BÀI TẬP THỰC HÀNH SỐ 5

THỰC HÀNH TẠO BÁO CÁO TRONG SQL SERVER

I. MỤC TIÊU

Kết thúc bài thực hành số 5, các nhóm phải hoàn thành các nhiệm vụ sau:

- Tạo các bảng History/Backup để backup dữ liệu định kỳ
- Tạo Trigger/Stored Procedure để update dữ liệu cho các bảng báo cáo
- Tạo báo cáo bằng **SQL Server Reporting Services (SSRS) hoặc Crystal Report**

II. CƠ SỞ LÝ THUYẾT

2.1. Tạo các bảng History/Backup để lưu các dữ liệu xóa logic

2.1.1. Xóa vật lý vs. xóa logic

Trong quá trình vận hành ứng dụng, người dùng sẽ vô tình hay cố ý chọn thao tác **Xóa dữ liệu**. Mặc định khi thiết kế, các thao tác xóa đều là Xóa vật lý (xóa hoàn toàn khỏi bộ nhớ / nơi lưu trữ). Tuy nhiên, trong một số trường hợp (ví dụ làm báo cáo, tra cứu lịch sử, …), chúng ta sẽ cần truy vấn lại các dữ liệu đã bị xóa để kiểm tra hay khôi phục lại. Vậy khi nào nên chọn xóa vật lý? Và khi nào nên chọn xóa logic?

- **Xóa vật lý**:
  - dữ liệu trong các bảng sẽ được xóa hoàn toàn khỏi bộ nhớ/nơi lưu trữ (bao gồm cả dữ liệu ở các bảng có contraint)
  - áp dụng với các dữ liệu không quan trọng, không cần tra cứu lại
  - không có quy định về thời gian lưu trữ
- **Xóa logic**:
  - dữ liệu được đánh dấu trạng thái Đã xóa (thường là sử dụng 1 trường deleteFlag – 0: chưa xóa, 1: đã xóa; và 1 trường deleteTime: datetime – lưu thời điểm xóa)
  - các dữ liệu quan trọng, cần tra cứu lại (ví dụ làm báo cáo, tra cứu lịch sử, …)
  - có quy định về thời gian lưu trữ (6 tháng, 1 năm, 3 năm, 5 năm, 10 năm,…)

2.1.2. Tạo các bảng History / Backup

2.1.2.1. Bảng History/Backup dùng với mục đích gì?

Dùng với mục đích backup dữ liệu, bao gồm:

- các dữ liệu bị xóa vật lý trong bảng gốc
- các dữ liệu cũ không còn sử dụng nhiều trong chu kỳ báo cáo hiện tại (ví dụ: các số liệu của các năm trước, trong trường hợp dữ liệu tăng rất nhanh, có thể backup đến các số liệu của tháng trước)

2.1.2.2. Các tạo bảng History/Backup

- có cấu trúc giống bảng gốc, nhưng có thêm 2 trường *deleteFlag* và *deleteTime*
- trong trường hợp dữ liệu nhiều, và để giúp tăng tốc độ truy cập, tra cứu, có thể tách ra thành nhiều bảng History/Backup ứng với cùng 1 bảng gốc
- cách đặt tên: *bk_Tên bảng gốc[_kỳ backup]*, trong đó *[_kỳ backup]* là tùy chọn, có thể là month/year.

2.1.2.3. Lưu trữ dữ liệu History/Backup đến bao giờ?

- Dữ liệu history/backup sẽ được lưu trữ theo quy định về thời gian lưu trữ (6 tháng, 1 năm, 3 năm, 5 năm, 10 năm,…)
- Hết thời gian này, các dữ liệu history/backup sẽ được xóa vật lý
- Việc xóa có thể thực hiện thủ công (bởi người quản trị DB), hoặc tự động (bằng trigger)

2.2. Stored Procedure vs. Trigger

2.2.1. Stored Procedure

Khi chúng ta tạo một ứng dụng với Microsoft SQL Server, ngôn ngữ lập trình T-SQL (Transact-SQL) là ngôn ngữ chính giao tiếp giữa ứng dụng và database của SQL Server.

Có hai phương pháp chính có thể dùng để lưu trữ và thực thi các chương trình tạo bằng Transact-SQL, đó là:

- lưu trữ các chương trình cục bộ dưới dạng file script và sử dụng các ứng dụng (như Query Analyzer) để gửi các lệnh đến SQL Server và xử lý các kết quả.
- lưu trữ các chương trình vào trong SQL Server dưới dạng Stored Procedure và tạo ứng dụng để gọi đến Stored Procedure đó.

Stored Procedure là một nhóm câu lệnh Transact-SQL đã được compiled (biên dịch) và chứa trong SQL Server dưới một tên nào đó và được xử lý như một đơn vị (chứ không phải nhiều câu SQL riêng lẻ).

2.2.1.1. Các đặc tính của Stored Procedure

- Là các chương trình trong SQL Server.
- Cho phép truyền các tham số đầu vào và chấp nhận trả về các giá trị chứa trong các tham số hoặc trả về các trạng thái giá trị để gọi những thủ tục hoặc thực hiện các xử lý theo lô để biết việc thực hiện thành công hay thất bại, nếu thất bại thì có thể đưa ra nguyên nhân thất bại.
- Bao gồm cả các lệnh gọi các thủ tục thực thi khác, chứa các lệnh SQL của chương trình để thực hiện các xử lý trong database.
- Ta có thể dùng Transact–SQL EXCUTE để thực thi các Stored Procedure. Stored Procedure khác với các hàm xử lý là giá trị trả về của chúng không chứa trong tên và chúng không được sử dụng trực tiếp trong biểu thức.
- So với các chương trình cục bộ, Stored Procedure có ưu điểm hơn là:
    - **Động**:
        - Stored Procedure cho phép điều chỉnh chương trình cho phù hợp: Chúng ta có chỉ tạo Stored Procedure một lần và lưu trữ trong database một lần, trong chương trình chúng ta có thể gọi nó với số lần bất kỳ. Stored Procedure có thể được chỉ rõ do một người nào đó tạo ra và sự thay đổi của chúng hoàn toàn độc lập với source code của chương trình.
        - Một khi Stored Procedure được tạo ra nó có thể được sử dụng lại. Điều này sẽ làm cho việc bảo trì (maintainability) dễ dàng hơn do việc tách rời giữa business rules (tức là những logic thể hiện bên trong Stored Procedure) và database. Ví dụ nếu có một sự thay đổi nào đó về mặt logic thì ta chỉ việc thay đổi code bên trong Stored Procedure mà thôi. Những ứng dụng dùng Stored Procedure này có thể sẽ không cần phải thay đổi mà vẫn tương thích với business rule mới.
        - Cũng giống như các ngôn ngữ lập trình khác Stored Procedure cho phép ta đưa vào các input parameters (tham số) và trả về các output parameters đồng thời nó cũng có khả năng gọi các Stored Procedure khác.
    - **Nhanh hơn**:

        - Khi thực thi một câu lệnh SQL thì SQL Server phải kiểm tra permission xem user gửi câu lệnh đó có được phép thực hiện câu lệnh hay không đồng thời kiểm tra cú pháp rồi mới tạo ra một execute plan và thực thi. Nếu có nhiều câu lệnh như vậy gửi qua network có thể làm giảm đi tốc độ làm việc của server. SQL Server sẽ làm việc hiệu quả hơn nếu dùng Stored Procedure vì người gửi chỉ gửi một câu lệnh đơn và SQL Server chỉ kiểm tra một lần sau đó tạo ra một execute plan và thực thi. Nếu Stored Procedure được gọi nhiều lần thì execute plan có thể được sử dụng lại nên sẽ làm việc nhanh hơn. Ngoài ra, cú pháp của các câu lệnh SQL đã được SQL Sever kiểm tra trước khi save nên nó không cần kiểm lại khi thực thi.
    - **Giảm thiểu bandwidth**:
        - Với một xử lý sử dụng Transact-SQL có tới hàng trăm câu lệnh đơn được đồng thời gửi đi dẫn tới tình trạng ngốn bandwidth hoặc có thể là quá tải. Stored Procedure cải thiện được vấn đề này bằng cách gửi theo trình tự xử lý. Đồng thời Stored Procedure còn có thể phân tích cú pháp và tối ưu hóa cậu lệnh trong lần thực thi đầu, giúp cải thiện câu lệnh tốt hơn.
    - **Bảo mật**:
        - Phân cấp quyền sử dụng cho các user, cấp quyền, giới hạn quyền cho các user thậm chí họ không được phép thực thi trực tiếp những Stored Procedure này. Khi đó sẽ hạn chế, loại bỏ các vấn đề xâm phạm dữ liệu không được cấp phép.
        - Giả sử chúng ta muốn giới hạn việc truy xuất dữ liệu trực tiếp của một user nào đó vào một số tables, ta có thể viết một Stored Procedure để truy xuất dữ liệu và chỉ cho phép user đó được sử dụng Stored Procedure đã viết sẵn mà thôi chứ không thể "đụng" đến các tables đó một cách trực tiếp.
        - Ngoài ra Stored Procedure có thể được encrypt (mã hóa) để tăng cường tính bảo mật.

2.2.1.2. Các loại Stored Procedure

Stored Procedure có thể được chia thành 5 nhóm như sau:

1. **System Stored Prcedure:**
    - Là những Stored Procedure chứa trong Master database và thường bắt đầu bằng tiếp đầu ngữ sp_.
    - Các Stored Procedure này thuộc loại built-in và chủ yếu dùng trong việc quản lý database (administration) và security.
    - Ví dụ bạn có thể kiểm tra tất cả các processes đang được sử dụng bởi user DomainName\Administrators, bạn có thể dùng sp_who @loginame='DomainName\Administrators'.
    - Có hàng trăm system Stored Procedure trong SQL Server. Bạn có thể xem chi tiết trong SQL Server Books Online.
2. **Local Stored Procedure**
    - Đây là loại thường dùng nhất.
    - Chúng được chứa trong user database và thường được viết để thực hiện một công việc nào đó. Thông thường người ta nói đến Stored Procedure là nói đến loại này.
    - Local Stored Procedure thường được viết bởi DBA hoặc programmer. Chúng ta sẽ bàn về cách tạo stored prcedure loại này trong phần kế tiếp.
3. **Temporary Stored Procedure**
    - Là những Stored Procedure tương tự như local Stored Procedure nhưng chỉ tồn tại cho đến khi connection đã tạo ra chúng bị đóng lại hoặc SQL Server shutdown.
    - Các Stored Procedure này được tạo ra trên TempDB của SQL Server nên chúng sẽ bị delete khi connection tạo ra chúng bị cắt đứt hay khi SQL Server down.
    - Temporary Stored Procedure được chia làm 3 loại: local (bắt đầu bằng #), global (bắt đầu bằng ##) và Stored Procedure được tạo ra trực tiếp trên TempDB.

- Loại local chỉ được sử dụng bởi connection đã tạo ra chúng và bị xóa khi disconnect, còn loại global có thể được sử dụng bởi bất kỳ connection nào.
- Permission cho loại global là dành cho mọi người (public) và không thể thay đổi.
- Loại Stored Procedure được tạo trực tiếp trên TempDB khác với 2 loại trên ở chỗ ta có thể set permission, chúng tồn tại kể cả sau khi connection tạo ra chúng bị cắt đứt và chỉ biến mất khi SQL Server shut down.

4. **Extended Stored Procedure**
   - Đây là một loại Stored Procedure sử dụng một chương trình ngoại vi (external program) vốn được compiled thành một DLL để mở rộng chức năng hoạt động của SQL Server.
   - Loại này thường bắt đầu bằng tiếp đầu ngữ xp_ .
   - Ví dụ:
     i. xp_sendmail dùng để gửi mail cho một người nào đó
     ii. xp_cmdshell dùng để chạy một DOS command… xp_cmdshell 'dir c:\' .
   - Nhiều loại extend Stored Procedure được xem như system Stored Procedure và ngược lại.
5. **Remote Stored Procedure**
   - Những Stored Procedure gọi Stored Procedure ở server khác.

2.2.1.3. Viết Stored Procedure

Tên và những thông tin về Stored Procedure khi được tạo ra sẽ chứa trong SysObjects table, còn phần text của nó chứa trong SysComments table.

Vì Stored Procedure cũng được xem như một object nên ta cũng có thể dùng các lệnh như CREATE, ALTER, DROP để tạo mới, thay đổi hay xóa bỏ một Stored Procedure.

2.2.1.3.1. Tạo Stored Procedure

Cú pháp:

```
CREATE PROCEDURE procedure_name
@parameter1 data_type [output] /*các tham số*/,
@parameter2 data_type [output]
AS
BEGIN
[khai báo các biến cho xử lý]
{Các câu lệnh transact-sql}
END
GO
```

Phần [output] là phần có thể có hoặc không để xác định loại tham số.

2.2.1.3.2. Thực thi Stored Procedure

Sử dụng lệnh EXECUTE (có thể viết tắt là EXEC) để thực thi một Stored Procedure.

```
EXECUTE procedure_name parameter_value1, parameter_value2,..
EXEC procedure_name parameter_value1, parameter_value2, ...
```

2.2.1.3.3. Xóa Stored Produre

```
DROP PROCEDURE procedure_name
DROP PROC procedure_name
```

2.2.1.4. Tham số trong Stored Procedure

Stored Procedure có thể có 2 loại tham số chính: tham số đầu vào và tham số đầu ra.

2.2.1.4.1. Tham số đầu vào

Đây là loại tham số mặc định, cho phép truyền các giá trị vào trong Stored Procedure để hỗ trợ xử lý.

Ví dụ:

```
CREATE PROC Cong
@So1 int,
@So2 int
AS
BEGIN
declare @Kq int
set @Kq = @So1 + @So2
```

```
print @Kq
END
GO
exec Cong 1, 2
```

Kết quả đoạn lệnh trên sẽ cho kết quả là "3"

2.2.1.4.2. Tham số đầu ra

Tham số dùng để nhận kết quả trả về từ Stored Procedure. Sử dụng từ khóa OUTPUT (hoặc viết tắt là OUT) để xác định tham số.

Ví dụ:

```
ALTER PROC Tru
@So1 int,
@So2 int,
@Kq int output
AS
BEGIN
set @Kq = @So1 - @So2
END
GO
```

2.2.1.5. Trả về giá trị trong Stored Procedure

Ngoài cách sử dụng tham số đầu ra để trả về giá trị, chúng ta có thể sử dụng RETURN để trả về giá trị từ Stored Procedure hoặc các câu lệnh SELECT khi truy vấn dữ liệu.

2.2.1.5.1. Trả về giá trị từ lệnh RETURN

Lệnh RETURN được sử dụng để trả về giá trị từ Stored Procedure mà không cần sử dụng tham số đầu ra. Giá trị trả về này có một số đặc điểm:

- Giá trị trả về chỉ có thể là số nguyên. Nếu trả về các loại giá trị khác thì lúc thực thi Stored Procedure sẽ báo lỗi (ngoại trừ 1 số kiểu dữ liệu được tự động chuyển đổi sang kiểu số nguyên như: float, double, ...).
- Giá trị trả về mặc định là 0.
- Có thể nhận giá trị trả về này bằng 1 biến.
- Sau khi gọi RETURN, Stored Procedure sẽ trả về giá trị và kết thúc xử lý.

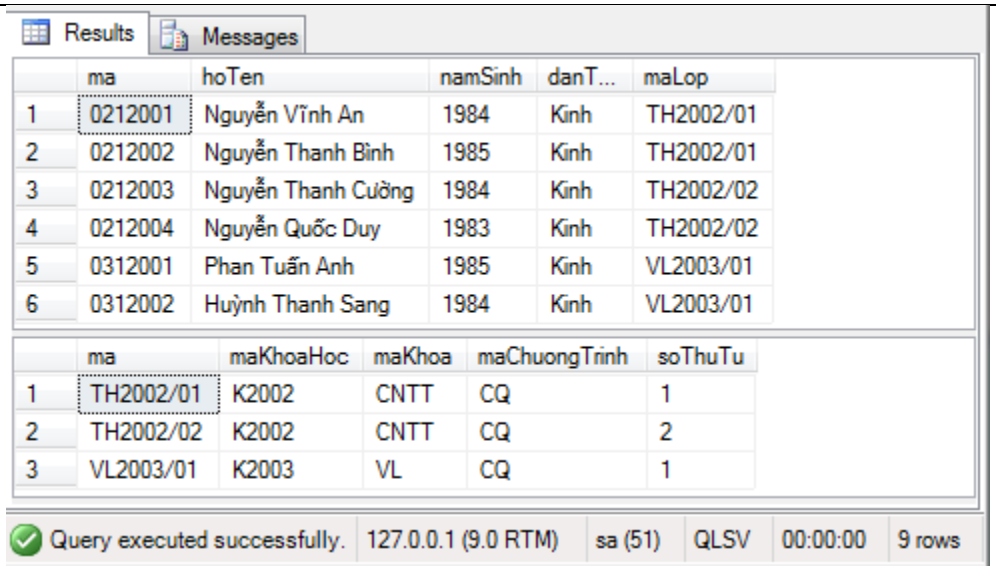
2.2.1.5.2. Trả về dữ liệu từ lệnh SELECT

Mỗi lệnh SELECT đặt trong Stored Procedure sẽ trả về 1 bảng.

Ví dụ:

```
CREATE PROC TestSelect
AS
BEGIN
SELECT * FROM SINHVIEN
SELECT * FROM LOP
END
GO
EXEC TestSelect
```

Kết quả in ra màn hình sẽ là:



Results | Messages

	ma	hoTen	namSinh	danT...	maLop
1	0212001	Nguyễn Vĩnh An	1984	Kinh	TH2002/01
2	0212002	Nguyễn Thanh Bình	1985	Kinh	TH2002/01
3	0212003	Nguyễn Thanh Cường	1984	Kinh	TH2002/02
4	0212004	Nguyễn Quốc Duy	1983	Kinh	TH2002/02
5	0312001	Phan Tuấn Anh	1985	Kinh	VL2003/01
6	0312002	Huỳnh Thanh Sang	1984	Kinh	VL2003/01

	ma	maKhoaHoc	maKhoa	maChuongTrinh	soThuTu
1	TH2002/01	K2002	CNTT	CQ	1
2	TH2002/02	K2002	CNTT	CQ	2
3	VL2003/01	K2003	VL	CQ	1

Query executed successfully. | 127.0.0.1 (9.0 RTM) | sa (51) | QLSV | 00:00:00 | 9 rows

2.2.1.6. *Kết hợp Stored Procedure với các lệnh T-SQL*

Các Stored Procedure thông thường được tạo ra nhằm giúp thực hiện một số chức năng cần thao tác trong cơ sở dữ liệu. Khi đó, ta cần phải kết hợp nhiều lệnh T-SQL thao tác với dữ liệu như (SELECT, INSERT, UPDATE, DELETE) và các cấu trúc điều khiển (IF, WHILE, CASE,...).

2.2.2. Trigger

Xem thêm:

- video: https://www.youtube.com/watch?v=MU36YoLpukQ
- tài liệu:
  - https://docs.microsoft.com/en-us/sql/t-sql/statements/create-trigger-transact-sql
  - http://docs.oracle.com/cd/B25329_01/doc/appdev.102/b25108/xedev_triggers.htm#BABGJFGH,
  - https://en.wikipedia.org/wiki/Database_trigger

Trigger có thể xem là một dạng đặc biệt của Stored Procedure, bởi vì bên trong nội dung của trigger lưu trữ các câu lệnh dùng để thực hiện một số hành động nào đó mà người lập trình sẽ chỉ ra.

Giống như Stored Procedure, trigger có thể được lồng đến 32 cấp và có thể được kích hoạt đệ quy.

Tuy nhiên, khác với Stored Procedure, trigger:

- hoàn toàn không có tham số.
- được thực hiện tự động khi dữ liệu của bảng có liên quan đến trigger bị cập nhật mà không cần sự can thiệp bằng các thao tác thủ công như kiểm tra dữ liệu, đồng bộ hóa dữ liệu, …
- Một trigger gắn với duy nhất một đối tượng table hoặc một đối tượng view.

2.2.2.1. *Các trường hợp sử dụng Trigger*

Chính nhờ vào tính năng đặc biệt là tự động thực hiện mà nội dung các lệnh bên trong trigger được dùng cho các công việc sau:

- Kiểm tra các ràng buộc toàn vẹn dữ liệu phức tạp.
- Thực hiện các xử lý được thiết kế thi hành tại server (trong mô hình client/server). Các xử lý mà ta muốn chúng sẽ được tự động thực hiện khi có thao tác INSERT, UPDATE hoặc DELETE xảy ra:

  - Khi có sự thay đổi dữ liệu trên 1 table và chúng ta muốn dữ liệu trên 1 hay nhiều table khác cũng tự động thay đổi theo cho phù hợp.
- Trigger cũng được dùng để thay thế các constraint trong trường hợp ta muốn việc kiểm tra ràng buộc dữ liệu kèm theo các câu thông báo thích hợp theo ý muốn người dùng (thay vì hiển thị thông báo mặc định của hệ thống – thường rất khó hiểu và có thể gây hiểu nhầm đối với người sử dụng).
- Trigger còn được sử dụng trong cơ chế backup tự động, phòng khi cần rollback.

2.2.2.1.1. Kiểm tra ràng buộc toàn vẹn dữ liệu với Trigger

Trên SQL Server, có 2 cách thực hiện ràng buộc toàn vẹn dữ liệu:

- **Ràng buộc toàn vẹn dữ liệu bằng phương pháp mô tả (declarative data integrity):**
  - Việc ràng buộc toàn vẹn dữ liệu chính là xác định ràng buộc khóa chính, khóa ngoại, kiểm tra miền giá trị, … và mô tả chúng tại thời điểm tạo table.
  - Đặc điểm của phương pháp này là việc kiểm tra sẽ được thực hiện trước khi cho phép ghi vào table.
- **Ràng buộc toàn vẹn dữ liệu bằng phương pháp theo thủ tục (procedural data integrity):**
  - việc ràng buộc toàn vẹn dữ liệu được xác định bởi tập hợp các câu lệnh T-SQL. Các câu lệnh chứa bên trong đối tượng Trigger sẽ được gọi thi hành khi có thao tác thêm, xóa, hoặc sửa dữ liệu xảy ra trên table tương ứng.
  - Đặc điểm của phương pháp này là việc kiểm tra sẽ được thực hiện sau khi dữ liệu được ghi vào table.
  - Ta chỉ sử dụng trigger khi mà các biện pháp bảo đảm ràng buộc dữ liệu khác như Constraints không thể thỏa mãn yêu cầu của ứng dụng.

2.2.2.1.2. Thực hiện các xử lý được thiết kế thi hành tại server khi có thao tác thay đổi dữ liệu trên các bảng của CSDL

Có 3 thao tác cơ bản làm thay đổi dữ liệu trên các bảng của CSDL, đó là thao tác thêm (insert), thao tác sửa (update) và thao tác xóa (delete). Như vậy, để đảm bảo dữ liệu nhất quán và đúng đắn, ta cần kiểm tra việc thực hiện của 3 thao tác này.

Có 2 cách kiểm tra: kiểm tra mức giao diện và kiểm tra mức cơ sở dữ liệu.

- **Kiểm tra mức giao diện**: công việc này chính là công việc lập trình trên các màn hình giao diện.
- **Kiểm tra mức cơ sở dữ liệu**: công việc này được thực hiện bởi các đối tượng constraint hoặc trigger.
  - **Đối với các ràng buộc toàn vẹn dữ liệu đơn giản** (như kiểm tra các ràng buộc miền giá trị, kiểm tra các ràng buộc giữa các thuộc tính trên cùng một bảng dữ liệu,…): nên sử dụng đối tượng constraint.
  - **Đối với các ràng buộc toàn vẹn dữ liệu phức tạp khác** (là những quy tắc được định nghĩa dùng để kiểm tra tính toàn vẹn của dữ liệu trên nhiều cột hoặc nhiều dòng của các bảng khác nhau): bắt buộc phải sử dụng đối tượng trigger. Đối tượng này cho phép chúng ta xây dựng các câu lệnh bên trong nó với mục tiêu là các câu lệnh này sẽ được thực hiện khi các thao tác làm thay đổi dữ liệu xảy ra trên bảng dữ liệu mà nó gắn.

2.2.2.1.3. Dùng Trigger để kiểm soát và xuất các câu thông báo thích hợp cho người dùng

Quan điểm cơ bản cần nghĩ đến trước tiên là chỉ những ràng buộc dữ liệu nào không thể dùng phương pháp mô tả (declarative data integrity) thì mới dùng trigger để giải quyết. Tuy nhiên, bạn không nhất thiết phải tuân thủ theo quan điểm này. Trên thực tế bạn hoàn toàn có thể dùng trigger để thay thế cho constraint trong trường hợp bạn muốn tự mình kiểm soát và cho ra các câu thông báo thích hợp cho người dùng.

Ví dụ: Chúng ta có các table có cấu trúc sau:

```
Create table DON_GIA_HANG_HOA
(
	ID_DGHH Numeric Identity(1,1) Primary Key,
	ID_HH varchar(50) Not null,
	Ngay_DGHH Datatime,
	Gia_DGHH Numeric,
	HienHanh Tinyint Default 1
	Foreign Key (ID_HH) References DM_HANG_HOA(ID_HH),
	Check(Gia_DGHH>0)
)

Create Table DM_HANG_HOA
(
	ID_HH varchar(50) primary key,
	Ten_HH varchar(50),
	DonGiaHienHanh numeric not null
	Check(DonGiaHienHanh>0)
)
```

- **Yêu cầu**: Khi thêm mới mẫu tin trên table DON_GIA_HANG_HOA cần update lại mục DonGiaHienHanh trên table DM_HANG_HOA sao cho DM_HANG_HOA.DonGiaHienHanh = DON_GIA_HANG_HOA.Gia_DGHH
- **Giải pháp**: Với các yêu cầu ràng buộc dữ liệu này, ta cần phải dùng trigger để thực hiện.

Trên thực tế đôi khi người lập trình muốn thay đổi cấu trúc lưu trữ của dữ liệu với mục tiêu tăng tốc độ xử lý hoặc việc xử lý tính toán dễ dàng hơn, điều này thường dẫn đến việc phá vỡ tính chuẩn của cơ sở dữ liệu và tính ràng buộc toàn vẹn dữ liệu (nghĩa là sẽ có một số dữ liệu được chứa cùng một lúc ở hai hay nhiều nơi khác nhau). Khi đó, ta có thể sử dụng Trigger để đảm bảo tính chính xác thì khi dữ liệu được cập nhật ở một table này thì cũng phải được cập nhật tự động ở các table còn lại.

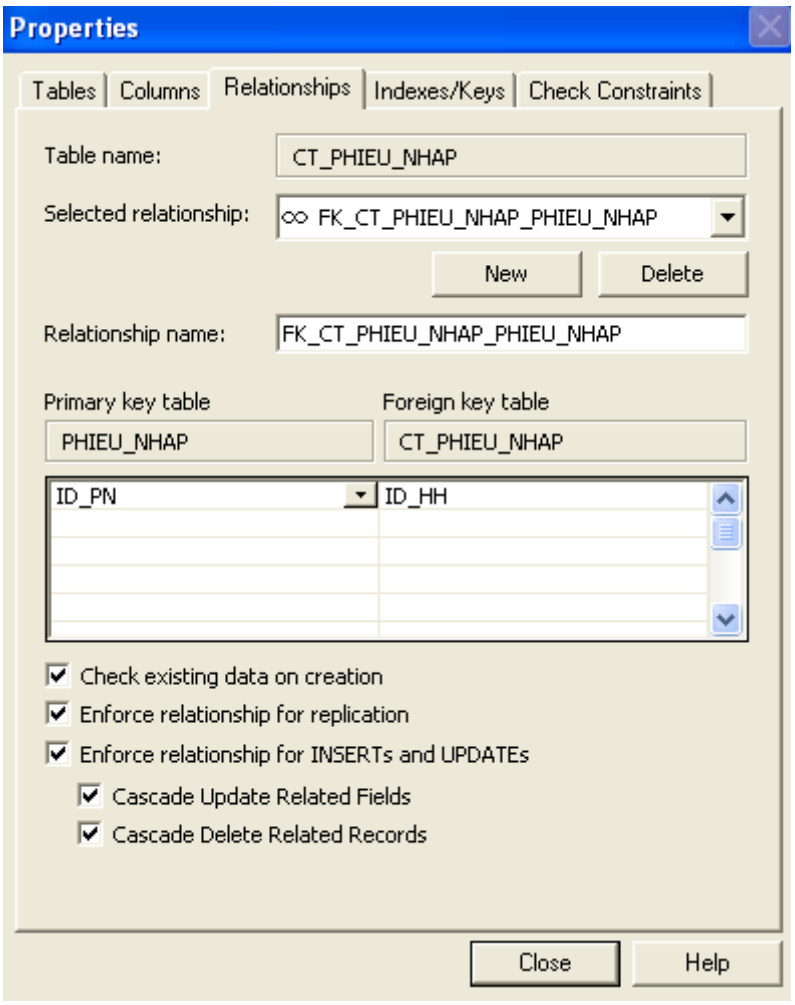
Ví dụ, ta có trường hợp sau:

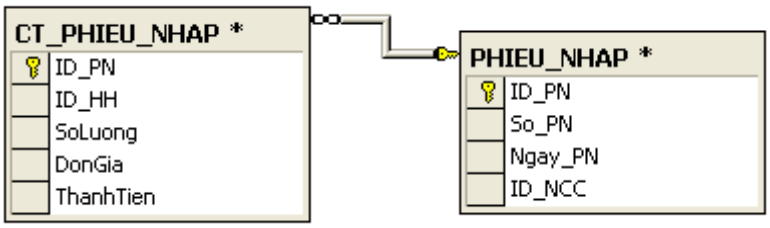
```
Create Table PHIEU_XUAT
(
	ID_PX numeric identity(1,1) primary key,
	So_PX varchar(50) unique,
	Ngay_PX datatime,
	ID_KHG varchar(50),
	TongSoTien numeric,
	Foreign key(ID_KHG) References KHACH_HANG(ID_KHG)
)
```

- Cột **TongSoTien** được bổ sung vào table **PHIEU_XUAT** với ý nghĩa dùng để chứa giá trị tổng số tiền của phiếu xuất. Giá trị này bằng tổng các giá trị chứa trên cột **ThanhTien** của cùng phiếu xuất trên table **CT_PHIEU_XUAT**.
- Vấn đề khó khăn ở đây là khi thêm, xóa mẫu tin hoặc sửa đổi giá trị của cột **ThanhTien** trên **CT_PHIEU_XUAT**, tính đúng đắn của cột **TongSoTien** trên **PHIEU_XUAT** phải luôn được đảm bảo.
- Trong tình huống này, cách duy nhất là sử dụng trigger để tự động cập nhật giá trị.

2.2.2.2. Các đặc trưng và hạn chế của Trigger

- Một trigger có thể xây dựng bên trong nó các câu lệnh tác động lên cột của table bất kỳ trong cơ sở dữ liệu hoặc tới các đối tượng bên ngoài cơ sở dữ liệu hiện hành.
- Có thể áp dụng trigger cho đối tượng View.
- Trigger chỉ có thể được kích hoạt một cách tự động bởi một trong các biến cố Insert, Update, Delete mà không thể chạy manually được.
- Một trigger có thể thực hiện nhiều hành động (multiple actions), và nó có thể được kích hoạt bởi nhiều hơn 1 biến cố.
- Trigger không thể được tạo trên 1 table tạm (temporary table, là table được tạo với tên table có ký tự # hoặc ## phía trước) hoặc table hệ thống (system table). Tuy nhiên cần chú ý rằng các câu lệnh bên trong trigger hoàn toàn có thể tham chiếu đến nội dung bên trong các table tạm và table hệ thống.
- Một hạn chế quan trọng mà bạn cần quan tâm là các trigger loại **INSTEAD OF DELETE** và **INSTEAD OF UPDATE** không thể định nghĩa trên các table có chứa khóa ngoại và trên dây quan hệ nối từ table chứa nó thông qua khóa ngoại đến table khác đã có thiết đặt tương ứng trên các dây quan hệ tính chất **Cascade Delete Related Records** và **Cascade Update Related Fieds**.
- Ví dụ:





Trong mối quan hệ ở hình trên, nếu ta thiết đặt tính Cascade Delete Related Records của dây quan hệ thì việc tạo trigger loại Instead Of Delete sẽ không thể thực hiện thành công.

2.2.2.3. Cách tạo Trigger

Cú pháp để tạo một Trigger cơ bản như sau:

```
1   CREATE TRIGGER trigger_name
2   ON {table | view} -- Chỉ định bảng hoặc view sử dụng Trigger
3   {FOR | AFTER | INSTEAD OF}
4   {[ INSERT ] [,] [ UPDATE ] [,] [ DELETE ]} -- Các biến cố tự động kích hoạt Trigger
5   AS {sql_statement,...}
```

2.2.2.3.1. Tham số {FOR | AFTER | INSTEAD OF}

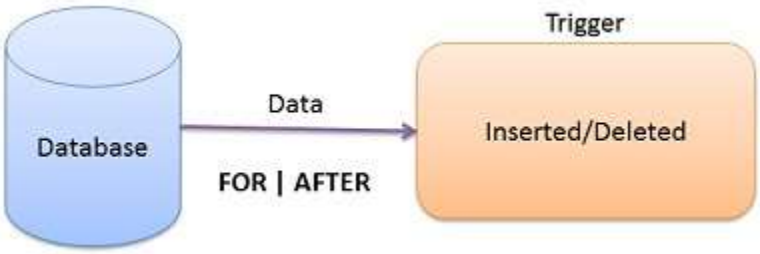
Khi thực hiện một Trigger thì SQL tự động tạo ra 2 bảng **Inserted** và **Deleted** trong bộ nhớ chính và cục bộ cho mỗi Trigger, có nghĩa là khi áp dụng Trigger trên bảng nào thì bảng Inserted và Deleted sẽ được sử dụng riêng cho đó bảng đó.

Cấu trúc 2 bảng **Inserted** và **Deleted** được tạo ra sẽ giống hệt cấu trúc của bảng mà Trigger đang thực thi và chúng chỉ tồn tại trong thời gian Trigger đó thực thi mà thôi.

Trong ví dụ trên: 2 bảng Inserted và Deleted có cấu trúc giống với bảng CTHD (Bảng mà Trigger đang thực thi) gồm các cột: SoHD, MaSP, SL

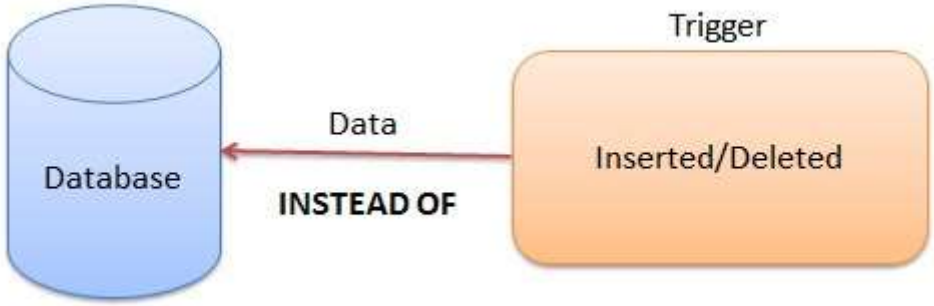
- **Inserted** là bảng chứa các dòng dữ liệu vừa được Insert hay Update vào bảng mà Trigger đang thực thi.
- **Deleted** là bảng chứa các dòng dữ liệu mới được xóa khỏi bảng bằng thao tác Delete hay Update.
- Khi thực hiện thao tác Update, thì đồng nghĩa với việc sẽ xóa những dòng dữ liệu cũ và thêm những dòng dữ liệu mới, khi đó tác Update sẽ vừa đồng thời thêm dữ liệu là các dòng mới vào 2 bảng Inserted và Deleted

2.2.2.3.2. Tham số FOR | AFTER



- Đối với tham số **For | After** thì Trigger sẽ được gọi **sau** khi có thao tác Insert hoặc Update.
- Thứ tự thực hiện là từ Database rồi đến bảng Inserted/Deleted
    - dữ liệu vừa mới Insert/Update vào sẽ nằm trong cả 2 bảng: bảng chính trong Database và bảng Inserted.
    - Khi thực hiện xóa một dòng dữ liệu thì dòng dữ liệu trên Database sẽ bị xóa, sau đó dòng bị xóa sẽ được thêm vào bảng Deleted.

2.2.2.3.3. Tham số INSTEAD OF



Khi sử dụng tham số **Instead of** thì Trigger sẽ bỏ qua việc tác động tới CSDL, thay vào đó nó thực hiện việc lưu dữ liệu vào bảng **Inserted** khi có thao tác Insert, lưu dữ liệu vào bảng **Deleted** đối với thao tác Delete.

Vì vậy, khi một dòng dữ liệu được thêm vào nó chỉ chứa trong bảng Inserted, và khi xóa một dòng dữ liệu thì nó đồng thời chứa trong bảng Deleted và vẫn còn tồn tại trong Database.
Trigger Instead of thường được dùng để cập nhật khung nhìn (View).

2.2.2.3.4. SQL Statement

Nội dung Trigger bao gồm các câu lệnh dùng để thực thi Trigger

2.2.2.4. Ví dụ minh họa về cách cài đặt và sử dụng về Trigger trên một dòng dữ liệu

Sử dụng lược đồ CSDL như sau:

```
KHACHHANG (MAKH, HOTEN, DCHI, SODT, NGSINH, DOANHSO, NGDK)
NHANVIEN (MANV,HOTEN, NGVL, SODT)
SANPHAM (MASP,TENSP, DVT, NUOCSX, GIA)
HOADON (SOHD, NGHD, MAKH, MANV, TRIGIA)
CTHD (SOHD,MASP,SL)
```

Ràng buộc: Ngày mua hàng (NGHD) của một khách hàng thành viên sẽ lớn hơn hoặc bằng ngày khách hàng đó đăng ký thành viên (NGDK).

```
CREATE TRIGGER CHECK_NGAYNV --Tên Trigger
ON HOADON
FOR UPDATE,INSERT
AS
    IF UPDATE(NGHD) --Kiểm tra việc cập nhật trên cột
    BEGIN
    DECLARE @NGHD SMALLDATETIME, @NGVL SMALLDATETIME
    SET @NGHD=(SELECT NGHD FROM INSERTED)
    SET @NGVL=(SELECT NGVL FROM NHANVIEN A,INSERTED B WHERE A.MANV=B.MANV)
    IF(@NGHD<@NGVL)
        BEGIN
        PRINT 'NGHD PHAI LON HON NGVL'
        ROLLBACK TRAN -- Câu lệnh quay lui khi thực hiện biến cố không thành công
        END
        END
```

+ Sử dụng cú pháp **IF UPDATE** để kiểm tra sự thay đổi trên cột, Trigger sẽ tự động thực hiện khi có thay đổi trên cột nhất định nào đó, thay vì chỉ định Trigger kích hoạt trên cả bảng.

```
IF UPDATE(Column_List) --Có thể kiểm tra trên một hay nhiều cột
```

+ Khai báo một biến cùng kiểu dữ liệu ta thực hiện câu lệnh:

```
DECLARE @Tên_biến Kiểu dữ liệu
DECLARE @NGHD SMALLDATETIME
```

+ Thiết lập giá trị cho một biến

```
SET @NGHD=(SELECT NGHD FROM INSERTED)
```

Thiết lập giá trị cho một biến có thể là một biểu thức hoặc một câu truy vấn.

Ở đây tôi sử dụng tham số **FOR** vì thế tôi có thể dễ dàng **Select** cột **NGHD** từ bảng **Inserted** mà không phải là bảng **HOADON**, như thế việc truy suất sẽ nhanh hơn vì ở đây bảng **Inserted** chỉ chứa dòng dữ liệu mới thêm vào, mà ta chỉ quan tâm đến những dòng dữ liệu mới thêm vào mà thôi.

+ Câu lệnh **PRINT** dùng để in thông báo lỗi ra màn hình.

```
PRINT 'THONG BAO LOI'
```

Hoặc sử dụng câu lệnh **RAISERROR**:

```
RAISERROR ('THONG BAO LOI')
```

Khi thực hiện một tác vụ (transaction) không thành công thì sẽ tự động quay lui (Rolled Back) bằng cách chèn thêm câu lệnh:

```
ROLLBACK TRAN
```

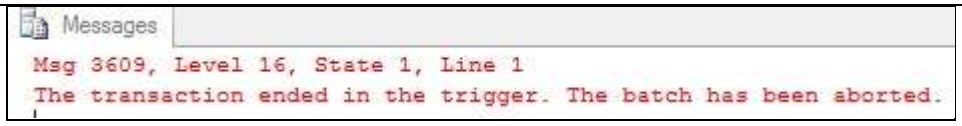
Câu lệnh RollBack Tran còn thực hiện trong việc định nghĩa một Trigger cho biến cố Delete không cho xóa một dòng dữ liệu.

VD: Tạo Trigger cho biến cố Delete như sau

```
CREATE TRIGGER CANNOT_DELETE
ON HOADON
FOR DELETE
AS
   ROLLBACK TRAN
```

Khi thực hiện xóa một dòng dữ liệu sẽ có thông báo lỗi

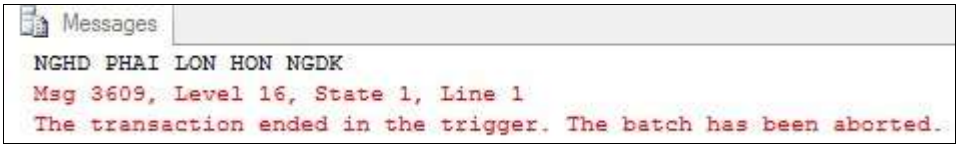
```
DELETE FROM HOADON WHERE MANV='NV01'
```



Sau khi cài đặt Trigger CHECK_NGAYNV, thử thêm một dòng dữ liệu có NGHD<NGDK:

```
1   INSERT INTO HOADON VALUES(1001,'23/07/2006','KH01','NV01',320000) -- Giả sử ngày
    DK là 27/07/2006
```

Ở đây NGHD=23/07/2006<27/07/2006
Kết quả:



2.3. Tạo báo cáo trong SQL Server

2.3.1. Tạo báo cáo bằng SQL Server Reporting Services (SSRS)

Bộ phần mềm SQL Server 2005, 2008 và 2008 R2 đều được đóng gói sẵn với SQL Server Reporting Services (SSRS) – một giải pháp tạo báo cáo chuyên dụng dành cho các doanh nghiệp. Với SSRS, bạn hoàn toàn có thể tạo, xuất bản, và quản lý số lượng bản báo cáo vô cùng lớn từ nhiều nguồn dữ liệu khác nhau.

2.3.1.1. Các thành phần chính của SSRS bao gồm:

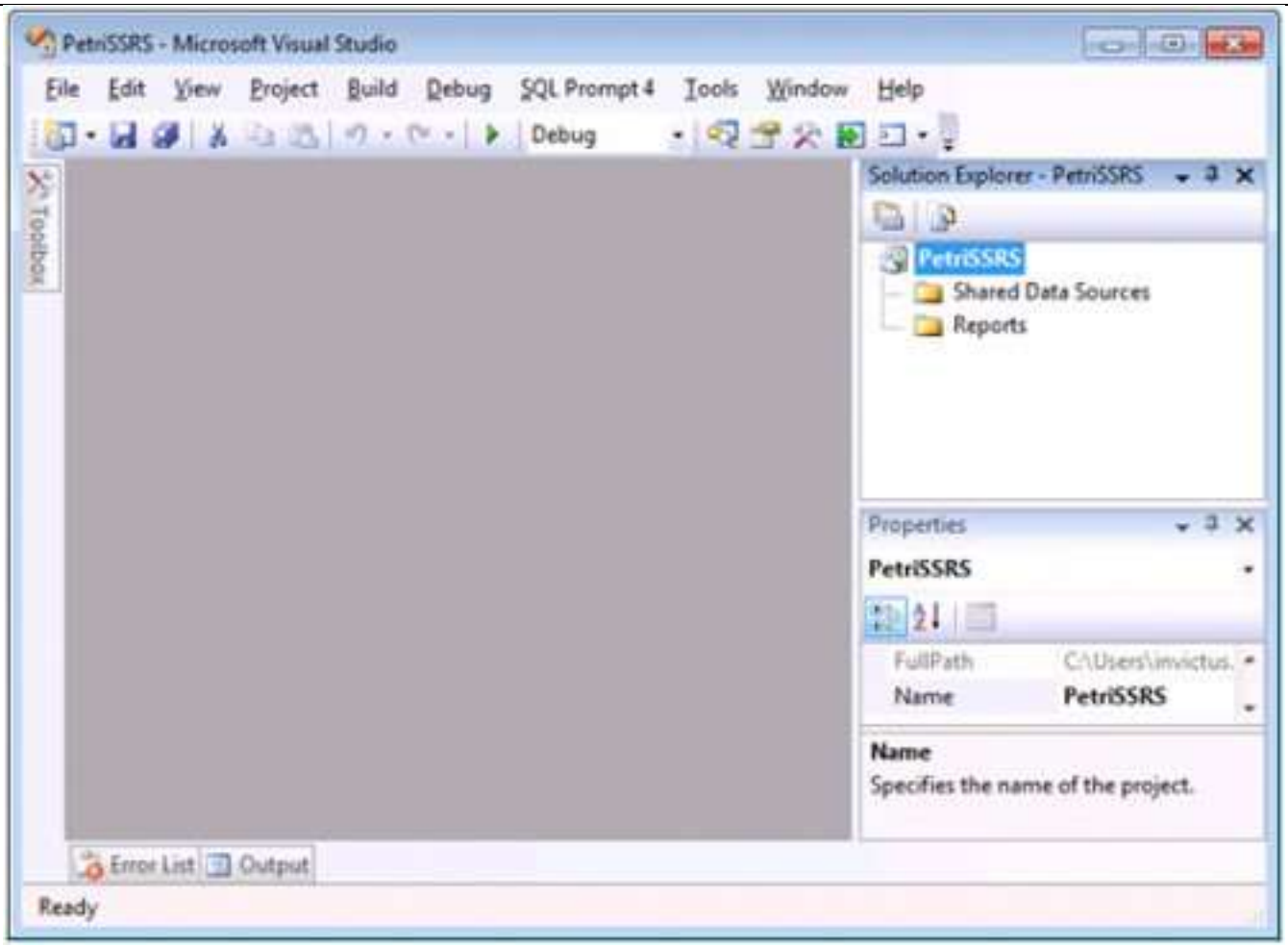
- *Report Server:* báo cáo về quá trình xử lý của các cơ sở dữ liệu và thành phần chính
- *Report Designer:* "môi trường" chính để tạo các báo cáo, hoạt động dựa trên *Visual Studio* của *SQL Server* và *Business Intelligence Development Studio (BIDS)*
- *Report Manager:* công cụ chung dựa trên nền tảng web dùng để quản lý các bản báo cáo, chức năng bảo mật, nguồn dữ liệu, mục góp ý… và nhiều tính năng khác.

Các phần sau sẽ hướng dẫn cách tạo một project SSRS mới, chia sẻ dữ liệu gốc, và báo cáo chính dựa trên nền cơ sở dữ liệu AdventureWorks.

2.3.1.2. Tạo mới project SSRS trong BIDS:

Khái niệm project – dự án bao gồm tất cả các object – đối tượng, thành phần. Ví dụ: các báo cáo hoặc gói SSIS – bên trong BIDS. Và mỗi một project lại dựa theo những phân loại nhất định.
Tại Start menu, các bạn chọn *Programs* > *Microsoft SQL Server 2008* (2005 hoặc bất cứ phiên bản nào) > *SQL Server Business Intelligence Development Studio*.

Chọn tiếp *File* > *New* > *Project* và *Report Server Project* từ danh sách *Business Intelligence Projects*, gõ tên cho project và nhấn OK:



Đây là 1 project "rỗng", nghĩa là chưa có gì cả. Nếu nút *Solution Explorer* không sử dụng được, các bạn chọn thẻ *Solution Explorer* ở phía bên phải hoặc *View* > *Solution Explorer*, cửa sổ tiếp theo sẽ hiển thị tên của project và tất cả các thành phần, đối tượng có liên quan.

2.3.1.3. Tạo bộ dữ liệu gốc chia sẻ

Nguồn dữ liệu SSRS có thể được "nhúng" kèm trong các báo cáo, hoặc chia sẻ giữa nhiều báo cáo của 1 project. Tùy từng loại báo cáo mà chúng ta sẽ chia sẻ dữ liệu gốc này theo cách khác nhau.

Ví dụ, nếu nguồn dữ liệu thay đổi, hoặc toàn bộ cơ sở dữ liệu được chuyển tới 1 server khác, người sử dụng chỉ cần thay đổi lại nguồn của dữ liệu chia sẻ, trong đó có bao gồm hàng chục, hàng trăm các báo cáo riêng rẽ được bao gồm bên trong đó.

Để tạo nguồn dữ liệu chia sẻ:

- Bên trong cửa sổ *Solution Explorer*, các bạn kích chuột phải và chọn *Shared Data Sources* > *Add New Data Source*.
- Đặt tên cho cơ sở dữ liệu, lưu ý rằng không được để khoảng trống tại đây.
- Chọn thể loại cơ sở dữ liệu phù hợp từ danh sách, ví dụ tại đây là *SQL Server*.
- Nhấn Edit để điền các chi tiết về giao thức kết nối.
- Tại cửa sổ *Connection Properties*: điền tên server, thông tin nhận dạng, và cơ sơ dữ liệu.
- Nhấn *Test Connection* để kiểm tra và *OK* khi hoàn tất.

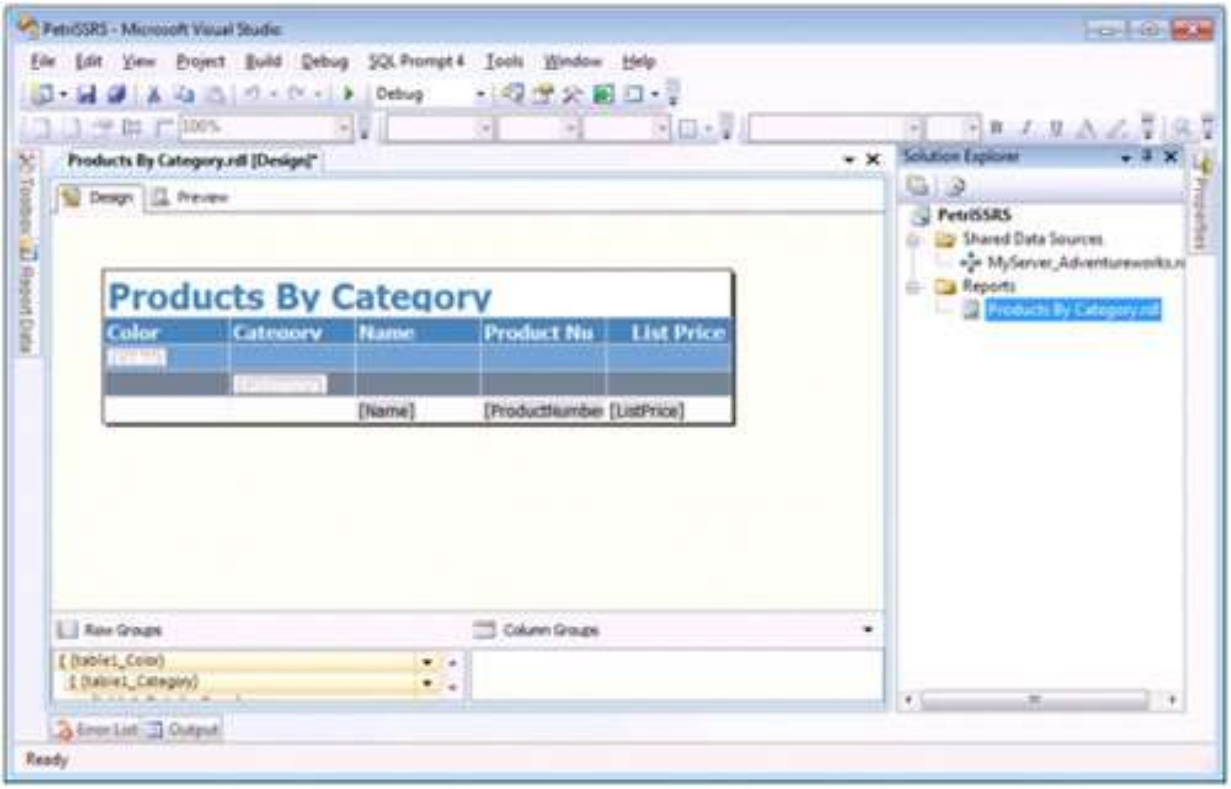
2.3.1.4. Các bước tạo báo cáo

- Trong *Solution Explorer*, kích chuột phải vào *Reports* > *Add New Report*.
- Tại màn hình *Select the Data Source*, chọn nguồn cơ sở dữ liệu vừa tạo ở bước trên.
- Trong màn hình *Design the Query:* gõ các câu lệnh truy vấn, hoặc dùng chức năng *Query Builder* để xây dựng với giao diện đồ họa. Trong bài thử nghiệm này, các bạn dựa theo đoạn mã truy vấn sau:

```
SELECT P.Name ,
ProductNumber ,
Color ,
ListPrice ,
```

```
SC.Name [Category]
FROM Production.Product P
LEFT OUTER JOIN Production.ProductSubCategory SC
ON P.ProductSubCategoryID = SC.ProductSubCategoryID
WHERE P.ProductSubCategoryID IS NOT NULL
AND P.Color IS NOT NULL
ORDER BY Category, ListPrice ASC
```

- Trên màn hình *Select the Report Type*, chọn *Tabular*. Lưu ý rằng định dạng theo kiểu *Tabular* sẽ trông giống như một bảng tính thông thường (hoặc bảng cơ sở dữ liệu), với các cột ở đầu trang, và số dòng phụ thuộc vào lượng dữ liệu của dataset. Sau đó nhấn *Next*.
- Tiếp đến bước *Design the Table*, thêm *Color* và *Category* vào trong mục *Group*, hãy thêm các trường còn lại vào mục *Details*, nhấn *Next*.
- Giữ nguyên lựa chọn *Stepped* tại bước *Choose the Table Layout* và *Next*.
- Tiếp theo, chọn kiểu cho bảng báo cáo và tiếp tục *Next*.
- Cuối cùng là đặt tên (ví dụ *Products By Category*) và *Finish*:



Trong thẻ *Design*, các bạn có thể chỉnh lại báo cáo theo một số cách sau:

- Thay đổi kích thước của bản báo cáo cùng các thành phần bằng thao tác kéo thả.
- Thêm các đối tượng khác như ảnh, từ thanh công cụ Toolbox.
- Định dạng lại ký tự và chữ số (nhấn chuột phải và chọn Text Box Properties).
- Thêm, di chuyển, xóa, hoặc thay đổi thuộc tính dữ liệu.

Những thông tin cơ bản trên chỉ là bước khởi đầu để giới thiệu về các chức năng của SSRS. Để nắm bắt nhiều chi tiết hơn về SQL Server Reporting Services, hãy tham khảo thêm tại đường dẫn sau đây: SQL Server Books Online, SQL Server Books Online Tutorials, Microsoft Press - Stacia Misner và CodePlex.

Xem thêm file: **Lab5- Tao bao cao bang SQL Server Reporting Service (SSRS).pdf**

2.3.2. Tạo báo cáo bằng Crystal Report trong Visual Studio

Crystal Report cho phép người dùng kết nối dữ liệu và tạo báo cáo động. Trong chế độ Database Expert, người dùng có thể chọn và liên kết các bảng từ nhiều nguồn dữ liệu khác nhau, bao gồm bảng tính Microsoft Excel, CSDL Oracle, SQL Server, Access, các view nghiệp vụ của BusinessObjects Enterprise, và các thông tin lưu trữ trên file.

Crystal Report còn cho phép người dùng nhóm dữ liệu kết xuất thành nhóm riêng, và có thể chia dữ liệu nhóm thành các miền dữ liệu nhỏ hơn dựa trên các điều kiện. Crystal Report cũng hỗ trợ các việc tạo các subreport, vẽ đồ thị và một số tính năng của hệ thống thông tin địa lý (GIS).

Crystal Report hỗ trợ kết nối tới các nguồn dữ liệu sau:

- CSDL: PostgreSQL, Sybase, IBM DB2, Ingres, Microsoft Access, Microsoft SQL Server, MySQL, Interbase, Btrieve, Informix và Oracle
- Bảng tính Microsoft Excel
- File: text, XML
- Các phần mềm văn phòng: Lotus Notes, Microsoft Exchange and Novell GroupWise
- APIs: ODBC, OLE DB
- SAP: BW, Info Sets, Tables, và BusinessObjects Universes

Tham khảo thêm link sau: http://www.tutorialspoint.com/crystal_reports/

2.3.2.1. Down load ứng dụng Crystal Report

Crystal Reports Developer Edition cho Visual Studio

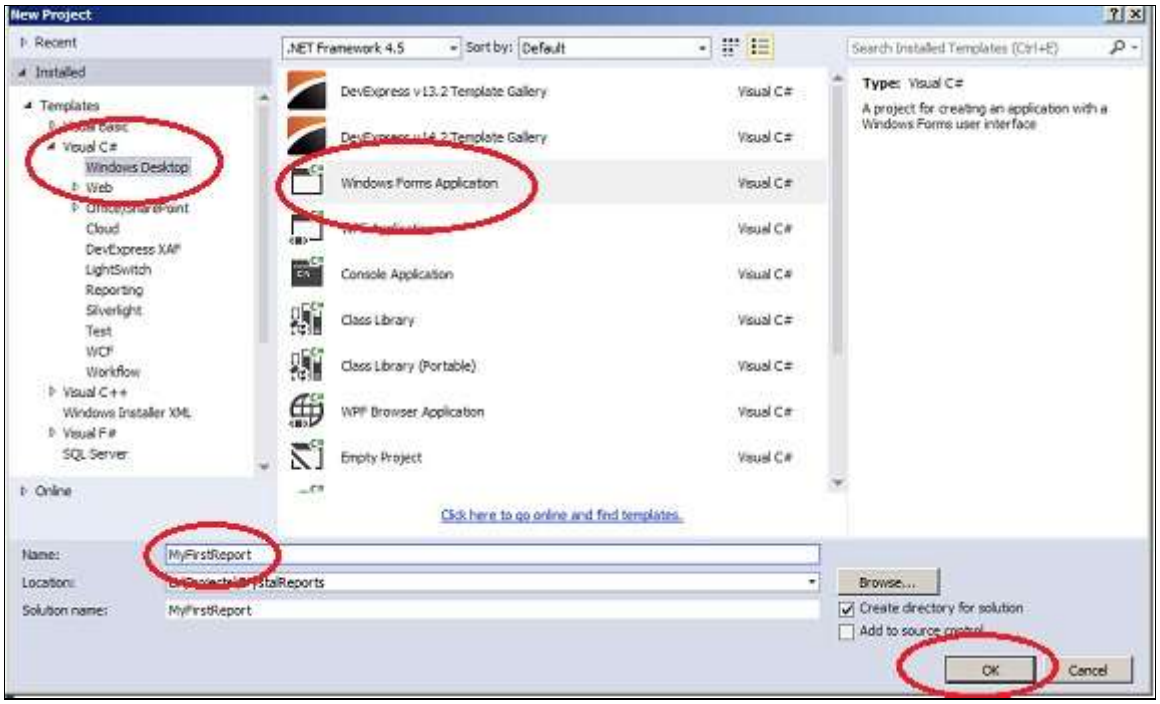
1. Download link – Service Pack 18
2. Download link – Service Pack 17
3. Download link – Service Pack 16

Crystal Reports Runtime

- 32 Bit Version of the Runtime Service Pack 18
- 64 Bit Version of the Runtime Service Pack 18

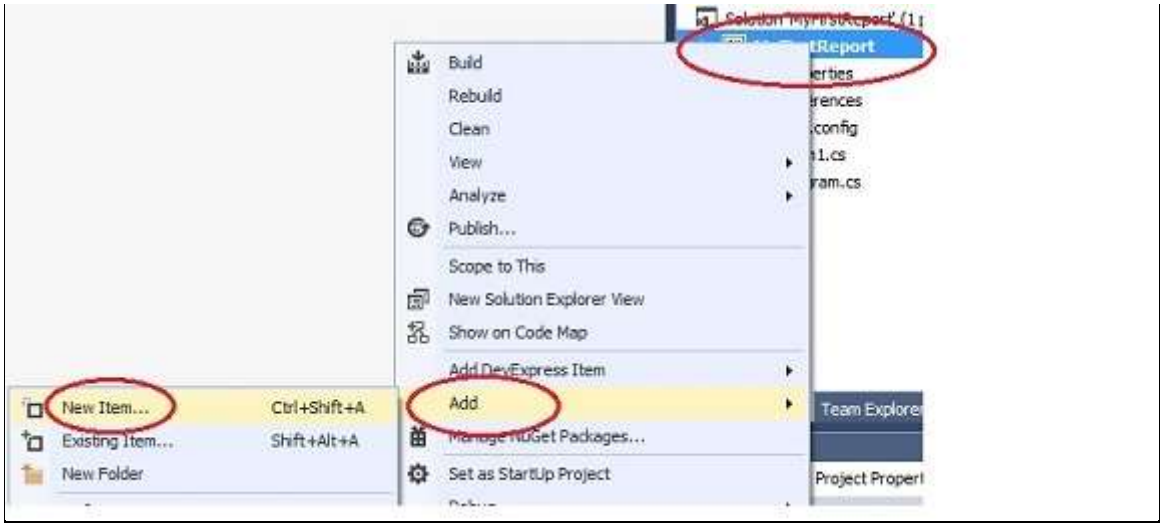
Phiên bản Crystal Report Service Pack 18 hỗ trợ các phiên bản Visual Studio 2010 đến 2015.

2.3.2.2. Tạo ứng dụng Visual Studio



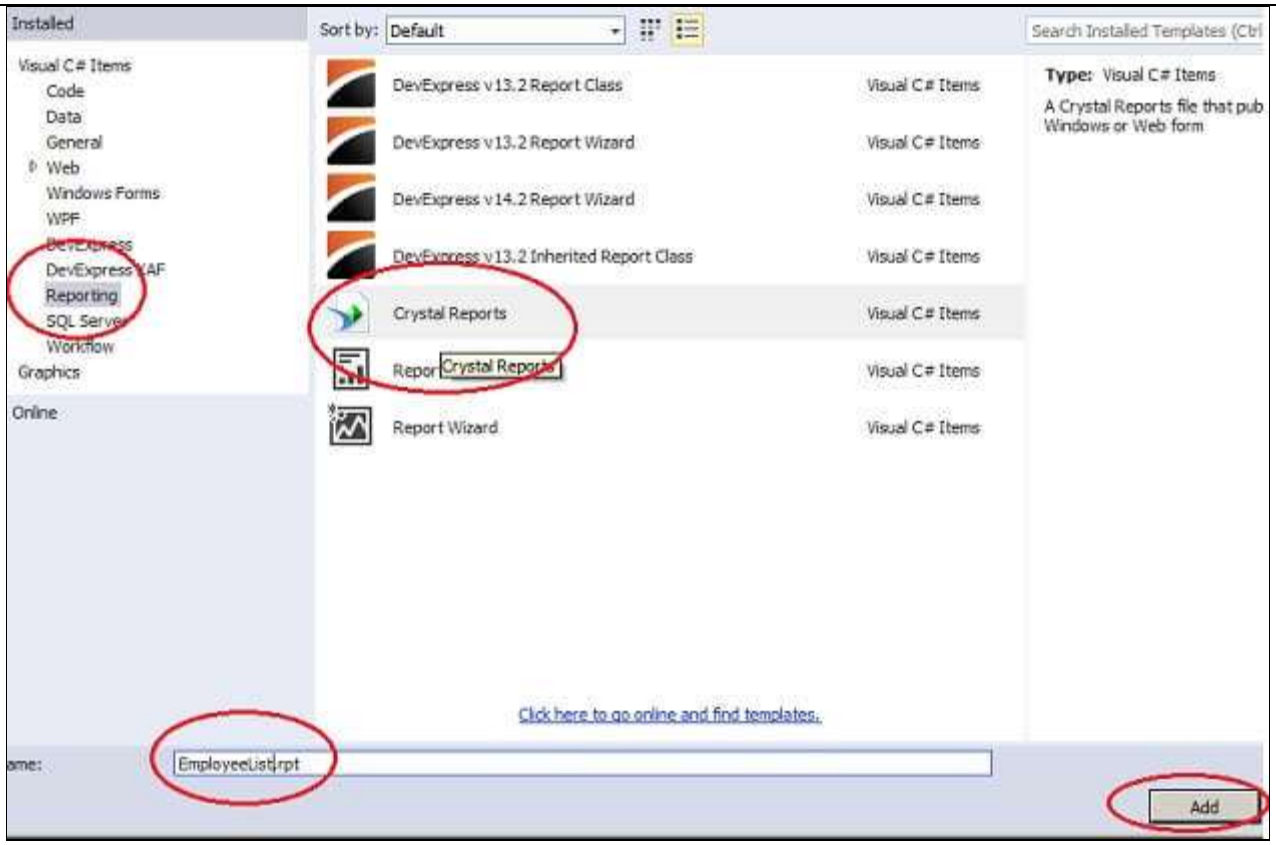
1. Mở Visual Studio và tạo 1 project mới.
2. Chọn C#.
3. Chọn Windows Desktop Application.
4. Chọn Windows Form Application.
5. Đặt tên cho project là *MyFirstReport*. Chọn *OK*. Visual C# Project sẽ hiển thị màn hình làm việc ra cho bạn.
6. Đổi tên form từ Form1 thành *frmCrystalReport*.

2.3.2.3. Tạo Crystal Report



Các bước để tạo mới một Crystal Report.

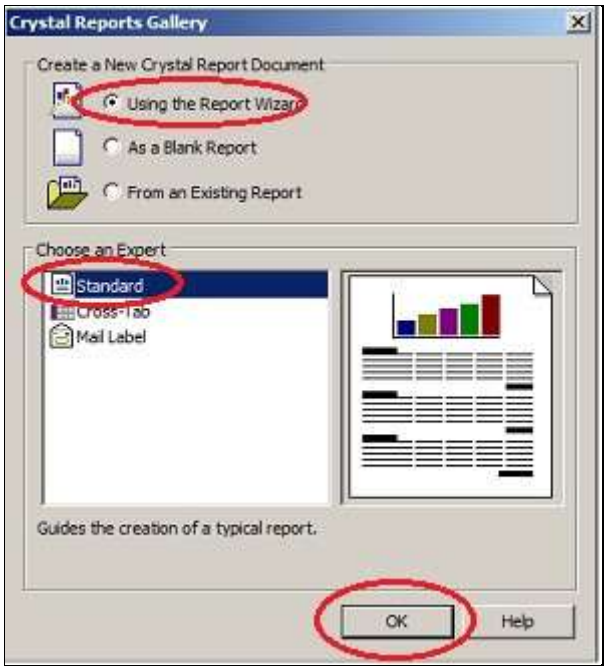
1. Trong *Solution Explorer*, chọn project
2. Nhấp chuột phải vào Project và chọn *Add New Item*. Chọn *CrystalReports* như hình dưới



3. Chọn Reporting Template Group.
4. Scroll và chọn Crystal Report Template.
5. Đặt tên là *EmployeeList*
6. Chọn *Add* để tạo Report. Sự lựa chọn này sẽ chuyển bạn sang màn hình *Crystal Report Gallery*.

2.3.2.3.1. Crystal Report Gallery

Có thể tạo Report bằng các lựa chọn như màn hình dưới:



Report Wizard

- **Using a Report Wizard**: Lựa chọn này sẽ đưa các bạn tới *Crystal Report Wizard*, hướng dẫn các bạn từng bước để tạo ra một report.
- **As a Blank Report**: Lựa chọn này sẽ tạo ra một report trống
- **From an Existing Report**: Bạn sẽ chọn một report có sẵn để sử dụng lại.

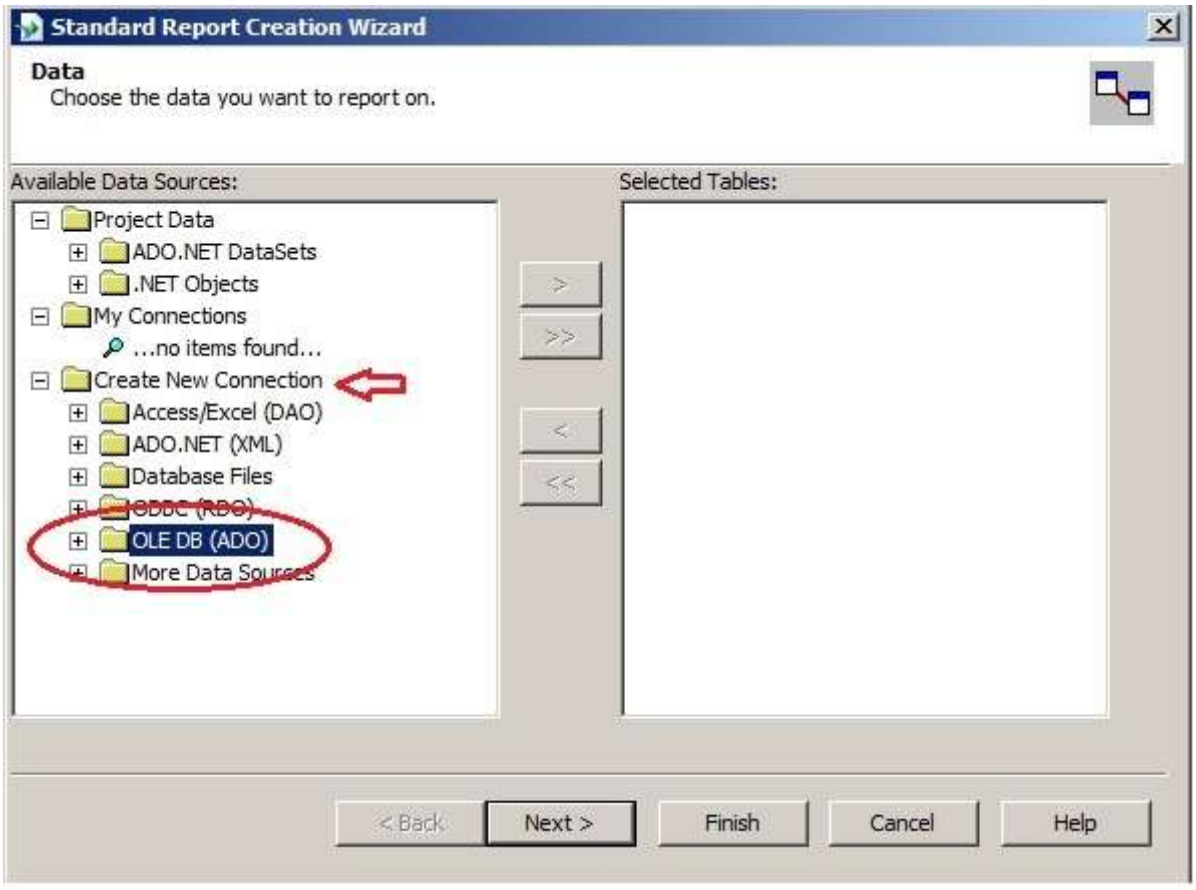
Report Expert

- **Standard Format**: Lựa chọn này cho phép tạo ra một report chuẩn.
- **Cross Tab**: Lựa chọn này cho phép tạo ra một report đặc biệt được biết là *cross tab report*.
- **Mail Label**: Lựa chọn này cho phép tạo ra report thực hiện công việc gần giống Mail Merge của Microsoft Office.

Trong ví dụ này, hãy chọn *Report Wizard* và *Standard Expert*, sau đó chọn *Next*. Chúng ta chuyển qua màn hình *Standard Report Creation Wizard*

2.3.2.3.2. Standard Report Creation wizard

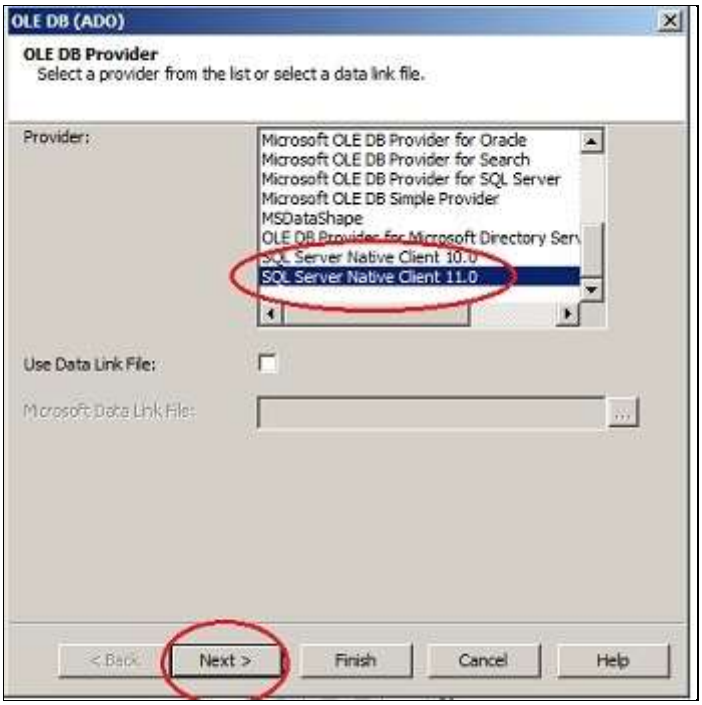
Hộp thoại dưới đây cho phép bạn chọn *data source* (nguồn dữ liệu) sẽ được dùng trong report. Bao gồm 3 lựa chọn:



- **Prject Data**: Lựa chọn này sẽ liệt kê các kết nối dữ liệu có sẵn trong project cho bạn chọn.
- **My Connections**: Lựa chọn này sẽ liệt kê các kết nối dữ liệu sử dụng gần đây nhất để bạn có thể chọn nhanh.
- **Create New Connection**: Lựa chọn này cho phép tạo kết nối dữ liệu mới.

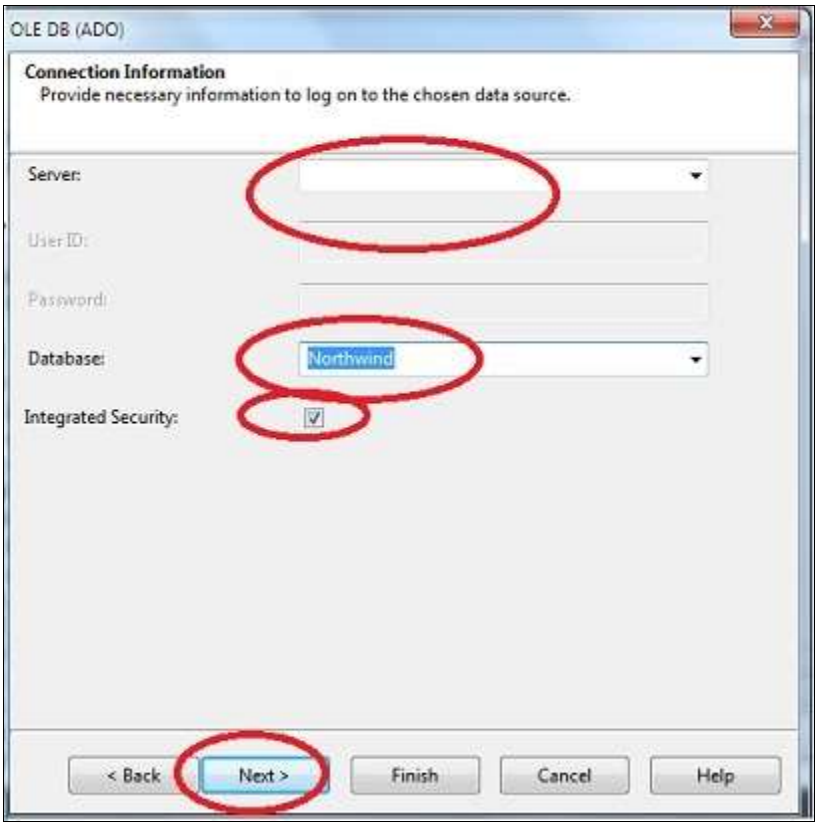
Chúng ta sẽ chọn *OLE DB (ADO)* để kết nối đến nguồn dữ liệu. Nhấp chuột vào dấu cộng (+) nằm bên phải của *OLE DB (ADO)*. Lựa chọn này sẽ mở ra hộp thoại *OLE DB (ADO) Dialog Box*

OLE DB (ADO) Dialog Box



Hộp thoại này sẽ liệt kê các OLE/DB Providers có sẵn: chọn OLE/DB Provider phù hợp để kết nối vào DB. Sau đó chọn *Next*.

OLE DB (ADO) Connection Info Dialog Box



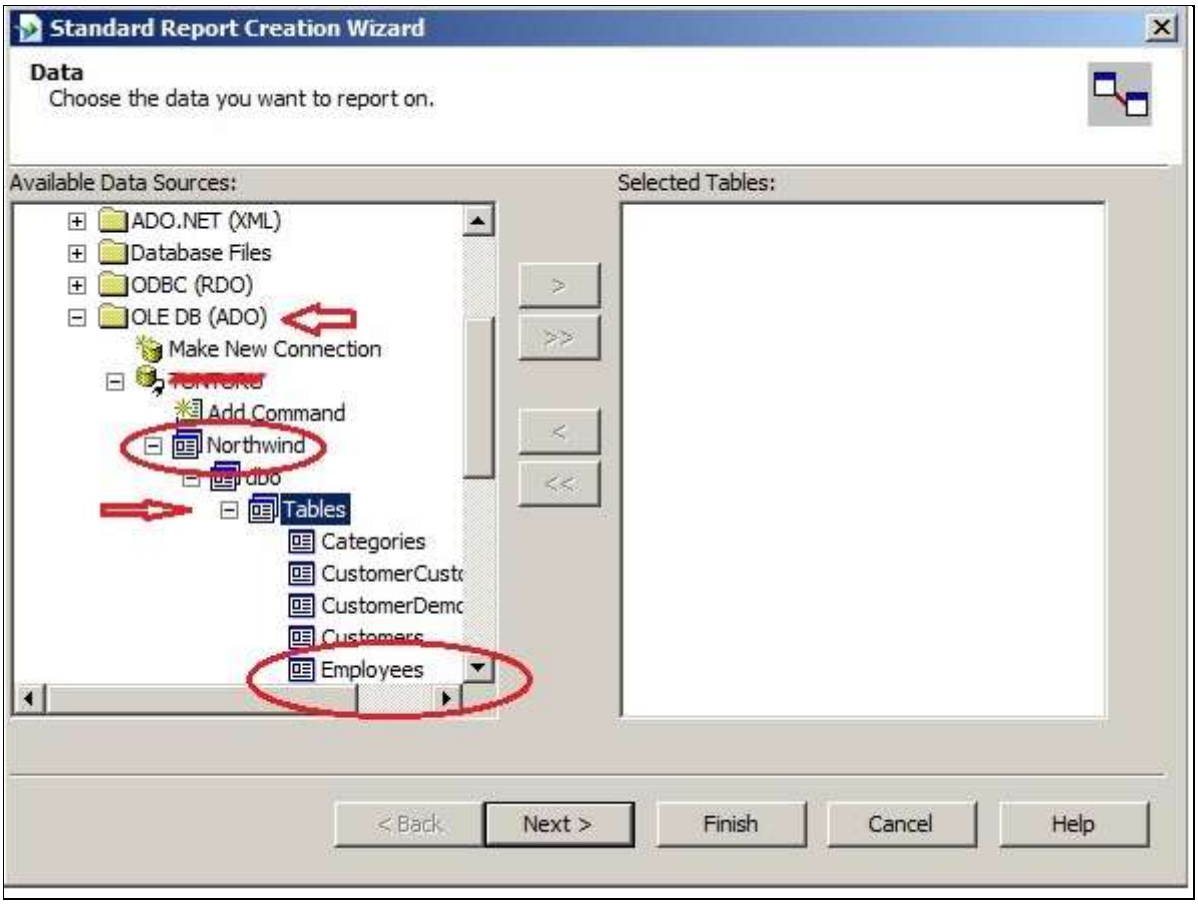
Trong màn hình này, bạn sẽ phải nhập các thông tin cần thiết để có thể kết nối vào DB.

Nhập thông tin của server, User ID, Password, (nếu bạn chọn Integrated security thì không cần nhập User ID và Password), chọn database **northwind** từ hộp thoại dropdown. Chọn *Next* khi hoàn thành.

OLE DB (ADO) Advance Information Dialog Box

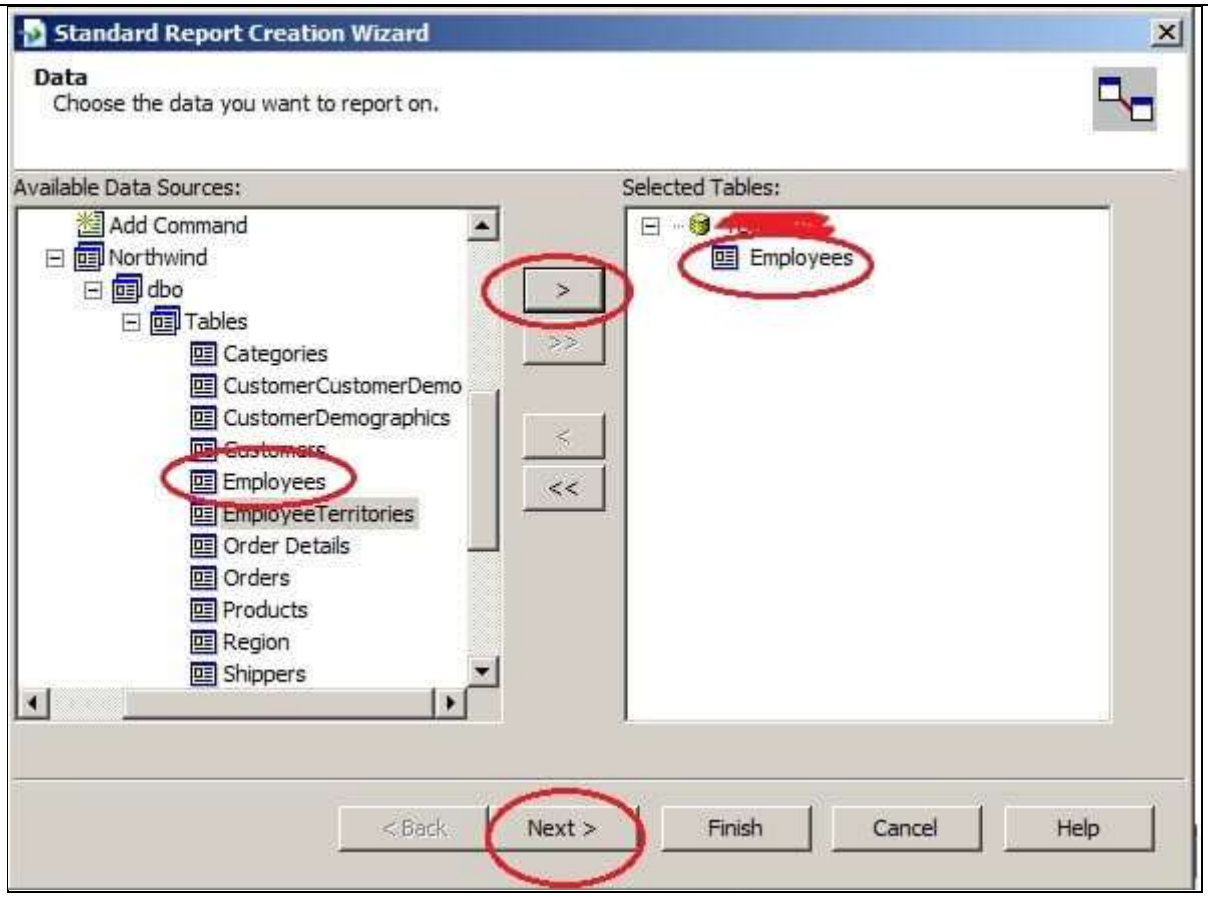
Kế tiếp bạn sẽ thấy *OLE DB (ADO) Advance Information Dialog Box*. Bạn có thể thay đổi các thông số cho phù hợp, rồi chọn *Finish* để quay trở lại *Standard Report Creation wizard*

2.3.2.3.3. Standard Report Creation wizard



Trong màn hình này, bạn phải thấy được DB *northwind*. Nếu không thấy, bạn phải kiểm tra lại các bước ở trên.

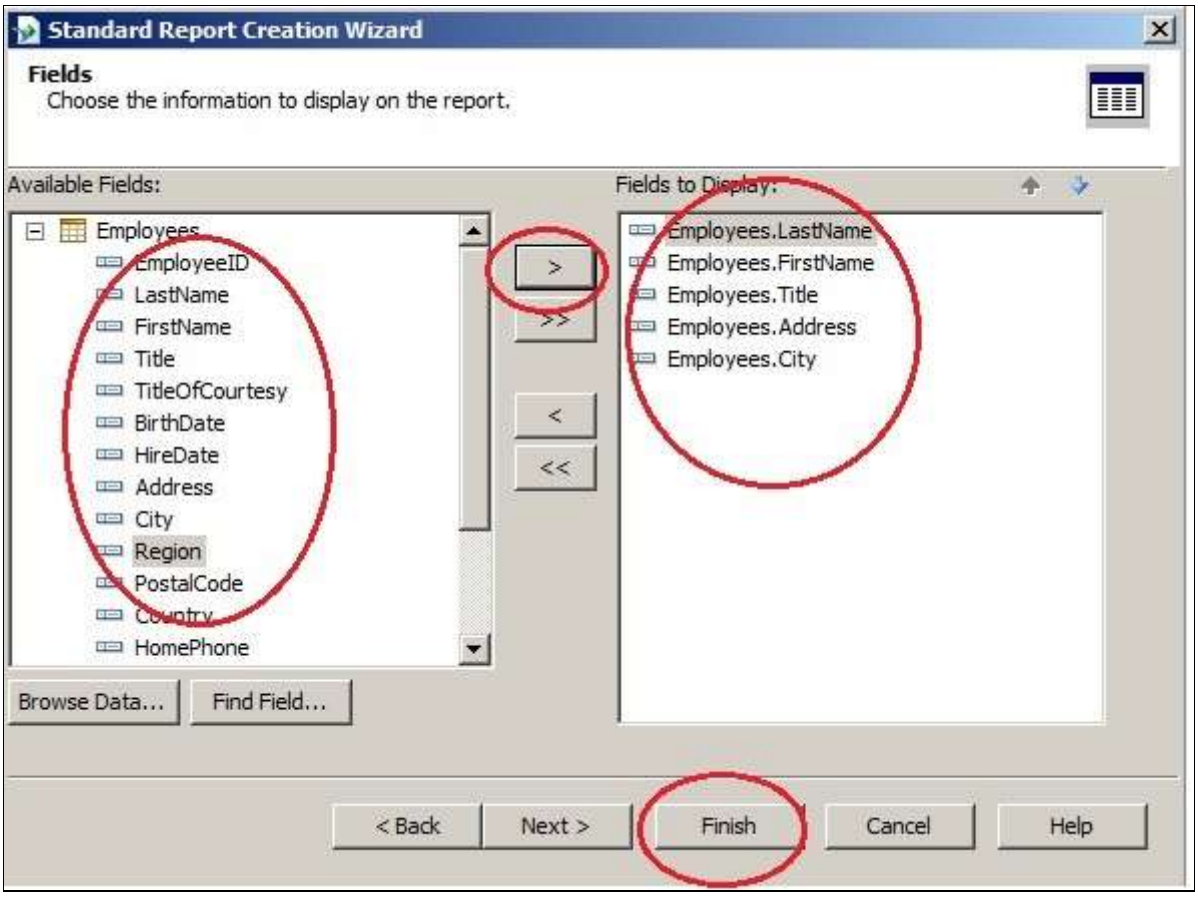
Chọn dấu cộng (+) để mở các nhánh của nút *Northwind* giống hình vẽ. Các Tables, Stored Procedures, Views của DB sẽ tự động hiển thị và sẵn sàng cho report sử dụng. Chọn table **Employees** giống hình minh họa.



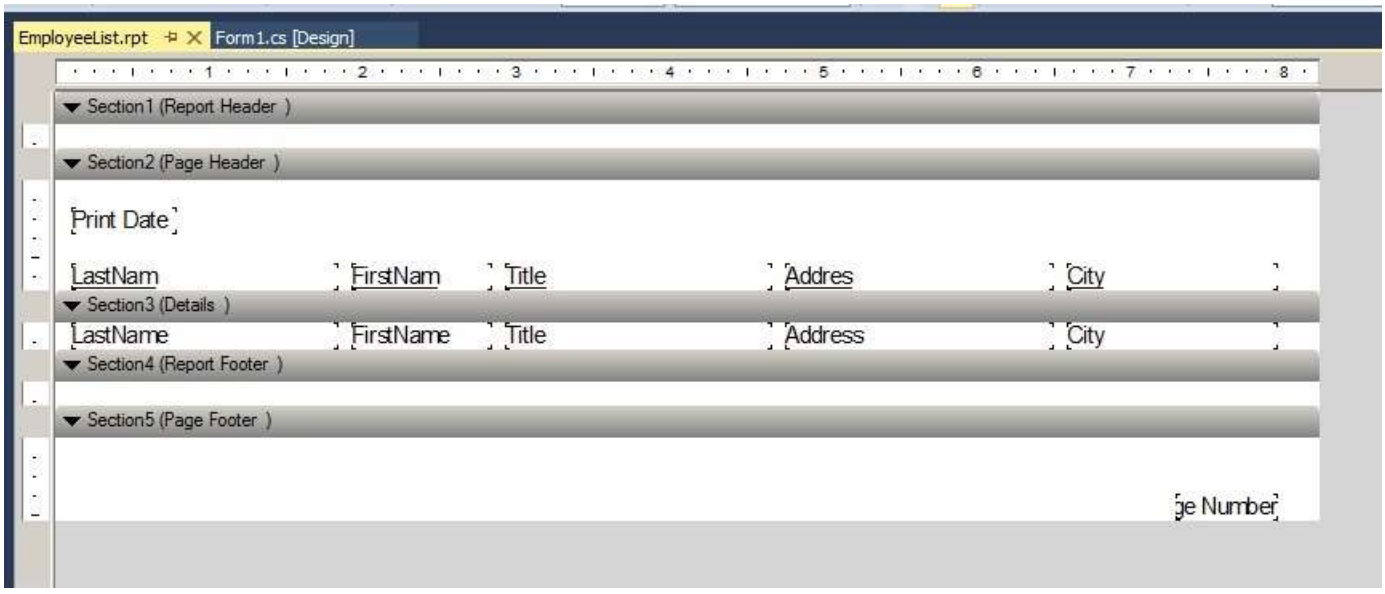
1. Chọn table Employees.
2. Chọn nút sang phải (>) để thêm bảng vào report.

Table Employees sẽ xuất hiện trên ô selected tables nằm ở phía bên phải. Chọn *Next*, bạn sẽ được chuyển sang màn hình Fields Dialog box.

2.3.2.3.4. Fields Dialog box

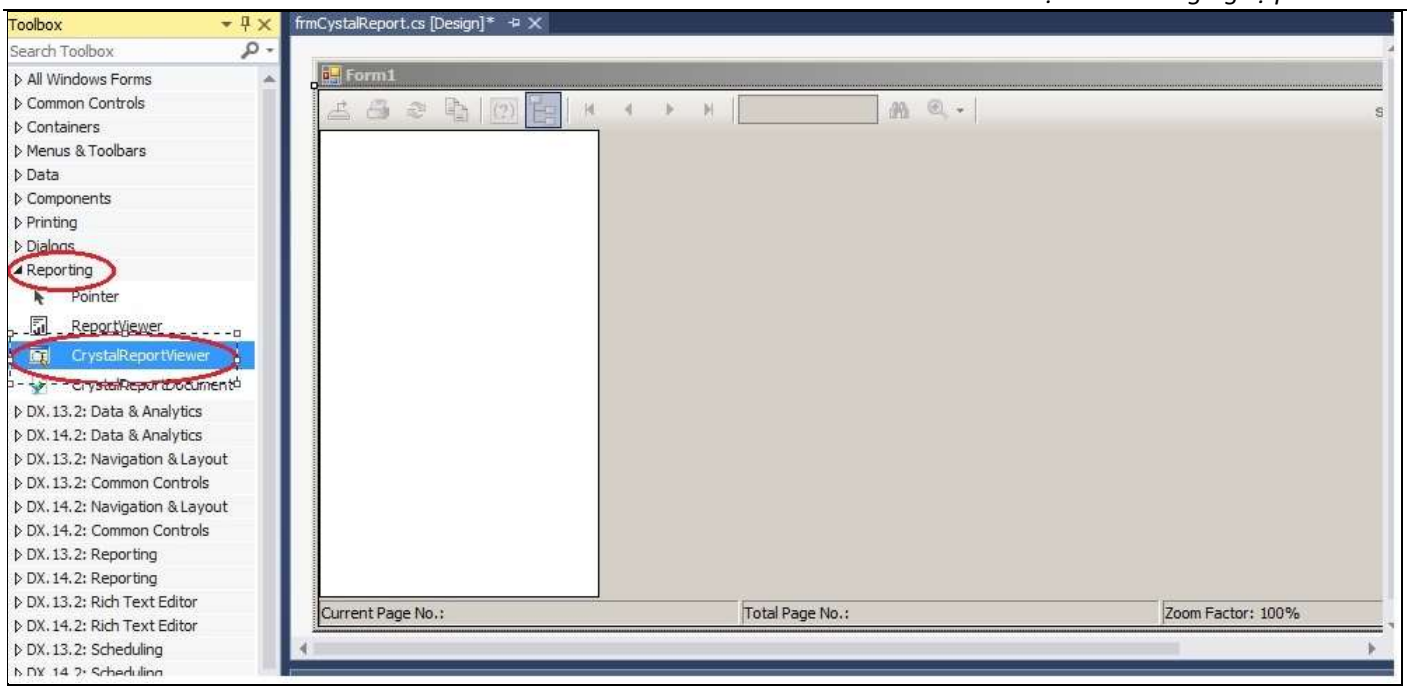


Bạn thực hiện lựa chọn giống hình. Xong chọn nút *Finish*. Kết quả giống hình dưới



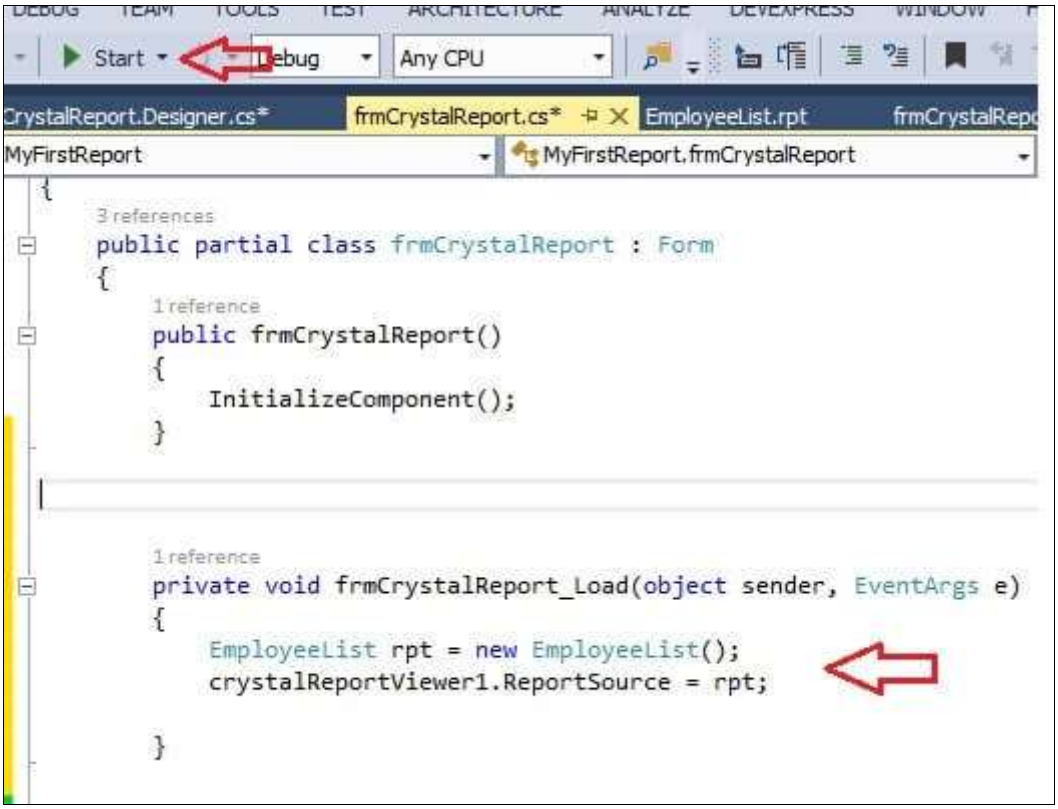
2.3.2.4. Crystal Report Viewer Control

Bước này sẽ hiển thị *Crystal Report* trong Windows Form của chúng ta: điều cần làm là thêm control *Crystal Report Viewer* vào form.



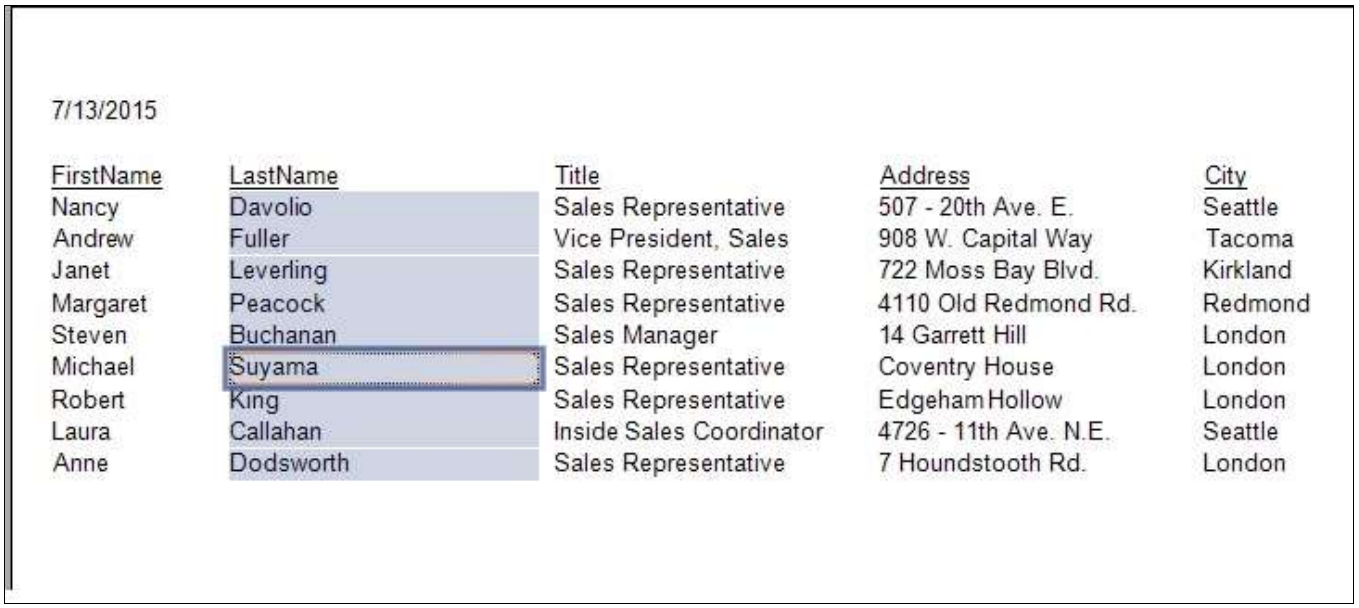
1. Mở form *frmCrystalReport* đã tạo ở bước trên.
2. Chọn tab Reporting từ thanh Toolbox
3. Chọn *CrystalReportView* và kéo thả vào form.

2.3.2.5. Bind Report to Crystal Report Viewer Control



1. Mở *frmCrystalReport*.
2. Viết code như hình minh họa.

2.3.2.6. Run Your First Report



7/13/2015

FirstName	LastName	Title	Address	City
Nancy	Davolio	Sales Representative	507 - 20th Ave. E.	Seattle
Andrew	Fuller	Vice President, Sales	908 W. Capital Way	Tacoma
Janet	Leverling	Sales Representative	722 Moss Bay Blvd.	Kirkland
Margaret	Peacock	Sales Representative	4110 Old Redmond Rd.	Redmond
Steven	Buchanan	Sales Manager	14 Garrett Hill	London
Michael	Suyama	Sales Representative	Coventry House	London
Robert	King	Sales Representative	Edgeham Hollow	London
Laura	Callahan	Inside Sales Coordinator	4726 - 11th Ave. N.E.	Seattle
Anne	Dodsworth	Sales Representative	7 Houndstooth Rd.	London