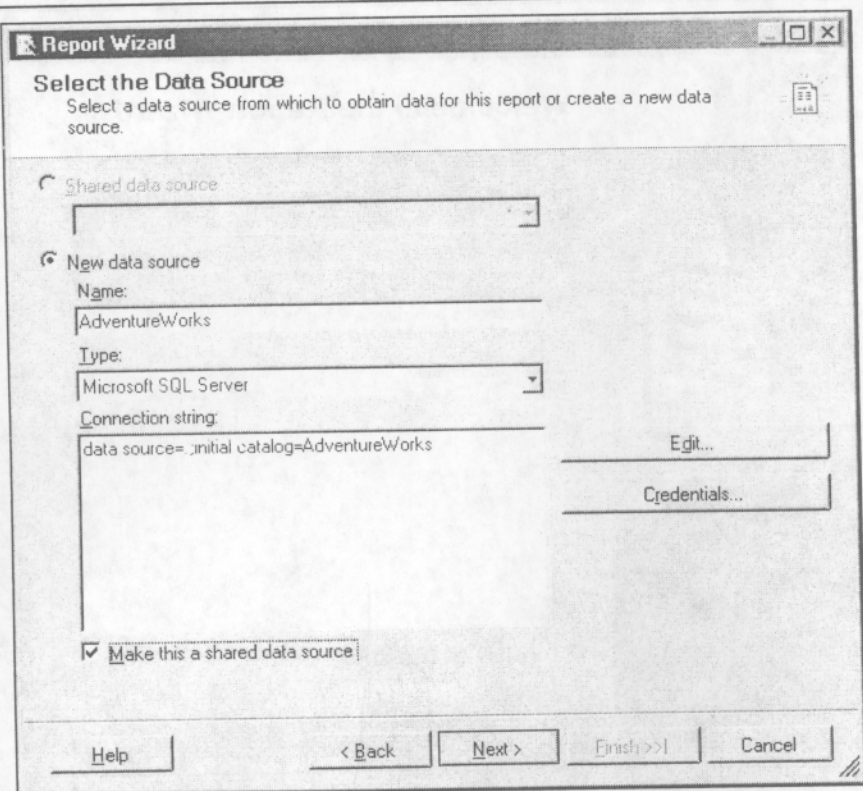
Hình 8.7: Tạo một report mới: New Project

Để tạo một report Reporting Services mới nhờ dùng Report Wizard, trước tiên bạn chọn tùy chọn Business Intelligence Projects từ danh sách Project Types. Sau đó, trong danh sách Templates ở phía bên phải của màn hình, chọn tùy chọn Report Project Wizard. Điền thông tin và các hộp ở cuối hộp thoại. Trong hộp text Name, nhập tên của dự án hiện hành. Hộp Location chỉ định nơi định vị của các file nguồn trong dự án report. Hộp Solution Name cho phép bạn đặt tên giải pháp Reporting Services. Ở đây, bạn có thể thấy trị AdwReports được dùng làm tên của dự án và giải pháp. Nhấp OK để mở hộp thoại Report Wizard Welcome như minh họa ở hình 8.8.



Hình 8.8: Hộp thoại Report Wizard Welcome

Hộp thoại Report Wizard Welcome cung cấp cho bạn một phần tổng quan về những bước mà Report Wizard sẽ làm theo trong tiến trình tạo một report. Trước tiên, bạn chọn một nguồn dữ liệu, thiết kế một mẫu truy vấn, rồi chọn loại report mà bạn muốn, và sau cùng chỉ định kiểu định dạng cho report. Nhấp Next để hiển thị hộp thoại Select The Data Source như bạn thấy ở hình 8.9.



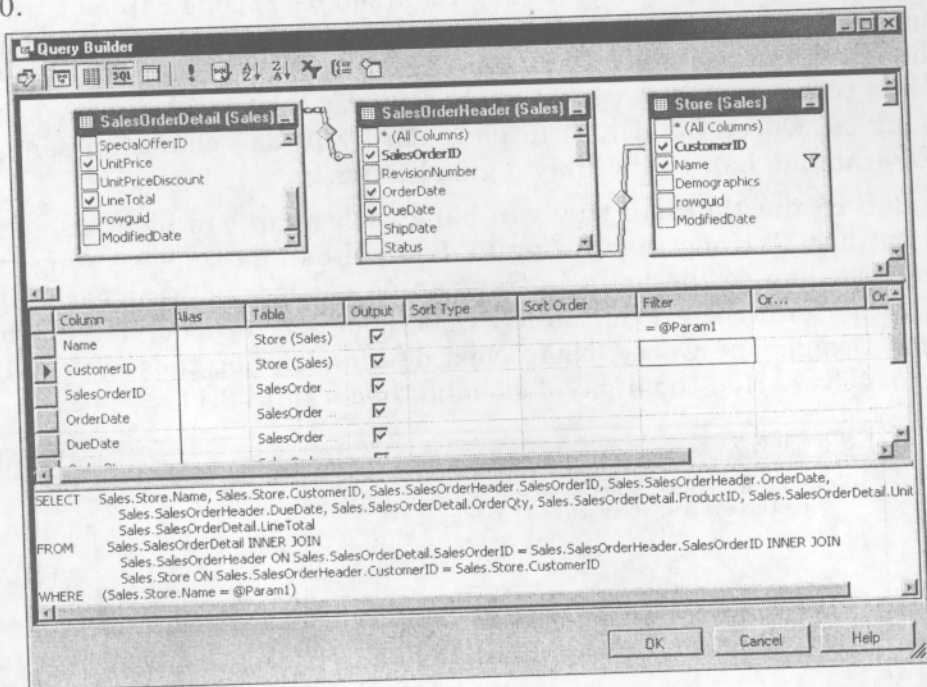
Hình 8.9: Hộp thoại Select the Data Source

Hộp thoại Select the Data Source cho phép bạn xác định nối kết của bạn với cơ sở dữ liệu. Để tạo Data Source, trước tiên hãy đặt tên cho nó. Tên có thể là bất kỳ tên nào mà bạn chọn. Nó chỉ có tác dụng giúp nhận biết nguồn dữ liệu. Kế tiếp, dùng danh sách xổ xuống Type để chọn loại hệ thống cơ sở dữ liệu mà nguồn dữ liệu sẽ sử dụng. Trị mặc định là Microsoft SQL Server, nhưng bạn cũng có thể chọn OLE DB, Microsoft SQL Server Analysis Service, Oracle, hay ODBC. Kế tiếp, trong hộp Connection String nhập chuỗi nối kết được yêu cầu để nối kết với cơ sở dữ liệu đích. Nếu bạn chưa quen với các trị chuỗi nối kết, bạn có thể nhấp Edit để hiển thị hộp thoại Data Link, hộp thoại này sẽ đưa bạn qua tiến trình tạo Data Source. Sau đó, hãy làm cho nó trở thành một hộp kiểm shared data source ở cuối màn hình nhằm cho phép bạn chia sẻ nguồn dữ liệu với các report khác. Bạn có thể chọn để tạo một nguồn dữ liệu chia sẻ, vốn được sử dụng bởi nhiều report khác nhau, hoặc để tạo một nguồn dữ liệu được sử dụng chỉ bởi report mà bạn đang tạo. Nếu bạn dự định tạo nhiều report xuất phát từ cùng một cơ sở dữ liệu, việc tạo một nguồn dữ liệu chia sẻ là một ý tưởng hay, vì nó có thể được sử dụng tự do bởi tất cả các report trong giải pháp của bạn và làm

Chương 8 - Các dịch vụ báo cáo

cho việc tạo một nguồn dữ liệu duy nhất cho mỗi report trở nên không cần thiết. Nhấp Next để hiển thị hộp thoại Design the Query.

Từ hộp thoại Design the Query, bạn có thể nhập một câu lệnh SQL bằng tay để xác định dataset mà report sẽ sử dụng, hoặc bạn có thể nhấp nút Query Builder để hiển thị Query Builder như minh họa ở hình 8.10.



Hình 8.10: Query Builder

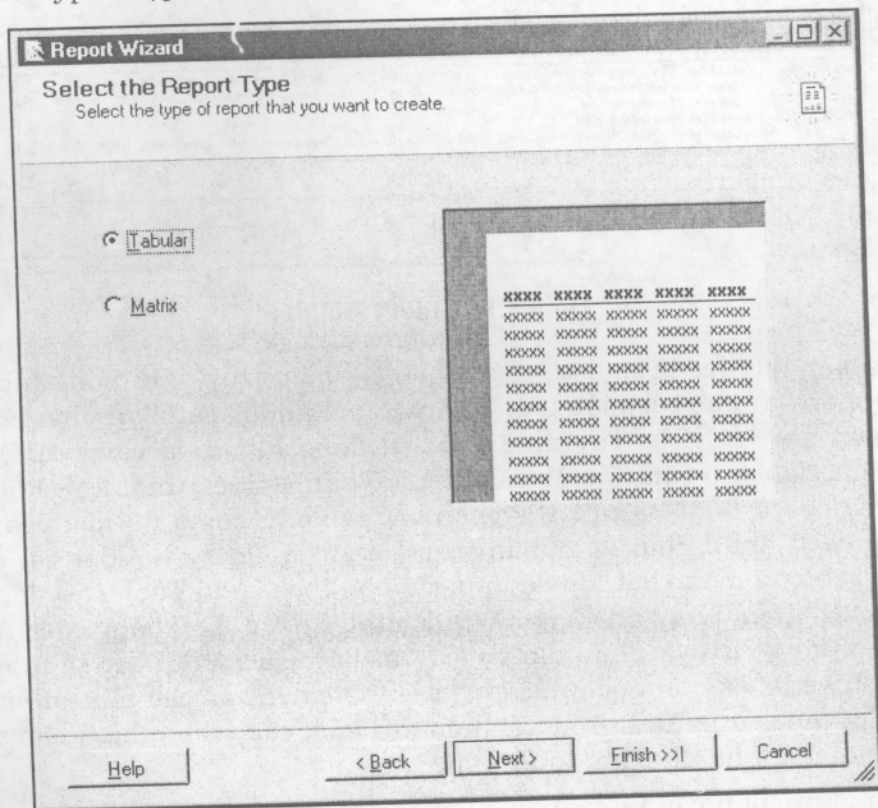
Query Builder là một công cụ thiết kế mẫu truy vấn tương tác vốn cho phép bạn tạo các mẫu truy vấn SQL mà không cần bạn phải là một chuyên gia về SQL. Tuy nhiên, để sử dụng hiệu quả công cụ Query Design, bạn vẫn cần có một kiến thức cơ bản về kiểu thiết kế và bố cục của cơ sở dữ liệu. Bạn có thể chọn các table từ cơ sở dữ liệu của bạn bằng cách nhấp phải vào phần trên cùng của Query Builder rồi chọn Add Table từ menu bật lên để hiển thị hộp thoại Add Tables. Ở đó, bạn có thể chọn một hay nhiều table (nhiều table được chọn bằng cách nhấn giữa phím CTRL và nhấp table mong muốn). Query Builder sẽ tự động phát hiện bất kỳ mối quan hệ giữa các table dựa vào các tên cột và các kiểu dữ liệu cũng như sẽ vẽ các liên kết giữa các table nhằm cho thấy các mối quan hệ đó bằng hình ảnh.

Sau khi chọn các table, bạn chọn các cột mong muốn cho mỗi table bằng cách đặt một dấu kiểm trong hộp kiểm đứng trước tên cột. Như bạn có thể đoán được, việc đánh dấu kiểm vào mục * (All Columns) sẽ

tự động chọn tất cả các cột từ table. Khi bạn chọn các table và các cột một cách tương tác rồi ấn định mối quan hệ giữa các table, Query Designer tự động tạo câu lệnh SQL mà bạn có thể thấy ở cuối hình 8.10.

Query Builder cũng có thể được sử dụng để tạo các mẫu truy vấn được thông số hóa trong đó người dùng cuối (end user) cung cấp một trị cho mẫu truy vấn vào lúc chạy. Để tạo một mẫu truy vấn được thông số hóa nhờ dùng Query Builder, bạn chỉ cần gõ một dấu chấm hỏi vào cột Filter trong hàng tên cột cơ sở dữ liệu mà bạn muốn dùng với một thông số. Query Builder sẽ tự động đổi ký tự dấu chấm hỏi sang trị `=@Param` mà bạn có thể thấy ở giữa hình 8.10.

Bạn có thể thử mẫu truy vấn bằng cách nhấp vào biểu tượng dấu chấm than (!) trong thanh công cụ. Sau khi bạn đã thiết kế xong mẫu truy vấn, bạn có thể lưu mẫu truy vấn đó và tiếp tục bằng cách nhấp OK. Câu lệnh SQL được tạo bởi Query Builder sẽ được viết vào hộp thoại Design The Query. Nhấp Next để hiển thị hộp thoại Select The Report Type. Hộp thoại này được minh họa ở hình 8.11.



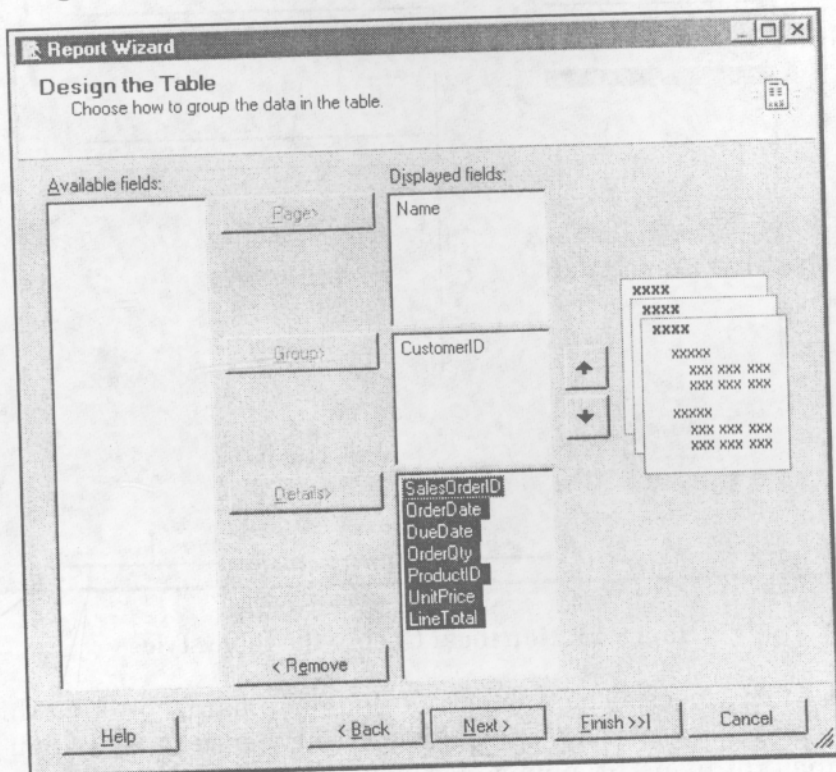
Hình 8.11: Hộp thoại Select the report type

Trong khi Report Designer cho phép bạn có một mức độ linh hoạt trong việc thiết kế các report, Report Wizard bị hạn chế hơn về loại report mà nó sẽ tạo cho bạn. Report Wizard sẽ tạo một trong hai loại report khác nhau: report dạng bảng hay report dạng ma trận.

Lưu ý

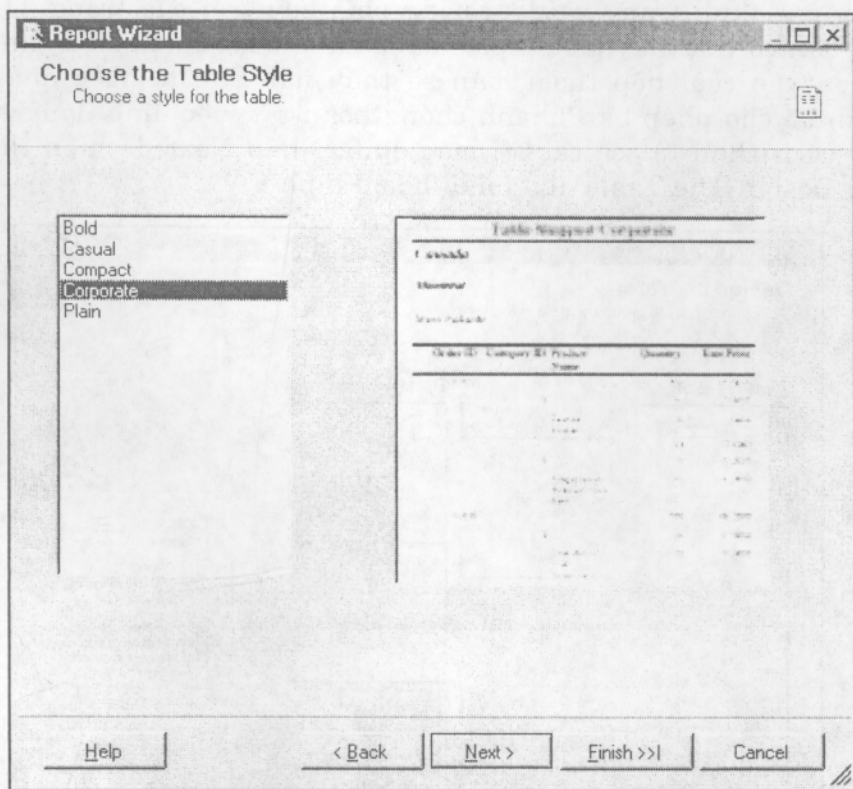
Tuy các loại report bị giới hạn, nhưng Report Wizard giúp bạn có một điểm khởi đầu tốt cho việc tạo một report cơ sở mà sau này bạn có thể tùy biến trong Report Designer.

Report dạng bảng (Tabular) tuân theo kiểu trình bày report thông thường trong đó các header được trình bày ở đầu trang và thông tin chi tiết được liệt kê bên dưới trong phần nội dung của report. Kiểu report Matrix trình bày một report kiểu crosstab trong đó có các header nằm ngang qua đỉnh trang và dọc xuống phía bên trái của trang. Ở hình 8.10, bạn có thể thấy kiểu report dạng bảng được chọn. Nếu bạn nhấp kỹ hơn vào cuối hộp thoại, bạn sẽ thấy nút Finish. Việc nhấp nút Finish sẽ cho phép bạn nhanh chóng hoàn tất việc định dạng report bằng cách chọn tất cả các trị mặc định. Nhấp Next để hiển thị hộp thoại Design The Table như minh họa ở hình 8.12.



Hình 8.12: Hộp thoại Design the table

Ban đầu, tất cả các cột có sẵn từ mẫu truy vấn được hiển thị trong danh sách Available Fields mà bạn có thể thấy ở phía bên trái của hình 8.12. Từ danh sách các trường có sẵn đó, bạn có thể rê các trường một cách có chọn lọc đến vùng report mà bạn muốn chúng xuất hiện. Nếu bạn muốn một trường được hiển thị trong phần header của trang, bạn hãy rê nó đến mục Page ở phía trên bên phải của màn hình. Nếu bạn muốn một trường được sử dụng cho các tổng nhóm, bạn hãy rê nó đến mục Group. Và nếu bạn muốn một trường là một trường dữ liệu, bạn rê nó đến mục Details. Ở hình 8.12, bạn có thể thấy trường Name được dùng làm header của trang, trường CustomerID được dùng làm header của nhóm, và các trường SalesOrderID, OrderDate, DueDate, OrderQty, ProductID, UnitPrice, và LineTotal được dùng trong vùng details (chi tiết). Ngay sau khi bạn đã trình bày các trường sẽ được sử dụng trong report, nhấp Next để hiển thị hộp thoại Report Wizard kế tiếp mà bạn có thể thấy ở hình 8.13.

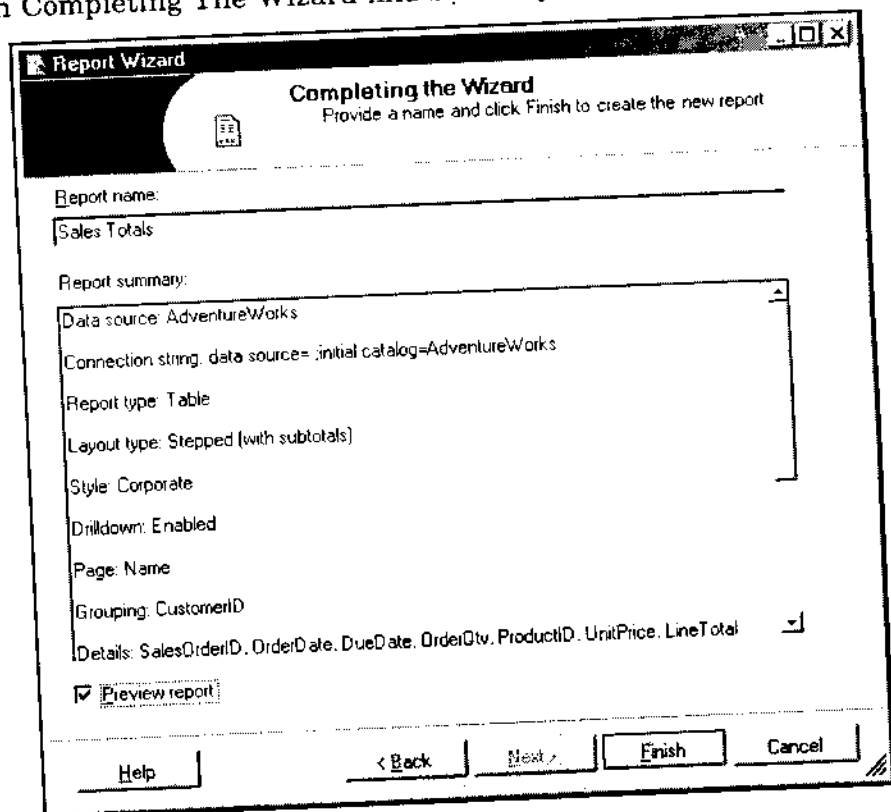


Hình 8.13: Hộp thoại Choose the table style

Hộp thoại Choose The Table Style cho phép bạn chọn kiểu thiết kế chung cho report sẽ được tạo bởi Reporting Services. Các kiểu table khác nhau thì tương tự, nhưng mỗi kiểu dùng một cách phối màu khác

Chương 8 - Các dịch vụ báo cáo

và kiểu định dạng cũng khác. Ở hình 8.13, bạn có thể thấy kiểu Corporate được chọn. Hộp thoại Choose The Table Style là hộp thoại tạo report sau cùng mà Report Wizard hiển thị. Nhấp Next để hiển thị màn hình Completing The Wizard như bạn thấy ở hình 8.14.



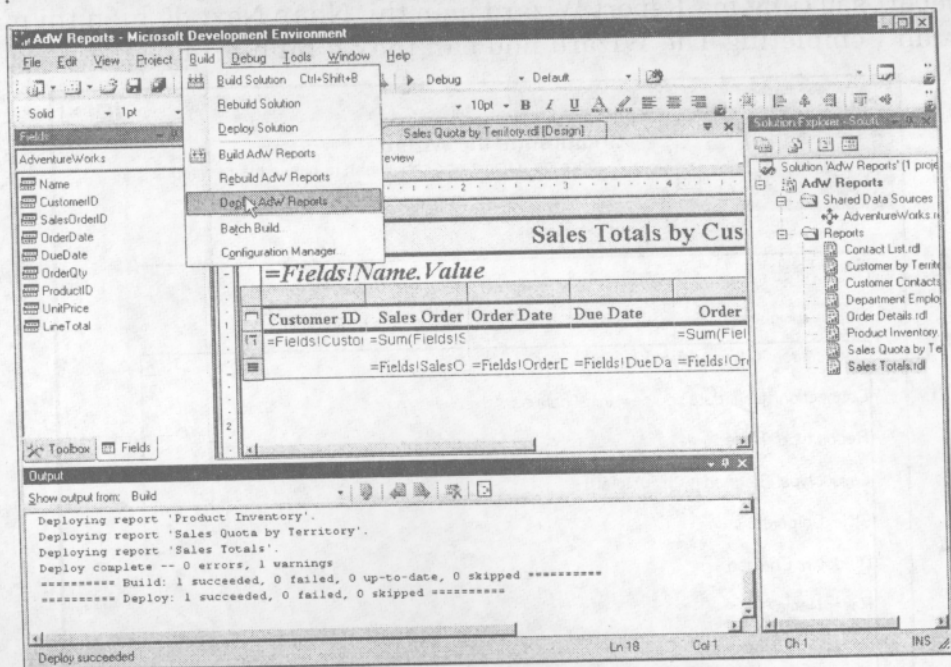
Hình 8.14: Màn hình Completing the wizard

Hộp thoại Completing The Wizard cho phép bạn xem lại tất cả những lựa chọn trong các hộp thoại wizard trước đây. Vào lúc này, bạn có thể dùng nút Back để lùi trang và trình bày các phần chỉnh sửa cho các đặc tính report, hoặc bạn có thể nhấp Finish để tạo report. Việc chọn Preview sẽ mô phỏng report cho bạn xem trong cửa sổ Preview của Report Designer. Sau khi report đã được tạo, nó được thêm vào giải pháp Reporting Services trong Business Intelligence Development Studio.

Triển khai một Report Reporting Services

Sau khi report đã được tạo, bước kế tiếp trong tiến trình tạo một ứng dụng Reporting Service là tạo report và triển khai nó ra Report Server. Việc tạo report sẽ tạo ra một hợp ngữ .NET DLL, và việc triển khai report sẽ sao chép hợp ngữ đó sang Reporting Services. Bạn có thể

triển khai các giải pháp báo cáo từ Report Designer bằng cách chọn tùy chọn Build > Deploy Reports như bạn thấy ở hình 8.15.

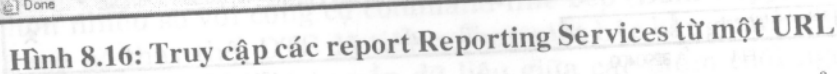


Hình 8.15: Triển khai giải pháp Reporting Services

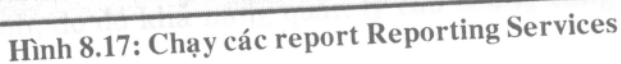
Nếu bạn chọn một trong những tùy chọn triển khai và report đã được thay đổi, Report Designer sẽ tự động tạo report trước khi nó được triển khai. Xuất từ các tiến trình tạo và triển khai được hiển thị trong cửa sổ Output mà bạn thấy ở cuối hình 8.15. Bất kỳ lỗi hay sự cố sẽ được liệt kê trong cửa sổ này. Nếu tiến trình triển khai report thành công thì thông báo về sự thành công này cũng được liệt kê trong cửa sổ Output.

Chạy một Report Reporting Services

Các report Reporting Services có thể được chạy bằng cách truy cập URL của chúng hoặc bằng cách nhúng chúng trong các ứng dụng của bạn. Bạn có thể truy cập và chạy các report Reporting Services bằng cách trình duyệt của bạn đến <http://<servername>/ReportServer> URL, trong đó tất cả các report Reporting Services sẽ được liệt kê. Hình 8.16 minh họa trang web ReportServer.



ReportServer URL liệt kê tất cả những report đã được triển khai Report Server. Mỗi giải pháp khác được chứa bên trong thư mục con của riêng nó. Để kiểm tra report đã được triển khai, bạn chỉ cần nhấp vào liên kết và Report Server sẽ mô phỏng report bên trong trình duyệt. Hình 8,17 minh họa report mẫu trong trình duyệt.



Report được mô phỏng trong trình duyệt tuân theo dạng đã được cài đặt trong bước thiết kế report. Vì report này sử dụng các thông số nên trường param1 được hiển thị ở đầu màn hình. Người dùng cuối cần nhập một trị vào trường này và nhấn nút View Report để mô phỏng report. Ở hình 8.17, bạn có thể thấy trị của A Bike Store được nhập vào cho thông số có thể thay thế được.

Ngoài ra, do report này đã được tạo nhờ dùng tùy chọn đi sâu vào nên một dấu cộng được hiển thị ở phía trước dòng chi tiết trên report. Việc nhấp vào dấu cộng (+) này sẽ hiển thị các dòng chi tiết sẽ được đưa vào dòng tóm tắt ở cấp độ trên cùng đó. Trong ví dụ này, việc nhấp dấu cộng (+) sẽ mở rộng vùng hiển thị, như minh họa ở hình 8.18.

Sales Totals - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Forward Stop Search Favorites Media

Address <http://localhost/ReportServer?%2fAdw%2fSales+Totals&rs:Command=Render> Go Links

Param1

1 of 1 Find | Next

Select a format Export

1	3290400	121	54961	61321.12	
43860	8/1/2001 12:00:00 AM	8/13/2001 12:00:00 AM	2	761	503.35
43860	8/1/2001 12:00:00 AM	8/13/2001 12:00:00 AM	1	770	503.35
43860	8/1/2001 12:00:00 AM	8/13/2001 12:00:00 AM	2	758	1049.75
43860	8/1/2001 12:00:00 AM	8/13/2001 12:00:00 AM	2	765	503.35
43860	8/1/2001 12:00:00 AM	8/13/2001 12:00:00 AM	1	732	440.17
43860	8/1/2001 12:00:00 AM	8/13/2001 12:00:00 AM	1	762	503.35

Local intranet

Hình 8.18: Đi sâu vào report

Ở hình 8.18, bạn có thể thấy các chi tiết được đưa vào dòng tóm tắt đã minh họa trong hình trước. Việc nhấp dấu trừ (-) sẽ thu nhỏ vùng hiển thị chi tiết và cho thấy dòng tóm tắt.

Việc chạy các report trực tiếp từ Reporting Services URL thì thích hợp cho việc thử hay kiểm tra, nhưng khi ứng dụng của bạn đang hoạt động, bạn sẽ muốn những report URL trong ứng dụng của bạn hoặc truy cập Report Server thông qua các cuộc gọi dịch vụ web.

CHƯƠNG 9

Các dịch vụ hợp nhất

Tổng quan

Các dịch vụ chuyển đổi dữ liệu (Data Transformation Services, gọi tắt là DTS) đã là một trong những tiện ích SQL Server hữu dụng và phổ biến nhất kể từ lúc nó được giới thiệu cùng với SQL Server 7. DTS ban đầu được thiết kế là một công cụ dùng để thực hiện việc truy xuất, biến đổi và tải dữ liệu (ETL) cho các kho chứa dữ liệu OLAP Services. Tuy nhiên, Microsoft đã nhanh chóng nhận ra tính hữu dụng của nó và làm cho DTS trở thành công cụ chính để nhập và xuất dữ liệu từ các cơ sở dữ liệu của SQL Server cũng như là một công cụ BI ETL. Ngoài việc dễ sử dụng hơn nhiều so với công cụ command-line bcp (Bulk Copy Program) loại cũ mà nó thay thế, DTS đã tiến xa hơn một bậc bằng cách cho phép chuyển đổi dữ liệu khi nó truyền dữ liệu giữa các điểm cuối nguồn và đích.

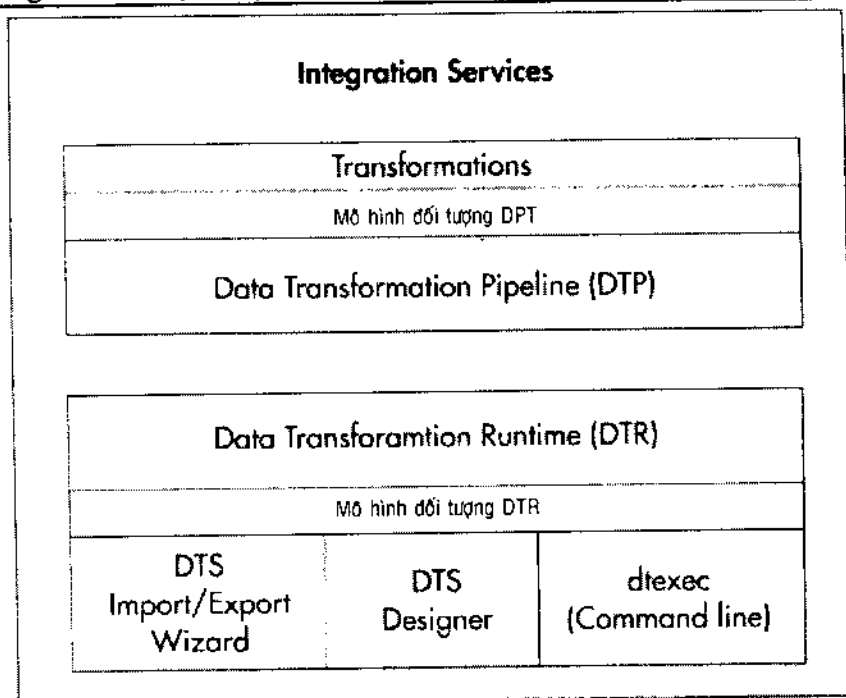
DTS vừa cung cấp một wizard thân thiện với người dùng được thiết kế để thực hiện các tiến trình truyền dữ liệu đơn giản vừa cung cấp một trình thiết kế đồ họa cho các hoạt động truyền và chuyển đổi dữ liệu tinh vi hơn. Các phiên bản SQL Server 7 và 2000 của DTS đã hỗ trợ 100 phần trăm OLE DB cho cơ sở dữ liệu nguồn và đích. Điều này có nghĩa là mặc dù DTS là thành phần của SQL Server nhưng thật sự nó có thể được sử dụng để truyền dữ liệu giữa bất kỳ hai nguồn dữ liệu OLE DB mà không cần SQL Server phải là nguồn hay đích dữ liệu. Chẳng hạn, ngoài khả năng nhập và xuất dữ liệu từ cơ sở dữ liệu SQL Server, DTS cũng có thể được sử dụng để truyền dữ liệu giữa các hệ thống cơ sở dữ liệu khác như Access, Oracle, và DB2 mà không hoàn toàn liên quan đến SQL Server. Loại linh hoạt này đã làm cho DTS trở thành một công cụ truyền cơ sở dữ liệu mạnh và hữu dụng.

Tuy nhiên, cho dù DTS hữu dụng và hấp dẫn như thế nào thì nó vẫn còn một số hạn chế khá quan trọng. Để sẵn sàng cho doanh nghiệp, DTS cần có khả năng mở rộng tốt hơn. Ngoài ra, các gói (package) DTS không được chuyển tải dễ dàng giữa các hệ thống. Nói cách khác, một package DTS được thiết kế để thực hiện một tiến trình từ hệ thống SQL Server A không thể dễ dàng được sử dụng lại để thực hiện cùng tiến trình truyền đó từ hệ thống B. Ngoài các hạn chế về mặt triển khai này, các phiên bản trước của DTS còn thiếu khả năng xử lý và ghi nhật trình lỗi mạnh, do đó khả năng quản lý của chúng cũng bị hạn chế.

Với SQL Server 2005, Microsoft hoàn toàn cải tiến DTS và viết lại nó từ đầu. Bằng cách phản ánh mọi bản chất mới của nó, Microsoft đã đổi tên DTS thành các dịch vụ hợp nhất (Integration Services). Mục đích của Microsoft dành cho Integration Services của SQL Server 2005 là biến nó thành một nền ETL doanh nghiệp dành cho Windows ngang hàng với bất kỳ sản phẩm BI ETL cấp doanh nghiệp độc lập. Với mục đích này, Microsoft đã viết Integration Services hoàn toàn mới nhờ dùng mã .NET đã được quản lý, tạo cho nó một nền tảng mạnh mẽ hơn nhiều. Trong quá trình này, Microsoft hoàn toàn thiết kế lại Integration Services, cho nó một cấu hình hoàn toàn mới, hỗ trợ tốt hơn khả năng lập trình và cải tiến hiệu suất vào lúc chạy. Integration Services mới có một trình thiết kế đồ họa mới và một tập hợp các tác vụ truyền và chuyển đổi dữ liệu nâng cao hơn. Integration Services, giống như DTS cũ, hỗ trợ 100 phần trăm nguồn và đích, nghĩa là nó có thể nối kết độc lập với cả hai nguồn dữ liệu nguồn và đích mà không cần mỗi nguồn dữ liệu phải là một hệ thống SQL Server. Trong chương này, bạn sẽ tìm hiểu về các tính năng mới trong Integration Services của SQL Server 2005. Trước tiên, bạn sẽ xem xét kiểu kiến trúc Integration Services mới, sau đó khảo sát một vòng qua các công cụ và thành phần của Integration Services.

Kiểu kiến trúc Integration Services mới

Kiểu kiến trúc Integration Services mới được chia thành hai phần chính: Data Transformation Pipeline (DTP) và Data Transformation Runtime (DTR). Microsoft đã phân chia Integration Services ra thành hai phần khác nhau như vậy chủ yếu là để phân biệt rõ ràng giữa dòng chảy dữ liệu (data flow) và dòng làm việc (work flow). Trong các phiên bản trước của DTS, động cơ data flow mạnh hơn các khả năng của work flow. Sự phân chia mới này về cơ bản làm cho phần work flow của Integration Services trở thành một thành phần hạng nhất trên cùng cấp độ với thành phần data flow. Về cơ bản, DTP mới thay DTS Data Pump cũ đã được sử dụng trong các phiên bản SQL Server 7 và 2000 của DTS. Chức năng chính của nó là xử lý dòng chảy dữ liệu giữa các đích nguồn và đích đến. DTR là một môi trường thực thi công việc vốn điều khiển dòng làm việc trong một package Integration Services. Mỗi thành phần này được thực thi nhờ dùng DLL của riêng nó cũng như đối tượng phân biệt của nó mà bạn có thể lập trình. Ở hình 9.1, bạn có thể thấy một phần tổng quan về kiểu kiến trúc Integration Services mới.

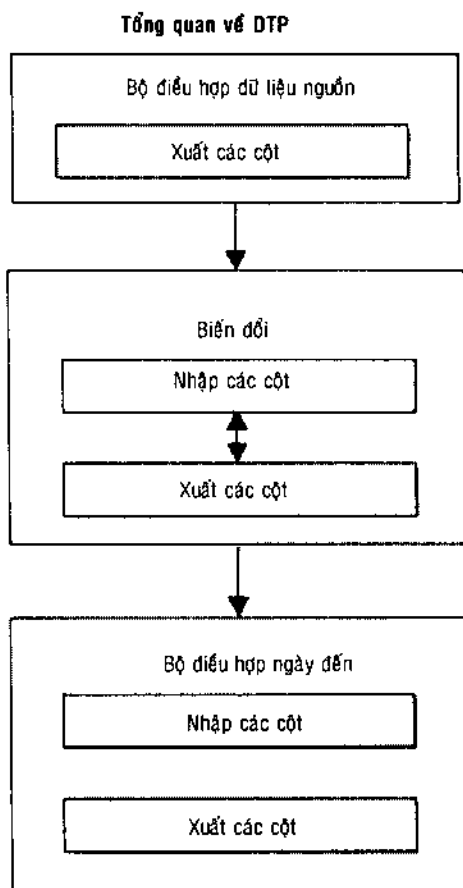


Hình 9.1: Kiểu kiến trúc Integration Services

Integration Services DTP và DTR mới được trình bày chi tiết hơn trong các mục sau. Bộ công cụ Integration Services mới cũng được nêu rõ hơn ở sau trong chương này.

Data Transformation Pipeline (DTP)

DTP chú ý đến dòng chảy dữ liệu và các tiến trình biến đổi xảy ra khi các hàng được di chuyển giữa nguồn dữ liệu và đích dữ liệu. DTP sử dụng các bộ điều hợp dữ liệu (data adapter) để nối kết với các nguồn dữ liệu nguồn và đích. Như bạn thấy ở hình 9.1, bộ máy DTP được truy cập nhờ sử dụng mô hình đối tượng DTP Pipeline. Mô hình đối tượng (object model) này là API được sử dụng bởi cả các tiến trình biến đổi cài sẵn do Microsoft cung cấp và bất kỳ tiến trình biến đổi tùy ý được tạo bởi người dùng. Các tiến trình biến đổi (transformations) di chuyển và tùy ý thao tác dữ liệu hàng khi chúng di chuyển dữ liệu từ các cột nguồn đến các cột đích. Hình 9.2 minh họa chi tiết hơn về kiểu kiểm tra DTP mới này.



Hình 9.2: Các thành phần của Data Transformation Pipeline

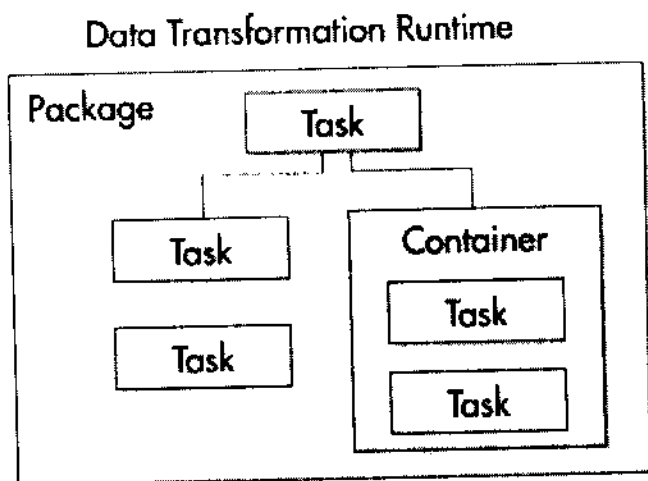
DTP sử dụng các bộ điều hợp dữ liệu để nối kết các điểm cuối nguồn và đích dữ liệu. Các bộ điều hợp dữ liệu nguồn (source data adapter) nối kết với nguồn của dữ liệu và cung cấp đầu vào cho các package Integration Services. Các bộ điều hợp dữ liệu đích (destination data adapter) nối kết với đích dữ liệu và xuất dữ liệu. SQL Server 2005 cung cấp nhiều bộ điều hợp dữ liệu nguồn và đích. Ngoài ra, Integration Services của SQL Server 2005 còn có kèm theo nhiều bộ điều hợp nguồn và đích cài sẵn, bao gồm các bộ điều hợp dành cho SQL Server, các file phẳng, và các nguồn dữ liệu khác tương thích OLE DB. Các bộ điều hợp dữ liệu cụ thể được hỗ trợ sẽ được trình bày chi tiết hơn trong mục “Các thành phần của Package Integration Services” ở sau trong chương này.

Trong khi công việc của các bộ điều hợp dữ liệu (data adapters) là thực hiện các nối kết với các điểm cuối nguồn và đích của dữ liệu, công việc của các tiến trình biến đổi Integration Services là di chuyển và tùy

ý thao tác dữ liệu khi nó được di chuyển giữa các điểm cuối nguồn và đích. Sự biến đổi Integration Services có thể đơn giản như một phép ánh xạ một một giữa các cột nguồn và cột đích, hoặc nó có thể phức tạp hơn nhiều, thực hiện các tác vụ chẳng hạn như di chuyển các cột một cách có chọn lọc giữa nguồn và đích, tạo các cột đích mới nhờ dùng các ánh xạ một đối nhiều hoặc tính các cột phái sinh. Integration Services của SQL Server 2005 có kèm theo nhiều tiến trình biến đổi cài sẵn được liệt kê trong mục “Các thành phần Integration Services” ở sau trong chương này. Ngoài các tiến trình biến đổi cài sẵn, bạn có thể tạo các tiến trình biến đổi tùy ý của riêng mình bằng cách tận dụng mô hình đối tượng DTP API.

Data Transformation Runtime (DTR)

DTR gồm có bộ máy DTR và các thành phần DTR. Các thành phần DTR là các đối tượng cho phép bạn điều khiển sự thực thi của Integration Services. Các thành phần DTR được sử dụng để tạo các dòng làm việc (work flow), các đối tượng chứa (container) vốn cung cấp các hoạt động có cấu trúc, các tác vụ (task) có khả năng truyền và biến đổi dữ liệu, và các ràng buộc (constraint) kiểm soát trình tự dòng làm việc trong một package. Hình 9.3 minh họa tổng quan về kiểu kiến trúc DTR mới.



Hình 9.3: Tổng quan về Data Transformation Runtime

Các thành phần DTR chính là các đối tượng chứa và các tác vụ. Các tác vụ (task) là các tập hợp thành phần DTR; mỗi tác vụ được tạo thành từ các nguồn và các đích dữ liệu cũng như các tiến trình biến đổi dữ liệu. Các đối tượng chứa (container) được sử dụng để tổ chức và cấu trúc các tác vụ liên quan. Các đối tượng chứa và các tác vụ này được nhóm lại với nhau để tạo thành các gói (package). Package Integration Service

là đơn vị vật lý nhóm mọi chức năng sẽ được thực hiện lại với nhau trong một hoạt động chuyển cho trước. Các package được thực thi bởi DTR để thực hiện các tiến trình chuyển dữ liệu. Các package Integration Services có thể được chạy lại một cách dễ dàng hoặc thậm chí được di chuyển đến một hệ thống khác và thực thi độc lập. Các thành phần cụ thể tạo nên DTR sẽ được trình bày chi tiết hơn trong mục "Các thành phần của Package Integration Services" ở sau trong chương này.

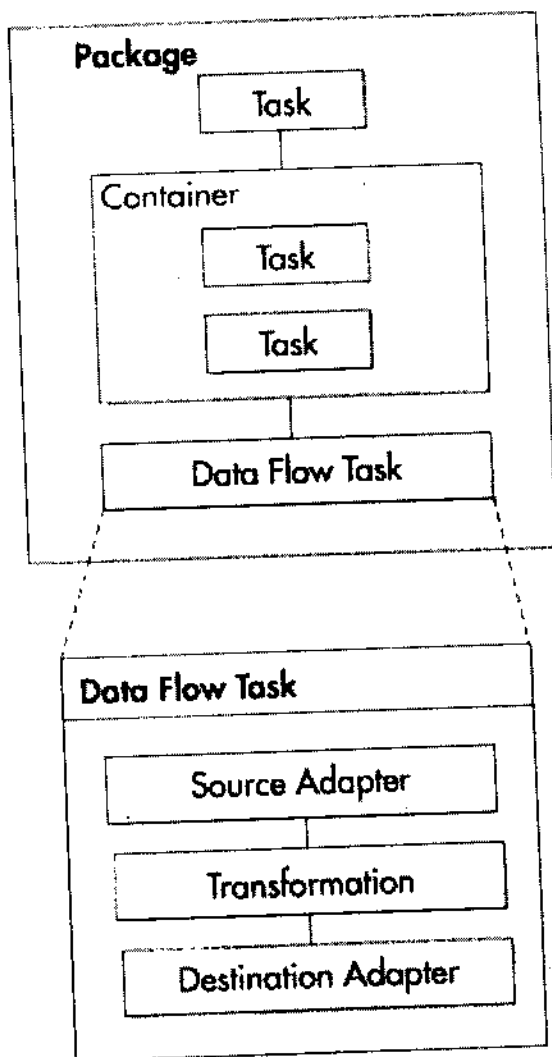
Mục đích chính của bộ máy DTR là điều khiển sự thực thi của các package Integration Services. DTR điều khiển dòng làm việc của các tác vụ chứa trong một package Integration Services. Ngoài ra, bộ máy DTR còn chứa kiểu trình bày package; chạy các package; và cung cấp các dịch vụ gỡ rối, ghi nhật trình, và xử lý event. Bộ máy DTR còn cho phép bạn quản lý các nối kết và truy cập các biến package Integration Services.

DTR được truy cập nhờ dùng cơ cấu đối tượng DTR. Cơ cấu đối tượng DTR vào lúc chạy là API hỗ trợ Integration Services Import/Export Wizard và Integration Services Designer ngoài công cụ dòng lệnh dtexec. Cả Import/Export Wizard và Designer đều được sử dụng để tạo các package. Bộ máy DTR được viết bằng C++ và API của nó được trình bày nhờ dùng cả C++ API gốc và hợp ngữ .NET vốn cho phép nó tiếp cận với những ứng dụng .NET đã được quản lý. Các chương trình sử dụng mô hình đối tượng DTR có thể tự động hóa việc tạo và thực thi các package Integration Services.

Các thành phần của Package Integration Services

Các package Integration Services là các tập hợp có tổ chức của các thành phần DTP và DTR. Package là đơn vị thực thi cho một hoạt động truyền Integration Services. Nói cách khác, để sử dụng Integration Services cho việc truyền dữ liệu và thực hiện các hoạt động ETL, bạn phải tạo một package có chứa tất cả những thành phần DTP vốn xác định các đích nguồn và đích đến cho dữ liệu và các tiến trình biến đổi sẽ xảy ra, cũng như các thành phần DTR vốn xác định dòng làm việc hay trình tự hoạt động sẽ được thực hiện bởi package Integration Services. Ngay khi package đã được tạo, bạn thực thi package để thực hiện việc truyền dữ liệu. Hình 9.4 minh họa tổng quan về một package Integration Services mẫu.

Các Package DTS



Hình 9.4: Tổng quan về package Integration Services

Các package Integration Services có thể được tạo nhờ dùng bộ công cụ được cung cấp với SQL Server 2005, hoặc chúng có thể được tạo bằng lập trình nhờ dùng DTR API. Các package Integration Services có thể được lưu trong cơ sở dữ liệu msdb của SQL Server hoặc ở dạng các file XML nằm trong các hệ thống file. Bạn có thể xem trước các công cụ thiết kế Integration Services trong mục “Các công cụ Integration Services” ở sau trong chương này.

Các tính năng của Package Integration Services

Kiểu kiến trúc Integration Services mới đã bổ sung thêm một số khả năng mới và có ý nghĩa vào Integration Services và điều đó biến nó trở thành một công cụ biến đổi dữ liệu mạnh mẽ hơn ở cấp doanh nghiệp. Trong các mục dưới đây, chúng ta sẽ xem xét một số phần cải tiến chính.

Xác lập các thuộc tính Run-Time (thời gian chạy) của các Package Integration Services

Một trong những hạn chế lớn trong các phiên bản trước đây của các Integration Services là các package bị buộc vào một nguồn và đích cho trước. Điều đó có nghĩa là thật khó lấy một package đã cho và dễ dàng sử dụng lại nó trên một server khác, cho dù server đó có hỗ trợ cùng các cơ sở dữ liệu. Tuy chúng ta có thể dùng các biến chung hay biến tổng thể và các giải pháp phức tạp khác để khắc phục hạn chế này nhưng các giải pháp đó không thật sự mạnh. SQL Server 2005 Integration Services cho phép một package xử lý nhiều điểm cuối nguồn và đích. Bạn có thể dùng những tính năng này để xác lập các công nghệ package vào lúc chạy từ các file thông số, Registry hay các tài liệu XML.

Ghi nhật trình

Integration Services không hỗ trợ bất kỳ khả năng ghi nhật trình (logging) cài sẵn nào. Việc ghi nhật trình sự hoạt động của các package đặc biệt hữu dụng cho việc kiểm tra và xử lý sự cố. Tuy bạn có thể tự mình thêm chúng vào nhưng công việc phát triển bắt buộc này dựa vào vai trò của bạn. SQL Server 2005 Integration Services mới cung cấp các tùy chọn ghi nhật trình (logging) cài sẵn cho các package, các tác vụ và các tiến trình biến đổi.

Điểm kiểm tra và sự khởi động lại

Một tính năng mới khác trong Integration Services của SQL Server 2005 là hỗ trợ các điểm kiểm tra "checkpoints" và sự khởi động lại "restarts" bên trong các package Integration Services. Tính năng mới này cho phép kết hợp các điểm kiểm tra khác nhau với nhiều bước trong các package Integration Services phức tạp. Do đó, nếu một package không thành công trong việc sử dụng các điểm kiểm tra thì toàn bộ package không cần được chạy lại từ phần bắt đầu. Thay vào đó, package có thể được khởi động lại tại điểm kiểm tra không thành công đầu tiên. Đối với các package chạy trong thời hạn lâu dài, đây có thể là một tiến trình giúp tiết kiệm rất nhiều thời gian, vì toàn bộ package không cần được xử lý lại. Package Integration Services có khả năng tiếp tục xử lý ở nơi nó đã dừng lại. Khi được kết hợp với tính năng xử lý event, nó trở

thành công cụ mạnh để quản lý tự động các tác vụ Integration Services của bạn.

Các biến

Hỗ trợ các biến là một tính năng mới khác trong SQL Server Integration Services. Với SQL Server 7 và SQL Server 2000, các package DTS đã hỗ trợ các biến để sử dụng trong các script biến đổi nhưng không hỗ trợ các biến thuộc phạm vi package. Việc thiếu các biến tổng thể này đã làm cho nó khó sử dụng lại các package DTS trên nhiều cơ sở dữ liệu và trên nhiều server. Để chắc chắn, đã có những giải pháp nhưng vì các package DTS không được thiết kế để sử dụng theo kiểu này nên các giải pháp đó hơi phức tạp và không mạnh mẽ. Với SQL Server 2005, kiểu thiết kế Integration Services mới cung cấp sự hỗ trợ cho package, tác vụ, và các biến ở cấp event tăng đáng kể tính linh hoạt của các package Integration Services và giúp dễ dàng sử dụng lại các package Integration Services. Các biến thuộc phạm vi package có thể được nhận thấy bởi tất cả các thành phần trong Integration Services. Các biến thuộc phạm vi tác vụ có thể được nhìn thấy và truy cập bởi các thành phần của tác vụ cụ thể đó nhưng không được bởi bất kỳ thành phần khác vốn là thành phần của package Integration Services. Một ví dụ về việc sử dụng biến Integration Services thuộc phạm vi package có thể là chứa tên của hệ thống SQL Server hiện hành, cho phép package dễ dàng được chạy trên một hệ thống khác bằng cách tên server của hệ thống đó thành biến Integration Services vào lúc chạy.

Các biến hệ thống

Tất cả các package Integration Services đều có nhiều biến hệ thống cài sẵn tiêu biểu cho các khía cạnh khác nhau của một package Integration Services. Bảng 9.1 liệt kê các biến hệ thống Integration Services mới.

Bảng 9.1: Các biến hệ thống Integration Services

Biến hệ thống	Mô tả
BreakpointTargetDescription	Chứa một phần mô tả của điểm ngắt (break point)
Cancel	Biểu thị sự thực thi sẽ bị hủy
CountDone	Chứa bộ đếm tiến độ chuyển
CreationDate	Chứa ngày tạo package
CreatorComputerName	Chứa tên của máy tính dùng để tạo package
CreatorName	Chứa tên của người dùng đã tạo package

CustomEventDescription	Mô tả một event tùy biến
CustomEventInfo	Chứa thông tin event tùy biến
CustomEventName	Chứa một tên event tùy biến
CustomEventValue	Chứa một trị event tùy biến
CustomEventGUID	Chứa GUID của event tùy biến
ErrorCode	Chứa một mã lỗi
ErrorDescription	Chứa một mô tả lỗi
ExecutionInstanceGuid	Chứa một GUID nhận biết phần thể hiện package hiện hành
ExecutionStatus	Chứa tình trạng thực thi của package
LocaleId	Chứa mã nhận dạng vị trí
MachineName	Chứa tên máy hiện hành
MaxCount	Chứa số hạng mục tối đa sẽ được chuyển
Packageld	Chứa phần nhận dạng package hiện hành
PackageName	Chứa tên package hiện hành
PercentComplete	Chứa tình trạng tiến triển của tiến trình chuyển
ProgressCountHigh	Chứa số đếm cao của tiến độ chuyển
ProgressCountLow	Chứa số đếm thấp của tiến độ chuyển
ProgressDescription	Chứa một phần mô tả về tiến độ chuyển
ProgressEvent	Chứa một phần nhận dạng event của tiến độ chuyển
Propagate	Cho biết một event có được lan truyền hay không
SourceDescription	Chứa một phần mô tả nguồn event
SourceID	Chứa một phần nhận dạng nguồn event
SourceName	Chứa một tên nguồn event
StartTime	Chứa thời gian bắt đầu thực thi của package
TaskID	Chứa một phần nhận dạng tác vụ
TaskName	Chứa một tên tác vụ
UserName	Chứa tên của user đang chạy package
VersionBuild	Chứa số tạo của package

VersionComment	Chứa phần mô tả của package
VersionGUID	Chứa GUID của package
VersionMajor	Chứa số phiên bản chính của package
VersionMinor	Chứa số phiên bản phụ của package

Điều khiển dòng chảy phức tạp

Integration Services của SQL Server 2005 còn hỗ trợ sự điều khiển dòng chảy phức tạp (complex flow control). Bên trong mỗi package Integration Services, bạn có thể chỉ định đường dẫn sẽ được lấy nếu một hoạt động cho trước thành công hay thất bại. Chẳng hạn, nếu một hoạt động thành công, bạn có thể cài đặt package Integration Services để tiến hành cùng với hành động kế tiếp. Ngược lại, nếu hoạt động thất bại bạn có thể lấy một hành động khác. Tính năng error flow (dòng chảy lỗi) mới này sẽ được trình bày chi tiết hơn trong mục sau. Bạn cũng có thể cài đặt dòng điều khiển (control flow) theo một cách sao cho nhiều tác vụ có thể được thực thi song song, hoặc bạn có thể buộc các tác vụ thực thi theo trình tự bằng cách chỉ định tác vụ kế tiếp sẽ không được thực thi chừng nào chưa hoàn tất tác vụ hiện tại. Bạn cũng có thể sử dụng một cấu trúc Integration Services mới được gọi là container (đối tượng chứa) để nhóm các tác vụ Integration Services có liên quan lại với nhau. Mỗi container có dòng điều khiển bên trong và các biến của riêng nó. Cũng có nhiều cấu trúc vòng lặp cho phép bạn cài đặt các hành động lặp lại. Có một container For Each Loop có khả năng lặp qua một nhóm các đối tượng, thực hiện hành động trên các đối tượng đó vào mỗi lần lặp. Ngoài ra, có một container For Loop có khả năng lượng giá một biểu thức và thực hiện các hành động lặp lại một cách có điều kiện. Các tác vụ này sẽ được trình bày chi tiết hơn ở sau trong chương này.

Các dòng lỗi

Một trong những tính năng mới và quan trọng của SQL Server 2005 Integration Services là khả năng hỗ trợ các dòng lỗi (error flow). Tính năng Error Flows mới về cơ bản cho phép bạn thêm khả năng xử lý lỗi vào các package Integration Services của bạn. Với tính năng Error flows, khi một tiến trình biến đổi Integration Services bắt gặp các hàng vốn cho ra các điều kiện lỗi, thay vì tạm ngưng tiến trình bằng một lỗi, Integration Services có thể tạo lộ trình hàng bị sự cố theo dòng lỗi đã được thiết lập. Chẳng hạn, dòng lỗi (error flow) có thể biểu thị hàng được viết một cách đơn giản ra một log file, hoặc nó cũng có thể hướng package vào một tác vụ vốn thực hiện các thường trình lỗi tình vi hơn nhiều, thậm chí có thể thao tác dữ liệu và đặt hàng vào trở lại đường truyền để xử lý lại.

Chế độ Immediate và chế độ Project

Tương tự như cách hoạt động của SQL Server 2000 DTS, trong đó DTS vừa có một giao diện wizard được thiết kế chủ yếu để thực thi các tiến trình truyền dữ liệu ad hoc vừa là một DTS Designer dùng để tạo các package DTS phức tạp hơn, Integration Services của SQL Server 2005 hỗ trợ cả hai chế độ immediate và project. Để sử dụng Integration Services trong chế độ immediate, bạn có thể chạy Integration Services Import/Export Wizard từ menu. Integration Services Import/Export Wizard có thể được sử dụng để tạo, thực thi và tùy ý lưu các package Integration Services vốn thực hiện các tiến trình truyền đơn giản. Integration Services Import/Export Wizard của SQL Server 2005 có diện mạo và cách hoạt động gần giống như phiên bản được tìm thấy trong SQL Server 2000.

Trong khi chế độ immediate hữu dụng cho các hoạt động truyền dữ liệu nhanh chỉ một lần thì chế độ project hữu dụng cho việc tạo các package Integration Services phức tạp hơn nhờ dùng Business Intelligence Development Studio. Business Intelligence Development Studio có chứa một Integration Services Designer hoàn toàn mới hỗ trợ một bộ các handler Data Flow, Control Flow, và Event dùng để tạo các package Integration Services. Integration Services Designer mới còn hỗ trợ hoàn toàn cho việc gỡ rối các package Integration Services. Integration Services Designer mới sẽ được trình bày chi tiết hơn một mục "Các công cụ Integration Services" ở sau trong chương này.

Ký nhận bằng kỹ thuật số của các Package Integration Services

Nhờ dùng một phương thức giống như tính năng ký nhận bằng kỹ thuật số dành sẵn cho các ứng dụng Microsoft .NET, các package Integration Services bây giờ cũng được ký nhận bằng kỹ thuật số. Điều này cho phép bạn xác thực một package khi nó được thực thi. Các package có thể được ký nhận bằng kỹ thuật số trong tiến trình thiết kế nhờ dùng Integration Services Designer. Ngay sau khi một package đã được ký nhận bằng kỹ thuật số, package đó là chỉ đọc và không còn được sửa đổi.

Các bộ điều hợp dữ liệu

DTP sử dụng các bộ điều hợp dữ liệu (data adapters) để nối kết các điểm cuối nguồn và đích dữ liệu. Các bộ điều hợp dữ liệu nguồn nối kết với nguồn của dữ liệu và cung cấp đầu vào cho các package Integration Services. Các bộ điều hợp dữ liệu đích nối kết với đích dữ liệu và xuất dữ liệu. SQL Server 2005 cung cấp nhiều bộ điều hợp dữ liệu nguồn và đích. Bảng 9.2 liệt kê các bộ điều hợp dữ liệu cài sẵn được cung cấp với SQL Server 2005.

Bảng 9.2: Các bộ điều hợp dữ liệu cài sẵn

Bộ điều hợp dữ liệu	Mô tả
Flat File Destination Adapter	Một bộ điều hợp hệ thống file viết dữ liệu phân cách text ra một file
Flat File Source Adapter	Một bộ điều hợp hệ thống file đọc dữ liệu phân cách text từ một file
OLE DB Destination Adapter	Một provider OLE DB viết dữ liệu ra một consumer OLE
OLE DB Source Adapter	Một consumer OLE DB đọc dữ liệu từ một provider OLE DB
Raw File Destination Adapter	Một bộ điều hợp hệ thống file viết dữ liệu ra một file
Raw File Source Adapter	Một bộ điều hợp hệ thống file đọc dữ liệu từ một file
SQL Server Destination Adapter	Một bộ điều hợp SQL Server dùng để viết dữ liệu vào một table hay khung xem (view)
Web Service Source Adapter	Một bộ điều hợp Web Service đọc dữ liệu từ một dịch vụ web XML

Các đối tượng chứa

Các đối tượng chứa (containers) là một cấu trúc mới mà Microsoft đã thêm vào Integration Services của SQL Server 2005. Mục đích chính cho các đối tượng chứa Integration Services là thêm cấu trúc và sự điều khiển dòng chảy vào các package Integration Services của bạn. Các đối tượng chứa nhóm các tác vụ có liên quan lại với nhau và được thiết kế để thực thi các tác vụ liên quan hay để cung cấp phạm vi cho các biến. Bảng 9.3 liệt kê các loại đối tượng chứa mà SQL Server 2005 Integration Services hỗ trợ.

Bảng 9.3: Các loại đối tượng chứa Integration Services

Đối tượng chứa	Mô tả
Package Container	Một tập hợp các phần tử package
Foreach Loop Container	Cung cấp dòng điều khiển lặp trong một package
For Loop Container	Cung cấp sự hỗ trợ cho các hành động lặp lại trong một package

Sequence Container	Nhóm các tác vụ và các đối tượng chứa có liên quan lại với nhau trong một package
TaskHost Container	Cung cấp các dịch vụ cho một tác vụ
Container Properties	Chứa các trị chung cho đối tượng chứa
Container Collections	Một tập hợp các đối tượng chứa

Các tác vụ

Các tác vụ Integration Services là những thành phần cơ bản nhất của package Integration Services. Tác vụ (task) Integration Services về cơ bản xác định một hành động sẽ được thực hiện. Các hành động này bao gồm sao chép các file, thực thi các control T-SQL và chạy các script cho đến thực hiện các tiến trình truyền FTP và chạy các mô hình khai thác dữ liệu. Nhiều tác vụ liên quan có thể được nhóm lại thành các đối tượng chứa (container). Bảng 9.4 liệt kê các tác vụ được đưa vào Integration Services SQL Server 2005.

Bảng 9.4: Các tác vụ Integration Services

Tác vụ	Mô tả
ActiveX Script Task	Thực thi một script ActiveX vốn thực hiện một hành động đã định
Analysis Services Execute DDL Task	Thực thi các câu lệnh T-SQL DDL
Analysis Services Processing Task	Xử lý các đối tượng Analysis Services
Bulk Insert Task	Chèn dữ liệu từ một file text vào một table
Data Flow Task	Sao chép và biến đổi dữ liệu giữa các nguồn dữ liệu
Data Mining Query Task	Thực thi các mẫu truy vấn khai thác dữ liệu
Execute Package Task	Thực thi các package khác
Execute Process Task	Thực thi một chương trình hay script
Execute SQL Task	Thực thi các câu lệnh T-SQL
File System Task	Thực thi các hành động trên hệ thống file
File Transfer Protocol Task	Thực thi các tiến trình truyền dữ liệu FTP

Message Queue Task	Gửi và nhận các thông báo từ các hàng đợi dữ liệu MSMQ
Script Task	Thực thi các script được viết bằng VB.NET nhờ dùng môi trường Microsoft Visual Studio for Applications (VSA).
Send Mail Task	Gửi một thông báo e-mail
XML Task	Truy cập dữ liệu trong các tài liệu XML

Các tiến trình biến đổi

Các tiến trình biến đổi Integration Services (Integration Services transformations) điều khiển những gì sẽ xảy ra với dữ liệu khi nó được chuyển từ bộ điều hợp dữ liệu nguồn đến bộ điều hợp dữ liệu đích. SQL Server 2005 hỗ trợ cả hai tiến trình truyền cài sẵn và tiến trình truyền tùy ý do người dùng ấn định. Bạn có thể tạo các tiến trình truyền tùy ý nhờ dùng API do mô hình đối tượng DTP cung cấp. Integration Services của SQL Server 2005 cung cấp một danh sách mở rộng của các tiến trình biến đổi chuẩn được cài sẵn, như minh họa trong bảng 9.5.

Bảng 9.5: Các tiến trình biến đổi chuẩn được cài sẵn

Tiến trình biến đổi	Mô tả
Aggregate Transformation	Thực hiện các phép gộp
Allocation Transformation	Phân tán trị của một cột nhập liệu qua nhiều cột xuất
Character Map Transformation	Áp dụng các hàm chuỗi cho dữ liệu ký tự
Conditional Split Transformation	Lượng giá dữ liệu và truyền nó đến các đầu ra khác
Copy/Map Transformation	Tạo các cột xuất bằng cách sao chép các cột nhập liệu
Data Mining Model Accuracy Transformation	Tính độ chính xác của các mô hình khai thác dữ liệu
Data Mining Model Training Transformation	Huấn luyện các mô hình khai thác dữ liệu
Data Mining Query Transformation	Chạy các mẫu truy vấn dự đoán việc khai thác dữ liệu
Data Conversion Transformation	Chuyển đổi dữ liệu của một cột nhập sang một kiểu dữ liệu xuất khác

Derived Column Transformation	Tạo một cột xuất từ kết quả của các biểu thức
Dimension Processing Transformation	Xử lý các kích thước khối OLAP
File Extractor Transformation	Đọc dữ liệu từ dòng chảy dữ liệu của một package và viết dữ liệu đó vào file
File Injector Transformation	Đọc dữ liệu từ một file và thêm dữ liệu đó vào dòng chảy dữ liệu của một package
Fuzzy Grouping Transformation	Chuẩn hóa các trị trong dữ liệu cột nhập
Fuzzy Lookup Transformation	Tra tìm các trị trong một table tham chiếu nhờ dùng sự tương kết
Logged Lineage Transformation	Cung cấp thông tin môi trường cho dòng chảy dữ liệu của package
Lookup Transformation	Tra tìm các trị dữ liệu trong một table tham chiếu nhờ dùng sự tương kết chính xác
Merge Transformation	Trộn hai dataset đã được phân loại
Merge Join Transformation	Nối hai dataset nhờ dùng một phép nối FULL, LEFT, hay INNER
Multicast Transformation	Phân bố dữ liệu nhập đến nhiều đầu ra
Partition Processing Transformation	Xử lý các phần chia OLAP
Pivot Transformation	Pivot dữ liệu nhập theo một trị của cột nhập liệu
Row Count Transformation	Đếm các hàng nhập liệu và chứa số đếm đó trong một biến
Sampling Transformation	Tạo một mẫu tiêu biểu cho dataset nhập liệu
Script Transformation	Thực thi một script để biến đổi dữ liệu nhập
Slowly Changing Dimension Transformation	Phối hợp việc cập nhật và chèn các hàng thành các kích thước OLAP

Sort Transformation	Phân loại dữ liệu nhập và sao chép dữ liệu đã được phân loại sang kết xuất biến đổi
Surrogate Key Transformation	Cung cấp thêm các thuộc tính tùy ý cho package Integration Services
Union All Transformation	Trộn nhiều dataset
UnPivot Transformation	Unpivot dữ liệu nhập theo một trị của cột nhập liệu

Xử lý Event

Khả năng đưa ra và xử lý các event là một tính năng mới khác trong SQL Server 2005

Integration Services. Việc xử lý event cho phép các package Integration Services phản hồi những event được đưa ra vào lúc chạy bởi các đối tượng chứa và tác vụ. Các event có thể được kích khởi bởi các thành phần package Integration Services để báo hiệu nhiều trạng thái khác nhau, bao gồm cả các điều kiện lỗi, khi một tác vụ đã bắt đầu, khi một tác vụ hoàn tất, hay một sự thay đổi trong tình trạng biến. Bảng 9.6 liệt kê các Event Handler của Integration Services.

Bảng 9.6: Các Event Handler của Integration Services

Event Handler	Mô tả
OnCustomerEvent	Được đưa ra theo yêu cầu bởi một tác vụ hay package
OnError	Được đưa ra bởi một tác vụ hay đối tượng chứa vào lúc có lỗi xảy ra
OnExecStatusChanged	Được đưa ra bởi một tác vụ hay đối tượng chứa khi tình trạng thực thi của nó thay đổi
OnPostExecute	Được đưa ra bởi một tác vụ hay đối tượng chứa sau khi nó đã chạy
OnPostValidate	Được đưa ra bởi một tác vụ sau khi nó đã được hợp chuẩn
OnPreExecute	Được đưa ra bởi một tác vụ hay đối tượng chứa trước khi nó chạy
OnPreValidate	Được đưa ra bởi một tác vụ trước khi nó được hợp chuẩn
OnProgress	Được đưa ra bởi một tác vụ hay đối tượng chứa khi bắt gặp một package tiến độ đã định

OnQueryCancel	Được đưa ra bởi một tác vụ hay đối tượng chứa để xác định nó có nên ngừng chạy hay không
OnTaskFailed	Được đưa ra bởi một tác vụ khi nó không thành công
OnVariableValueChanged	Được đưa ra một cách tùy ý bởi một biến khi trị của nó thay đổi
OnWarning	Được đưa ra bởi một tác vụ hay đối tượng chứa khi một cảnh báo được tạo ra.

Các Log Provider

Các log provider là một tính năng hoàn toàn mới khác mà Microsoft đã thêm vào các package Integration Services. Các log provider Integration Services cho phép bạn thêm tính năng ghi nhật trình vào các package, các đối tượng chứa và các tác vụ của bạn. Ghi nhật trình (logging) được sử dụng để ghi lại những thông tin lỗi hay thông tin quan trọng khác về thời gian chạy hay tình trạng. Bảng 9.7 liệt kê các log provider mà Microsoft đã đưa vào SQL Server 2005.

Bảng 9.7: Các Log Provider

Log Provider	Mô tả
The Text File log provider	Viết các mục nhập log vào các file text ASCII nhờ dùng dạng CSV (Comma-Separated Value), sử dụng phần mở rộng file mặc định là .log.
The SQL Profiler log provider	Viết dữ liệu SQL trace vào một trace file nhờ dùng phần mở rộng file mặc định là .trc
The SQL Server log provider	Viết các mục nhập log vào table sysdtslog90 trong một cơ sở dữ liệu SQL Server
The Windows Event log provider	Viết các mục nhập vào script Application
The XML File log provider	Viết các log dữ liệu vào một file XML nhờ dùng phần mở rộng file mặc định là .xml

Các công cụ Integration Services

Với kiểu kiến trúc hoàn toàn mới, bộ công cụ Integration Services đã được cải tiến hoàn chỉnh và có một diện mạo mới. Tuy có một số điểm tương tự trong một số công cụ đơn giản hơn - chẳng hạn Integration Services Import/Export Wizard được sử dụng để thực hiện các hoạt động truyền dữ liệu ad hoc - các công cụ tinh vi hơn chẳng hạn như Integration Services Designer thì hoàn toàn khác. Dù bạn là một chuyên gia về DTS hay là một người mới sử dụng DTS, Integration Services của SQL Server 2005 đều đòi hỏi bạn phải biết cách sử dụng các công cụ mới từ phần cơ bản. Trong mục này, bạn sẽ xem tổng quan về các công cụ Integration Services mới và những tiện ích đi kèm với SQL Server.

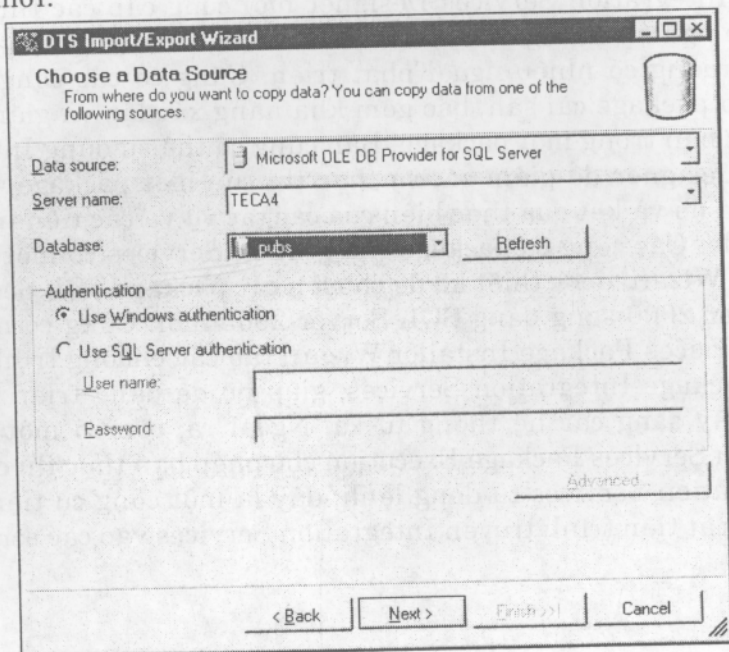
Bộ công cụ Integration Services của SQL Server 2005 được phân chia ra thành hai vùng cơ bản: các công cụ dùng để tạo và thực thi các package Integration Services và các công cụ dùng để làm việc với các package DTS hiện có. Các công cụ Integration Services Designer bao gồm Integration Services Import/Export Wizard cơ bản và Designer cao cấp hơn. Như bạn sẽ thấy trong mục kế tiếp, Integration Services Designer phản ánh sự phân chia về mặt cấu trúc của dòng điều khiển và dòng chảy dữ liệu bằng cách cung cấp các bộ hiệu chỉnh khác nhau cho việc thiết kế dòng chảy dữ liệu và dòng điều khiển package. Integration Services Designer cung cấp một dạng biểu diễn bằng đồ họa của dòng làm việc (work flow) vốn xảy ra bên trong một package Integration Services. Integration Services Designer mới cung cấp các tính năng ở cấp doanh nghiệp chẳng hạn như điều khiển nguồn để tạo thuận lợi cho các môi trường có nhiều người phát triển. Cũng có các công cụ triển khai và gỡ package cài sẵn, bao gồm khả năng xác lập và giám sát các điểm ngắt bên trong một package. Bạn cũng có thể sử dụng Integration Services Designer để giám sát sự thực thi của một package - toàn bộ tiến độ của nó và kết quả thực hiện của các tác vụ và các tiến trình biến đổi riêng lẻ. Các công cụ package Integration Services có một Package Migration Wizard được thiết kế để chuyển các package SQL Server 7 và SQL Server 2000 sang dạng SQL Server 2005 mới. Cũng có một Integration Services Package Installer Wizard tạo các chương trình cài đặt cho các package Integration Services, giúp nó dễ dàng triển khai các package này sang các hệ thống từ xa. Ngoài ra, còn có một tiện ích Integration Services Package Execution cho phép bạn thực thi các package Integration Services từ dòng lệnh; đây là một công cụ tiện lợi cho việc hợp nhất tiến trình truyền Integration Services vào các script quản lý của bạn.

Các công cụ thiết kế Integration Services

Trong mục này, bạn sẽ xem xét kỹ hơn về các công cụ thiết kế Integration Services mới. Trước tiên, bạn sẽ xem Integration Services Import/Export Wizard, sau đó là Integration Services Designer.

Integration Services Import/Export Wizard

Integration Services Import/Export Wizard là điểm vào đầu tiên của SQL Server 2005 được hợp nhất vào Integration Services mới. Giống như các tiền thân của nó trong SQL Server 7 và SQL Server 2000, SQL Server 2005 Integration Services Import/Export Wizard một loạt các hộp thoại đưa bạn qua tiến trình chọn nguồn dữ liệu, đích dữ liệu và các đối tượng sẽ được chuyển. Integration Services Import/Export Wizard còn cho phép bạn tùy ý lưu và thực thi các package Integration Services. Bạn có thể mở Integration Services Import/Export Wizard bằng cách chọn tùy chọn Integration Services Import/Export từ menu SQL Server hoặc bằng cách nhập dtswizard tại dòng lệnh. Việc lưu các package được tạo với Integration Services Import/Export Wizard và sau đó hiệu chỉnh chúng trong Integration Services Designer là một cách tuyệt vời mà bạn nên biết thêm về Integration Services - đặc biệt nếu bạn chỉ đang mới bắt đầu với Integration Services hoặc nếu bạn đang chuyển tiếp sang SQL Server 2005 Integration Services mới từ một trong các phiên bản trước. Hình 9.5 minh họa Integration Services Import/Export Wizard mới.



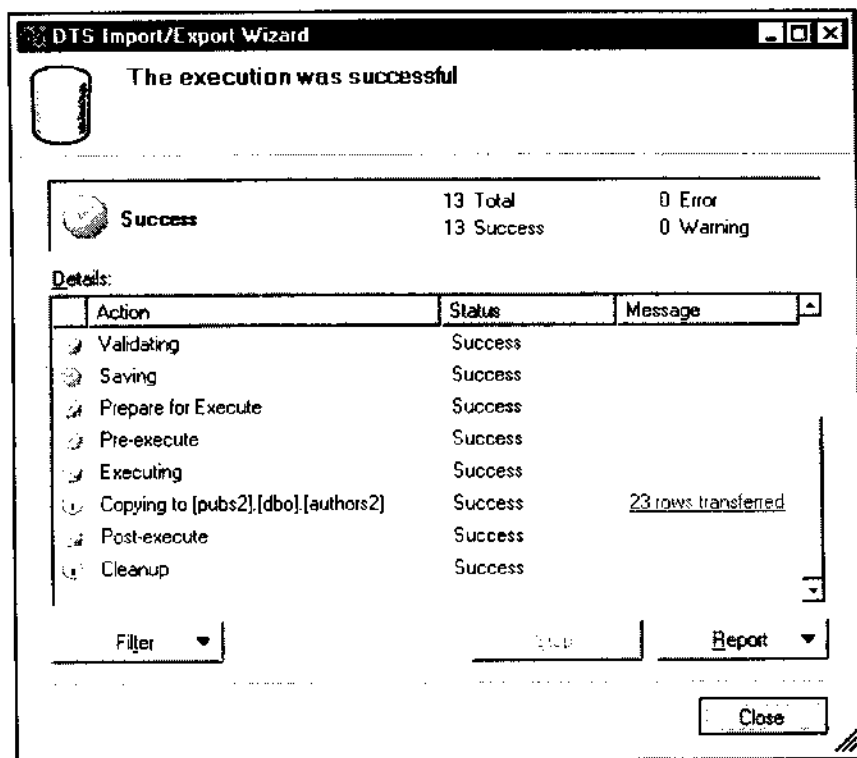
Hình 9.5: Integration Services Import/Export Wizard - Data Source Selection

Integration Services Import/Export Wizard trước tiên đưa bạn qua tiến trình chọn một nguồn dữ liệu. Trong danh sách xổ xuống Data Source, bạn chọn provider mà bạn muốn sử dụng. Tùy thuộc vào provider mà bạn chọn, các tùy chọn dành cho phần còn lại của màn hình sẽ thay đổi. Nếu bạn chọn Microsoft OLE DB Provider for SQL Server, bạn sẽ thấy một màn hình như minh họa ở hình 9.5, trong đó bạn chọn server mà bạn muốn nối kết với, tiếp sau đó là cơ sở dữ liệu và loại xác thực mà bạn cần sử dụng. Việc nhấp Next sẽ đưa bạn qua các hộp thoại wizard kế tiếp. Hộp thoại kế tiếp cho phép bạn chọn đích dữ liệu, về cơ bản hộp thoại này giống như hộp thoại chọn nguồn dữ liệu ngoại trừ nó xác định nơi dữ liệu sẽ được chuyển đến.

Lưu ý

Trong khi SQL Server 2005 Integration Services Wizard cung cấp tính năng cơ bản giống như Integration Services Import/Export Wizard đã được đưa vào SQL Server 7 và 2000, phiên bản mới này không có khả năng thực hiện các phép ánh xạ dữ liệu tùy ý cũng như khả năng nhập các script biến đổi dữ liệu tùy ý. Để sử dụng các tính năng cao cấp hơn này, bạn cần dùng Integration Services Designer.

Sau khi bạn chọn nguồn và đích dữ liệu, wizard nhắc bạn tùy ý lưu và thực thi package Integration Services. Khi mỗi tác vụ trong package thực thi, cửa sổ chuyển (transfer) được cập nhật động, cho thấy tiến độ chuyển của package Integration Services. Khi package Integration Services đã được thực thi thành công, Integration Services Import/Export Wizard sẽ hiển thị hộp thoại như bạn thấy ở hình 9.6.



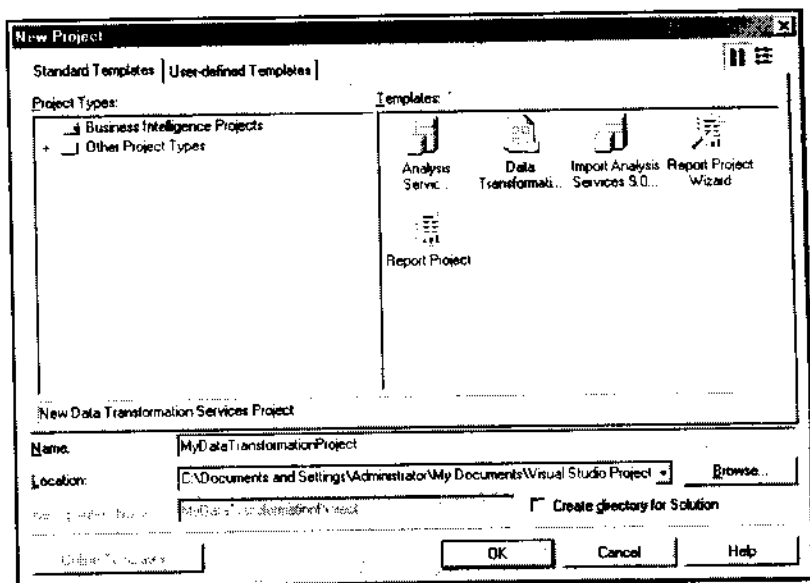
Hình 9.6: Integration Services Import/Export Wizard - Package Execution

Integration Services Designer

Trong khi Integration Services Import/Export Wizard hữu dụng cho các tiến trình truyền ad hoc đơn giản thì các tác vụ trích, biến đổi, và tải (ETL) đòi hỏi sự tinh vi hơn và mạnh hơn wizard Integration Services. Về bản chất, các tác vụ ETL vượt xa các tiến trình truyền dữ liệu đơn giản từ đích này sang đích khác. Chúng thường kết hợp dữ liệu từ nhiều nguồn, thao tác dữ liệu, ánh xạ các giá trị sang các cột mới, tạo các cột từ các trị đã được tính và cung cấp nhiều loại tác vụ làm sạch cũng như xác nhận dữ liệu. Các tính năng này nằm ngoài phạm vi của Integration Services Import/Export Wizard đơn giản. Đó chính là vai trò của Integration Services Designer mới. Integration Services Designer là một bộ công cụ đồ họa mà bạn có thể sử dụng để tạo, thực thi và gỡ rối các package Integration Services. Bạn dùng Integration Services Designer để mở các package Integration Services đơn giản đã được tạo nhờ dùng, hoặc bạn có thể sử dụng nó để tạo các giải pháp chuyển và biến đổi dữ liệu mạnh.

Integration Services Designer được khởi động từ Business Intelligence Development Studio bằng cách chọn File > New > Project để mở

hợp thoại New Project. Sau đó để tạo một dự án Integration Services mới, bạn chọn Business Intelligence Projects từ danh sách Project Types sau đó chọn Data Transformation Project từ danh sách các Templates, như minh họa ở hình 9.7.



Hình 9.7: Mở một dự án biến đổi dữ liệu

Lưu ý

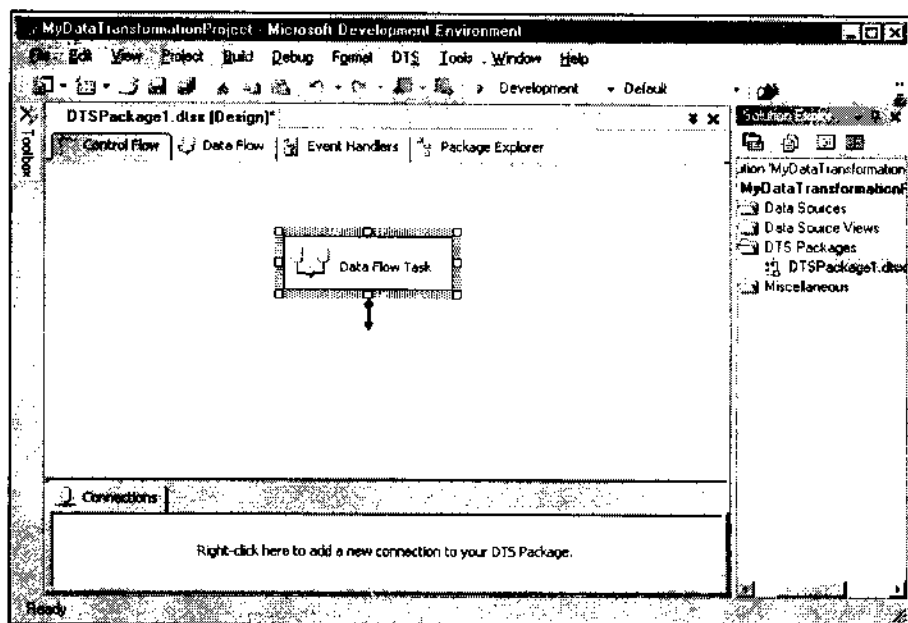
Đừng nhầm lẫn bởi việc Integration Services Designer được khởi động từ Business Intelligence Development Studio. Nó không bị giới hạn trong chỉ các dự án Analysis Services. Integration Services Designer và các dự án được phát triển trong Business Intelligence Development Studio hoàn toàn có khả năng hoạt động với kiểu dữ liệu quan hệ và các kiểu dữ liệu khác, không bị giới hạn trong chỉ dữ liệu analysis service (dịch vụ phân tích).

Ngay sau khi bạn đã tạo dự án (project), bạn có thể mở Integration Services Designer bằng cách nhấp phải Packages trong cửa sổ Solution Explorer hiển thị ở phần bên phải của màn hình. Sau đó chọn tùy chọn New Package để khởi động Integration Services Designer. Khi Integration Services Designer lần đầu tiên khởi động, bạn sẽ thấy một bề mặt thiết kế trông không giống như các Integration Services Designer trước đây, vì vậy sự khởi đầu có thể là một thách thức đối với bạn.

Control Flow Designer mới

Trong khi bạn có thể sử dụng Integration Services Designer theo nhiều cách khác nhau, phương pháp dễ dàng nhất có lẽ là bắt đầu với

tab Control Flow rồi mở Control Flow Toolbox. Sau khi Toolbox được hiển thị, nhấp tác vụ Data Flow rồi rê nó sang bề mặt thiết kế. Bề mặt Control Flow của Integration Services Designer sẽ có dạng như minh họa ở hình 9.8.



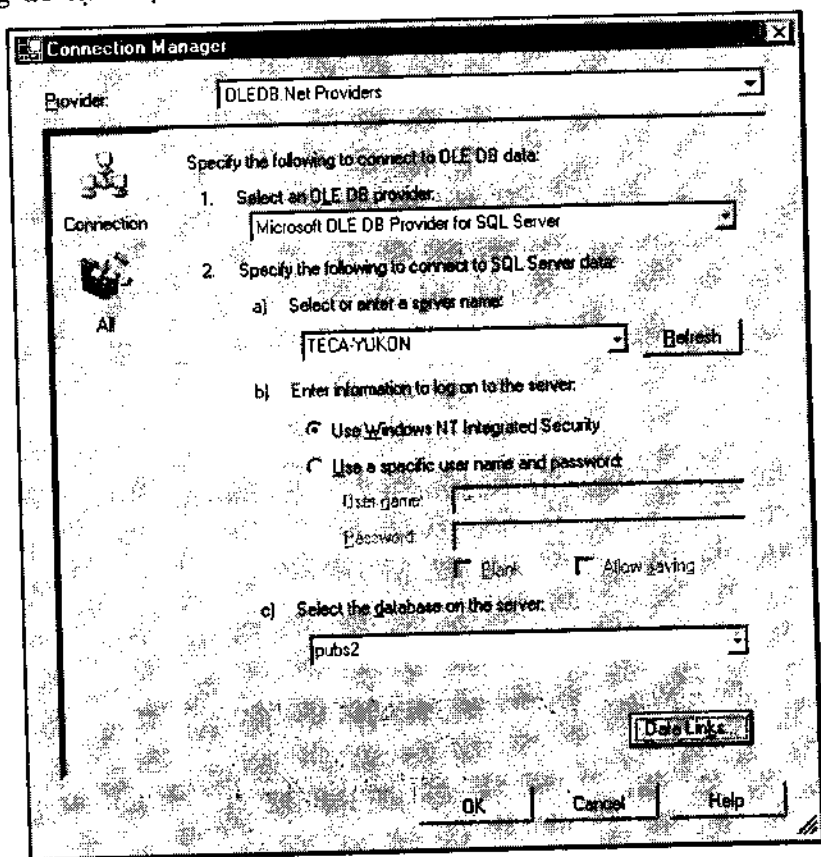
Hình 9.8: Control Flow Designer

Vào lúc này, package Integration Services khá đơn giản vì nó chỉ có một tác vụ Data Flow. Rõ ràng bạn có thể làm cho package này trở nên phức tạp hơn nhiều bằng cách bổ sung thêm các tác vụ từ toolbox Control Flow và thậm chí bố cục nhiều tác vụ liên quan thành các đối tượng chứa (container).

Data flow Designer mới

Vào lúc này, package biết rằng nó sẽ thực hiện một hoạt động chuyển dữ liệu, nhưng nó không biết cần chuyển những dữ liệu nào hoặc dữ liệu bắt nguồn từ nơi nào hay được chuyển đến nơi nào. Sau khi thêm tác vụ Data Flow, bước kế tiếp là xác định các dòng chảy dữ liệu thật sự. Để xác định các dòng chảy dữ liệu, bạn nhấp double tác vụ Data Flow để hiển thị tab Data Flow. Ban đầu, tab Data Flow sẽ trống. Để thêm các nối kết nguồn dữ liệu và đích dữ liệu, bạn mở Data Flow Toolbox ở phía bên trái của IDE rồi rê và thả một OLE DB Source data flow và một mục OLE DB Destination data flow sang bề mặt Data Flow Designer. Ngoài ra, bạn có thể thêm một Flat File Destination dùng để xuất bất kỳ các hàng lỗi sang một log file ASCII.

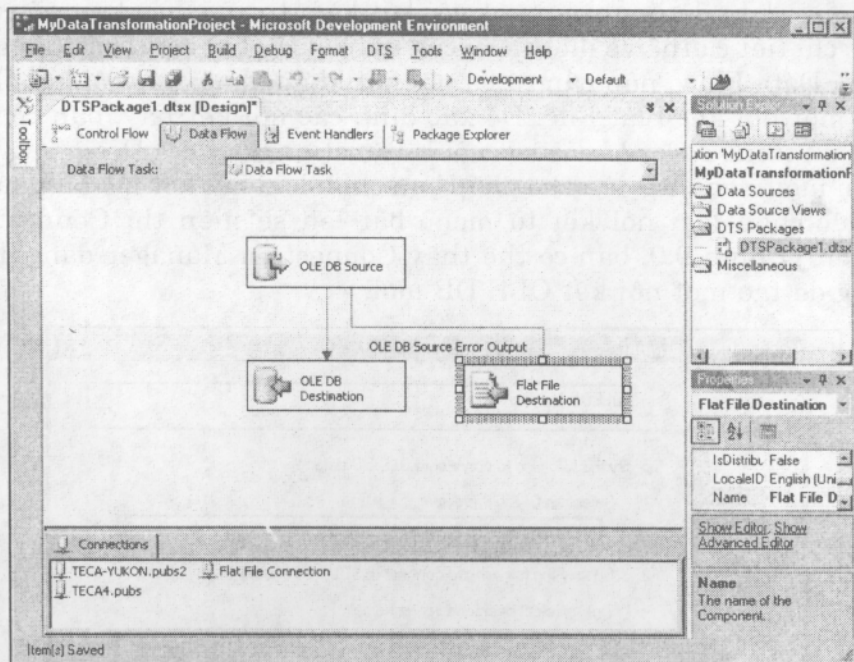
Bây giờ bạn đã cho Integration Services biết rằng bạn đang chuyển dữ liệu từ một nguồn OLE DB sang một đích OLE DB khác, nhưng bạn vẫn cần xác định các nối kết nguồn và đích. Ngoài ra, bạn cần cài đặt một nối kết cho flat file error flow. Về cơ bản, mỗi nguồn hay đích data flow (dòng chảy dữ liệu) cần một nối kết kèm theo, về cơ bản nó cung cấp các chi tiết cụ thể về điểm cuối của nối kết. Bởi vì ví dụ này dùng ba mục của Data Flow, nên cần có ba nối kết khác nhau: hai nối kết OLE DB và một nối kết Flat File. Để xác định các nối kết này, bạn có thể nhấp phải trong ô Connection ở cuối Integration Services Designer, vốn hiển thị một menu bật lên cho phép bạn chọn loại nối kết mà bạn muốn tạo. Việc chọn loại nối kết từ menu bật lên sẽ hiển thị Connection Manager. Ở hình 9.9, bạn có thể thấy Connection Manager đang được sử dụng để tạo một nối kết OLE DB mới.



Hình 9.9: Connection Manager

Sau khi các nối kết đã được tạo, bạn cần kết hợp mỗi nối kết với mục data flow thích hợp. Để liên kết một nối kết với một data flow, bạn nhấp phải mục data flow, chọn edit hoặc tùy chọn Advanced Edit từ menu ngữ cảnh, sau đó chọn nối kết từ danh sách xổ xuống với các nối

kết hiển thị trong hộp thoại Edit. Khi bạn xác định mỗi nối kết, bạn chỉ định ánh xạ cột và bất kỳ tiến trình biến đổi khác sẽ được thực hiện. Khi bạn hoàn chỉnh cấu hình, tab Data Flow của Integration Services Designer sẽ xuất hiện giống như minh họa ở hình 9.10.

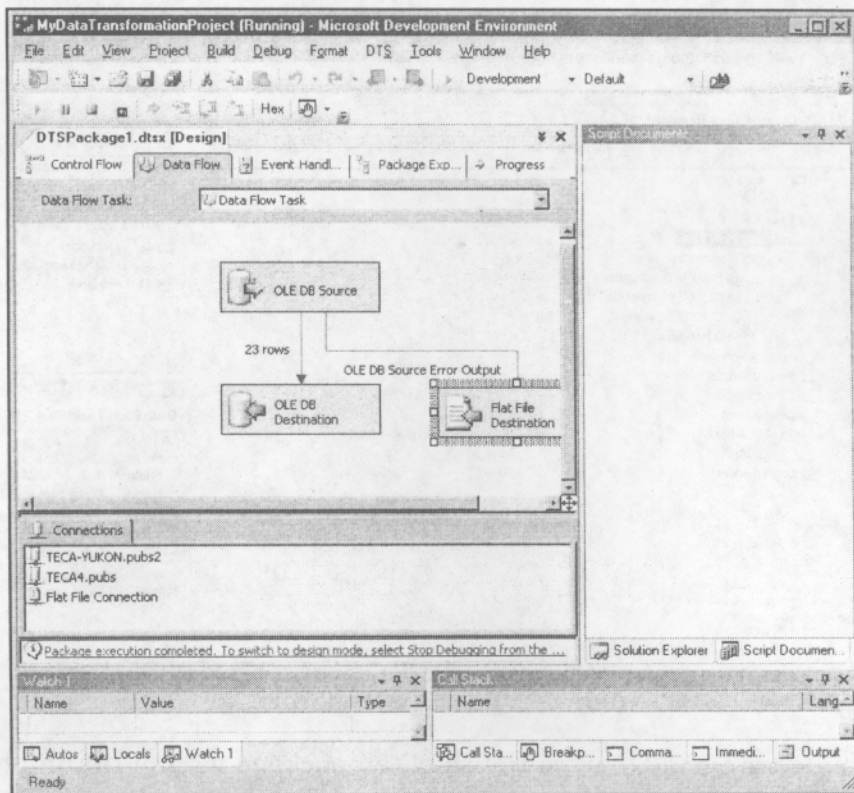


Hình 9.10: Data Flow Designer

Ở giữa tab Data Flow Design, bạn có thể thấy các mục data flow OLE DB Source, OLE DB Destination, và Flat File Destination. Một data flow với mũi tên thẳng màu xanh lục nối OLE DB Source với OLE DB Destination, và một error flow với một mũi tên cong màu đỏ nối OLE DB Source với Flat File Destination. Các nối kết của mỗi mục này được hiển thị trong ô Connections ở cuối màn hình hiển thị.

Dạng biểu diễn bằng đồ họa của tiến trình thực thi Package

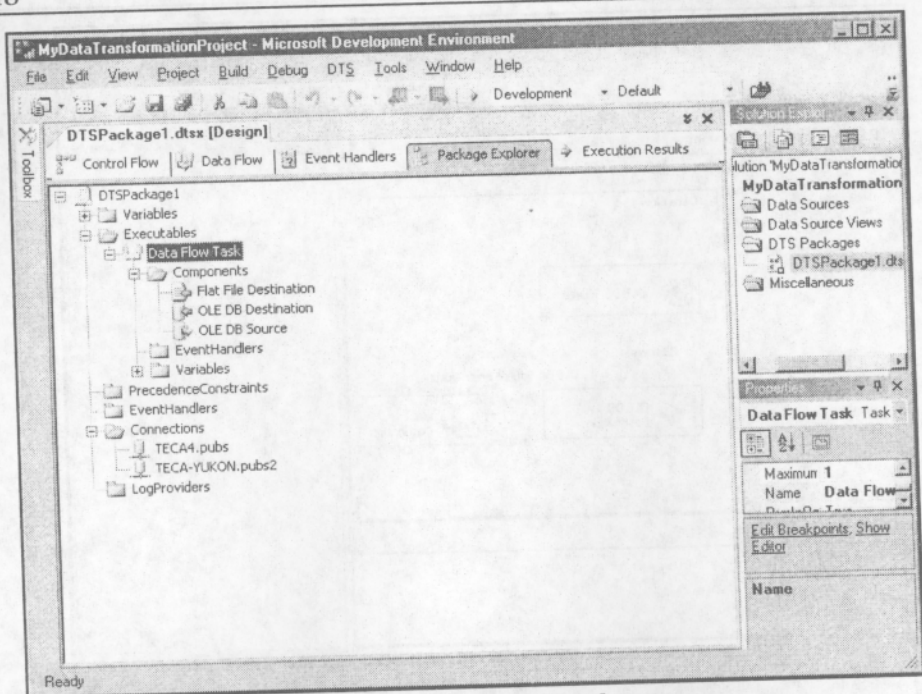
Ngay sau khi mọi data flow và các nối kết đã được xác định, package có thể được chạy bằng cách nhấp vào mũi tên màu xanh lục trên thanh công cụ hoặc bằng cách chọn tùy chọn Start từ menu Debug. Trước khi chạy package, bạn có thể tùy ý xác định các điểm ngắt trong package bằng cách chọn một mục, tiếp sau đó là tùy chọn New Breakpoint từ menu Debug. Khi package được chạy, các cửa sổ Watch và Call Stack tự động được mở và Data Flow Item đang thực thi đổi sang màu xanh lục. Bạn có thể thấy kết quả chạy package mẫu ở hình 9.11.



Hình 9.11: Chạy package bên trong Designer

Package Explorer mới

Một tính năng mới khác trên Integration Services Designer là Package Explorer. Package Explorer cung cấp một khung xem cây phân cấp của Integration Services Package vốn được hiển thị trong designer. Hình 9.12 minh họa Package Explorer.

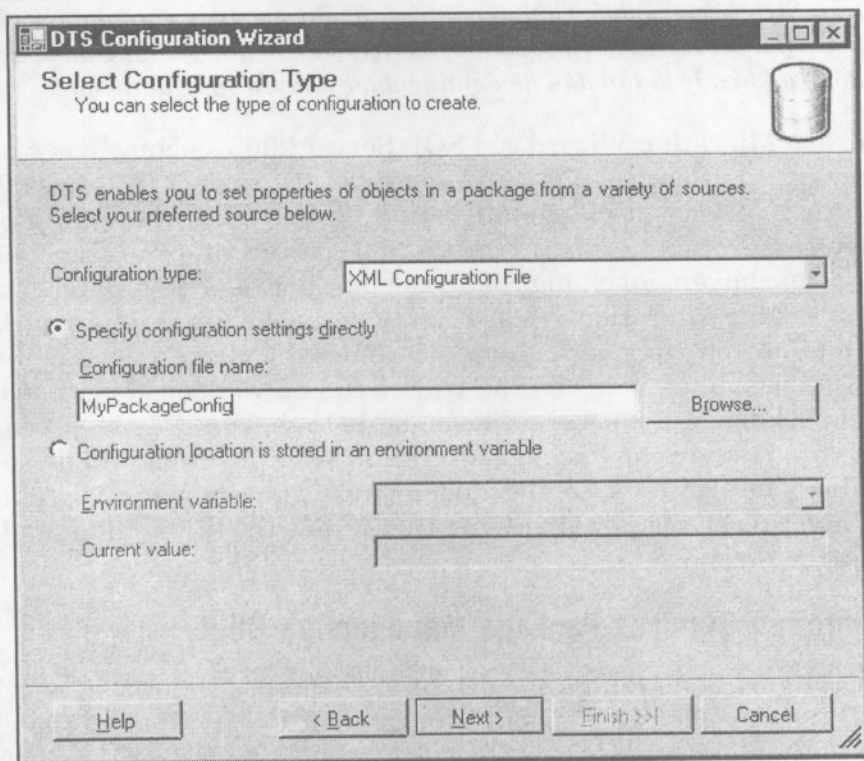


Hình 9.12: Package Explorer

Ở cấp độ trên cùng, bạn có thể thấy tên package và mỗi thành phần của package được liệt kê ngay bên dưới package. Nhấp vào dấu cộng ở phía trước mỗi thành phần để hiển thị các hạng mục thật sự. Cần lưu ý rằng package Explorer không chỉ là một công cụ view-only (chỉ xem). Từ khung xem Explorer, bạn có thể xóa các mục cũng như hiệu chỉnh các thuộc tính của chúng. Tuy nhiên, bạn có thể thêm các mục mới. Điều đó có thể được thực hiện chỉ trong bề mặt thiết kế.

Sự cấu hình

Các tính năng cấu hình Integration Services mới được thiết kế để giúp bạn dễ dàng triển khai các package Integration Services. Tính năng cấu hình mới cho phép bạn cập nhật động một package để chạy trong một môi trường khác. Chẳng hạn, các nối kết đòi hỏi các chuỗi nối kết và các chuỗi nối kết này thường thích hợp trong chỉ môi trường đã cho. Các cấu hình cho phép bạn cập nhật động các kiểu trị thuộc site/server khi package được triển khai. Bạn có thể tạo các cấu hình nhờ dùng Configuration Wizard. Bạn có thể khởi động Configuration Wizard nhờ dùng tùy chọn menu Integration Services | Configurations để hiển thị hộp thoại Package Configurations Organization, trong hộp thoại này bạn có thể nhấp Add để khởi động Configuration Wizard. Hình 9.13 minh họa Configuration Wizard.



Hình 9.13: Configuration Wizard

Các công cụ đóng gói Integration Services

Mục cuối cùng của chương này sẽ giới thiệu cho bạn một số tiện ích quản lý package khác được cung cấp với SQL Server 2005. Một Package Migration Wizard được cung cấp nhằm giúp bạn chuyển đổi các package Integration Services hiện có thành dạng package mới của SQL Server 2005. Cũng có một tiện ích Package Management cho phép bạn làm việc với các package Integration Services hiện có. Ngoài ra, một công cụ dòng lệnh mới cho phép bạn chạy các package từ dòng lệnh.

Package Migration Wizard

Xem xét sự thay đổi hoàn toàn về kiểu kiến trúc và sự lựa chọn .NET Framework, không có gì đáng ngạc nhiên khi các package DTS được tạo cho SQL Server 7 và SQL Server 2000 phải được chuyển sang dạng mới trước khi chúng có thể được sửa đổi.

Lưu ý

Các Package DTS hiện có được tạo cho SQL Server 2000 có thể chạy

trên SQL Server 2005. Chúng cũng có thể đưa vào các giải pháp package SQL Server 2005 Integration Services mới. Chúng không được sửa đổi trừ khi đây là lần đầu tiên chúng được chuyển sang dạng mới.

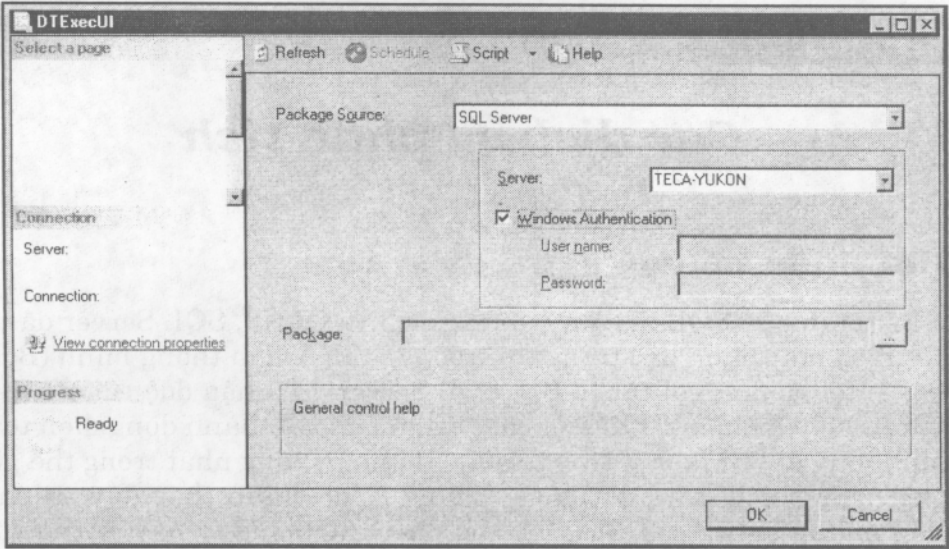
Package Migration Wizard của SQL Server 2005 sẽ chuyển các package Integration Services hiện có sang dạng package SQL Server 2005 Integration Services. Package Migration Wizard cố lấy các thành phần package Integration Services hiện có như các tác vụ, các ràng buộc và các biến rồi chuyển đổi chúng thành các mục package SQL Server tương đương. Package Migration Wizard có thể xử lý các thành phần package DTS do Microsoft cung cấp, nhưng nó không thể chuyển đổi các tác vụ tùy ý. Các tác vụ tùy ý duy trì cấu trúc cũ của nó và được gói gọn ở dạng một subpackage liên kết với package đã được chuyển. Các script có thể là một vấn đề khác cho Package Migration Wizard. Trong khi đa số các script biến đổi ActiveX có thể được chuyển đổi sang tác vụ ActiveX Script mới thì các script hiện có vốn tham chiếu mô hình đối tượng DTS cũ không được chuyển đổi.

Integration Services Package Management Utility

Một công cụ dòng lệnh khác được cung cấp với Integration Services của SQL Server 2005 là Package Management Utility. Bạn có thể chạy Package Management Utility bằng cách nhập dtutil trên dòng lệnh. Công cụ dtutil cho phép bạn truy cập các package chứa trong cơ sở dữ liệu msdb của SQL Server. Nó cũng có thể được sử dụng để sao chép, xóa hay ký nhận các package hiện có.

Các tiện ích thực thi Package Integration Services

Hai tiện ích khác dùng để thực thi các package là dtexec và dtexecui. Công cụ dtexec được chạy từ dòng lệnh, trong khi công cụ dtexecui hiển thị một giao diện người dùng nhằm cho phép bạn tải và chạy các package Integration Services. Một khía cạnh đáng lưu ý của công cụ dtexec là để tăng cường bảo mật nó có thể được chạy nhờ dùng các đối số được mã hóa. Công cụ dtexecui có thể được sử dụng để tạo một file text chứa các lệnh và các thông số cần thiết để chạy công cụ dòng lệnh dtexec (bao gồm cả tùy chọn để tạo các đối số được mã hóa). Hình 9.14 minh họa công cụ dtexecui.



Hình 9.14: Công cụ dtexecui.

CHƯƠNG 10

Các dịch vụ phân tích

Giới thiệu sơ lược

Kể từ ấn bản SQL Server 7 và các dịch vụ OLAP, SQL Server đã trở thành bộ máy đứng đầu trong thị trường kinh doanh thông minh (Business Intelligence, gọi tắt là BI). SQL Server 7 và hậu duệ của nó SQL Server 2000 đã biến BI từ một công nghệ mới trở thành dòng điện toán chính. Bây giờ BI là một trong những thị phần nóng nhất trong thế giới công nghệ thông tin. Một trong những lý do chính thúc đẩy sự phát triển nhanh chóng của BI là nó cho phép một công ty hay tổ chức lấy thông tin và giá trị từ các tài nguyên thông tin nhiều hơn trước đây. Điều này đã trở thành một yếu tố quan trọng trong nhiều năm qua. Với các tài nguyên bạn đã có, ngành công nghệ thông tin đã tăng thêm một giá trị mang tính chiến lược cho các tổ chức kinh doanh.

BI cho phép một doanh nghiệp lấy được nhiều thông tin ý nghĩa hơn từ dữ liệu hoạt động mà nó đang sử dụng trong dòng ứng dụng kinh doanh của nó. Bằng cách lấy dữ liệu hoạt động từ các ứng dụng nhập đơn đặt hàng, vận chuyển hàng và bán hàng, các công nghệ OLAP, lưu trữ dữ liệu và khai thác dữ liệu cho phép các doanh nghiệp tích lũy và tập hợp các mẫu thông tin quan trọng theo những cách mà không thể thực hiện được với các kỹ thuật truy cập dữ liệu quan hệ thuần túy. Kết quả sau cùng cho thấy nhiều thông tin ý nghĩa hơn về doanh nghiệp của bạn cũng như các yếu tố quan trọng tác động đến nó. Không có bữa ăn nào miễn phí và BI cũng không ngoại lệ đối với qui tắc này. Con đường đến với BI là không miễn phí nhưng trong quá trình quản lý lâu dài, bạn sẽ thấy các phí này là hợp lý. Khả năng lấy được nhiều thông tin ý nghĩa hơn từ dữ liệu hoạt động giúp biến IT thành một điểm chiến lược cốt lõi cho công việc kinh doanh của bạn.

Trong năm qua, thuật ngữ BI đã phát triển vượt xa ý nghĩa thông thường, trong đó BI về cơ bản đồng nghĩa với OLAP. Bây giờ BI đã mở rộng để đưa vào tất cả những công nghệ nào cho phép doanh nghiệp cung cấp thông tin mang tính quyết định. Với SQL Server 2005, Microsoft chắc chắn đã lựa chọn định nghĩa mở rộng này của BI. BI đã chiếm một vai trò lớn trong SQL Server 2005. Không chỉ OLAP và sự phân tích, BI còn có nhiều công cụ thông tin khác chẳng hạn như Integration Services và Reporting Services. Các chương 8 và 9 đã trình bày các tính năng mới trong Integration Services và Reporting Services một cách

chi tiết. Trong chương này, bạn sẽ xem tổng quan về các tính năng Analysis Services mới của SQL Server 2005.

BI là một điểm nhấn quan trọng trong ấn bản này của SQL Server. Nó có rất nhiều tính năng mới, nhưng trong chương này, bạn sẽ được giới thiệu về những tính năng mới quan trọng nhất trong SQL Server 2005 Analysis Services. Phần đầu tiên của chương này sẽ trình bày ngắn gọn về OLAP và vai trò của Analysis Services. Kế tiếp, bạn sẽ được giới thiệu về một số tính năng quan trọng nhất trong Analysis Services Engine và mô hình kích thước thống nhất (Unified Dimensional Model) mới của Microsoft. Sau cùng, chương này trình bày các tính năng quản lý và phát triển Analysis Services cũng như các thuật toán khai thác dữ liệu mới mà Microsoft đã thêm vào Analysis Services.

Tổng quan về Analysis Services

Các dịch vụ phân tích (Analysis Services) và các dịch vụ OLAP trước đây của nó ban đầu được thiết kế là một giải pháp báo cáo cho dữ liệu được chứa trong một data warehouse (kho lưu trữ dữ liệu) hay data mart (chợ dữ liệu). Thông tin chứa trong một data warehouse thường không phải là dữ liệu quan hệ mà là dữ liệu tóm tắt được lấy từ một data store (cửa hiệu dữ liệu) quan hệ. Dữ liệu tóm tắt này được chứa nhờ dùng một phương pháp đã tối ưu để đạt được khả năng báo cáo đặc biệt nhanh và linh hoạt. Các kiểu lưu trữ dữ liệu này được chứa ở dạng các schema hình sao hay hình bông tuyết. Dữ liệu trong một data warehouse hay data mart được xử lý nhờ dùng công nghệ xử lý phân tích trực tuyến (Online Analytical Processing, gọi tắt là OLAP). Không giống như công nghệ quan hệ, vốn cho ra kết quả bằng việc đọc và nối dữ liệu khi mẫu truy vấn được ấn hành, OLAP được tối ưu hóa để định hướng dữ liệu tóm tắt nhằm cho ra các kết quả một cách nhanh chóng. Cải thiện hiệu suất truy vấn là lý do thúc đẩy OLAP. Ví dụ, để có được tổng doanh số bán hàng từ các địa phương, khu vực, quốc gia và rộng khắp thế giới cho một tổ chức nào đó, một mẫu truy vấn quan hệ có thể phải xử lý hàng trăm hàng hoặc thậm chí hàng triệu hàng - một tiến trình có thể rất dài ngay cả trên những hệ thống nhanh nhất. Ngược lại, do hoạt động chủ yếu trên thông tin tóm tắt nên OLAP có thể cần đọc chỉ hai hoặc ba điểm dữ liệu để có được cùng đáp án. Rõ ràng, điều này giúp đạt được một hiệu suất rất cao. Sự thực thi nhanh hơn nhiều của OLAP cho phép truy vấn và xử lý dữ liệu đặc biệt mà không khả thi khi dùng các cách truy cập dữ liệu quan hệ thông thường.

Thay vì làm việc với các tập hợp table liên quan, các công nghệ OLAP làm việc với các khối (cube), bao gồm các kích thước và số đo. Một kích thước (dimension) là một loại có tính mô tả. Chẳng hạn, một kích thước có thể là một vị trí địa lý hay một loại sản phẩm. Một số đo

(measure) là một giá trị định lượng chẳng hạn như số tiền bán hàng, số lượng hàng tồn kho hay tổng chi phí. Các phép gộp phái sinh từ nguồn dữ liệu ban đầu được chứa trong mỗi ô của khối. Phương pháp bố cục dữ liệu này giúp dễ dàng lọc dữ liệu cũng như làm cho các mẫu truy vấn kế tiếp trở nên nhanh chóng và hiệu quả hơn. Tuy nhiên, có một sự cân đối. Trong khi các phép gộp OLAP là chìa khóa để có thể đạt được hiệu suất truy vấn trong các mẫu truy vấn data warehouse thì chi phí lưu trữ dữ liệu gộp là dung lượng đĩa. Thật vậy, số lượng phép gộp có thể dễ dàng vượt quá số lượng hàng chi tiết ban đầu. Ngoài ra, dòng ứng dụng kinh doanh thường không chứa dữ liệu của chúng trong các cơ sở dữ liệu OLAP. Thay vào đó, dữ liệu để tải một data warehouse thường được trích từ các cơ sở dữ liệu quan hệ và được tải đến cơ sở dữ liệu OLAP trong một tiến trình được gọi là ETL (Extraction Transformation and Loading).

Các loại lưu trữ OLAP

Ba phương pháp chính được sử dụng để lưu trữ dữ liệu kích thước dùng trong data warehousing là: multidimensional OLAP (MOLAP), relational OLAP (ROLAP), và hybrid OLAP (HOLAP). Mỗi phương pháp này có các yêu cầu lưu trữ dữ liệu và tốc độ truy xuất dữ liệu đặc trưng riêng của nó. SQL Server 2005 hỗ trợ cả ba phương pháp này.

MOLAP

Multidimension OLAP (MOLAP) chứa kích thước và dữ kiện trong một data store nhờ dùng các index nén. Các phép gộp được chứa để giúp truy cập nhanh dữ liệu. Các bộ máy truy vấn MOLAP thường độc quyền và được tối ưu hóa cho dạng lưu trữ mà data store MOLAP sử dụng. MOLAP cung cấp tiến trình xử lý mẫu truy vấn nhanh hơn ROLAP và thường cần ít dung lượng lưu trữ hơn. Tuy nhiên, nó không có khả năng mở rộng tốt bằng và yêu cầu phải có một cơ sở dữ liệu riêng để lưu trữ.

ROLAP

Relational OLAP (ROLAP) chứa các phép gộp (aggregate) trong các table cơ sở dữ liệu quan hệ. Việc sử dụng ROLAP trong các cơ sở dữ liệu quan hệ giúp nó tận dụng các tài nguyên cơ sở dữ liệu hiện có, ngoài ra nó còn cho phép các ứng dụng ROLAP mở rộng tốt. Tuy nhiên, việc sử dụng ROLAP trong các table để chứa các phép hợp thường cần nhiều dung lượng lưu trữ đĩa hơn MOLAP và không nhanh bằng.

HOLAP

Hybrid OLAP (HOLAP) là một sự bắt chéo giữa MOLAP và ROLAP. Giống như ROLAP, HOLAP để dữ liệu chính được lưu trữ trong cơ sở dữ liệu nguồn. Giống như MOLAP, HOLAP chứa các phép gộp trong một

data store tách biệt với cơ sở dữ liệu quan hệ chính. Điều này giúp HOLAP có các ưu điểm của cả MOLAP và ROLAP. Tuy nhiên, không giống như MOLAP và ROLAP, vốn tuân theo các chuẩn đã định, HOLAP không có sự thực thi đồng bộ.

Sau đây chúng ta sẽ xem một số cải tiến mới trong Analysis Services của SQL Server 2005.

Các cải tiến của bộ máy Analysis Services

Một trong những vùng cải tiến chính của Analysis Services là ở chính bộ máy Analysis Services. Nhiều cải tiến này đã khắc phục được những hạn chế có trong phiên bản SQL Server 2000 của Analysis Services và nhiều cải tiến khác đã mang đến cho sản phẩm này những vùng tính năng hoàn toàn mới.

Hỗ trợ nhiều instance

Với SQL Server 2000, Analysis Services không hỗ trợ nhiều instance cho dù bộ máy cơ sở dữ liệu quan hệ có khả năng hỗ trợ đến 16 instance (thể hiện). Sự hỗ trợ nhiều instance đặc biệt hữu dụng cho các ASP (Application Services Provider) khi có nhiều khách hàng, mỗi khách hàng duy trì các instance cơ sở dữ liệu của riêng mình và tất cả đều nằm trên một server chia sẻ. Các phiên bản trước của Analysis Services chưa thật sự được triển khai trong những tình huống này. Với SQL Server 2005, Analysis Services bây giờ hỗ trợ đến 50 instance trên mỗi server. Các instance của SQL Server 2005 Analysis Services cũng có thể được cài đặt để chạy cạnh nhau với các phiên bản trước của Analysis Services.

Hỗ trợ Failover Clustering

SQL Server 2005 đã mở rộng sự hỗ trợ failover clustering cho Analysis Services. SQL Server 2000 Analysis Services không hỗ trợ failover clustering cho Analysis Services. Tiến trình cài đặt SQL Server 2005 Analysis Services có khả năng nhận biết cluster và liên tục cài đặt Analysis Services trên các nút cluster. Với SQL Server 2005, failover clustering bây giờ nhận biết Analysis Services, SQL Server Agent, và Notification Services biến failover clustering thành một giải pháp khả dụng hoàn toàn ở cấp server.

Sự hợp nhất với .NET Framework

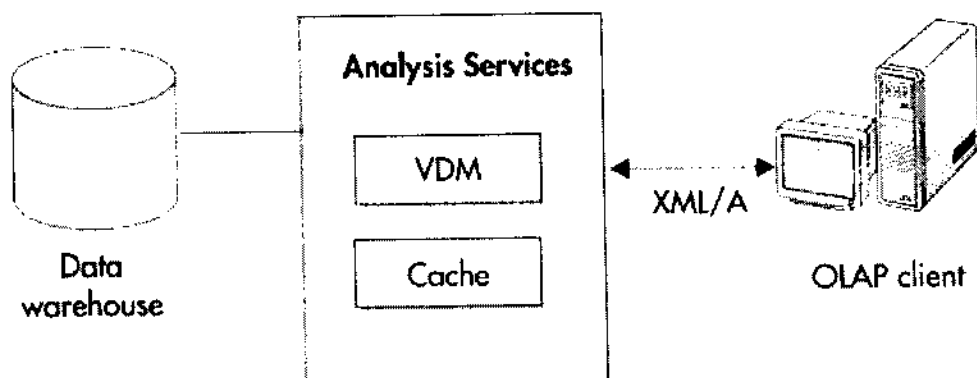
Cũng giống như bộ máy cơ sở dữ liệu quan hệ SQL Server 2005 được hợp nhất với .NET Framework, Analysis Services cũng có sự hợp nhất .NET. Sự hợp nhất Analysis Services .NET mới cho phép hỗ trợ đầy đủ

XML và symbol Object Access Protocol (SOAP) cũng như hỗ trợ cho việc tạo các thủ tục lưu trữ và các kích khởi (trigger) trong các ngôn ngữ .NET chẳng hạn như Visual Basic và C#.

Mô hình kích thước thống nhất

Một trong những thay đổi lớn nhất của Analysis Services trong SQL Server 2005 là sự giới thiệu mô hình kích thước thống nhất (Unified Dimensional Model, gọi tắt là UDM). Được xem là bước cải tiến kế tiếp trong xử lý OLAP vượt xa các khối, UDM hứa hẹn cung cấp một mô hình báo cáo thống nhất bằng cách kết hợp phần tốt nhất của OLAP và báo cáo quan hệ. Với các công nghệ trước đây, một số báo cáo (report) dựa vào tính quan hệ chẳng hạn như các đơn đặt hàng và các hóa đơn thì rất khó tạo khi dùng các công cụ OLAP. Tương tự, kiểu truy vấn đặc biệt với hiệu suất cao của báo cáo mà OLAP hỗ trợ không thể được điều ứng tốt với báo cáo quan hệ (relational reporting). UDM cung cấp một "sân chung" có khả năng xử lý cả hai loại yêu cầu rất khác biệt này. Hình 10.1 minh họa một tổng quan cấp cao của mối quan hệ giữa các ứng dụng và UDM.

Unified Dimensional Model



Hình 10.1 Mô hình kích thước thống nhất

Trong SQL Server 2005, khối về cơ bản là dạng biểu diễn bên ngoài của UDM. Trong khi một khối vẫn được trình bày cho ứng dụng báo cáo bên trong thì cơ chế để truy cập dữ liệu lại hoàn toàn khác. UDM chứa metadata kích hoạt nhiều tính năng mà các khối MOLAP, ROLAP hay HOLAP trong SQL Server 2000 Analysis Services không có. Trong SQL Server 2005, các ứng dụng OLAP nối với Analysis Services nhờ dùng XMLA và truy vấn UDM, vốn có thể được tạo trực tiếp trên cả hai nguồn dữ liệu phân tích và quan hệ.

Proactive Caching

UDM có thể tự động lưu trữ dữ liệu trong cache, cung cấp sự truy cập dữ liệu kiểu MOLAP nhanh chóng mà không cần xác định rõ sự lưu trữ MOLAP. Bằng cách dùng các control kiểu thanh trượt, bạn có thể điều khiển độ trễ và khoảng thời gian tồn tại của dữ liệu trong cache. Một xác lập độ trễ và 0 có nghĩa là mọi dữ liệu sẽ được lưu trữ trong cache ở dạng dữ liệu MOLAP. Thanh trượt data lafttime điều khiển dữ liệu sẽ tồn tại trong cache khoảng bao lâu. Bạn có thể chỉ định các thời khoảng thay đổi, chẳng hạn như hàng ngày, hàng tuần hay hàng tháng và khi hết thời khoảng này cache sẽ được xóa sạch. Bên trong, proactive cache sử dụng một cấu trúc trên đĩa tương tự như một khối MOLAP.

Proactive caching thật sự khắc phục được những hạn chế triển khai khối vốn có trong SQL Server 2000 Analysis Services. Với SQL Server 2000 Analysis Services, một khối cần được xử lý (được tập hợp với dữ liệu) trước khi nó có thể được triển khai. Với các tập hợp dữ liệu lớn, thời gian xử lý các tập hợp này có thể rất dài. Proactive caching giải quyết vấn đề này bằng cách cho phép bạn triển khai các khối trước khi chúng được xử lý. Khối được tập hợp tự động khi các yêu cầu được đưa ra cho dữ liệu.

Hỗ trợ trigger (bộ kích khởi)

Sự hỗ trợ trigger trong SQL Server 2005 Analysis Services là một cải tiến quan trọng khác. Giống như các bảng đối chiếu cơ sở dữ liệu quan hệ của chúng, các trigger Analysis Services có thể kích khởi các thủ tục lưu trữ khi một hành động cơ sở dữ liệu nhất định xảy ra. Các trigger Analysis Services chạy đồng bộ, nghĩa là việc kích khởi trigger bị chặn lại cho đến khi thủ tục lưu trữ đã kích khởi được thực thi.

Hỗ trợ sự truy vết

Một cải tiến quan trọng khác cho Analysis Services của SQL Server 2005 là hỗ trợ sự truy vết. Các event truy vết thì không đồng bộ và được sử dụng để giám sát hiệu suất hệ thống cũng như xử lý các sự cố.

Hỗ trợ scripting

Analysis Services của SQL Server 2005 bây giờ có hỗ trợ việc tạo các cơ sở dữ liệu Analysis Services và các đối tượng thông qua scripting (tạo kịch bản). SQL Server 2005 cung cấp ngôn ngữ định nghĩa đối tượng (Object Definition Language, gọi tắt là ODL) dựa trên XML mới, ngôn ngữ này có thể dùng để tạo, chỉnh sửa và xóa các đối tượng cơ sở dữ liệu Analysis Services. ODL cũng có khả năng khởi tạo những hành động server như xử lý khối và so sánh các phiên bản cơ sở dữ liệu. ODL sẽ được trình bày chi tiết hơn ở sau trong chương này.

Các cải tiến cục bộ hóa

Một cải tiến lớn khác trong bộ máy Analysis Services là nâng cao sự hỗ trợ cục bộ hóa. Bộ máy Analysis Services của SQL Server 2005 có thể chứa thông tin đối tượng trong nhiều ngôn ngữ. Điều này cho phép các ứng dụng Analysis Services hiển thị cả metadata khối và dữ liệu kinh doanh trong ngôn ngữ gốc của người dùng cuối. Bộ máy Analysis Services còn hỗ trợ các xác lập ngôn ngữ mặc định cho các ứng dụng client. Điều này giúp các ứng dụng client đa ngôn ngữ tự động dùng ngôn ngữ thích hợp để hiển thị thông tin đối tượng.

Hỗ trợ các hàng dữ kiện mồ côi trong table

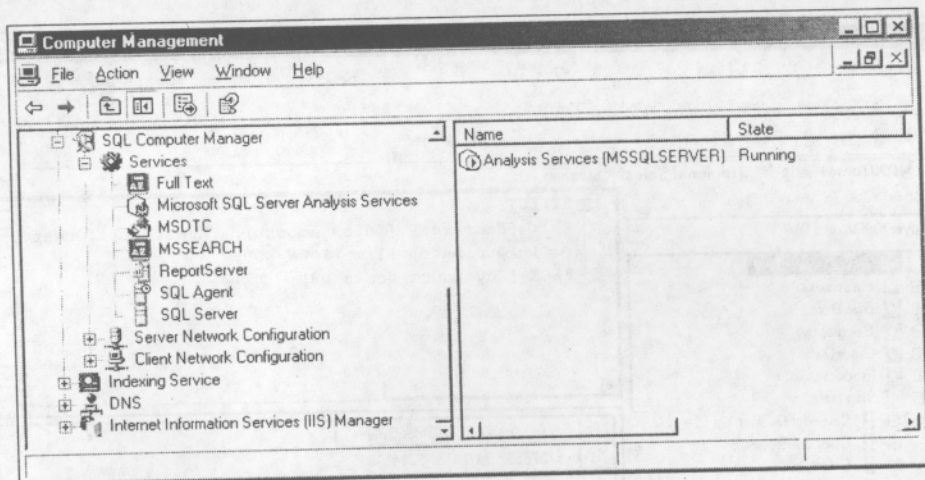
Một điểm hạn chế khác của SQL Server 2000 Analysis Services mà SQL Server 2005 khắc phục là vấn đề của các hàng mồ côi (orphaned rows). SQL Server 2000 Analysis Services bỏ qua các hàng có một thành phần chưa xác định cho một kích thước. Điều này dẫn đến sự không tương kết giữa các tổng khối khi được so sánh với dữ liệu từ nguồn dữ liệu. Analysis Services trong SQL Server 2005 cho phép bạn chỉ định cách bộ máy Analysis Services xử lý các hàng dữ kiện trong table vốn có thông tin kích thước bị thiếu. Bạn có thể chọn tiếp tục bỏ qua thông tin bị thiếu này hoặc buộc Analysis Services tạo một thành phần kích thước Unknown cho một hàng dữ kiện trong table đang thiếu thông tin kích thước này.

Các cải tiến về sự quản lý Analysis Services

Các công cụ quản lý cho Analysis Services đã thay đổi hoàn toàn trong SQL Server 2005. Trong ấn bản SQL Server 2000, Analysis Services đã được quản lý nhờ dùng Analysis Manager. Với SQL Server 2005, Analysis Manager cũ không còn nữa. Nó đã được thay bằng SQL Server Computer Manager và SQL Server Management Studio.

SQL Server Computer Manager

SQL Server Computer Manager là công cụ dùng để mở và dừng dịch vụ Analysis Services. Bạn truy cập bằng cách nhấp phải My Computer rồi chọn tùy chọn Manage từ menu bật lên. Để mở và dừng các dịch vụ phân tích (Analysis Services), bạn mở rộng nút Services and Applications rồi chọn nút SQL Computer Manager. Thao tác này sẽ hiển thị màn hình như minh họa ở hình 10.2.



Hình 10.2 SQL Server Computer Manager

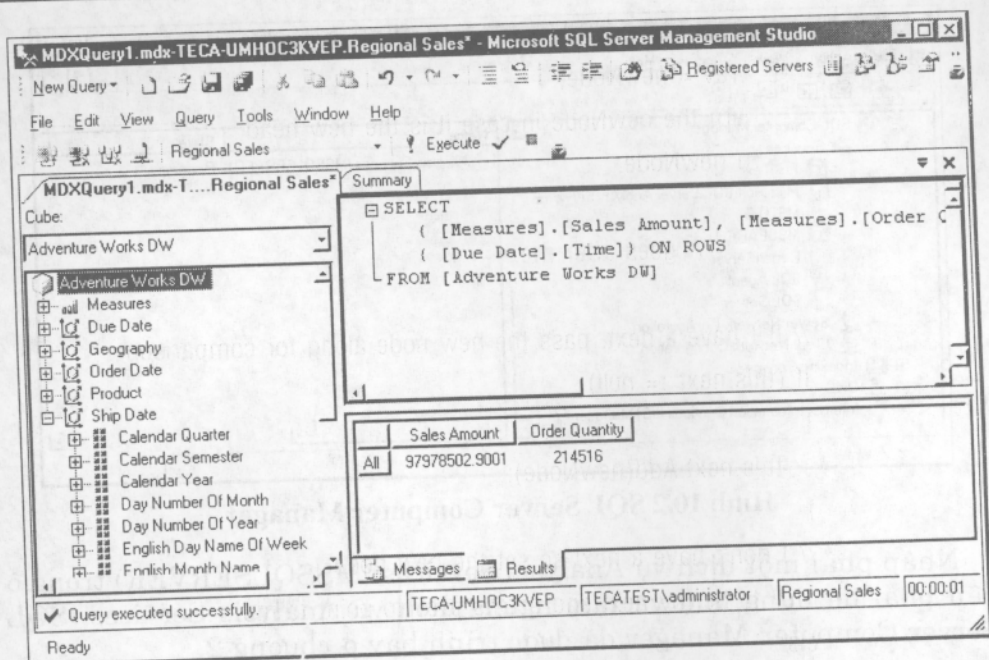
Nhấp phải một dịch vụ Analysis Services (MSSQLSERVER) trong ô bên phải để dừng, mở và tạm ngưng dịch vụ Analysis Services. SQL Server Computer Manager đã được trình bày ở chương 2.

SQL Server Management Studio

SQL Server Management Studio là công cụ quản lý chính cho cả SQL Server và Analysis Services. Bạn có thể sử dụng SQL Server Management Studio để thực hiện nhiều chức năng quản lý khác nhau. Nó có thể tạo ra các cơ sở dữ liệu Analysis Services, ngoài ra nó còn tạo script và triển khai các cơ sở dữ liệu Analysis Server. Bạn có thể dùng SQL Server Management Studio để xác lập các permission cho các đối tượng Analysis Services cũng như để giám sát sự truy cập end-user vào Analysis Services.

Multidimensional Expression (MDX) Query Editor

Ngoài các chức năng quản lý trên SQL Server Management Studio còn có multidimensional Expression (MDX) Editor, đây là bộ hiệu chỉnh mà bạn có thể dùng để viết và thực thi các mẫu truy vấn đặc biệt đồng thời tạo các đối tượng Analysis Services nhờ dùng các script. Hình 10.3 minh họa bộ hiệu chỉnh MDX mới này.



Hình 10.3 SQL Server Management Studio MDX Editor

Như bạn có thể thấy ở hình 10.3, bộ hiệu chỉnh MDX mới của SQL Server 2005 hỗ trợ đầy đủ các từ khóa được tạo mã màu và một cửa sổ kết xuất chứa các kết quả truy vấn MDX. Cũng có một trình duyệt metadata khối hợp nhất mà bạn có thể thấy ở phía bên trái của hình này. SQL Server Management Studio đã được trình bày ở chương 2.

Sự bảo mật

Sự bảo mật là một động lực thúc đẩy lớn cho Microsoft trong vài năm qua, và Analysis Services không ngoại lệ đối với sáng kiến này. Analysis Services dành cho SQL Server 2005 có hơn 100 cải tiến về sự bảo mật, một số cải tiến quan trọng nhất sẽ được trình bày trong mục dưới đây.

Sự bảo mật theo mặc định

Analysis Services được thiết kế để bảo mật theo mặc định. Trước tiên, dịch vụ được cài đặt để chạy với các đặc quyền tối thiểu, giảm sự bộc lộ của công ty nếu hệ thống bị điều đình. Kế tiếp, khi sản phẩm được cài đặt lần đầu tiên, mọi tùy chọn bảo mật (security) được mở theo mặc định. Tương tự, mọi tính năng vốn có thể làm cho hệ thống dễ bị tấn công với những mối đe dọa sẽ được tắt theo mặc định, bao gồm:

- ❖ Tắt sự truy cập HTTP

- ❖ Tắt các nối kết vô danh
- ❖ Tắt các thủ tục lưu trữ
- ❖ Tắt các mẫu truy vấn Openrowset

Nếu cần, tất cả những tính năng này có thể được mở lại dễ dàng bởi người quản trị hệ thống, nhưng chúng được tắt theo mặc định.

Sự mã hóa

Sự mã hóa (encryption) cũng được sử dụng để cải tiến sự bảo mật trong Analysis Services dành cho SQL Server 2005. Sự mã hóa bây giờ có trong nhiều vùng khác nhau. Trước tiên, kênh truyền thông giữa ứng dụng client và server có thể được mã hóa và server được cấu hình để truy cập chỉ các nối kết đã mã hóa. Các khối cục bộ có thể được mã hóa, ngoài ra các file dự phòng Analysis Services cũng được tùy ý mã hóa.

Các đặc quyền quản trị

Sự giới thiệu các đặc quyền quản trị là một tính năng khác liên quan đến sự bảo mật của Analysis Services trong SQL Server 2005. Với SQL Server 2000, bạn cần là một thành viên của nhóm OLAP Administrators để thực hiện các thay đổi cho cấu hình và cơ sở dữ liệu Analysis Services. Các đặc quyền quản trị mới trong SQL Server 2005 Analysis Services cho phép bạn tạo ra các hệ thống quản trị khác nhau cho mỗi cơ sở dữ liệu. Có các permission mới để đọc metadata cũng như một permission khác cho phép một account (tài khoản) xử lý các khối.

Các cải tiến về sự sao lưu dự phòng và phục hồi

Sự sao lưu dự phòng và phục hồi Analysis Services cũng đã được cải tiến trong SQL Server 2005. Các cải tiến mới về sự sao lưu dự phòng bao gồm loại bỏ giới hạn dự phòng 2GB, khả năng nén và mã hóa bản sao dự phòng (backup), và khả năng phục hồi dễ dàng một bản sao dự phòng trở lại một thể hiện khác của Analysis Services.

Loại bỏ giới hạn dự phòng 2GB

Các bản sao dự phòng SQL Server 2000 Analysis Services bị giới hạn trong các file dự phòng chưa tới 2GB. Giới hạn này đã được tăng lên trong SQL Server 2005, và bản sao dự phòng của Analysis Services hỗ trợ các file lên đến giới hạn của NTFS là 16TB.

Khả năng nén và mã hóa bản sao dự phòng

Bản sao dự phòng (backup) của Analysis Services 2005 bây giờ có thể vừa nén vừa mã hóa các file được dự phòng. Các tùy chọn này sẽ tăng thêm một ít thời gian cho cửa sổ được yêu cầu để thực hiện việc sao lưu

dự phòng, nhưng tạo ra một bản sao dự phòng nhỏ hơn và an toàn hơn. Bạn cũng có tùy chọn để lướt qua phần sao lưu dự phòng thông tin bảo mật.

Khả năng phục hồi dễ dàng bản sao dự phòng trở lại một thể hiện khác

SQL Server 2005 Analysis Services hỗ trợ khả năng lưu dự phòng các cơ sở dữ liệu Analysis Services cũng giống như bạn làm với các cơ sở dữ liệu SQL Server quan hệ và phục hồi chúng trở lại một thể hiện khác của Analysis Services. Điều này giúp cho việc di chuyển các cơ sở dữ liệu Analysis Services giữa các server trở nên dễ dàng hơn nhiều.

Các cải tiến về sự phát triển

Analysis Services đã trải qua nhiều cải tiến đáng kể về sự phát triển với ấn bản SQL Server 2005. Business Intelligence Development Studio đã thay thế Analysis Manager như là công cụ phát triển chính. Ngoài ra, Analysis Services bây giờ được truy cập nhờ dùng một giao thức hoàn toàn mới, XML for Analysis (XMLA) và có hai cơ cấu đối tượng .NET hoàn toàn mới: một dùng để quản lý, Analysis Management Objects (AMO), và một dành cho sự phát triển các ứng dụng, ADO MD.NET.

Business Intelligence Development Studio

Business Intelligence Development Studio mới là công cụ chính dùng để phát triển các ứng dụng Analysis Services. Với SQL Server 2005, có một sự phân biệt rõ ràng giữa Analysis Services Management và sự phát triển của các cơ sở dữ liệu Analysis Services. Trong khi SQL Server Management Studio được sử dụng để quản lý Analysis Services, Business Intelligence Development Studio được sử dụng để phát triển các giải pháp Analysis Services. Giống như SQL Server Management Studio, Business Intelligence Development Studio dựa trên vỏ Visual Studio. Business Intelligence Development Studio đã được trình bày chi tiết ở chương 2.

Các chế độ Online và Offline

Không giống như các phiên bản trước đây của Analysis Manager, luôn hoạt động trong một chế độ nối kết, Business Intelligence Development Studio mới hoạt động trong cả hai chế độ online và offline. Theo mặc định, Business Intelligence Development Studio hoạt động trong một chế độ offline (ngoại tuyến). Ở chế độ này, Business Intelligence Development Studio không được nối kết với server Analysis Services và mọi sự thay đổi cũng như các đối tượng mà bạn xác định sẽ nằm bên trong môi trường phát triển cho đến khi nào bạn chọn triển khai dự án.

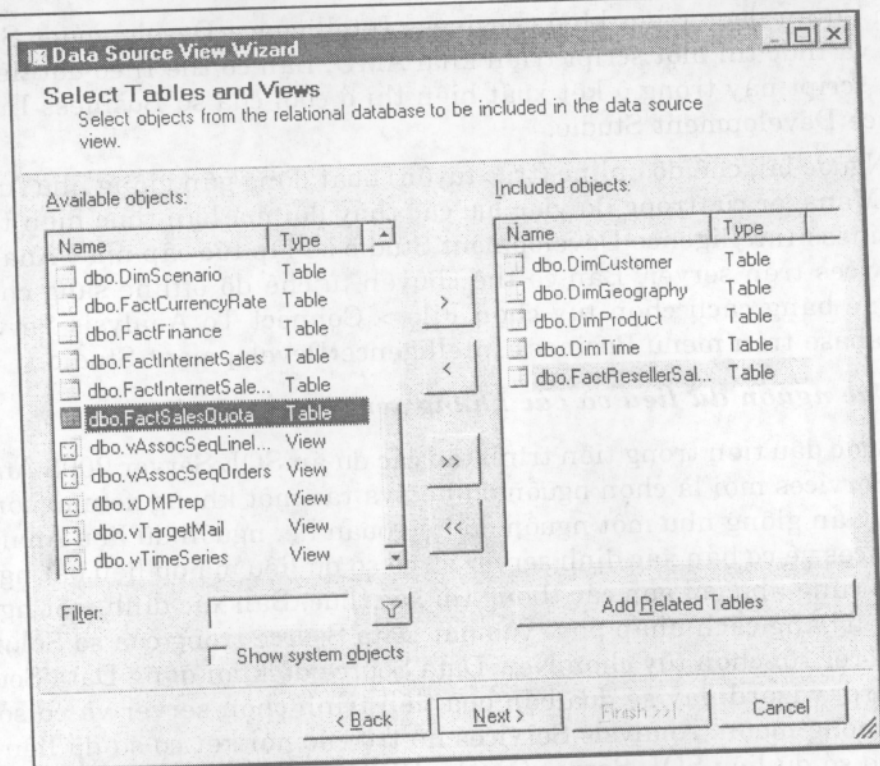
Khi dự án được triển khai, Business Intelligence Development Studio tạo và thực thi một script triển khai AMO. Bạn có thể theo dõi tiến độ của script này trong ô kết xuất hiển thị ở cuối cửa sổ Business Intelligence Development Studio.

Ngược lại, chế độ online (trực tuyến) hoạt động gần giống như Analysis Manager cũ, trong đó việc lưu các thay đổi mà bạn thực hiện trong Business Intelligence Development Studio sẽ lập tức cập nhật Analysis Services trên server. Bạn có thể chuyển từ chế độ offline sang chế độ online bằng cách chọn tùy chọn File > Connect To Analysis Services Database trên menu Business Intelligence Development Studio.

Các nguồn dữ liệu và các khung xem nguồn dữ liệu

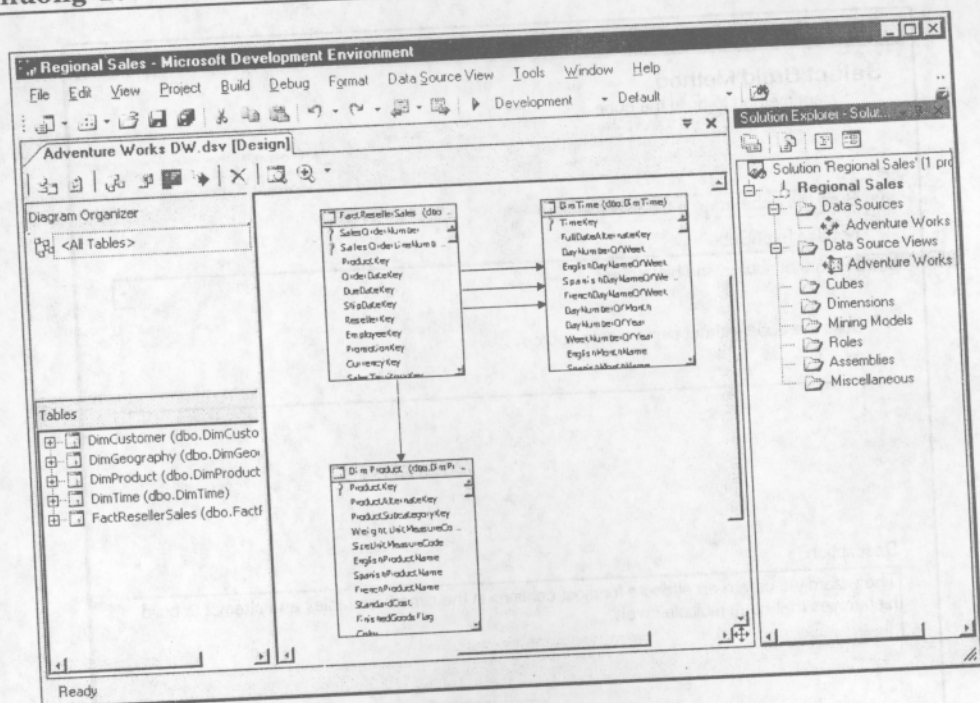
Bước đầu tiên trong tiến trình tạo các dự án SQL Server 2005 Analysis Services mới là chọn nguồn dữ liệu và tạo một khung xem nguồn dữ liệu. Gần giống như một nguồn dữ liệu quan hệ, nguồn dữ liệu Analysis Services về cơ bản xác định server và cơ sở dữ liệu mà dữ liệu bắt nguồn từ đó cũng như gói gọn các thông tin xác thực. Bạn xác định một nguồn dữ liệu bằng cách nhấp phải vào nút Data Source trong cửa sổ Solution Explorer rồi chọn tùy chọn New Data Source để khởi động Data Source Wizard, wizard này sẽ đưa bạn qua tiến trình chọn server và cơ sở dữ liệu mong muốn. Analysis Services hỗ trợ các nối kết cơ sở dữ liệu với các cơ sở dữ liệu SQL Server, Oracle, DB2, và Teradata.

Sau khi nguồn dữ liệu đã được tạo, bạn xác định một khung xem nguồn dữ liệu (data source view) để ấn định các table dữ kiện và kích thước mà bạn muốn dùng để tạo khối của bạn. Để tạo một khung xem nguồn dữ liệu mới, bạn nhấp phải vào nút Data Source View trong cửa sổ Solution Explorer rồi chọn tùy chọn New Data Source View. Thao tác này sẽ khởi động Data Source View Wizard. Bước đầu tiên trong wizard này cho phép bạn chọn nguồn dữ liệu thích hợp. Sau khi chọn nguồn dữ liệu, bạn chọn các table dữ kiện và kích thước nhờ dùng hộp thoại Select Tables And Views mà bạn thấy ở hình 10.4.



Hình 10.4 Tạo một khung xem nguồn dữ liệu

Hộp thoại Select Tables And Views của Data Source View Wizard cho phép bạn tùy ý lọc danh sách để hiển thị một tập hợp con của cơ sở dữ liệu có sẵn, điều này tiện lợi khi xử lý một số lượng lớn các table. Để chọn các table và các khung xem, bạn nhấp đúp vào chúng trong cột bên trái. Thao tác này sẽ tập hợp danh sách Included Objects mà bạn thấy ở phía bên phải của hình. Khi bạn đã làm việc xong với Data Source View Wizard, Data Source View Designer như minh họa ở hình 10.5 sẽ được khởi động tự động.

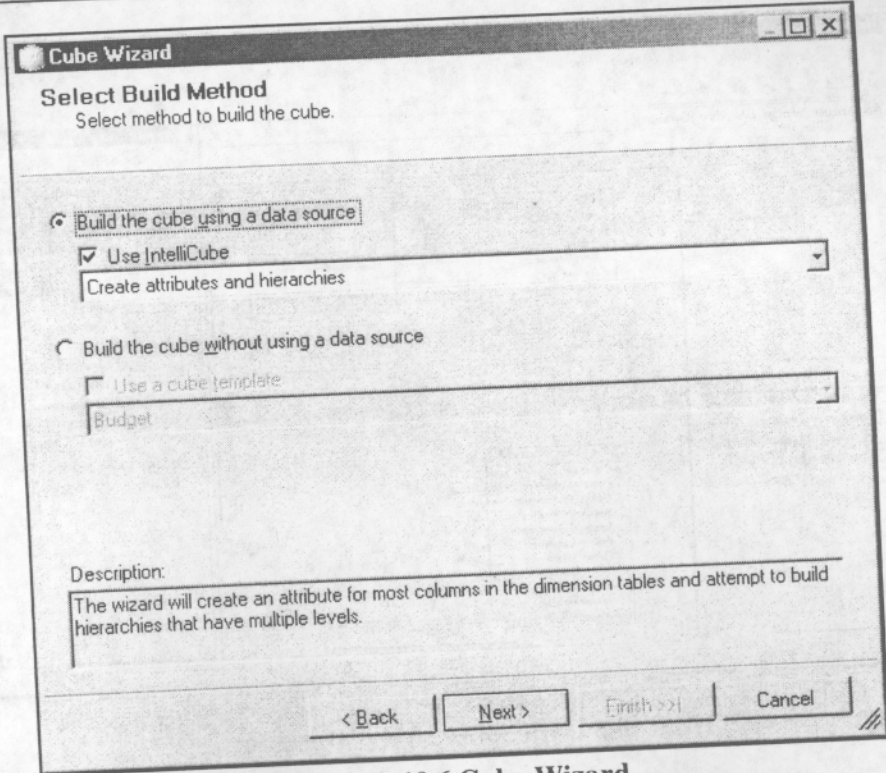


Hình 10.5 Data Source View Designer

Khung xem nguồn dữ liệu cơ bản là một sự trừu tượng của một hay nhiều nguồn dữ liệu nền tảng, và Data Source View Designer cho phép bạn tùy biến thông tin được đưa vào khung xem nguồn dữ liệu. Nhờ dùng Data Source View Designer, bạn có thể tùy biến nhiều khía cạnh khác nhau và điều khiển cách trình bày dữ liệu. Chẳng hạn, bạn có thể xác định và thay đổi các mối quan hệ cơ sở dữ liệu, đặt lại tên các table và tạo các cột đã được tính. Các thay đổi được thực hiện cho khung xem nguồn dữ liệu không được lan truyền trở lại các nguồn dữ liệu nền tảng. Chúng chỉ có tác dụng trong khung xem nguồn dữ liệu.

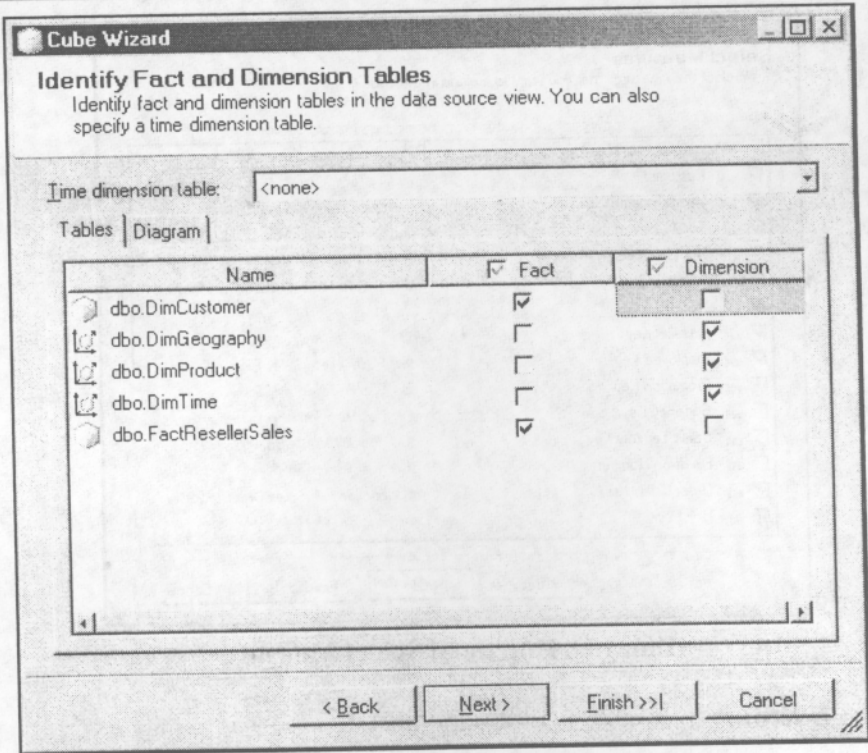
Cube Wizard

Để giúp bạn thiết kế các khối, Analysis Services có một Cube Wizard hoàn toàn mới. Sau khi, bạn đã tạo các nguồn dữ liệu và các khung xem nguồn dữ liệu, bạn chạy Cube Wizard bằng cách nhấp phải vào nút Cube trong cửa sổ Solution Explorer rồi chọn tùy chọn New Cube từ menu bật lên. Thao tác này sẽ khởi động Analysis Services 2005 Cube Wizard, như minh họa ở hình 10.6.



Hình 10.6 Cube Wizard

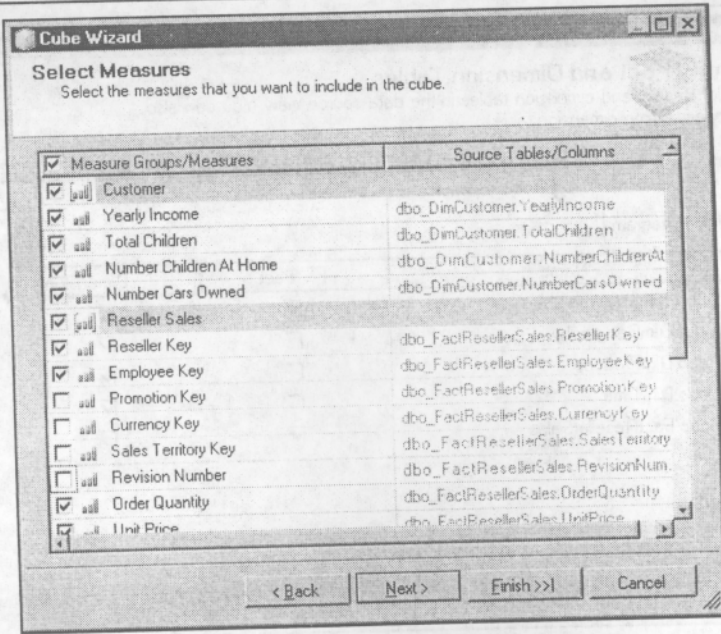
Cube Wizard trong SQL Server 2005 mạnh hơn rất nhiều so với Cube Wizard trong phiên bản Analysis Services của SQL Server 2000. SQL Server 2005 Cube Wizard cho phép bạn tạo một khối hoặc ở dạng từ dưới lên bằng cách chọn nguồn dữ liệu và khung xem nguồn dữ liệu mà bạn đã ấn định hoặc ở dạng từ trên xuống bằng cách trước tiên thiết kế khối và metadata của nó. Ở dạng từ dưới lên, tính năng Intelli Cube mới sẽ phân tích các table đã được chọn và tự động đề nghị các table dữ kiện cũng như kích thước phù hợp với các thuộc tính schema của chúng. Mặt khác, để tạo khối theo dạng từ trên xuống, bạn có thể chọn tùy chọn Build The Cube Without A Data Source. Trong tình huống này, bạn ấn định bằng tay mọi thuộc tính của khối. Hình 10.7 minh họa kết quả chọn table của IntelliCube.



Hình 10.7 Nhận biết các table dữ kiện và kích thước

IntelliCube thực hiện tốt việc chọn tự động các table dữ kiện và kích thước thích hợp, nhưng nó không hoàn hảo và bạn có thể tự do thay đổi các loại table mà IntelliCube tạo ra. Sau khi chọn các table mong muốn, Cube Wizard hướng dẫn bạn qua tiến trình chọn các số đo cho khối. Về cơ bản, IntelliCube chọn tất cả các cột số làm các số đo có thể có. Sau đó, bạn có thể chọn các cột mà bạn muốn. Hình 10.8 minh họa hộp thoại Select Measure của Cube Wizard.

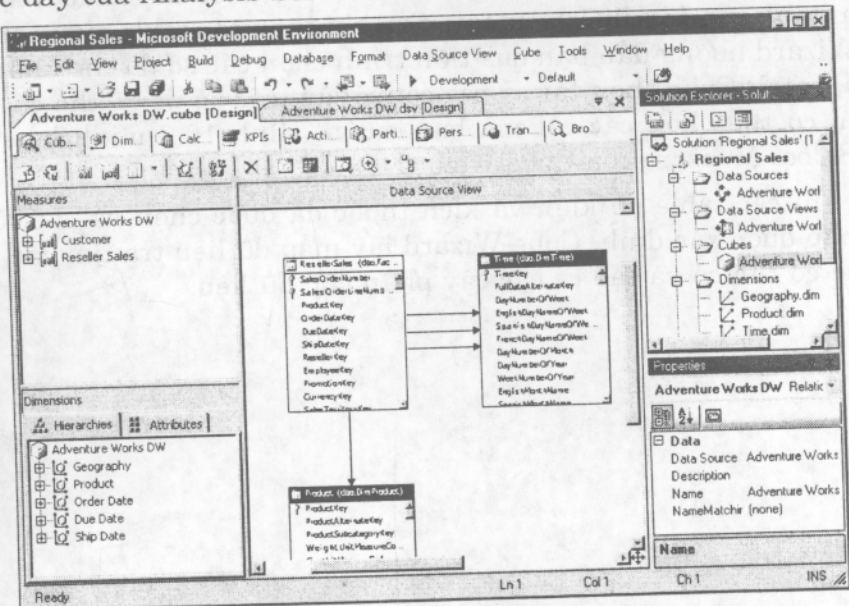
Sau khi các table dữ kiện và kích thước đã được chọn và các số đo thích hợp được xác định, Cube Wizard lấy mẫu dữ liệu tra tìm các mối quan hệ có thể có và tạo ra các cây phân cấp dữ liệu.



Hình 10.8 Hộp thoại Select Measure

Cube Editor

Ngay khi Cube Wizard đã hoàn tất, Cube Editor được hiển thị tự động. Analysis Services 2005 Cube Editor được cải tiến đáng kể và cung cấp nhiều tính năng vượt xa các tính năng đã có trong các phiên bản trước đây của Analysis Services. Hình 10.9 minh họa Cube Editor.



Hình 10.9 Cube Editor

SQL Server 2005 Analysis Services Cube Editor cung cấp 9 tab riêng biệt, mỗi tab cho phép bạn làm việc với một khía cạnh khác của khối. 9 khung xem khối mà Cube Editor cung cấp là

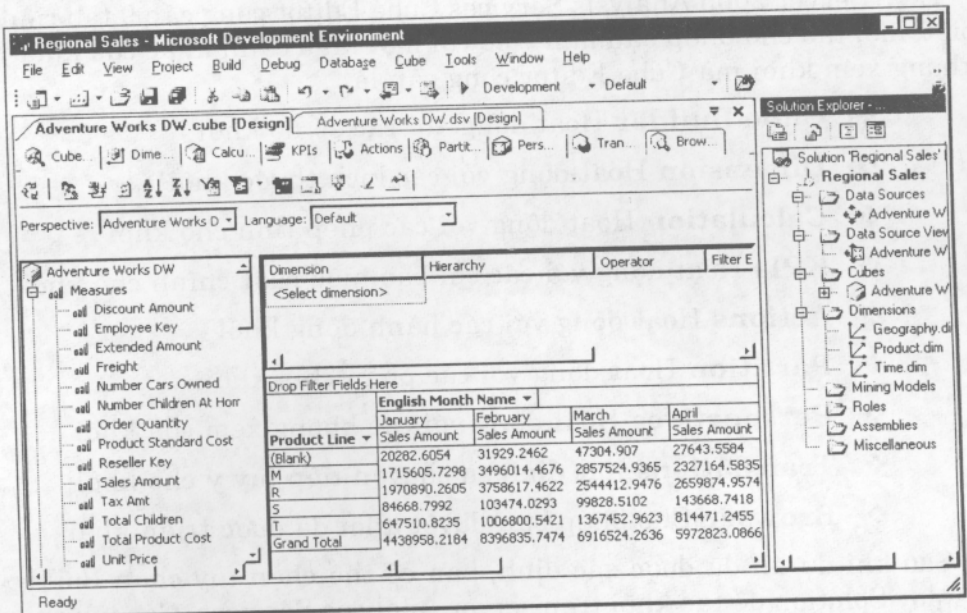
- ❖ **Cube Builder** Hoạt động với các số đo khối
- ❖ **Dimension** Hoạt động với các kích thước khối
- ❖ **Calculation** Hoạt động với các phép tính cho khối
- ❖ **KPIs** Hoạt động với các chỉ báo hiệu suất chính cho khối
- ❖ **Actions** Hoạt động với các hành động khối
- ❖ **Partition** Hoạt động với các phần chia khối
- ❖ **Perspectives** Hoạt động với các khung xem của khối
- ❖ **Translations** Xác định sự chuyển tiếp tùy ý cho khối
- ❖ **Browser** Cho phép bạn duyệt khối đã được triển khai

Sau khi dự án đã được xác định, bạn có thể chọn tùy chọn Build > Deploy Solution để tạo khối trên server Analysis Services. Các tùy chọn dự án điều khiển việc khối sẽ được xử lý hay không sau khi nó được triển khai ra server. Theo mặc định, khối sẽ được xử lý khi nó được triển khai lần đầu tiên.

Cube Browser

Ngay sau khi khối đã được triển khai và xử lý, bạn có thể xem và định hướng qua các kích thước cũng như các số đo của khối nhờ dùng trình duyệt cài sẵn của Cube Editor. Hình 10.10 minh họa một ví dụ của Cube Browser.

Cube Browser được tạo nhờ dùng OWC (Office Web Component). Bạn dùng Cube Browser bằng cách rê và thả các kích thước từ các thuộc tính của khối nằm ở phía bên trái của màn hình sang các trục biểu thị hàng và cột của OWC nằm ở giữa màn hình. Sau đó, bạn chọn các số đo mong muốn rồi thả chúng vào trường dữ liệu. Cube Browser tự động truy xuất dữ liệu và hiển thị nó trong cửa sổ trình duyệt.

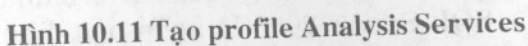


Hình 10.10 Cube Browser

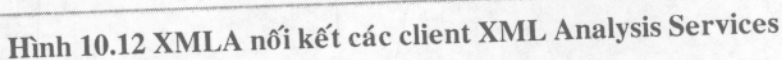
Profiler

Với SQL Server 2005, Analysis Services không còn là một server hộp đen nữa. SQL Server 2000 Analysis Services và SQL Server 7 OLAP Services không thật sự cung cấp bất kỳ cách nào cho người quản trị để xem server đang thực hiện điều gì. Với SQL Server 2005, Profiler có khả năng truy vết mọi chức năng khác nhau đang chạy trên Analysis Services cũng như nó đã làm với cơ sở dữ liệu SQL Server quan hệ. Bạn khởi động Profiler bằng cách chọn tùy chọn Profiler từ menu Start > All Programs > Microsoft SQL Server 2005 của Windows. Hình 10.11 minh họa Profiler đang chạy trên Analysis Services.

Profiler cũng là một công cụ tuyệt vời để tìm hiểu thêm về MDX. Ngoài ra, Profiler còn là một công cụ xử lý sự cố mạnh mẽ để theo dõi những hoạt động của server. Bạn có thể dùng Analysis Services Profiler để bắt và mở lại các event trên server.



XML for Analysis (XMLA) là một giao thức độc lập nền dựa trên các dịch vụ web và SOAP. Microsoft SQL Server 2005 Analysis Services sử dụng XMLA để xử lý tất cả các cuộc giao tiếp ứng dụng client với Analysis Services. Nó có cả hai công cụ quản lý và phát triển của Microsoft: SQL Server Management Studio và Business Intelligence Development Studio. XMLA được tối ưu hóa cho Internet và được thiết kế để giảm bớt các tuyến đi vòng đến server. Ở hình 10.12, bạn có thể thấy cách client OLAP và các ứng dụng quản lý sử dụng XMLA để nối kết với nền Analysis Services.



XMLA hỗ trợ hai loại chức năng cơ bản: thực thi các yêu cầu và phát hiện các yêu cầu.

Thực thi các yêu cầu

Thực thi các yêu cầu (execute requests) là thực hiện một hành động nào đó; chúng thay đổi trạng thái của các đối tượng trên server. Bạn có thể dùng các yêu cầu thực thi để tạo, thay đổi và xóa các đối tượng cũng như xử lý các khối.

Phát hiện các yêu cầu

Phát hiện các yêu cầu (discover requests) dùng để truy xuất thông tin về các đối tượng trên server. Bằng cách dùng các yêu cầu phát hiện, bạn có thể truy xuất các định nghĩa về phân chia, khối và sự bảo mật. Bạn cũng có thể truy vấn trạng thái hệ thống của server để tìm hiểu thêm về số lượng nối kết đang được sử dụng cũng như việc sử dụng nguồn tài nguyên của server.

Các cải tiến về ODL

Một trong những cải tiến quản lý quan trọng nhất trong Analysis Services dành cho SQL Server 2005 là ngôn ngữ định nghĩa đối tượng (Object Definition Language, gọi tắt là ODL). ODL mang cùng loại tính năng tạo đối tượng đến cho Analysis Services mà SQL Server quan hệ luôn có. Analysis Services ODL cho phép bạn viết các script vốn có thể tự động tạo tất cả các đối tượng cơ sở dữ liệu Analysis Services của bạn. Bạn cũng có thể dùng các script này để điều khiển việc tạo phiên bản cơ sở dữ liệu Analysis Services.

Analysis Services DDL mới của SQL Server 2005 là một chi tiết mở được tạo trên XML for Analysis (XMLA). Dựa vào một chuẩn mở rộng, các script Analysis Services ODL có thể được tạo nhờ dùng bất kỳ bộ hiệu chỉnh nhận biết XML.

Các cải tiến về MDX

MDX vẫn còn là ngôn ngữ truy vấn cốt lõi cho các cơ sở dữ liệu Analysis Services, và trong SQL Server 2005 MDX có nhiều cải tiến quan trọng.

Các script MDX

Một trong những thay đổi lớn nhất cho MDX trong SQL Server 2005 là khả năng nhóm nhiều câu lệnh MDX lại với nhau thành một script. Việc tạo script cho phép nhiều câu lệnh MDX được thực thi theo trình tự. Mỗi câu lệnh MDX trong script được tách biệt với câu lệnh kế tiếp bởi một dấu chấm phẩy.

Cú pháp được đơn giản

MDX bây giờ có cung cấp một cú pháp ngắn gọn cho các thành phần đã được tính. Cú pháp ngắn gọn mới này lấy các kích thước để nhận biết các thành phần, làm cho việc chỉ định rõ các kích thước lặp lại trong các câu lệnh MDX của bạn trở nên không cần thiết.

Sự chuyển đổi kiểu tự động

Sự chuyển đổi kiểu tự động giúp chuyển đổi tự động từ một thành phần sang một tập hợp và ngược lại. Điều này loại bỏ nhu cầu viết các dấu ngoặc khi chỉ định các tập hợp, làm cho MDX trở nên đơn giản hơn và cũng dễ đọc hơn.

Xử lý các thành phần còn thiếu

Một số cải tiến khác trong SQL Server 2005 MDX là khả năng xử lý các thành phần còn thiếu. Với SQL Server 2000, khi bạn xác định một report và một hay nhiều thành phần mà report đã sử dụng không còn tồn tại nữa, report này sẽ không thành công. Có nhiều yếu tố khác có thể gây ra loại tình trạng này, bao gồm các thay đổi trực tiếp cho cấu trúc khối nền tảng cũng như các thay đổi dần dần cho các kích thước. Trong MDX mới của SQL Server 2005, bạn có thể dùng thuộc tính kích thước MDXMissingMemberMode để cho phép một report tiếp tục thực hiện chức năng ngay cả khi các thành phần của một kích thước bị thiếu.

Gộp các thành phần đếm phân biệt

Nhờ được thêm một tính năng khác vốn không có trong Analysis Services của SQL Server 2000, MDX trong SQL Server 2005 bây giờ có thể gộp các nội dung của các thành phần đếm phân biệt.

Các tập hợp trong mệnh đề WHERE

Việc sử dụng các tập hợp (set) trong mệnh đề MDX WHERE là một cải tiến quan trọng khác cho Analysis Services trong SQL Server 2005. Về cơ bản, việc dùng các tập hợp trong mệnh đề MDX WHERE giống như dùng từ khóa OR trong mệnh đề SQL WHERE. Các tập hợp cho phép mẫu truy vấn trả về các kết quả từ tất cả những thành phần có trong mệnh đề WHERE.

ADOMD.NET

ADOMD.NET là một data provider.NET gốc hoàn toàn mới được thiết kế để truy cập các nguồn dữ liệu đa kích thước. ADOMD.NET được thiết kế để thay thế thư viện đối tượng truy cập dữ liệu đa kích thước ADO MD cũ hơn dựa trên COM. Các ứng dụng client được tạo nhờ dùng bất kỳ ngôn ngữ .NET như Visual Basic, C#, Managed C++, hay J# có

thể sử dụng ADOMD.NET để truy xuất dữ liệu và thông tin metadata từ Analysis Services của SQL Server 2005. Bên trong, ADOMD.NET sử dụng giao thức XMLA để nối kết với server Analysis Services. Các ứng dụng ADOMD.NET có thể nối kết với Analysis Server theo hai cách: XMLA qua HTTP hay XMLA qua TCP/IP.

Analysis Services Management Objects (AMO)

Analysis Services Management Objects (AMO) là một cơ cấu đối tượng hoàn toàn mới khác mà Microsoft đã giới thiệu với SQL Server 2005. AMO là hậu duệ của các đối tượng hỗ trợ quyết định (Decision Support Objects, gọi tắt là DSO) cũ hơn dựa trên COM vốn được cung cấp với SQL Server 2000. DSO bây giờ không còn nữa. Một phiên bản của DSO được đưa vào SQL Server 2005 nhưng nó chủ yếu có sẵn để bảo đảm tính tương thích ngược.

AMO được tạo nhờ dùng .NET framework và được thiết kế độc nhất để quản lý Analysis Services. AMO hoạt động ở một cấp độ cao hơn XMLA và giống như các cơ cấu đối tượng Analysis Services mới khác, AMO dùng XMLA để giao tiếp với server Analysis Services. Cả SQL Server Management Studio và Business Intelligence Development Studio đều sử dụng AMO.

AMO hỗ trợ các nối kết quản lý an toàn với Analysis Services. Nó hỗ trợ sự xác thực Windows cũng như cung cấp một kênh giao tiếp được mã hóa giữa ứng dụng client và server. AMO còn tối ưu hóa nối kết với server bằng cách nén XML được gửi giữa client và server Analysis Services.

AMO có nhiều ưu điểm hơn mô hình DSO cũ. Trước tiên, AMO liệt kê các đối tượng một cách thông minh, cho nó hiệu suất tốt hơn khi liệt kê một số lượng lớn các hạng mục. AMO còn cung cấp khả năng lưu dự phòng và phục hồi hệ thống. Ngoài ra, AMO làm cho sự phân tích tác động trở nên có sẵn với các ứng dụng của bạn. Sự phân tích tác động (impact analysis) cho phép ứng dụng của bạn xác định những đối tượng Analysis Services nào sẽ chịu ảnh hưởng bởi một hành động cho trước. Chẳng hạn, nếu một câu lệnh thay đổi các kích thước của khối, nó có thể yêu cầu khối phải được xử lý lại. Sự phân tích tác động cho bạn thấy hiệu ứng của các câu lệnh.

Khai thác dữ liệu

Khai thác dữ liệu (data mining) cho phép các tổ chức hay các công ty tận dụng thông tin được lấy từ dòng kinh doanh và dùng thông tin đó để tiên đoán các xu hướng kinh doanh trong tương lai. Các tiên đoán này có thể giúp một doanh nghiệp đưa ra các quyết định tốt hơn về hướng phát triển trong tương lai cũng như cách sử dụng tốt nhất các

Chương 10 - Các dịch vụ phân tích

nguồn tài nguyên của nó. SQL Server 2000 đã cung cấp hai thuật toán khai thác dữ liệu cơ bản: các cây quyết định (decision tree) và sự nhóm cụm (clustering). SQL Server 2005 bổ sung thêm nhiều thuật toán khai thác dữ liệu mới. Các thuật toán khai thác dữ liệu được đưa vào SQL Server 2005 bao gồm: Decision Trees, Time Series, Sequence Clustering, Naive Bayes, và Association Rules.

Decision Trees

Thuật toán Microsoft Decision Trees (DT) được thiết kế chủ yếu để tiên đoán. Thuật toán này dùng để tiên đoán các biến liên tục và rời rạc.

Time Series

Thuật toán Time Series mới giới thiệu khái niệm của quá khứ, hiện tại, và tương lai vào hình thức tiên đoán. Được thiết kế để tiên đoán các bước kế tiếp theo trình tự số, thuật toán Time Series không chỉ chọn các tiên đoán tốt nhất cho một đích đã định mà còn nhận biết các khoảng thời gian tốt nhất mà bạn mong đợi hiệu ứng của mỗi tiên đoán được chú ý.

Clustering và Sequence Clustering

Thuật toán Clustering được thiết kế để tìm một số đếm cluster phù hợp cho mô hình của bạn với các thuộc tính của dữ liệu huấn luyện. Sequence clustering cho phép bạn tìm các cluster trong các trình tự dữ liệu. Nói cách khác, nó là clustering nhảy thứ tự.

Naive Bayes

Thuật toán Naive Bayes mới là một thuật toán tiên đoán. Được thiết kế để đạt được sự thực thi nhanh nhất, thuật toán này tiên đoán các mối quan hệ giữa các loại hạng mục.

Association Rules

Được thiết kế để phân tích dữ liệu giao tác, thuật toán Association Rules được sử dụng để tìm các nhóm hạng mục tồn tại trong một giao tác đơn.

PHỤ LỤC

Cài đặt và nâng cấp

Trong phụ lục này, bạn sẽ xem tổng quan về các tùy chọn cài đặt và nâng cấp SQL Server 2005.

Các ấn bản SQL Server 2005

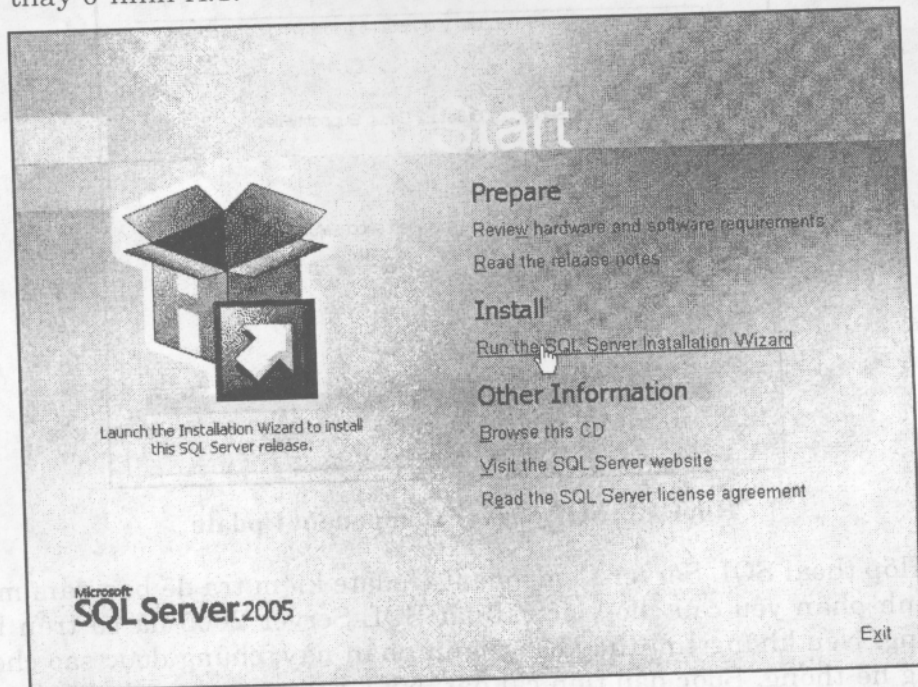
Giống như các ấn bản trước của SQL Server, Microsoft đưa ra nhiều ấn bản khác nhau của SQL Server 2005. Microsoft sẽ cung cấp các phiên bản SQL Server khác nhau sau đây:

- ❖ **SQL Server 2005 Enterprise Edition** Enterprise Edition cung cấp đầy đủ mọi tính năng có sẵn trong SQL Server 2005.
- ❖ **SQL Server 2005 Standard Edition** Standar cung cấp cho SQL Server các tính năng cốt lõi nhưng không cung cấp mọi tính năng hướng doanh nghiệp vốn hỗ trợ các cơ sở dữ liệu rất lớn và khả năng mở rộng cấp cao.
- ❖ **SQL Server 2005 Developer's Edition** Developer Edition cung cấp các tính năng giống như Enterprise Edition. Tuy nhiên, nó không được đăng ký cho mục đích sản xuất.
- ❖ **SQL Server 2005 Express** SQL Server 2005 Express là sự thay thế cho MSDE. Giống như MSDE, nó có thể được chạy hoàn toàn miễn phí. Không giống như MSDE, nó không có người điều khiển tải công việc.

Cài đặt SQL Server 2005

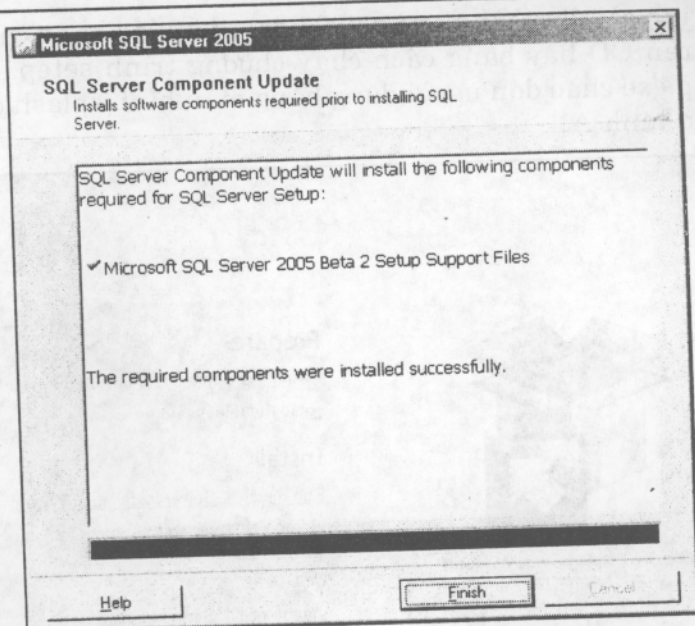
Sự cài đặt SQL Server 2005 khác với tiến trình cài đặt cho bất kỳ phiên bản trước của SQL Server và giải quyết nhiều vấn đề từng tồn tại trong phần cài đặt của các phiên bản trước đây. Không giống như các chương trình cài đặt SQL Server trước đây, vốn dựa vào .EXE. Chương trình cài đặt dành cho SQL Server 2005 được chuẩn hóa trên Microsoft Installer (MSI) 3.0. Việc dựa vào MSI làm cho SQL Server dễ sử dụng hơn với các phần cài đặt không chủ đích cũng như cho sự phân bố phần mềm nhờ dùng Microsoft Systems Management Server (SMS). Chương trình cài đặt mới cũng cung cấp sự cài đặt một bước cho các failover cluster.

Tiến trình cài đặt SQL Server 2005 được bắt đầu bằng cách dùng file Autorun trên CD hay bằng cách chạy chương trình setup.exe. Chương trình setup.exe chào đón người dùng bằng màn hình splash đầu tiên mà bạn thấy ở hình A.1.



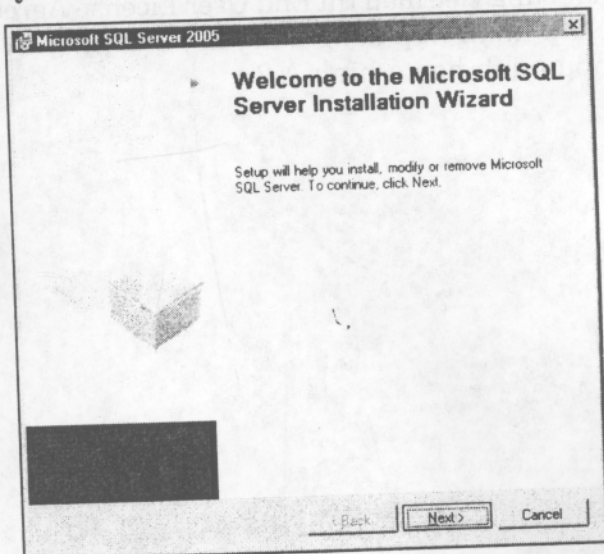
Hình A.1: Màn hình splash cài đặt SQL Server 2005

Nhấp liên kết Run The SQL Server Installation Wizard để bắt đầu tiến trình cài đặt bằng việc hiển thị End User License Agreement (EULA). Sau khi bạn chấp nhận hợp đồng đăng ký, bạn sẽ thấy màn hình cài đặt đầu tiên như minh họa ở hình A.2.



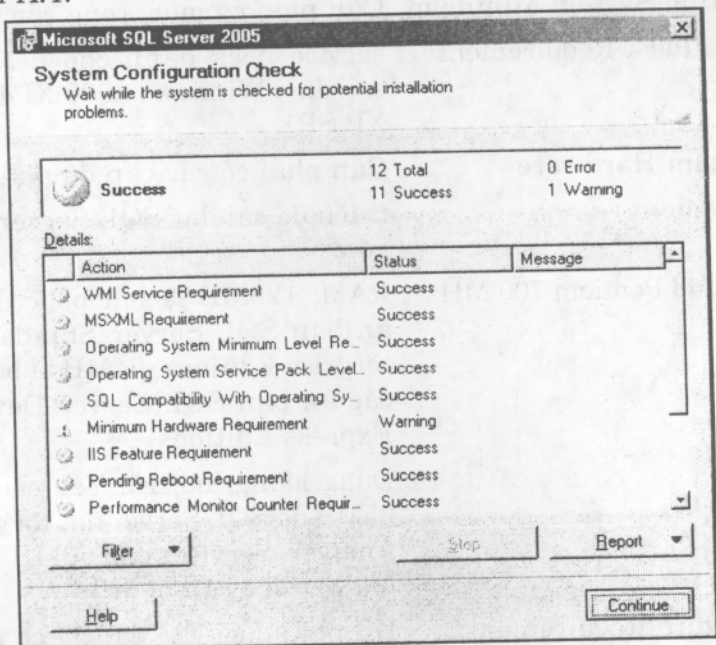
Hình A.2: SQL Server Component Update

Hộp thoại SQL Server Component Update kiểm tra để bảo đảm mọi thành phần yêu cầu cho việc cài đặt SQL Server 2005 đã có trên hệ thống. Nếu không tìm thấy các thành phần này, chúng được sao chép sang hệ thống. Bước đầu tiên cài đặt .NET Framework mới nhất cũng như sao chép các file setup sang hệ thống cục bộ. Sau khi đã có mọi thành phần cài đặt yêu cầu, nhấn Finish để hiển thị màn hình Welcome To The SQL Server Installation Wizard như minh họa ở hình A.3.



Hình A.3: Màn hình Welcome to the SQL Server Installation Wizard

Hộp thoại Welcome về cơ bản có tác dụng thông báo cho bạn rằng bạn sắp bắt đầu cài đặt sản phẩm SQL Server. Nhấp hộp thoại Welcome trước để hiển thị hộp thoại System Configuration Check như minh họa ở hình A.4.



Hình A.4: Hộp thoại System Configuration Check

Hộp thoại System Configuration Check lướt qua hệ thống để bảo đảm nó đáp ứng mọi yêu cầu cài đặt SQL Server 2005. Chương trình cài đặt thực hiện một loạt tiến trình kiểm tra hệ thống nhằm bảo đảm hệ thống có khả năng chạy SQL Server 2005. Khi mỗi hạng mục được kiểm tra, chương trình cài đặt tạo cờ cho nó bằng một dấu kiểm màu xanh lục biểu thị sự thành công, một dấu cảnh báo màu vàng cho biết có thể có một sự cố, hay một chữ X màu đỏ biểu thị có một lỗi trầm trọng. Sự cài đặt có thể được tiến hành với các cảnh báo, nhưng nếu có một chữ X màu đỏ, bạn phải xử lý lỗi trước khi tiếp tục cài đặt. Chọn hạng mục được tạo cờ và nhấp Report để xem thông tin về mỗi hạng mục. Bảng A.1 liệt kê các phần kiểm tra hệ thống phải được thông qua trước khi SQL Server 2005 có thể được cài đặt.

Bảng A.1: Các phần kiểm tra hệ thống

Yêu cầu

WMI Service Requirement

MSXML Requirement

Mô tả

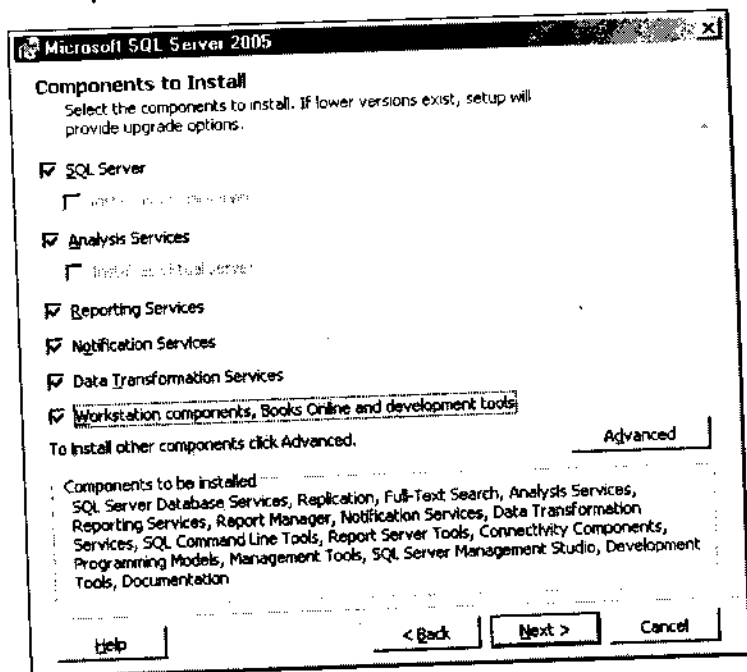
Phải cài đặt WMI Service.

Phải cài đặt phần cập nhật .NET Framework Quick Fix Engineering (QFE) .

Operating System	Cần phải có một trong các hệ điều hành
Minimum Level Requirement	sau: Windows 2000, Windows XP, Windows Server 2003.
Operating System Minimum Service Pack Requirement	Cần phải có một trong các cấp độ service pack dành cho hệ điều hành sau đây: Windows 2000 XP4, Windows XP SP1.
Minumum Hardware Requirement	Cần phải có các cấp độ phần cứng tối thiểu để cài đặt SQL Server 2005, bao gồm:
CPU: Intel Pentium 700 MHz	RAM: 128MB là bắt buộc đối với các ấn bản SQL Server Standard và Enterprise Editions; 64MB là bắt buộc với các ấn bản SQL Server Developer và Express Editions.
	Dung lượng lưu trữ: Yêu cầu tối thiểu 95MB để SQL Server đạt tối đa 300MB, Analysis Services cần 50MB và Reporting Service cần 50MB.
IIS Feature Requirement	IIS phải được cài đặt để chạy Reporting Services và các ứng dụng SQLXML. Nếu IIS không được cài đặt, sự cài đặt có thể được tiến hành nhưng ứng dụng Reporting Services sẽ không có sẵn cho việc cài đặt
Pending Reboot Requirement	Các bước cài đặt trước có thể yêu cầu khởi động lại hệ thống. Nếu vậy, phải thực hiện khởi động lại hệ thống.
Performance Monitor Counter Requirement	Các khóa đăng ký bộ đếm thực thi phải có khả năng gia tăng chính xác.
Default Installation Path	Người dùng đang chạy phần cài đặt
Permission Requirement	phải có các đặc quyền quản lý.
Internet Explorer Requirement	Yêu cầu phải có Internet Explorer 6.0 SP1 để chạy Business Intelligence Development Studio.
Manifest Requirement	Setup cấu hình các file sau đây để sử dụng MDAC 9.0: mmc.exe, wmioprse.exe, msftedl.exe, inetinfo.exe,

dllhost.exe, w3wp.exe và
aspnet_wp.exe.

Nếu mọi yêu cầu hệ thống được đáp ứng thì nút Continue sẽ được kích hoạt. Việc nhấn Continue sẽ hiển thị hộp thoại Registration Information, vốn yêu cầu bạn nhập tên của bạn, tên công ty, và khóa sản phẩm gồm 25 chữ số. Sau khi bạn điền vào tất cả những thông tin này và nhấn Next, chương trình cài đặt sẽ hiển thị hộp thoại Components To Install mà bạn có thể thấy ở hình A.5.



Hình A.5: Components To Install

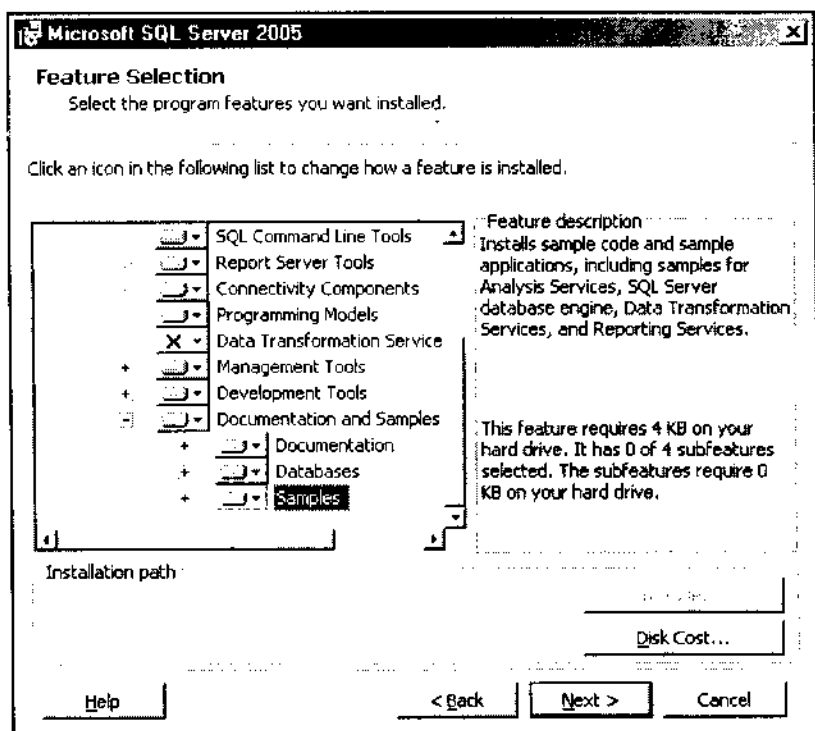
Hộp thoại Components To Install cho phép bạn chọn những thành phần SQL Server 2005 nào bạn muốn cài đặt. Nếu IIS hiện không có trên hệ thống thì tùy chọn Reporting Services sẽ không có sẵn. Nếu bạn đang cài đặt SQL Server 2005 trên một cluster thì tùy chọn Install As Virtual Server sẽ được kích hoạt.

Lưu ý

Để cài đặt một hệ thống quản lý, bạn chỉ cần chọn tùy chọn Workstation components.

Trong khi cấp độ chọn thành phần này là thích hợp cho hầu hết người dùng, còn có một hộp thoại chọn tính năng cho phép bạn chọn chính xác những thành phần nào sẽ được cài đặt. Bạn có thể hiển thị

hộp thoại Feature Selection như minh họa ở hình A.6 bằng cách nhấp Advanced.



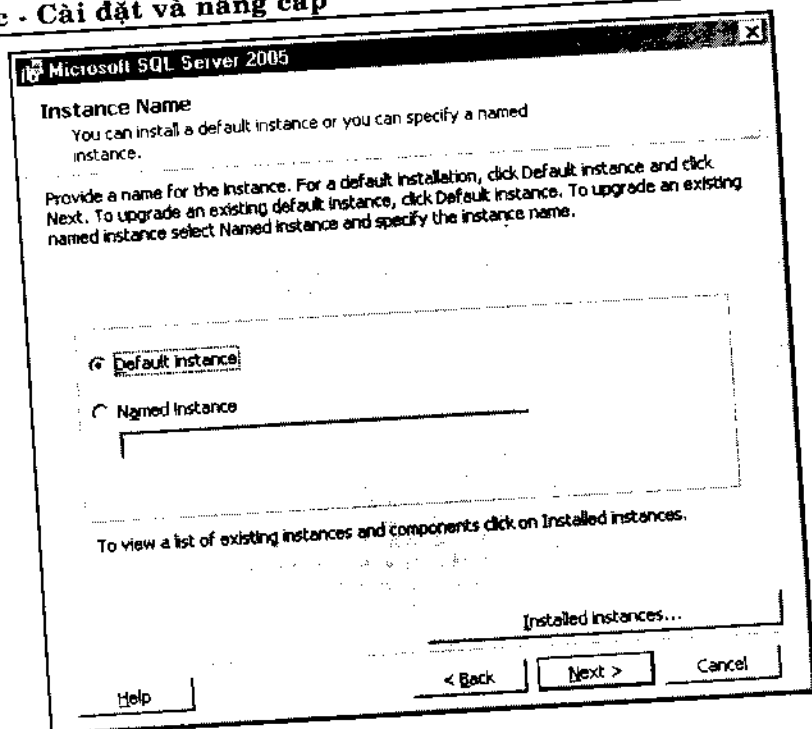
Hình A.6: Hộp thoại Feature Selection

Hộp thoại Feature Selection cho phép bạn khai thác mỗi thành phần mà tùy chọn cài đặt, không kể các tính năng.

Lưu ý

Theo mặc định, không có cơ sở dữ liệu hay ứng dụng mẫu nào được cài đặt. Nếu bạn muốn cài đặt cơ sở dữ liệu AdventureWorks mẫu và/hoặc các mẫu lập trình, bạn phải dùng hộp thoại Feature Selection để chọn tùy chọn Databases And Samples.

Sau khi bạn đã chọn những thành phần mà bạn muốn cài đặt, nhấp Next để hiển thị hộp thoại Instance Name như bạn thấy ở hình A.7.



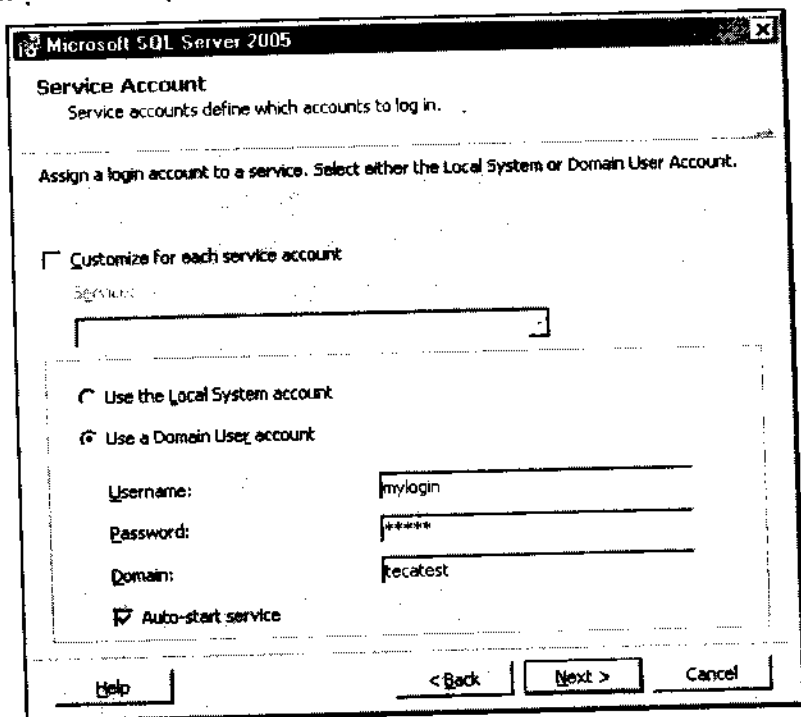
Hình A.7: Instance Name

Nếu đây là lần đầu tiên bạn chạy qua chương trình cài đặt, bạn sẽ có tùy chọn để cài đặt instance mặc định hoặc cài đặt một instance đã được đặt tên. Bạn đã cài đặt một instance SQL Server mặc định, chương trình setup sẽ phát hiện ra instance đó và cung cấp cho bạn chỉ tùy chọn để cài đặt một instance đã được đặt tên. Bạn có thể có đến 50 instance đã đặt tên trên một hệ thống. Đa số các phần thực thi sẽ sử dụng chỉ một instance duy nhất, nhưng sự cài đặt cho các ISP và ASP thường sẽ yêu cầu nhiều hơn. Nếu bạn tạo một instance đã đặt tên, mỗi tên phải dài từ 16 ký tự trở xuống. Các tên instance không nhạy kiểu chữ, nhưng ký tự đầu tiên trong tên phải là một mẫu tự. Chúng không thể có bất kỳ khoảng cách được nhúng nào và không được chứa dấu xuyệt ngược (\), dấu phẩy (,), dấu hai chấm (:), dấu trích dẫn đơn ('), dấu gạch nối (-), dấu và (&), dấu thăng (#), hay dấu a cong (@). Các tên instance cũng không được chứa từ dành riêng như "Default" hay "MSSQLServer".

Lưu ý

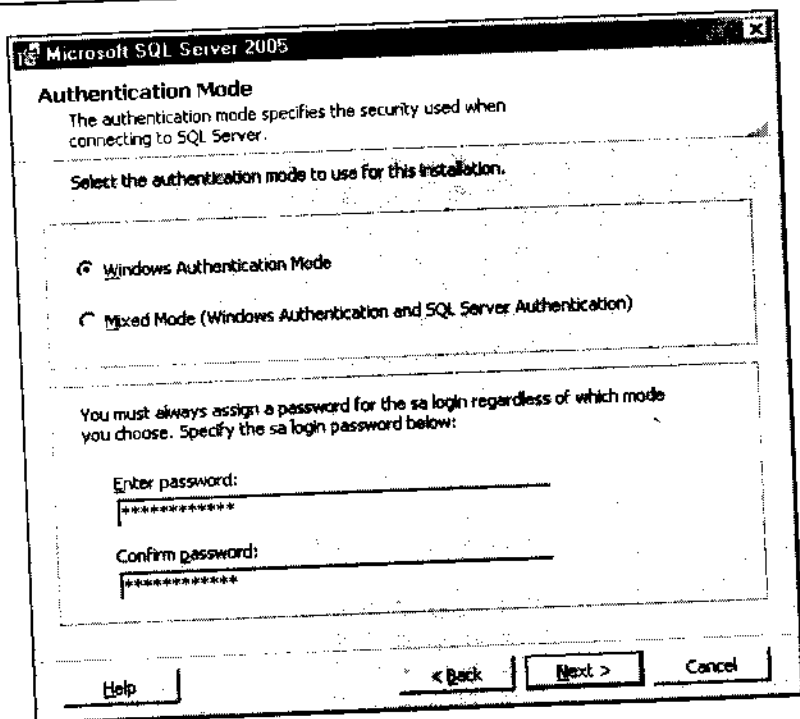
Nếu bạn tạo instance đã đặt tên, tên dịch vụ SQL Server sẽ được đặt tên như sau: **MSSQL\$InstanceName** (trong đó InstanceName được thay bằng tên instance mà bạn tạo).

Sau khi bạn chấp nhận instance mặc định hay tạo một instance đã đặt tên và nhấp Next, hộp thoại Service Account mà bạn thấy ở hình A.8 sẽ được hiển thị.



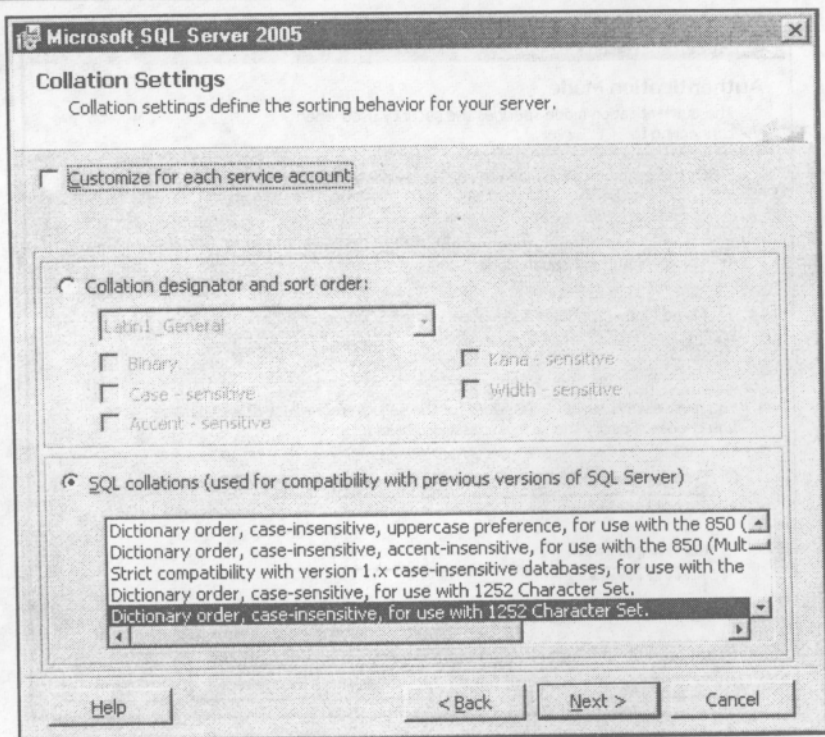
Hình A.8: Service Account

Các trị trong hộp thoại Service Account chỉ định các user account được sử dụng bởi dịch vụ SQL Server cũng như SQL Server Agent, Analysis Services, và Reporting Server. Đây là một lựa chọn quan trọng bởi vì nó kiểm soát các permission mà mỗi dịch vụ này được phép. Theo mặc định, hộp thoại Service Account hướng dẫn bạn chọn một Domain User account cho dịch vụ SQL Server. Bạn có thể dùng account quản lý, nhưng do mức đặc quyền cao của nó nên đây không phải là một ý tưởng hay. Bạn cũng có thể chọn account Local System; tuy nhiên, account này rất mạnh, có các permission giống như quản lý và chỉ giới hạn trong khả năng truy cập các tài nguyên mạng của nó. Thông thường, bạn sẽ muốn tạo một domain user account cụ thể cho SQL Server để chạy dưới sự kiểm soát của nó và chọn account này. Điều này giúp bạn có thể kiểm soát các permission mà nhiều dịch vụ sở hữu. Sau khi bạn chỉ định account dịch vụ, nhấp Next để hiển thị hộp thoại Authentication Mode như minh họa ở hình A.9.



Hình A.9: Authentication Mode

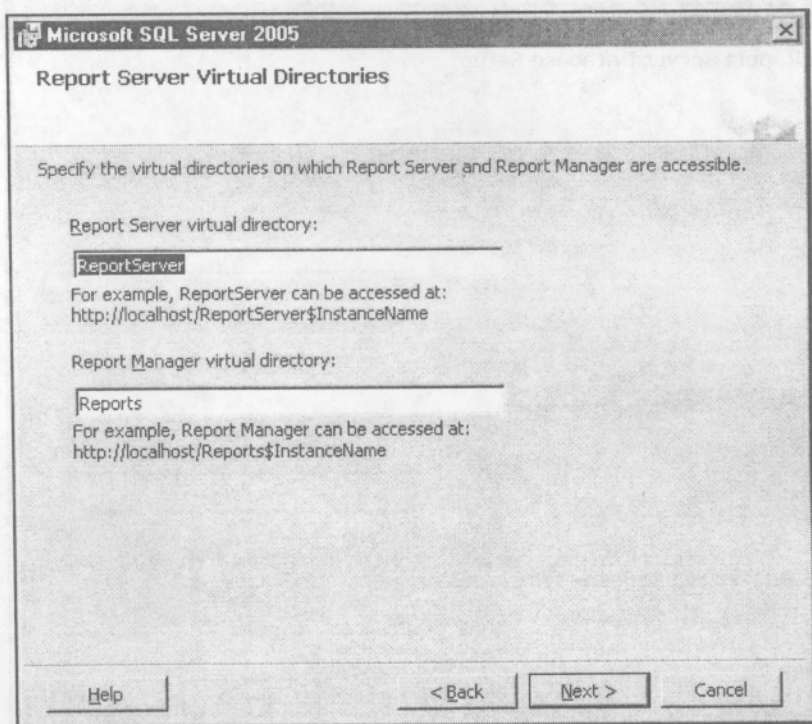
Hộp thoại Authentication Mode xác định loại xác thực user (người dùng) mà SQL Server 2005 sẽ sử dụng. Trị mặc định là Windows Authentication, nghĩa là các user account Windows cũng sẽ được sử dụng trong SQL Server. Đây thường là điều bạn muốn bởi vì nó giúp quản lý dễ dàng hơn ở chỗ chỉ một tập hợp logging account cần được quản lý và tập hợp login đó được duy trì bởi hệ điều hành chủ. Nó cũng an toàn hơn bởi vì với Windows authentication, ứng dụng không cần chuyển user ID qua mạng. Bạn cũng có thể chọn Mixed Mode Authentication, nghĩa là cả Windows login và SQL Server login đều có thể được chấp nhận. Trong trường hợp của các login SQL Server, bạn phải thêm các login này bằng tay vào SQL Server và chúng được duy trì độc lập với login Windows. Sau khi chọn chế độ xác thực mà server sẽ sử dụng, bạn cần chọn một password cho login SQL Server System Administrator (sa). Vì lý do bảo mật, bạn phải chọn một password không trống. Bạn nên xem xét kỹ việc tạo một password mạnh dài ít nhất tám ký tự, gồm các ký tự, số và các ký tự đặc biệt. Nhấp Next để hiển thị hộp thoại Collation Settings như minh họa ở hình A.10.



Hình A.10: Collation Settings

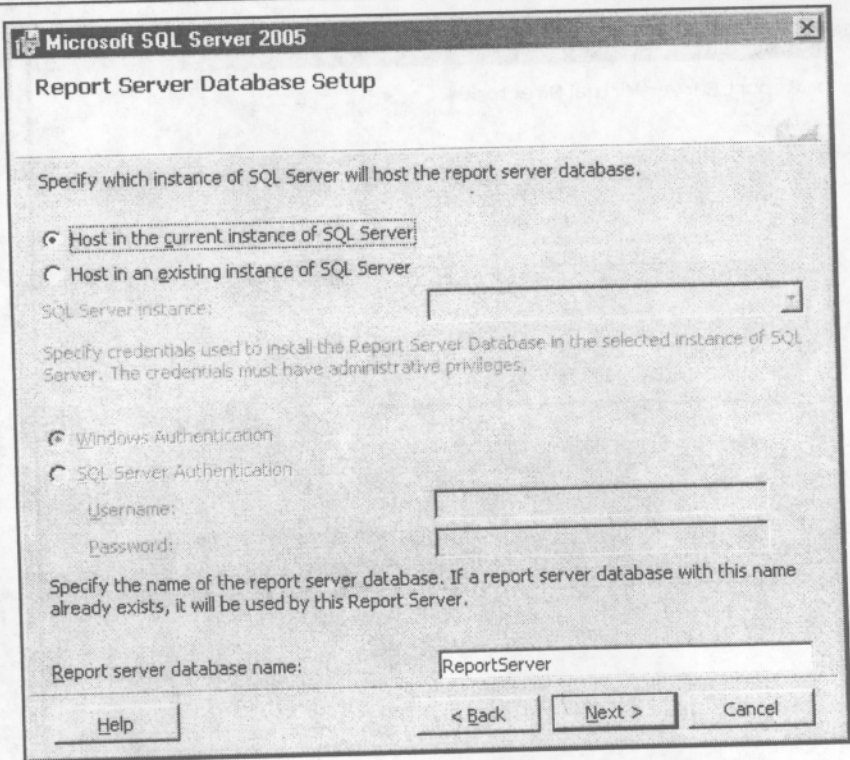
Hộp thoại Collation Settings cho phép bạn chỉ định thứ tự phân loại mặc định mà SQL Server 2005 sẽ sử dụng. Trong khi xác lập collation được chỉ định trong lúc cài đặt sẽ thiết đặt sự đối chiếu mặc định của SQL Server 2005, thứ tự đối chiếu cũng được thiết đặt cho mỗi cơ sở dữ liệu riêng lẻ. Nếu bạn chọn SQL collations để cài đặt Analysis Services, hộp thoại sẽ được hiển thị để hỏi bạn có muốn dùng Latin1_General collation dành cho Analysis Services hay không.

Nếu bạn chọn để cài đặt Reporting Services, việc nhấp Next sẽ hiển thị hộp thoại Report Server Virtual Directories như minh họa ở hình A.11.



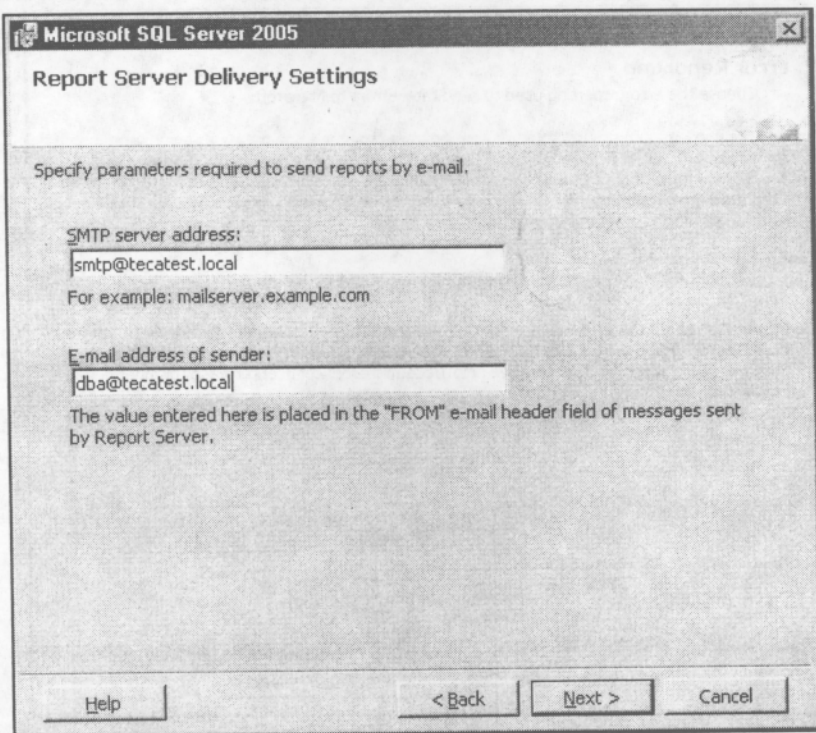
Hình A.11: Report Server Virtual Directories

Hộp thoại Report Server Virtual Directories được hiển thị chỉ nếu bạn đã chọn để cài đặt thành phần Reporting Services trên hộp thoại Components To Install đã minh họa trước đây trong hình A.5. Hộp thoại Reporting Services Virtual Directories cho phép bạn chỉ định thư mục ảo IIS sẽ được sử dụng để xuất bản các report Reporting Services cũng như Virtual Directory dùng để quản lý Reporting Services. Nếu bạn đang làm việc với instance SQL Server mặc định, các tên ReportingServer và Reporting sẽ được sử dụng. Khi bạn tạo một instance đã đặt tên, tên instance này được gắn thêm vào các hàng ReportServer\$ hay Reporting\$ để tạo các tên của các thư mục ảo xuất bản và quản lý. Ngay sau khi bạn đã chỉ định các tên thư mục ảo Reporting Services, việc nhấp Next sẽ hiển thị hộp thoại Report Server Database Setup như bạn thấy ở hình A.12.



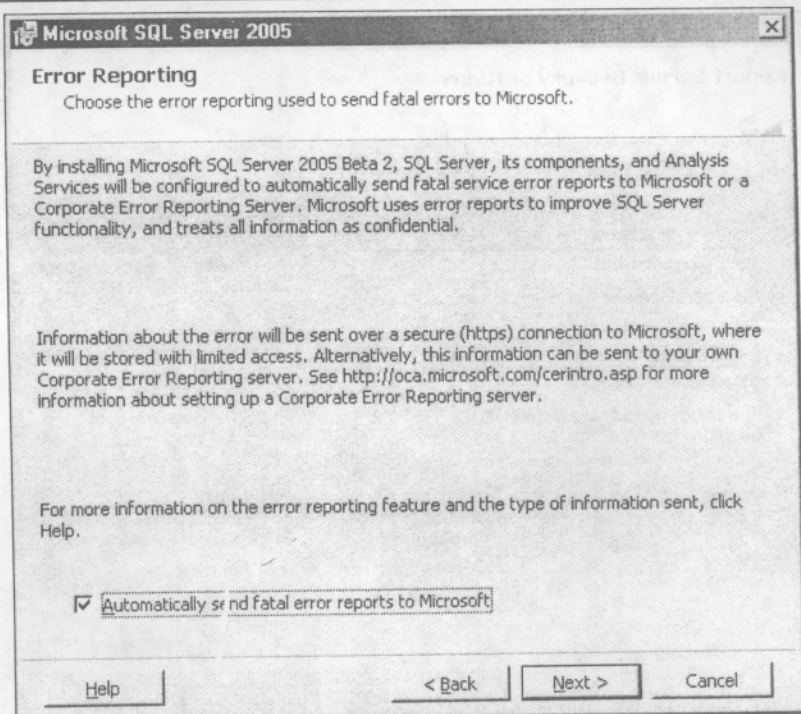
Hình A.12: Report Server Database Setup

Hộp thoại Report Server Database Setup cho phép bạn chọn nơi cài đặt cơ sở dữ liệu SQL Server được sử dụng bởi Reporting Services. Đây là cơ sở dữ liệu mà Reporting Services sẽ dùng để chứa các định nghĩa report của nó. Bản thân các report có thể sử dụng bất kỳ nguồn dữ liệu khác. Theo mặc định, chương trình cài đặt SQL Server 2005 sẽ tạo cơ sở dữ liệu Reporting Services trong instance SQL Server hiện hành. Tuy nhiên, bạn cũng có thể chọn cài đặt cơ sở dữ liệu Reporting Services trong một hệ thống SQL Server khác. Với hầu hết các tổ chức quy mô vừa và lớn hơn, việc tách biệt các Reporting Services ra thành server riêng của nó và giữ nó tách biệt với cơ sở dữ liệu sản phẩm là điều nên làm vì lý do thực thi. Hộp thoại Report Server Setup còn cho phép bạn chỉ định tên sẽ được sử dụng cho cơ sở dữ liệu Reporting Services. Đối với instance mặc định, tên cơ sở dữ liệu ReportServer sẽ được sử dụng. Nếu bạn đang cài đặt Reporting Services trên một instance đã đặt tên, thì tên cơ sở dữ liệu sẽ là ReportingServer\$InstanceName, trong đó InstanceName được thay thế bởi tên instance SQL Server của bạn. Cũng giống như với các tên thư mục ảo, bạn có thể thay đổi trị này. Sau khi bạn xác lập các tùy chọn cơ sở dữ liệu Reporting Services, nhấp Next để hiển thị màn hình cài đặt như minh họa ở hình A.13.



Hình A.13: Report Server Delivery Settings

Hộp thoại Reporting Services Delivery Settings cho phép bạn chỉ định địa chỉ server SMTP mà Reporting Services sẽ sử dụng để chuyển các thông báo e-mail và các report. Nó cũng cho phép bạn chỉ định địa chỉ khứ hồi mà Reporting Services sẽ dùng khi gửi các thông báo e-mail. Việc điền những thông tin này là tùy ý - bạn có thể trở lại và xác lập chúng hoặc thay đổi các trị sau khi tiến trình cài đặt đã hoàn tất. Việc nhấp Next sẽ làm hiển thị hộp thoại SQL Server 2005 Error Reporting. Hình A.14 minh họa hộp thoại Error Reporting.

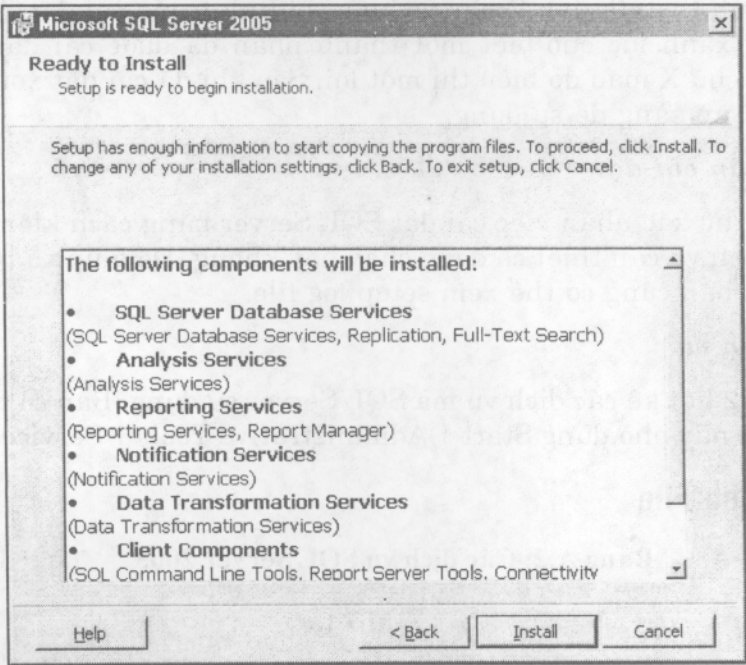


Hình A.14: Error Reporting.

Giống như sự hỗ trợ Watson mà Microsoft đã thêm vào Windows XP, màn hình báo cáo lỗi của SQL Server 2005 cho phép bạn tùy ý báo cáo các lỗi SQL Server trầm trọng cho Microsoft. Microsoft không thu thập bất kỳ thông tin cá nhân từ những báo cáo này. Chúng chỉ được sử dụng để giúp nhận dạng và loại bỏ các sự cố vốn có thể xảy ra bên trong SQL Server. Các báo cáo lỗi của SQL Server 2005 gửi thông tin sau đây đến Microsoft:

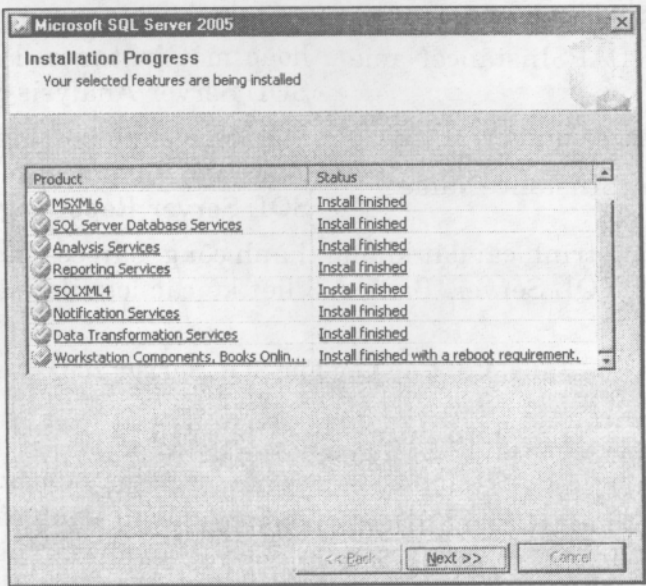
- ❖ Tình trạng của SQL Server khi lỗi xảy ra
- ❖ Phiên bản hệ điều hành
- ❖ Cấu hình phần cứng cơ bản
- ❖ Digital Product ID của SQL Server (dùng để nhận dạng đăng ký của bạn)
- ❖ Địa chỉ IP của server
- ❖ Thông tin từ sự lưu trữ về tiến trình đã gây ra lỗi

Báo cáo lỗi này là hoàn toàn tùy ý. Khi báo cáo lỗi tự động được mở theo mặc định, bạn có thể dễ dàng tắt nó bằng cách hủy chọn hộp kiểm Automatically Send Fatal Error Reports To Microsoft. Sau khi bạn đã làm việc xong với hộp thoại Error Reporting, nhấp Next để hiển thị hộp thoại Ready To Install như mặc định ở hình A.15.



Hình A.15: Ready To Install

Hộp thoại Ready To Install cho phép bạn xác nhận các lựa chọn của bạn. Nếu bạn cần thay đổi bất kỳ điều gì, bạn có thể dùng nút Back để lùi lại qua các màn hình cài đặt trước đây. Nhấp Install trên hộp thoại Ready To Install để bắt đầu tiến trình cài đặt SQL Server 2005. Khi tiến trình cài đặt diễn ra, màn hình như minh họa ở hình A.16 sẽ được hiển thị.



Hình A.16: Màn hình Installation Progress

Màn hình Installation Progress hiển thị tình trạng cài đặt. Một dấu kiểm màu xanh lục cho biết một thành phần đã được cài đặt thành công. Một chữ X màu đỏ biểu thị một lỗi. Sau khi đã cài đặt xong, SQL Server sẽ sẵn sàng để sử dụng.

• *Xác nhận cài đặt*

Bạn có thể xác nhận việc cài đặt SQL Server bằng cách kiểm tra để xem các dịch vụ cần thiết có đang chạy hay không. Để giúp xử lý các sự cố cài đặt, bạn cũng có thể xem setup log file.

Các dịch vụ

Bảng A.2 liệt kê các dịch vụ mà SQL Server sử dụng. Bạn có thể xem các dịch vụ này nhờ dùng Start | Administrative Tools | Service applet.

Setup Log File

Bảng A.2: Các dịch vụ SQL Server 2005

Dịch vụ	Mô tả
SQL Server (MSSQLSERVER) and/or MSSQL\$InstanceName	Bộ máy cơ sở dữ liệu SQL Server và/hoặc một instance đã đặt tên của bộ máy cơ sở dữ liệu SQL Server
SQL Server Agent (MSSQLSERVER) and/or SQLAgent\$InstanceName	Nơi lập thời biểu công việc SQL Server và/hoặc một instance đã đặt tên của nơi lập thời biểu công việc SQL Server
MSSQLServerOLAPService and/or MSOLAP\$InstanceName	SQL Server Analysis Services và/hoặc một instance đã đặt tên của SQL Server Analysis Services
ReportServer and/or ReportServer\$InstanceName	SQL Server Reporting Services và/hoặc một instance đã đặt tên của SQL Server Reporting Services

Nếu chương trình cài đặt không thành công, bạn có thể nhìn vào các setup log file SQL Server. Bảng A.3 liệt kê các log file mà SQL Server 2005 sử dụng.

Bảng A.3: Các Log File SQL Server 2005

Log File	Thư mục	Mô tả
sqlstpX.log	%temp%	File này chứa thông tin cài đặt từ trình cài đặt Windows. Mỗi lần chạy sẽ tăng trị của X.

Sqlsetup<xxxx>_machine>_support.log	%temp%	Được tạo bởi chẳng hạn như setup.
SQLSetup*.txt	%sqlserver%\90\Setup\Bootstrap\LOG	Các file setup bootstrap của SQL Server
sqlrunXlog.log	%temp%	Chứa thông tin thủ tục cài đặt. Mỗi lần chạy sẽ gia tăng trị của X.
dasetup.log	%windows%	Nhật trình cài đặt Microsoft Data Access Components.
errorlog.log	%sqlserver%\mssql\log\errorlog	Nhật trình lỗi SQL Server.

Nâng cấp lên SQL Server 2005

Ngoài việc tạo ra các phần cài đặt mới, chương trình cài đặt SQL Server 2005 cũng có thể được sử dụng để nâng cấp các phần cài đặt SQL Server 7 và SQL Server 2000 hiện có. Tuy nhiên, nó không thể được sử dụng để nâng cấp các phần cài đặt SQL Server 6.5 hiện có.

Nâng cấp từ SQL Server 7 và 2000

Chương trình setup hỗ trợ các phần nâng cấp trực tiếp từ SQL Server 7 và SQL Server 2000 lên SQL Server 2005. Các cấu trúc trên đĩa thiết yếu của SQL Server là như nhau, và chương trình setup có thể thực hiện thành công sự di chuyển tại chỗ cho các phần cài đặt SQL Server 7 và SQL Server 2000 lên SQL Server 2005. Bảng A.4 liệt kê các tình huống nâng cấp được hỗ trợ cho SQL Server 7 và SQL Server 2000.

Bảng A.4: Các tùy chọn nâng cấp SQL Server 2005

Ấn bản SQL Server hiện có	Nâng cấp lên SQL Server 2005 Standard Edition (32 bit & 64 bit)	Nâng cấp lên SQL Server 2005 Enterprise Edition (32 bit & 64 bit)
MSDE 2000	Được	Không được
SQL Server 7 Standard Edition	Được	Được
SQL Server 7 Enterprise Edition	Không được	Được
SQL Server 2000 Standard Edition	Được	Được
SQL Server 2000 Enterprise Edition	Không được	Được

Nâng cấp từ SQL Server 6.5 (hoặc phiên bản cũ hơn)

Việc nâng cấp trực tiếp từ SQL Server 6.5 hay bất kỳ ấn bản cũ hơn của SQL Server lên SQL Server 2005 là không được hỗ trợ. Các cấu trúc trên đĩa và sự lưu trữ cơ sở dữ liệu được sử dụng bởi 6.5 và các phiên bản cũ hơn thì khác với các cấu trúc trên đĩa mà SQL Server 2005 sử dụng. Do đó, bạn không thể thực hiện trực tiếp việc nâng cấp tại chỗ. Cách duy nhất để thực hiện việc nâng cấp tại chỗ là gián tiếp, bằng cách trước tiên nâng cấp SQL Server 6.5 lên SQL Server 2000 rồi nâng cấp SQL Server 2000 lên SQL Server 2005. Tuy nhiên, việc thực hiện hai phần nâng cấp này để đạt được một có lẽ không phải là phương pháp hiệu quả nhất. Giả sử bạn có phần cứng thích hợp và các cấp độ hệ điều hành tối thiểu, chương trình setup có thể được sử dụng để cài đặt một instance mới của SQL Server 2005 trên một hệ thống đang chạy SQL Server 6.5. Tuy nhiên, hầu hết các tiến trình cài đặt SQL Server 6.5 hiện có sẽ có thể đang chạy trên phần cứng đã lỗi thời, và trong hầu hết trường hợp, bạn sẽ muốn có phần cứng mới. Nói tóm lại, cách tốt nhất để chuyển từ SQL Server 6.5 sang SQL Server 2005 là thực hiện một sự cài đặt mới của SQL Server 2005 trên phần cứng sạch rồi thực hiện nâng cấp bằng tay bằng việc chuyển dữ liệu và các đối tượng cơ sở dữ liệu với DTS và biên dịch lại các thủ tục lưu trữ cũng như các đối tượng cơ sở dữ liệu khác.

Mục lục

PHẦN I

CÁC TÍNH NĂNG QUẢN TRỊ CƠ SỞ DỮ LIỆU

Chương 1 : Các tính năng của bộ máy lưu trữ và kiến trúc cơ sở dữ liệu	7
Hỗ trợ phần cứng mới	7
Bộ máy SQL Server	10
Sự bảo mật	18
Chương 2 : Các công cụ quản trị và phát triển cơ sở dữ liệu	27
Các công cụ quản lý và phát triển	27
Khi nào cần sử dụng SQL Server Management Studio và Business Intelligence Development Studio	41
Các công cụ thực thi	45
Các cơ cấu quản lý mới	50
Chương 3 : Tính khả dụng và các tính năng phục hồi	53
Sự bảo vệ để tránh lỗi cơ sở dữ liệu hay Server	53
Các cải tiến về tính khả dụng của cơ sở dữ liệu	59

PHẦN II

CÁC TÍNH NĂNG PHÁT TRIỂN CƠ SỞ DỮ LIỆU

Chương 4 : Các tính năng lập trình	68
Sự hợp nhất Common Language Run-time (CLR)	68
Các cải tiến của T-SQL	91
Các cải tiến của ADO.NET	98
Chương 5 : Các dịch vụ thông báo	108
Tổng quan về Notification Services	108
Kiểu kiến trúc của Notification Services	110
Phát triển các ứng dụng Notification Services	112
Chương 6 : SQL Server Service Broker	129
Tổng quan về SQL Server Service Broker	129
Phát triển với SQL Server Service Broker	135
Quản trị Service Broker	143

Chương 7 : Sự hợp nhất XML	146
Tổng quan	146
Kiểu dữ liệu XML gốc	147
Sự hỗ trợ XQuery	154
Các index XML	154
Các cải tiến FOR XML	156
Các cải tiến OPENXML	159
XML Bulk Load	161
Sự truy cập HTTP SOAP gốc	162
Các cải tiến XML cho server phân tích	163
Các khung xem danh mục XML hệ thống	164

PHẦN III

CÁC TÍNH NĂNG KINH DOANH THÔNG MINH

Chương 8 : Các dịch vụ báo cáo	166
Kiểu kiến trúc Reporting Services	167
Các thành phần Reporting Services	169
Tạo Report	178
Chương 9 : Các dịch vụ hợp nhất	191
Tổng quan	191
Kiểu kiến trúc Integration Services mới	192
Các thành phần của Package Integration Services	196
Các công cụ Integration Services	209
Các công cụ đóng gói Integration Services	219
Chương 10 : Các dịch vụ phân tích	222
Giới thiệu sơ lược	222
Tổng quan về Analysis Services	223
Các cải tiến của bộ máy Analysis Services	225
Các cải tiến về sự quản lý Analysis Services	228
Các cải tiến về sự phát triển	232
Khai thác dữ liệu	244
Phụ lục : Cài đặt và nâng cấp	246
Cài đặt SQL Server 2005	246

QUẢN LÝ CƠ SỞ DỮ LIỆU VỚI SQL SERVER 2005

KS: NGUYỄN NAM THUẬN – LỮ ĐỨC HÀO

NHÀ XUẤT BẢN GIAO THÔNG VẬN TẢI

Chịu trách nhiệm xuất bản:

LÊ TỬ GIANG

Biên tập : Hoàng Chí Dũng

Trình bày : Thế Anh

Bìa : Lê Thành

TỔNG PHÁT HÀNH

CÔNG TY CỔ PHẦN VĂN HÓA NHÂN VĂN

Số 1 Trường Chinh – P.11 – Q.Tân Bình – TP. Hồ Chí Minh

ĐT: 9712285 – 9710306 – 8490048 • FAX: 9712286

NHÀ SÁCH NHÂN VĂN • 189 CMT8 – P.7 – Q.Tân Bình – TP. Hồ Chí Minh

ĐT: 9700420

NHÀ SÁCH NHÂN VĂN • 486 Nguyễn Thị Minh Khai – P.2 – Q.3 – TP. Hồ Chí Minh

ĐT: 8396733

SIÊU THỊ SÁCH NHÂN VĂN • 394 Quốc lộ 15 – P. Trung Dũng – TP. Biên Hòa

ĐT: (061)917839

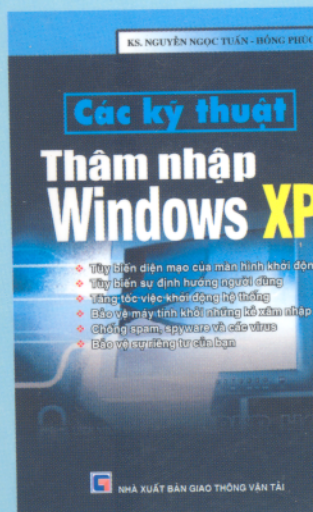
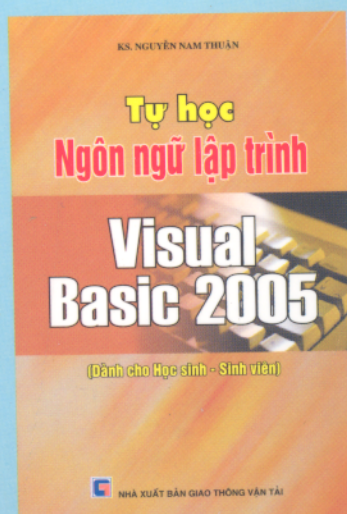
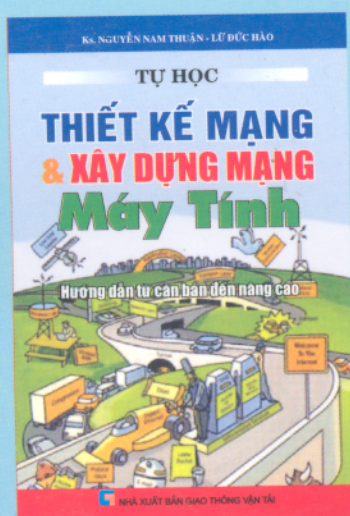
In 1000 cuốn, khổ 16 x 24cm. Tại Công ty cổ phần In Gia Định, Số 9D
Nơ Trang Long – Q. Bình Thạnh, TP. HCM, ĐT: 8412644.

Giấy đăng ký KHXB số: 178-230/XB-QLXB, do Cục xuất bản cấp ngày
03 tháng 03 năm 2005.

Giấy trích ngang KHXB số 178/XB do Nhà Xuất Bản GTVT cấp ngày 29
tháng 12 năm 2005.

In xong và nộp lưu chiểu tháng 01 năm 2006.

Quản lý cơ sở dữ liệu Với Microsoft SQL Server 2005



CÔNG TY CỔ PHẦN VĂN HÓA NHÂN VĂN
Nhà Sách NHÂN VĂN

*Số 01 Trường Chinh, P. 11, Q. Tân Bình
Tel: 9712285 - 9710306 * Fax: 9712286
*486 Nguyễn Thị Minh Khai, P.2, Quận 3,
Thành Phố Hồ Chí Minh - Tel/Fax: 8396733

quản lý csdl với microsoft

