Chương 1

LẬP TRÌNH CSDL VỚI SQL SERVER

Nội dung

1. Các đối tượng liên quan đến một CSDL trên SQL Server.
2. Lập trình trên SQL Server
3. Thủ tục (Store procedures)
4. Hàm người dùng định nghĩa (Functions)
5. Triggers

1. Các đối tượng liên quan đến một CSDL trên SQL Server

1.1. **Giới thiệu SQL Server:**

- SQL Server là một hệ quản trị cơ sở dữ liệu quan hệ (RDBMS)
- SQL Server là một hệ quản trị cơ sở dữ liệu theo mô hình Client/Server.

SQL Server cung cấp đầy đủ các công cụ để:

- Dễ dàng xây dựng một cơ sở dữ liệu quan hệ lớn (mỗi cơ sở dữ liệu có thể chứa 2 tỷ quan hệ và mỗi quan hệ có thể chứa đến 1024 thuộc tính)
- Giải quyết tình trạng va chạm giữa các user khi cùng truy xuất một dữ liệu tại cùng một thời điểm.
- Bảo đảm các ràng buộc toàn vẹn trên cơ sở dữ liệu.
- Bảo vệ an toàn cơ sở dữ liệu (quản lý nhiều mức độ để truy cập vào cơ sở dữ liệu).
- Truy vấn dữ liệu nhanh.

1.2 . Database:

1.2.1 Khái niệm:

- Mỗi SQL Server có thể chứa nhiều database. Một database bao gồm tập hợp các table và các đối tượng khác như: diagrams, views, trigger,….
- Một hệ phục vụ SQL Server có tối đa 32.767 database. Mỗi database có kích thước tối thiểu là 1 MB.
- Để có thể tạo một database người dùng phải là thành viên của sysadmin và dbcreator.
- User tạo database cũng sẽ trở thành chủ sở hữu của database.

Có 3 loại tập tin được sử dụng để lưu trữ database:

- Mỗi database có một tập tin dữ liệu cơ sở (primary data file) dùng để chứa dữ liệu và các thông tin khởi động database.
- Một database còn có các tập tin phụ dùng để chứa tất cả dữ liệu liên quan nhưng không được đặt bên trong tập tin dữ liệu cơ sở.
- Một database có ít nhất một tập tin lưu vết (log file) chứa các thông tin giao dịch của database dùng để phục hồi dữ liệu. Kích thước tối thiểu của một log file là 512 KB.

1.2.2. Tables:

- Table là nơi lưu trữ dữ liệu thật sự của databse. Dữ liệu được lưu trữ trên table theo dạng hàng, cột.
- Mỗi một database trên SQL Server có thể lưu được 2 tỷ table.
- Mỗi table có tối đa 1024 cột
- Số lượng các hàng và tổng kích thước của table được giới hạn bởi dung lượng cho phép của kho lưu trữ.
- Kích thước tối đa cho mỗi hàng là 8060 bytes
- Mỗi table có thể chứa đến 249 chỉ mục loại nonclustered và 1 chỉ mục loại clustered

1.2.3. Views

- Là các bảng dữ liệu “ảo” đặc biệt đáp ứng nhu cầu rút trích dữ liệu của một hoặc nhiều table.
- Việc tạo một view chỉ thực hiện trên database hiện hành.
- Một view có thể tham khảo tối đa 1024 cột
- Về bản chất một view là một table “ảo”. Như vậy Chúng ta có thể thao tác trên view như thao tác trên một table.

1.2.4. Diagrams

- Là đối tượng dùng để tạo, quản và xem các đối tượng database ở dạng đồ họa. Giống như Relationship của Access, diagrams cho phép Chúng ta tạo các mối quan hệ giữa các table trong một database một cách trực quan.
- Khi tạo các mối quan hệ giữa các table trên diagrams, SQL Server sẽ tự động phát sinh các trigger kiểm tra ràng buộc dữ liệu tương ứng, điều này giúp bảo vệ toàn vẹn cơ sở dữ liệu.

2. Lập trình trên SQL Server

2.1. Biến cục bộ trong Transact-SQL.

2.1.1 Khai báo : để khai báo biến cục bộ trong T_SQL trong câu lệnh Declare theo cú pháp sau :

Cú pháp :

Declare @tenbien kiểu_dữ_liệu [,…]

Trong đó :

- Tenbien : tên của biến được khai báo, tên biến luôn bắt đầu bằng ký tự @.
- Kiểu _dữ_liệu :là các kiểu dữ liệu cơ bản của SQL Server hoặc các kiểu dữ liệu do người dùng định nghĩa. Các kiểu dữ liệu text, ntext hoặc image không được chấp thuận trong việc khai báo biến.

2.1.2 Gán trị cho biến

Sử dụng lệnh SET hoặc SELECT cùng với phép gán(=)

Thông thường lệnh SET chỉ để gán các giá trị cụ thể, hoặc các biểu thức tính toán hoặc các giá trị tính toán từ các biến khác.

Ví dụ : gán trị là 0 cho biến @SLDAT

```
DECLARE @SLDAT INT
SET @SLDAT=0
```

2.1.3 Xem giá trị hiện hành của biến

Sử dụng lệnh PRINT hoặc SELECT in giá trị của biến ra màn hình.

Cú pháp :

PRINT @SLDAT

Hoặc SELECT @SLDAT

2.1.4 Phạm vi hoạt động của biến

Trong T_SQL phạm vi hoạt động của biến chỉ nằm bên trongmột thủ tục nội tại (Store Procedure) hoặc một lô (batch) chứa các câu lệnh mà biến được khai báo bên trong đó.

Ví dụ :

```
DECLARE @SLDAT INT
SET @SLDAT=10
PRINT @SLDAT
GO
SET @SLDAT=20
PRINT @SLDAT
GO
```

Hệ thống sẽ hiển thị thông báo lỗi bởi vì lô thứ hai không hiểu biến @SLDAT đã được khai báo trong lô thứ nhất.

Lưu ý :

Đối với các lệnh CREATE như là : CREATE DEFAULT, CREATE PROCEDURE, CREATE RULE, CREATE TRIGGER, CREATE VIEW không được phép kết hợp với các lệnh khác trong cùng một lô.

2.2. Biến hệ thống

Không giống các ngôn ngữ lập trình khác, T_SQL không có khái niệm biến toàn cục. Thay vào đó Microsoft SQL Server cung cấp cho người lập trình danh sách các biến hệ thống.

- Biến hệ thống luôn bắt đầu bằng @@
- Người lập trình không thể can thiệp trực tiếp để gán giá trị vào biến hệ thống, giá trị mà chúng đang lưu trữ là do hệ thống Microsoft SQL Server cung cấp.

Một số biến hệ thống thường dùng

Tên biến	Kiểu trả về	Dùng để trả về
Connections	Số nguyên	Tổng số các kết nối vào Micosoft SQL Server từ khi nó được khởi động.
Error	Số nguyên	Số mã lỗi của câu lệnh thực hiện gần nhất. Khi một câu lệnh thực hiện thành công thì biến này có giá trị là 0.
Fetch_Status	Số nguyên	Trạng thái của việc đọc dữ liệu trong bảng theo cơ chế đọc từng dòng mẫu tin (cusor). Khi đọc dữ liệu của mẫu tin thành công thì biến này có giá trị là 0.
Language	Chuỗi	Tên ngôn ngữ mà hệ thống Microsoft SQL Server đang sử dụng. Mặc định là US_English.

Tên biến	Kiểu trả về	Dùng để trả về
RowCount	Số nguyên	Tổng số mẫu tin được tác động vào câu lệnh truy vấn gần nhất.
ServerName	Chuỗi	Tên của máy tính cục bộ được cài đặt SQL Server.
Version	Chuỗi	Phiên bản, ngày của sản phẩm Microsoft SQL Server và loại CPU, hệ điều ành của máy chủ cài Micosoft SQL Server.

2.3. Các toán tử

- Toán tử số học : +, -, *, /, % (phép chia lấy phần dư)
- Toán tử nối chuỗi : +
- Toán tử so sánh : >, >=, =, <, <=, <>, !< (không nhỏ hơn), !> (không lớn hơn).
- Toán tử luận lý : AND, OR, NOT

2.4 Các câu lệnh truy vấn dữ liệu

2.4.1 Lệnh SELECT

```
SELECT [DISTINCT|ALL] <*| [columnExpression [AS
                    newName]] [,…]> [INTO < List of variables>]
FROM  tableName [alias] [,…]
[WHERE  <  row conditions>]
[GROUP BY columnList
[HAVING <group of rows conditions>]]
[ORDER BY columnList];
```

1. Mệnh đề SELECT cho phép thực hiện phép chiếu của ĐSQH, columnExpression là một biểu thức được chọn, mỗi biểu thức được phân cách nhau bởi dấu phẩy.

- Biểu thức có thể là một hằng, một biến (cột), hoặc sự kết hợp giữa các hằng, các biến với các phép toán. Mỗi biểu thức có thể có một bí danh (alias) đứng ngay phía sau được gọi là bí danh cột (column alias); bí danh cột chỉ được sử dụng trong mệnh đề SELECT.
- Nếu danh sách các biểu thức là dấu * (asterisk) thì tất cả các cột được chọn. Cột có thể có dạng tableName.columnName.

Từ khóa DISTINCT loại bỏ các giá trị trùng nhau trong câu truy vấn. Nếu có nhiều cột được chọn thì DISTINCT ảnh hưởng đến toàn bộ các cột này. Từ khóa DISTINCT phải được đặt ngay sau từ khóa SELECT.

Từ khóa ALL lấy tất cả các giá trị (kết quả của câu truy vấn) kể cả các giá trị trùng nhau. Từ khóa ALL phải được đặt ngay sau từ khóa SELECT.

2. Mệnh đề FROM thực hiện phép tích của ĐSQH, dùng để chỉ ra các bảng dữ liệu cần lấy ra, mỗi bảng được cách nhau bởi dấu phẩy. Mỗi bảng có thể có một bí danh (alias) đứng ngay phía sau.

3. Mệnh đề WHERE thực hiện các phép chọn, phép kết của ĐSQH. Row conditions là các điều kiện được xét trên mỗi hàng, các hàng nào thỏa mãn các điều kiện này thì được đưavào kết quả của truy vấn.

4. Mệnh đề GROUP BY được dùng để phân chia các hàng của một bảng thành các nhóm nhỏ hơn. Các hàm nhóm có thể được sử dụng để trả về thông tin chung cho mỗi nhóm

5. Mệnh đề HAVING được dùng để xác định các nhóm được đưa vào kết quả của truy vấn. Group of rows conditions là các điều kiện được xét cho mỗi nhóm.

6. Mệnh đề ORDER BY luôn luôn là mệnh đề cuối cùng của lệnh SELECT. Thứ tự ngầm định là tăng dần (ASC – Ascending) hoặc giảm dần (DESC – Descending). Từ khóa ASC hoặc DESC đứng ngay sau tên cột trong mệnh đề ORDER BY.

2.4.2 Lệnh INSERT INTO

Lệnh Insert into cho phép thêm mới một dòng dữ liệu vào trong bảng

Cú pháp :

INSERT INTO Ten_bang [(danhsachcaccot)]

VALUES (Danhsachgiatri)

Trong đó :

- Ten_bang : tên bảng được thêm mới dòng dữ liệu
- Danhsachcaccot : danh sách các cột hiện có trong bảng.

Bạn có thể không cần chỉ định tên các cột nhưng phải đưa ưữ liệu vào theo đúng thứ tự vật lý của các cột bên trong bảng.

2.4.3 Lệnh DELETE FROM

Lệnh DELETE FROM cho phép hủy bỏ các dòng dữ liệu hiện đang có bên trong một bảng

Cú pháp :

```
DELETE [FROM] Ten_bang
 [FROM Ten_bang_1 INNER|LEFT|RIGHT JOIN  Ten_bang_2
  ON Bieu_thuc_lien_ket]
[WHERE Dieu_kien_xoa]
```

Trong đó :

- Ten_bang : tên bảng có các dòng dữ liệu muốn xóa
- Ten_bang_1, Ten_bang_2 : tên các bảng có quan hệ dữ liệu.

2.4.4 Lệnh UPDATE SET

Lệnh UPDATE SET cho phép cập nhật các dòng dữ liệu hiện đang có bên trong một bảng

Cú pháp :

```
UPDATE Ten_bang
SET Ten_cot=bieu_thuc[,…]
[FROM Ten_bang_1 INNER|LEFT|RIGHT JOIN Ten_bang_2
  ON Bieu_thuc_lien_ket]
[WHERE Dieu_kien_sua_doi]
```

Trong đó :

- Ten_bang : tên bảng có các dòng dữ liệu muốn cập nhật
- Ten_cot : tên cột muốn sửa đổi giá trị dự liệu

- Bieu_thuc : là một giá trị cụ thể hay một hàm tính toán mà giá trị trả về của nó sẽ được cập nhật vào cột trong bảng chỉ định.

-Ten_bang_1, Ten_bang_2 : tên các bảng có quan hệ dữ liệu.

- Dieu_kien_sua : là biểu thức luận lý chỉ định các dòng dữ liệu phải thỏa điều kiện đưa ra mới được sửa đổi.

3. Thủ tục nội tại (Store procedures)

3.1 Khái niệm về thủ tục nội tại

Giống như ý nghĩa của việc sử dụng thủ tục dùng chung trong những ngôn ngữ lập trình khác, thủ tục nội tại trong Microsoft SQL Server dùng để tạo ra những xử lý thường dùng bên trong ứng dụng và nhằm để chia nhỏ các xử lý theo mô hình thiết kế xử lý top- down, nhằm đơn giản hóa các xử lý phức tạp.

Thủ tục nội tại thật sự là một tập hợp chứa các dòng lệnh, các biến, các cấu trúc điều khiển bên trong ngôn ngữ T_SQL, dùng để thực hiện một hành động nào đó, tất cả các nội dung của thủ tục nội tại sẽ được lưu trữ tại CSDL của Microsoft SQL Server.

Các đặc trưng của một thủ tục nội tại :

- Tên thủ tục nội tại.
- Tham số truyền giá trị vào và tham số nhận giá trị trả về.
- Gọi thi hành một thủ tục nội tại khác.
- Phạm vi hoạt động của một thủ tục nội tại chỉ có tính cục bộ bên trong một CSDL lưu trữ thủ tục đó.
- Được gọi thi hành trong môi trường không phải là Microsoft SQL Server.
- Được biên dịch ở lần thi hành đầu tiên và cách hoạt động của nó được cất lại, nên các lần gọi thi hành sau nó sẽ hoạt động nhanh hơn nhiều.

3.2 Các hành động cơ bản với thủ tục nội tại

3.2.1 Tạo mới một thủ tục nội tại

Cú pháp :

```
CREATE PROC[EDURE] Tên_thủ_tục
[Danh sách các tham số]
AS
[DECLARE Biến_cục_bộ]
 Các_lệnh
GO
```

3.2.2 Gọi thực hiện một thủ tục nội tại

Cú pháp :

```
EXEC[UTE] Tên_thủ_tục
```

3.2.3 Hủy bỏ một thủ tục nội tại

Cú pháp :

DROP PROC[EDURE] Tên_thủ_tục

3.2.4 Thay đổi nội dung của một thủ tục nội tại

Cú pháp :

ALTER PROC[EDURE] Tên_thủ_tục

AS

[DECLARE biến_cục_bộ]

Các _lệnh

4. Hàm người dùng định nghĩa (Functions)

4.1 Hàm không có tham số

a/ Trả về giá trị kiểu vô hướng

- Trả về giá trị kiểu vô hướng là số
- Trả về giá trị kiểu vô hướng là chuỗi
- Trả về giá trị kiểu vô hướng là giá trị boolean

b/ Trả về giá trị kiểu table

4.2 Hàm có tham số

a/ Trả về giá trị kiểu vô hướng

- Trả về giá trị kiểu vô hướng là số
- Trả về giá trị kiểu vô hướng là chuỗi
- Trả về giá trị kiểu vô hướng là giá trị boolean

b/ Trả về giá trị kiểu table

5. Trigger.

- Trigger là một loại stored procedure được định nghĩa đặc biệt để thực thi một cách tự động khi có một câu lệnh UPDATE, INSERT hoặc DELETE tác động vào table.
- Trigger là một công cụ mạnh, khi dữ liệu bị sửa đổi nó sẽ tự động thực hiện việc ép buộc các giao dịch sửa đổi này theo các qui tắc đã định (các ràng buộc dữ liệu) nhằm mục đích đảm bảo tính toàn vẹn dữ liệu.

Trigger kiểm tra INSERT

Trigger loại này dùng để tự động kiểm tra vi phạm và bắt buộc thao tác thêm mẫu tin vào table phải tuân theo qui tắc định trước dựa trên tính duy nhất của khóa chính, và các mối quan hệ ràng buộc giữa các table.

Ví dụ : trường hợp thêm một mẫu tin vào Table “con” chúng ta cần phải kiểm tra xem giá trị thuộc tính khóa ngoại có tồn tại trong tập giá trị của thuộc tính khóa chính của table “cha”. Nếu chưa tồn tại -> báo lỗi.

Trigger kiểm tra Delete

Trigger loại này dùng để tự động kiểm tra vi phạm và bắt buộc thao tác xóa mẫu tin trên table phải tuân theo qui tắc định trước dựa trên mối quan hệ ràng buộc giữa các table.

Ví dụ : trường hợp xóa mẫu tin trên một table mà table này tồn tại mối quan hệ “cha- con” với table khác và trong mối quan hệ này nó có vai trò là “cha”.

Như vậy, về mặt an toàn dữ liệu khi xóa mẫu tin trên table “cha” -> thì cần phải xóa những mẫu tin trong table “con” có liên quan (qua thuộc tính khóa ngoại).

Trigger kiểm tra Update

Trước tiên ta cần hiểu về bản chất UPDATE, UPDATE thật ra là bao gồm hai hành động : DELETE và INSERT.

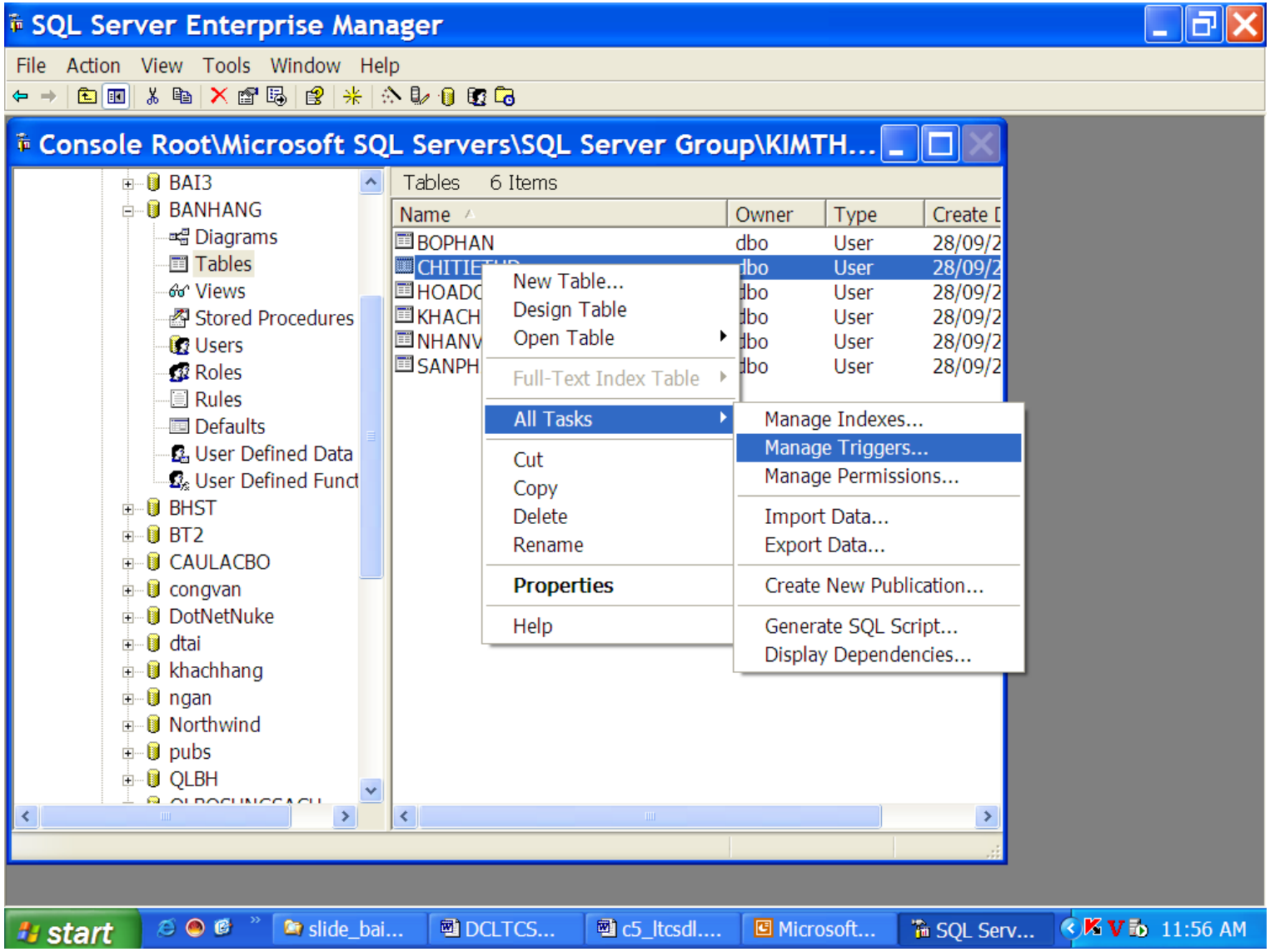
Như vậy, kiểm tra một thao tác UPDATE là kiểm tra liên tục hai hành động là : xóa mẫu tin đang tồn tại và thêm mẫu tin mới.

5.1 Tạo Trigger sử dụng công cụ (ENTERPRISE MANAGER)

- **Tạo Trigger**

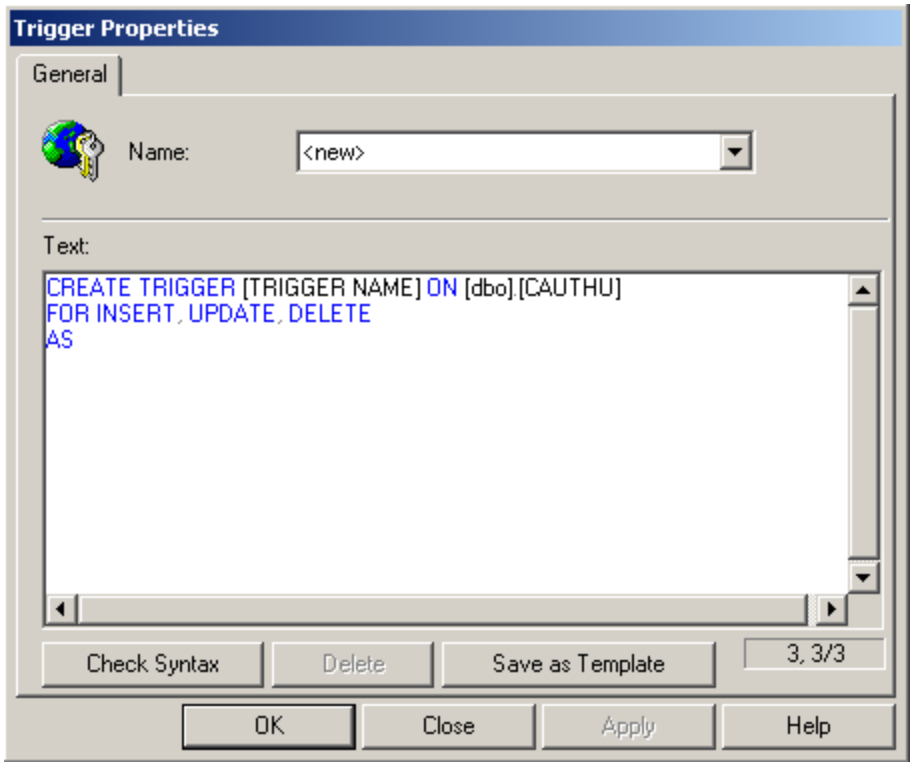
Các bước thực hiện :

1. Chọn Server Group -> chọn server
2. Kích chọn mục Database -> chọn database chưa table muốn tạo trigger-> kích chọn mục Tables.
3. Trong khung trình bày các đối tượng table của database -> kích phải chuột tại tên table muốn tạo trigger -> kích All Task -> kích Manager Triggers…



Kích chọn Manager Triggers

4. Tại Mục Name -> kích <new>
5. Tại mục Text, gõ vào các câu lệnh cho trigger ở trong khung văn bản
6. Để kiểm tra cú pháp, kích Check Syntax



❑ Xóa Trigger sử dụng công cụ (ENTERPRISE MANAGER)

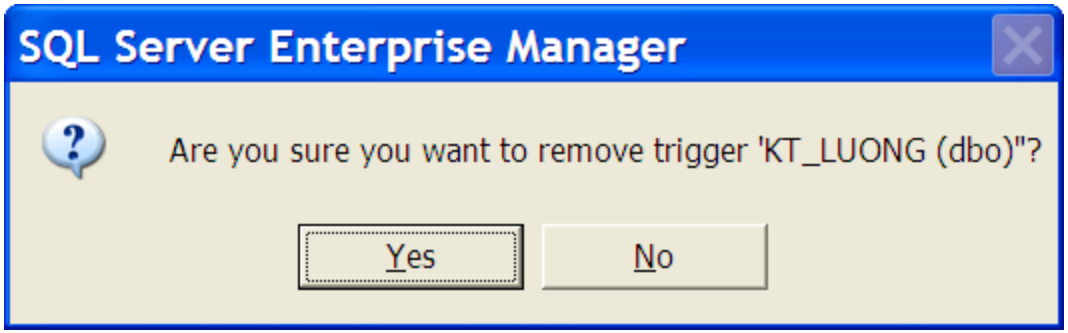
Các bước thực hiện :

1. Chọn Server Group -> chọn server
2. Kích chọn mục Database -> chọn database chưa table muốn xóa trigger-> kích chọn mục Tables.
3. Trong khung trình bày các đối tượng table của database -> kích phải chuột tại tên table muốn xóa trigger -> kích All Task -> kích Manager Triggers…

4. Tại Mục Name -> kích chọn trigger cần xóa.

5. Kích Delete.

6. Xác nhận lại quyết định xóa.



❑ Sửa đổi Trigger đang tồn tại

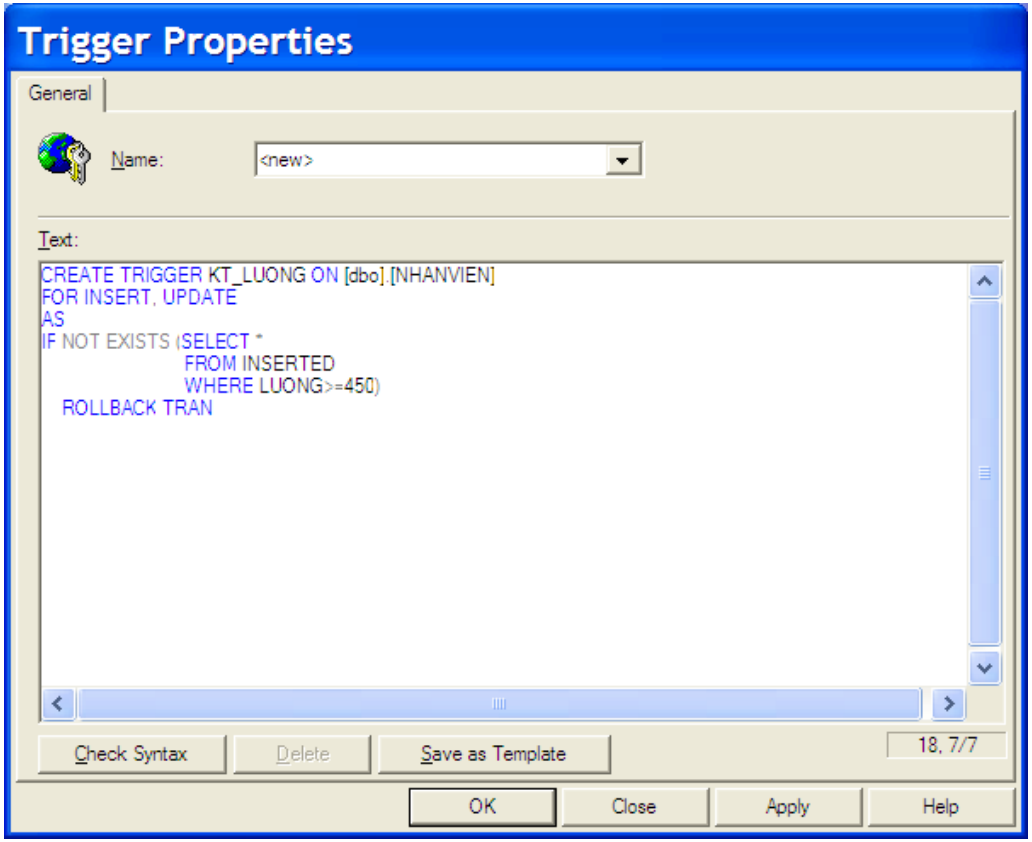
Các bước thực hiện :

1. Chọn Server Group -> chọn server
2. Kích chọn mục Database -> chọn database chưa table muốn sửa trigger-> kích chọn mục Tables.
3. Trong khung trình bày các đối tượng table của database -> kích phải chuột tại tên table muốn sửa trigger -> kích All Task -> kích Manager Triggers…

4. Tại Mục Name -> kích chọn tên trigger cần sửa.

5. Kích Check Syntax để kiểm tra lỗi cú pháp.

6. Kích Apply để áp dụng sửa đổi.



5.2 Tạo Trigger sử dụng lệnh T-SQL

Câu lệnh CREATE TRIGGER có thể được định nghĩa với các mệnh đề FOR UPDATE, FOR INSERT, hoặc FOR DELETE.

Chương 2

XỬ LÝ DỮ LiỆU VỚI ADO.NET

Nội dung chương 2

1. GiỚI THIỆU ADO.NET
2. CÁC ĐỐI TƯỢNG TRONG ADO.NET

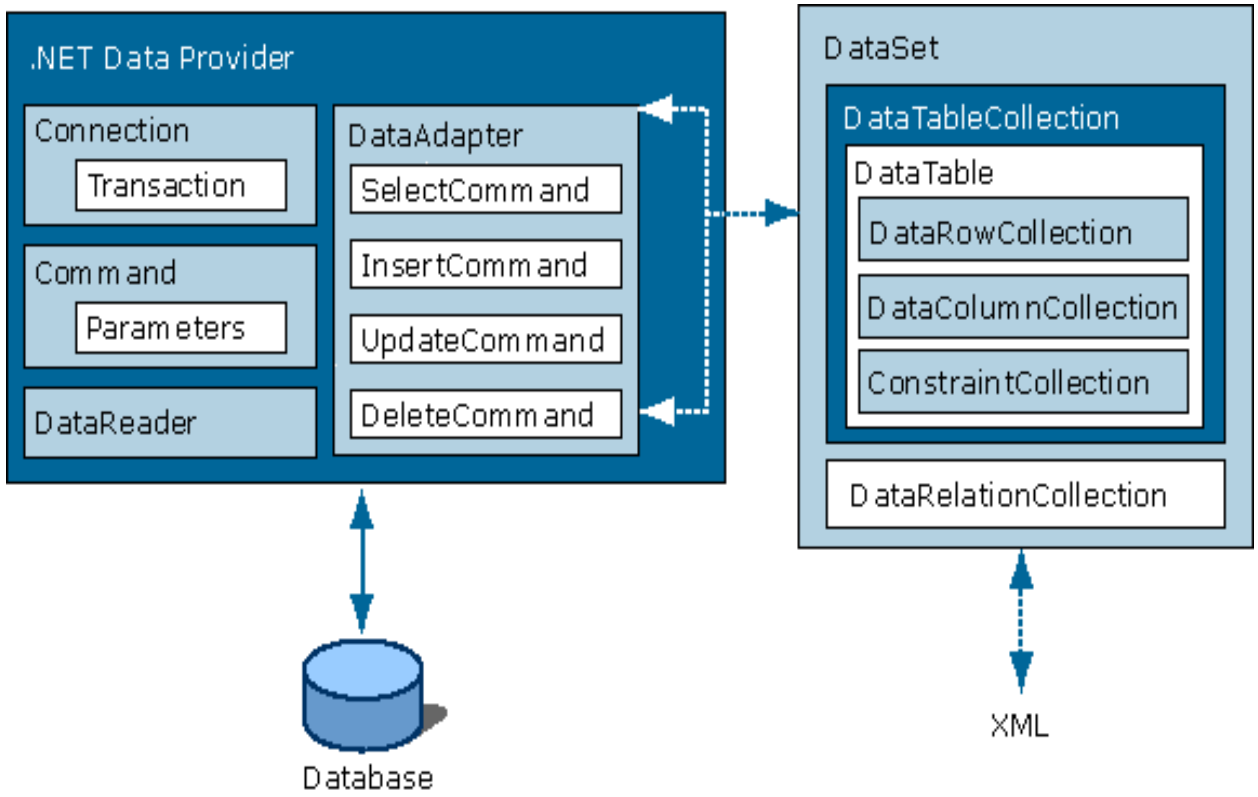
1. Giới thiệu ADO.NET

Trong thực tế, có rất nhiều ứng dụng cần tương tác với cơ sở dữ liệu .NET Framework cung cấp một tập các đối tượng cho phép truy cập vào cơ sở dữ liệu, tập các đối tượng này được gọi chung là ADO.NET.

ADO.NET tương tự với ADO, điểm khác biệt chính ở chỗ ADO.NET là một kiến trúc dữ liệu rời rạc và không kết nối liên tục (Disconnected Data Architecture). Việc kết nối không liên tục đến cơ sở dữ liệu đã đem lại nhiều thuận lợi, trong đó điểm lợi nhất là việc giảm đi một lưu lượng lớn truy cập vào cơ sở dữ liệu cùng một lúc, tiết kiệm đáng kể tài nguyên bộ nhớ. Giảm thiểu đáng kể vấn đề hàng trăm ngàn kết nối cùng truy cập vào cơ sở dữ liệu cùng một lúc.

1.1 Kiến trúc ADO.NET

ADO.NET được chia ra làm hai phần chính rõ rệt, được thể hiện qua hình 1.1



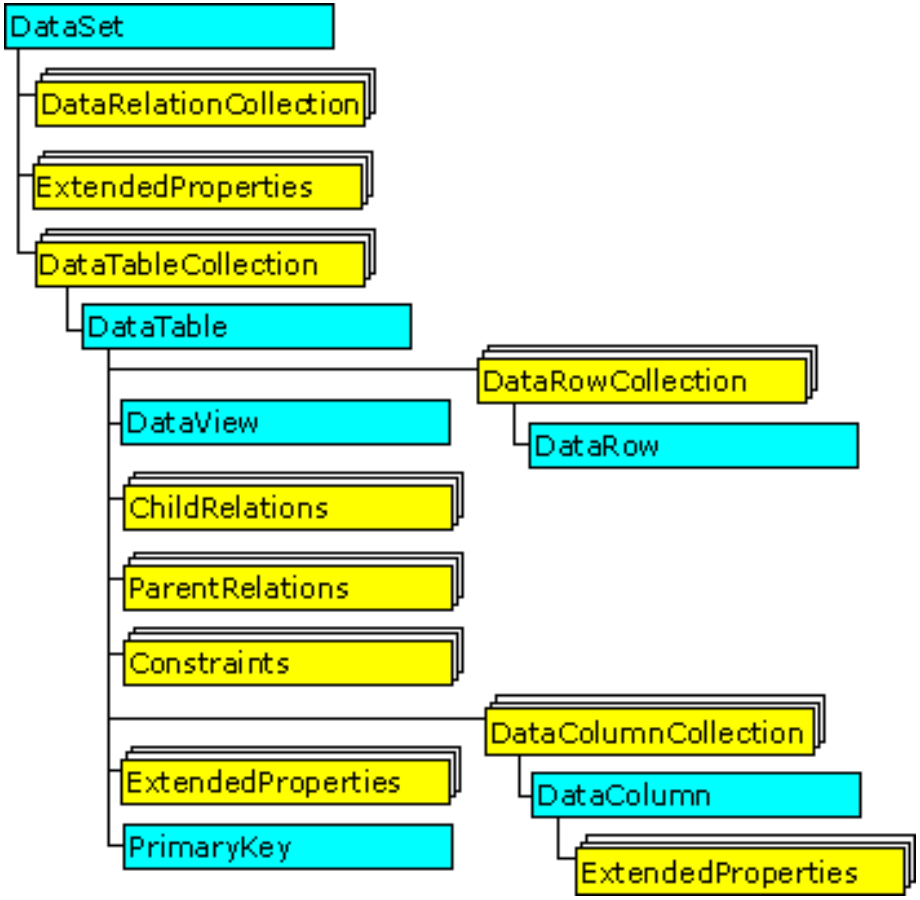
Hình 1.1 Kiến trúc ADO.NET

- **Tập các đối tượng đại diện cho dữ liệu cần xử lý (Content Component)**

Trong ADO.NET các class chính giúp tạo nên các đối tượng đại diện cho dữ liệu cần xử lý bao gồm :

- ❑ DataSet
- ❑ DataTable
- ❑ DataView
- ❑ DataRow
- ❑ DataColumn
- ❑ DataRelation

DataSet là một đối tượng mới, không chỉ là dữ liệu mà DataSet có thể coi như là một bản sao gọn nhẹ của CSDL trong bộ nhớ với nhiều bảng và các mối quan hệ. DataSet hỗ trợ XML thông qua đối tượng XMLDataDocument.



Hình 2.2 Mô hình đối tượng Dataset

Sơ đồ đối tượng trên minh họa các đối tượng trong tập hợp Content Component và sự liên quan giữa các đối tượng đó.

- **Managed provider component**

  Gồm các đối tượng như DataAdapter, DataReader,… giữ nhiệm vụ làm việc trực tiếp với dữ liệu như tập tin (file), Database,…

  DataAdapter là đối tượng kết nối giữa DataSet và CSDL, nó bao gồm 2 đối tượng Connection và Command để cung cấp dữ liệu cho DataSet cũng như cập nhật dữ liệu từ DataSet xuống CSDL.

  DataReader là một đối tượng mới, nó giúp truy cập dữ liệu nhanh chóng, nhưng chỉ được phép đọc và di chuyển tới.

2. Các đối tượng trong ADO.NET

ADO.NET cung cấp đầy đủ các chức năng truy xuất CSDL chung cho các ngôn ngữ lập trình trong .NET FrameWork. Các đối tượng trong ADO.NET đều bắt nguồn từ tên miền :

SysTem.Data

SysTem.Data.OleDb

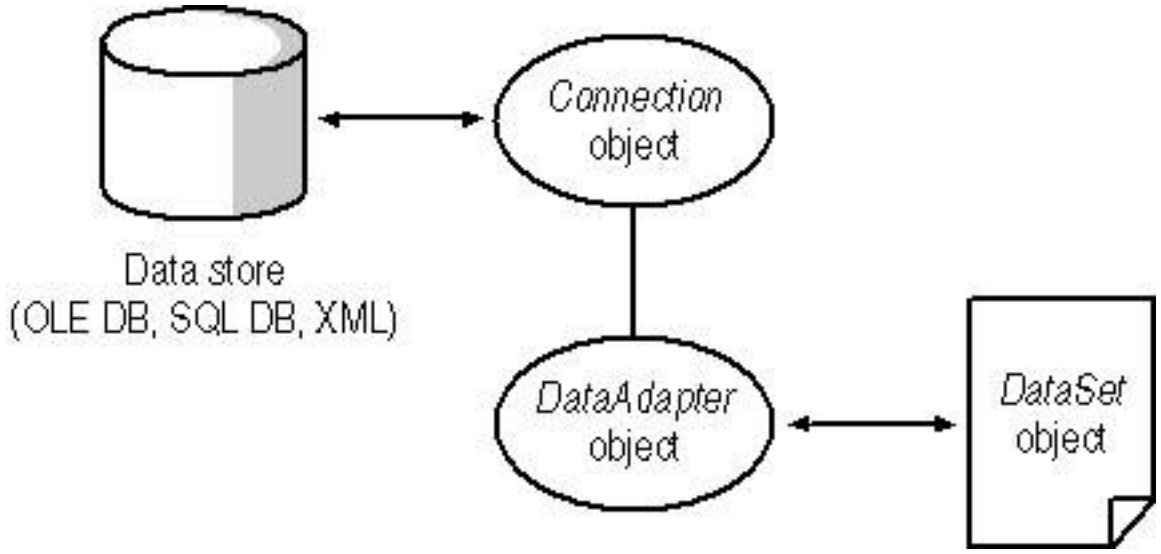
SysTem.Data.SqlClient

Tên miền SysTem.Data.OleDb chứa các lớp được thiết kế để làm việc với bất kỳ nguồn dữ liệu nào, ngược lại SysTem.Data.SqlClient gồm các lớp được thiết kế tối ưu cho CSDL của SQL Server 2000.

Để sử dụng chức năng của các đối tượng này, chúng ta phải tham chiếu đến tên miền SysTem.Data

2.1 Đối tượng Connection

Vai trò của Connection trong ADO.NET cũng như trong ADO là tạo kết nối giữa ứng dụng và nguồn dữ liệu. Điểm khác biệt chính là ADO.NET không còn hỗ trợ các thao tác dữ liệu có kết nối, Connection trong ADO.NET chỉ đóng vai trò tạo kênh thông tin giữa ứng dụng và CSDL, không thực hiện các lệnh truy xuất, cập nhật dữ liệu như trong ADO.



Hình 2.3 Connection và các đối tượng lệnh

2.1.1 Data Provider

Các Data Provider có sẵn trong ADO.NET là :

SysTem.Data.OleDb

SysTem.Data.SqlClient

Ứng với mỗi tên miền, chúng ta có một Connection tương ứng :

SysTem.Data.OleDb.OleDbConnection

SysTem.Data.SqlClient.SqlConection

2.1.2 ConnectionString

Trước khi thực hiện kết nối, cần khai báo các thông tin cần thiết cho Connection thông qua thuộc tính ConnectionString. Cách khai báo tùy theo Data Provider, trong phạm vi môn học này, do dữ liệu được lưu trữ trong SQL Server 2000 nên chúng ta sử dụng SqlClient Provider.

SqlConnection bao gồm các thành phần chính như sau :

- Data Source : tên máy nơi cài đặt SQL Server có nguồn dữ liệu muốn kết nối.
- Initial Catalog : tên Database muốn kết nối.
- Integrated Security : True(SSPI)/False chỉ định dùng cơ chế bảo mật của SQL Server (False).
- User ID : tên người dùng (phải khai báo khi Integrated Security =False).
- Password : mật khẩu (phải khai báo khi Integrated Security =False).

Việc sử dụng cơ chế bảo mật nào tùy theo SQL Server qui định.

using SysTem.Data.SqlClient

string Chuoi_ket_noi =“Data Source=

<địa chỉ TCP/IP hay tên máy>

;Initial Catalog= ***<tên Database>***

; User ID= ***<tên người dùng>***

; Password= ***<mật khẩu>*** **"**

hoặc

using SysTem.Data.SqlClient

string Chuoi_ket_noi =“Data Source=
<địa chỉ TCP/IP hay tên máy>
;Initial Catalog= <tên Database>
; Integrated Secrity=SSPI ”

SqlConnection Ket_noi = New SqlConnection(Chuoi_ket_noi);

Các phương thức trên đối tượng Connection

Phương thức	Mô tả
Close	Đóng kết nối với nguồn dữ liệu. Sử dụng phương thức này để đóng Connection đang mở.
Dispose	Xóa mọi tài nguyên liên quan đến Connection trên vùng nhớ
Open	Thực hiện kết nối với các thông tin đã khai báo trong ConnectionString.

2.2 DataAdapter

Để lấy dữ liệu từ nguồn dữ liệu về cho DataSet chúng ta sử dụng một đối tượng gọi là DataAdapter. Đối tượng này cho phép chúng ta lấy cấu trúc và dữ liệu của các bảng trong nguồn dữ liệu về cho DataSet. Lúc này chúng ta có thể thiết lập quan hệ cho các bảng để có được một mô hình CSDL thu gọn theo yêu cầu của ứng dụng . Chúng ta có thể thao tác di chuyển, thêm, sửa hủy trên các bảng của DataSet và cập nhật các thay đổi đó về nguồn dữ liệu thông qua đối tượng DataAdapter.

DataAdapter là một bộ gồm 4 đối tượng Command

• SelectCommand : cho phép lấy thông tin từ nguồn dữ liệu về.

• InsertCommand : cho phép chèn thêm dữ liệu (dòng) vào bảng trong nguồn dữ liệu.

• UpdateCommand : cho phép sửa đổi dữ liệu trên bảng trong nguồn dữ liệu.

• DeleteCommand : cho phép hũy bỏ dữ liệu trên bảng trong nguồn dữ liệu.

Chúng ta chỉ cần chỉ rõ nội dung lệnh truy vấn cho SelectCommand, nội dung các đối tượng còn lại có thể sử dụng đối tượng thích hợp để tự động phát sinh hoặc chỉ rõ lệnh cho từng đối tượng.

2.2.1 Tạo một đối tượng DataAdapter

Cũng như Command, chúng ta cần khai báo rõ DataAdapter sử dụng theo Data Provider SqlDataAdapter. Lớp này thuộc tên miền :

SysTem.Data.SqlClient.SqlDataAdapter

SqlDataAdapter có những dạng hàm constructor nạp chồng như sau:

Dạng 1 : không có đối số, cho khởi gán một thể hiện mới lớp SqlDataAdapter.

public SqlDataAdapter();

SqlDataAdapter ada=new SqlDataAdapter();

Dạng 2: có một đối số, cho khởi gán một thể hiện mới lớp SqlDataAdapter với sqlCommand được khai báo như một thuộc tính.

public SqlDataAdapter(SqlCommand selectCommand);

Ví dụ :

string strSQL="select makh, tenkh from khachhang"

SqlCommand cmd=new SqlCommand(strSQL,myconnect);

SqlDataAdapter ada=new SqlDataAdapter(cmd);

Dạng 3: có hai đối số, cho khởi gán một thể hiện mới lớp SqlDataAdapter với string là một selectCommand và một đối tượng SqlConnection.

public SqlDataAdapter(string *selectCommandText,*

SqlConnection *selectConnection***);**

Ví dụ :

string strSQL=“select makh, tenkh from khachhang”;

SqlDataAdapter ada=new SqlDataAdapter(strSQL,conn);

Dạng 4: có hai đối số, cho khởi gán một thể hiện mới lớp SqlDataAdapter với string thứ nhất là một selectCommand và một string thứ hai là một chuỗi connectionstring.

public SqlDataAdapter(string *selectCommandText,*

string *selectConnectionstring***);**

Ví dụ :

```
string strconn="Intergrated Security=SSPI; Initial Catalog=
                                    QLBH; Data Source=(local)";
string strSQL="select makh, tenkh from khachhang";
SqlDataAdapter ada=new SqlDataAdapter(strSQL,strconn);
```

Các thuộc tính của lớp DataAdapter

Các thuộc tính	Mô tả
AcceptChanges DuringFill	Thuộc tính read-write này trả về hoặc đặt một trị bool cho biết liệu xem hàm AcceptChanges có được gọi hay không đối với DataRow sau khi nó được thêm vào DataTable. Trị mặc định là true.
ContinueUpdate OnError	Thuộc tính read-write này trả về hoặc đặt một trị bool cho biết liệu xem cập nhật có tiếp tục hay không khi một sai lầm xảy ra trong khi cập nhật dữ liệu. Trị mặc định là false.

Các thuộc tính	Mô tả
DeleteCommand	Thuộc tính read-write này trả về hoặc đặt lệnh Delete, hoặc Store Procedure dùng để loại bỏ các mẫu tin ra khỏi dữ liệu nguồn. Các hàng dữ liệu tương ứng trong DataSet sẽ bị gỡ bỏ khỏi dữ liệu nguồn khi UpDate command được gọi trên DataAdapter.
InsertCommand	Tượng trưng cho một lệnh INSERT hoặc Store Procedure dùng chèn một mẫu tin mới vào dữ liệu nguồn. Các hàng dữ liệu tương ứng được chèn vào trong DataSet cũng sẽ được chèn vào dữ liệu nguồn khi UpDate command được gọi trên DataAdapter.
MissingMapping Action	Thuộc tính read-write này trả về hoặc đặt một trị cho biết liệu xem những bảng dữ liệu nguồn hoặc cột dữ liệu nguồn không được ánh xạ phải được parse với tên dữ liệu nguồn hay không hay phải tung ra một biệt lệ. Đây có nghĩa là thuộc tính này sẽ quyết định hành động phải lấy khi không có ánh xạ đối với một bảng dữ liệu hoặc cột nguồn.

Các thuộc tính	Mô tả
MissingMapping Schema	Thuộc tính read-write này trả về hoặc đặt một trị cho biết liệu xem những thiếu sót về bảng dữ liệu nguồn, cột dữ liệu nguồn và mối quan hệ phải được thêm vào schema trong DataSet hay không hay phải bỏ qua hoặc tung một biệt lệ. Đây có nghĩa là thuộc tính này sẽ quyết định hành động phải lấy khi thiếu sót về bảng dữ liệu hoặc cột nguồn và mối quan hệ trên dữ liệu nguồn từ DataSet.
SelectCommand	Tượng trưng cho một lệnh Select hay một Store Procedure dùng chọn ra những mẫu tin từ dữ liệu nguồn sử dụng phương thức Fill của DataAdapter.

Các thuộc tính	Mô tả
UpDateCommand	Tượng trưng cho một lệnh UpDate hoặc một Store Procedure dùng cập nhật các mẫu tin trên một dữ liệu nguồn. Các hàng dữ liệu tương ứng được cập nhật trong DataSet cũng được cập nhật vào dữ liệu nguồn khi UpDataCommand được gọi trên DataAdapter
TableMappings	Tượng trưng cho một collection những ánh xạ giữa bảng dữ liệu hiện thời với một đối tượng DataTable. Trị trả về là một collection các đối tượng DataTableMapping.

Các phương thức của Lớp DataAdapter

Các phương thức	Mô tả
Fill	Phương thức này cho điền mẫu tin dữ liệu từ DataAdapter lên một đối tượng DataSet
GetFillParameters	Overloaded. Phương thức này cho tìm lại những tham số được sử dụng khi một lệnh SELECT được thi hành. Trị trả về phải được trữ trong một dãy kiểu IDataParameter.
UpDate	Overloaded. Phương thức này cho trữ dữ liệu tử một DataSet lên dữ liệu nguồn. Ở đây có nghĩa là đối tượng Command được gắn liền với InsertCommand, UpDateCommand và DeleteCommand sẽ được thi hành được thêm, sửa và xóa.

Các sự kiện của Lớp DataAdapter

Các sự kiện	Mô tả
FillError	Sự kiện này được kích hoạt bất cứ khi nào khi có lỗi xảy ra trong thời gian tác vụ Fill được thi hành.
RowUpdate	Sự kiện này được kích hoạt sau khi một lệnh cập nhật được thi hành đối với dữ liệu nguồn. Sự kiện này được gọi khi bạn có thay đổi trong DataSet hoặc DataTable khi bạn gọi phương thức Update của DataAdapter.
RowUpdating	Sự kiện này được kích hoạt trước khi một lệnh cập nhật được thi hành đối với dữ liệu nguồn. Sự kiện này được gọi khi bạn có thay đổi trong DataSet hoặc DataTable khi bạn gọi phương thức Update của DataAdapter. Bạn có thể dùng sự kiện này phối hợp với sự kiện RowUpdate để theo dõi một tác vụ “cập nhật”

2.2.2 Các chức năng DataAdapter

a. Lấy dữ liệu từ nguồn

Sau khi có đối tượng DataAdapter, chúng ta có thể sử dụng đối tượng này để lấy dữ liệu về cho các đối tượng chứa dữ liệu như DataSet, DataTable qua phương thức Fill.

• Đỗ dữ liệu vào DataSet có sẵn. Dữ liệu được lấy về DataSet dưới dạng các DataTable, với tên mặc định là Table, Table1, Table2,…

Fill(<dataset>)

• Đỗ dữ liệu vào DataTable có sẵn.

Fill(<datatable>)

• Đỗ dữ liệu vào DataSet cho bảng <tên DataTable>; nếu chưa có bảng sẽ được tạo. Phương thức trả về số mẫu tin lấy về được.

Fill(<dataset>,<tên datatable>)

b. Lấy cấu trúc dữ liệu từ nguồn

- Đỗ cấu trúc dữ liệu vào DataSet có sẵn, phương thức trả về một tập hợp các bảng được thêm vào DataSet.

  FillSchema(<dataset>,<kiểu cấu trúc>)

- Đỗ cấu trúc dữ liệu vào DataTable có sẵn, phương thức trả về DataTable.

  FillSchema(<datatable>,<kiểu cấu trúc>)

- Đỗ cấu trúc dữ liệu vào DataSet cho bảng DataTable; nếu chưa có bảng sẽ được tạo ra.

  FillSchema(<dataset>,<kiểu cấu trúc>,<tên datatable>)

<kiểu cấu trúc> : qui định việc ánh xạ tên có được chấp nhận hay không. Sau đây là các giá trị của kiểu cấu trúc :

Giá trị	Mô tả
SchemaType.Mapped	Sử dụng các TableMappings cho các cấu trúc đưa vào DataSet nếu trùng hợp
SchemaType.Source	Không sử dụng các TableMappings

Thông thường SchemaType.Mapped được sử dụng.

Nếu bảng đã có cấu trúc sẵn, các cột có tên khác với cột có sẵn trên bảng sẽ được thêm vào.

Tùy theo cấu trúc trên dữ liệu nguồn, những thuộc tính sau sẽ được đưa vào cấu trúc của cột :

- AllowDBNull : cho phép nhận giá trị null hay không
- AutoIncrement : tự động tăng
- MaxLength : kích thước tối đa của cột (tính theo Byte)
- ReadOnly : chỉ cho phép đọc
- Unique : cột có trị duy nhất

c. Tạo bộ lệnh cập nhật cho DataAdapter

Dựa vào nội dung lệnh truy xuất của DataAdapter, chúng ta có thể sử dụng một đối tượng để tự động tạo các lệnh còn lại, đó là CommandBuilder thuộc tên miền :

SysTem.Data.SqlClient.SqlCommandBuilder

Cách thực hiện như sau :

SqlCommandBuilder cb=new <loại CommandBuilder>(<đối tượng DataAdapter>

d. cập nhật dữ liệu về nguồn

• Cập nhật các thay đổi trên <dataset> vào nguồn dữ liệu (<dataset> : đối tượng DataSet mà DataAdapter sẽ cập nhật)

Update(<dataset>)

• Cập nhật các thay đổi trên <datatable> vào nguồn dữ liệu (<datatable> : đối tượng DataTable mà DataAdapter sẽ cập nhật)

Update(<datatable>)

• Cập nhật các thay đổi trên bảng có tên <tên bảng> trong DataSet vào nguồn dữ liệu

Update(<dataset>, <tên bảng>)

e. Ví dụ : minh họa kết nối dữ liệu

Để sử dụng các đối tượng truy xuất dữ liệu trong mã lệnh, hãy làm theo các bước sau:

1. Tạo đối tượng data connection.

2. Tạo đối tượng data adapter.

3. Tạo đối tượng data set.

4. Gọi các phương thức của đối tượng adapter để nạp hay cập nhật data set.

5. Sử dụng data binding hay các kỹ thuật khác để hiển thị dữ liệu từ data set.

Windows Form dưới đây sẽ dùng một DataGridView để lấy dữ liệu từ bảng Sanpham trong cơ sở dữ liệu QLBH.

Các dòng mã chính :

- **Tạo ra chuỗi kết nối vào cơ sở dữ liệu**

```
string connectionString = "server=(local); uid=sa; pwd=;
                database=QLBH; Integrated Security=True;";
```

- **Tạo câu lênh truy vấn chọn dữ liệu**

```
string commandString = "Select Masp, Tensp, dvt, dongia
                                        from Sanpham";
```

- **Tạo đối tượng DataAdapter và chuyển cho nó chuỗi truy vấn và kết nối**

```
SqlDataAdapter DataAdapter = new SqlDataAdapter
                                (commandString, connectionString);
```

- **Tạo đối tượng DataSet mới**

```
DataSet dst_sp = new DataSet( );
```

- **Đẩy bảng dữ liệu Sanpham lấy từ DataAdapter vào dataSet**

```
DataAdapter.Fill(dst_sp,”Sanpham");
```

- **Hiển thị dữ liệu trong dataSet trên DataGridView**

```
DataGridWiew_SP.Databindings=dst_sp;
```

2.3 DataTable

Dữ liệu các bảng trong nguồn dữ liệu được lấy về và đưa vào các DataTable. DataTable thuộc tên miền :

SysTem.Data.DataTable

Có các cách khai báo như sau :

```
DataTable Bang_x =new DataTable()
```

Hoặc

```
DataTable Bang_x =new DataTable(<tên bảng>)
```

DataTable hình thành từ DataColumn, DataRow mà chúng ta sẽ lần lượt tìm hiểu sau.

- **Các thuộc tính của DataTable**

Giá trị	Mô tả
ChildRelations	Trả về tập hợp những quan hệ trong đó bảng đóng vai trò bảng cha (bảng một)
Columns	Trả về tập hợp các cột trong bảng (thuộc lớp DataColumn)
Constraints	Trả về tập hợp các ràng buộc trong bảng
DataSet	Trả về DataSet chứa bảng
DefaultView	Trả về DataView phát sinh từ bảng
ParentRelations	Trả về tập hợp những quan hệ trong đó bảng đóng vai trò bảng con (bảng nhiều)
PrimaryKey	Mảng các cột có chức năng làm khóa chính của bảng
Rows	Trả về tập hợp các dòng dữ liệu của bảng
TableName	Tên của DataTable

2.3.1 Các phương thức của DataTable

a. Để phát sinh một dòng mới có cấu trúc của bảng

NewRow()

Phương thức này sẽ trả về một DataRow mới có cấu trúc như của bảng với các giá trị mặc định nhưng chưa đưa vào tập hợp Rows (nghĩa là chưa thuộc về bảng, tình trạng Detached), và một khi được đưa vào sẽ có tình trạng Added

b. Để xóa tất cả các dòng dữ liệu trên bảng

Clear()

c. Để sao chép cấu trúc, ràng buộc của bảng thành bảng khác

Clone()

Phương thức trả về một DataTable đã có sẵn cấu trúc và ràng buộc của bảng nhưng không có dữ liệu .

d. Để sao chép cấu trúc, ràng buộc và dữ liệu của bảng thành bảng khác

Copy()

Phương thức trả về một DataTable đã có sẵn cấu trúc, ràng buộc và dữ liệu của bảng. Bảng trả về có cùng cấp với bảng-nếu bảng là đối tượng của một lớp kế thừa bảng trả về cũng thuộc lớp đó.

e. Để lấy ra một bản sao những thay đổi trên bảng

GetChanges()

Phương thức trả về một DataTable gồm những dòng dữ liệu đã thay đổi kể từ lần lấy dữ liệu từ nguồn về hoặc từ lần cập nhật trước vào bảng bằng phương thức AcceptChanges.

GetChanges(<tình trạng dòng>)

Cũng như trên nhưng chỉ trả về những dòng có tình trạng đúng với <tình trạng dòng>.

f. Để lấy ra một mảng các dòng bị lỗi trên bảng

GetErrors()

Phương thức trả về một mảng các dòng bị lỗi. Phương thức này được gọi sau khi gọi phương thức GetChanges nhằm phát hiện các dòng bị lỗi để xử lý.

g. Để cập nhật các thay đổi vào bảng

AcceptChanges()

Phương thức cập nhật các thay đổi kể từ lần cập nhật trước hoặc khi bảng được mở vào bảng. Sau khi thực hiện tất cả các dòng đều có tình trạng Unchanged. Những dòng có tình trạng Deleted bị loại bỏ khỏi bảng.

h. Để hủy bỏ các thay đổi của bảng

RejectChanges()

Phương thức phục hồi lại các giá trị kể từ lần cập nhật trước hoặc khi bảng được mở vào bảng. Sau khi thực hiện tất cả các dòngmới thêm vào đều bị loại bỏ. Những dòng có tình trạng Deleted, Modified được phục hồi lại tình trạng gốc.

2.3.2 Tập hợp Rows

Rows là tập hợp các dòng dữ liệu của bảng. Mọi tham chiếu đến dòng đều thông qua tập hợp này. Sau đây là một số chức năng của tập hợp :

- **Lấy số dòng trong tập hợp** : cho biết số dòng dữ liệu trong tập hợp.

  Rows.Count

- **Tham chiếu đến dòng trong tập hợp**

  Rows(<chỉ số dòng>)

- **Truy xuất giá trị của ô**

  Rows(<chỉ số dòng>)(<tên cột>)

  Rows(<chỉ số dòng>)(<chỉ số cột>)

- **Thêm dòng vào bảng**

Thêm một dòng có sẵn vào bảng. Dòng này được tạo ra từ phương thức NewRow của đối tượng DataTable.

DataRow Dong_moi

Dong_moi=<DataTable>.NewRow()

// Tiến hành cập nhật giá trị cho các cột

<DataTable>.Rows.Add(<dòng>)

Tạo dòng mới với các trị trong mảng trị và đưa vào bảng

Rows.Add(<mảng trị>)

- **Xóa bỏ dòng trên bảng**

Xoá <dòng> ra khỏi bảng

Rows.Remove(<dòng>)

Xóa dòng tại vị trí <chỉ số> ra khỏi bảng

Rows.RemoveAt(<chỉ số>)

Xóa toàn bộ các dòng dữ liệu của bảng

Rows.Clear()

2.3.3 Tập hợp Columns

Columns là tập hợp chứa các cột trong cấu trúc của bảng. Mọi tham chiếu đến cột đều thông qua tập hợp này. Sau đây là một số chức năng của tập hợp :

- **Lấy số cột trong tập hợp** : cho biết số cột dữ liệu trong tập hợp.

  Columns.Count

- **Tham chiếu đến cột trong tập hợp**

  Columns.Item(<tên cột>)

  Columns.Item(<chỉ số cột>)

  Columns(<tên cột>)

  Columns(<chỉ số cột>)

Các thuộc tính của DataColumn

Giá trị	Mô tả
AllowDBNull	Thuộc tính cho biết cột có chấp nhận giá trị Null không (đọc ghi)
AutoIncrement	Thuộc tính cho biết giá trị cột có tự động tăng khi thêm dòng mới không (đọc ghi)
AutoIncrementSeed	Giá trị bắt đầu cho cột khi thêm dòng đầu tiên, nếu AutoIncrement là True
AutoIncrementStep	Bước tăng cho dòng thêm mới kế tiếp, , nếu AutoIncrement là True
Caption	Tiêu đề của cột
ColumnName	Tên cột
DataType	Kiểu dữ liệu của cột
DefaultValue	Giá trị mặc định cho cột khi thêm dòng mới
MaxLength	Độ rộng tối đa cho cột kiểu chuỗi
Table	Trả về DataTable chứa cột

2.4 DataSet

Một trong những điểm khác biệt chính giữa ADO và ADO.NET là đối tượng DataSet. DataSet là một mô hình CSDL quan hệ thu nhỏ nhằm đáp ứng các yêu cầu của ứng dụng.

DataSet chứa các bảng (DataTable), các quan hệ (DataRelation) và các ràng buộc (Constraint)

DataSet thuộc miền sau : **SysTem.Data.DataSet**

2.4.1 Các thuộc tính của DataSet

Thuộc tính	Mô tả
Relations	Tập hợp các quan hệ (DataRelation) một nhiều của DataSet (chỉ đọc)
Tables	Tập hợp các bảng (DataTable) của DataSet (chỉ đọc)

2.4.2 Các phương thức của DataSet

a. Thêm bảng vào DataSet

Muốn đưa một DataTable vào DataSet, chúng ta dùng phương thức Add của tập hợp Tables :

- Thêm một bảng mới với tên là <tên bảng>, đưa vào tập hợp Tables

Tables.Add(<tên bảng>)

- Đưa <bảng> vào tập hợp Tables

Tables.Add(<bảng>)

Chú ý : tên bảng trong DataSet có phân biệt chữ HOA chữ thường.

b. Xóa bảng ra khỏi tập hợp Tables của DataSet

Các phương thức sau dùng để xóa bảng ra khỏi tập hợp Tables

- Xóa <bảng> khỏi tập hợp Tables

  Tables.Remove(<bảng>)

- Xóa bảng có tên <tên bảng> khỏi tập hợp Tables

  Tables.Remove(<tên bảng>)

- Xóa bảng có chỉ số là <chỉ số> khỏi tập hợp Tables

  Tables.RemoveAt(<chỉ số>)

- Xóa tất cả các bảng khỏi DataSet

  Tables.Clear()

c. Kiểm tra bảng thuộc về DataSet

Trả về True nếu trong Tables có tên <tên bảng> , ngược lại là False

Tables.Contains(<tên bảng>)

d. Xóa bỏ mọi dữ liệu trên DataSet

Clear()

e. Để cập nhật các thay đổi trên DataSet

AcceptChanges()

Phương thức này của DataSet cập nhật các thay đổi kể từ lúc lấy dữ liệu về hoặc từ lần gọi AcceptChanges trước.

Trên các DataTable, DataRow cũng có phương thức tương ứng. Khi gọi AcceptChanges của DataSet sẽ kéo theo gọi AcceptChanges của DataTable, đến lượt kéo theo gọi AcceptChanges của DataRow.

f. Để hủy bỏ thay đổi trên DataSet

RejectChanges()

Phương thức này của DataSet phục hồi tất cả các thay đổi kể từ lúc lấy dữ liệu về hoặc từ lần gọi AcceptChanges trước.

Khi gọi RejectChanges của DataSet sẽ kéo theo gọi RejectChanges của DataTable, DataRow.

g. Xóa bỏ DataSet

Gọi phương thức Dispose để giải phóng mọi tài nguyên trên vùng nhớ DataSet đang sử dụng.

2.4.3 DataRelation

DataSet quản lý quan hệ giữa các DataTable trong DataSet qua tập hợp Relations. Đây là những đối tượng DataRelation chứa thông tin về mối quan hệ đã tạo ra trong DataSet.

Quan hệ giữa hai hai bảng nói lên sự liên quan của một dòng trên một bảng cha với các dòng trên bảng con, do đó, chúng ta có thể thấy được mối quan hệ đó qua việc hiển thị dữ liệu các dòng tương ứng trên bảng con (một cách tự động qua việc liên kết lưới bảng con với quan hệ đã tạo) khi dòng hiện hành trên bảng cha thay đổi. Ràng buộc, ngược lại, nhằm bảo vệ tính toàn vẹn dữ liệu thông qua mối quan hệ sẽ kiểm tra sự tồn tại các dòng con khi hủy trên bảng cha.

DataRelation thuộc tên miền : **SysTem.Data.DataRelation**

a. Cú pháp khai báo

Có các cách khai báo như sau :

New DataRelation(<tên>,<cột bảng cha>,<cột bảng con>)

New DataRelation(<tên>,<mảng cột bảng cha>,

<mảng cột mảng con>)

Chú ý : chúng ta chỉ thiết lập quan hệ giữa hai bảng chỉ khi hai bảng đó thuộc về một DataSet.

b. Các thuộc tính của DataRelation

Thuộc tính	Mô tả
ChildColumns	Trả về các cột trên bảng con tham gia trong mối quan hệ
ChildTable	Trả về bảng con trong quan hệ
DataSet	Trả về DataSet chứa quan hệ
ParentColumns	Trả về các cột trên bảng cha tham gia trong quan hệ
ParentTable	Trả về bảng cha trong quan hệ
RelationName	Tên quan hệ

c. Minh họa thiết lập quan hệ

Ví dụ sau đây sẽ thiết lập quan hệ giữa hai bảng HOADON, KHACHHANG

```
DataSet QLBH=new DataSet();

QLBH.Tables.AddRange(tbl_khachhang)

QLBH.Tables.AddRange(tbl_hoadon)
```

Có thể sử dụng một trong các cách sau :

```
DataRelation Quan_he=new DataRelation("kh_hd",
                                      tbl_khachhang.Columns("Makh"),
                                      tbl_hoadon.Columns("Makh"))

QLBH.Relations.Add(Quan_he)
```

Hoặc

```
DataRelationQLBH.Relations.Add((“kh_hd”,
                                tbl_khachhang.Columns(“Makh”),
                                tbl_hoadon.Columns(“Makh”))
```

d. Bổ sung cột vào bảng sau khi thiết lập quan hệ

Để tạo thêm cột tính toán dựa trên các cột đã có, chúng ta có thể tạo thêm cột mới và sử dụng thuộc tính Expression để tính toán số liệu cho cột mới tạo này.

Thuộc tính Expression cho phép :

• Tính toán dữ liệu từ các cột có sẵn trên DataTable

Ví dụ: Tbl_nhanvien có cột gioitinh kiểu bit, chúng ta tạo thêm cột Phai để hiển thị "Nam/Nữ"

tbl_nhanvien.Columns.Add("Phai",

Type.GetType("SysTem.String"),"IIF(gioitinh,'Nam','Nữ')");

• Lấy dữ liệu của các bảng trong cùng DataSet thông qua quan hệ

Ví dụ : trong tbl_hoadon, tạo thêm cột ten_kh dựa trên quan hệ được xây dựng trên bảng tbl_khachhang và tbl_hoadon có tên kh_hd trong DataSet.

Tbl_hoadon.Columns.Add("Ten_kh",

Type.GetType("SysTem.String"),"Parent(kh_hd).ten_kh")

Ví dụ : trong tbl_khachhang, tạo thêm cột So_sanh dựa trên quan hệ được xây dựng trên bảng tbl_khachhang và tbl_hoadon có tên kh_hd trong DataSet, cho biết khách hàng đã mua hàng bao nhiêu lần rồi.

Tbl_khachhang.Columns.Add("So_sanh",_

Type.GetType(SysTem.Integer"),"count(Child(kh_hd).makh"))

2.5 DataView

Với chức năng hiển thị dữ liệu tùy biến, DataView cho phép hiển thị dữ liệu của một DataTable qua các điều khiển khác nhau như dùng một điều khiển hiển thị tất cả các dòng dữ liệu trên DataTable và một điều khiển khác hiển thị dữ liệu của những dòng bị đánh dấu hủy trên DataTable đó.

Mỗi DataTable có một view mặc định thông qua thuộc tính DefaultView.

DataView thuộc tên miền SysTem.Data.DataView

2.5.1 Tạo DataView

Tạo một đối tượng DataView mới

New DataView()

Tạo một đối tượng DataView mới từ <bảng>

New DataView(<bảng>)

Tạo một DataView mới từ <bảng> gồm những dòng thoả điều kiện <biểu thức lọc> và có tình trạng như <tình trạng dòng>, được hiển thị theo cách sắp xếp <biểu thức sắp xếp>

New DataView(<bảng>,<biểu thức lọc>,_

<biểu thức sắp xếp>,<tình trạng dòng>)

2.5.2 Các thuộc tính của DataView

Thuộc tính	Mô tả
AllowDelete	Giá trị cho biết thao tác xóa dòng trên DataView có được phép không (đọc ghi)
AllowEdit	Giá trị cho biết thao tác sửa đổi trên DataView có được phép không (đọc ghi)
AllowNew	Giá trị cho biết thao tác thêm mới trên DataView với phương thức AddNew có được phép không (đọc ghi)
Count	Số mẫu tin trên DataView sau khi áp dụng thuộc tính RowFilter và RowStartedFilter
Item	Trả về một dòng dữ liệu trên bảng theo tham số truyền vào
Rowfilter	Biểu thức lọc của DataView để thay đổi cách hiển thị dữ liệu

Thuộc tính	Mô tả
RowStateFilter	Giá trị cho biết trạng thái dòng dữ liệu đang hiển thị trên DataView gồm các trị sau : • Added : dòng thêm mới nhưng chưa cập nhật • CurrentRows : bao gồm các dòng không thay đổi, dòng mới và dòng đã thay đổi • Deleted : dòng đánh dấu hủy • ModifiedCurrent : dòng đã được thay đổi và giá trị hiển thị là phiên bản hiện hành, tức phiên bản đã sửa đổi từ dữ liệu gốc • ModifiedOriginal : dòng đã được thay đổi và giá trị hiển thị là phiên bản gốc • None : không gì cả • OriginalRows :dòng gốc bao gồm các dòng không thay đổi và dòng đã đánh dấu hủy • Unchanged : dòng không thay đổi

Thuộc tính	Mô tả
Sort	Tên cột hoặc các cột của DataTabledùng sắp xếp dữ liệu trên DataView ngăn cách nhau bằng dấu phẩy (đọc ghi). Sắp xếp tăng ghi ASC (hoặc không ghi), sắp xếp giảm ghi DESC
Table	Tên bảng (đọc ghi)

Mặc định DataView là chỉ đọc , tuy nhiên chúng ta có thể thay đổi các thuộc tính AllowNew, AllowEdit và AllowDelete để thao tác trên dữ liệu của bảng tương ứng.

2.6 Đối tượng Command

Đối tượng Command trên .NET cho phép bạn thi hành trực tiếp những câu lệnh SQL, chẳng hạn như INSERT, SELECT, UPDATE và DELETE lên dữ liệu nguồn để thêm, chọn, cập nhật và xóa dữ liệu trên CSDL. Bạn cũng có thể dùng đối tượng command để thi hành những stored procedure trong CSDL.

Lớp SqlCommand tượng trưng cho những đối Command trên Data Provider SQL.

- **Các thuộc tính của lớp Command**

Các thuộc tính	Mô tả
CommandText	Có thể là một câu lệnh SQL, tên một Store Procedure hoặc tên một bảng tùy vào trị của CommandType.
CommandTimeout	Đây là thời gian (tính theo giây) chờ thi hành Command này, mặc định là 30 giây. Nếu Command không được thi hành trong khoảng thời gian này thì một biệt lệ sẽ được tung ra.
CommandType	Thuộc tính này cho biết làm thế nào để phân biệt nột dung của CommandText là Text, StoredProcedure hoặc TableDirect…
Connection	Đạy là connection nối kết cơ sở dữ liệu. Chẳng hạn SqlConnection

Các thuộc tính	Mô tả
DesignTimeVisible	Thuộc tính này dùng để cho biết liệu đối tượng Command có nên thực hiện lên trên một ô control Windows Form Designer hay không. Trị mặc định là False
Parameters	Thuộc tính này cho tìm lại collection (SqlParsmeter Collection) các thông số của thuộc tính CommandText
Transaction	Thuộc tính này được dùng để tìm lại hoặc đặt giao dịch mà command đang thi hành. Bạn phải để ý là đối tượng transaction cũng được kết nối về cùng một đối tượng connection như đối tượng command.
UpdatedRowsource	Thuộc tính này lấy hoặc đặt, để kết quả thi hành câu lệnh Command này sẽ áp dụng đối với đối tượng DataRow khi nó được sử dụng bởi hàm Update của DbDataAdapter. Thuộc tính này sẽ mang một trong những giá trị của enumeration

Ví dụ : Sử dụng SqlCommand để đọc dữ liệu từ CSDL

```
//ket noi CSDL
 public SqlConnection conn;
 connectionStr = "Data Source=.;Initial Catalog=QLBH;Integrated
                                                    Security=True;";
 conn = new SqlConnection(connectionStr);
 conn.Open();

//Tao doi tuong Command
string str = "Select manv, hotennv from nhanvien";
SqlCommand cmd = new SqlCommand(str,conn);
```

- **Các phương thức của lớp COMMAND**

Để gọi thi hành một Command bạn sử dụng các phương thức của lớp đối tượng Command như sau :

Các phương thức	Mô tả
ExcecuteNonQuery	Cho thi hành một câu lệnh SQL đối với một Connection và trả về số mẫu tin bị ảnh hưởng. Chỉ dùng để thi hành câu lệnh không trả về mẫu tin nào cả, chẳng hạn lệnh DELETE.
ExecuteReader	Overloaded. Hàm này được dùng khi bạn muốn thi hành một câu lệnh có trả về các mẫu tin, chẳng hạn như lệnh SELECT. Các mẫu tin sẽ được trả về cho đối tượng SqlCommand. Bạn để ý rằng lớp **DataReader** là một lớp forward-only data nghĩa là chỉ dùng để đọc lại dữ liệu để cho hiển thị cứ không cập nhật được.

Các phương thức	Mô tả
ExecuteScalar	Hàm này cho thi hành một câu truy vấn (query), và trả về cột đầu tiên của mẫu tin đầu tiên của result set được trả về. Nếu có nhiều mẫu tin hoặc nhiều cột thì hàm sẽ phớt lờ. Hàm này chạy nhanh hơn ExecuteReader và ít tốn overhead. Như vây chỉ dùng hàm này khi bạn chỉ muốn một mẫu tin trả về hoặc khi bạn dùng một hàm nhóm như COUNT chẳng hạn.
ExecuteXmlReader	Hàm này cũng tương tự như ExecuteReader, nhưng trả lại các mẫu tin dưới dạng XML.

Lưu ý : bạn không được dùng ExecuteNonQuery để thi hành một câu lệnh SELECT vì ExecuteNonQuery không trả về dữ liệu

2.7 DataReader

Một DataReader sẽ cung cấp một hình thức đọc dữ liệu từ CSDL.DataReader là giải pháp đối với dòng chảy dữ liệu tuôn về phía trước (forward streaming data) thông qua ADO.NET.

Tương tự như các đối tượng ADO.NET các SQL Server .NET data provider cung cấp đối tượng DataReader thông qua lớp SqlDataReader.

DataReader chạy rất hữu hiệu khi nói đến việc sử dụng ký ức, vì chỉ một mẫu tin được đọc trong một lúc, trong khi DataTable thì cấp phát ký ức cho tất cả các mẫu tin được tìm thấy.

- **Khai báo và khởi gán đối tượng DataReader**

Bạn có thể thể hiện một đối tượng DataReader bằng cách dùng hàm ExecuteReader thuộc lớp Command. Khi bạn sử dụng DataReader thì connection được gắn liền với nó và không được làm gì thêm, vì thế bạn phải đóng DataReader lại trước khi dùng connection cho việc khác. Vì thế rất dễ dẫn đến time-out nếu như DataReader khóa chặt các mẫu tin làm cho các truy vấn khác phải chờ đợi.

Ví dụ : Đoạn mã sau gọi hàm ExecuteDataReader của đối tượng Command. Hàm này trả về một thể hiện của lớp DataReader.

```
//goi ham ExecuteReader
 SqlDataReader reader = cmd.ExecuteReader();
```

- **Các thuộc tính của lớp DataReader**

Các thuộc tính	Mô tả
Depth	Thuộc tính read-only này cho biết tầng nấc nằm lồng đối với một hàng dữ liệu XML. Thuộc tính này không hỗ trợ bởi SQL Server Data Provider, có nghĩa là bao giờ nó cũng trả về 0 vì đó là trị depth đối với bảng dữ liệu nằm ngoài nhất.
FieldCount	Thuộc tính read-only này trảvề số cột trên một hàng dữ liệu, và 0 nếu như DataReader không được trả về một hàng dữ liệu hợp lệ.
IsClosed	Thuộc tính read-only này cho biết liệu xem DataReader được đóng lại hay chưa? True là đã đóng.

Các thuộc tính	Mô tả
Item	Đi lấy trị của một cột ở dạng thức native (bẩm sinh). Trên C# thuộc tính này là indexer đối với lớp DataReader.
RecordsAffected	Số hàng dữ liệu bị ảnh hưởng sau một giao tác (insert, updated, deleted). Thuộc tính này chỉ dùng sau khi đã đọc xong tất cả các hàng dữ liệu và DataReader bị đóng lại.

- **Các phương thức của lớp DataReader**

Các phương thức	Mô tả
Close	Cho đóng lại một đối tượng DataReader.
Read	Cho đọc mẫu tin kế tiếp trên data reader. Bạn phải gọi hàm này sau khi đối tượng DataReader được thể hiện. Data reader không canh về hàng đầu tiên theo mặc nhiên. Hàm sẽ trả về **true** nếu như data reader canh về hàng dữ liệu kế tiếp, không thì là false.
NextResult	Cho data reader đi tới kết quả kế tiếp trong một giao tác theo lô. Hàm này không có tác dụng nếu như lệnh truy vấn không trả về kết quả là một lô các hàng dữ liệu hoặc khi kết quả hiện hành là hàng dữ liệu cuối cùng trên result set. Hàm này sẽ trả trị là **true** nếu data reader đi tới có kết quả, bằng không sẽ là false.
Getxxx	Các hàm này đọc một trị, một kiểu dữ liệu từ một cột. Ví dụ : GetChar sẽ trả về trị cho một cột ở dạng ký tự, GetString cho về một chuỗi ký tự.

3 KẾT NỐI ĐẾN CSDL TRONG CHẾ ĐỘ DESIGN

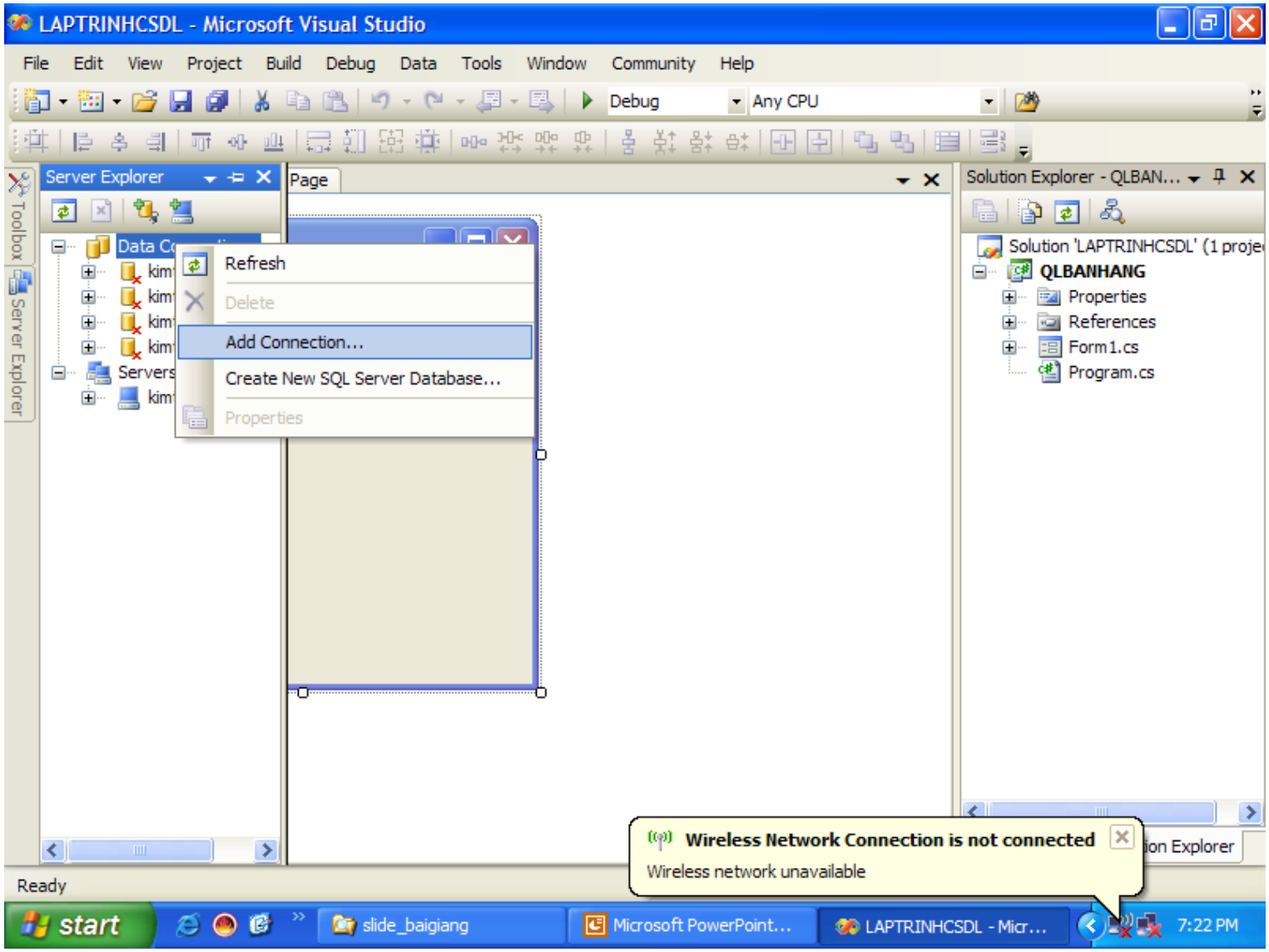
3.1 KẾT NỐI CSDL

Sử dụng Server Explorer để kết nối đến CSDL trong Visual Studio.

Để kết nối đến CSDL trong môi trường thiết kế của Visual Studio, làm theo các bước sau:

Từ View menu, chọn Server Explorer. Visual Studio hiển thị cửa sổ Server Explorer.

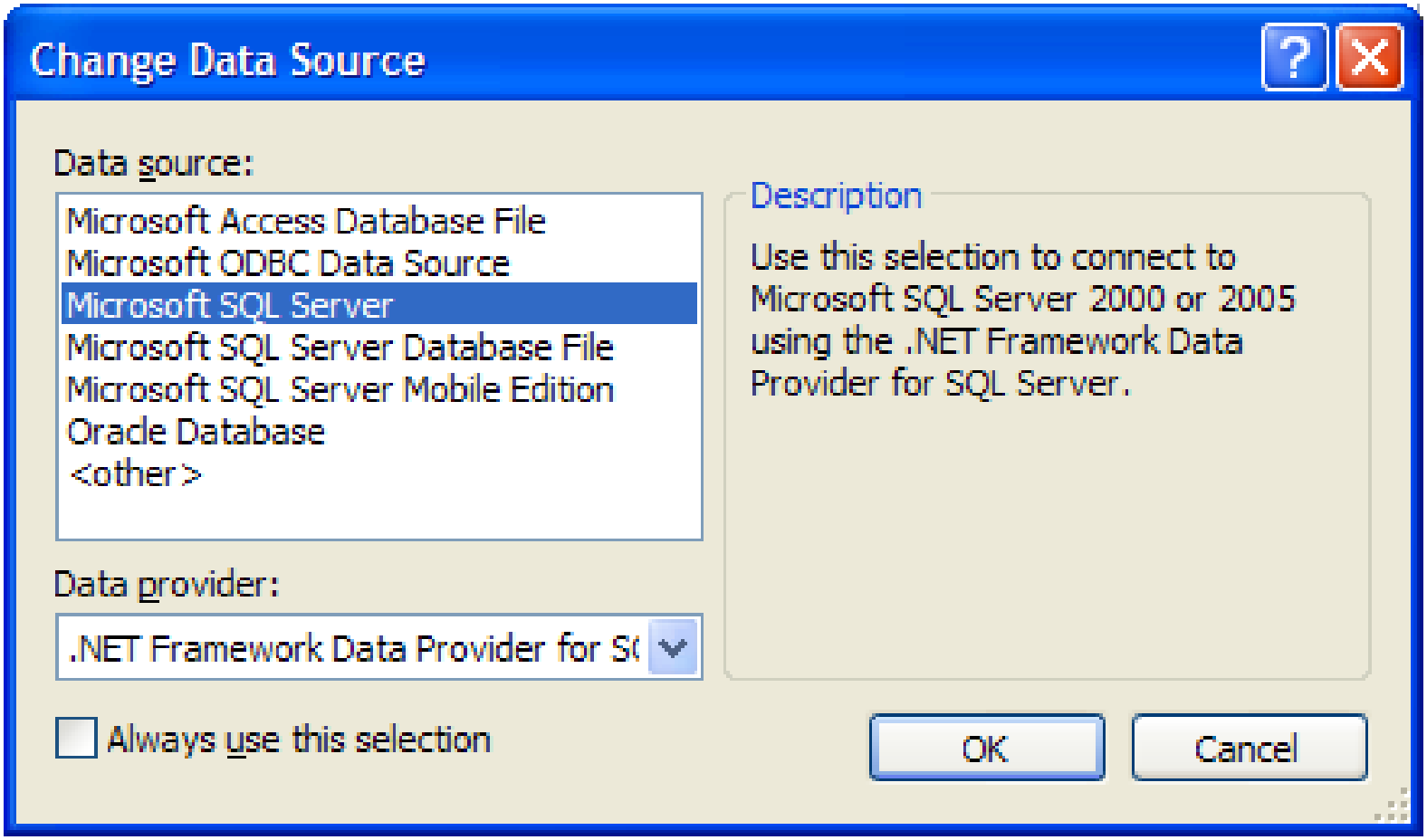
Trong Server Explorer, nhấn Connect To Database. Visual Studio .NET hiển thị hộp thoại Add Connect như trong hình 3.1


LAPTRINHCSDL - Microsoft Visual Studio
File Edit View Project Build Debug Data Tools Window Community Help
Debug
Any CPU
Server Explorer
Toolbox
Server Explorer
Page
Refresh
Delete
Add Connection...
Create New SQL Server Database...
Properties
Solution Explorer - QLBAN...
Solution 'LAPTRINHCSDL' (1 proje
QLBANHANG
Properties
References
Form1.cs
Program.cs
Wireless Network Connection is not connected
Wireless network unavailable
Ready
start
slide_baigiang
Microsoft PowerPoint...
LAPTRINHCSDL - Micr...
7:22 PM

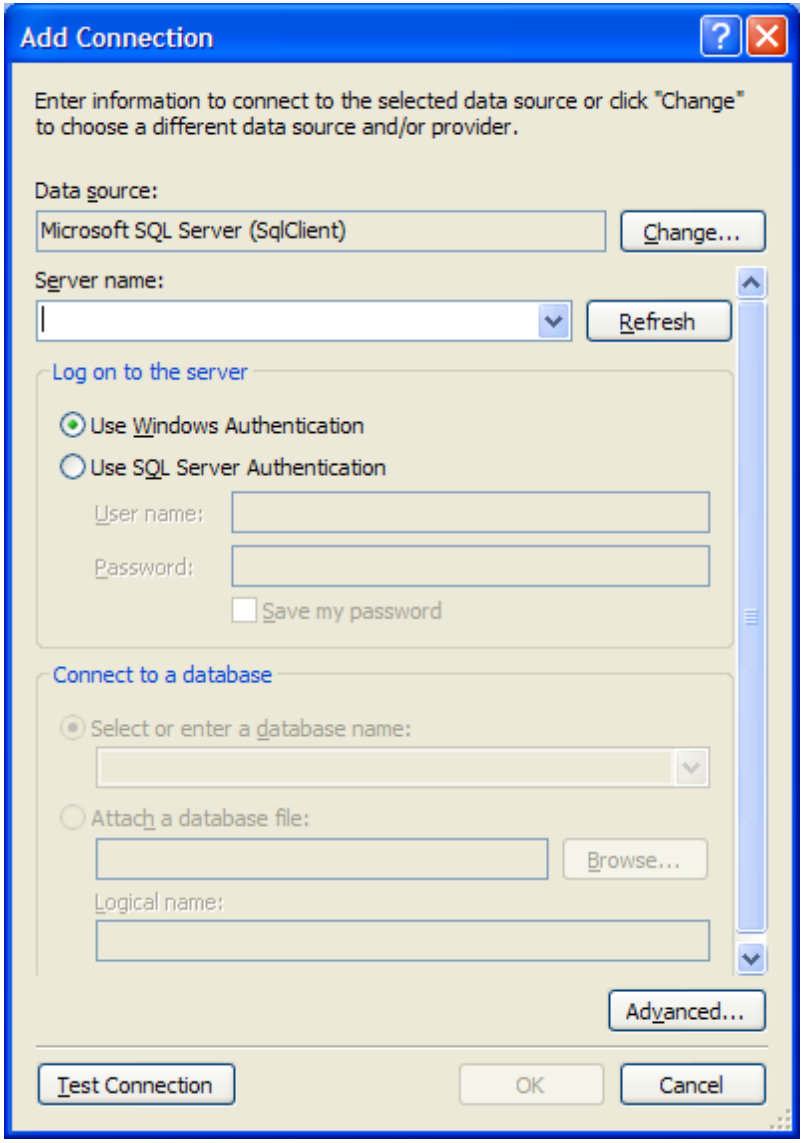
Hình 3.1

3. Nhấn nút Change để chọn kiểu CSDL cho kết nối. Provider theo mặc định là Microsoft SQL Server Provider. Đây là lựa chọn đúng nếu bạn sử dụng CSDL Microsoft SQL Server. Để truy cập các loại CSDL khác, chọn provider tương ứng. Ví dụ như để truy cập CSDL Microsoft Access 2000, chọn Microsoft Access Database File, hình 3.2

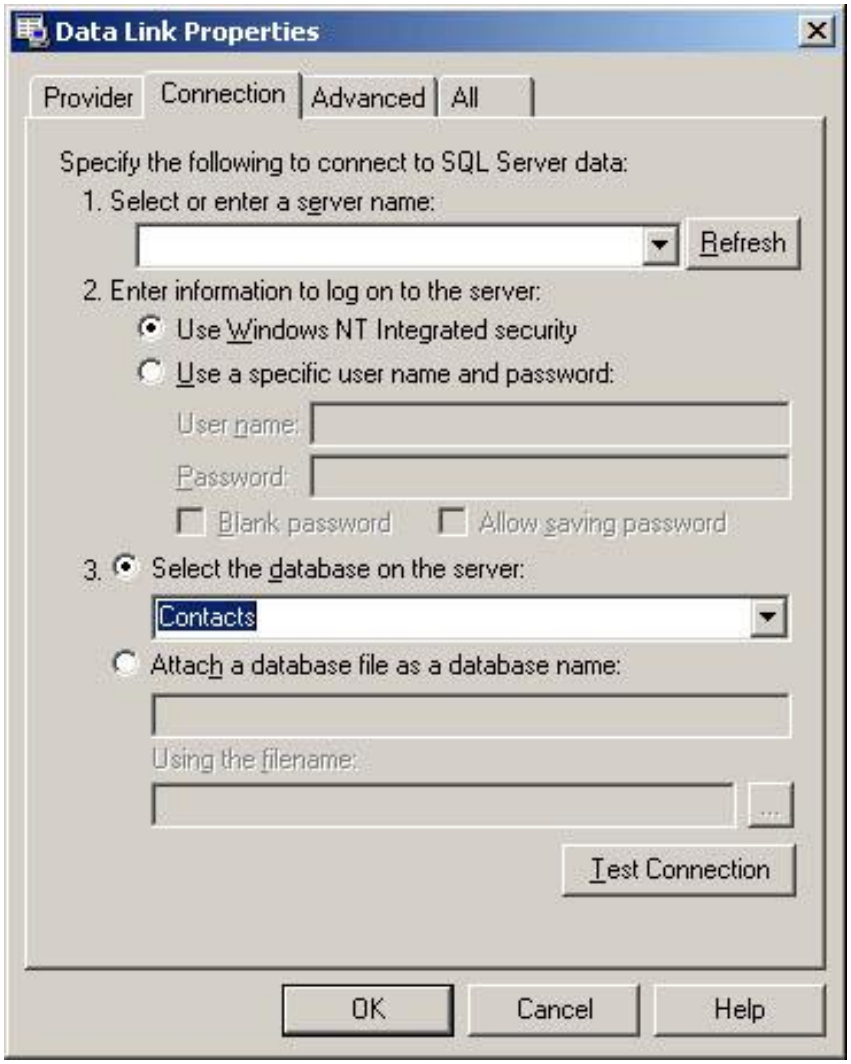
4. Nhấn OK để xác định CSDL cần kết nối. Chọn tên Server muốn kết nối CSDL (hình 3.2), sau đó chọn tên CSDL để kết nối dữ liệu hình 3.3


Change Data Source
Data source:
Microsoft Access Database File
Microsoft ODBC Data Source
Microsoft SQL Server
Microsoft SQL Server Database File
Microsoft SQL Server Mobile Edition
Oracle Database
<other>
Description
Use this selection to connect to Microsoft SQL Server 2000 or 2005 using the .NET Framework Data Provider for SQL Server.
Data provider:
.NET Framework Data Provider for S(
Always use this selection
OK
Cancel

Hình 3.2

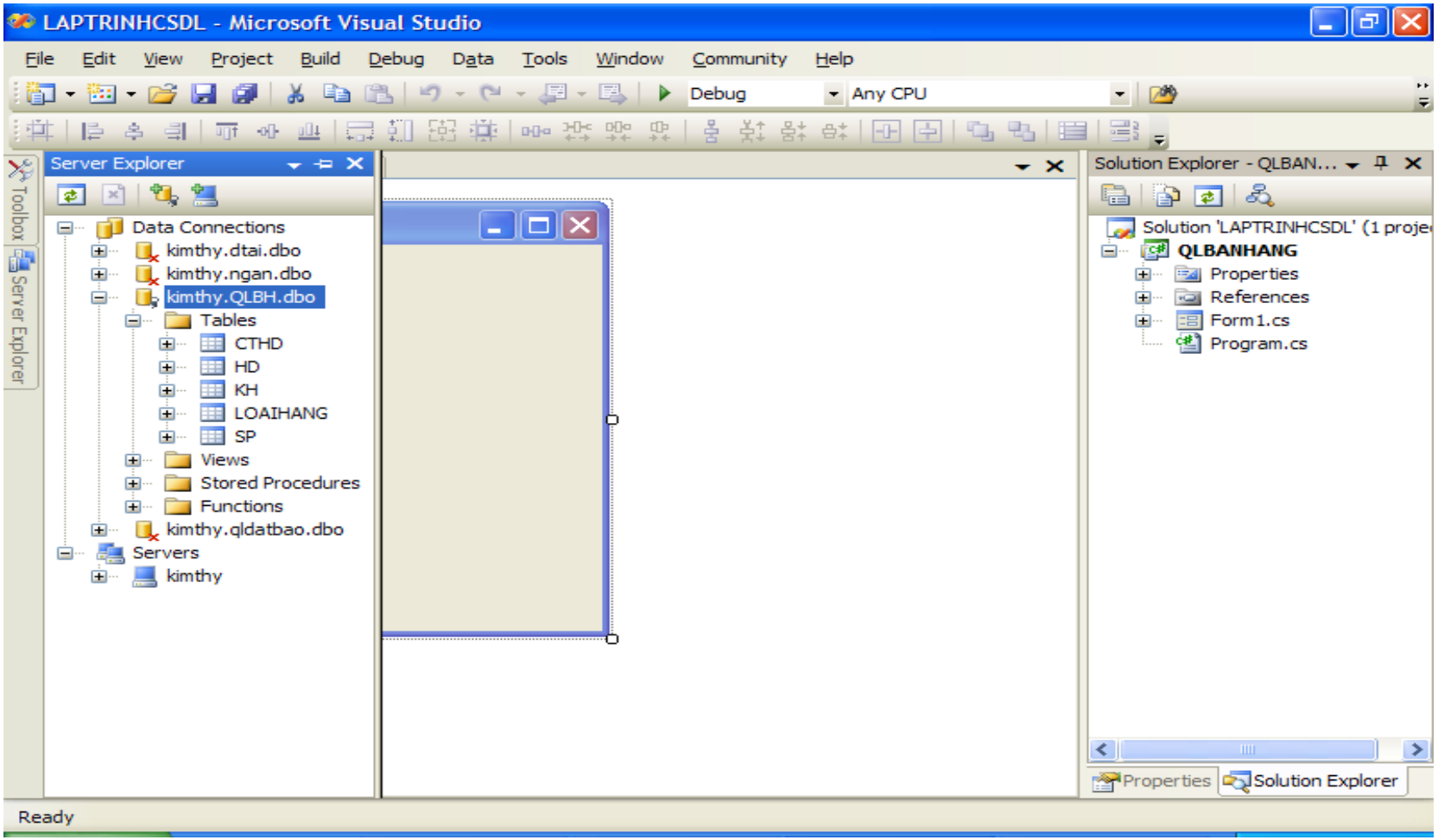

Add Connection
Enter information to connect to the selected data source or click "Change" to choose a different data source and/or provider.
Data source:
Microsoft SQL Server (SqlClient)
Change...
Server name:
Refresh
Log on to the server
Use Windows Authentication
Use SQL Server Authentication
User name:
Password:
Save my password
Connect to a database
Select or enter a database name:
Attach a database file:
Browse...
Logical name:
Advanced...
Test Connection
OK
Cancel

Hình 3.3

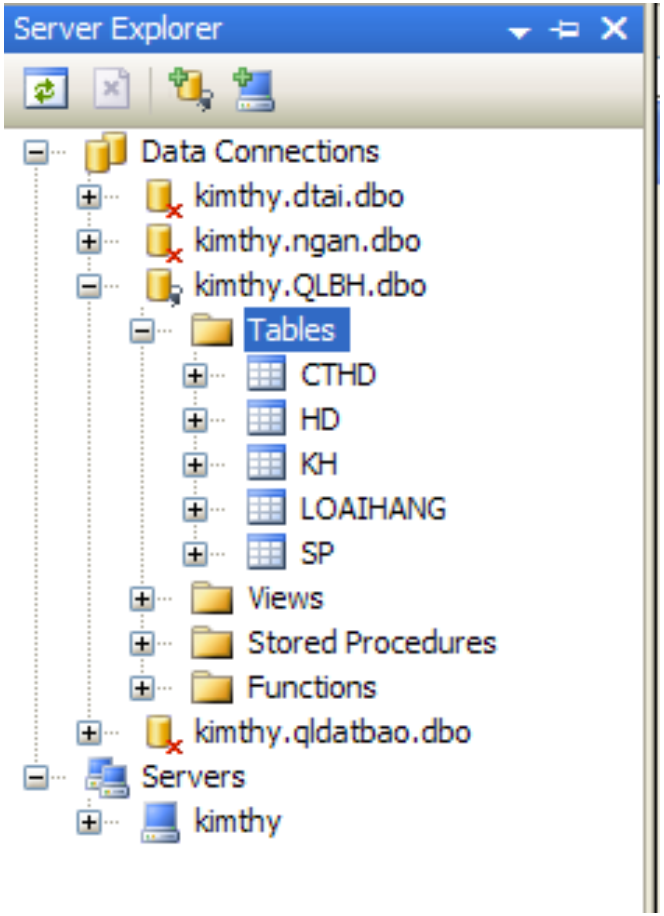


5. Nhập vào các thiết lập cho kết nối rồi nhấn Test Connection để chắc chắn rằng các thiết lập của bạn đúng. Nhấn OK nếu kết nối thành công. Visual Studio sẽ thêm các kết nối vào Server Explorer như chỉ ra trong hình 3.4

Hình 3.4 *chọn Database trên Server*



Hình 3.5 Database kết nối



6. Nhấn vào dấu cộng trong Server Explorer sẽ mở rộng các đề mục bên trong. Để xem các bảng trong một kết nối dữ liệu, mở rộng các đề mục bên dưới kết nối dữ liệu, sau đó mở rộng các đề mục dưới các bảng. Visual Studio sẽ hiển thị các bảng như chỉ ra trong hình 3.6

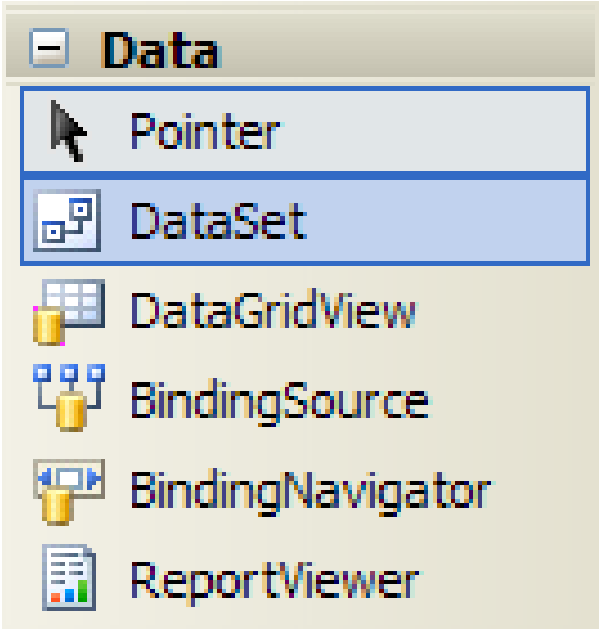
Hình 3.6 Các đề mục đã mở rộng trong Server Explorer

3.2 Tạo bộ điều phối dữ liệu DataAdapter

Bạn vừa tạo ra và kích hoạt một kết nối CSDL, bạn cần phải tạo ra thêm một bộ điều phối dữ liệu gọi là Data Adapter để rút trích thông tin trong CSDL đã kết nối. Data Adapter sẽ định nghĩa chính xác những thông tin mà bạn muốn lấy trong CSDL, nó là nền tảng để tập dữ liệu Dataset sử dụng sau này, đây là bước bắt buộc khi bạn muốn sử dụng Dataset.

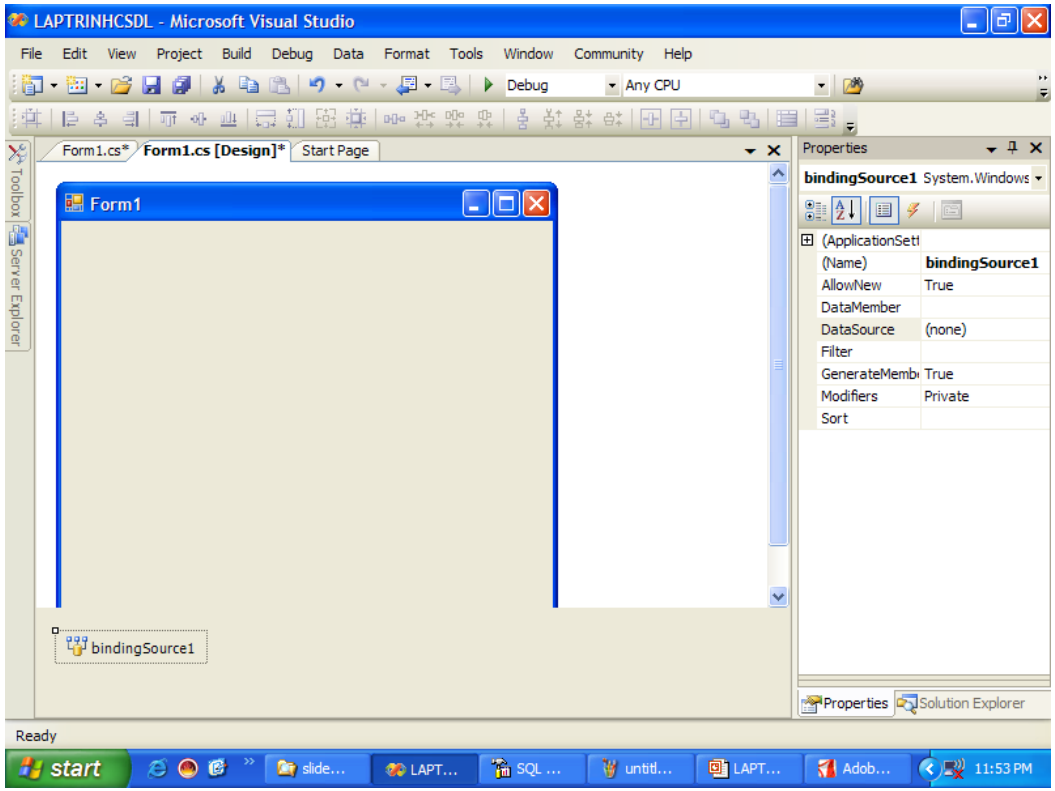
Sử dụng đối tượng điều khiển BindingSource

1. Chọn tab Data trong cửa sổ ToolBox, tab Data này chức các điều khiển cho phép bạn truy xuất đến CSDL trong chương trình (hình 3.7)



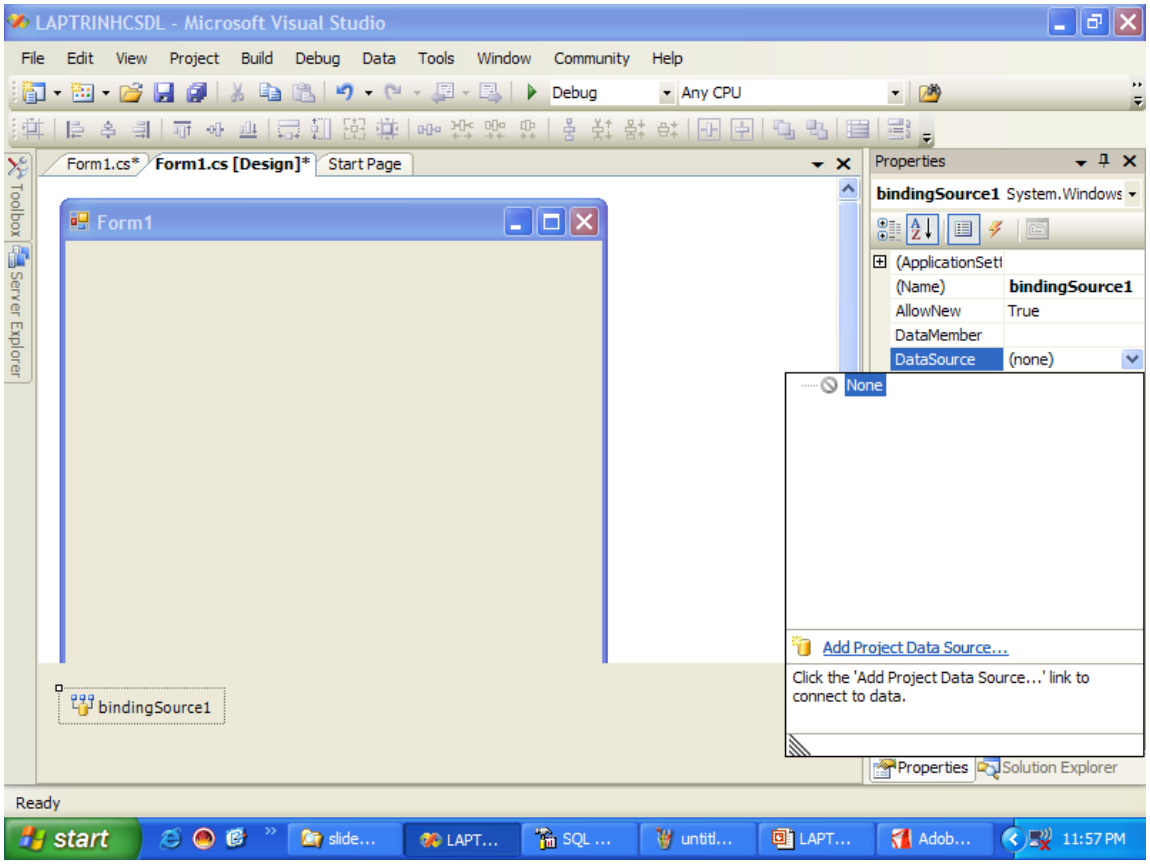
Hình 3.7

1. Kéo đối tượng BindingSource từ tab Data vào cửa sổ form. BindingSource được thiết kế cho mục đích kết nối đến CSDL đã kết nối bằng Server Explorer (phần 3.1) hoặc CSDL chưa có sẵn kết nối (hình 3.8)



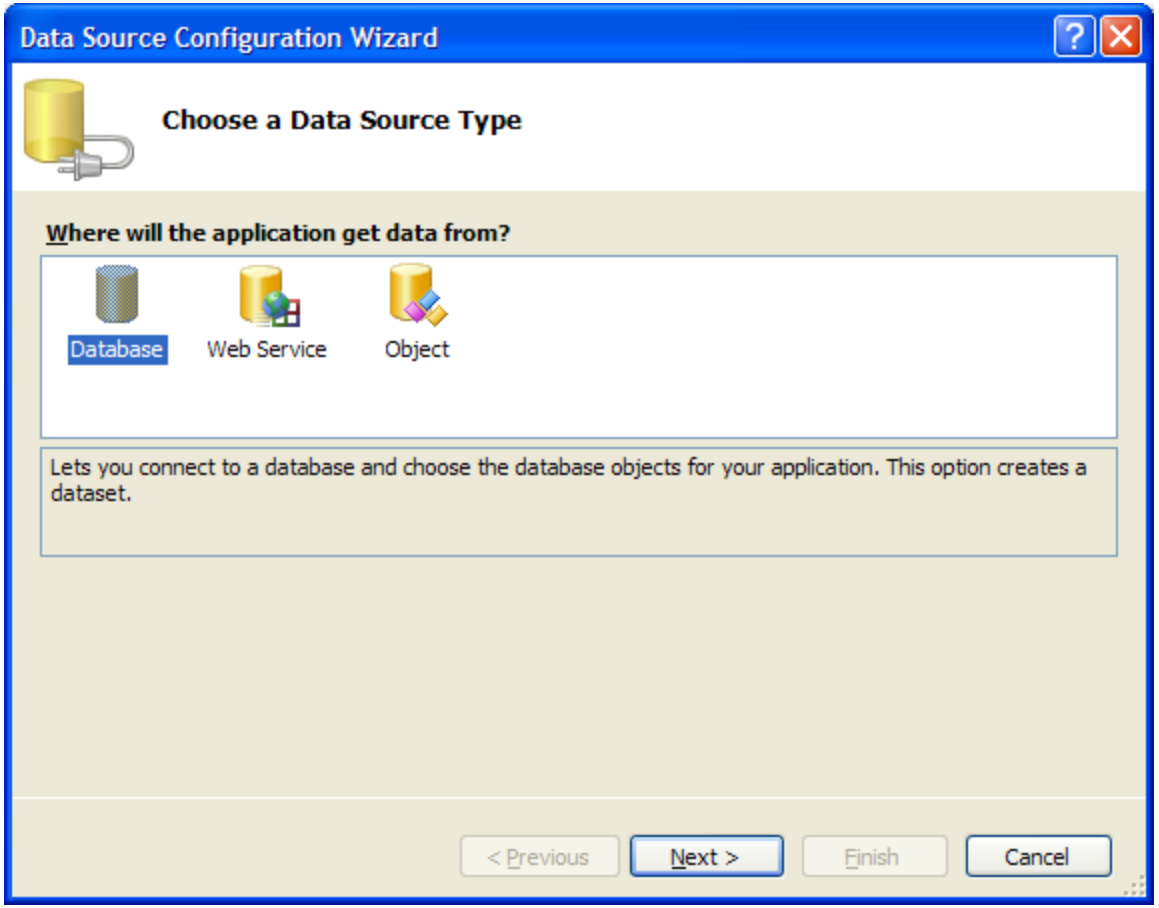
Hình 3.8

2. Kích đối tượng BindingSource trên khay công cụ bên dưới form, nhấn F4 để hiển thị cửa sổ Properties, kích chọn Datasource > Add project Data Source (hình 3.9)



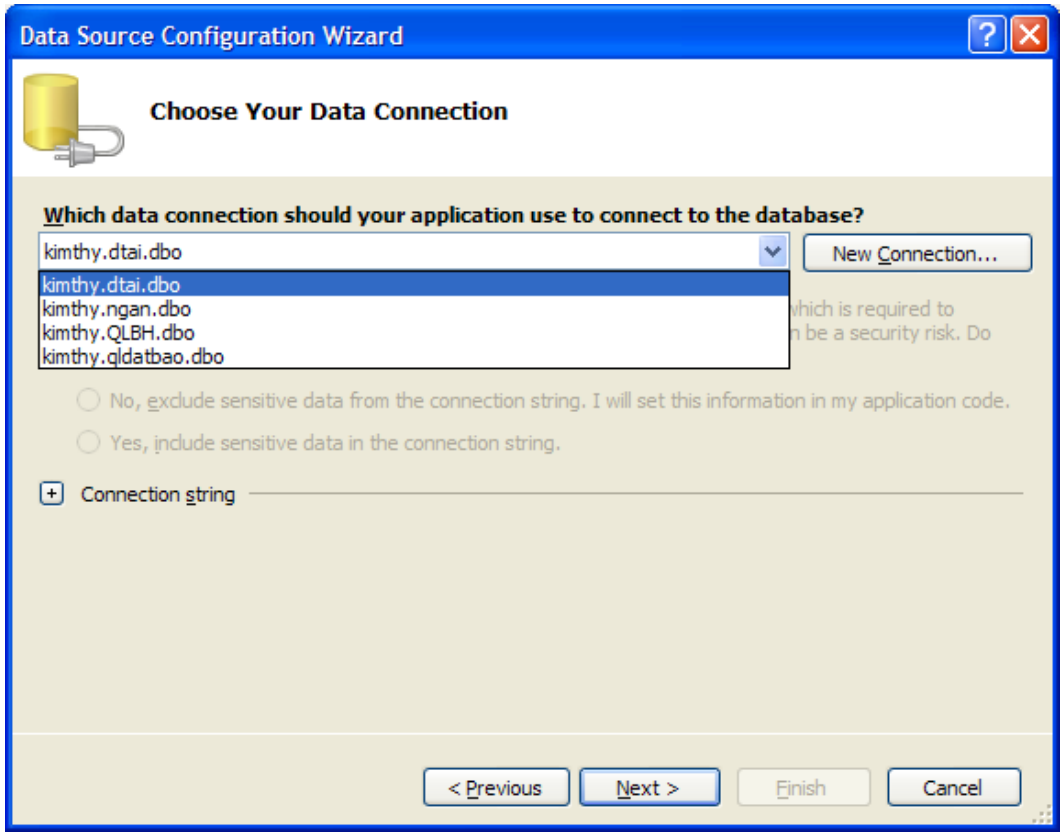
Hình 3.9

3. Xuất hiện cửa sổ Data Source ConfigurationWizard (hình 3.10), kích Next để chọn Database



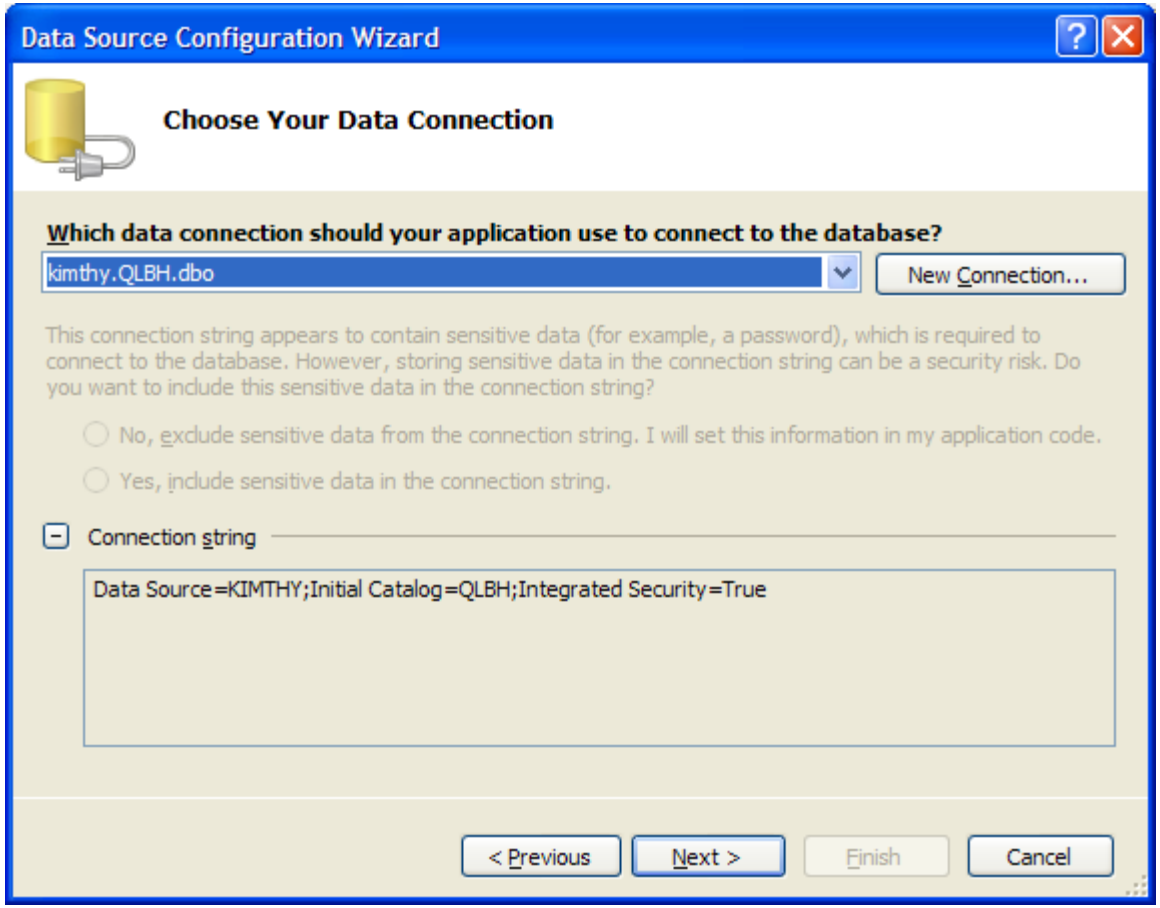
Hình 3.10

4. Nếu bạn đã tạo sẵn kết nối bằng Server Explorer bạn hãy kích chọn Data Connection cần kết nối. Nếu bạn chưa tạo kết nối bằng Server Explorer thì kích New Connection (hình 3.11)



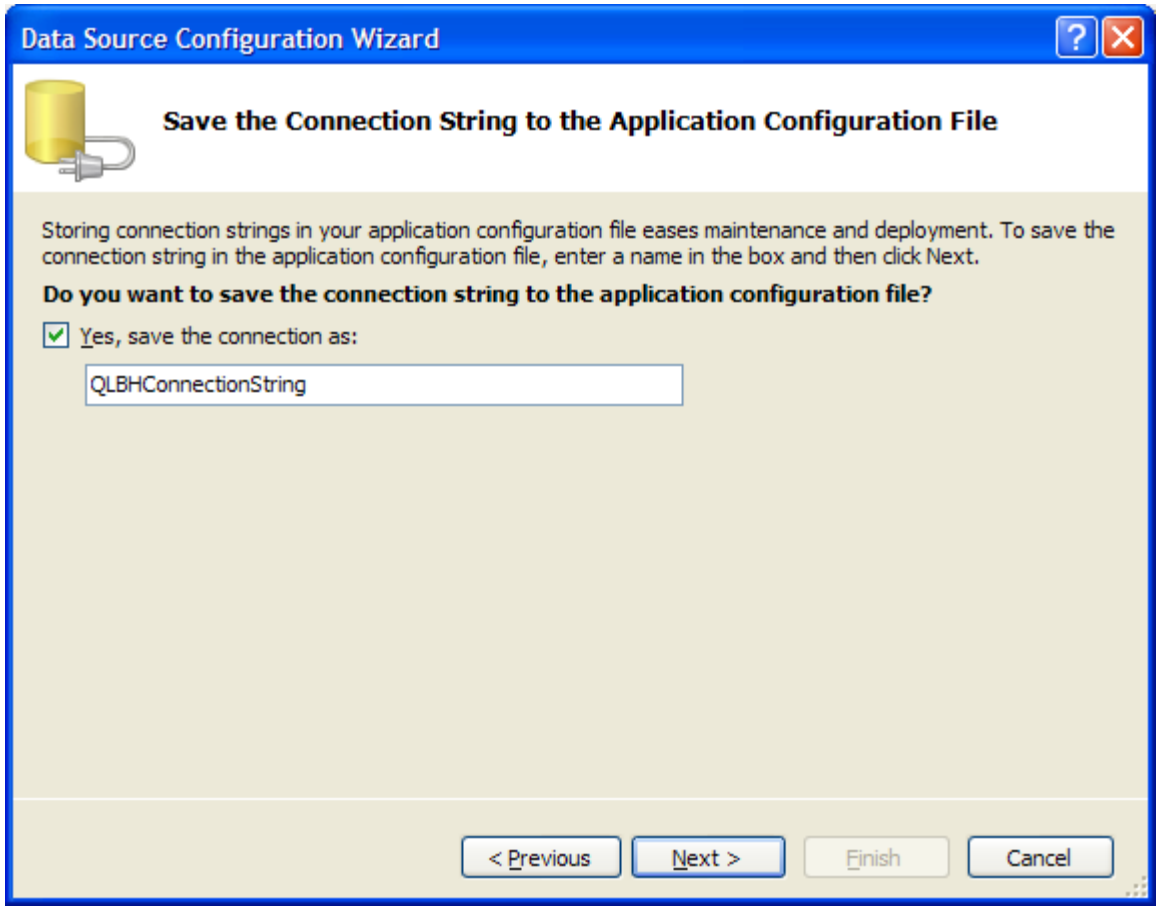
Hình 3.11

5. Kích Connection string, xuất hiện cửa sổ lệnh bên dưới, cho thấy chuỗi lệnh kết nối CSDL (hình 3.12).



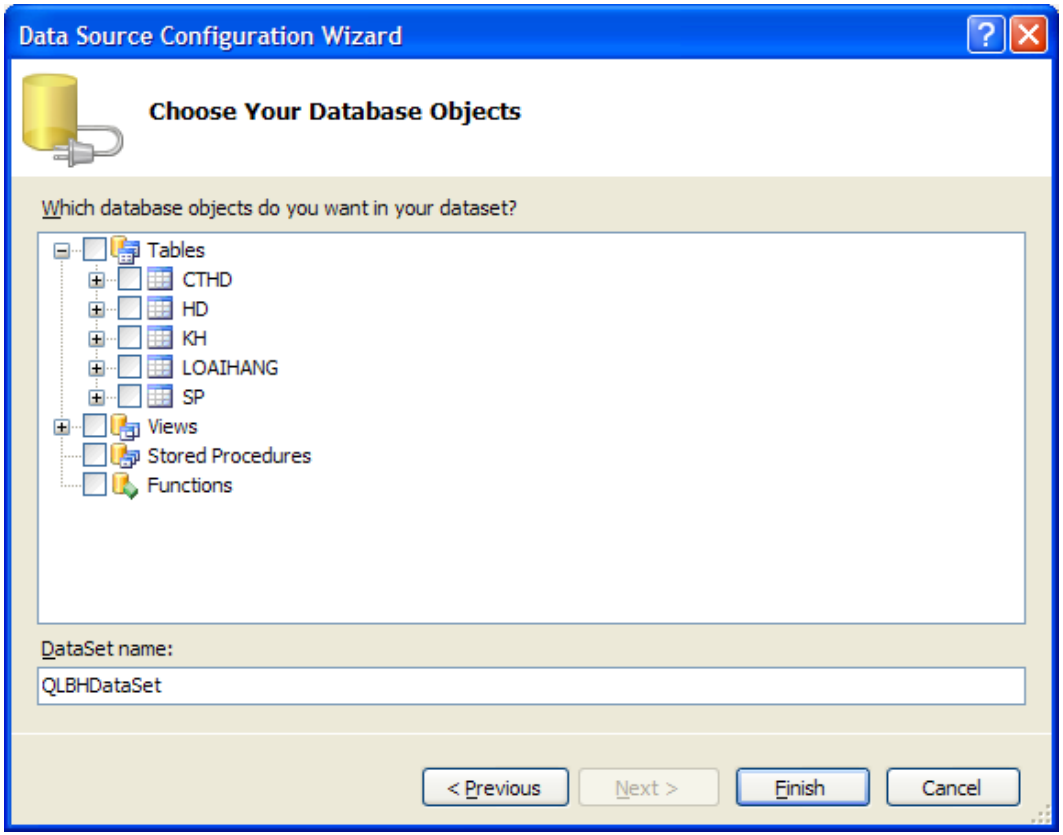
Hình 3.12

6. Kích Next, xuất hiện cửa sổ hình 3.13.



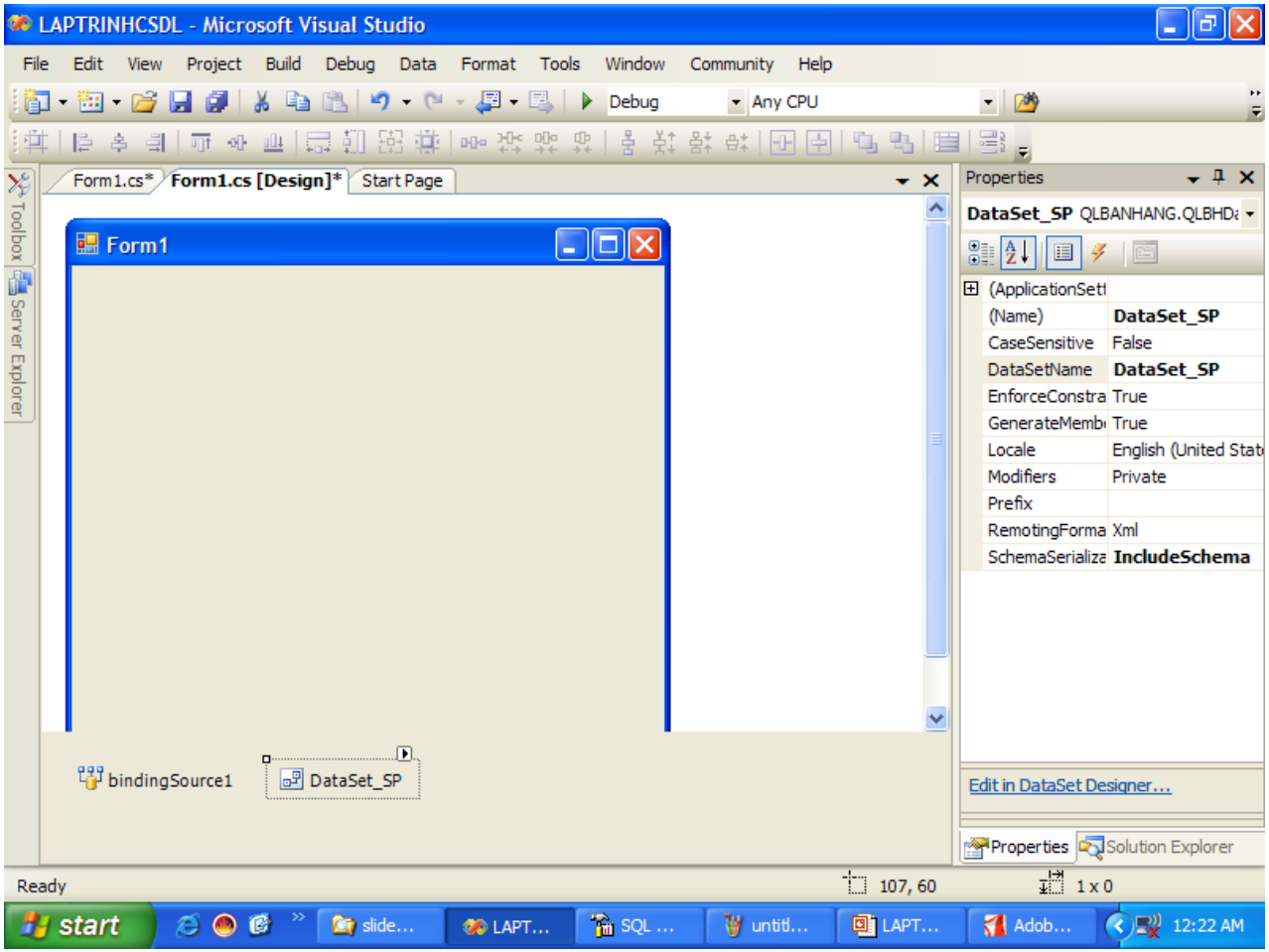
Hình 3.13

7. Kích Next, xuất hiện cửa sổ cho phép bạn chọn các đối tượng CSDL (hình 3.14), chẳng hạn như Tables, Views, StoredProcedure hoặc Functions. Sau đó bạn đặt tên cho Dataset, kích Finish.



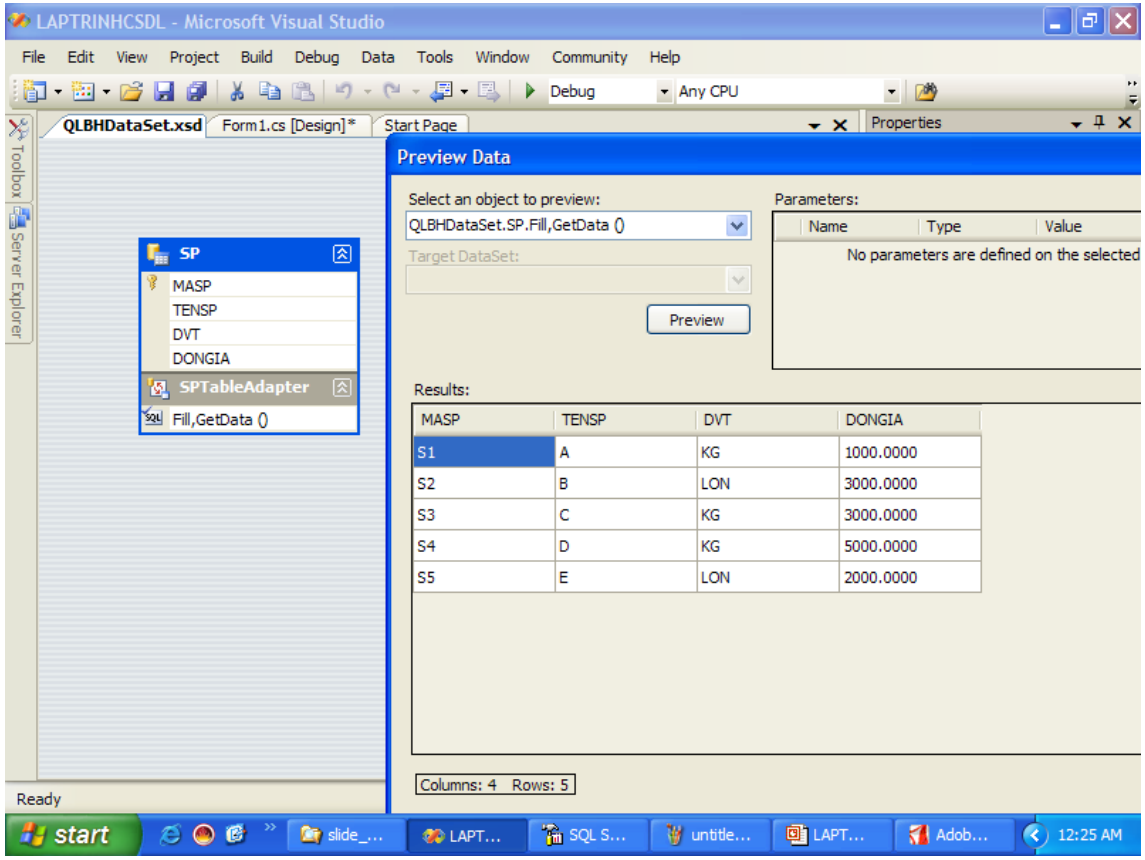
Hình 3.14

Đối tượng trình diễn dữ liệu (Dataset) là hình ảnh dữ liệu có được từ DataAdapter (hình 3.15).



Hình 3.15

Để xem dữ liệu trong Dataset trong chế độ Design, nhấp chuột vào đối tượng DataSet và chọn Edit in Dataset Designer từ pop-up menu. Visual Studio hiển thị Dataset trong cửa sổ XML Designer như chỉ ra trong hình 3.16



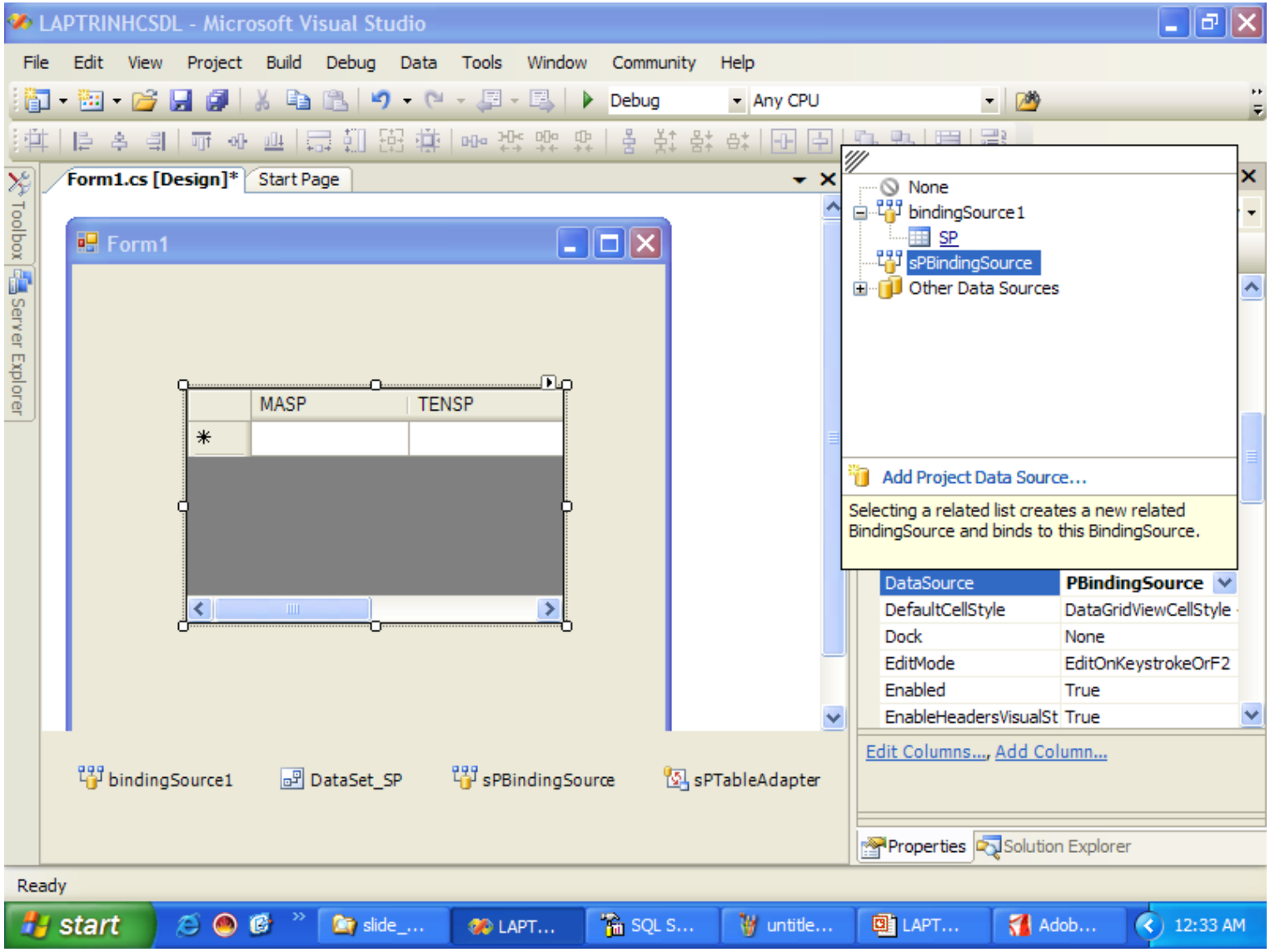
Hình 3.16

3.3 Hiển thị data set

Để hiển thị data set trên Windows form trong thời gian thực thi, làm theo các bước sau:

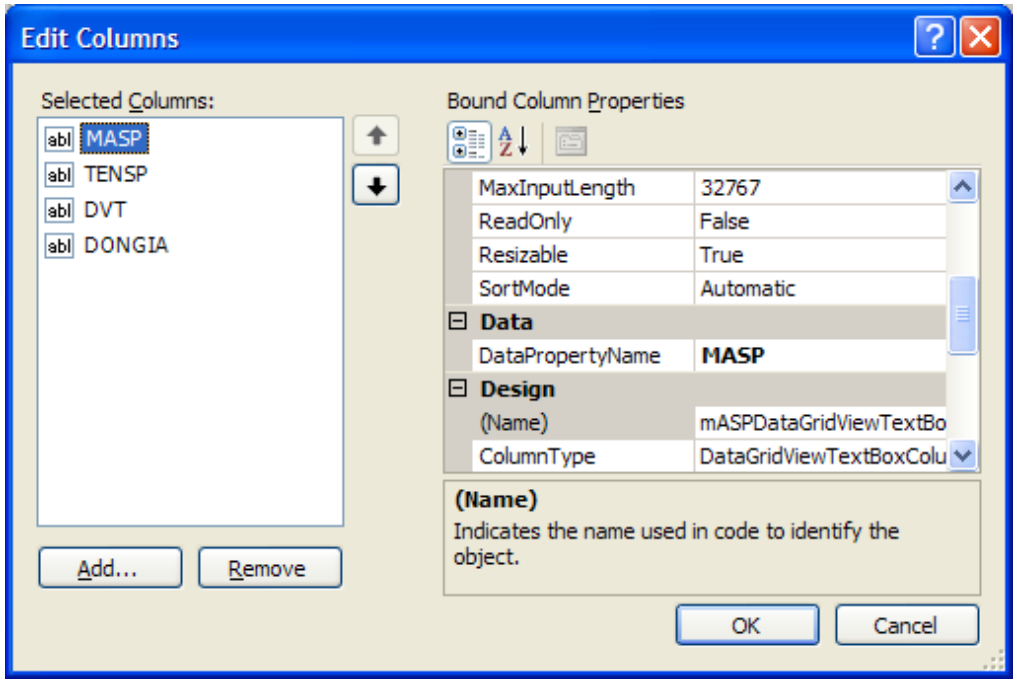
1. Thêm một control vào form để hiển thị dữ liệu. Chẳng hạn, thêm control dataGridView vào form.

2. Thiết lập nguồn dữ liệu của control là Dataset. Ví dụ như, đối với control dataGridView, nhấn vào DataSource trong cửa sổ Properties, thiết lập thuộc tính DataSource là đối tượng DataSet, như được chỉ ra ở hình 3.17

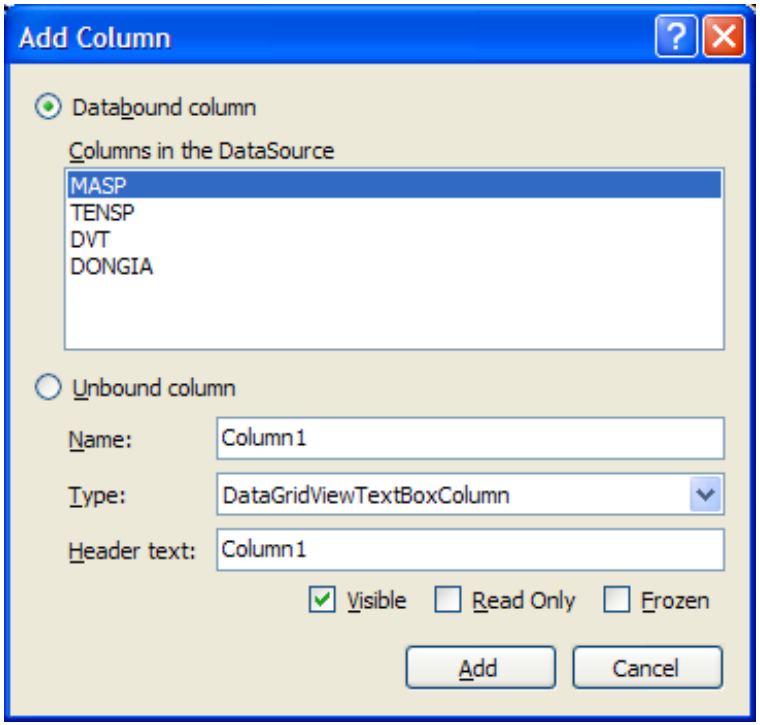


Hình 3.17 *Kết nối dữ liệu trên dataGridView*

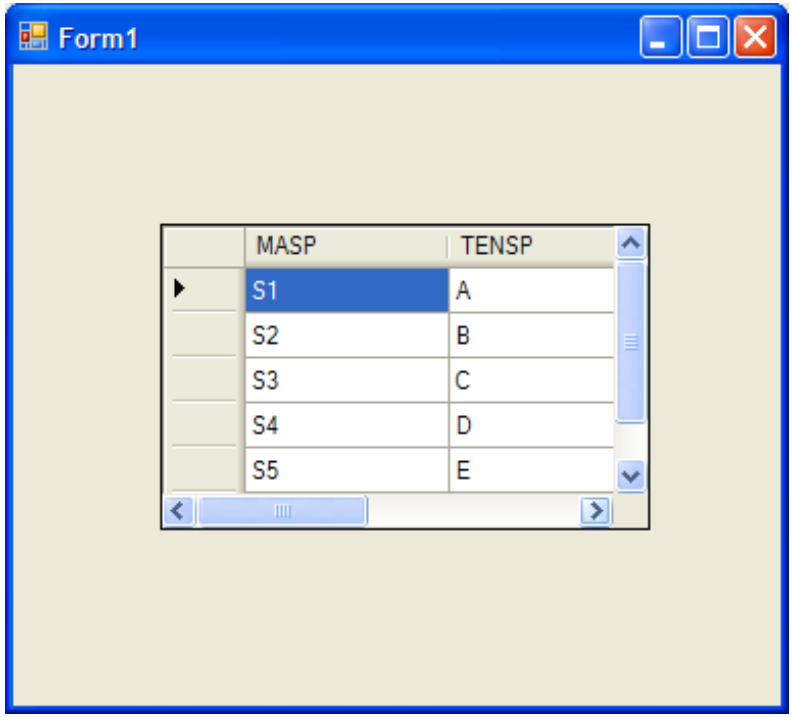
3. Chọn các cột để hiển thị trên control. Đối với dataGridView control, nhấn Columns trong hộp thoại Properties, bỏ chọn cột bằng cách kích chọn tên cột > kích Remove (hình 3.18), bạn cũng có thể thêm các cột để hiển thị từ danh sách Add Columns (hình 3.19). Nhấn OK sau khi bạn làm xong.



Hình 3.18 *Chọn các cột hiển thị trên dataGridView*



Hình 3.19 *Thêm các cột để hiển thị trên dataGridView*



Hình 3.20 d*ataGridView trong thời gian thực thi*

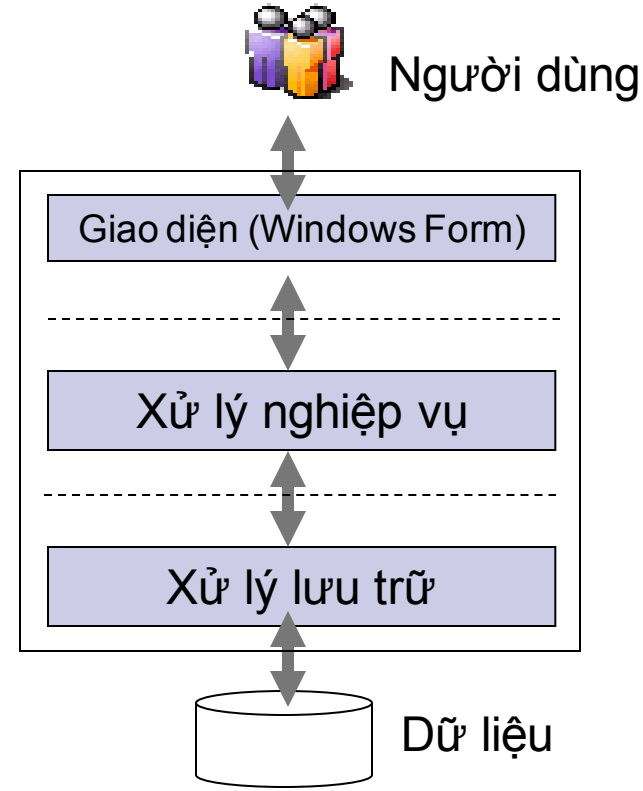
Chương 3

XÂY DỰNG LỚP XỬ LÝ

Để hiển thị dữ liệu trên Windows Form, chúng ta đã viết khá nhiều dòng lệnh trên đó chẳng hạn như : tạo chuỗi kết nối, thực hiện đọc bảng dữ liệu, viết xử lý liên kết dữ liệu … Việc để lẫn lộn những đoạn code về truy cập dữ liệu và xử lý trên giao diện gây không ít khó khăn trong việc xây dựng, phát triển và bảo trì ứng dụng.

Vì thế, trong phần này chúng ta sẽ học cách thức xây dựng lớp xử lý. Lớp xử lý đảm nhận trách nhiệm thực hiện các thao tác truy xuất và cập nhật dữ liệu,hiển thị dữ liệu trên Form và tiếp nhận thông tin người dùng.

Việc phân chia công việc cụ thể cho từng đối tượng không những giúp cho chúng ta xây dựng và phát triển ứng dụng một cách hiệu quả mà còn dễ dàng trong quá trình bảo trì, phù hợp với xu hướng phát triển phần mềm sử dụng các ngôn ngữ lập trình hiện đại.



Mô hình xử lý của ứng dụng

1.Xây dựng lớp xử lý lưu trữ :

1.1. Thiết kế tổng quan :

Để dễ dàng theo dõi cấu trúc chi tiết của lớp xử lý lưu trữ (XL_BANG), chúng ta bắt đầu tìm hiểu thiết kế tổng quan của nó.

Lớp xử lý dữ liệu (XL_BANG) thực hiện các chức năng :

- Truy xuất dữ liệu từ cơ sở dữ liệu.
- Thực hiện các câu lênh Sql.

```
using System.Data;
using System.Data.SqlClient;
public class XL_BANG
{
#region "Khai báo biến thành viên "
#endregion

#region "Danh sách các thuộc tính "
#endregion

#region "Nhóm hàm cung cấp thông tin "
#endregion

#region "Nhóm hàm xử lý tính toán "
#endregion

#region "Xử lý sự kiện "
#endregion
}
```

```
using System.Data;
using System.Data.SqlClient;

/// <summary>
/// Summary description for XL_BANG
/// </summary>
public class XL_BANG : DataTable
{
"Khai báo biến thành viên "

"Danh sách các thuộc tính "

"Nhóm hàm cung cấp thông tin "

"Nhóm hàm xử lý tính toán "

"Xử lý sự kiện "
}
```

Phân vùng với region

Nhóm từ khóa #region và #endregion tạo ra các phân vùng, giúp chúng ta dễ dàng quản lý các đoạn lệnh trong quá trình xây dựng lớp.

Như bạn đã thấy, lớp XL_BANG được kế thừa từ lớp DataTable, đồng nghĩa với việc nó sẽ thừa hưởng tất cả những thuộc tính, phương thức,… từ lớp DataTable.

Trong lớp xử lý trên, chúng ta có thực hiện các thao tác truy xuất và cập nhật dữ liệu, do đó chúng ta cần sử dụng bộ thư viện của ADO.Net. Bộ thư viện được sử dụng trong lớp xử lý này là System.Data.SqlClient.

1.2. Cấu trúc chi tiết lớp XL_BANG :

1.2.1 Khai báo biến thành viên :

```
#region "Khai báo biến thành viên "
 //Đối tượng truy cập dữ liệu
 private SqlDataAdapter mBo_doc_ghi;

//Biến chuỗi chứa nội dung truy cập dữ liệu
 private string mChuoi_Sql;

//Biến chứa tên bảng muốn truy vấn
 private string mTen_bang;

//Biến kết nối dùng chung đến nguồn dữ liệu
 private SqlConnection mKet_noi;

//Biến chứa thông tin vị trí nguồn dữ liệu.
//Giá trị này phải được gán trước khi sử dụng lớp.
 public string Chuoi_CSDL;

#endregion
```

Các từ khóa khai báo :

Giá trị	Mô tả
public	Cho biết phương thức được gọi ở mọi nơi
protected	Cho biết phương thức chỉ được gọi trong phạm vi của Class khai báo và các lớp con (Subclass)
private	Cho biết phương thức chỉ được gọi trong phạm vi của Class

1.2.2 Danh sách các thuộc tính :

Ứng với mỗi biến thành viên cần giao tiếp ra bên ngoài, chúng ta cung cấp thuộc tính tương ứng để làm việc.

```
#region "Danh sách các thuộc tính "
    public string Chuoi_Sql
    {
        get
            {
                return mChuoi_Sql;
            }
        set
            {
                mChuoi_Sql = value;
            }
     }
```

```
public string Ten_bang
    {
        get
        {
            return mTen_bang;
        }
        set
        {
            mTen_bang = value;
        }
    }
    public SqlConnection Ket_noi
    {
        get
        {
            return mKet_noi;
        }
        set
        {
            mKet_noi = value;
        }
    }
```

1.2.4 Nhóm hàm cung cấp thông tin

```
#region "Nhóm hàm cung cấp thông tin "
  //Thực hiện lấy cấu trúc và dữ liệu vào DataTable
//sau đó phát sinh bộ lệnh cập nhật dữ liệu
    public void Doc_bang()
    {
        if (mChuoi_Sql == "")
            mChuoi_Sql = "Select * From " + mTen_bang;
        if (mKet_noi == null)
        {
            mKet_noi = new SqlConnection();
            mKet_noi.ConnectionString = Chuoi_lien_ket + "Data Source="
                                                        + Chuoi_CSDL;
        }
```

```
try
        {
            mBo_doc_ghi = new SqlDataAdapter(mChuoi_Sql, mKet_noi);
            mBo_doc_ghi.Fill(this);
            mBo_doc_ghi.FillSchema(this, SchemaType.Mapped);
            mBo_doc_ghi.SelectCommand.CommandText =
                                                    "Select * from mTen_bang";
            SqlCommandBuilder Bo_phat_sinh = new
                                        SqlCommandBuilder(mBo_doc_ghi);
        }
        catch
        {
        }
    }
#endregion
```

1.2.5 Nhóm hàm xử lý tính toán

```
#region "Nhóm hàm xử lý tính toán "
//Hàm cập nhật các thay đổi trên DataTable vào CSDL
public Boolean Ghi()
{
  Boolean ketqua=true;
  try
    {
        mBo_doc_ghi.Update(this);
        this.AcceptChanges();
    }
  catch
    {
        this.RejectChanges();
        ketqua=false;
    }
   return ketqua;
}
```

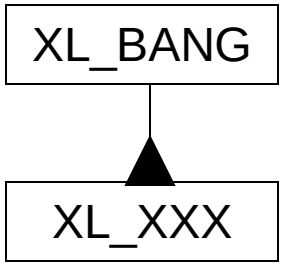
```
//Hàm thực hiện một câu lệnh truy vấn (hành động), nếu thành công trả
//về số mẫu tin được cập nhật, ngược lại hàm trả về -1

public Int32 Thuc_hien_lenh(string pLenh)
{
  try
  {  SqlCommand Cau_lenh=new SqlCommand(pLenh,mKet_noi);
      mKet_noi.Open();
      int ketqua=Cau_lenh.ExecuteNonQuery();
      mKet_noi.Close();
      return ketqua;
   }
  catch  {return -1;}
}
```

```
//Hàm thực hiện nội dung lệnh tính toán thống kê, nếu thành công trả
//về kết quả thống kê, ngược lại trả về Nothing
public Object Thuc_hien_lenh_tinh_toan(string pLenh)
{
 try
  {
     SqlCommand Cau_lenh=new SqlCommand(pLenh,mKet_noi);
     mKet_noi.Open();
     Object ketqua=Cau_lenh.ExecuteScalar();
     mKet_noi.Close();
     return ketqua;
   }
  catch
   {
     return null;
   }
}
#endregion
```

2. Xây dựng lớp xử lý nghiệp vụ

Dựa trên lớp xử lý lưu trữ (XL_BANG), xây dựng các lớp xử lý nghiệp vụ tương ứng với mỗi bảng trong CSDL.



Sơ đồ lớp XL_XXX

Trong đó :

Lớp XL_BANG : đã được xây dựng phần trên

Ký hiệu XXX: tên các bảng tương ứng trong CSDL

Các lớp xử lý nghiệp vụ được kế thừa từ lớp XL_BANG, đồng nghĩa với việc nó sẽ thừa hưởng tất cả những thuộc tính, phương thức,… từ lớp XL_BANG.

Trong lớp xử lý nghiệp vụ, chúng ta có thực hiện các thao tác truy xuất và cập nhật dữ liệu.

chương 6
CRYSTAL REPORT

Nội dung chương 6

1. Crystal Reports .NET
2. Sử dụng Crystal Reports .NET trong C#.NET

1. Crystal Report.Net

- **Giới thiệu Crystal Report**

Crystal Report là phần mềm thiết kế báo biểu chuyên nghiệp được tích hợp trong các phiên bản của Visual Studio(the standard reporting tool for Visual Studio.Net). Phiên bản Studio.Net của Microsoft được tích hợp Crystal Report (hiện nay đã có Crystal Report 2008).

Bản thân Crystal Report là một phần mềm tạo báo biểu độc lập với rất nhiều chức năng thiết kế báo biểu và dịch vụ. Người dùng có thể kết nối với nhiều nguồn dữ liệu khác nhau bằng các ODBC Driver. Báo biểu khi tạo ra cũng có thể được lưu trữ thành những file .rpt độc lập, ở dạng có dữ liệu hay không có dữ liệu. Sau đó file .rpt có thể được chuyển tới người dùng khác và mở bằng Crystal Report hay có thể kết hợp với các ứng dụng viết bằng Visual Basic,Visual C++, C#.Net .

Xét về mặt thiết kế báo biểu,Crystal Report cung cấp đầy đủ các chức năng định dạng dữ liệu và các chức năng phân nhóm, tính toán, subreport và kể cả khả năng lập trình bằng formula dựa trên các formula field.Người dùng ngoài việc sử dụng formula filed còn có thể tự xây dựng bộ thư viện hàm của riêng mình và đưa vào Crystal Report thông qua các DLL.

Bên cạnh khả năng thiết kế báo biểu thông thường,Crystal Report còn cung cấp chức năng thiết kế biểu đồ dựa trên nguồn dữ liệu lấy từ CSDL.

Xét về mặt sử dụng báo biểu,công cụ hiển thị của Crystal Report cho phép người dùng tương tác rất linh hoạt.Báo biểu hiển thị có thể được lọc lại các dữ liệu cần thiết hay xem một phần báo biểu bằng cách sử dụng cấu trúc hiển thị dữ liệu dạng cây. Các Section trong báo biểu cũng có thể mở rộng hay thu hẹp để hiển thị hay che bớt những dữ liệu không cần thiết.Một khi báo biểu đã được xây dựng,người dùng còn có thể Export sang các dạng file khác như Word,Excel,HTML,…

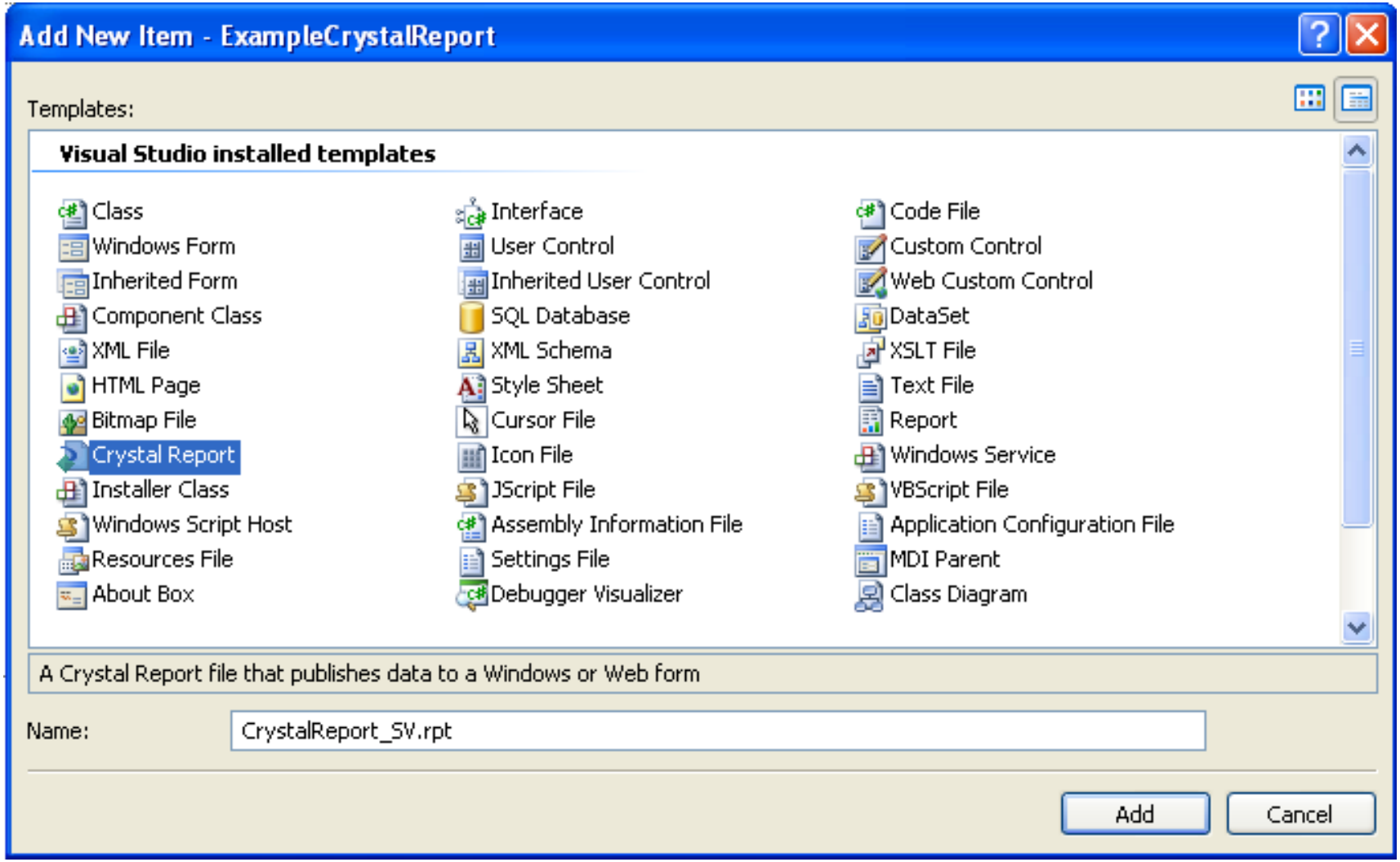
Bằng cách tích hợp Crystal Report,Visual Studio.Net đem lại cho người dùng một công cụ xây dựng báo biểu hiệu quả,tiết kiệm nhiều thời gian so với việc phải sử dụng các đối tượng in ấn để tự phát sinh báo biểu . Chúng ta có thể sử dụng Report Expert để tạo ra báo biểu dựa vào Wizard và Template định sẵn hay thiết kế báo biểu bằng tay.

- Phiên bản đi kèm với Visual Studio.Net 2003 là Crystal Report 9.0.
- Phiên bản đi kèm với Visual Studio.Net 2005 là Crystal Report 10.0.

Ưu điểm của Crystal Report:

- Làm việc dễ dàng với Unicode
- Tích hợp chung với Visual Studio.Net
- Tạo report có thể hiển thị ở cả hai môi trường Winform và Webform
- Dễ dàng deploy,có thể sử dụng các merge modules để tạo file setup
- Sử dụng kiến trúc ADO.NET để kết nối CSDL nhanh hơn
- Có khả năng tạo XML Report Web Services

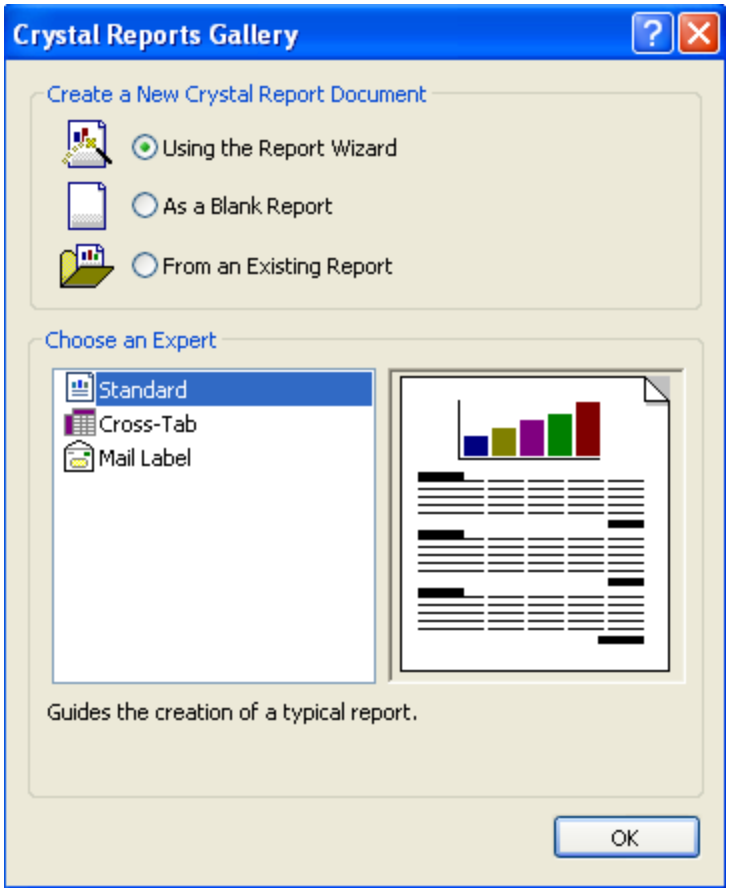
1.2 Tạo mới Crystal Report



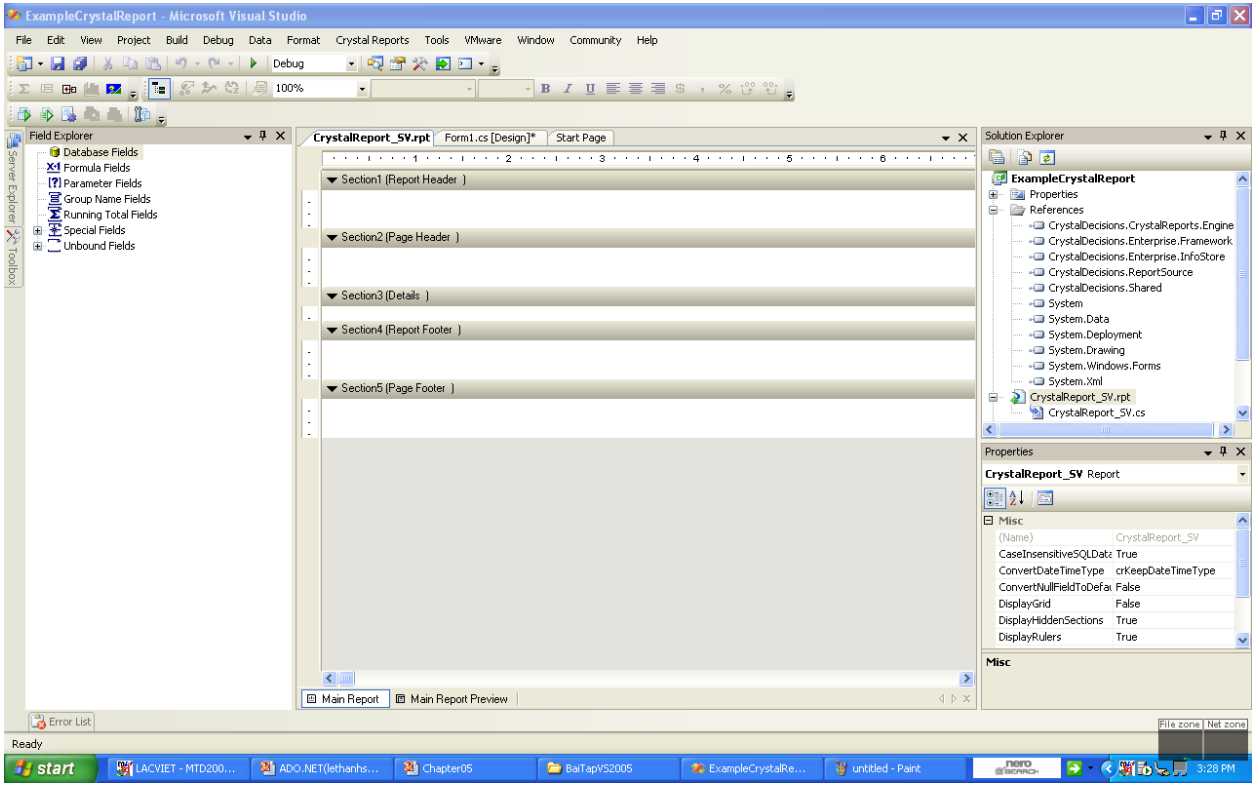
Báo biểu là một phần của porject do đó, để tạo mới một báo biểu chúng ta thực hiện thông qua menu Project | Add New Item hoặc nhấn phím tắt Ctrl +Shift +A .

- Trong phần Templates,chọn mục Crystal Report để tạo mới một báo biểu.
- Trong phần Name của hộp thoại Add New Item nhập vào tên file báo biểu cần tạo.Ví dụ: CrystalReport_SV.rpt.
- Khi nhấn nút Add,Visual Studio.Net tự động chuyển đổi sang giao diện của Crystal Report để chúng ta thực hiện việc tạo mới báo

Có thể chọn tạo mới báo biểu ở một trong hai hình thức:Wizard (Using the Report Wizard) hay tự thiết kế từ đầu(As a Blank Report).Chúng ta cũng có thể mở một báo biểu đã tạo sẵn (From an Existing Report) để đưa vào project :



Khi chọn Blank report, khác với sử dụng phần mềm Crystal Report độc lập,Visual Studio.Net sẽ chuyển ngay sang màn hình thiết kế mà không chọn CSDL cho báo biểu. Để chọn CSDL cho báo biểu,nhấp chuột phải trên mục Database Fields và chọn mục Database Expert.



Các Section trong Report Design Environments(RDE)

- Report Header :hiển thị ở phần đầu trang 1,không hiển thị ở các trang sau 2,3,4,…
- Report Footer :hiển thị ở phần cuối trang cuối cùng.
- Page Header :hiển thị ở phần đầu tất cả các trang.
- Page Footer :hiển thị ở phần cuối tất cả các trang.
- Group Header :hiển thị ở phần đầu của mỗi group.
- Group Footer :hiển thị ở phần cuối của mỗi group.
- Detail :hiển thị phần thông tin dữ liệu,từng record sẽ được lặp lại.

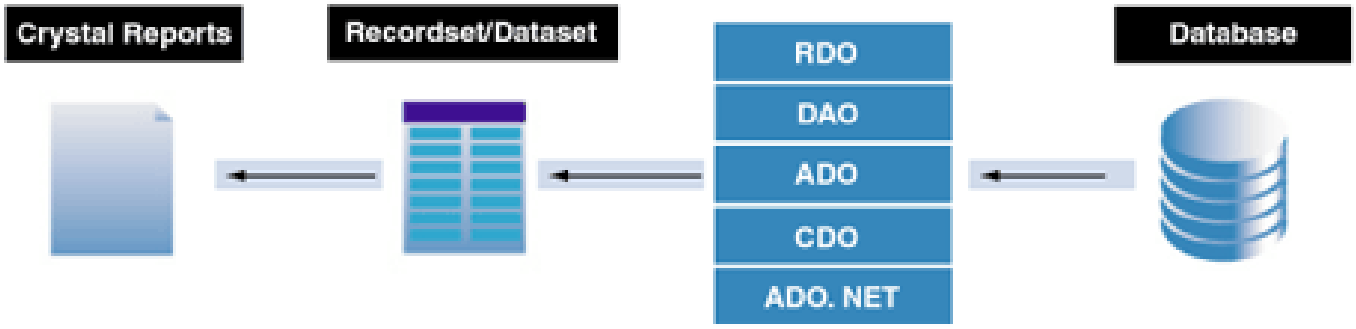
Crystal Report kết nối với nguồn dữ liệu thông qua các trình dữ liệu của nó. Mỗi trình được viết để xử lý cho một loại dữ liệu hoặc cho một kỹ thuật truy xuất dữ liệu cụ thể.

Để tạo thuận lợi cho người lập trình, các trình dữ liệu của Crystal Report cung cấp hai mô hình truy xuất: PULL và PUSH.

Pull Model(Mô hình kéo)

Trong mô hình này,trình sẽ kết nối với nguồn dữ liệu và lấy về dữ liệu yêu cầu.Kết nối với nguồn và lệnh SQL truy xuất dữ liệu đều do Crystal Report xử lý,chúng ta không phải viết dòng lệnh nào.Nếu không có dòng lệnh nào viết cho lúc thực hiện,báo biểu sử dụng mô hình Pull.

Push Model (Mô hình đẩy)



Ngược lại,mô hình đẩy (Push) đòi hỏi chúng ta phải viết lệnh kết nối dữ liệu, thực hiện truy xuất dữ liệu để tạo RecordSet hoặc DataSet chứa các field cần thiết cho báo biểu.Mô hình này cho phép chia sẻ kết nối trong ứng dụng và lọc dữ liệu trước khi đưa vào báo biểu.

1.3 Tạo nguồn dữ liệu cho báo biểu từ DataSet

Một điểm đặc biệt trong Visual Studio.Net là có thể tạo nguồn dữ liệu cho báo biểu từ một DataSet (mô hình Push).

Sử dụng DataSet làm nguồn dữ liệu của báo biểu cho phép tạo ra những báo biểu không cần kết nối với CSDL hay thậm chí tạo ra những báo biểu trong các ứng dụng hoạt động không cần có CSDL đi cùng.Ví dụ như lấy cấu trúc DataSet từ một file XML làm nguồn dữ liệu cho báo biểu.

Để gán nguồn dữ liệu cho báo biểu ,chúng ta cần một biến đối tượng là một thể hiện của báo biểu đã thiết kế.Khi đã có một DataSet,chúng ta sử dụng phương thức SetDataSource của đối tượng báo biểu để gán DataSet đó làm nguồn dữ liệu:

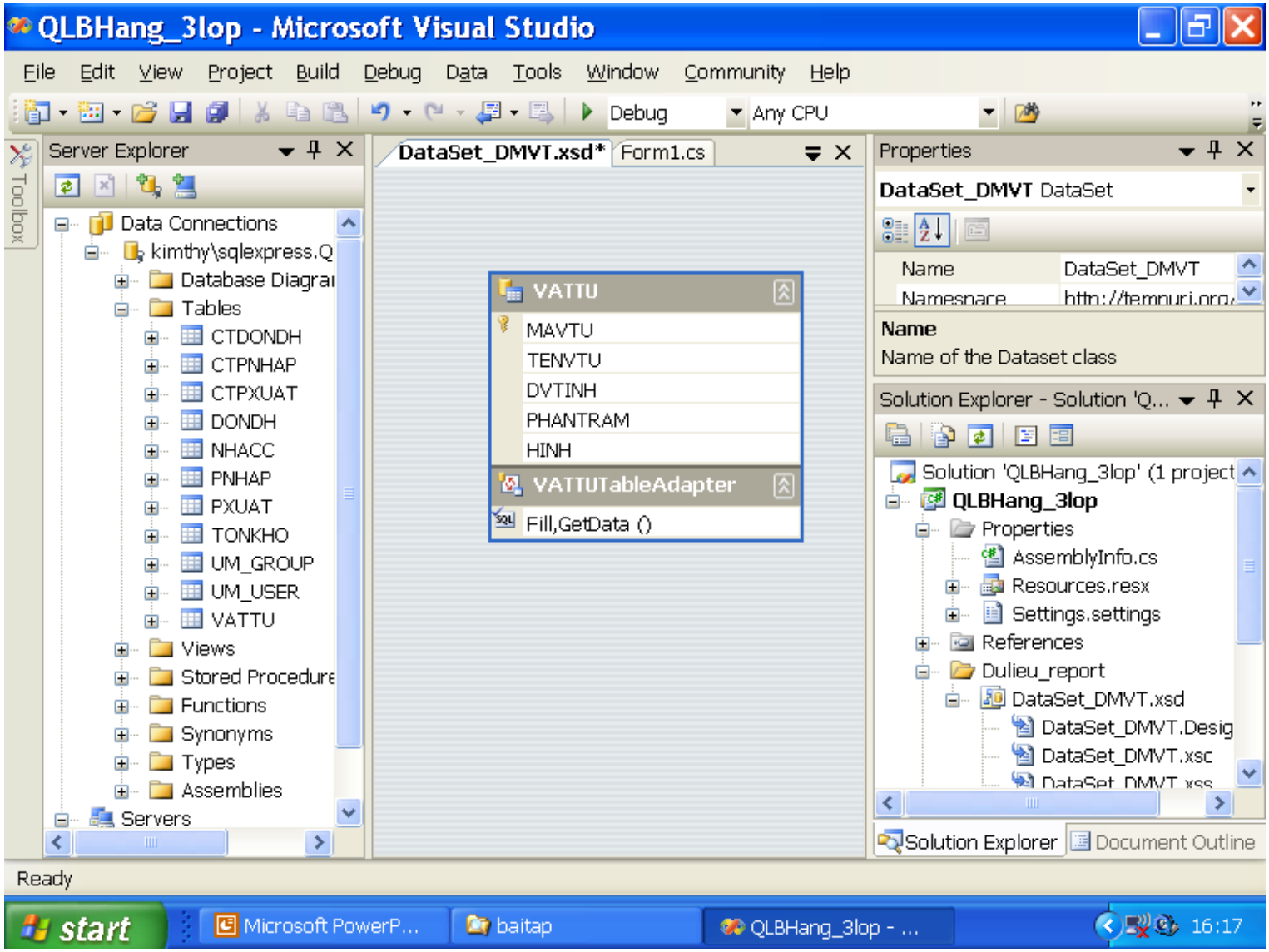
```
DataSet ds;
…
CrystalReport1 rpt = new CrystalReport1();
Rpt.SetDataSource(ds);
```

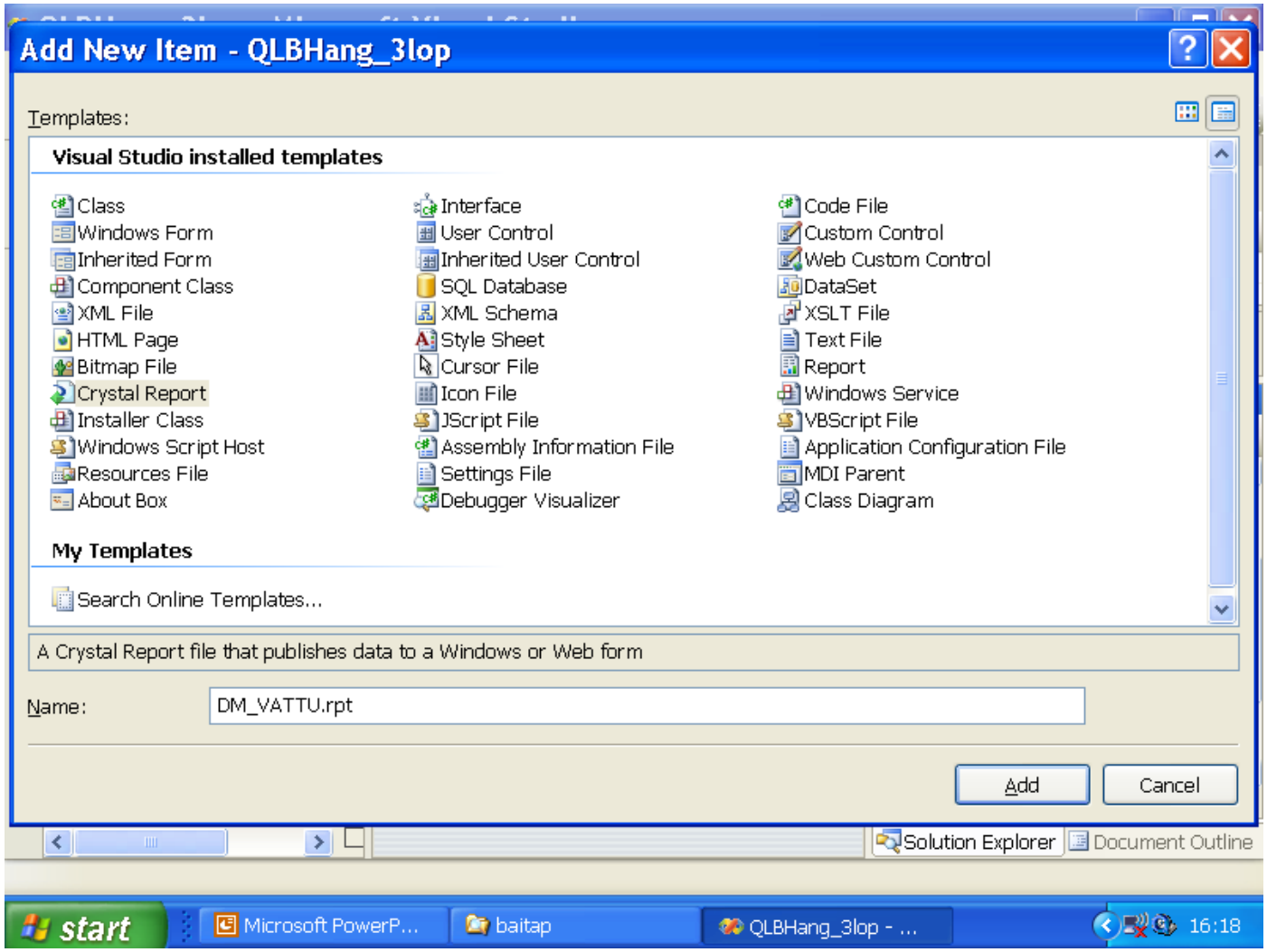
Ví dụ dưới đây minh hoạ cách thiết kế báo biểu sử dụng DataSet làm nguồn dữ liệu

Trong Project,tạo một DataSet mới:

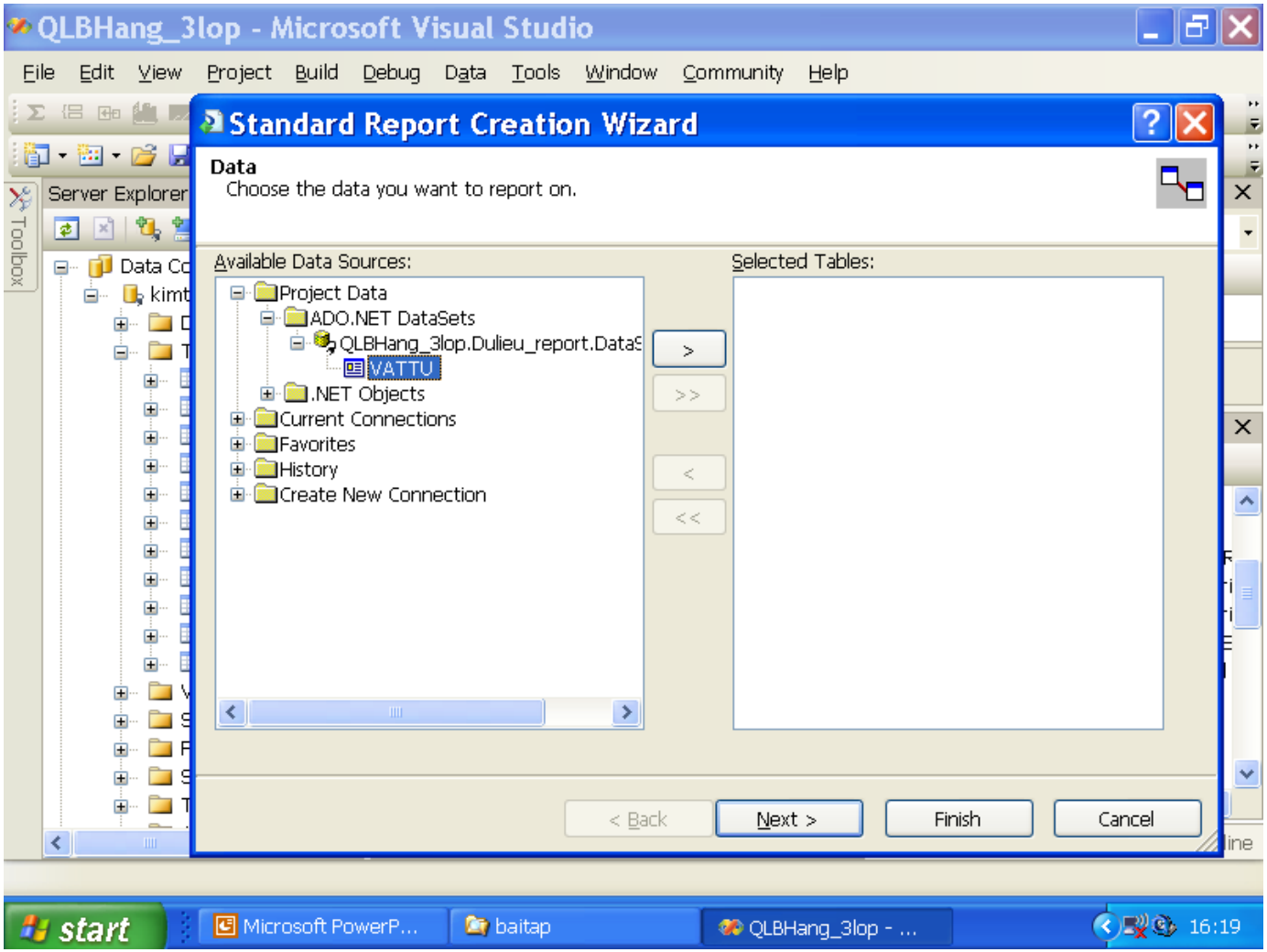
- Menu Project |Add New Item,chọn mục DataSet,trong phần Name,gõ vào tên của DataSet là: Dataset_DMVT.xsd
- Lưu DataSet lại sau khi tạo xong cấu trúc bảng
- Tạo mới một báo biểu (đặt tên là DM_VATTU.rpt),khi chọn nguồn dữ liệu cho báo biểu chúng ta chọn mục ADO.NET DataSet (trong Project Data).
- Chọn DataSet vừa tạo.
- Chọn OK để lưu lại nguồn dữ liệu của báo biểu

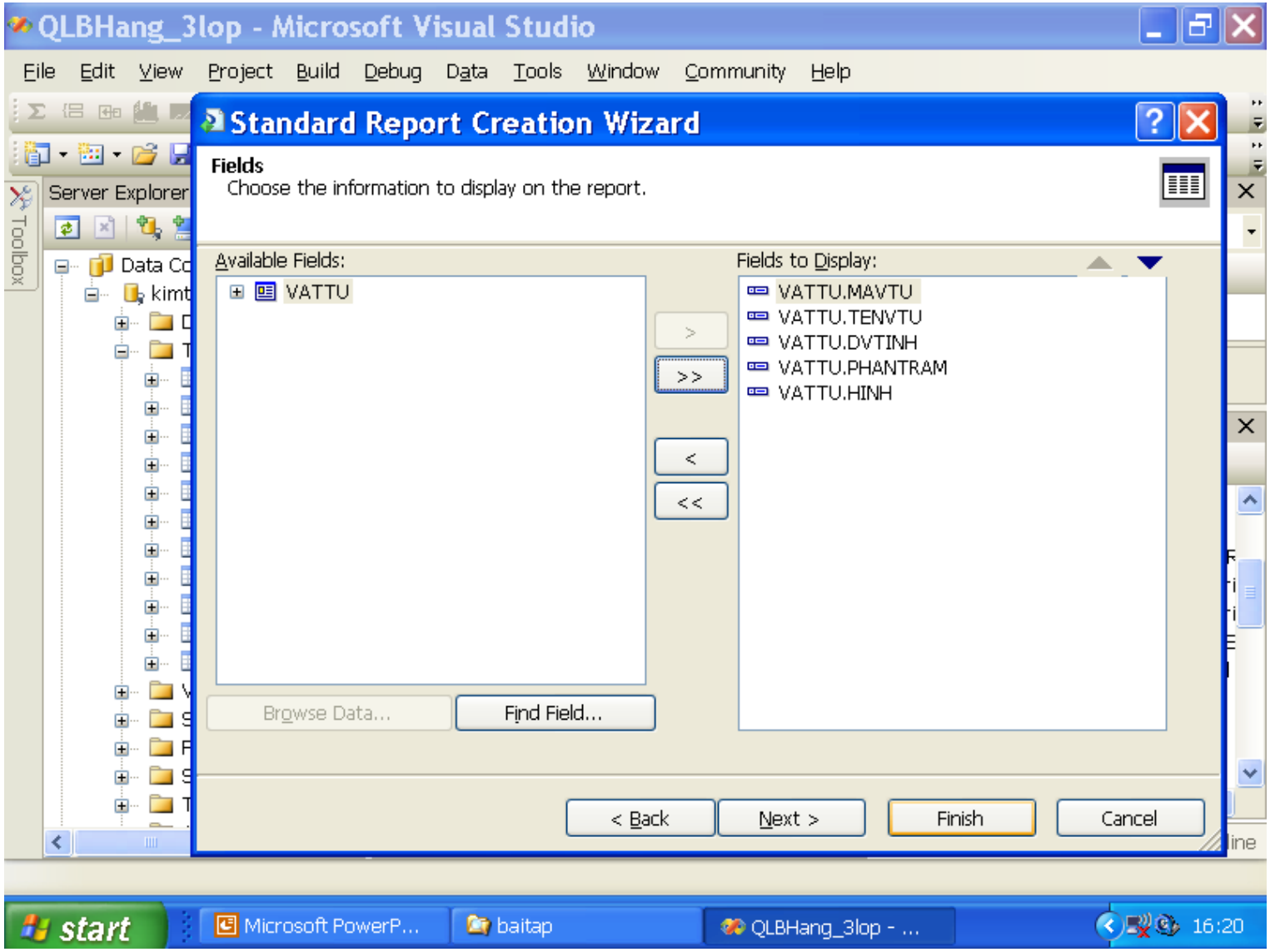
Thiết kế DataSet :

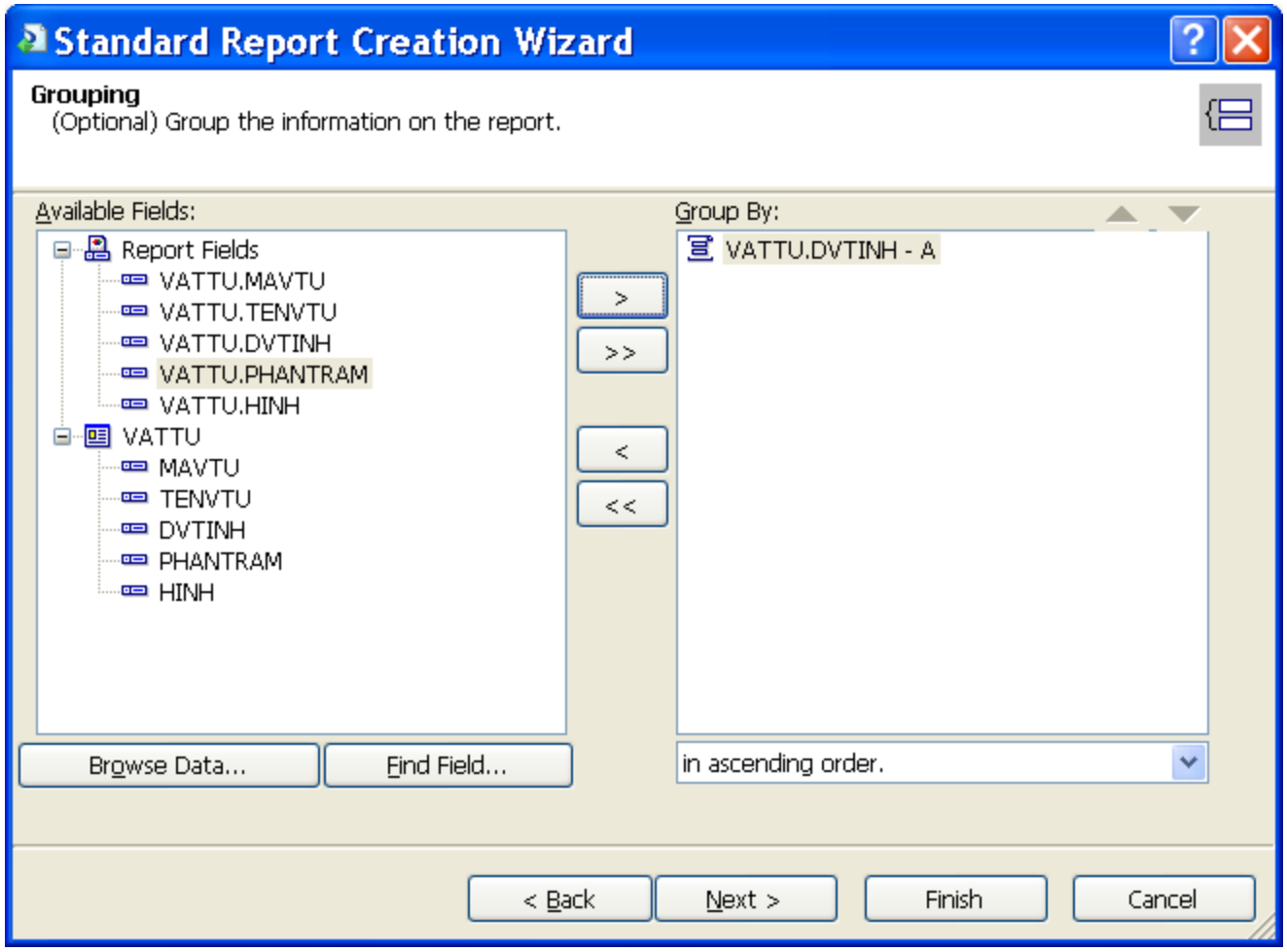


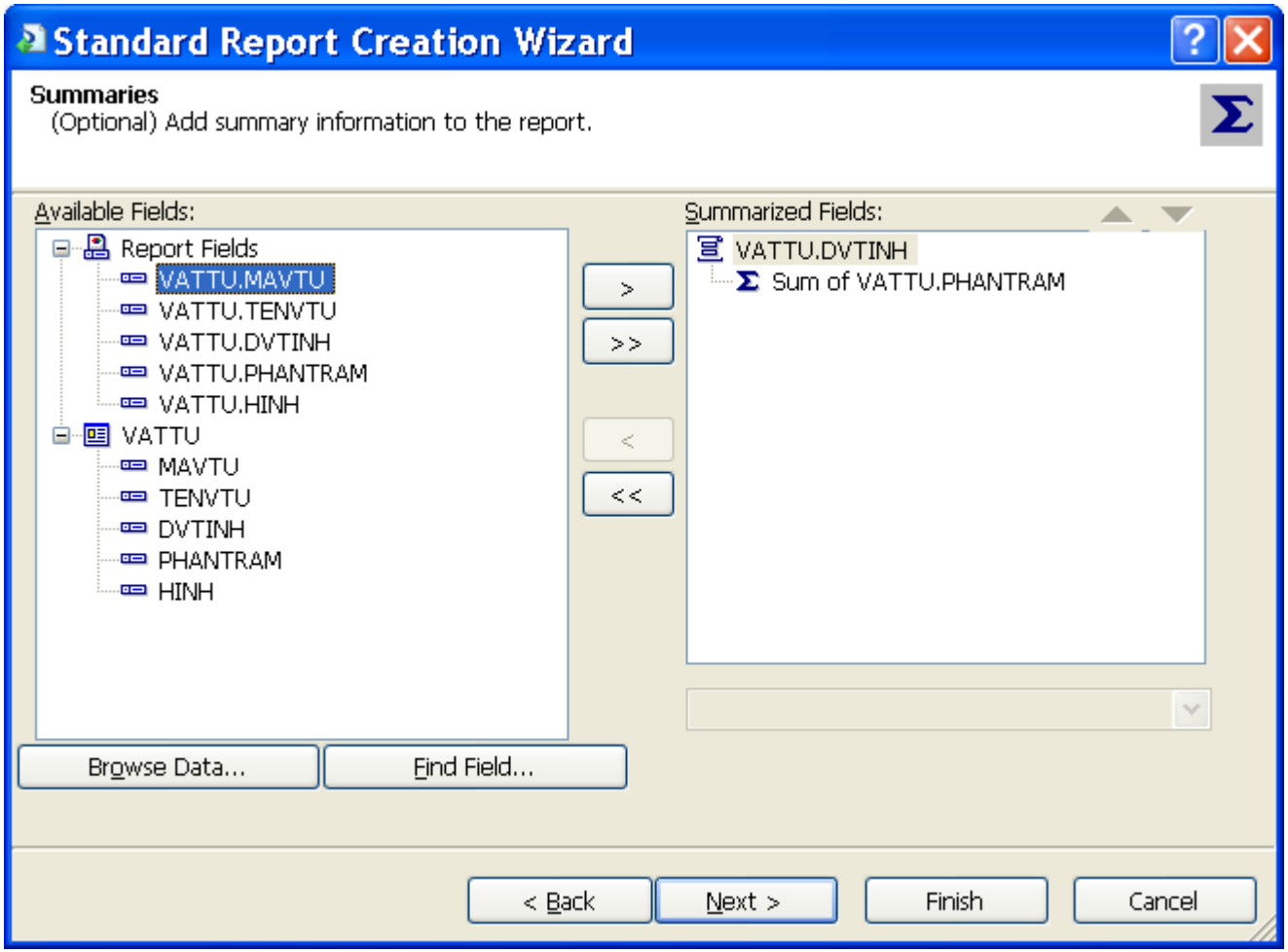

Add New Item - QLBHang_3lop
Templates:
Visual Studio installed templates
Class
Windows Form
Inherited Form
Component Class
XML File
HTML Page
Bitmap File
Crystal Report
Installer Class
Windows Script Host
Resources File
About Box
Interface
User Control
Inherited User Control
SQL Database
XML Schema
Style Sheet
Cursor File
Icon File
JScript File
Assembly Information File
Settings File
Debugger Visualizer
Code File
Custom Control
Web Custom Control
DataSet
XSLT File
Text File
Report
Windows Service
VBScript File
Application Configuration File
MDI Parent
Class Diagram
My Templates
Search Online Templates...
A Crystal Report file that publishes data to a Windows or Web form
Name:
DM_VATTU.rpt
Add
Cancel
Solution Explorer
Document Outline
start
Microsoft PowerP...
baitap
QLBHang_3lop - ...
16:18

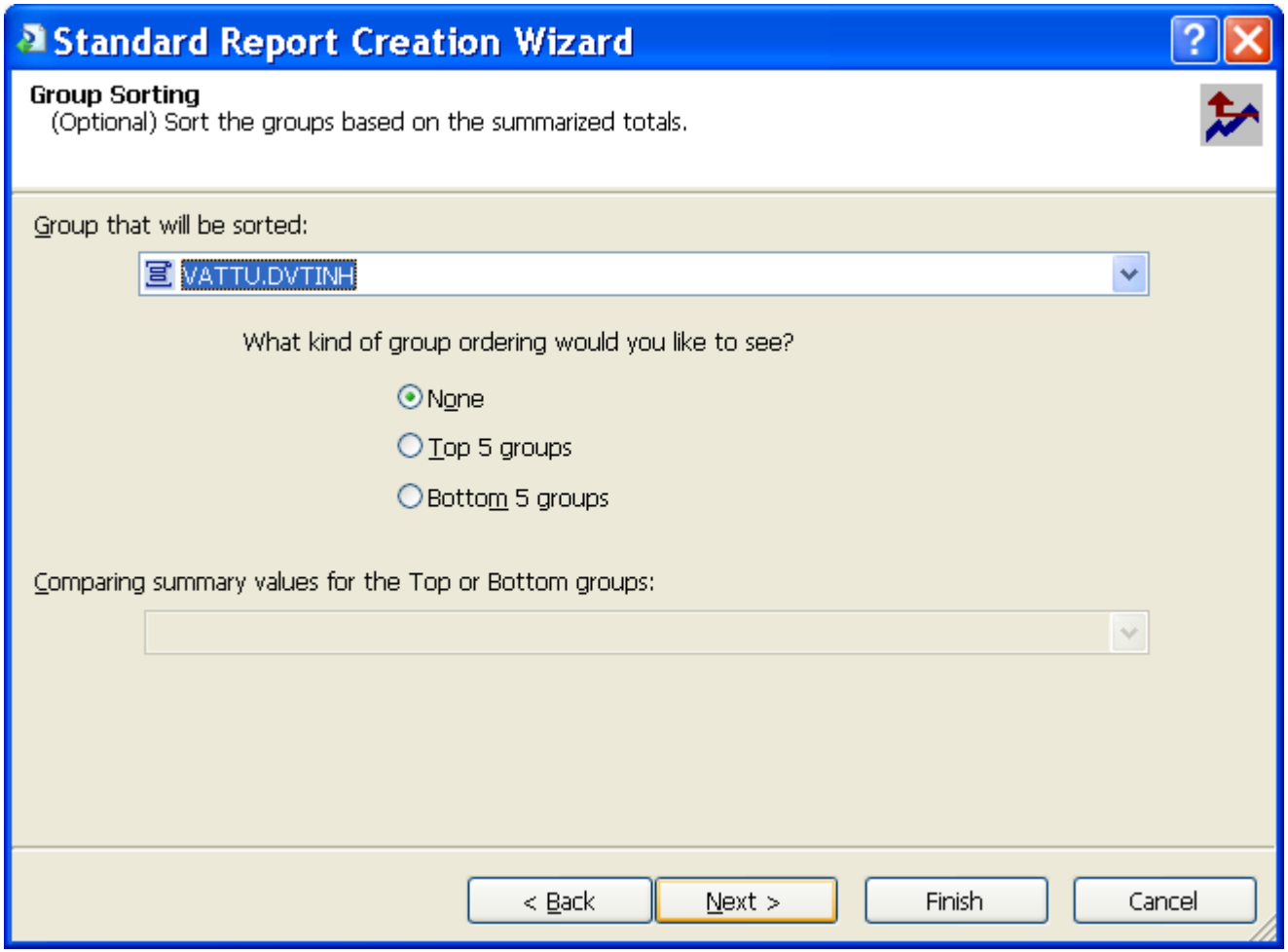
Chọn nguồn dữ liệu cho báo biểu chúng ta chọn mục ADO.NET DataSet (trong Project Data).

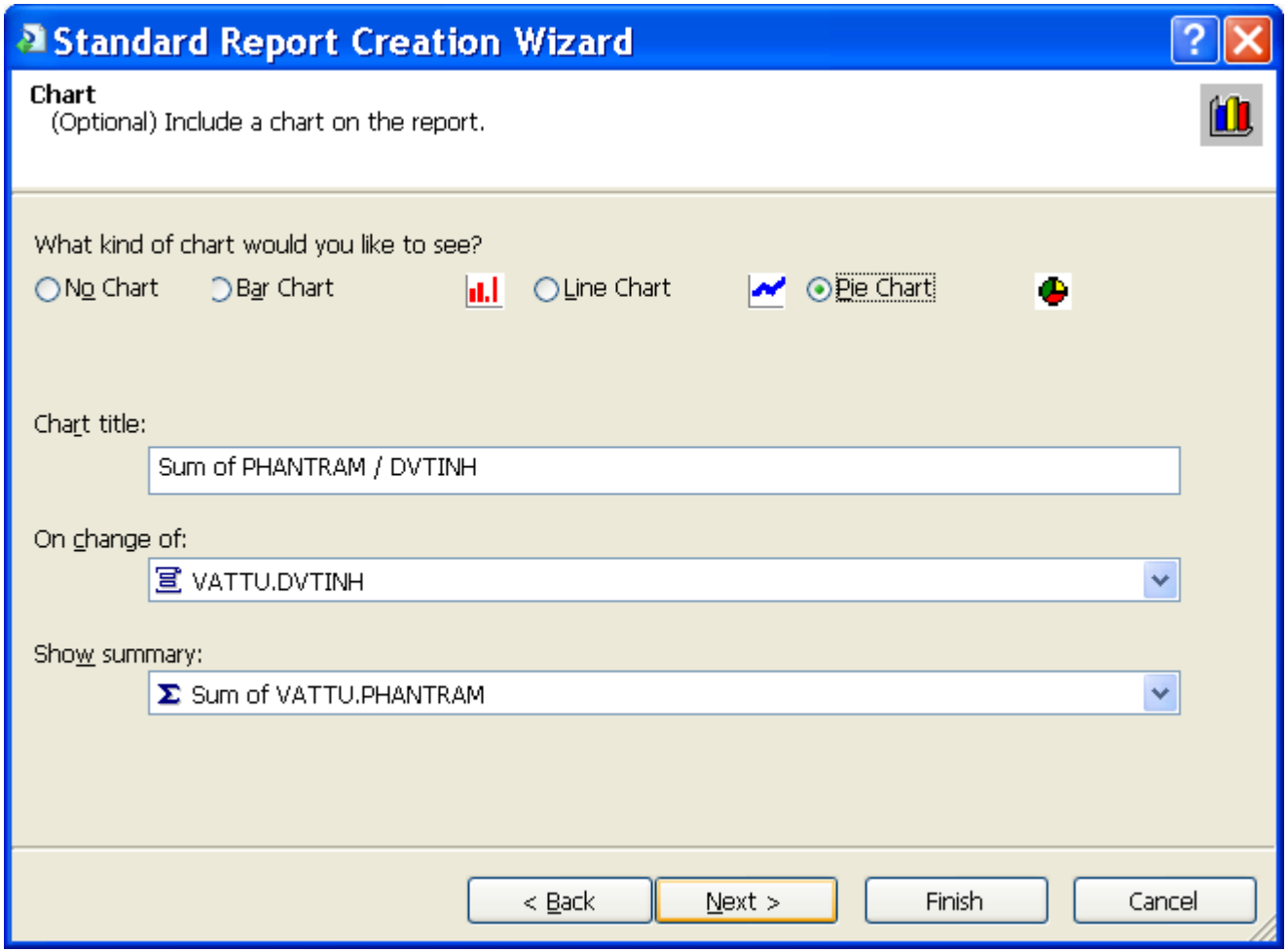


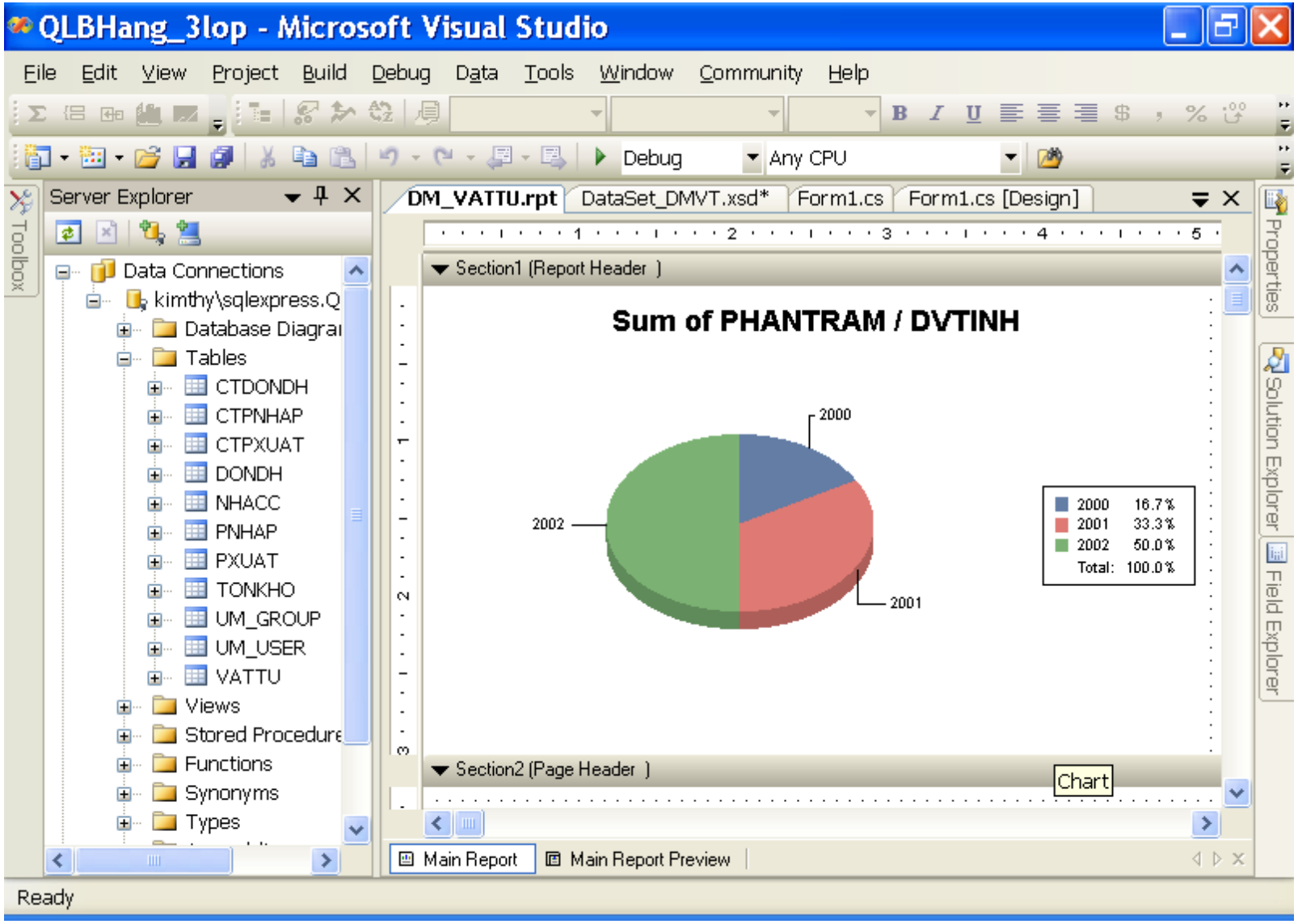

QLBHang_3lop - Microsoft Visual Studio
File Edit View Project Build Debug Data Tools Window Community Help
Standard Report Creation Wizard
Fields
Choose the information to display on the report.
Available Fields:
VATTU
Fields to Display:
VATTU.MAVTU
VATTU.TENVTU
VATTU.DVTINH
VATTU.PHANTRAM
VATTU.HINH
>
>>
<
<<
Browse Data...
Find Field...
< Back
Next >
Finish
Cancel
Server Explorer
Toolbox
start
Microsoft PowerP...
baitap
QLBHang_3lop - ...
16:20


Standard Report Creation Wizard
Grouping
(Optional) Group the information on the report.
Available Fields:
Report Fields
VATTU.MAVTU
VATTU.TENVTU
VATTU.DVTINH
VATTU.PHANTRAM
VATTU.HINH
VATTU
MAVTU
TENVTU
DVTINH
PHANTRAM
HINH
>
>>
<
<<
Group By:
VATTU.DVTINH - A
Browse Data...
Find Field...
in ascending order.
< Back
Next >
Finish
Cancel


Standard Report Creation Wizard
Summaries
(Optional) Add summary information to the report.
Available Fields:
Report Fields
VATTU.MAVTU
VATTU.TENVTU
VATTU.DVTINH
VATTU.PHANTRAM
VATTU.HINH
VATTU
MAVTU
TENVTU
DVTINH
PHANTRAM
HINH
>
>>
<
<<
Summarized Fields:
VATTU.DVTINH
Sum of VATTU.PHANTRAM
Browse Data...
Find Field...
< Back
Next >
Finish
Cancel


Standard Report Creation Wizard
Group Sorting
(Optional) Sort the groups based on the summarized totals.
Group that will be sorted:
VATTU.DVTINH
What kind of group ordering would you like to see?
None
Top 5 groups
Bottom 5 groups
Comparing summary values for the Top or Bottom groups:
< Back
Next >
Finish
Cancel


Standard Report Creation Wizard
Chart
(Optional) Include a chart on the report.
What kind of chart would you like to see?
No Chart
Bar Chart
Line Chart
Pie Chart
Chart title:
Sum of PHANTRAM / DVTINH
On change of:
VATTU.DVTINH
Show summary:
Sum of VATTU.PHANTRAM
< Back
Next >
Finish
Cancel


QLBHang_3lop - Microsoft Visual Studio
File Edit View Project Build Debug Data Tools Window Community Help
Debug
Any CPU
Server Explorer
Data Connections
kimthy\sqlexpress.Q
Database Diagrai
Tables
CTDONDH
CTPNHAP
CTPXUAT
DONDH
NHACC
PNHAP
PXUAT
TONKHO
UM_GROUP
UM_USER
VATTU
Views
Stored Procedure
Functions
Synonyms
Types
DM_VATTU.rpt
DataSet_DMVT.xsd*
Form1.cs
Form1.cs [Design]
Section1 (Report Header)
Sum of PHANTRAM / DVTINH
2000
2002
2001
2000 16.7%
2001 33.3%
2002 50.0%
Total: 100.0%
Section2 (Page Header)
Chart
Main Report
Main Report Preview
Ready
Toolbox
Properties
Solution Explorer
Field Explorer

2. Sừ dụng Crystal Reports Viewer để hiển thị báo biểu

Để xuất kết quả của báo biểu cho người dùng xem ta cần phải sử dụng một điều khiển hỗ trợ,gọi là CrystalReportViewer. Điều khiển này có sẵn trên thanh ToolBox của Visual Studio.Net,ta có thể thiết kế điều khiển lên một đối tượng Form và khai báo các thuộc tính để xuất báo biểu lên điều khiển.Ngoài ra ta có thể sử dụng code để thực hiện công việc hiển thị báo biểu trên CrystalReportViewer,khi sử dụng CrystalReportViewer ta cần phải chỉ ra vị trí của nó nằm trong namespace

Ví dụ sau đây khai báo một form và đối tượng CrystalReportViewer để hiển thị báo biểu DMVT.rpt

…

```
using  CrystalDecision.Windows.Forms
using System.Data.SqlClient;

//Khai báo các thành phần ADO.NET
        SqlConnection conn;
        SqlDataAdapter da;
        DataSet ds = new DataSet();
```

```
//Tạo kết nối và truy vấn dữ liệu
        String ketnoi = "Data Source=(Local);Initial
Catalog=QLVT;  Integrated Security=True";
        conn = new SqlConnection(ketnoi);
        String command = "SELECT * FROM VATTU";

//Đưa dữ liệu được truy xuất vào DataAdapter và DataSet
        da = new SqlDataAdapter(command, conn);
        da.Fill(ds, “VATTU");
```

```
//Tạo đối tượng report
        DMVR cr = new DMVT();

//Kết nối dữ liệu giữa DataSet và CrystalReport(DMVT) sau đó
gắn kết CrystalReport lên điều khiển CrystalReportViewer_VT

cr.SetDataSource(ds.Tables[0]);
crystalReportViewer_VT.ReportSource = cr;
```