BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC DÂN LẬP HẢI PHÒNG
-----o0o-----

ĐỒ ÁN TỐT NGHIỆP

NGÀNH CÔNG NGHỆ THÔNG TIN

HẢI PHÒNG 2017

BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC DÂN LẬP HẢI PHÒNG

-----o0o-----

TÌM HIỂU VỀ LINQ TO SQL VÀ ỨNG DỤNG

ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC HỆ CHÍNH QUY
Ngành: Công nghệ Thông tin

Sinh viên thực hiện: Nguyễn Quốc Trung Anh
Mã số sinh viên: 1312101025
Cán bộ hướng dẫn: Th.S. Vũ Anh Hùng

HẢI PHÒNG - 2017

BỘ GIÁO DỤC VÀ ĐÀO TẠO **CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM**
TRƯỜNG ĐẠI HỌC DÂN LẬP HẢI PHÒNG ***Độc lập - Tự do - Hạnh phúc***
-----o0o-----

NHIỆM VỤ THIẾT KẾ TỐT NGHIỆP

Sinh viên: Nguyễn QUốc Trung Anh Mã sinh viên: 1312101025

Lớp: CT1701 Ngành: Công nghệ Thông tin

Tên đề tài: Tìm hiểu về LINQ TO SQL và ứng dụng

NHIỆM VỤ ĐỀ TÀI

1. Nội dung và các yêu cầu cần giải quyết trong nhiệm vụ đề tài tốt nghiệp

a. Nội dung

b. Các yêu cầu cần giải quyết

2. Các số liệu cần thiết để thiết kế, tính toán

3. Địa điểm thực tập

CÁN BỘ HƯỚNG DẪN ĐỀ TÀI TỐT NGHIỆP

Người hướng dẫn thứ nhất:

Họ và tên: Vũ Anh Hùng

Học hàm, học vị: Thạc Sĩ

Cơ quan công tác: Trường Đại Học Dân Lập Hải Phòng

Nội dung hướng dẫn:

.....

.....

.....

.....

Người hướng dẫn thứ hai:

Họ và tên:

Học hàm, học vị:

Cơ quan công tác:

Nội dung hướng dẫn:

.....

.....

.....

.....

Đề tài tốt nghiệp được giao ngày tháng năm 2017

Yêu cầu phải hoàn thành trước ngày tháng năm 2017

Đã nhận nhiệm vụ: Đ.T.T.N

Sinh viên

Đã nhận nhiệm vụ: Đ.T.T.N

Cán bộ hướng dẫn Đ.T.T.N

Th.S Vũ Anh Hùng

Hải Phòng, ngày ..... tháng ..... năm 2017

HIỆU TRƯỞNG

GS.TS.NGƯT Trần Hữu Nghị

PHẦN NHẬN XÉT TÓM TẮT CỦA CÁN BỘ HƯỚNG DẪN

1. Tinh thần thái độ của sinh viên trong quá trình làm đề tài tốt nghiệp:

.....

.....

.....

.....

.....

.....

2. Đánh giá chất lượng của đề tài tốt nghiệp (so với nội dung yêu cầu đã đề ra trong nhiệm vụ đề tài tốt nghiệp)

.....

.....

.....

.....

.....

.....

.....

3. Cho điểm của cán bộ hướng dẫn:

(Điểm ghi bằng số và chữ)

.....

.....

Ngày ..... tháng ..... năm 2017

Cán bộ hướng dẫn chính

(Ký, ghi rõ họ tên)

PHẦN NHẬN XÉT ĐÁNH GIÁ CỦA CÁN BỘ CHẤM PHẢN BIỆN ĐỀ TÀI TỐT NGHIỆP

1. Đánh giá chất lượng đề tài tốt nghiệp (về các mặt như cơ sở lý luận, thuyết minh chương trình, giá trị thực tế, ...)

2. Cho điểm của cán bộ phản biện

(Điểm ghi bằng số và chữ)

.....

.....

Ngày ..... tháng ..... năm 2017

Cán bộ chấm phản biện

(Ký, ghi rõ họ tên)

LỜI CẢM ƠN

Để hoàn thành đồ án tốt nghiệp này, em xin tỏ lòng biết ơn sâu sắc đến Thầy ThS. Vũ Anh Hùng – khoa Công nghệ thông tin, trường Đại học Dân Lập Hải Phòng, sự chỉ bảo dẫn dắt của thầy đã giúp em có những định hướng tốt việc triển khai và thực hiện các yêu cầu trong quá trình làm đồ án tốt nghiệp.

Em chân thành cảm ơn quý Thầy, Cô trong khoa Công nghệ thông tin, Trường Đại học Dân Lập Hải Phòng đã tận tình truyền đạt kiến thức trong những năm em học tập. Với vốn kiến thức được tiếp thu trong quá trình học không chỉ là nền tảng cho quá trình nghiên cứu đồ án mà còn là hành trang quí báu để em bước vào đời một cách vững chắc và tự tin.

Cuối cùng em kính chúc quý Thầy, Cô dồi dào sức khỏe và thành công trong sự nghiệp cao quý, đạt được nhiều thành công tốt đẹp trong công cuộc trồng người của mình.

Em xin chân thành cảm ơn!

Hải Phòng, ngày 31 tháng 08 năm 2017

Sinh viên

Nguyễn Quốc Trung Anh

MỤC LỤC

DANH MỤC HÌNH

LỜI MỞ ĐẦU

Công tác quản lý là một công tác không thể thiếu của tất cả các tổ chức về kinh tế. Với các lý do ấy thì phát triển Công nghệ thông tin đã trở thành một ngành kinh tế quan trọng , đặc biệt là Công nghệ phần mềm. Sự ra đời của các sản phẩm phần mềm đặc biệt là các phần mềm ứng dụng như quản lý trong vài năm gần đây mang lại nhiều thuận lợi trong công tác quản lý hàng hóa, quản lý nhận sự tránh sự nhầm lẫn, thất thu, mất mát. Tuy nhiên bên cạnh những tiện lợi mà các chương trình này mang lại, vẫn còn nhiều khó khăn, nhược điểm cần được khắc phục. Nhược điểm của các chương trình còn nhiều lý do như: Bản thân các nhà lập trình còn hạn chế về trình độ cũng như kinh nghiệm làm phần mềm.

Trong phát triển phần mềm, nhu cầu truy xuất và thao tác dữ liệu là vô cùng cần thiết. Đặc biệt, với SQLExpress để truy vấn dữ liệu ta phải dùng đến các câu lệnh – Query rất phức tạp, hơn nữa để dùng trong C# lại thêm một tầng phức tạp nữa với các câu lệnh : ConnectionString khởi tạo kết nối tới Database, tự khai báo các biến để chạy một lệnh - command, rồi còn phải tính toán đầu ra của câu lệnh,…. Thật là quá phức tạp nếu như ta có một chương trình khủng.

Vậy phải làm sao để giải quyết vấn đê này?

Một đề xuất là sử dụng LinQ to SQL.

Sự ra đời của LINQ to SQL giúp các lập trình viên .NET bớt đi gánh nặng phụ thuộc bên thứ 3 (dùng SQL để truy vấn). LINQ to SQL đồng thời đồng bộ hóa dữ liệu lấy ra và trả về khi truy xuất dữ liệu bằng việc các Data Model hứng dữ liệu trả về được tạo tự động sao cho tương thích với kiểu dữ liệu tương ứng của chúng khi ánh xạ vào cơ sở dữ liệu. Điều này làm tránh tình trạng mất hoặc sai lệch dữ liệu khi truy xuất và thao tác với Database. Trên hết, một công cụ được phát hành cho nền tảng duy nhất là .NET với những anh em trong gia đình .NET sử dụng thì hiệu suất LINQ to SQL hẳn là phải tốt hơn so với những công cụ bên thứ 3.

Đề tài: " Tìm hiểu về LinQ to SQL và ứng dụng", với nội dung gồm 3 chương:

Chương 1: TÌM HIỂU VỀ LINQ TO SQL

Chương 2: XÂY DỰNG CHƯƠNG TRÌNH ỨNG DỤNG

Chương 3: KẾT QUẢ CHƯƠNG TRÌNH THỰC NGHIỆM

Đề tài đã giúp em vận dụng được những hiểu biết của mình về LinQ to SQL để xây dựng được chương trình ứng dụng thực tế cho bài toán thu học phí của sinh viên trường Đại học Dân lập Hải Phòng, đáp ứng được một số yêu cầu công việc đặt ra hàng ngày.

Chương 1

TÌM HIỂU VỀ LINQ TO SQL

1.1. LinQ là gì?

Dữ liệu, linh hồn của một phần mềm, và việc xử lý dữ liệu luôn là một trong những vấn đề quan trọng nhất của mỗi phần mềm. Với những dữ liệu có cấu trúc, ta thường cài đặt các câu lệnh xử lý dữ liệu sao cho phù hợp với cấu trúc nhất.

Tuy nhiên mọi chuyện trở nên phức tạp hơn khi cấu trúc dữ liệu bị thay đổi, có nhiều mảng muốn xử lý cùng lúc, đặc biệt là XML.

Rất may, Microsoft đã hiểu được tầm quan trọng của việc xử lý dữ liệu, và đã tung ra LinQ ở phiên bản .NET Framework 3.5 vào tháng 11/2007.

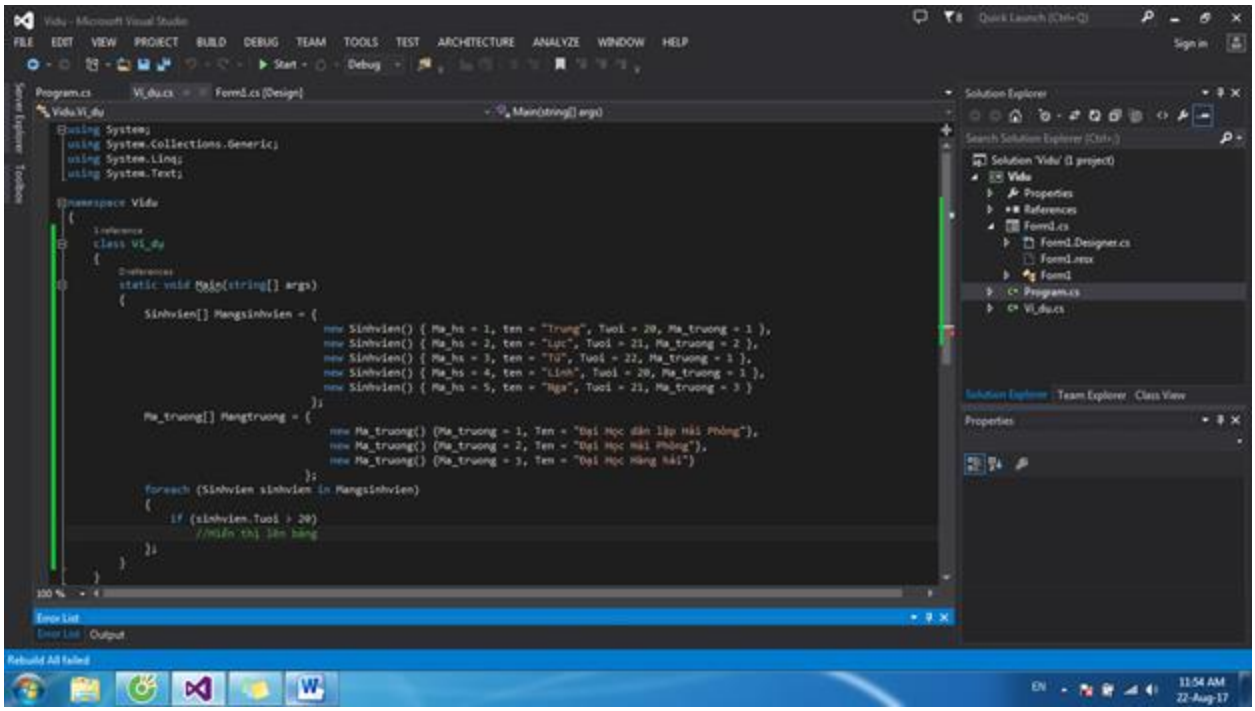
LINQ (**Language Integrated Query**, tạm dịch là ngôn ngữ truy vấn tích hợp) đưa ra 1 mô hình bền vững để hoạt động với các dạng nguồn dữ liệu và định dạng dữ liệu khác nhau.

Trong LINQ, bạn phải làm quen với chuyện làm việc với các đối tượng (objects). LINQ cho phép dùng các đoạn code đơn giản để truy vấn và chuyển đổi dữ liệu trong các tài liệu XML, cơ sở dữ liệu SQL, tập dữ liệu ADO.NET, các tập hợp .NET, và bất kỳ định dạng nào mà LINQ provider hỗ trợ.

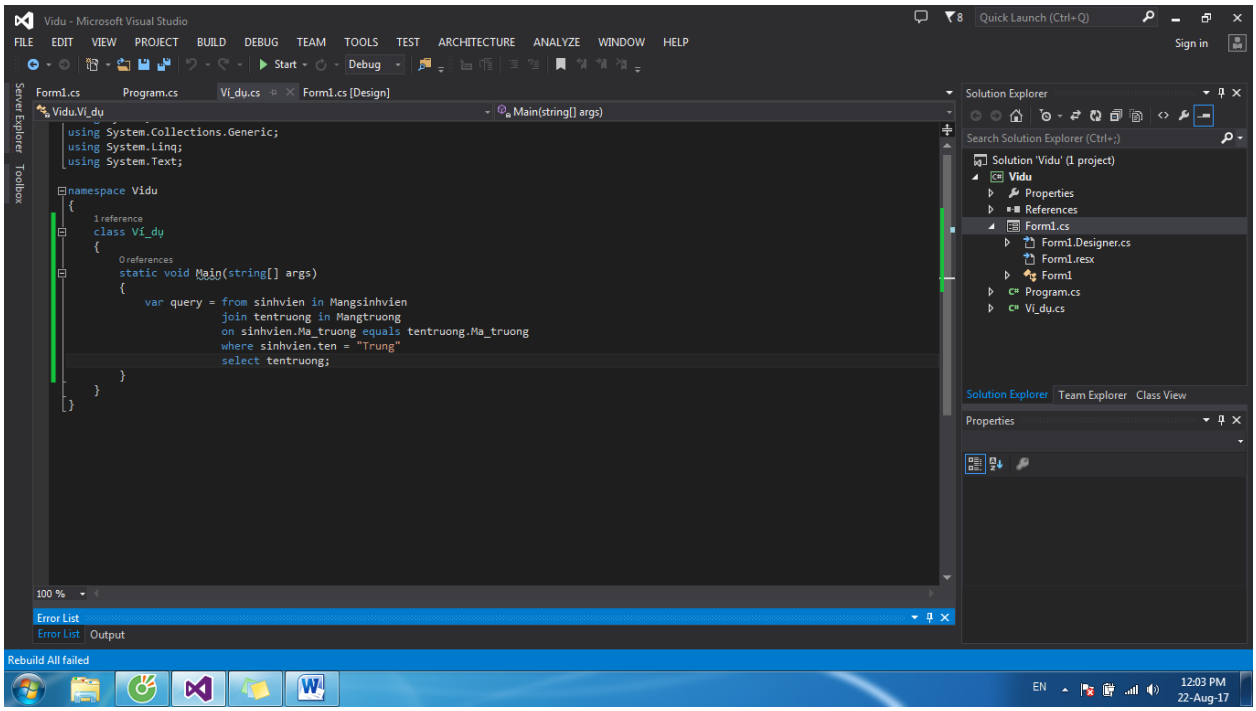
1.1.1. Tại sao phải dùng tới LinQ?

Để hiểu được tầm quan trọng của LinQ, hãy cùng xét 1 ví dụ nho nhỏ sau: Có 1 mảng Student chứa thông tin tất cả các học sinh của 1 trường, và 1 mảng Classroom chứa thông tin phòng học.

Sẽ thật dễ dàng nếu yêu cầu "tìm tất cả sinh viên có độ tuổi lớn hơn 20".

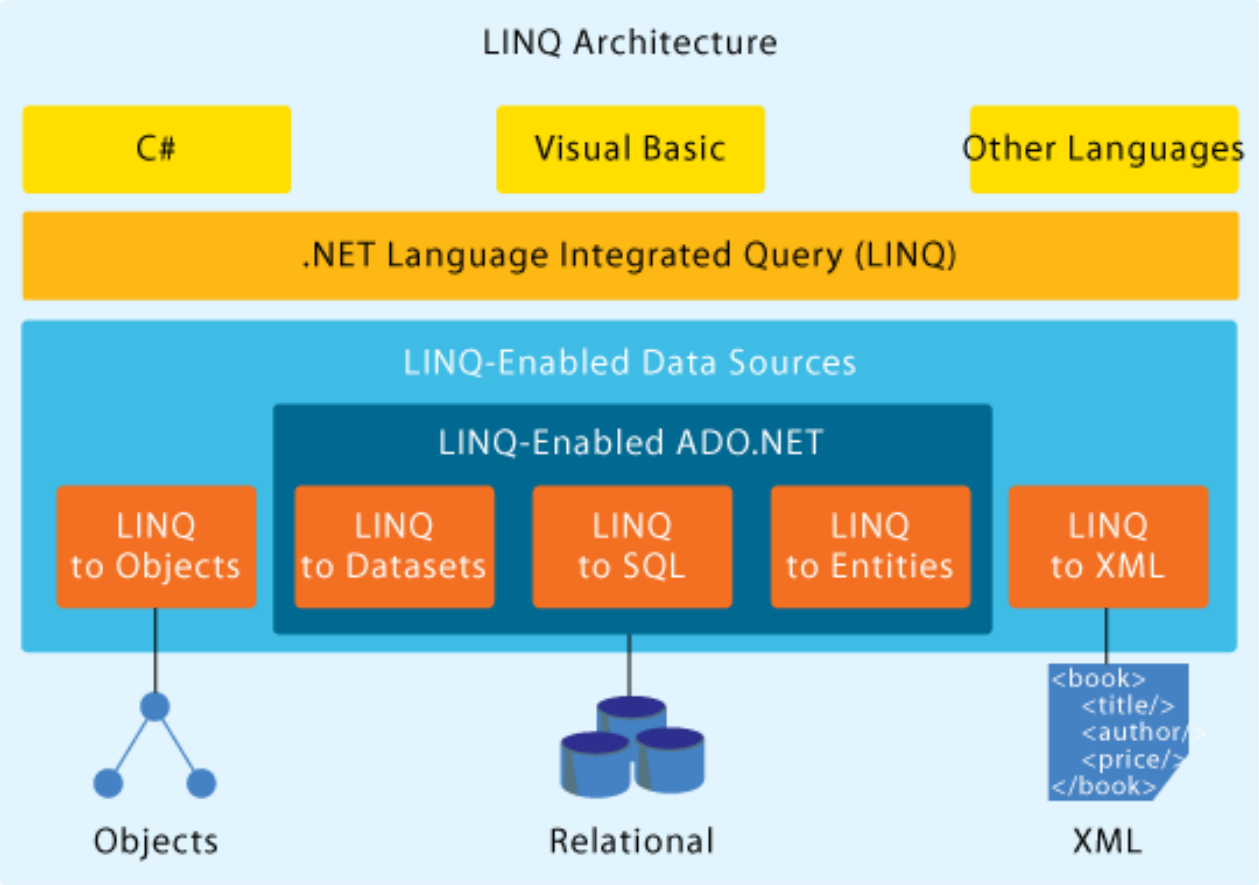


Nhưng nếu vấn đề đặt ra là "tìm sinh viên tên Trung học ở trường nào" thì sao? Phải tạo 1 mảng mới, trộn các dữ liệu từ 2 **Mangsinhvien** và **Mangtruong** và chạy vòng foreach? Tuy nhiên với LinQ, mọi thứ dường như đơn giản hơn rất nhiều:



Chỉ cần 1 câu lệnh " JOIN " của LinQ là xong. Rất đơn giản!!!

1.1.2. Kiến trúc và thành phần của LinQ



Hình 1.1 Kiến trúc và thành phần của LINQ

❖ **SQL Sever Databases:** LinQ to Sql

+ Cho phép truy vấn các cơ sở dữ liệu quan hệ

❖ **XML documents:** LinQ to XML

+ Sử dụng LinQ to XML với mục đích truy vấn file XML Documents

❖ **ADO.NET datasets :** LINQ to Dataset

+ Cho phép truy vấn các Dataset hoặc DataTable

❖ **.NET collections, strings, files, …:** LinQ to Objects

+ Cho phép truy vấn các đối tượng trong một tập các phần tử nguồn, các kiểu đối tượng này phải thực thi giao diện Ienumrable hoặc Ienumrable<T>.

❖ **Entyti Framework :** LinQ to Entities

+ Cho phép truy vấn các thực thể bên trong Entity Framework.

1.2. LinQ to SQL

LINQ to SQL là một phiên bản hiện thực hóa của O/RM (object relational mapping) có bên trong .NET Framework bản "Orcas" (nay là .NET 3.5), nó cho phép bạn mô hình hóa một cơ sở dữ liệu dùng các lớp .NET. Sau đó bạn có thể truy vấn cơ sở dữ liệu (CSDL) dùng LINQ, cũng như cập nhật/thêm/xóa dữ liệu từ đó.

LINQ to SQL hỗ trợ đầy đủ transaction, view và các stored procedure (SP). Nó cũng cung cấp một cách dễ dàng để thêm khả năng kiểm tra tính hợp lệ của dữ liệu và các quy tắc vào trong mô hình dữ liệu của bạn.

1.2.1.Vì sao LinQ to Sql ra đời?

Mục đích ra đời của LINQ hay LINQ to SQL là để phục vụ cho hệ quản trị cơ sở dữ liệu SQL Server và nền tảng .NET nói chung hay ngôn ngữ lập trình C#/VB.NET nói riêng chứ không phải mục đích đại trà cho các ngôn ngữ lập trình hay các hệ quản trị cơ sở dữ liệu khác. Ta biết rằng:

- C# là một ngôn ngữ lập trình hướng đối tượng hoàn toàn.
- SQL Server là một hệ quản trị cơ sở dữ liệu (DBMS) theo mô hình quan hệ, mô hình CSDL quan hệ ghi các dữ liệu theo dòng trong các bảng dữ liệu.

Muốn dùng C# viết mã để lấy dữ liệu từ SQL Server nhưng tổ chức mô hình dữ liệu của C# và SQL Server là khác nhau. Để giải quyết tình trạng trên, một kỹ thuật gọi là ORM (Object Relational Mapping) ra đời nhằm mục đích chuyển đổi dữ liệu giữa các hệ thống khác (không phải là mô hình hướng đối tượng) sang các đối tượng trong ngôn ngữ lập trình hướng đối tượng. Tiếp đó LINQ to SQL ra đời dựa trên kỹ thuật ORM xóa bỏ khoảng cách giữa mô hình lập trình hướng đối tượng C#/VB.NET với hệ quản trị cơ sở dữ liệu SQL Server

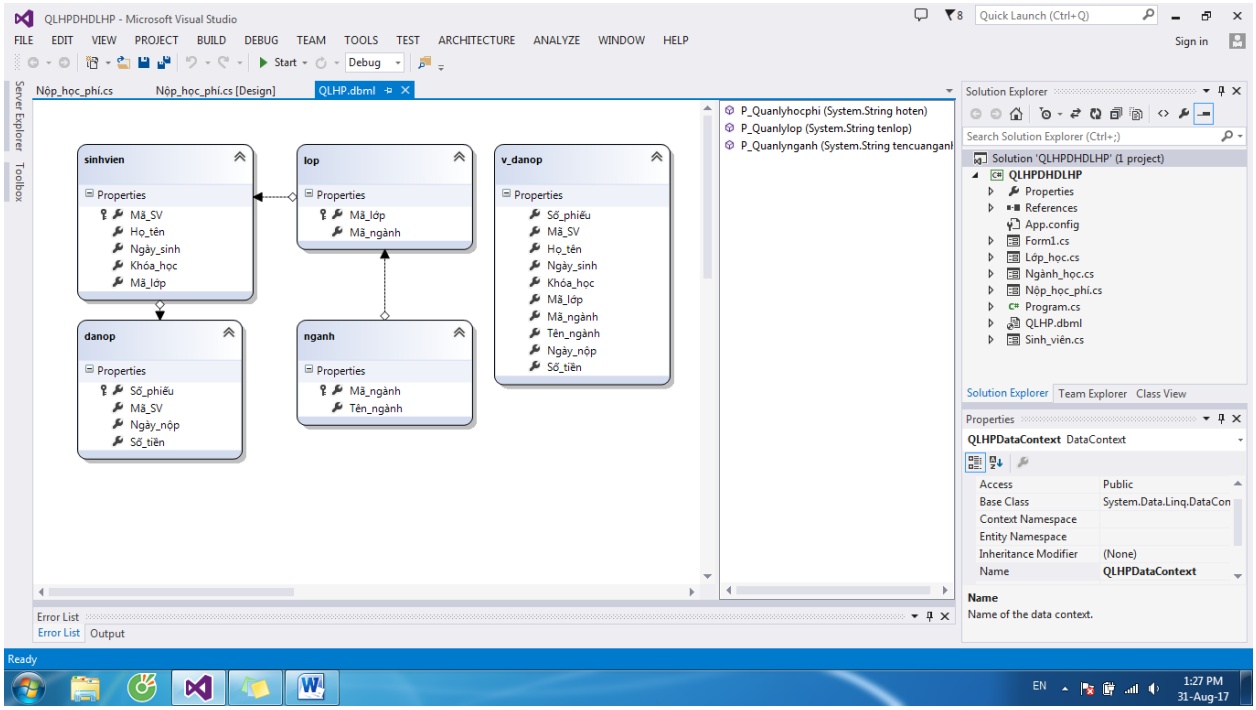
khi đã mô hình hóa theo hướng đối tượng các bảng trong Database thành các lớp tương ứng khi lập trình.

Sự ra đời của LINQ to SQL giúp các lập trình viên .NET bớt đi gánh nặng phụ thuộc bên thứ 3 (dùng SQL để truy vấn). LINQ to SQL đồng thời đồng bộ hóa dữ liệu lấy ra và trả về khi truy xuất dữ liệu bằng việc các Data Model hứng dữ liệu trả về được tạo tự động sao cho tương thích với kiểu dữ liệu tương ứng của chúng khi ánh xạ vào cơ sở dữ liệu. Điều này làm tránh tình trạng mất hoặc sai lệch dữ liệu khi truy xuất và thao tác với Database. Trên hết, một công cụ được phát hành cho nền tảng duy nhất là .NET với những thành viên trong gia đình .NET sử dụng thì hiệu suất LINQ to SQL hẳn là phải tốt hơn so với những công cụ bên thứ 3.

1.2.2.Vì sao nên sử dụng LinQ to SQL?

a) Một công cụ hỗ trợ đắc lực

Khi sử dụng LINQ to SQL với hệ quản trị cơ sở dữ liệu SQL Server, em không phải tạo các lớp Data Model để hứng dữ liệu trả về khi truy vấn dữ liệu vì LINQ to SQL đã tạo sẵn những lớp này với đầy đủ các thuộc tính và kiểu dữ liệu phù hợp với kiểu dữ liệu các cột bạn qui định trong Database (các thuộc tính của mỗi lớp ánh xạ vào các cột của bảng tương ứng trong CSDL).



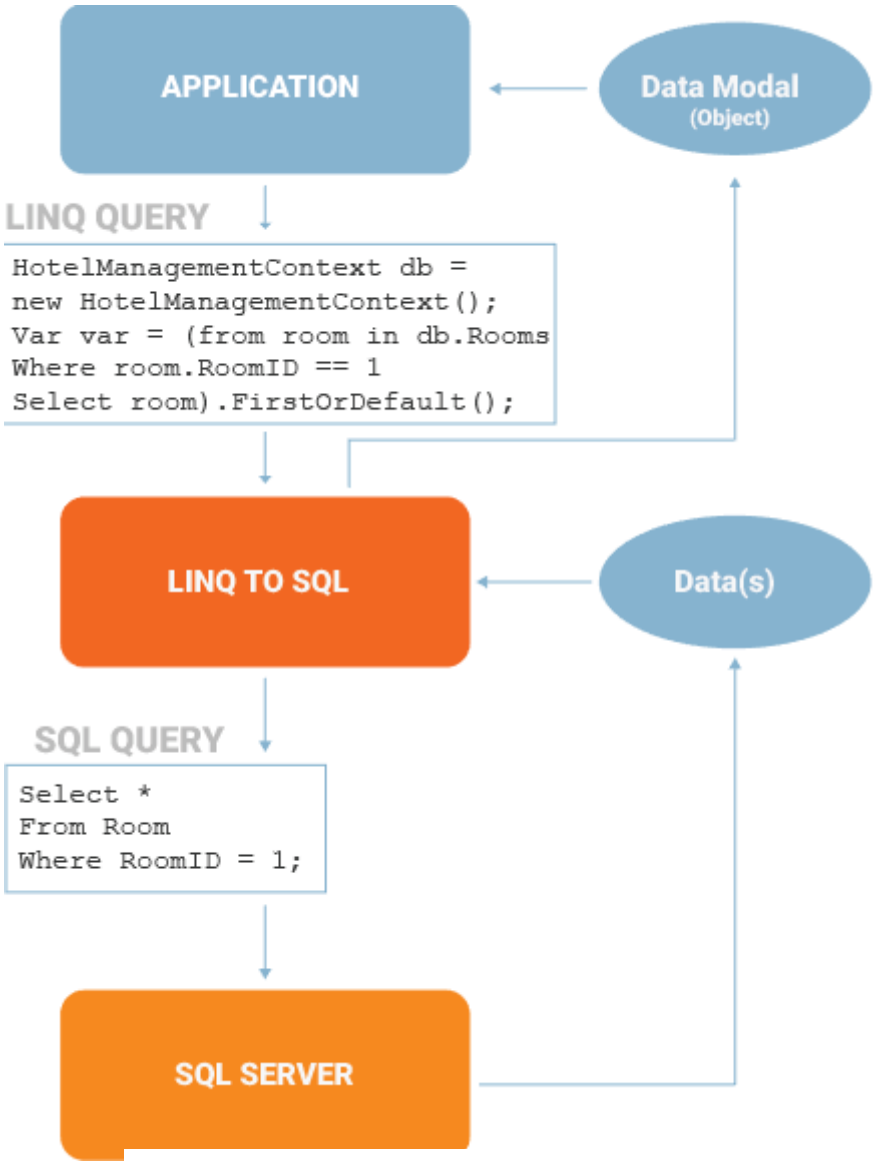
Hình 1.2 File lưu trữ của LinQ to SQL

Như với hình trên, sau khi lưu trữ file QLHP.dbml, hệ thống sẽ tạo ra các lớp sinhvien-lop-nganh-danop-v_danop, với đầy đủ các thuộc tính và kiểu dữ liệu tương thích với dữ liệu tương ứng của chúng trong Database, tránh làm mất hoặc hư dữ liệu sau khi truy xuất và thao tác với cơ sở dữ liệu.

b) Giao diện trực quan và tự động

LINQ to SQL cung cấp giao diện trực quan về mối quan hệ các bảng dữ liệu sau khi được mô hình hóa. Các lớp DataContext sẽ được tạo ra tự động khi bạn Import file LINQ to SQL vào Project. Các lớp DataContext nhận nhiệm vụ mở kết nối đến cơ sở dữ liệu, thực hiện truy vấn hay thay đổi dữ liệu. Các lớp thuộc tính được mô hình hóa từ các bảng dữ liệu trong hệ quản trị cơ sở dữ liệu được truy cập thông qua các lớp DataContext. Lớp DataContext này gần như là một lớp bao (Wrapper Class), những thay đổi từ các bảng dữ liệu trong cơ sở dữ liệu thì lớp này sẽ cập nhật và thay đổi tương ứng (chúng cũng sẽ cập nhật vào các lớp Data Modal được tạo tự động). Điều này khiến cho việc thay đổi thuộc

tính dữ liệu trong database diễn ra dễ dàng và người lập trình không mất quá nhiều công sức để chỉnh sửa lại code (những thay đổi được tự động cập nhật lại).



Hình 1.3 Giao diện trực quan và tự động của LinQ to SQL

c) LinQ nhưng bản chất vẫn là SQL

Đúng như cái tên LINQ to SQL, các câu truy vấn LINQ sẽ được chuyển sang câu truy vấn SQL trước khi đưa vào SQL Server để truy vấn dữ liệu (LINQ to SQL giống như việc mang SQL vào và viết bằng C#, tuy vậy bản chất vẫn là SQL).

LINQ to SQL là một công cụ tốt đối với những lập trình viên .NET. Cấu trúc,

câu lệnh của nó dễ học và quen thuộc (C#/VB.NET). LINQ to SQL rút ngắn thời gian phát triển phần mềm trên nền tảng .NET khi muốn thao tác và truy xuất dữ liệu từ cơ sở dữ liệu, hệ thống tạo vào hỗ trợ tất cả những thứ cơ bản như lớp DataContext, các lớp Data Model để hứng dữ liệu, lập trình viên chỉ cần truy xuất và thao tác với dữ liệu.

1.3. Những hạn chế của LinQ to SQL

- ❖ Chỉ thao tác duy nhất với hệ quản trị cơ sở dữ liệu SQL server.
- ❖ Chỉ có thể tự động tạo Data Model từ Database chứ không thể tạo Database từ Data Model.

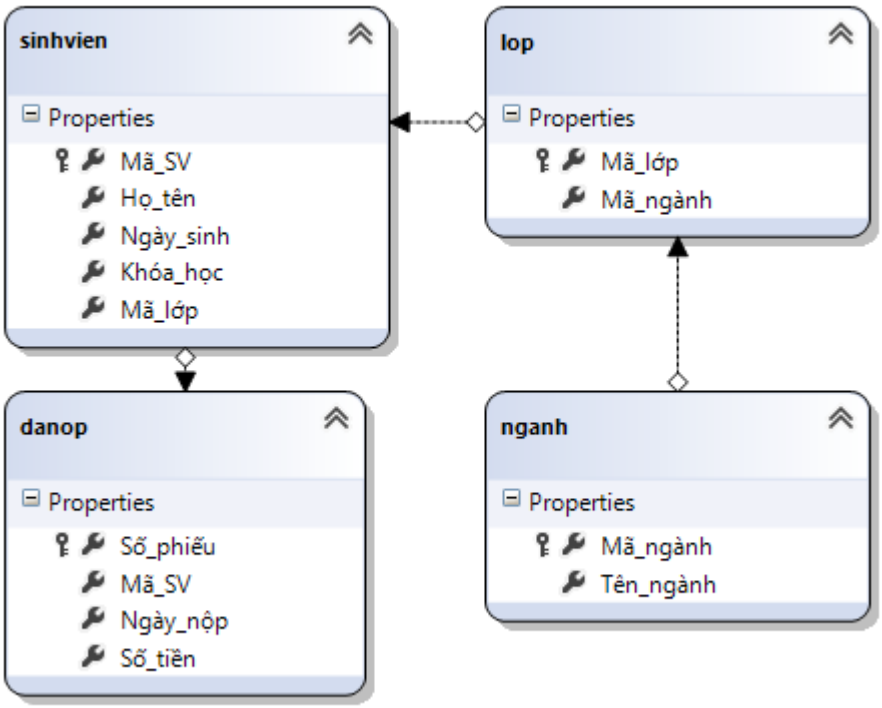
Chỉ cho phép ánh xạ 1:1 giữa các Table với các lớp Data Model (tức là không thể tạo 1 Data Model là kết quả kết hợp từ 2 bảng dữ liệu trở lên).

1.4. Những khái niệm cơ bản của LINQ to SQL

- Object Model
- ORM (Object Relations Mapping)
- O/R Designer (Object Relations Designer)
- Entity Class
- Association
- DataContext

1.4.1. Object Model

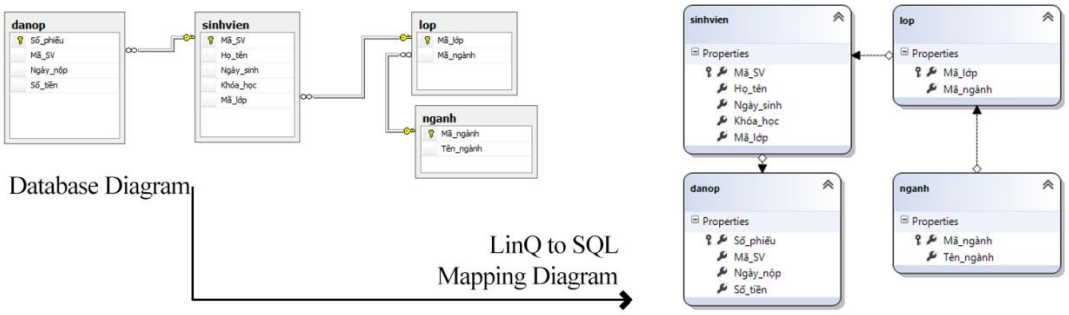
- Là mô hình được ánh xạ từ mô hình dữ liệu của cơ sở dữ liệu.
- Người lập trình sẽ làm việc với mô hình này như thể là đang lập trình hướng đối tượng (với các class được phát sinh).



Hình 1.4 Mô hình ánh xạ từ mô hình CSDL

1.4.2. ORM (Object Relations Mapping)

- Chính là phương pháp ánh xạ trực tiếp 1-1 giữa các đối tượng trong cơ sở dữ liệu quan hệ với các class của .NET.
- Từ mô hình quan hệ Database Diagram chuyển thành LINQ to SQL Mapping Diagram.
- Visual Studio IDE từ phiên bản 2008 có hỗ trợ 2 ORM là : LINQ to SQL và ADO.NET ENTITY FRAME WORK (có từ version 3.5 SP1).



Hình 1.5 Mô hình ánh xạ của ORM

1.4.3. O/R Designer (Object Relations Designer)

- Là một công cụ trong Visual Studio IDE có từ phiên bản 2008, dùng để hỗ trợ việc tạo ra Oblect Model.
- Phát sinh ra các ENTITY Class (Các Table, View từ cơ sở dữ liệu được ánh xạ thành các class) trong các Object đang làm việc. Khi sử dụng ENTITY Class sẽ ánh xạ đến một Table và một Property sẽ ánh xạ đến một Column của Table.

Database Object	LINQ Object
Database	DataContext
Table	Class and Collection
View	Class and Collection
Column	Property
Relationship	Nested Collection
Stored Proceduce	Method

Hình 1.6 Mô hình ánh xạ của O/R Designer

1.4.4. Entity Class

- Các Table, View từ CSDL được ánh xạ thành các Class được gọi là Entity Class.
- Khi sử dụng. Entity Class sẽ ánh xạ đến một Table và một Property sẽ ánh xạ đến một Column của một Table

1.4.5. Association

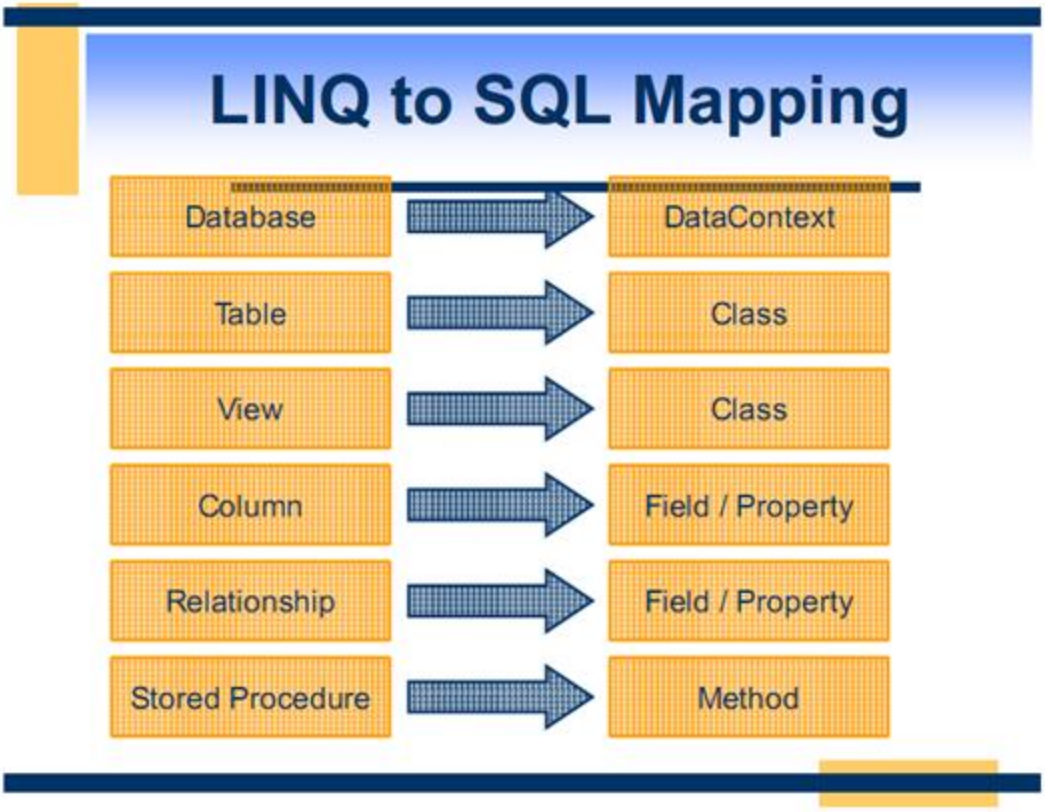
- Mối kết hợp, hay còn gọi là mối quan hệ giữa 2 ENTITY dựa trên primary key (khóa chính) và foreign key

1.4.6. DataContext

- Là một trong .NET, dùng để hỗ trợ việc kết nối CSDL. Ngoài ra nó còn quản lý các định danh (IDENTITY) của các đối tượng trong CSDL như Table, View,…
- Có thể tạo ra một Strong-Typed DATACONTEXT cho riêng ứng dụng -> đây là cách thông dụng nhất

1.5 Mô hình ánh xạ của LinQ to SQL

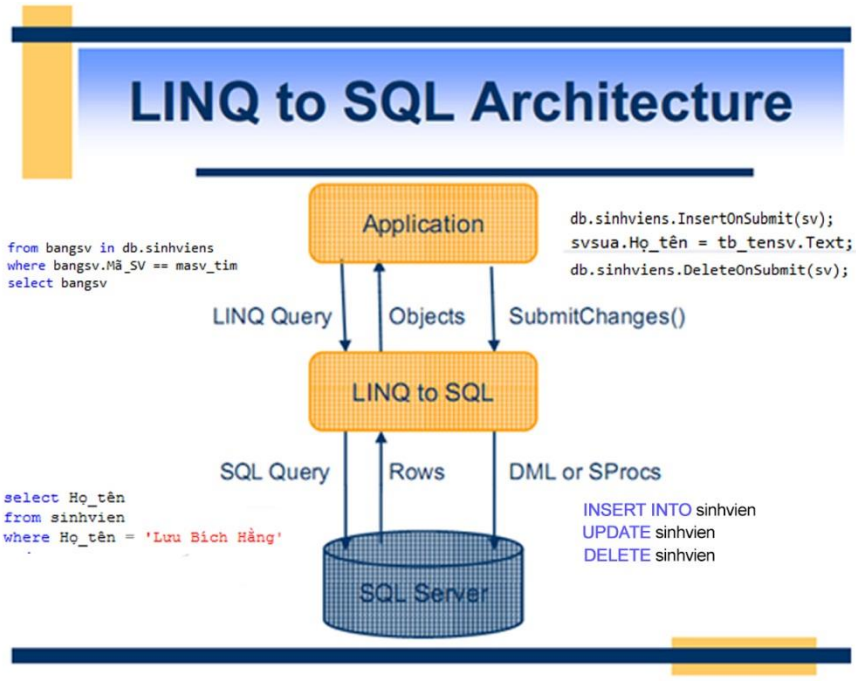
- Ánh xạ cơ sở dữ liệu theo hướng đối tượng



Hình 1.7 Mô hình ánh xạ của LinQ to SQL

1.6 Tầng kiến trúc

- Là cầu nối giao tiếp giữa Application và SQL Server



Hình 1.8 Tầng kiến trúc của LinQ to SQL

1.7 Các Actions của LinQ to SQL

- **Select Data :** Là hành động truy vấn và lấy dữ liệu từ cơ sở dữ liệu bằng câu lệnh LinQ to SQL.

Ví dụ:

```
var svnew = from bangsv in db.sinhviens
                select bangsv;
```

- **Insert Data :** Là hành động thêm mới cơ sở dữ liệu khi dữ liệu được nhập từ phần mềm vào cơ sở dữ liệu bằng câu lệnh LinQ to SQL.

Ví dụ:

```
sinhvien sv = new sinhvien();
sv.Mã_SV = tb_masv.Text;
sv.Họ_tên = tb_tensv.Text;
sv.Ngày_sinh = Convert.ToDateTime(tb_ngaysinh.Text);
sv.Mã_lớp = tb_malop.Text;
sv.Khóa_học = Convert.ToInt16(tb_khoahoc.Text);
db.sinhviens.InsertOnSubmit(sv);
db.SubmitChanges();
```

- **Update Data :** Là hành động sửa chữa dữ liệu trong cơ sở dữ liệu bằng câu lệnh LinQ to SQL.

Ví dụ:

```
//tìm sinh viên cần sửa có mã sinh viên được nhập vào từ bảng sinh viên
var svtim = (from bangsv in db.sinhviens
             where bangsv.Mã_SV == tb_masv.Text
             select bangsv);
//duyệt từng vật tư tìm được trong bảng sinh viên để thực hiện sửa các thông tin theo yêu cầu
foreach(var sv in svtim)
{
    sv.Họ_tên = tb_tensv.Text;
    sv.Ngày_sinh = Convert.ToDateTime(tb_ngaysinh.Text);
    sv.Mã_lớp = tb_malop.Text;
    sv.Khóa_học = Convert.ToInt16(tb_khoahoc.Text);
}
```

- **Delete Data :** Là hành động xóa dữ liệu trong cơ sở dữ liệu bằng câu lệnh LinQ to SQL.

Ví dụ:

```
var svtim = (from bangsv in db.sinhviens
             where bangsv.Mã_SV == tb_masv.Text
             select bangsv);
//duyệt sinh viên tìm được trong bảng sinh viên để xóa
foreach (var sv in svtim)
{
    db.sinhviens.DeleteOnSubmit(sv);
}
db.SubmitChanges();
```

- **Join :** Là hành động liên kết các bảng trong cơ sở dữ liệu bằng câu lệnh LinQ to SQL.

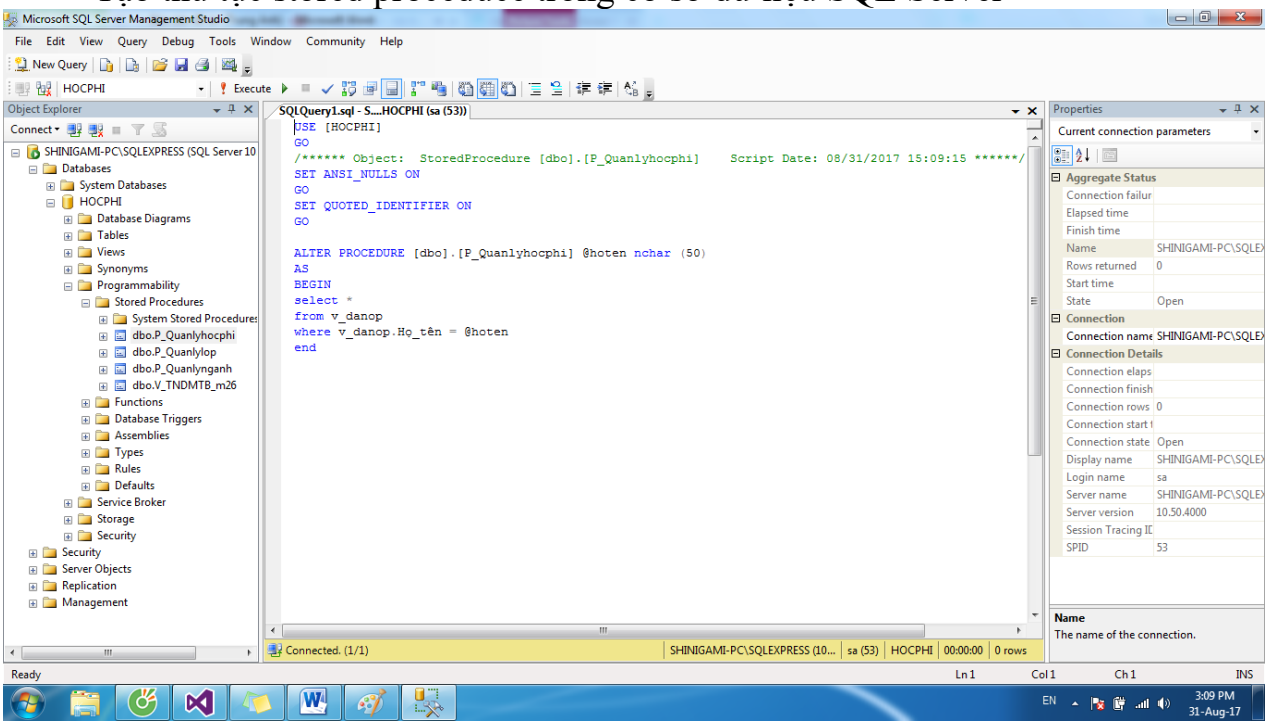
Ví dụ:

```
Table<lop> lp = db.GetTable<lop>();
Table<sinhvien> Sinhviens = db.GetTable<sinhvien>();
Table<nganh> ng = db.GetTable<nganh>();
//tạo truy vấn
if (tb_masv.Text != "")
{
    var query = from k in lp
                join sv in Sinhviens on k.Mã_lớp equals sv.Mã_lớp
                join n in ng on k.Mã_ngành equals n.Mã_ngành
                where sv.Mã_SV == tb_masv.Text
                group k by new { k.Mã_lớp, sv.Mã_SV, sv.Họ_tên, sv.Khóa_học, sv.Ngày_sinh, n.Tên_ngành  } into nh
                select new { nh.Key.Mã_SV, nh.Key.Họ_tên, nh.Key.Ngày_sinh, nh.Key.Khóa_học, nh.Key.Mã_lớp, nh.Key.Tên_ngành };
    dtg_ket_qua.DataSource = query;
}
```

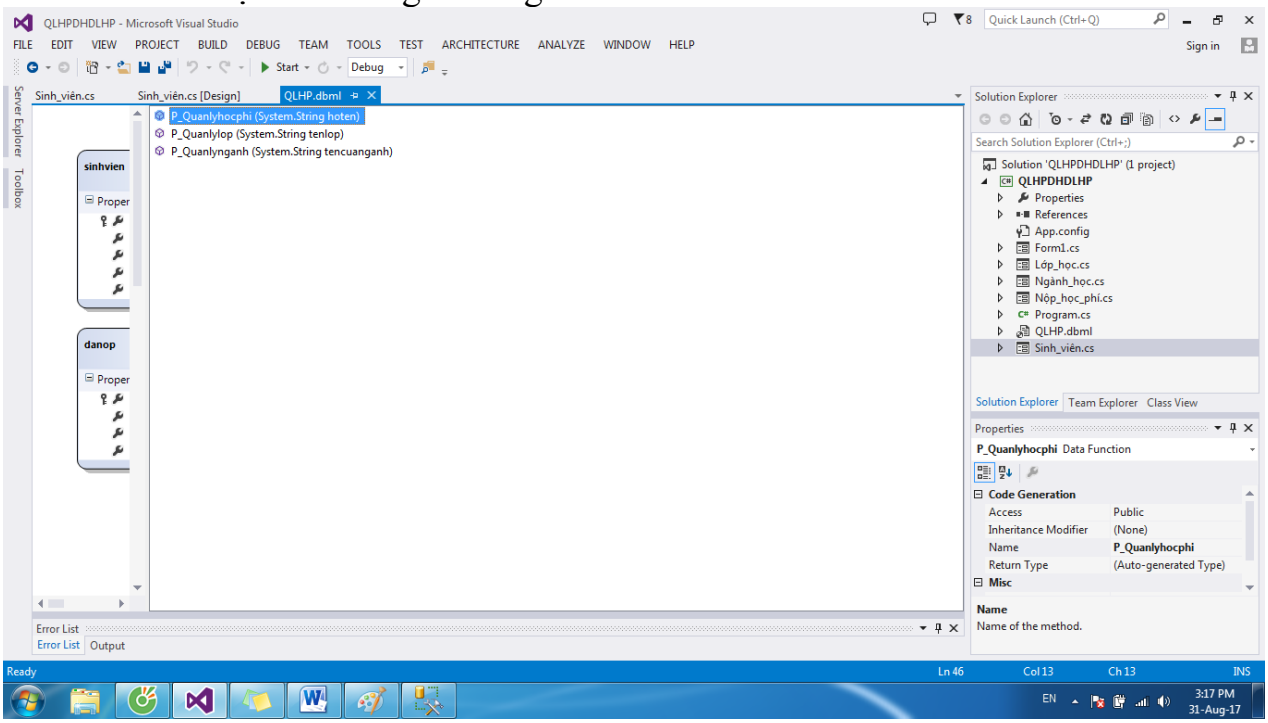
- **Using stored proceduce:** là hành động lấy dữ liệu từ cơ sở dữ liệu bằng cách gọi thủ tục được tạo ra trong cơ sở dữ liệu.

Ví dụ:

+ Tạo thủ tục stored proceduce trong cơ sở dữ liệu SQL Server



+ Đưa thủ tục vào trong chương trình:



+ Dùng câu lệnh LinQ to SQL để gọi thủ tục đã tạo

```
private void lb_timsinhvien_SelectedIndexChanged(object sender, EventArgs e)
{
    //gọi thủ tục để tìm thông tin các sinh viên đã nhập vào để có tên sinh viên đang chọn
     var kqtim = db.P_Quanlyhocphi(lb_timsinhvien.Text);
     dtg_ket_qua.DataSource = kqtim;

}
```

CHƯƠNG 2

XÂY DỰNG CHƯƠNG TRÌNH ỨNG DỤNG

2.1 Xây dựng chương trình ứng dụng

2.1.1 Phát biểu bài toán

Hoạt động quản lí học phí của sinh viên đại học Dân lập Hải Phòng được mô tả như sau :

a) Trong trường có rất nhiều ngành học khác nhau, thông tin về mỗi ngành học bao gồm: *Mã ngành, Tên ngành*. Trong đó *Mã ngành* xác định duy nhất mỗi ngành.

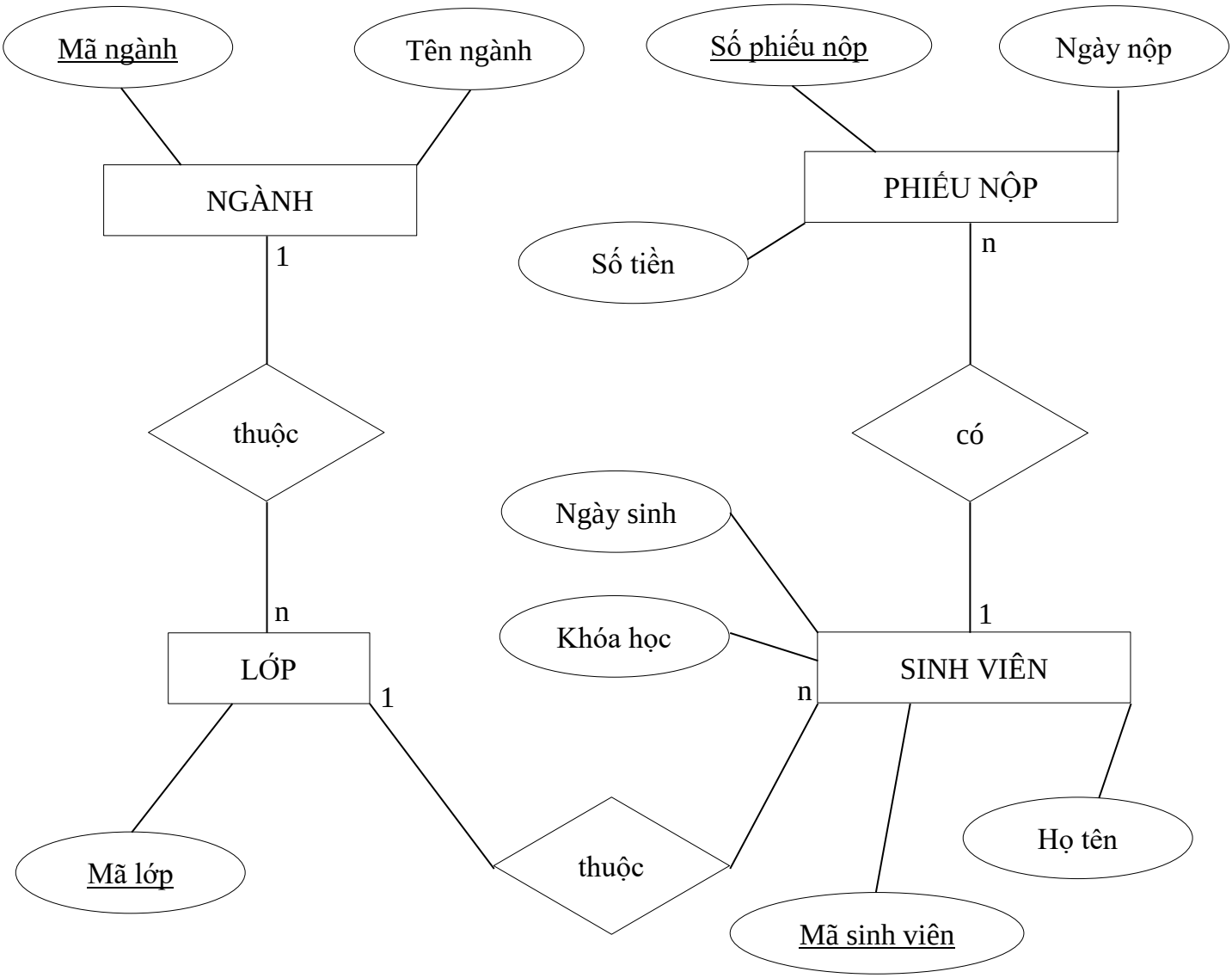
b) Có rất nhiều lớp học khác nhau, mỗi lớp có một *Mã lớp* duy nhất. Nhiều lớp học thuộc một ngành đào tạo.

c) Trong trường có rất nhiều sinh viên, thông tin về mỗi sinh viên bao gồm: *Mã sinh viên, Họ tên sinh viên, Ngày sinh, Khóa học*. Trong đó *Mã sinh viên* xác định duy nhất mỗi sinh viên. Nhiều sinh viên thuộc một lớp.

d) Khi sinh viên nộp tiền học phí thì thông tin sẽ được ghi lại gồm: *Số phiếu nộp, Ngày nộp học phí, Số tiền sinh viên nộp*. Trong đó *Số phiếu nộp* xác định duy nhất. Mỗi lần sinh viên nộp học phí thì sẽ có một phiếu nộp duy nhất được in ra.

2.1.2 Thiết kế cơ sở dữ liệu

a) Vẽ mô hình ER



Hình 2.1 Mô hình ER của bài toán

b) Chuyển đổi mô hình ER thành các bảng quan hệ

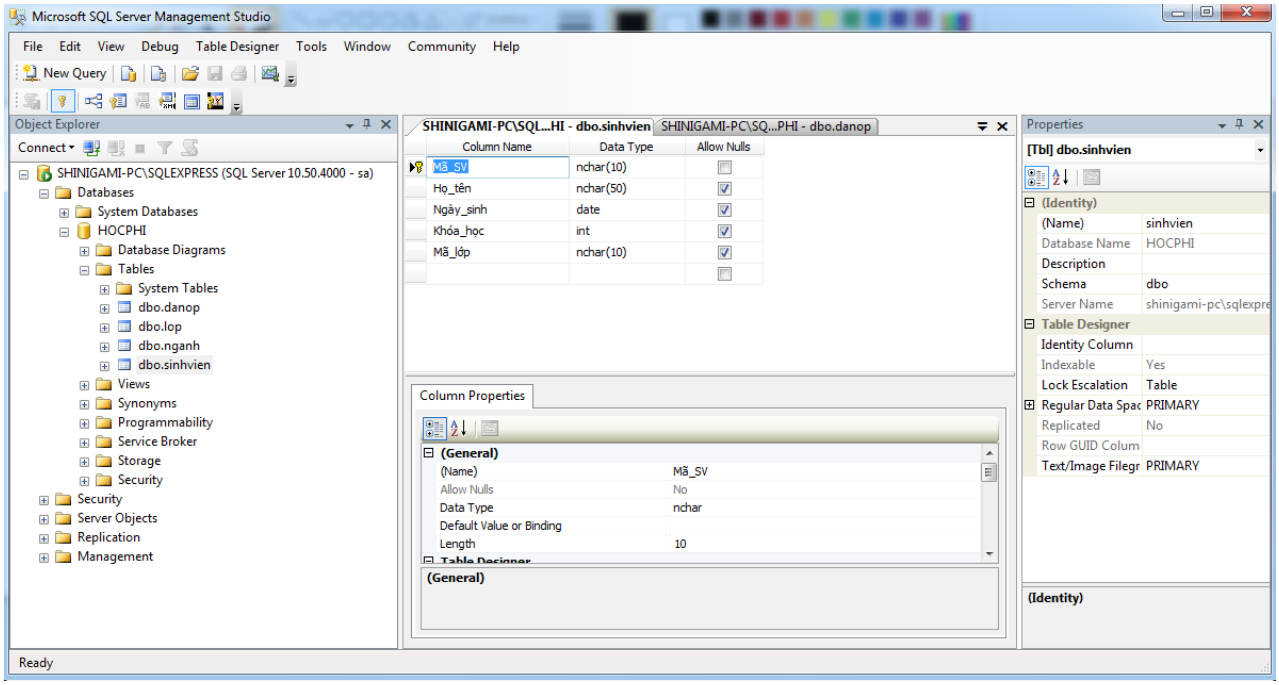
Mô hình ER được chuyển đổi thành các bảng quan hệ như sau:

+ Bảng NGÀNH để lưu trữ các thông tin về ngành học gồm các cột: Mã ngành, Tên ngành. Khóa chính là Mã ngành.
+ Bảng LỚP để lưu trữ các thông tin về lớp học gồm các cột: Mã lớp, Mã ngành. Khóa chính là Mã lớp, khóa ngoài là Mã ngành.
+ Bảng SINH VIÊN để lưu trữ thông tin về sinh viên gồm các cột: Mã sinh viên, Họ tên, Ngày sinh, Khóa học, Mã lớp. Khóa chính là Mã sinh viên, khóa ngoài là Mã lớp
+ Bảng PHIẾU NỘP để lưu trữ thông tin về các phiếu thu của sinh viên gồm các cột: Số phiếu, Ngày nộp, Số tiền, Mã sinh viên. Trong đó Số phiếu là khóa chính, khóa ngoài là Mã sinh viên

2.1.3 Tạo cơ sở dữ liệu bằng SQL Server

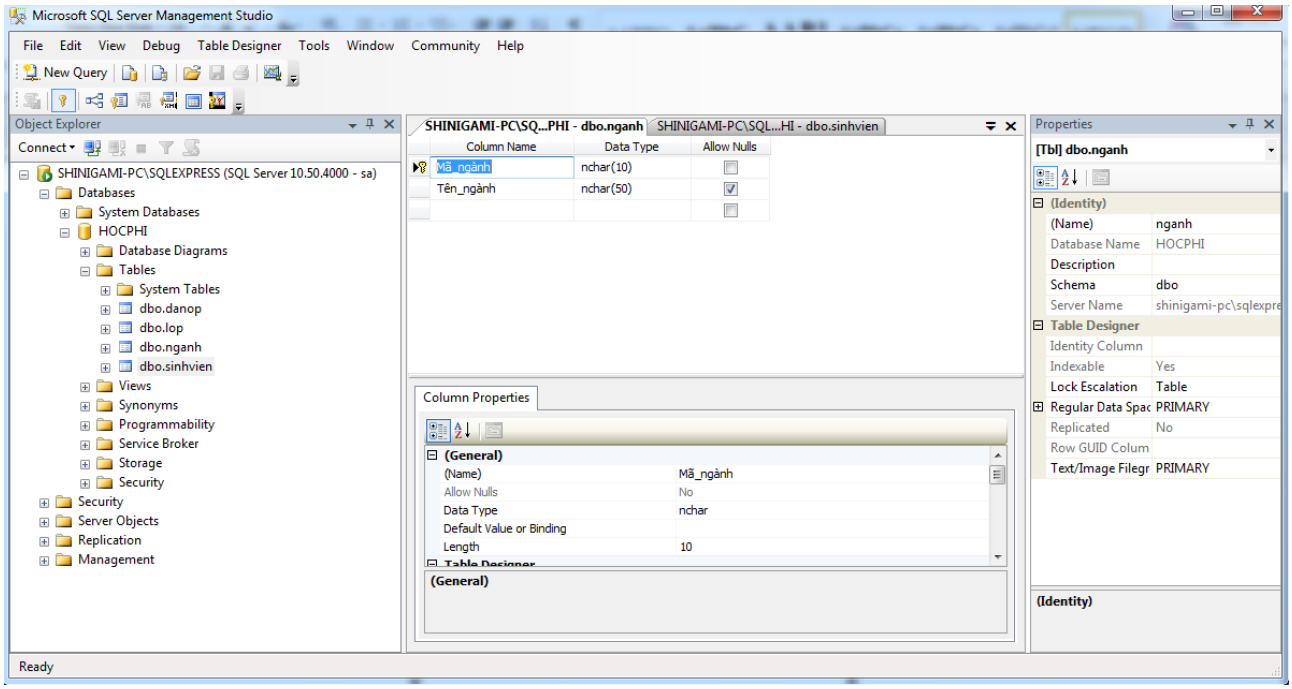
Từ các bảng quan hệ trên, ta tạo được một DATABASE HOCPHI gồm các bảng như sau:

1, Bảng SINH VIÊN



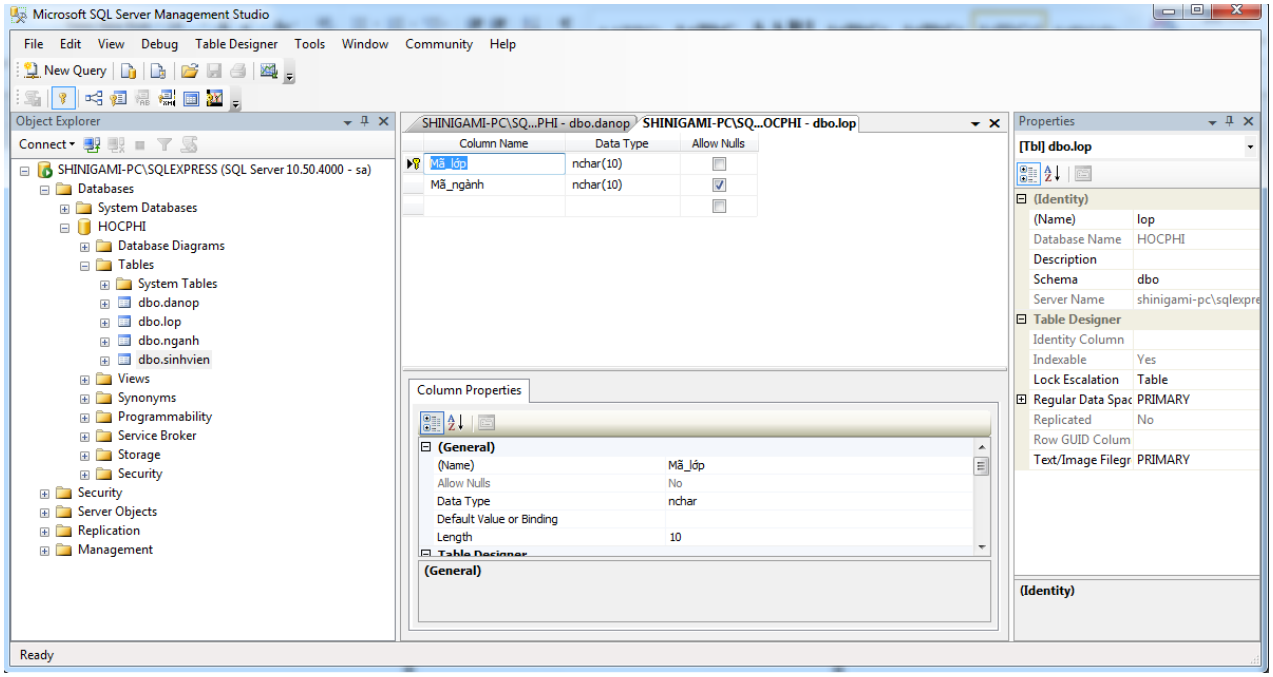
Hình 2.2 Bảng "Sinh viên" trong cơ sở dữ liệu HOCPHI

2, Bảng NGÀNH



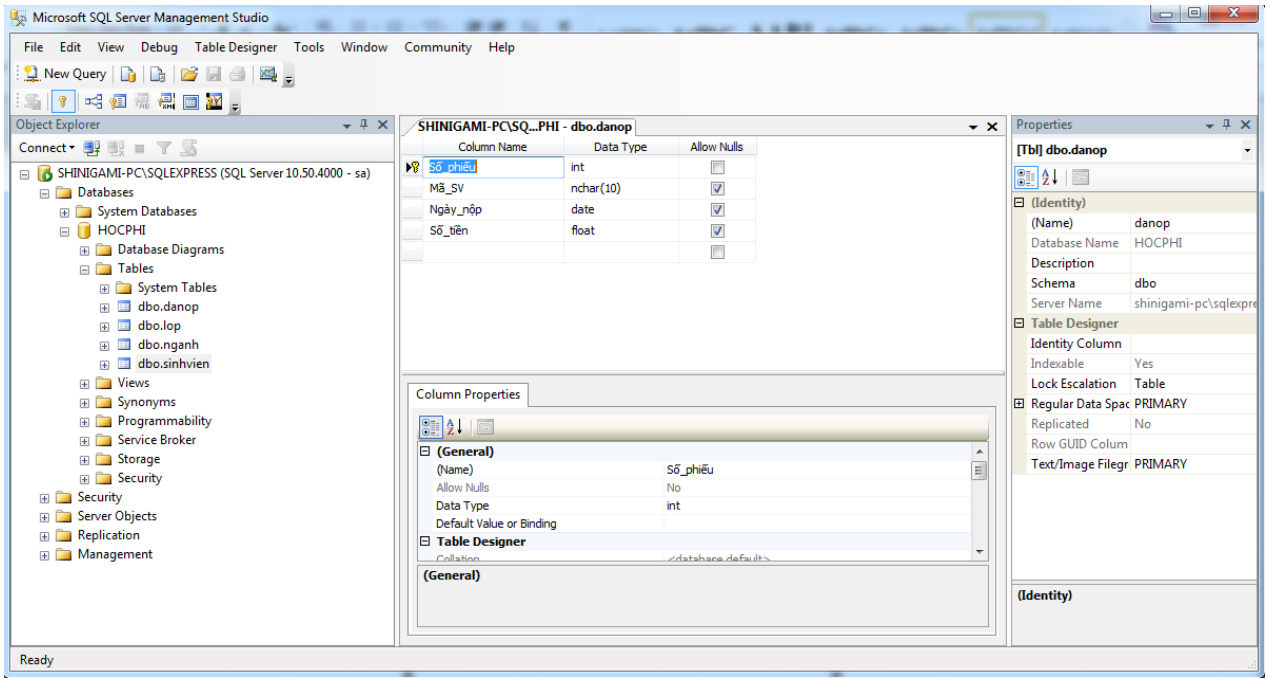
Hình 2.3 Bảng "Ngành" trong cơ sở dữ liệu HOCPHI

3, Bảng LỚP



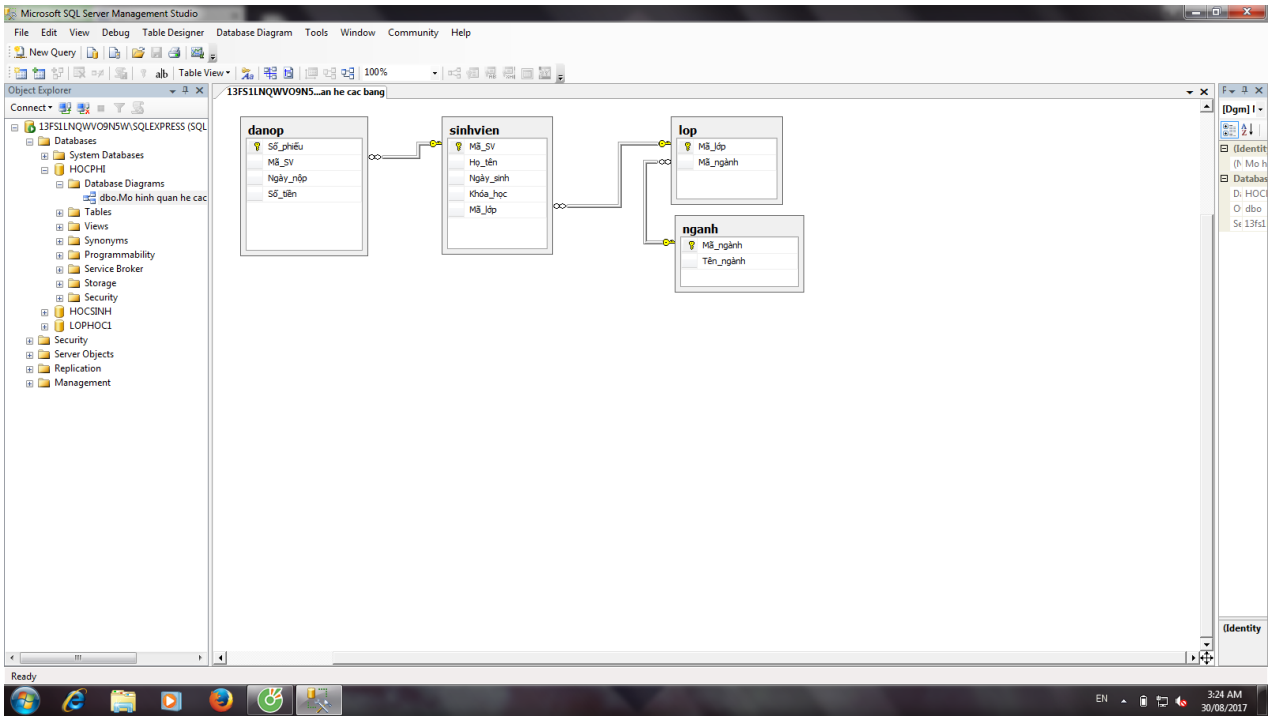
Hình 2.4 Bảng "Lớp" trong cơ sở dữ liệu HOCPHI

4, Bảng PHIẾU NỘP



Hình 2.5 Bảng nộp học phí trong cơ sở dữ liệu HOCPHI

5, Mô hình quan hệ các bảng trong cơ sở dữ liệu

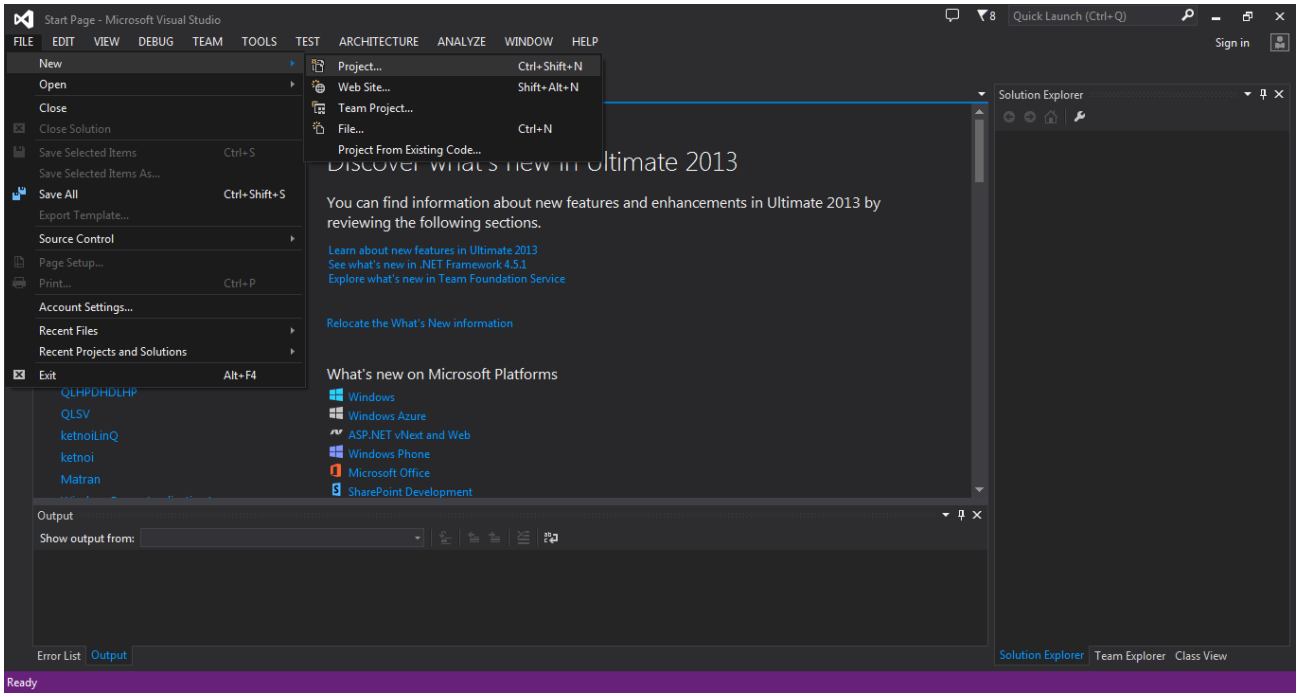


Hình 2.6 Mô hình quan hệ các bảng trong cơ sở dữ liệu HOCPHI

2.2 Xây dựng chương trình

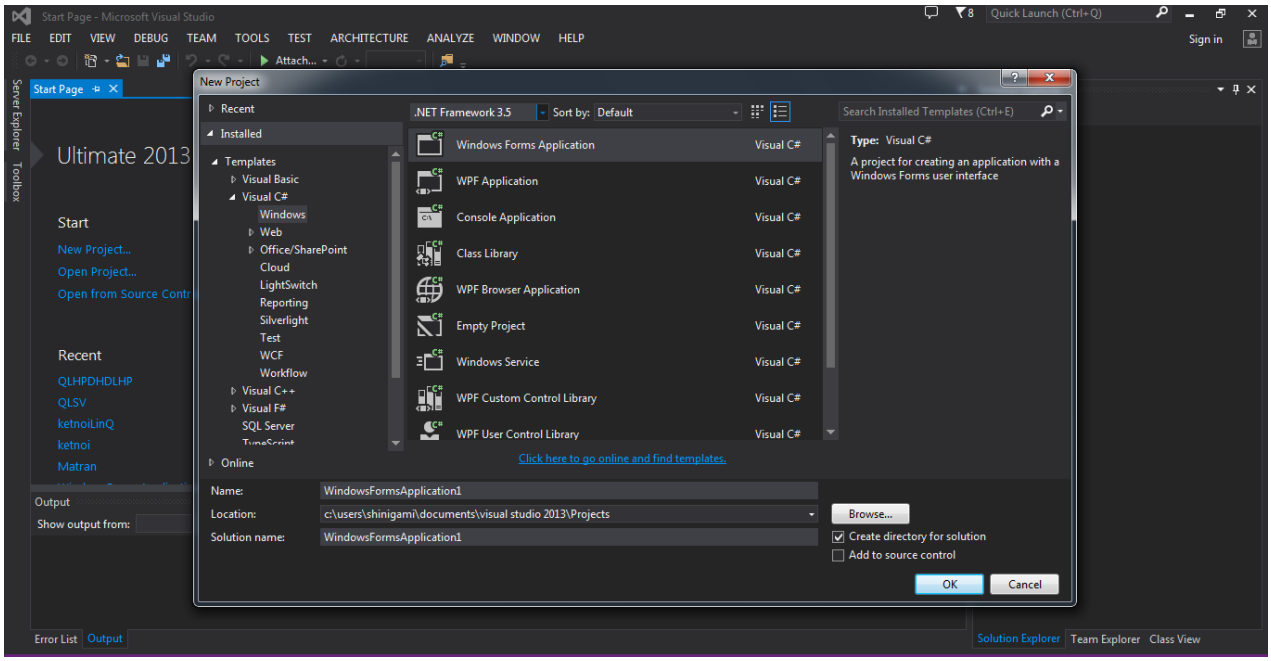
2.2.1 Tạo chương trình và lập kết nối LINQ to SQL với SQL SERVER

Bước 1 : Tạo giao diện phần mềm làm việc
Chọn **FILE → NEW → PROJECT**

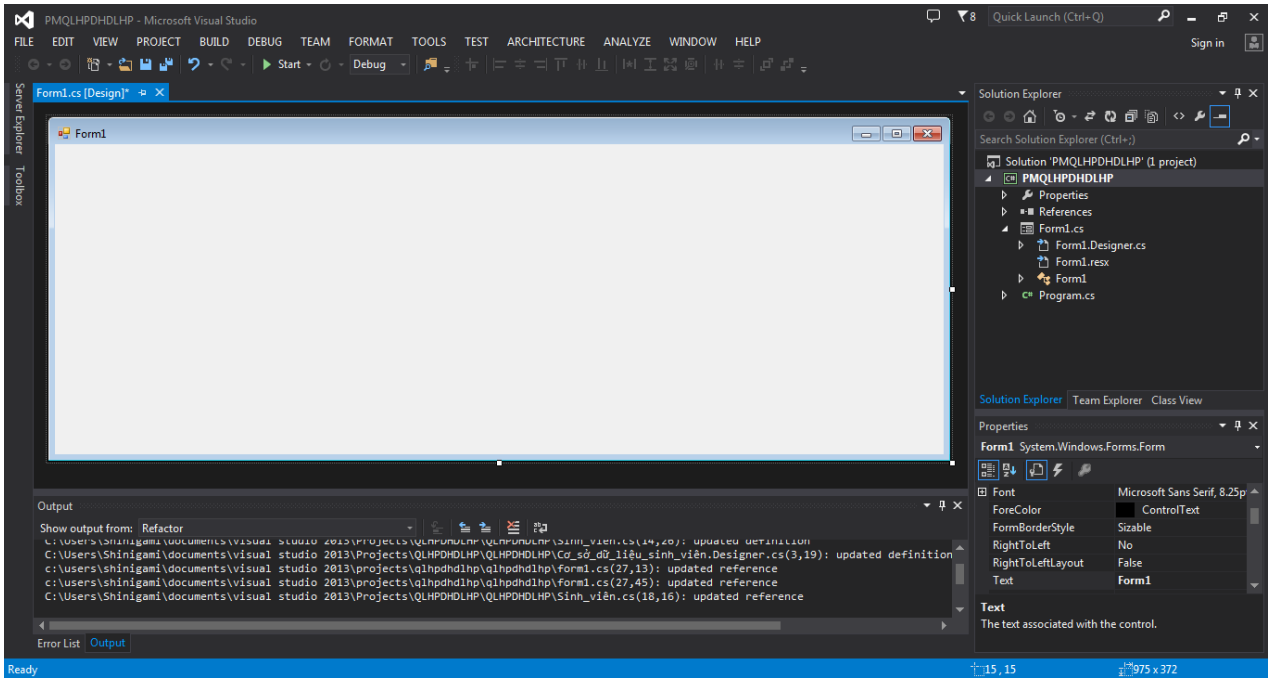


Hình 2.7 Giao diện tạo Project làm việc trong Visual Studio 2013

+ Chọn **Installed → Windows → Windows Forms Application** (trên nền .NET Framework 3.5)



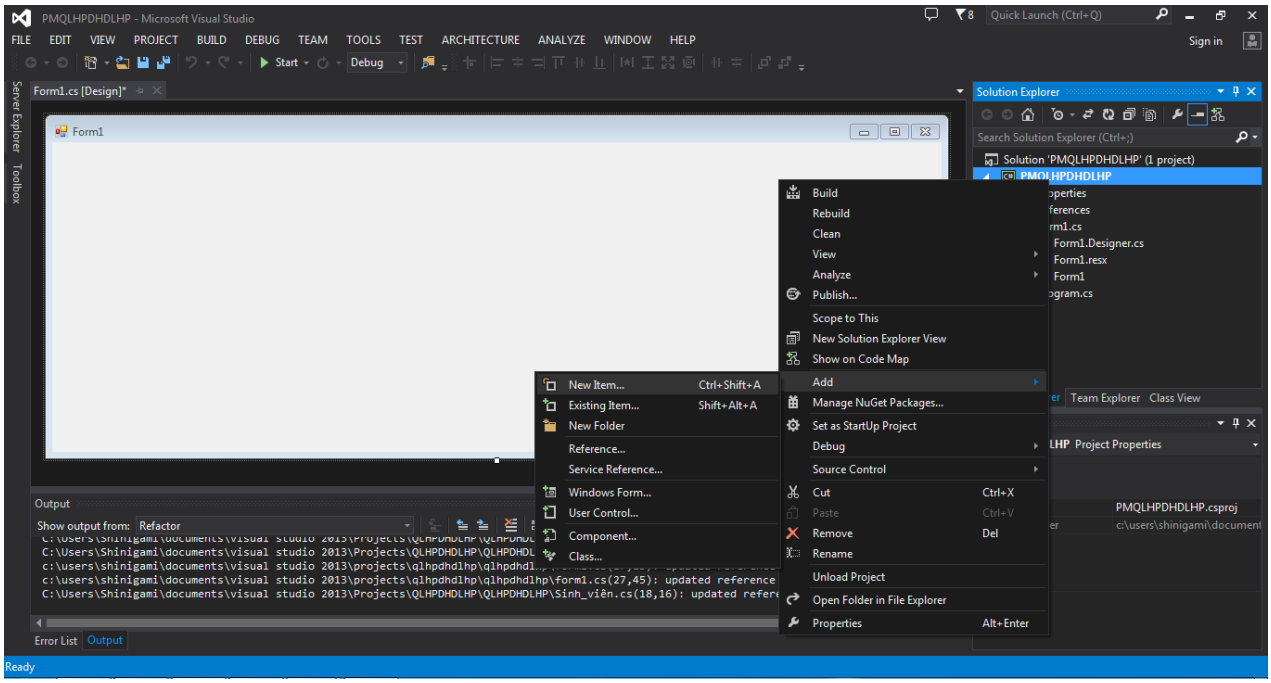
Hình 2.8 Tạo Project làm việc



Hình 2.9 Giao diện Project làm việc sau khi tạo xong

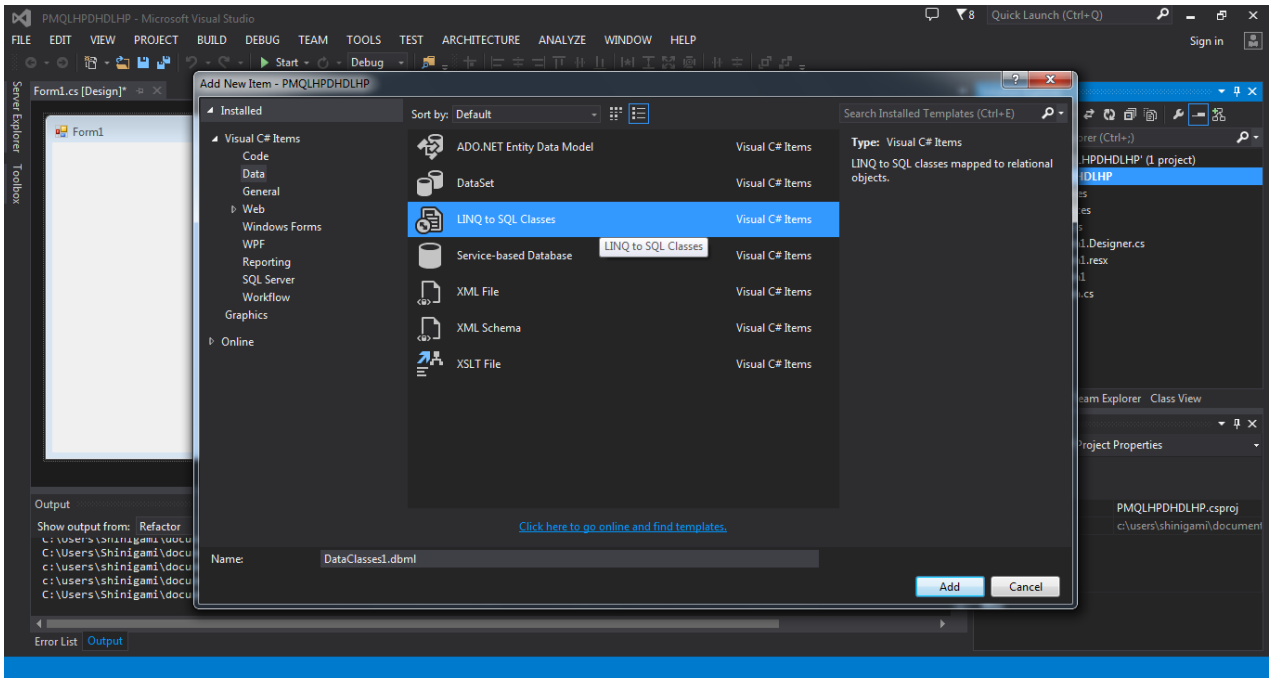
Bước 3: Tạo kết nối LINQ to SQL với SQL Server

+ Click chuột phải vào tên của **Project → Chọn ADD → Chọn New Item (Ctrl + Shift + A)**

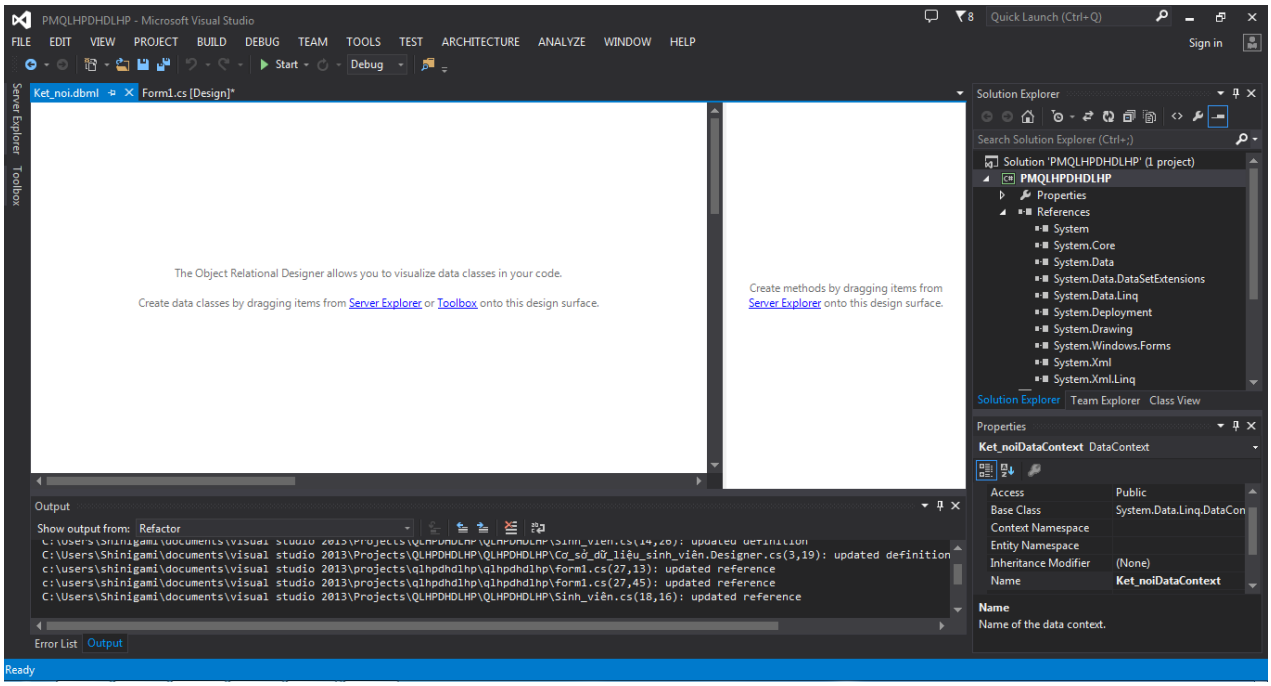


Hình 2.10 Tạo kết nối LinQ

+ Chọn Data → Chọn LINQ to SQL Classes

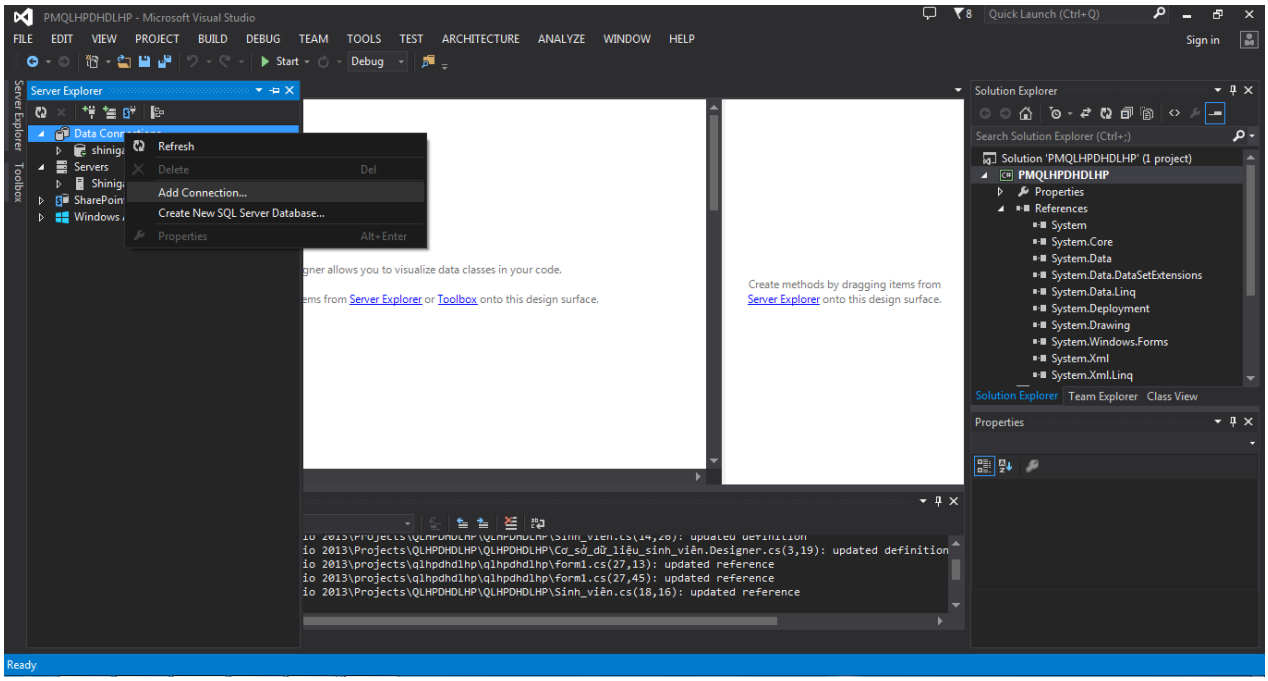


Hình 2.11 Tạo kết nối LinQ to SQL



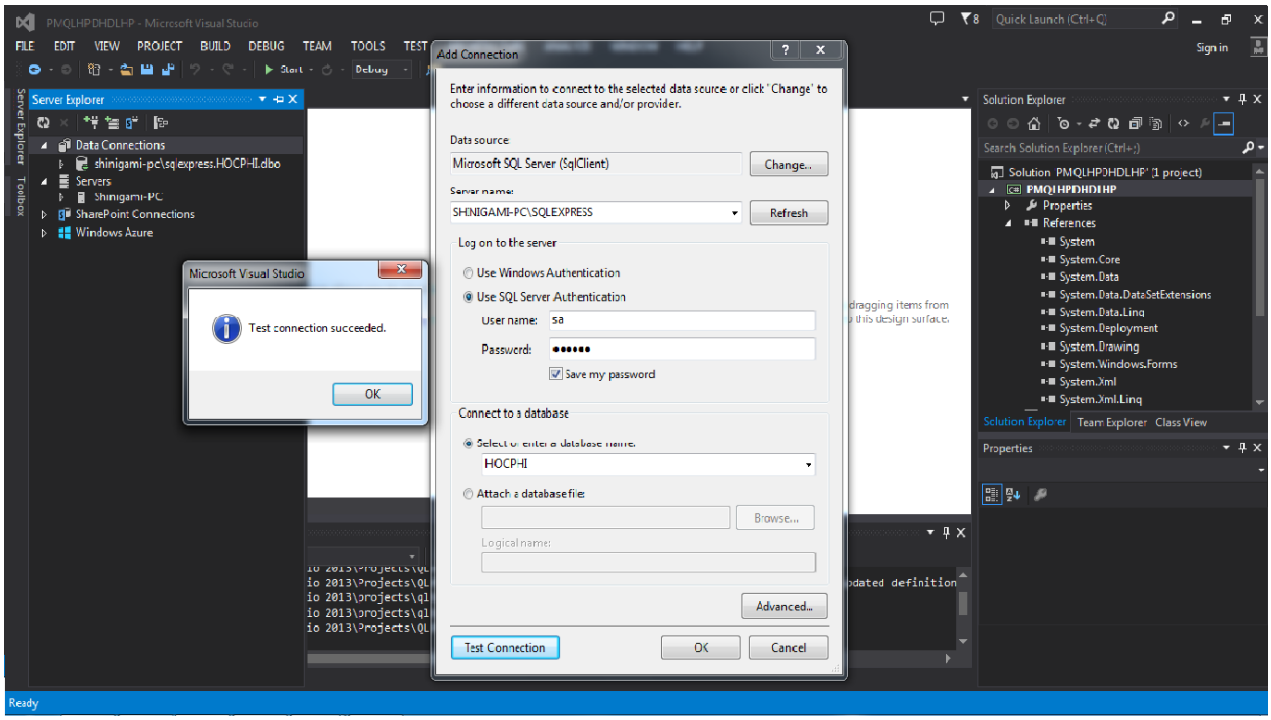
Hình 2.12 Giao diện khi tạo Form để kết nối LINQ to SQL thành công

+ Tạo kết nối LINQ to SQL với phần mềm Microsoft SQL server 2008 R2 bằng cách: Click vào Server Explorer → Chọn Data Connections → Chọn Add Connection……



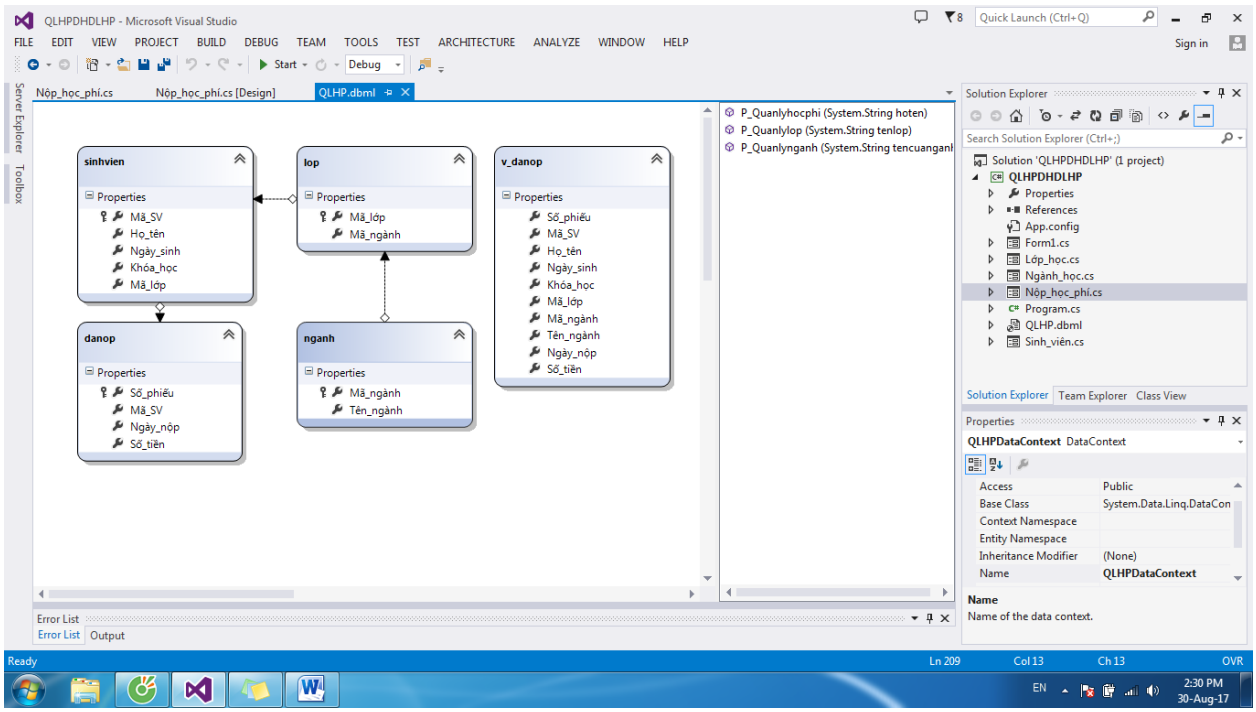
Hình 2.13 Kết nối LINQ to SQL

+ Kết nối với cơ sở dữ liệu HOCPHI và Test Connection thành công



Hình 2.14 Kết nối LINQ to SQL thành công

+ Các bảng và view sau khi kết nối thành công



Hình 2.15 Các bảng và view sau khi kết nối LINQ to SQL thành công

2.2.2 Tạo Form Menu và các Form thành phần của chương trình

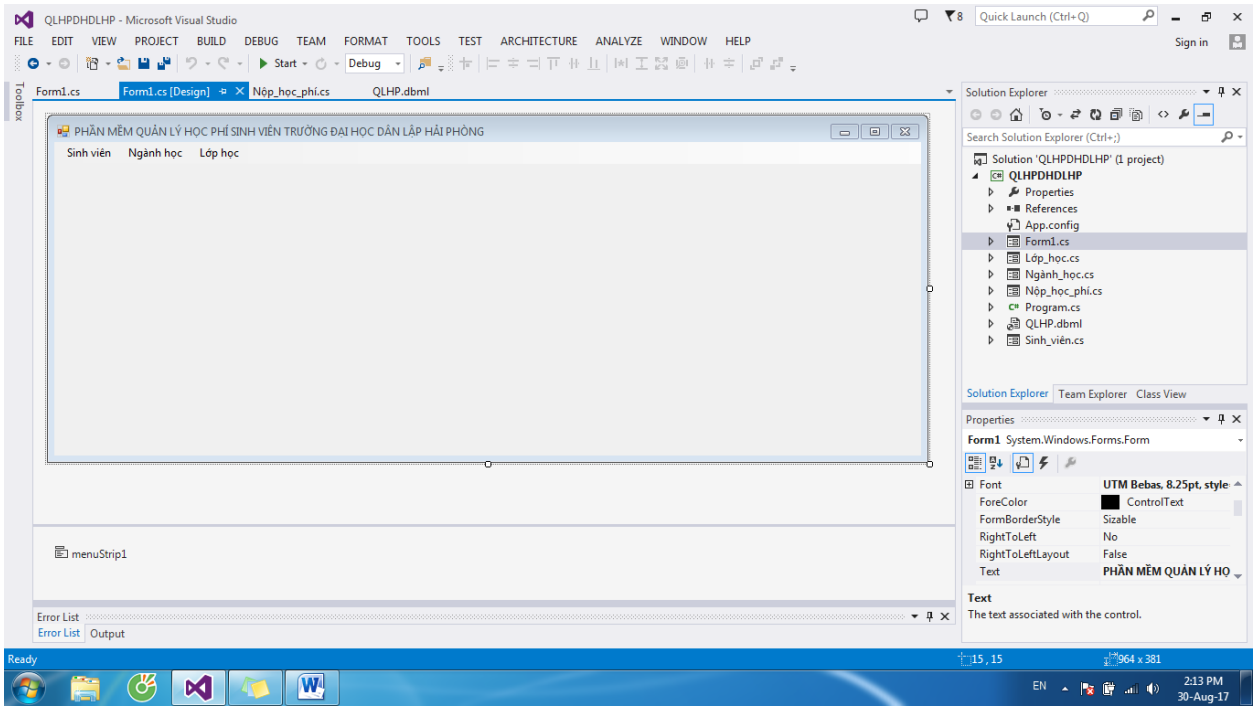
❖ **Giao diện Form Menu**

+ Thay vì phải chạy các Form của chương trình chúng ta sẽ viết một Form Menu để khi chạy chương trình chúng ta chỉ cần chạy Form này là sẽ gọi được 4 Form còn lại.

+ Đổi tên Form 1 thành "PHẦN MỀM QUẢN LÝ HỌC PHÍ SINH VIÊN TRƯỜNG ĐẠI HỌC DÂN LẬP HẢI PHÒNG" bằng cách:

Click chuột phải vào khoảng trống trong form => Chọn Properties => Chọn phần "Text" => Đổi tên.

+ Từ phần Toolbox => Chọn Menu & Toolbar => Chọn MenuStrip để làm thanh Menu chọn các Form tương ứng.

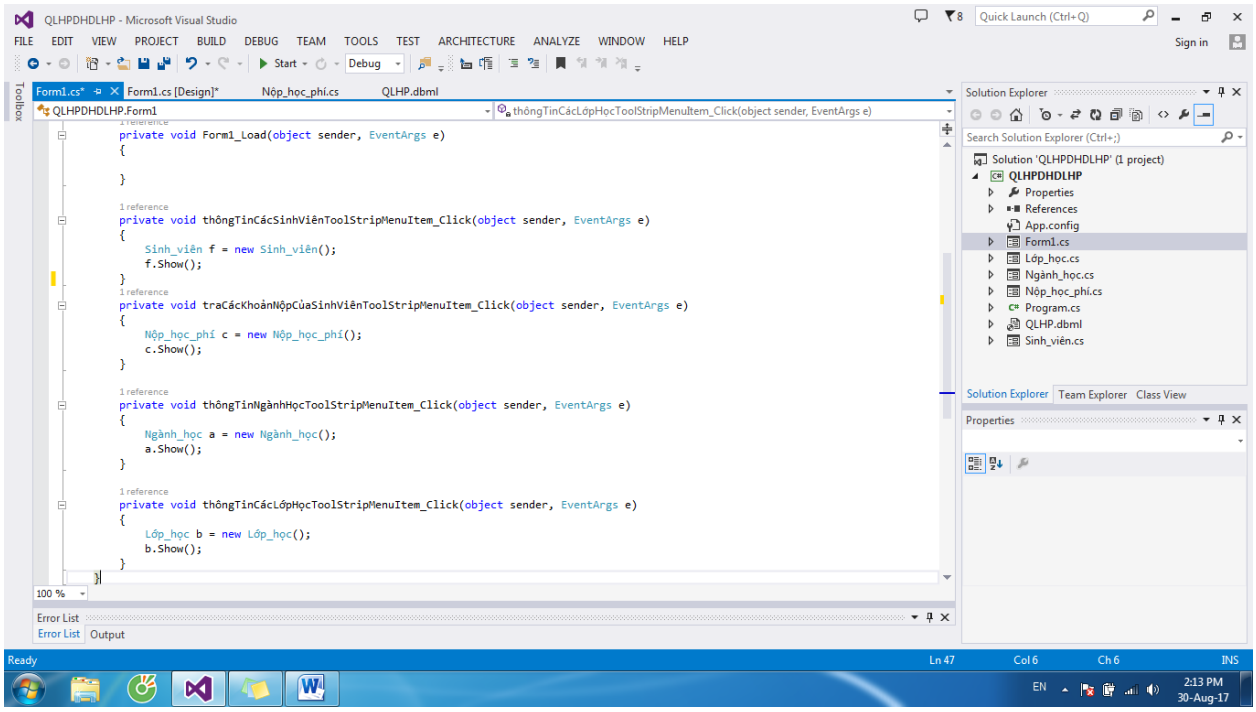


Hình 2.16 Giao diện Form Menu

❖ **Code chạy giao diện Menu**

```
    private void thôngTinCácSinhViênToolStripMenuItem_Click(object sender, EventArgs e)
{
    Sinh_viên f = new Sinh_viên();
    f.Show();
}
    private void tra CácKhoảnNộpCủaSinhViênToolStripMenuItem_Click(object sender,
EventArgs e)
```

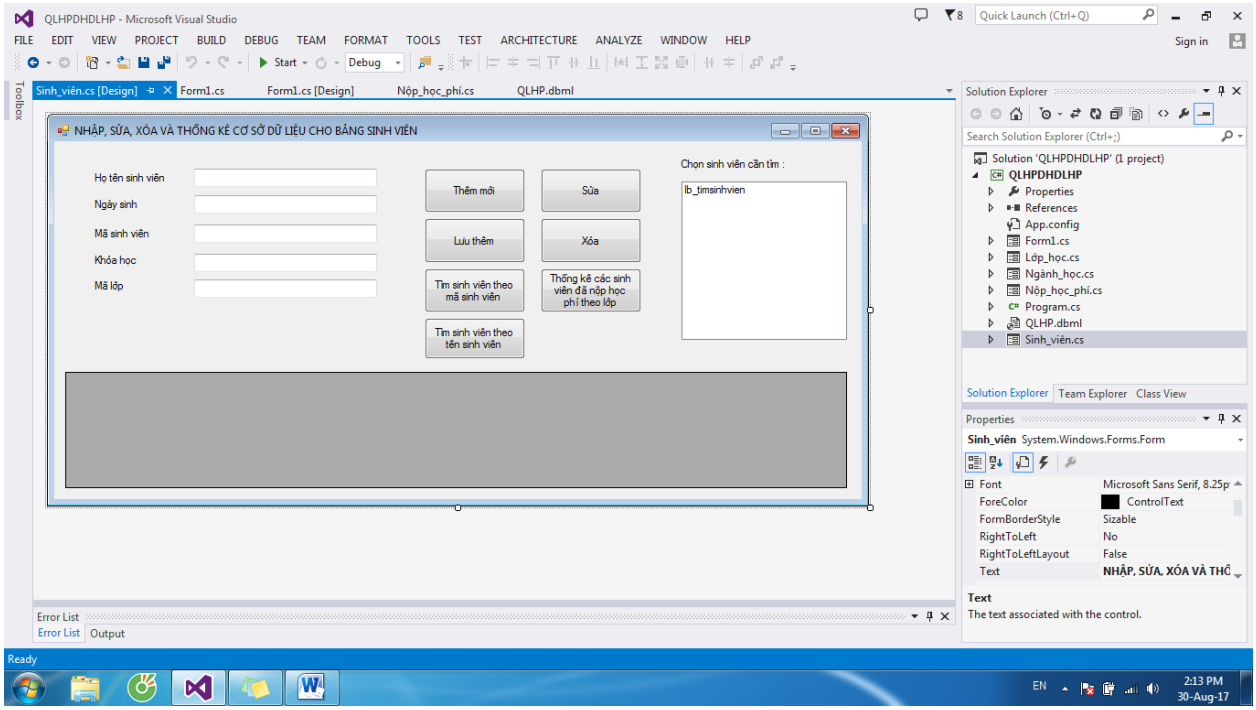
```
{
   Nộp_học_phí  c = new Nộp_học_phí ();
   c.Show();
}
   private void thôngTinNgànhHọcToolStripMenuItem_Click(object sender, EventArgs e)
{
   Ngành_Học a = new Ngành_Học();
   a.Show();
}
   private void thôngTinNgànhHọcToolStripMenuItem_Click(object sender, EventArgs e)
{
   Lớp_Học b = new Lớp_Học();
   b.Show();
}
```



Hình 2.17 Giao diện code của Form Menu

❖ Giao diện Form Sinh viên

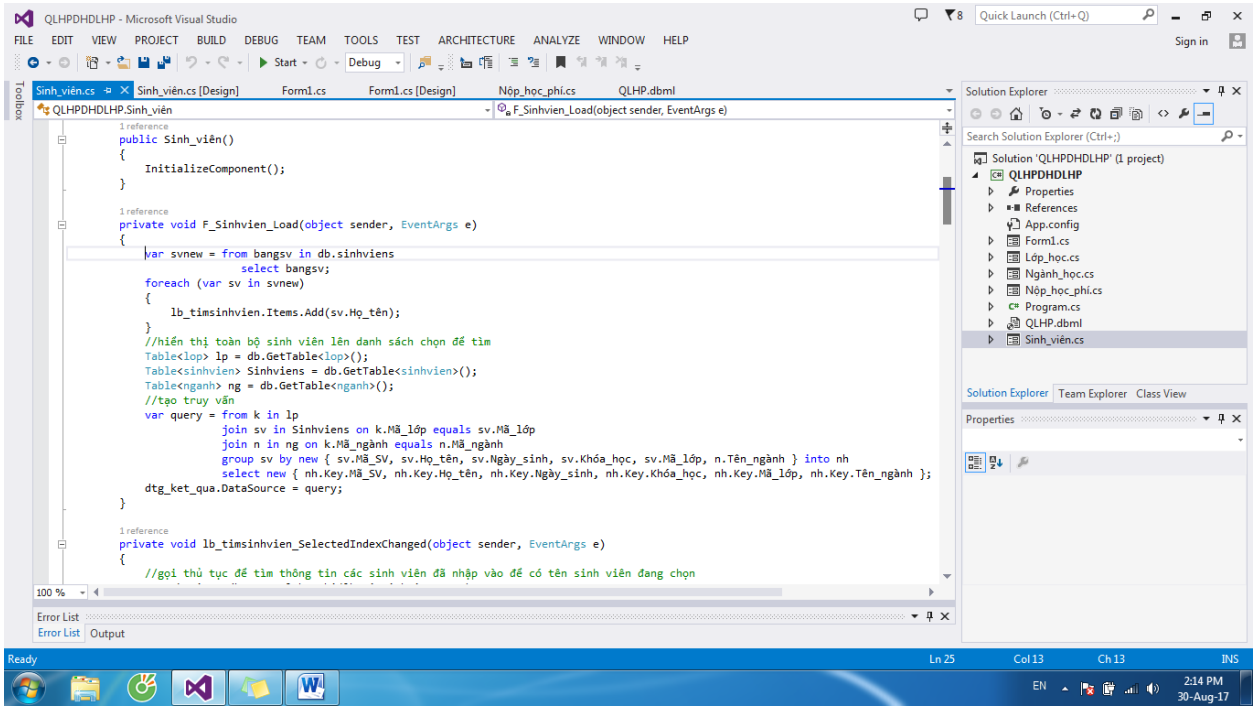
+ Form sinh viên bao gồm nút có chức năng thêm, sửa, xóa, lưu, tìm kiếm và thống kê dữ liệu của bảng SINHVIEN trong cơ sở dữ liệu HOCPHI.
+ Ở đây em có tạo thêm 1 List Box để hỗ trợ tìm kiếm nhanh tên của các sinh viên.



Hình 2.18 Giao diện Form Sinh viên

❖ Tạo truy vấn cho Form sinh viên

```
private void F_Sinhvien_Load(object sender, EventArgs e)
        {
            //Sử dụng Action Select để liên kết giữa các bảng để lấy dữ liệu
            var svnew = from bangsv in db.sinhviens
                        select bangsv;
            foreach (var sv in svnew)
            {
                lb_timsinhvien.Items.Add(sv.Họ_tên);
            }
            //Tạo truy vấn dữ liệu
            Table<lop> lp = db.GetTable<lop>();
            Table<sinhvien> Sinhviens = db.GetTable<sinhvien>();
            Table<nganh> ng = db.GetTable<nganh>();
            //Sử dụng Action Join để liên kết giữa các bảng để lấy dữ liệu
            var query = from k in lp
                        join sv in Sinhviens on k.Mã_lớp equals sv.Mã_lớp
                        join n in ng on k.Mã_ngành equals n.Mã_ngành
                        group sv by new { sv.Mã_SV, sv.Họ_tên, sv.Ngày_sinh,
sv.Khóa_học, sv.Mã_lớp, n.Tên_ngành } into nh
                        select new { nh.Key.Mã_SV, nh.Key.Họ_tên, nh.Key.Ngày_sinh,
nh.Key.Khóa_học, nh.Key.Mã_lớp, nh.Key.Tên_ngành };
            dtg_ket_qua.DataSource = query;
        }
```

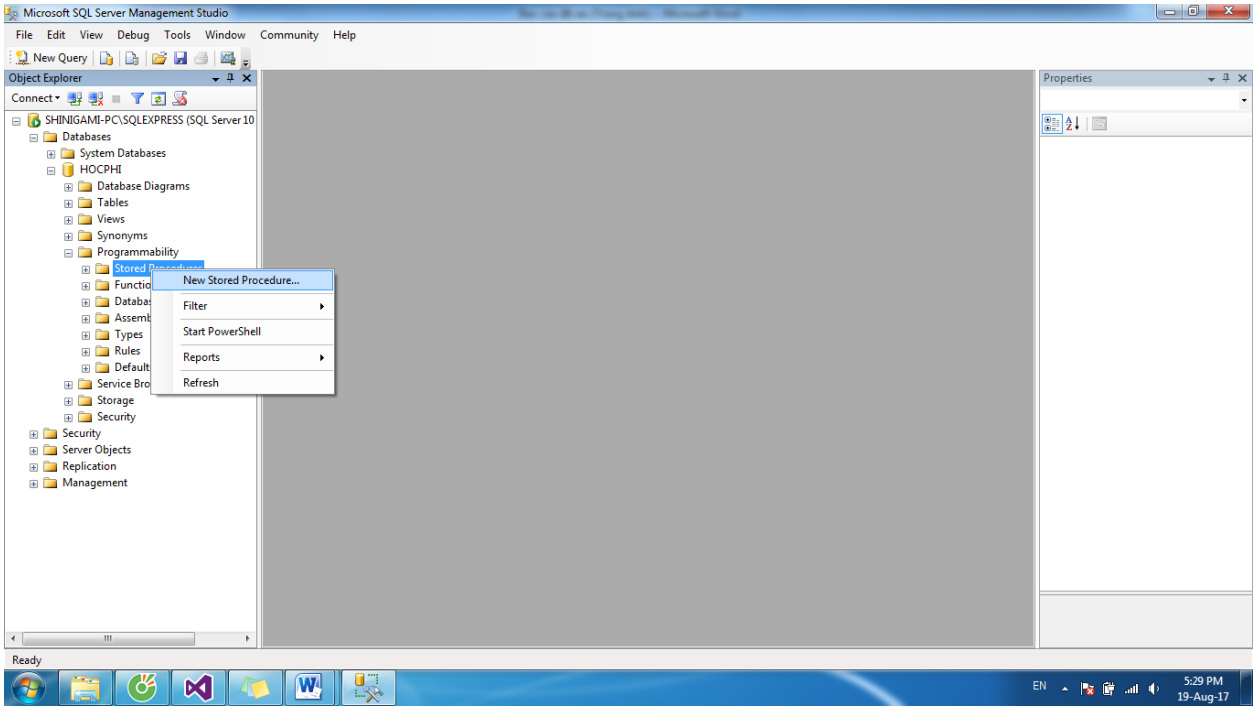


Hình 2.19 Tạo truy vấn cho Form sinh viên

- **Tạo thủ tục Stored Prceduces để truy vấn dữ liệu cho List Box của bảng sinh viên**

Bước 1: đăng nhập vào phần mềm quản trị cơ sở dữ liệu SQL Server, chọn Database HOCPHI

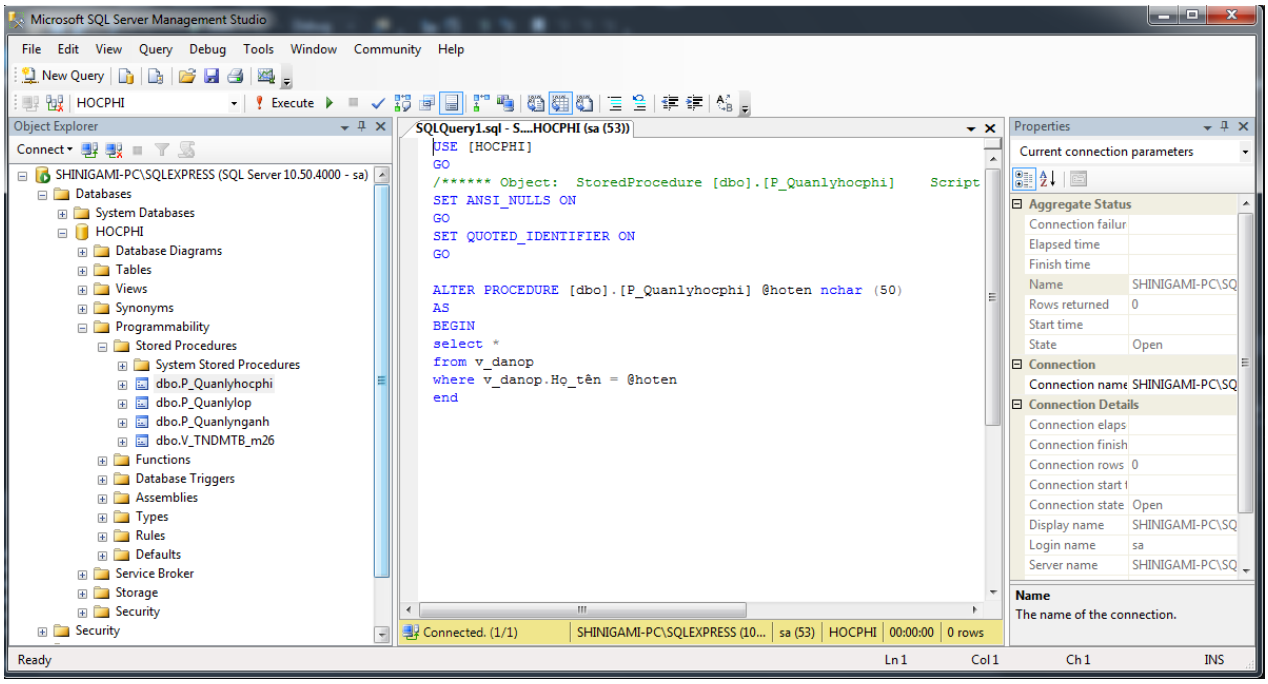
Chọn **Programmability** => Click chuột phải vào **Stored Prceduces** => **New Stored Prceduces**



Hình 2.20 Tạo thủ tục Stored proceduces truy vấn theo tên sinh viên

Bước 2: Viết câu lệnh truy vấn có tham số là họ tên sinh viên để tạo thủ tục **Stored Prceduces**

Bước 3: Tương tự ta viết câu lệnh truy vấn có tham số là mã lớp và mã ngành để tạo thủ tục **Stored Prceduces** cho Form Lớp và Form ngành.

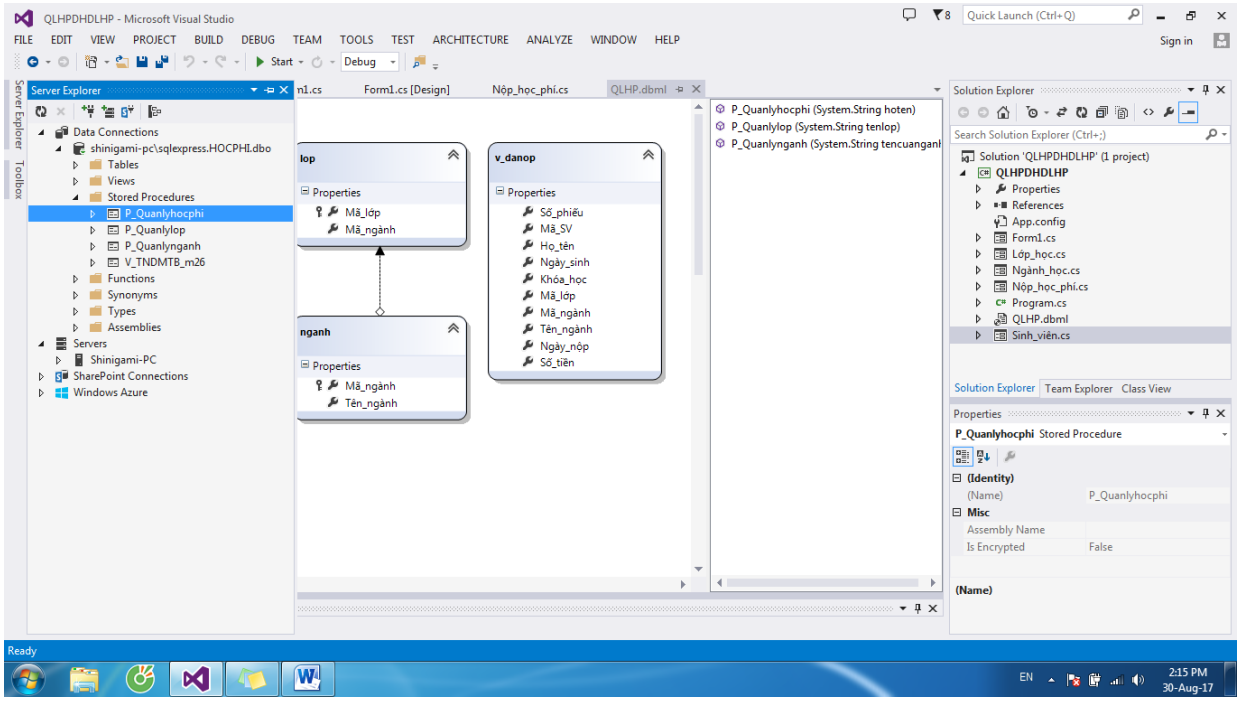


Hình 2.21 Tạo thủ tục Stored proceduces truy vấn theo tên sinh viên

❖ **Đưa thủ tục Stored Prceduces vào LinQ to SQL để truy vấn dữ liệu theo tên Sinh viên của List Box**

+ Tương tự như đưa các bảng và View vào trong LINQ to SQL, chúng ta cũng đưa các thủ tục **Stored Prceduces** vào để truy vấn dữ liệu.

+ Dùng **Stored Prceduces** để truy vấn dữ liệu trong LINQ to SQL để có thể tăng tính bảo mật, tăng thời gian viết code, giảm thiểu tình trạng quá tải cho phần mềm quản trị cơ sở dữ liệu

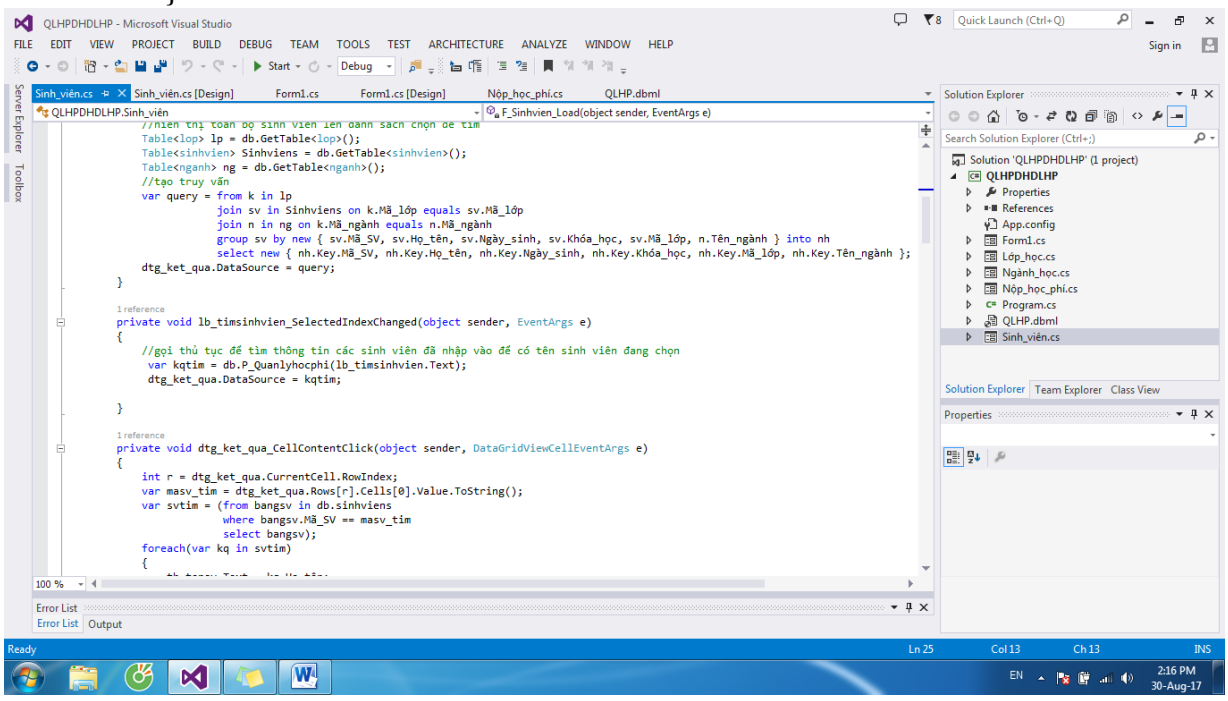


Hình 2.22 Đưa thủ tục Stored Prceduces vào LinQ to SQL

❖ Gọi thủ tục Stored Prceduces bằng câu lệnh LinQ to SQL

Đoạn code như sau:

```
private void lb_timsinhvien_SelectedIndexChanged(object sender, EventArgs e)
        {
            //gọi thủ tục để tìm thông tin các sinh viên đã nhập vào để có tên sinh viên đang
chọn
            var kqtim = db.P_Quanlyhocphi(lb_timsinhvien.Text);
            dtg_ket_qua.DataSource = kqtim;
        }
```



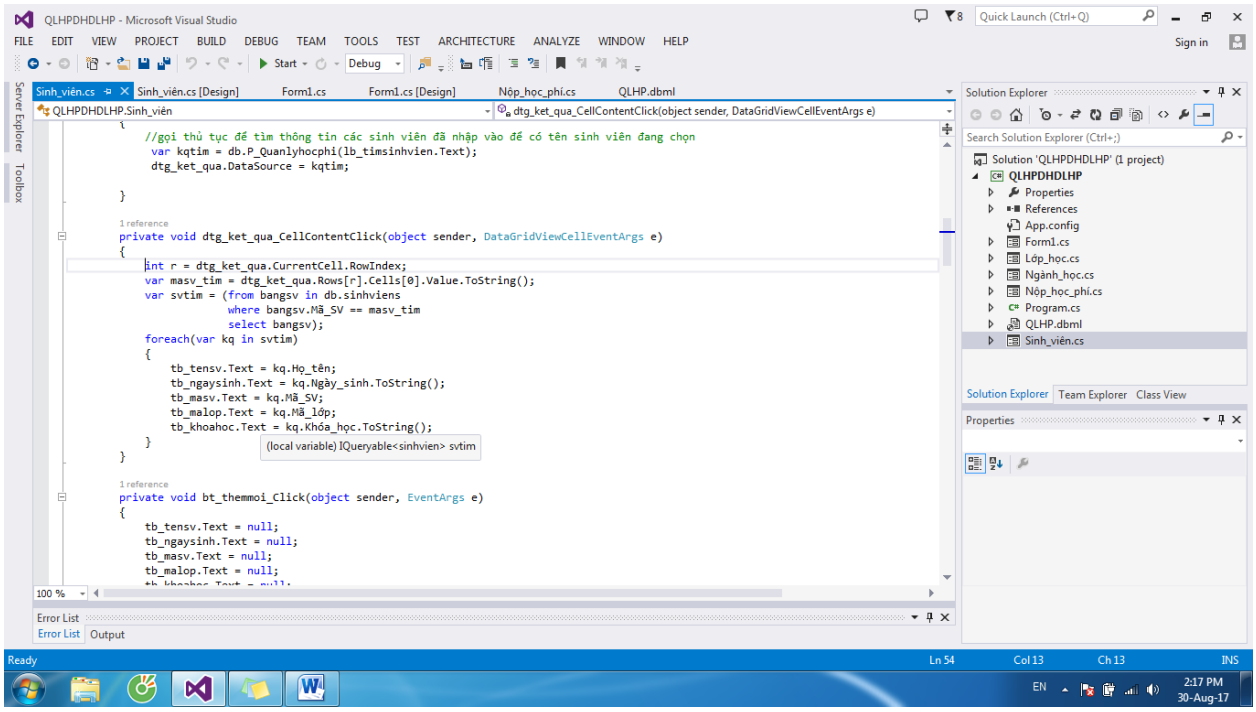
Hình 2.23 Gọi thủ tục Stored Prceduces bằng câu lệnh LinQ to SQL

❖ Câu lệnh cho Data Grid View của Form sinh viên.

Đoạn code như sau:

```
private void dtg_ket_qua_CellContentClick(object sender, DataGridViewCellEventArgs
e)
        {
            int r = dtg_ket_qua.CurrentCell.RowIndex;
            var masv_tim = dtg_ket_qua.Rows[r].Cells[0].Value.ToString();
            var svtim = (from bangsv in db.sinhviens
                         where bangsv.Mã_SV == masv_tim
                         select bangsv);
            foreach(var kq in svtim)
            {
                tb_tensv.Text = kq.Họ_tên;
                tb_ngaysinh.Text = kq.Ngày_sinh.ToString();
                tb_masv.Text = kq.Mã_SV;
                tb_malop.Text = kq.Mã_lớp;
                tb_khoahoc.Text = kq.Khóa_học.ToString();
            }
        }
```

+ Ở đây em sử dụng chức năng Select của LinQ to SQL



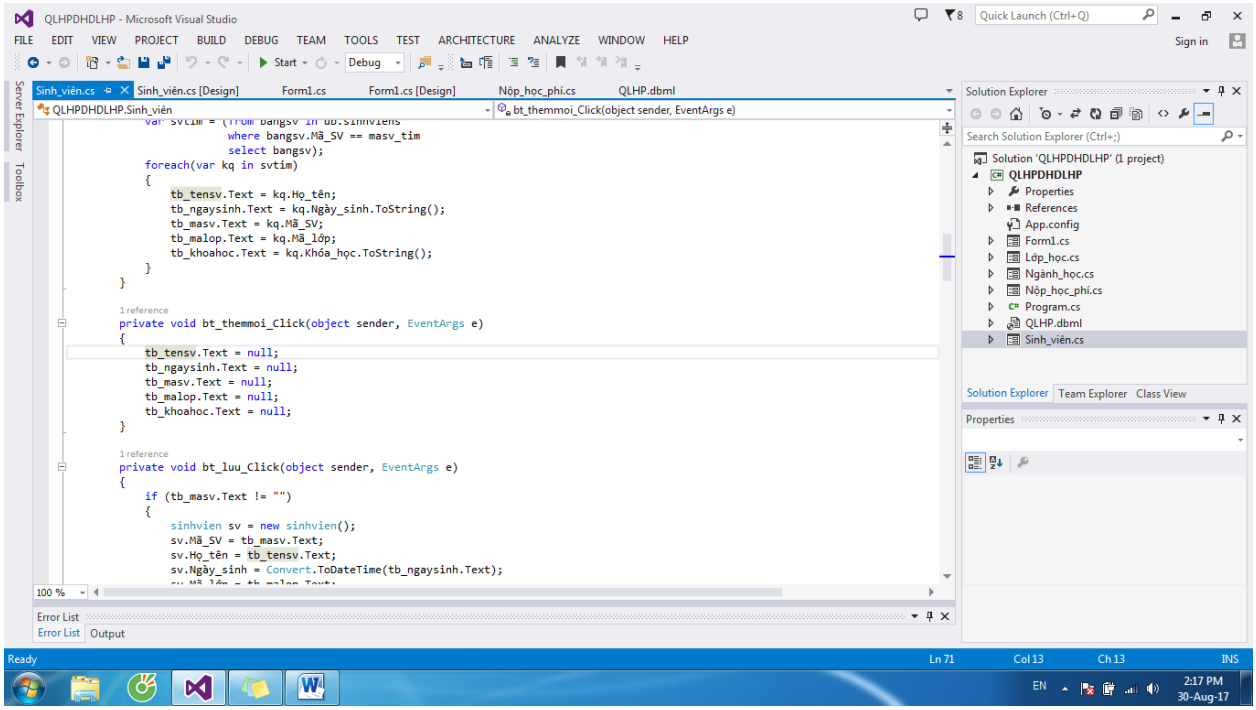
Hình 2.24 Code làm việc của bảng Data Grid View

❖ Câu lệnh cho nút Thêm mới

Đoạn code như sau:

```
private void bt_themmoi_Click(object sender, EventArgs e)
 {
   tb_tensv.Text = null;
   tb_ngaysinh.Text = null;
  tb_masv.Text = null;
    tb_malop.Text = null;
```

```
        tb_khoahoc.Text = null;
    }
```



Hình 2.25 Mã lệnh " Thêm mới "

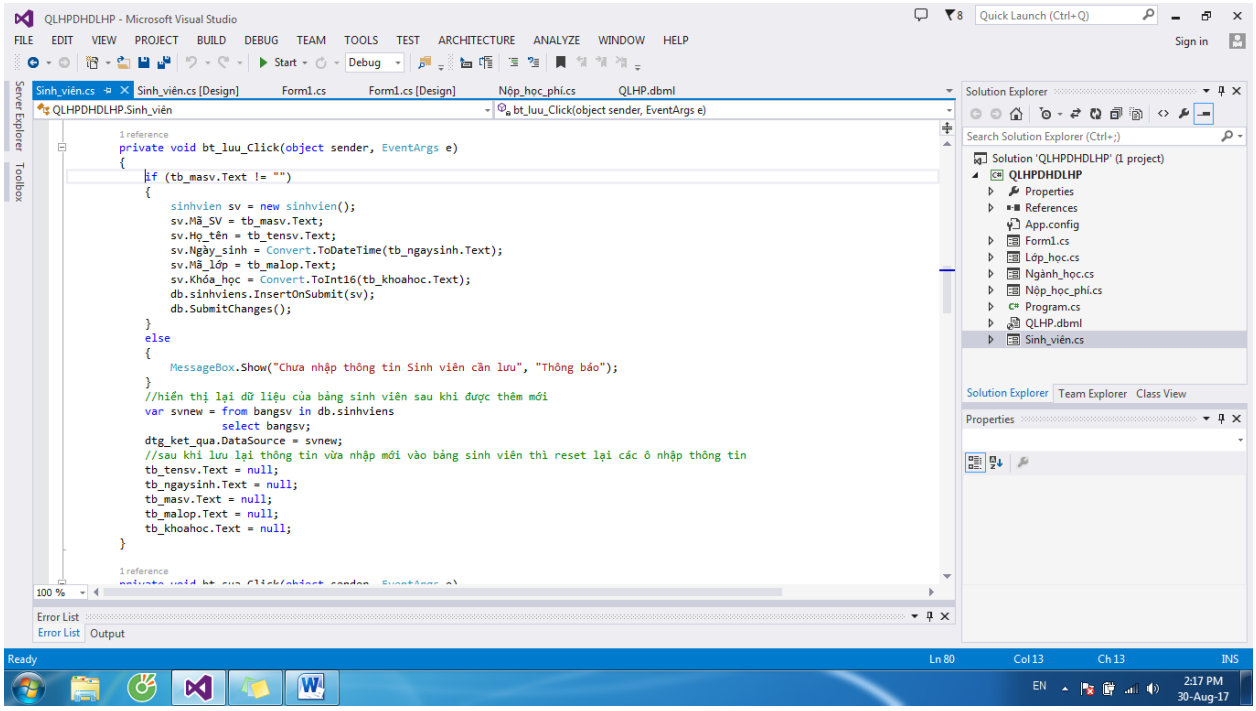
❖ Câu lệnh cho nút Lưu thêm mới

Đoạn code như sau:

```
private void bt_luu_Click(object sender, EventArgs e)
    {
        if (tb_masv.Text != "")
        {
            sinhvien sv = new sinhvien();
            sv.Mã_SV = tb_masv.Text;
            sv.Họ_tên = tb_tensv.Text;
            sv.Ngày_sinh = Convert.ToDateTime(tb_ngaysinh.Text);
            sv.Mã_lớp = tb_malop.Text;
            sv.Khóa_học = Convert.ToInt16(tb_khoahoc.Text)
            db.sinhviens.InsertOnSubmit(sv);
            db.SubmitChanges();
        }
        else
        {
            MessageBox.Show("Chưa nhập thông tin Sinh viên cần lưu", "Thông báo");
        }
        //hiển thị lại dữ liệu của bảng sinh viên sau khi được thêm mới
        var svnew = from bangsv in db.sinhviens
                    select bangsv;
        dtg_ket_qua.DataSource = svnew;
        //sau khi lưu lại thông tin vừa nhập mới vào bảng sinh viên thì reset lại các ô nhập
thông tin
```

```
        tb_tensv.Text = null;
        tb_ngaysinh.Text = null;
        tb_masv.Text = null;
        tb_malop.Text = null;
        tb_khoahoc.Text = null;
        }
```

+ Ở đây em sử dụng chức năng Insert của LinQ to SQL để đưa các thông tin thêm mới vào trong CSDL



Hình 2.26 Mã lệnh " Lưu thêm "

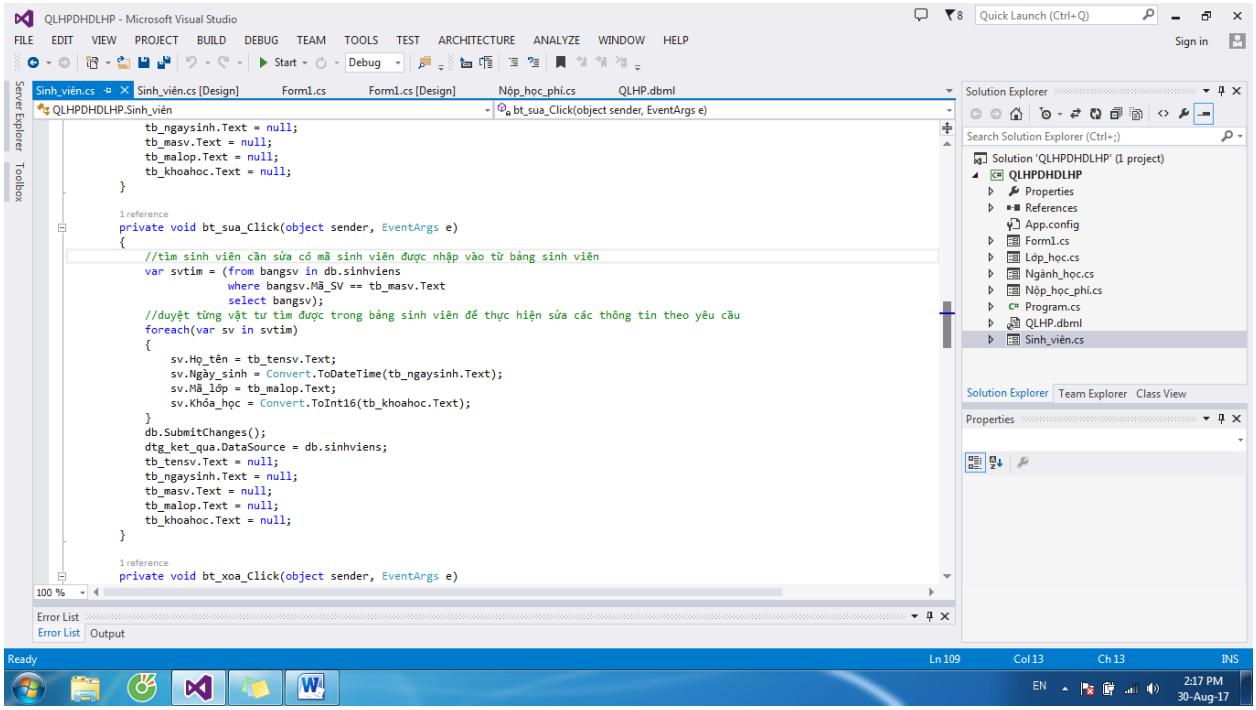
❖ Câu lệnh cho nút Sửa

Đoạn code như sau:

```
private void bt_sua_Click(object sender, EventArgs e)
    {
        //tìm sinh viên cần sửa có mã sinh viên được nhập vào từ bảng sinh viên
        var svtim = (from bangsv in db.sinhviens
                     where bangsv.Mã_SV == tb_masv.Text
                     select bangsv);
                //duyệt từng vật tư tìm được trong bảng sinh viên để thực hiện sửa các
    thông tin theo yêu cầu
        foreach(var sv in svtim)
        {
            sv.Họ_tên = tb_tensv.Text;
            sv.Ngày_sinh = Convert.ToDateTime(tb_ngaysinh.Text);
            sv.Mã_lớp = tb_malop.Text;
            sv.Khóa_học = Convert.ToInt16(tb_khoahoc.Text);
        }
        db.SubmitChanges();
        dtg_ket_qua.DataSource = db.sinhviens;
        tb_tensv.Text = null;
        tb_ngaysinh.Text = null;
```

```
            tb_masv.Text = null;
            tb_malop.Text = null;
            tb_khoahoc.Text = null;
            }
```

+ Ở đây em sử dụng chức năng Select và Update của LinQ to SQL



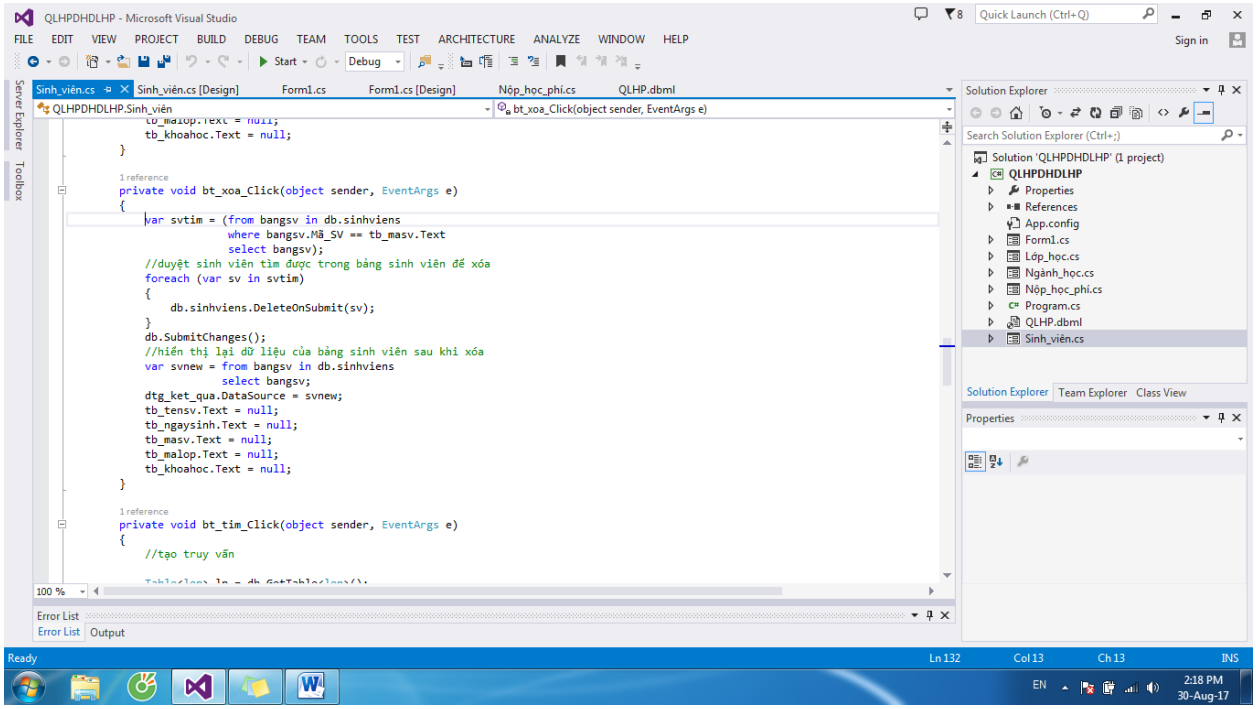
Hình 2.27 Mã lệnh " Sửa "

❖ Câu lệnh cho nút Xóa

Đoạn code như sau:

```
private void bt_xoa_Click(object sender, EventArgs e)
{
var svtim = (from bangsv in db.sinhviens
            where bangsv.Mã_SV == tb_masv.Text
            select bangsv);                    //Chức năng SELECT
//duyệt sinh viên tìm được trong bảng sinh viên để xóa
foreach (var sv in svtim)
{
   db.sinhviens.DeleteOnSubmit(sv);     // Chức năng DELETE
}
db.SubmitChanges();
//hiển thị lại dữ liệu của bảng sinh viên sau khi xóa
var svnew = from bangsv in db.sinhviens
          select bangsv;
dtg_ket_qua.DataSource = svnew;
tb_tensv.Text = null;
tb_ngaysinh.Text = null;
tb_masv.Text = null;
```

```
    tb_malop.Text = null;
    tb_khoahoc.Text = null;
        }
```



2.28 Mã lệnh " Xóa "

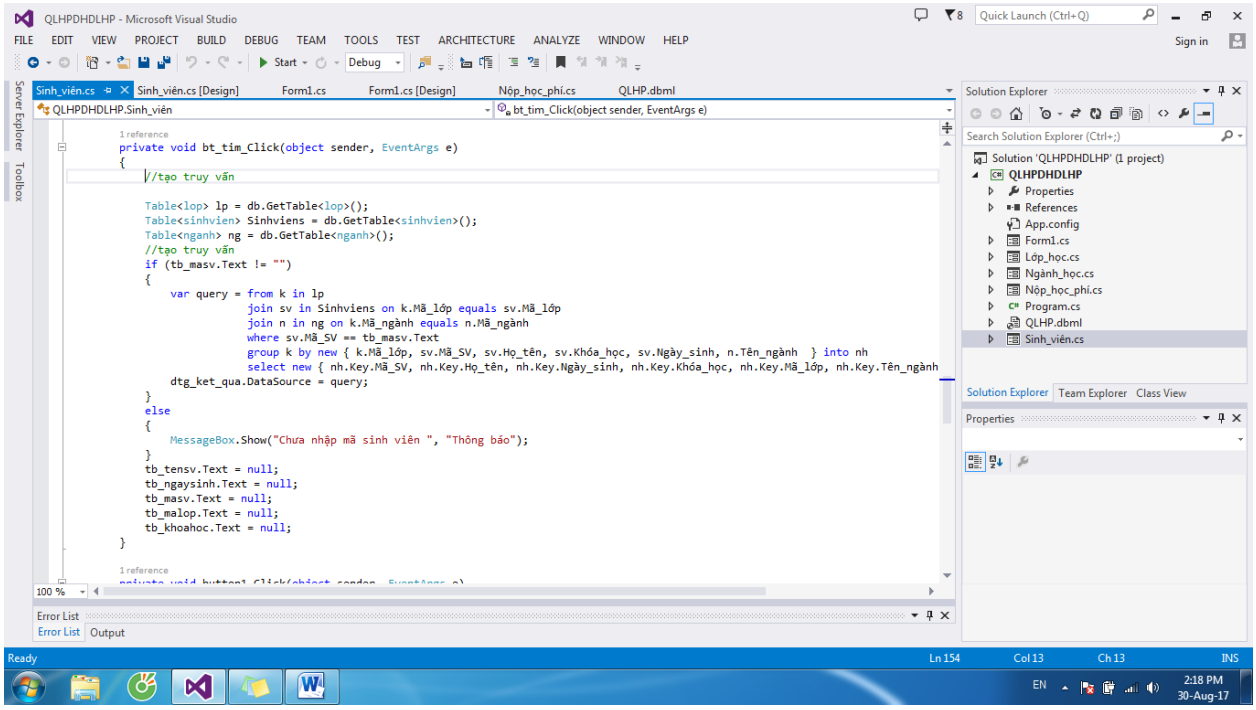
❖ Câu lệnh cho nút Tìm kiếm sinh viên theo mã sinh viên

+ Đoạn code như sau:

```
private void bt_tim_Click(object sender, EventArgs e)
{
//tạo truy vấn
Table<lop> lp = db.GetTable<lop>();
Table<sinhvien> Sinhviens = db.GetTable<sinhvien>();
Table<nganh> ng = db.GetTable<nganh>();
if (tb_masv.Text != "")
{
   var query = from k in lp
               join sv in Sinhviens on k.Mã_lớp equals sv.Mã_lớp
               join n in ng on k.Mã_ngành equals n.Mã_ngành
               where sv.Mã_SV == tb_masv.Text
                     group k by new { k.Mã_lớp, sv.Mã_SV, sv.Họ_tên, sv.Khóa_học,
sv.Ngày_sinh, n.Tên_ngành  } into nh
                           select new { nh.Key.Mã_SV, nh.Key.Họ_tên, nh.Key.Ngày_sinh,
nh.Key.Khóa_học, nh.Key.Mã_lớp, nh.Key.Tên_ngành };
   dtg_ket_qua.DataSource = query;
}
else
{
   MessageBox.Show("Chưa nhập mã sinh viên ", "Thông báo");
```

```
        }
        tb_tensv.Text = null;
        tb_ngaysinh.Text = null;
        tb_masv.Text = null;
        tb_malop.Text = null;
        tb_khoahoc.Text = null;
            }
```

+ Ở đây em sử dụng chức năng Join(liên kết giữa các bảng) của LinQ to SQL để lấy ra các thông tin cần tìm kiếm.



Hình 2.29 Mã lệnh "Tìm kiếm sinh viên theo mã sinh viên"

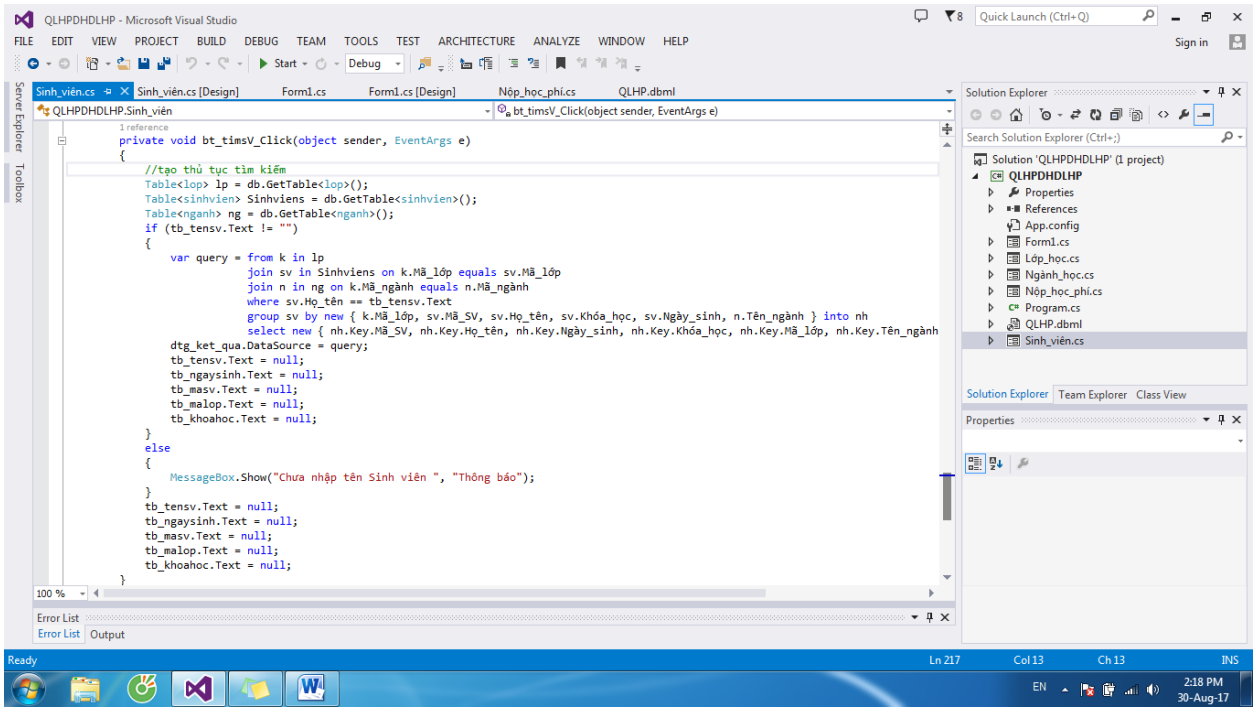
❖ Câu lệnh cho nút Tìm kiếm sinh viên theo tên sinh viên

+ Đoạn code như sau:

```
private void bt_timsV_Click(object sender, EventArgs e)
    {
        //tạo thủ tục tìm kiếm
        Table<lop> lp = db.GetTable<lop>();
        Table<sinhvien> Sinhviens = db.GetTable<sinhvien>();
        Table<nganh> ng = db.GetTable<nganh>();
        if (tb_tensv.Text != "")
        {
            var query = from k in lp
                        join sv in Sinhviens on k.Mã_lớp equals sv.Mã_lớp
                        join n in ng on k.Mã_ngành equals n.Mã_ngành
                        where sv.Họ_tên == tb_tensv.Text
                              group sv by new { k.Mã_lớp, sv.Mã_SV, sv.Họ_tên,
    sv.Khóa_học, sv.Ngày_sinh, n.Tên_ngành } into nh
                        select new { nh.Key.Mã_SV, nh.Key.Họ_tên, nh.Key.Ngày_sinh,
    nh.Key.Khóa_học, nh.Key.Mã_lớp, nh.Key.Tên_ngành };
```

```
            dtg_ket_qua.DataSource = query;
            tb_tensv.Text = null;
            tb_ngaysinh.Text = null;
            tb_masv.Text = null;
            tb_malop.Text = null;
            tb_khoahoc.Text = null;
        }
        else
        {
            MessageBox.Show("Chưa nhập tên Sinh viên ", "Thông báo");
        }
        tb_tensv.Text = null;
        tb_ngaysinh.Text = null;
        tb_masv.Text = null;
        tb_malop.Text = null;
        tb_khoahoc.Text = null;
           }
```

+ Ở đây em sử dụng chức năng Join(liên kết giữa các bảng) của LinQ to SQL để lấy ra các thông tin cần tìm kiếm.



Hình 2.30 Mã lệnh "Tìm kiếm sinh viên theo tên sinh viên"

❖ Câu lệnh cho nút Thống kê các sinh viên đã nộp học phí theo lớp

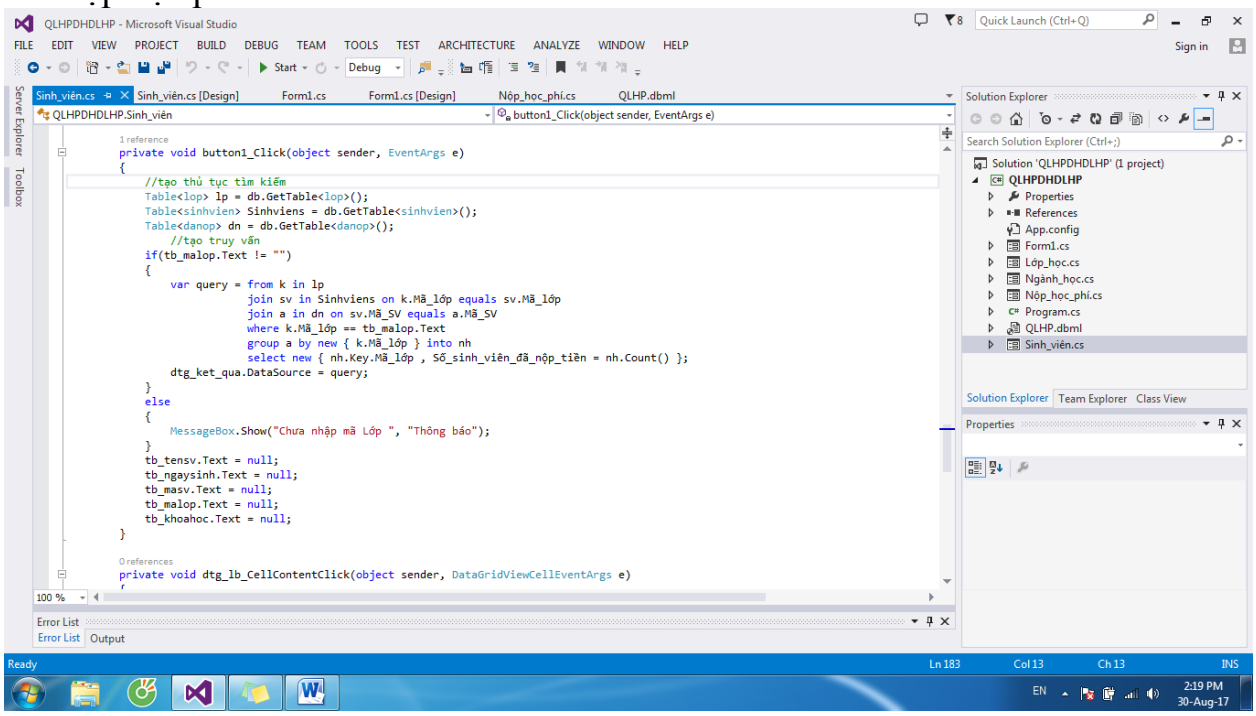
+ Đoạn code như sau:

```
private void button1_Click(object sender, EventArgs e)
    {
        //tạo thủ tục tìm kiếm
        Table<lop> lp = db.GetTable<lop>();
        Table<sinhvien> Sinhviens = db.GetTable<sinhvien>();
        Table<danop> dn = db.GetTable<danop>();
        if(tb_malop.Text != "")
```

```
        {
            var query = from k in lp
                        join sv in Sinhviens on k.Mã_lớp equals sv.Mã_lớp
                        join a in dn on sv.Mã_SV equals a.Mã_SV
                        where k.Mã_lớp == tb_malop.Text
                        group a by new { k.Mã_lớp } into nh
                               select new { nh.Key.Mã_lớp , Số_sinh_viên_đã_nộp_tiền =
nh.Count() };
            dtg_ket_qua.DataSource = query;
        }
        else
        {
            MessageBox.Show("Chưa nhập mã Lớp ", "Thông báo");
        }
        tb_tensv.Text = null;
        tb_ngaysinh.Text = null;
        tb_masv.Text = null;
        tb_malop.Text = null;
        tb_khoahoc.Text = null;
           }
```

+ Ở đây em sử dụng chức năng Join(liên kết giữa các bảng) của LinQ to SQL để lấy ra các thông tin cần tìm kiếm.

+ Dùng câu lệnh Count() của LinQ to SQL để thống kê ra số trong 1 lớp đã nộp học phí

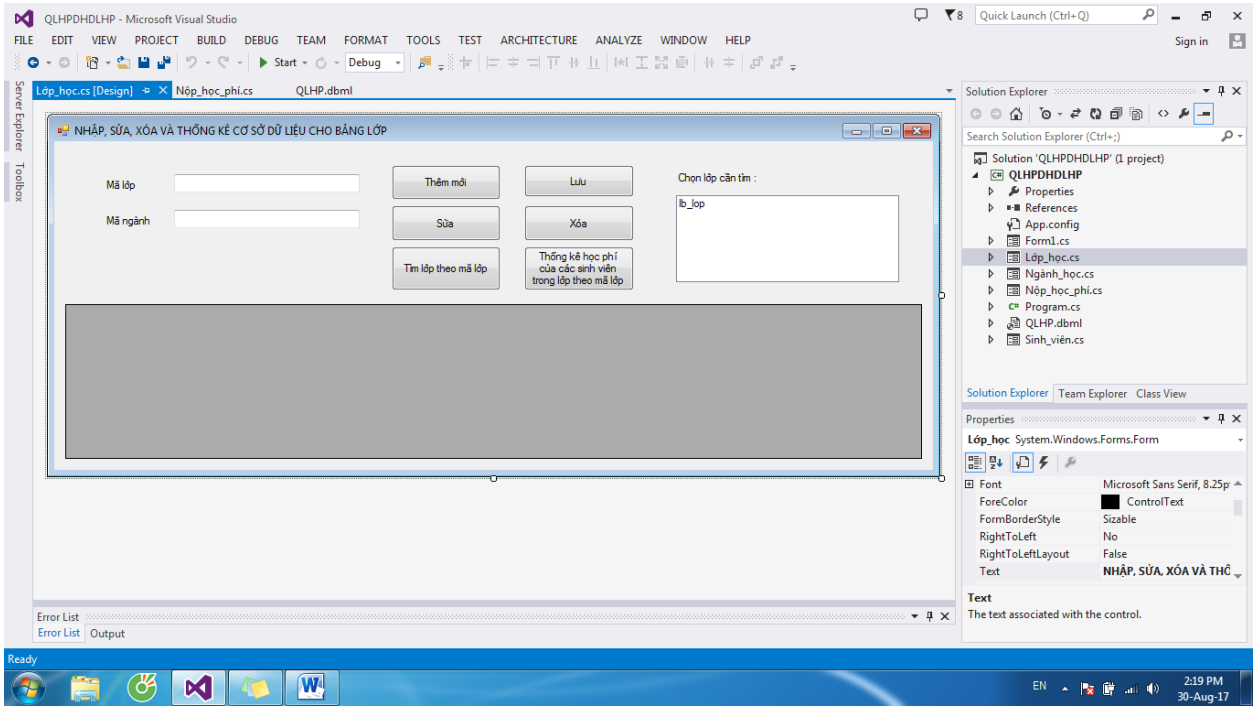


Hình 2.31 Mã lệnh "Thống kê các sinh viên đã nộp học phí theo lớp"

❖ Giao diện Form Lớp học

+ Form Lớp bao gồm các nút Thêm mới, Lưu, Xóa, Sửa, Tìm kiếm theo Mã lớp va thống kê học phí của các sinh viên trong 1 lớp theo mã lớp

+ Ở đây em có tạo thêm 1 List Box để hỗ trợ tìm kiếm nhanh tên của các lớp.



Hình 2.32 Giao diện Form Lớp học

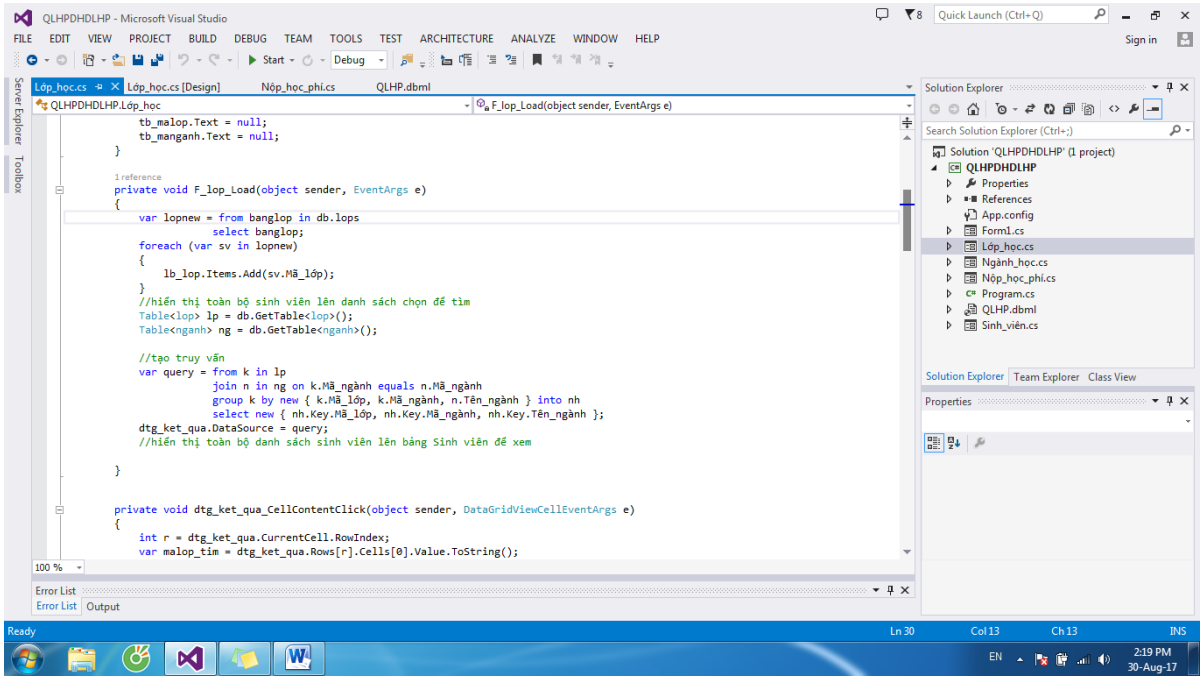
❖ Tạo truy vấn cho Form Lớp

+ Đoạn code như sau:

```
private void F_lop_Load(object sender, EventArgs e)
      {
         var lopnew = from banglop in db.lops
                      select banglop;
         foreach (var sv in lopnew)
         {
            lb_lop.Items.Add(sv.Mã_lớp);
         }
         // hiển thị toàn bộ sinh viên lên danh sách chọn để tìm
         Table<lop> lp = db.GetTable<lop>();
         Table<nganh> ng = db.GetTable<nganh>();

         // tạo truy vấn
         var query = from k in lp
                     join n in ng on k.Mã_ngành equals n.Mã_ngành
                     group k by new { k.Mã_lớp, k.Mã_ngành, n.Tên_ngành } into nh
                     select new { nh.Key.Mã_lớp, nh.Key.Mã_ngành, nh.Key.Tên_ngành };
         dtg_ket_qua.DataSource = query;
         // hiển thị toàn bộ danh sách sinh viên lên bảng Sinh viên để xem
      }
```

+ Ở đây em sử dụng chức năng Select(chọn lọc) và Join(liên kết giữa các bảng) của LinQ to SQL để lấy ra các thông tin cần tìm kiếm.

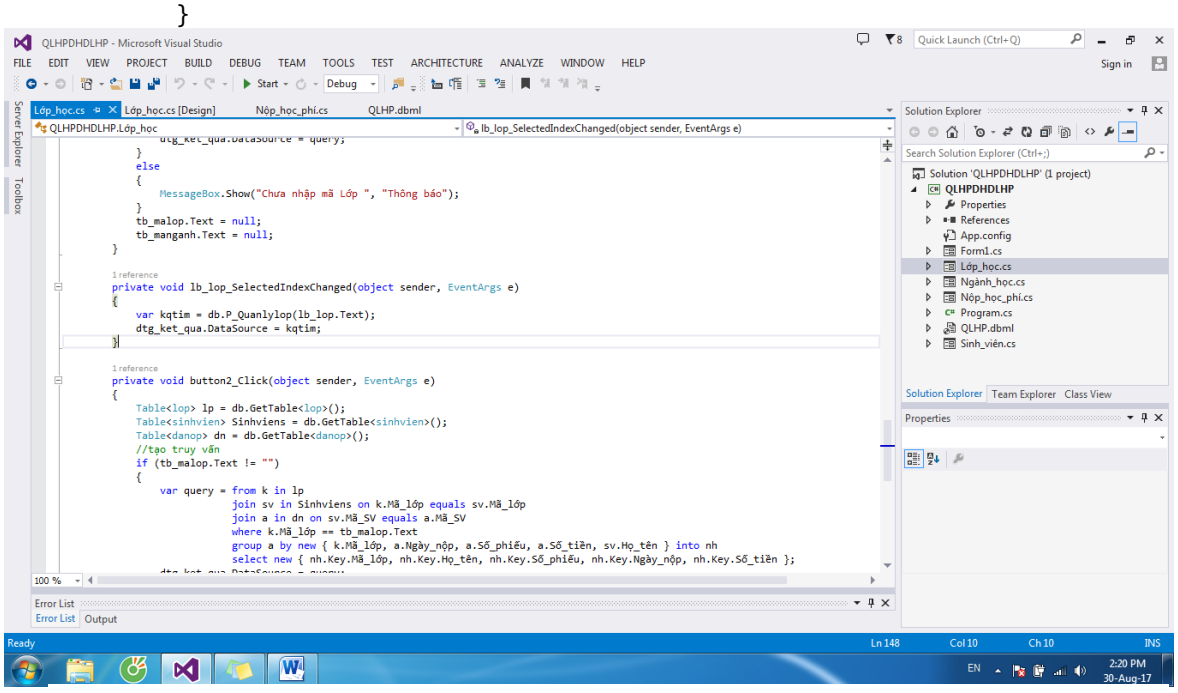


Hình 2.33 Tạo truy vấn cho Form lớp

❖ Gọi thủ tục Stored Prceduces bằng câu lệnh LinQ to SQL

+ Đoạn code như sau:

```
private void lb_lop_SelectedIndexChanged(object sender, EventArgs e)
        {
            var kqtim = db.P_Quanlylop(lb_lop.Text);
            dtg_ket_qua.DataSource = kqtim;
        }
```

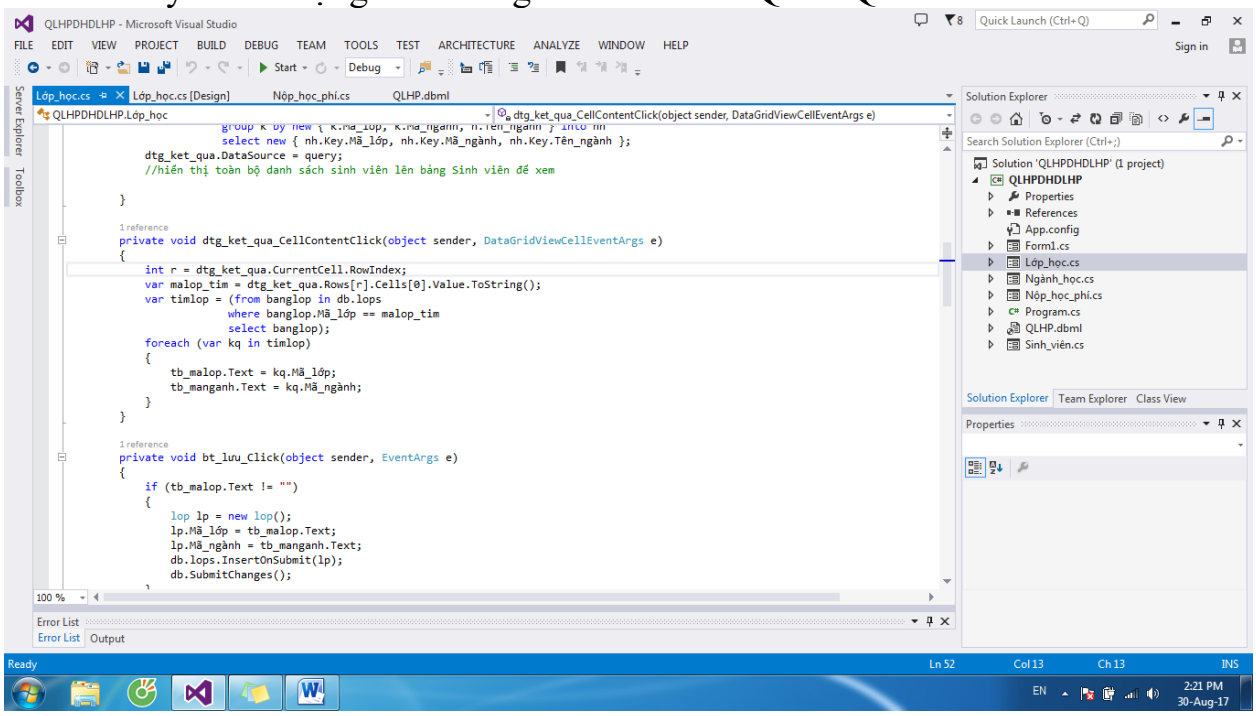


Hình 2.34 Gọi thủ tục Stored Prceduces bằng câu lệnh LinQ to SQL

+ Đoạn code như sau:

```
private void dtg_ket_qua_CellContentClick(object sender, DataGridViewCellEventArgs e)
 {
    int r = dtg_ket_qua.CurrentCell.RowIndex;
    var malop_tim = dtg_ket_qua.Rows[r].Cells[0].Value.ToString();
    var timlop = (from banglop in db.lops
                  where banglop.Mã_lớp == malop_tim
                  select banglop);
    foreach (var kq in timlop)
    {
       tb_malop.Text = kq.Mã_lớp;
       tb_manganh.Text = kq.Mã_ngành;
    }
       }
```

+ Ở đây em sử dụng chức năng Select của LinQ to SQL



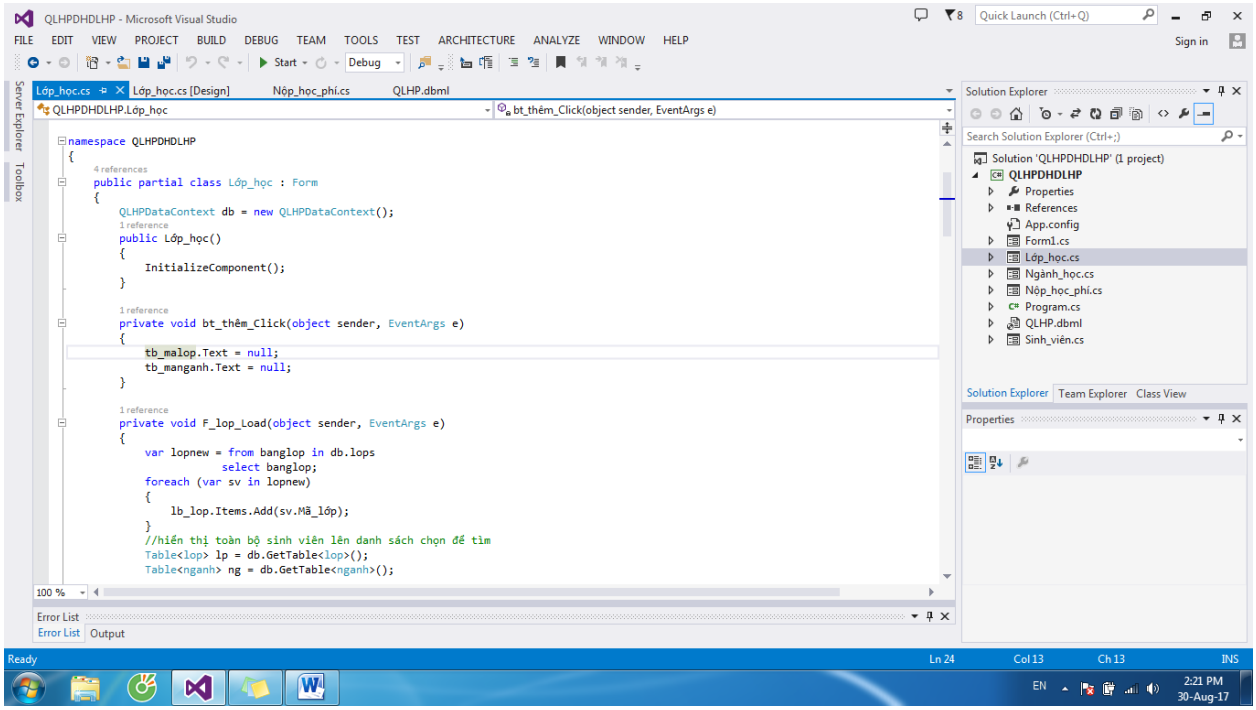
Hình 2.35 Code làm việc của bảng Data Grid View

❖ Câu lệnh cho nút Thêm mới

+ Đoạn code như sau:

```
private void bt_thêm_Click(object sender, EventArgs e)
   {
      tb_malop.Text = null;
      tb_manganh.Text = null;
```

}



Hình 2.36 Mã lệnh " Thêm mới "

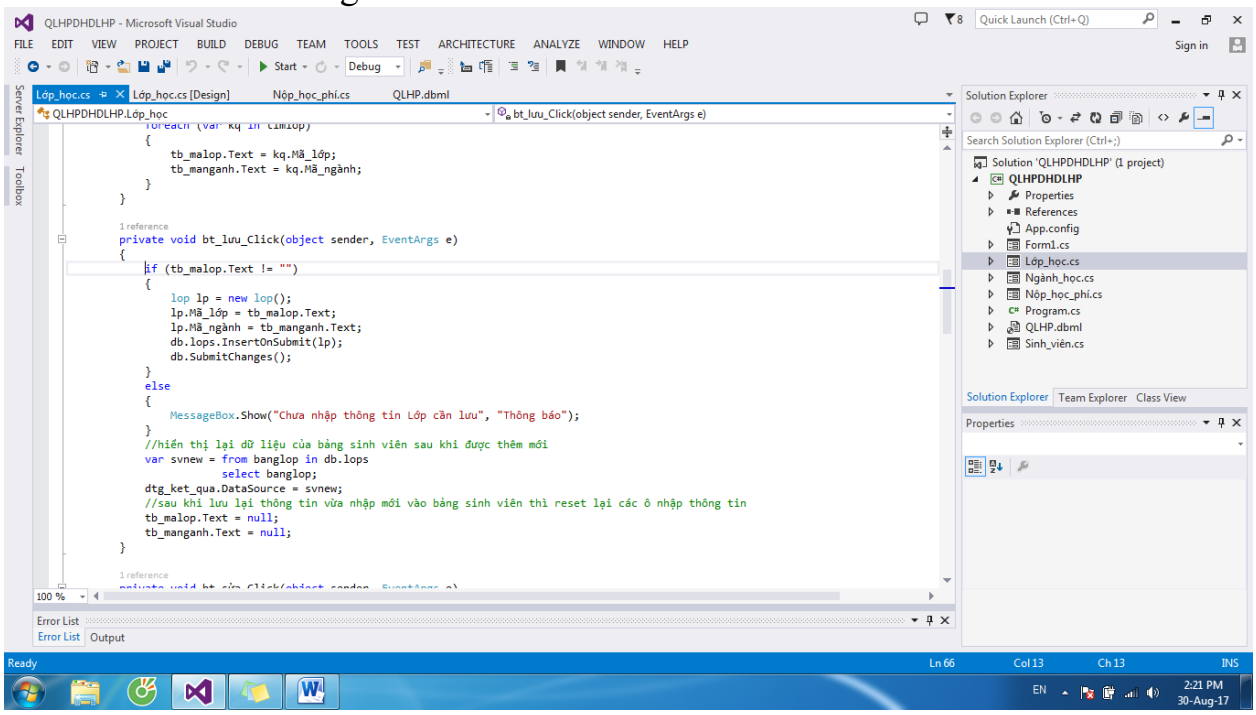
❖ Câu lệnh cho nút Lưu thêm mới

Đoạn code như sau:

```
private void bt_lưu_Click(object sender, EventArgs e)
    {
        if (tb_malop.Text != "")
 {
        lop lp = new lop();
        lp.Mã_lớp = tb_malop.Text;
                lp.Mã_ngành = tb_manganh.Text;
                db.lops.InsertOnSubmit(lp);
                db.SubmitChanges();
}
else
{
   MessageBox.Show("Chưa nhập thông tin Lớp cần lưu", "Thông báo");
}
//hiển thị lại dữ liệu của bảng sinh viên sau khi được thêm mới
var svnew = from banglop in db.lops
          select banglop;
dtg_ket_qua.DataSource = svnew;
//sau khi lưu lại thông tin vừa nhập mới vào bảng sinh viên thì reset lại các ô nhập thông
tin
tb_malop.Text = null;
tb_manganh.Text = null;
```

}

+ Ở đây em sử dụng chức năng Insert của LinQ to SQL để đưa các thông tin thêm mới vào trong CSDL



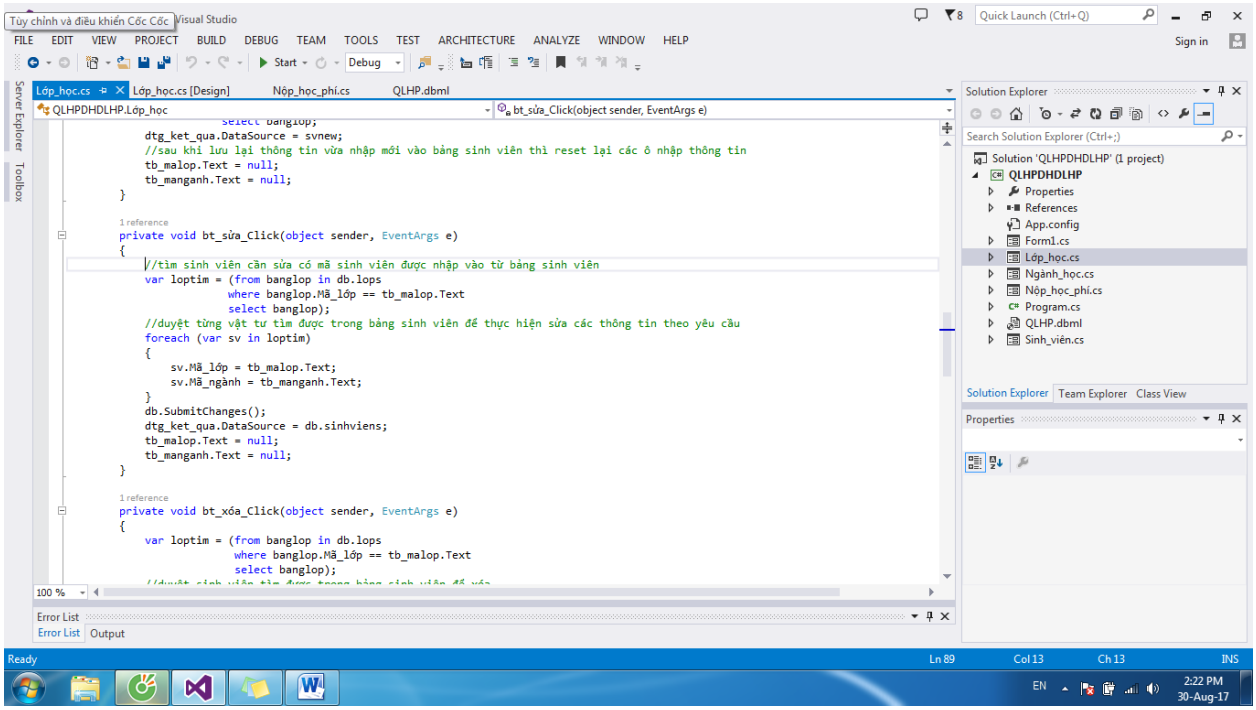
Hình 2.37 Mã lệnh " Lưu thêm "

❖ Câu lệnh cho nút Sửa

Đoạn code như sau:

```
private void bt_sửa_Click(object sender, EventArgs e)
{
//tìm sinh viên cần sửa có mã sinh viên được nhập vào từ bảng sinh viên
var loptim = (from banglop in db.lops
              where banglop.Mã_lớp == tb_malop.Text
              select banglop);
//duyệt từng vật tư tìm được trong bảng sinh viên để thực hiện sửa các thông tin theo yêu cầu
foreach (var sv in loptim)
{
   sv.Mã_lớp = tb_malop.Text;
   sv.Mã_ngành = tb_manganh.Text;
}
db.SubmitChanges();
dtg_ket_qua.DataSource = db.sinhviens;
tb_malop.Text = null;
tb_manganh.Text = null;
   }
```

+ Ở đây em sử dụng chức năng Update của LinQ to SQL để đưa các thông tin đã sửa vào trong CSDL



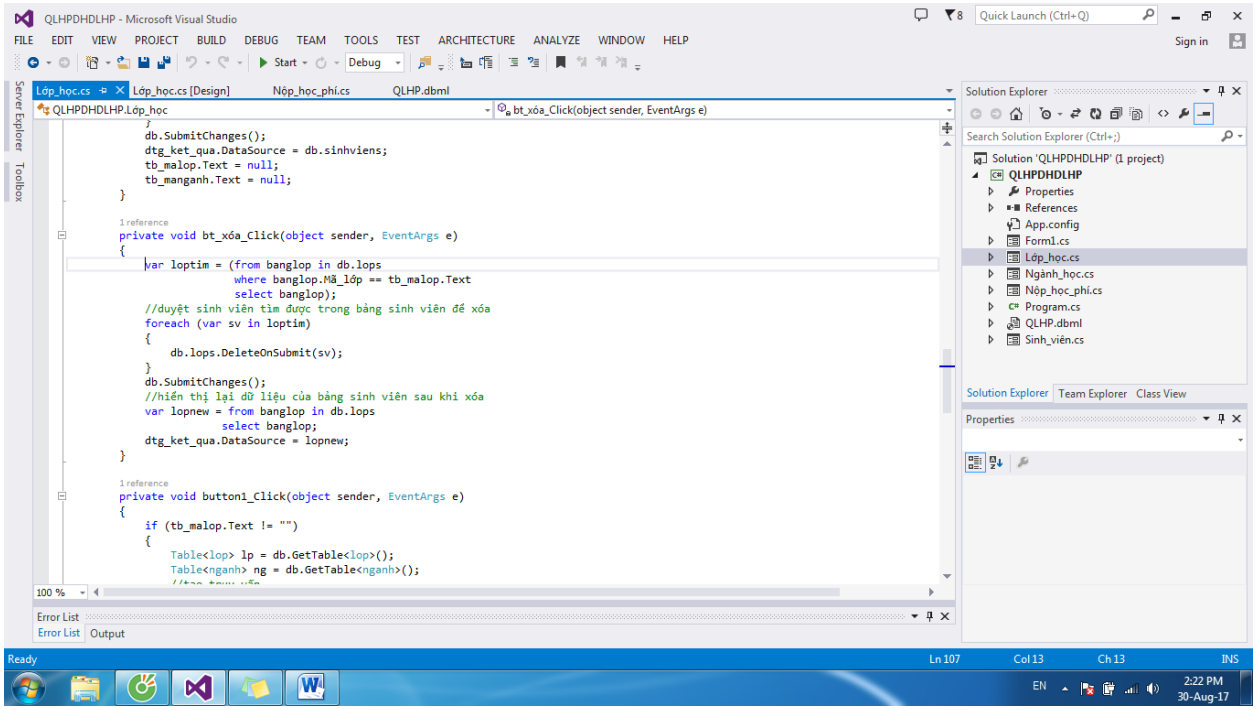
Hình 2.38 Mã lệnh " Sửa "

❖ Câu lệnh cho nút Xóa

Đoạn code như sau:

```
private void bt_xóa_Click(object sender, EventArgs e)
{
   var loptim = (from banglop in db.lops
                 where banglop.Mã_lớp == tb_malop.Text
                 select banglop);
   //duyệt sinh viên tìm được trong bảng sinh viên để xóa
   foreach (var sv in loptim)
   {
      db.lops.DeleteOnSubmit(sv);
   }
   db.SubmitChanges();
   //hiển thị lại dữ liệu của bảng sinh viên sau khi xóa
   var lopnew = from banglop in db.lops
                select banglop;
   dtg_ket_qua.DataSource = lopnew;
     }
```

+ Ở đây em sử dụng chức năng DELETE của LinQ to SQL để xóa các thông tin thêm mới vào trong CSDL



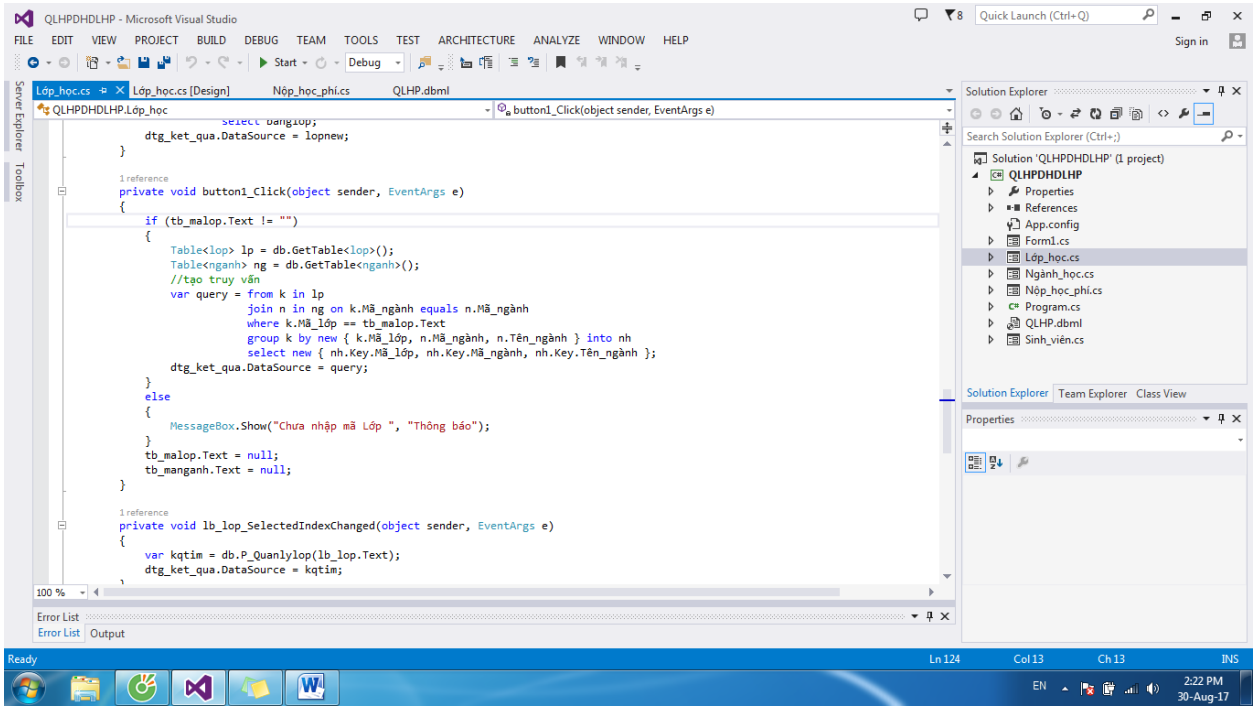
Hình 2.39 Mã lệnh " Xóa "

❖ Câu lệnh cho nút Tìm kiếm lớp theo mã lớp

+ Đoạn code như sau:

```
private void button1_Click(object sender, EventArgs e)
{
if (tb_malop.Text != "")
{
   Table<lop> lp = db.GetTable<lop>();
   Table<nganh> ng = db.GetTable<nganh>();
   //tạo truy vấn
   var query = from k in lp
               join n in ng on k.Mã_ngành equals n.Mã_ngành
               where k.Mã_lớp == tb_malop.Text
               group k by new { k.Mã_lớp, n.Mã_ngành, n.Tên_ngành } into nh
                   select new { nh.Key.Mã_lớp, nh.Key.Mã_ngành, nh.Key.Tên_ngành };
   dtg_ket_qua.DataSource = query;
}
else
{
   MessageBox.Show("Chưa nhập mã Lớp ", "Thông báo");
}
tb_malop.Text = null;
tb_manganh.Text = null;
   }
```

+ Ở đây em sử dụng chức năng Join(liên kết giữa các bảng) của LinQ to SQL để lấy ra các thông tin cần tìm kiếm.



Hình 2.40 Mã lệnh "Tìm kiếm lớp theo mã lớp"

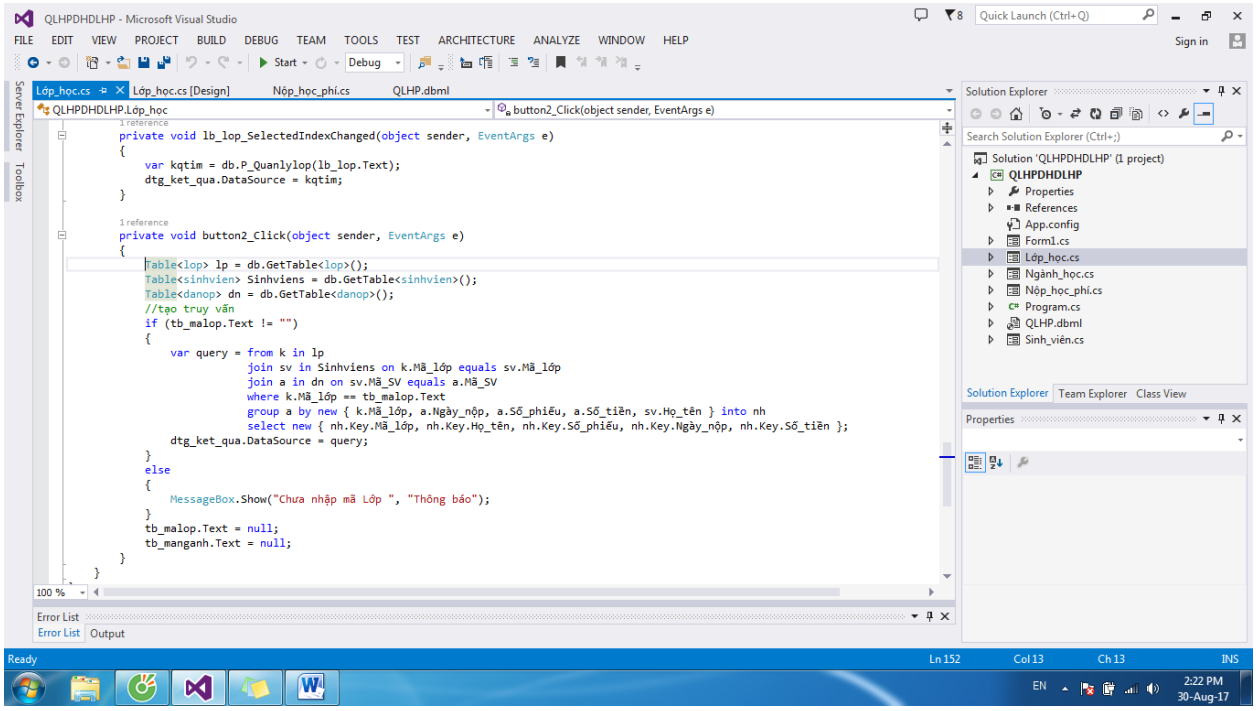
❖ Câu lệnh cho nút Thống kê học phí của các sinh viên trong lớp theo mã lớp

+ Đoạn code như sau:

```
private void button2_Click(object sender, EventArgs e)
{
   Table<lop> lp = db.GetTable<lop>();
   Table<sinhvien> Sinhviens = db.GetTable<sinhvien>();
   Table<danop> dn = db.GetTable<danop>();
   //tạo truy vấn
   if (tb_malop.Text != "")
   {
      var query = from k in lp
                  join sv in Sinhviens on k.Mã_lớp equals sv.Mã_lớp
                  join a in dn on sv.Mã_SV equals a.Mã_SV
                  where k.Mã_lớp == tb_malop.Text
                        group a by new { k.Mã_lớp, a.Ngày_nộp, a.Số_phiếu, a.Số_tiền,
   sv.Họ_tên } into nh
                              select new { nh.Key.Mã_lớp, nh.Key.Họ_tên, nh.Key.Số_phiếu,
   nh.Key.Ngày_nộp, nh.Key.Số_tiền };
      dtg_ket_qua.DataSource = query;
   }
   else
   {
      MessageBox.Show("Chưa nhập mã Lớp ", "Thông báo");
   }
   tb_malop.Text = null;
   tb_manganh.Text = null;
```

```
}
```

+ Ở đây em sử dụng chức năng Join(liên kết giữa các bảng) của LinQ to SQL để lấy ra các thông tin cần tìm kiếm.

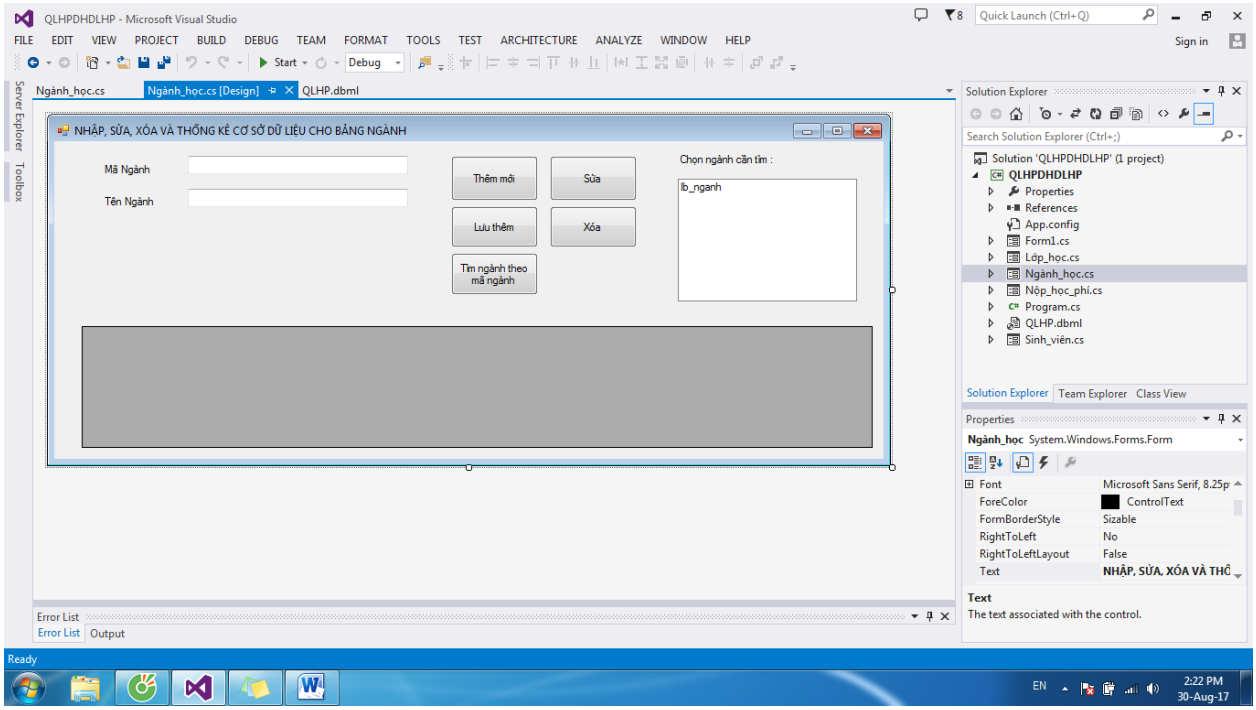


Hình 2.41 Mã lệnh "Thống kê học phí của các sinh viên trong lớp theo mã lớp"

❖ Giao diện Form Ngành học

+ Form Lớp bao gồm các nút chức năng Thêm, Sửa, Xóa, Lưu, tìm kiếm và thống kê dữ liệu của bảng NGÀNH trong cơ sở dữ liệu HOCPHI.

+ Ở đây em có tạo thêm 1 List Box để hỗ trợ tìm kiếm nhanh mã của các Ngành.

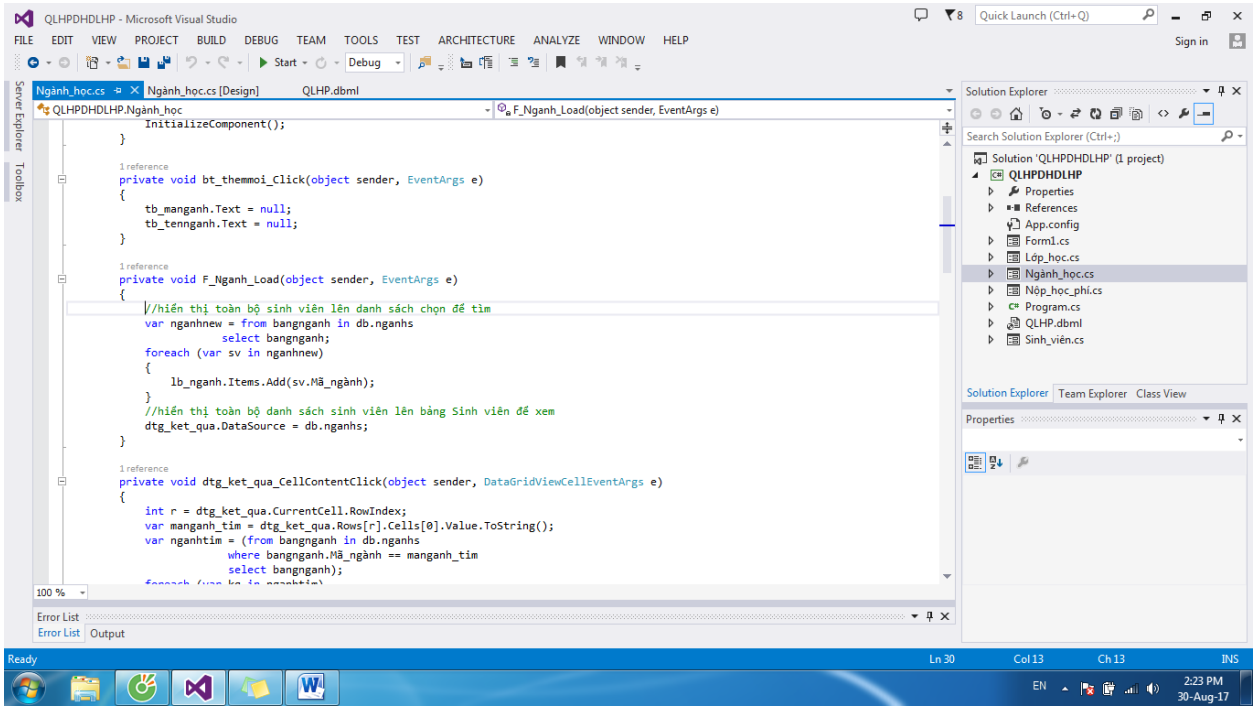


Hình 2.42 Giao diện Form Ngành học

❖ Tạo truy vấn cho Form Ngành

+ Đoạn code như sau:

```
private void F_Nganh_Load(object sender, EventArgs e)
{
//hiển thị toàn bộ sinh viên lên danh sách chọn để tìm
var nganhnew = from bangnganh in db.nganhs
        select bangnganh;
foreach (var sv in nganhnew)
{
   lb_nganh.Items.Add(sv.Mã_ngành);
}
//hiển thị toàn bộ danh sách sinh viên lên bảng Sinh viên để xem
dtg_ket_qua.DataSource = db.nganhs;
}
```

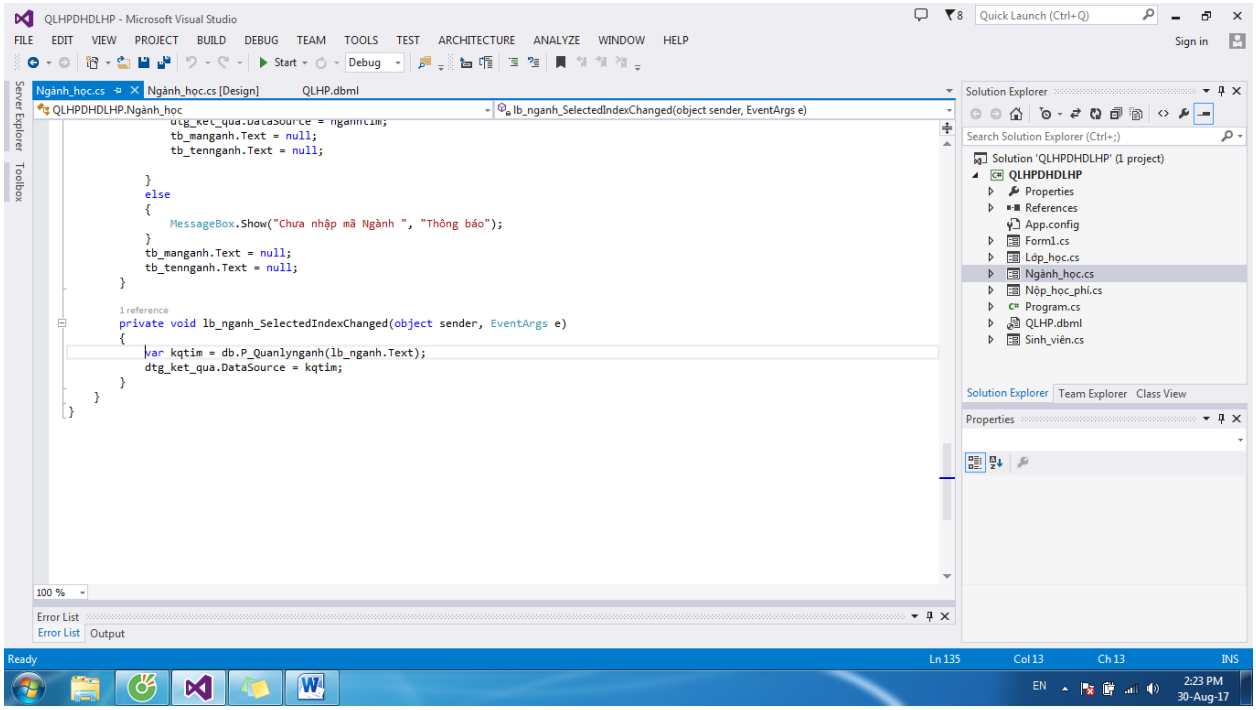


Hình 2.43 Tạo truy vấn cho Form Ngành

❖ Gọi thủ tục Stored Prceduces bằng câu lệnh LinQ to SQL

+ Đoạn code như sau:

```
private void lb_nganh_SelectedIndexChanged(object sender, EventArgs e)
    {
        var kqtim = db.P_Quanlynganh(lb_nganh.Text);
        dtg_ket_qua.DataSource = kqtim;
     }
```

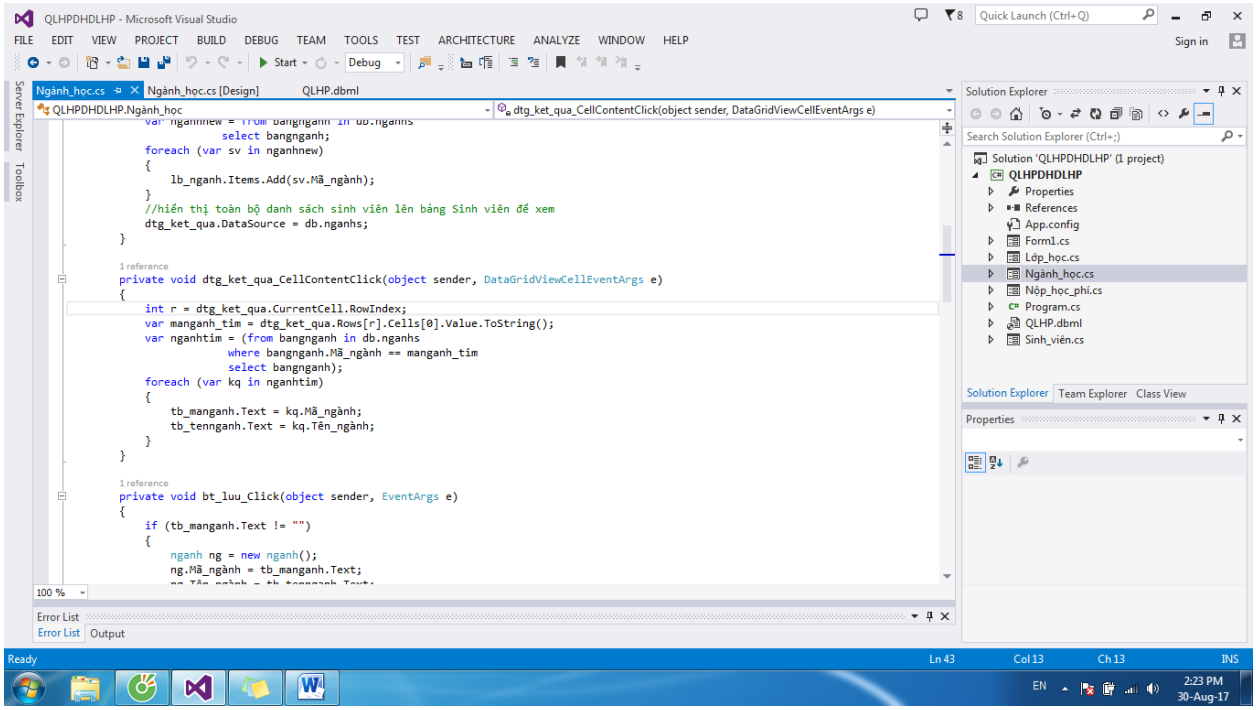


Hình 2.44 Gọi thủ tục Stored Prceduces bằng câu lệnh LinQ to SQL

❖ Câu lệnh cho Data Grid View của Form Lớp

+ Đoạn code như sau:

```
private void dtg_ket_qua_CellContentClick(object sender, DataGridViewCellEventArgs
e)
    {
      int r = dtg_ket_qua.CurrentCell.RowIndex;
      var manganh_tim = dtg_ket_qua.Rows[r].Cells[0].Value.ToString();
      var nganhtim = (from bangnganh in db.nganhs
                  where bangnganh.Mã_ngành == manganh_tim
                  select bangnganh);
      foreach (var kq in nganhtim)
      {
         tb_manganh.Text = kq.Mã_ngành;
         tb_tennganh.Text = kq.Tên_ngành;
          }
    }
```

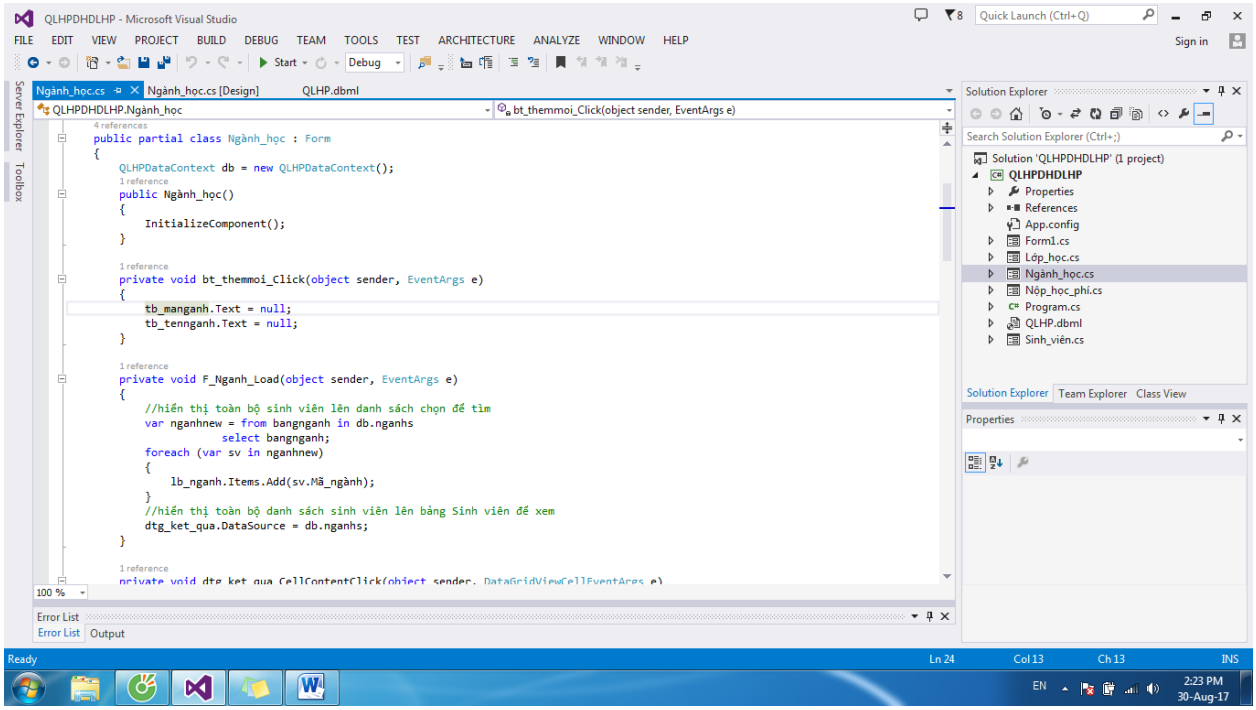


Hình 2.45 Code làm việc của bảng Data Grid View

❖ Câu lệnh cho nút Thêm mới

+ Đoạn code như sau:

```
private void bt_themmoi_Click(object sender, EventArgs e)
      {
         tb_manganh.Text = null;
         tb_tennganh.Text = null;
      }
```



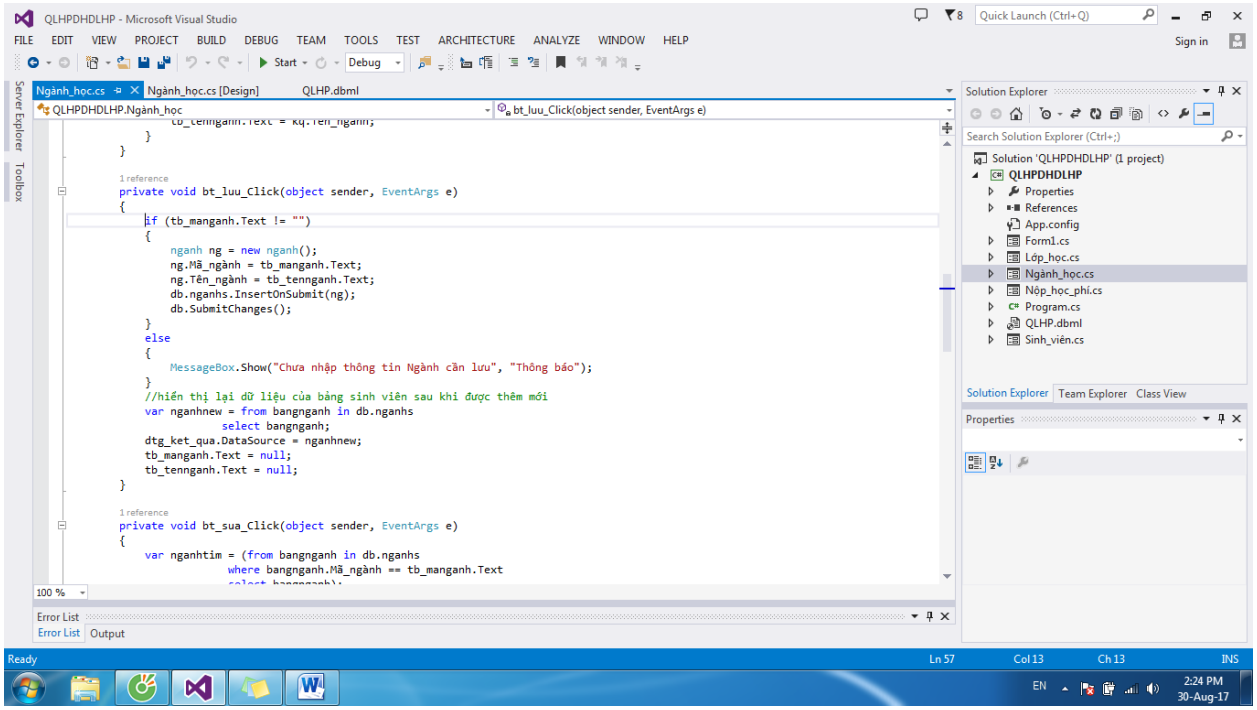
Hình 2.46 Mã lệnh " Thêm mới "

❖ Câu lệnh cho nút Lưu thêm mới

+ Đoạn code như sau:

```
private void bt_luu_Click(object sender, EventArgs e)
   {
      if (tb_manganh.Text != "")
       {
           nganh ng = new nganh();
           ng.Mã_ngành = tb_manganh.Text;
           ng.Tên_ngành = tb_tennganh.Text;
           db.nganhs.InsertOnSubmit(ng);
           db.SubmitChanges();
        }
        else
        {
           MessageBox.Show("Chưa nhập thông tin Ngành cần lưu", "Thông báo");
        }
        //hiển thị lại dữ liệu của bảng sinh viên sau khi được thêm mới
        var nganhnew = from bangnganh in db.nganhs
                  select bangnganh;
        dtg_ket_qua.DataSource = nganhnew;
        tb_manganh.Text = null;
        tb_tennganh.Text = null;
        }
```

+ Ở đây em sử dụng chức năng Insert của LinQ to SQL để đưa các thông tin thêm mới vào trong CSDL



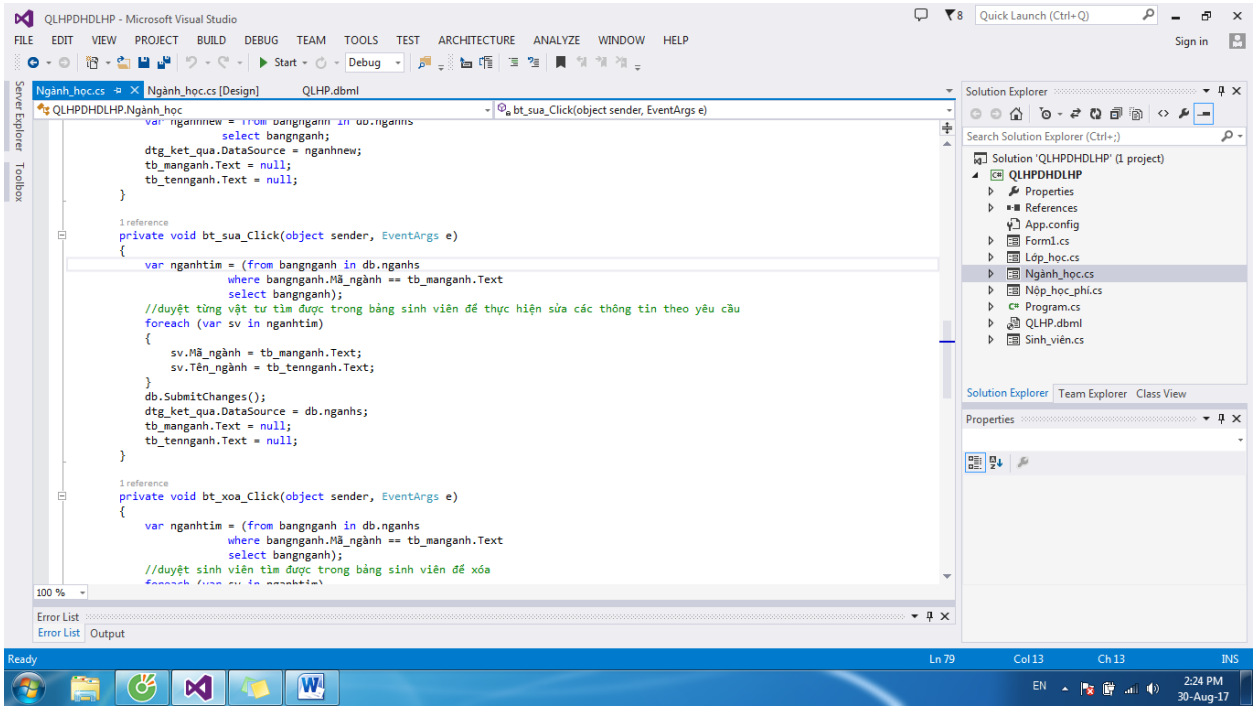
Hình 2.47 Mã lệnh " Lưu thêm "

❖ Câu lệnh cho nút Sửa

+ Đoạn code như sau:

```
private void bt_sua_Click(object sender, EventArgs e)
        {
           var nganhtim = (from bangnganh in db.nganhs
                    where bangnganh.Mã_ngành == tb_manganh.Text
                    select bangnganh);
           //duyệt từng vật tư tìm được trong bảng sinh viên để thực hiện sửa các thông tin
theo yêu cầu
           foreach (var sv in nganhtim)
           {
              sv.Mã_ngành = tb_manganh.Text;
              sv.Tên_ngành = tb_tennganh.Text;
           }
           db.SubmitChanges();
          dtg_ket_qua.DataSource = db.nganhs;
           tb_manganh.Text = null;
         tb_tennganh.Text = null;
                  }
```

+ Ở đây em sử dụng chức năng Update của LinQ to SQL để đưa các thông tin đã sửa vào trong CSDL



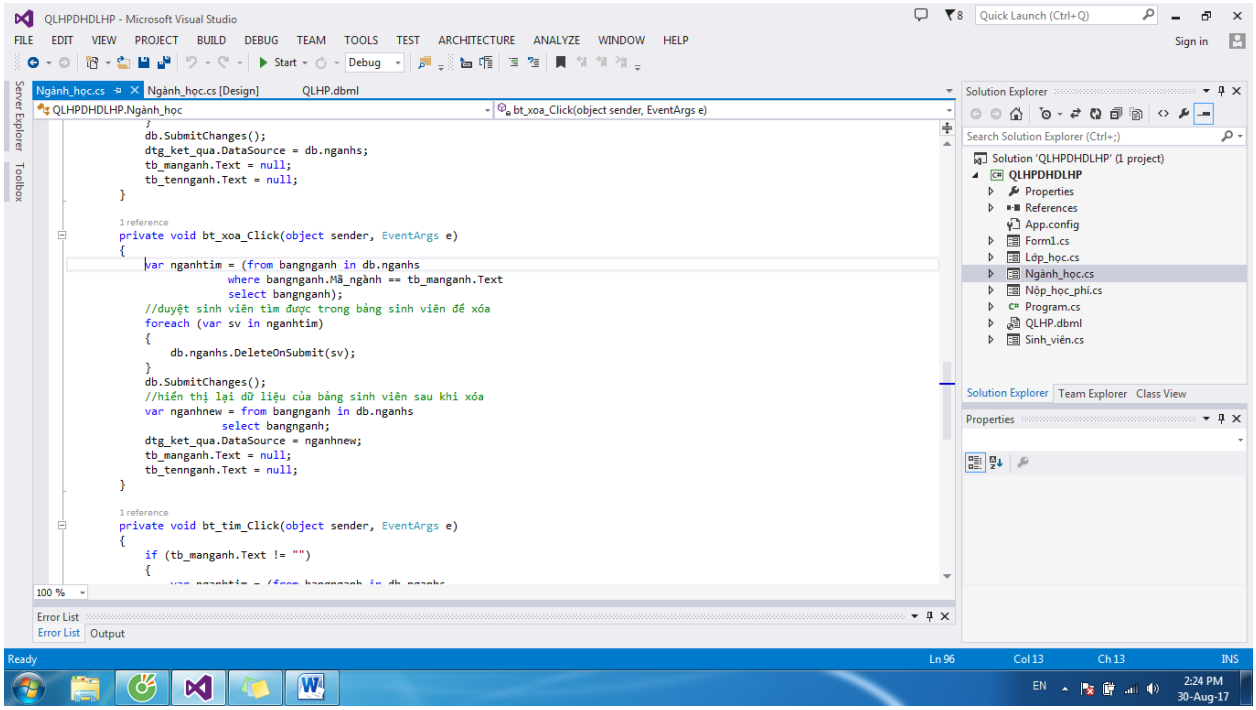
Hình 2.48 Mã lệnh " Sửa "

❖ Câu lệnh cho nút Xóa

+ Đoạn code như sau:

```
private void bt_xoa_Click(object sender, EventArgs e)
      {
         var nganhtim = (from bangnganh in db.nganhs
                         where bangnganh.Mã_ngành == tb_manganh.Text
                         select bangnganh);
         //duyệt sinh viên tìm được trong bảng sinh viên để xóa
         foreach (var sv in nganhtim)
         {
            db.nganhs.DeleteOnSubmit(sv);
         }
         db.SubmitChanges();
            //hiển thị lại dữ liệu của bảng sinh viên sau khi xóa
            var nganhnew = from bangnganh in db.nganhs
                           select bangnganh;
            dtg_ket_qua.DataSource = nganhnew;
            tb_manganh.Text = null;
         tb_tennganh.Text = null;
      }
```

+ Ở đây em sử dụng chức năng DELETE của LinQ to SQL để xóa các thông tin thêm mới vào trong CSDL



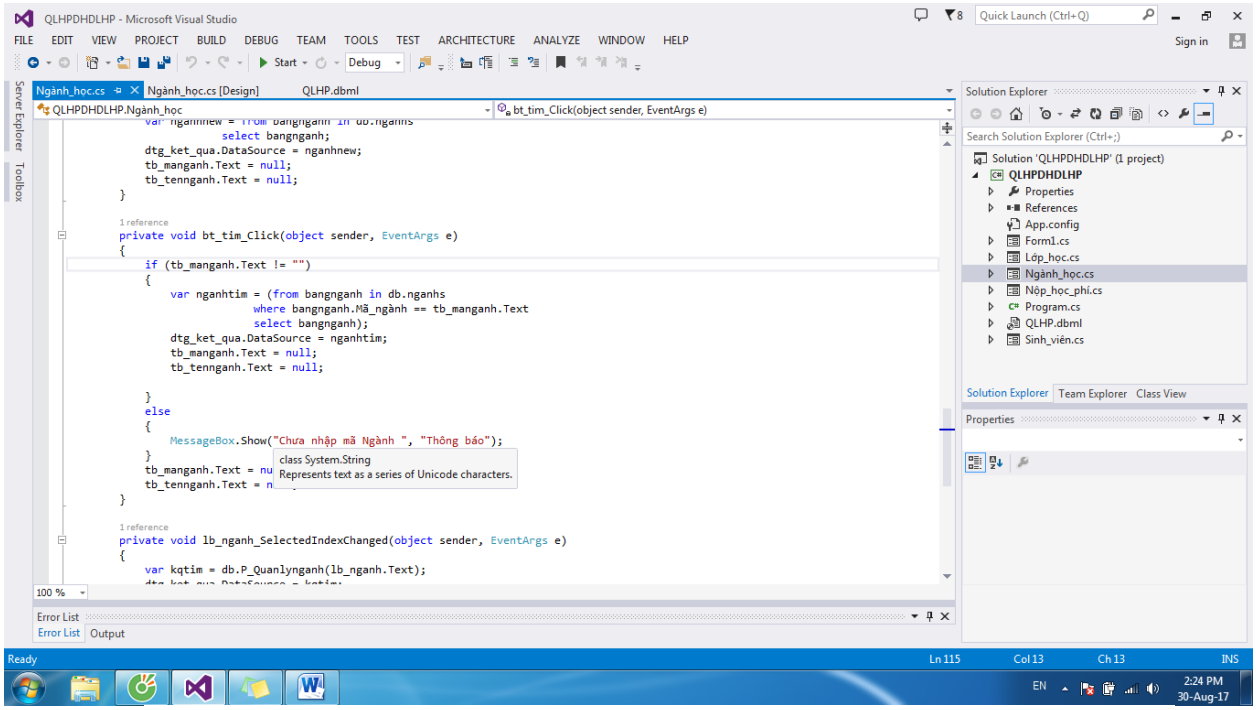
Hình 2.49 Mã lệnh " Xóa "

❖ Câu lệnh cho nút Tìm kiếm ngành theo mã ngành

+ Đoạn code như sau:

```
private void bt_tim_Click(object sender, EventArgs e)
      {
        if (tb_manganh.Text != "")
        {
          var nganhtim = (from bangnganh in db.nganhs
                    where bangnganh.Mã_ngành == tb_manganh.Text
                    select bangnganh);
          dtg_ket_qua.DataSource = nganhtim;
          tb_manganh.Text = null;
          tb_tennganh.Text = null;
        }
        else
        {
          MessageBox.Show("Chưa nhập mã Ngành ", "Thông báo");
        }
        tb_manganh.Text = null;
       tb_tennganh.Text = null;
          }
```

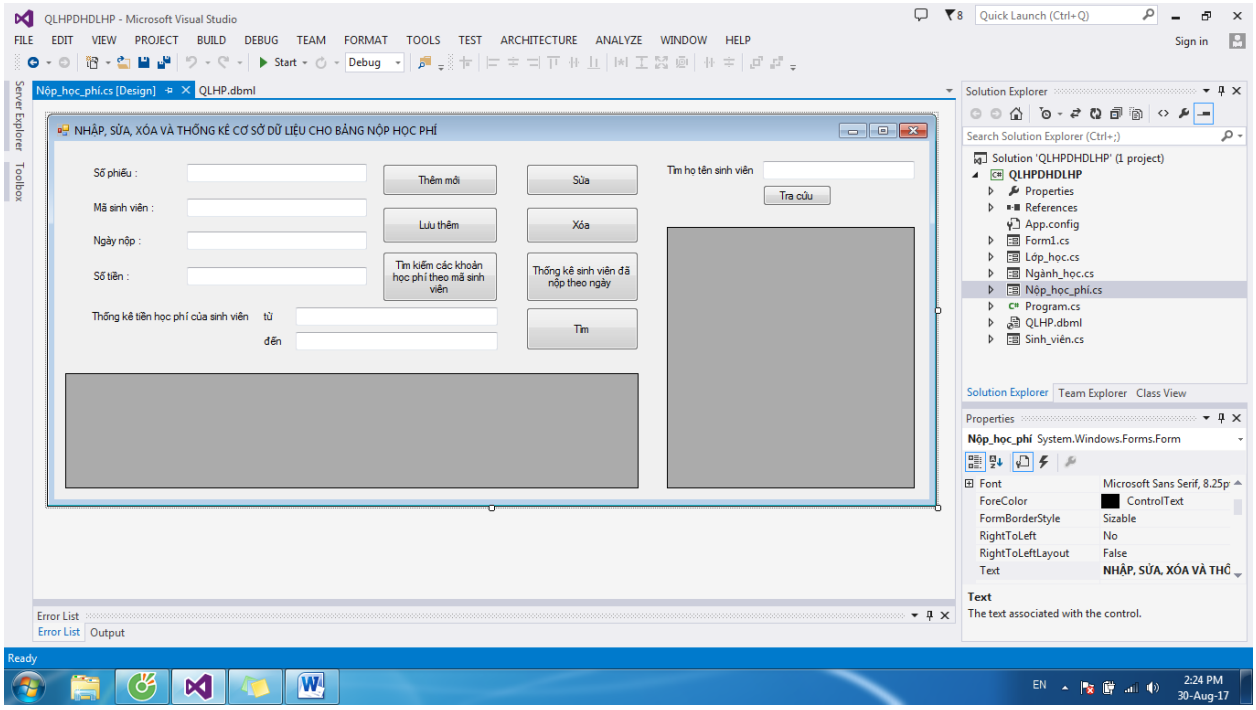
+ Ở đây em sử dụng chức năng Join(liên kết giữa các bảng) của LinQ to SQL để lấy ra các thông tin cần tìm kiếm.



Hình 2.50 Mã lệnh "Tìm kiếm lớp theo mã ngành"

❖ Giao diện bảng Nộp học phí của sinh viên

+ Form Lớp bao gồm các nút Thêm mới, Lưu, Xóa, Sửa, Tìm kiếm theo tên và mã sinh viên, thống kê học phí của các sinh viên trong 1 lớp theo khoảng giá trị.



2.51 Giao diện bảng nộp học phí của sinh viên

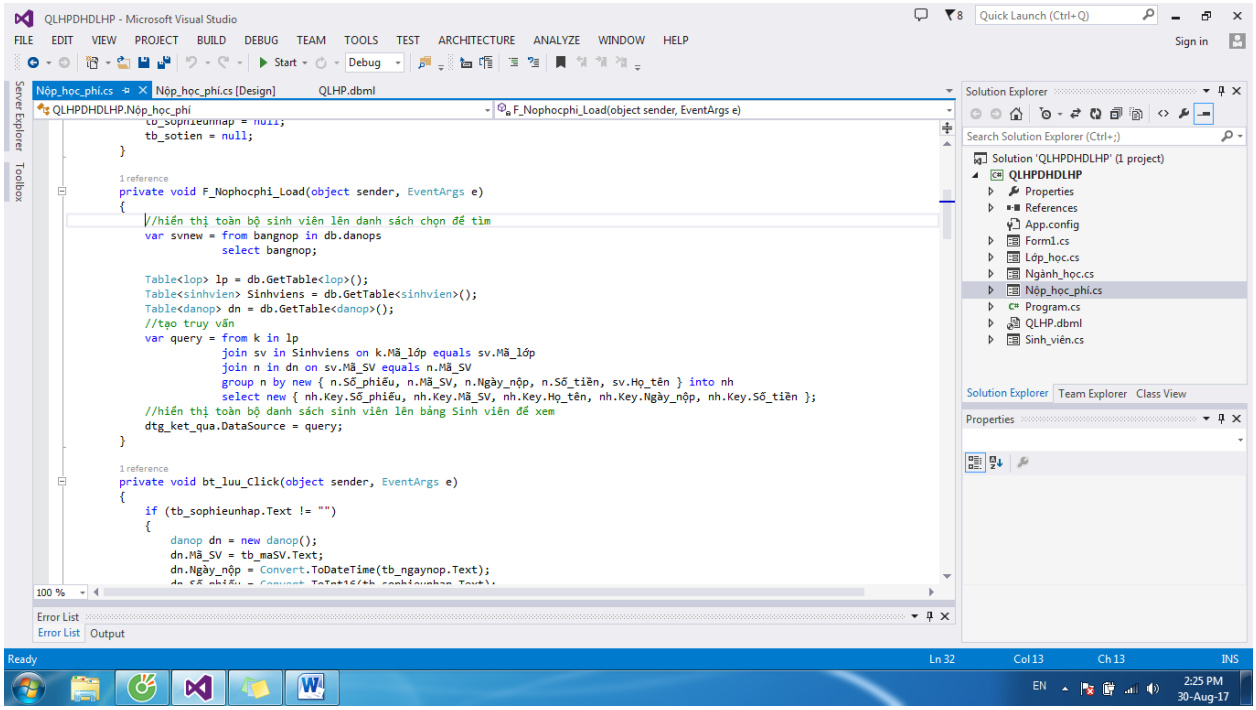
❖ **Tạo truy vấn cho Form Nộp học phí của sinh viên**

+ Đoạn code như sau:

```
private void F_Nophocphi_Load(object sender, EventArgs e)
        {
            //hiển thị toàn bộ sinh viên lên danh sách chọn để tìm
            var svnew = from bangnop in db.danops
                        select bangnop;

            Table<lop> lp = db.GetTable<lop>();
            Table<sinhvien> Sinhviens = db.GetTable<sinhvien>();
            Table<danop> dn = db.GetTable<danop>();
            //tạo truy vấn
            var query = from k in lp
                        join sv in Sinhviens on k.Mã_lớp equals sv.Mã_lớp
                        join n in dn on sv.Mã_SV equals n.Mã_SV
                        group n by new { n.Số_phiếu, n.Mã_SV, n.Ngày_nộp, n.Số_tiền, sv.Họ_tên
} into nh
                        select new { nh.Key.Số_phiếu, nh.Key.Mã_SV, nh.Key.Họ_tên,
nh.Key.Ngày_nộp, nh.Key.Số_tiền };
            //hiển thị toàn bộ danh sách sinh viên lên bảng Sinh viên để xem
            dtg_ket_qua.DataSource = query;
            }
```

+ Ở đây em sử dụng chức năng Select(chọn lọc) và Join(liên kết giữa các bảng) của LinQ to SQL để lấy ra các thông tin cần tìm kiếm.



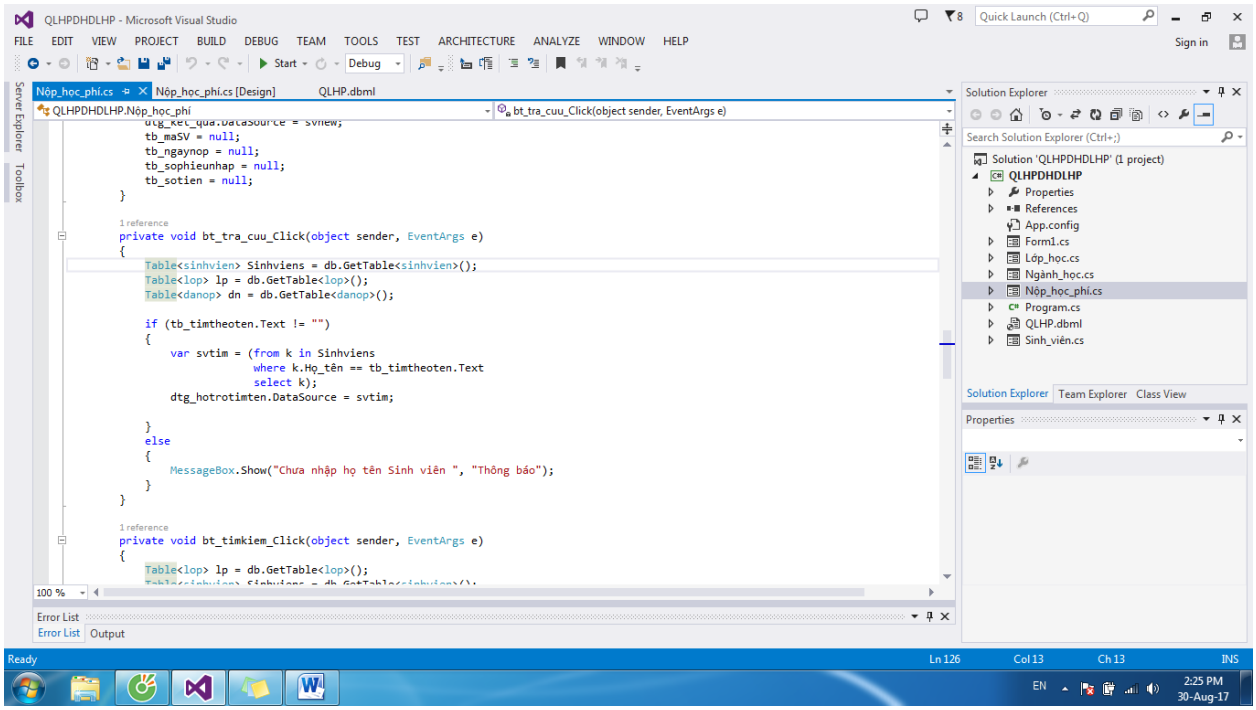
Hình 2.52 Tạo truy vấn cho Form Nộp học phí của sinh viên

❖ Câu lệnh tra cứu tên của 1 sinh viên bất kì

+ Đoạn code như sau:

```
private void bt_tra_cuu_Click(object sender, EventArgs e)
      {
         Table<sinhvien> Sinhviens = db.GetTable<sinhvien>();
            Table<lop> lp = db.GetTable<lop>();
            Table<danop> dn = db.GetTable<danop>();
            if (tb_timtheoten.Text != "")
            {
                        var svtim = (from k in Sinhviens
                         where k.Họ_tên == tb_timtheoten.Text
                         select k);
               dtg_hotrotimten.DataSource = svtim;
            }
            else
            {
               MessageBox.Show("Chưa nhập họ tên Sinh viên ", "Thông báo");
            }
                 }
```

+ Ở đây em sử dụng chức năng SELECT của LinQ to SQL để lấy ra các thông tin cần tìm kiếm.



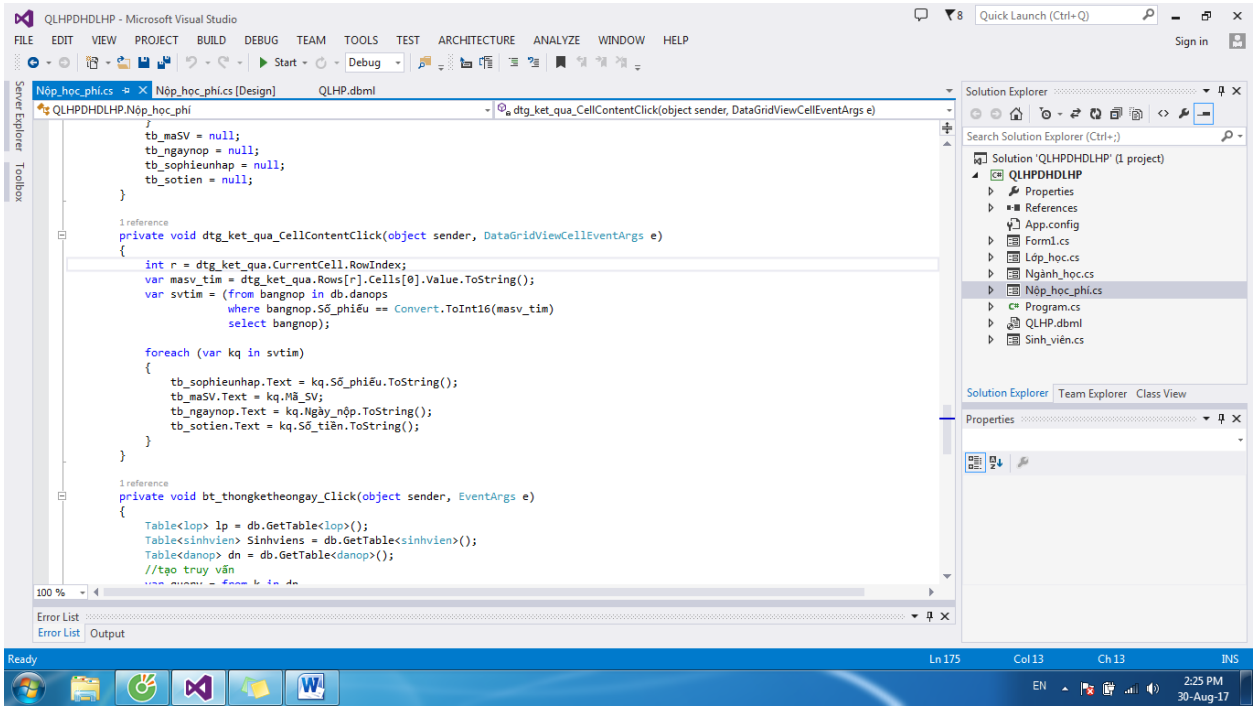
Hình 2.53 Mã lệnh tra cứu tên của 1 sinh viên bất kì

❖ Câu lệnh cho Data Grid View của Form nộp học phí

+ Đoạn code như sau:

```
private void dtg_ket_qua_CellContentClick(object sender, DataGridViewCellEventArgs
e)
    {
       int r = dtg_ket_qua.CurrentCell.RowIndex;
       var masv_tim = dtg_ket_qua.Rows[r].Cells[0].Value.ToString();
       var svtim = (from bangnop in db.danops
                    where bangnop.Số_phiếu == Convert.ToInt16(masv_tim)
                    select bangnop);

       foreach (var kq in svtim)
       {
          tb_sophieunhap.Text = kq.Số_phiếu.ToString();
          tb_maSV.Text = kq.Mã_SV;
          tb_ngaynop.Text = kq.Ngày_nộp.ToString();
          tb_sotien.Text = kq.Số_tiền.ToString();
       }
            }
```



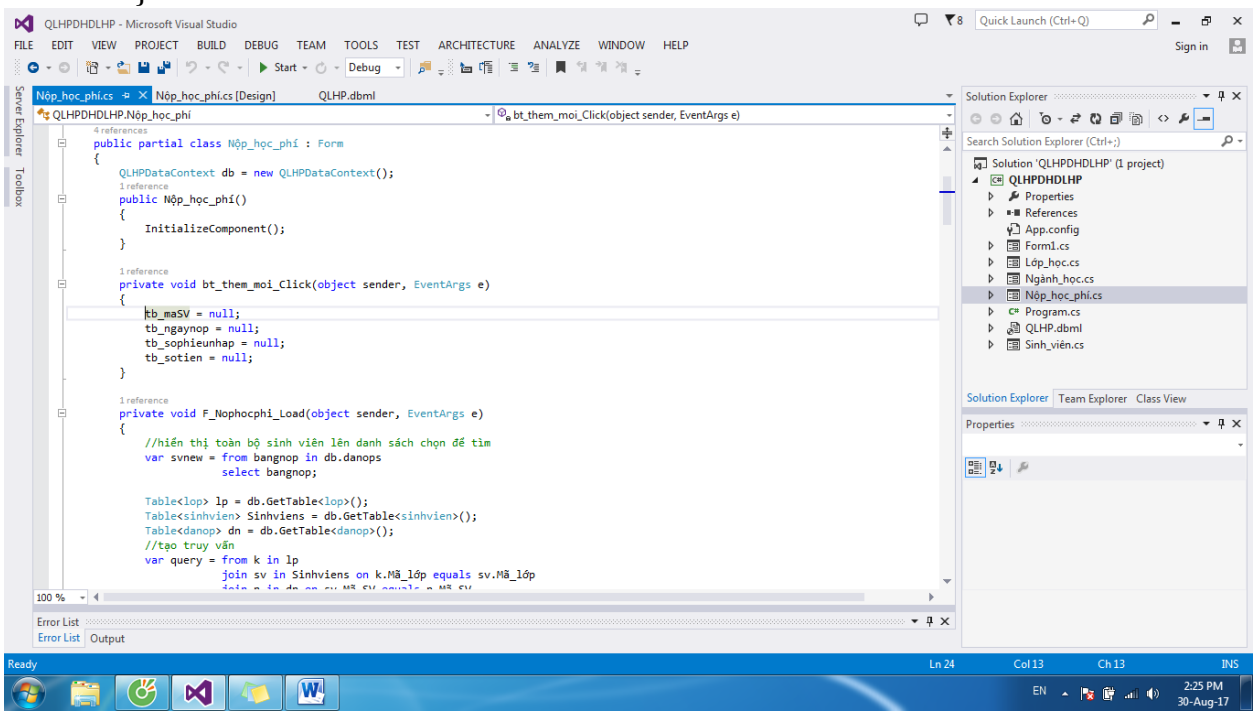
Hình 2.54 Code làm việc của bảng Data Grid View

❖ Câu lệnh cho nút Thêm mới

+ Đoạn code như sau:

```
private void bt_them_moi_Click(object sender, EventArgs e)
```

```
    {
        tb_maSV = null;
        tb_ngaynop = null;
        tb_sophieunhap = null;
        tb_sotien = null;
          }
```



2.55 Mã lệnh " Thêm mới "

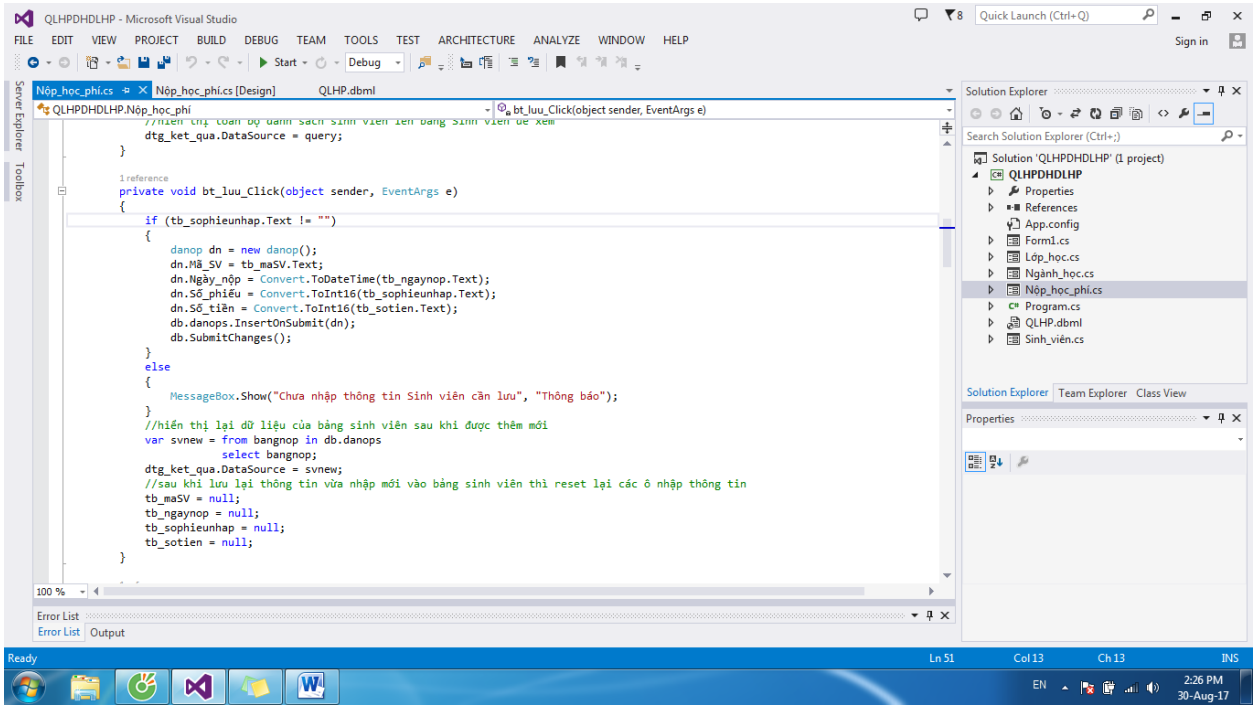
❖ Câu lệnh cho nút Lưu thêm mới

+ Đoạn code như sau:

```
private void bt_luu_Click(object sender, EventArgs e)
    {
        if (tb_sophieunhap.Text != "")
        {
            danop dn = new danop();
            dn.Mã_SV = tb_maSV.Text;
            dn.Ngày_nộp = Convert.ToDateTime(tb_ngaynop.Text);
            dn.Số_phiếu = Convert.ToInt16(tb_sophieunhap.Text);
            dn.Số_tiền = Convert.ToInt16(tb_sotien.Text);
            db.danops.InsertOnSubmit(dn);
            db.SubmitChanges();
        }
        else
        {
            MessageBox.Show("Chưa nhập thông tin Sinh viên cần lưu", "Thông báo");
        }
```

```
        //hiển thị lại dữ liệu của bảng sinh viên sau khi được thêm mới
        var svnew = from bangnop in db.danops
                    select bangnop;
        dtg_ket_qua.DataSource = svnew;
        //sau khi lưu lại thông tin vừa nhập mới vào bảng sinh viên thì reset lại các ô nhập thông tin
        tb_maSV = null;
        tb_ngaynop = null;
        tb_sophieunhap = null;
        tb_sotien = null;
            }
```

+ Ở đây em sử dụng chức năng Insert của LinQ to SQL để đưa các thông tin thêm mới vào trong CSDL



Hình 2.56 Mã lệnh " Lưu thêm "

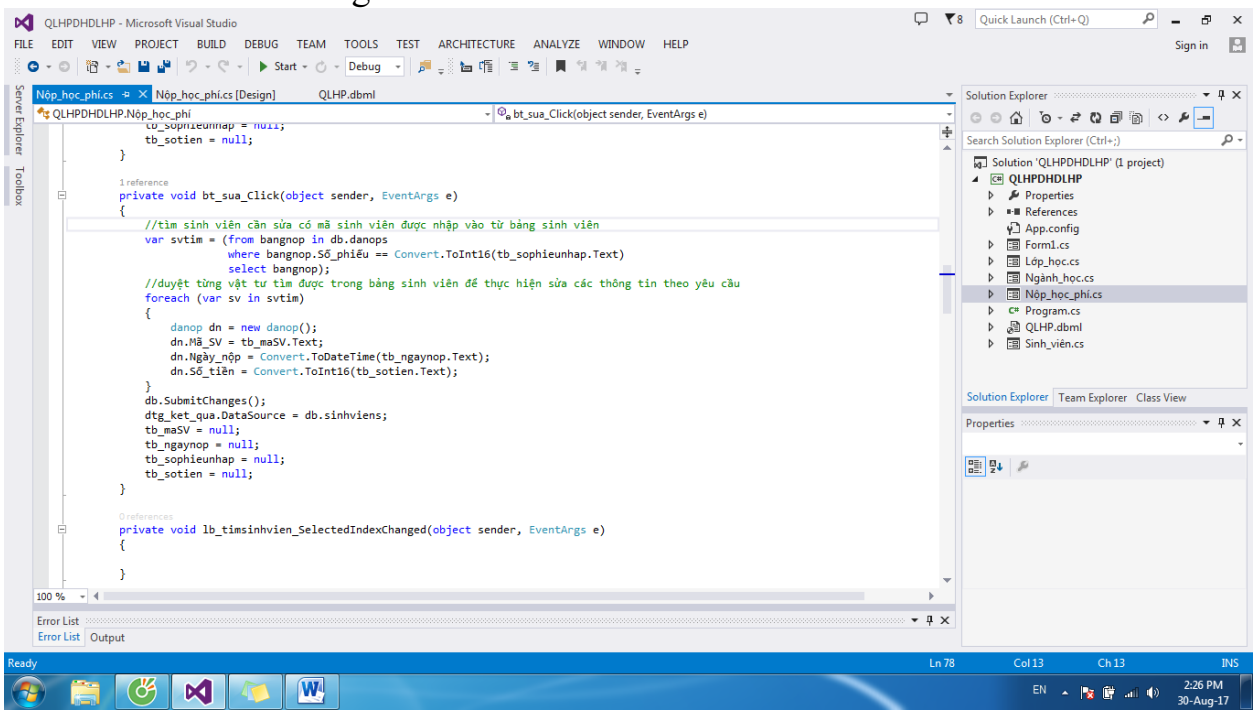
❖ Câu lệnh cho nút Sửa

+ Đoạn code như sau:

```
private void bt_sua_Click(object sender, EventArgs e)
    {
        //tìm sinh viên cần sửa có mã sinh viên được nhập vào từ bảng sinh viên
        var svtim = (from bangnop in db.danops
                     where bangnop.Số_phiếu == Convert.ToInt16(tb_sophieunhap.Text)
                     select bangnop);
        //duyệt từng vật tư tìm được trong bảng sinh viên để thực hiện sửa các thông tin theo yêu cầu
        foreach (var sv in svtim)
```

```
        {
            danop dn = new danop();
            dn.Mã_SV = tb_maSV.Text;
            dn.Ngày_nộp = Convert.ToDateTime(tb_ngaynop.Text);
            dn.Số_tiền = Convert.ToInt16(tb_sotien.Text);
        }
        db.SubmitChanges();
        dtg_ket_qua.DataSource = db.sinhviens;
        tb_maSV = null;
        tb_ngaynop = null;
        tb_sophieunhap = null;
        tb_sotien = null;
            }
```

+ Ở đây em sử dụng chức năng Update của LinQ to SQL để đưa các thông tin thêm mới vào trong CSDL



Hình 2.57 Mã lệnh " Sửa "

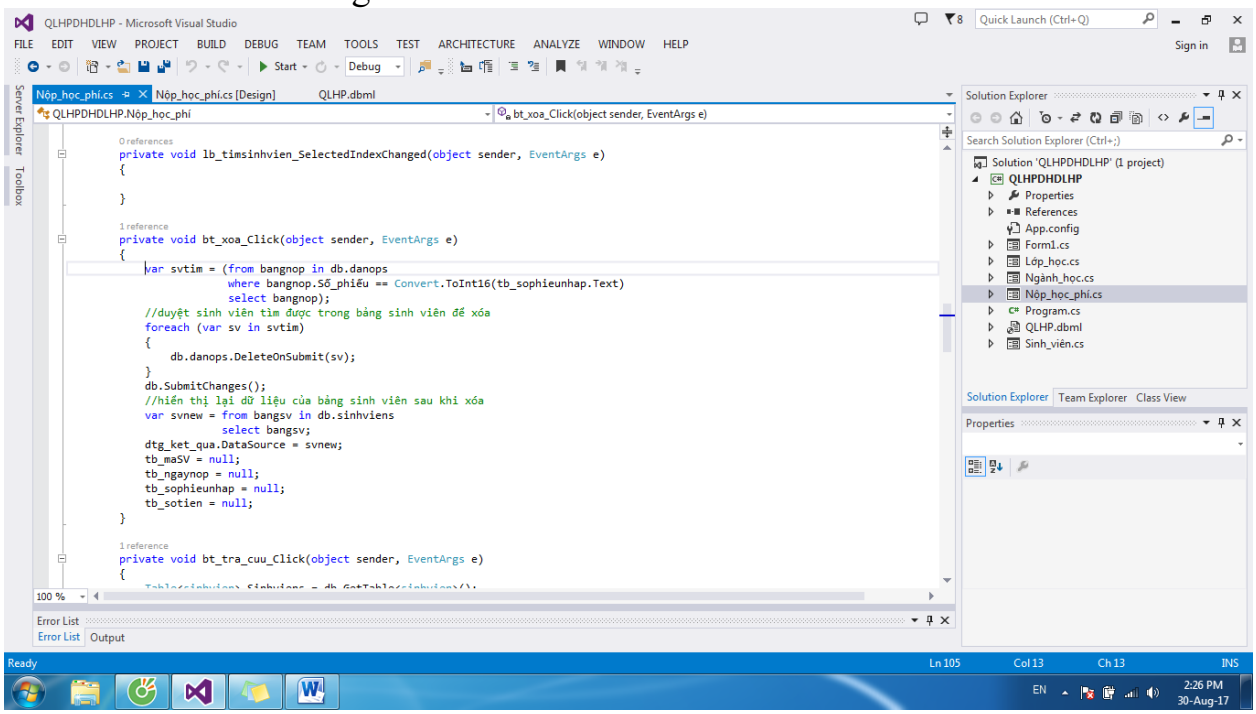
❖ Câu lệnh cho nút Xóa

+ Đoạn code như sau:

```
private void bt_xoa_Click(object sender, EventArgs e)
    {
        var svtim = (from bangnop in db.danops
                     where bangnop.Số_phiếu == Convert.ToInt16(tb_sophieunhap.Text)
                     select bangnop);
        //duyệt sinh viên tìm được trong bảng sinh viên để xóa
        foreach (var sv in svtim)
```

```
        {
            db.danops.DeleteOnSubmit(sv);
        }
        db.SubmitChanges();
        //hiển thị lại dữ liệu của bảng sinh viên sau khi xóa
        var svnew = from bangsv in db.sinhviens
                    select bangsv;
        dtg_ket_qua.DataSource = svnew;
        tb_maSV = null;
        tb_ngaynop = null;
        tb_sophieunhap = null;
        tb_sotien = null;
            }
```

+ Ở đây em sử dụng chức năng Delete của LinQ to SQL để đưa các thông tin thêm mới vào trong CSDL



Hình 2.58 Mã lệnh " Xóa "

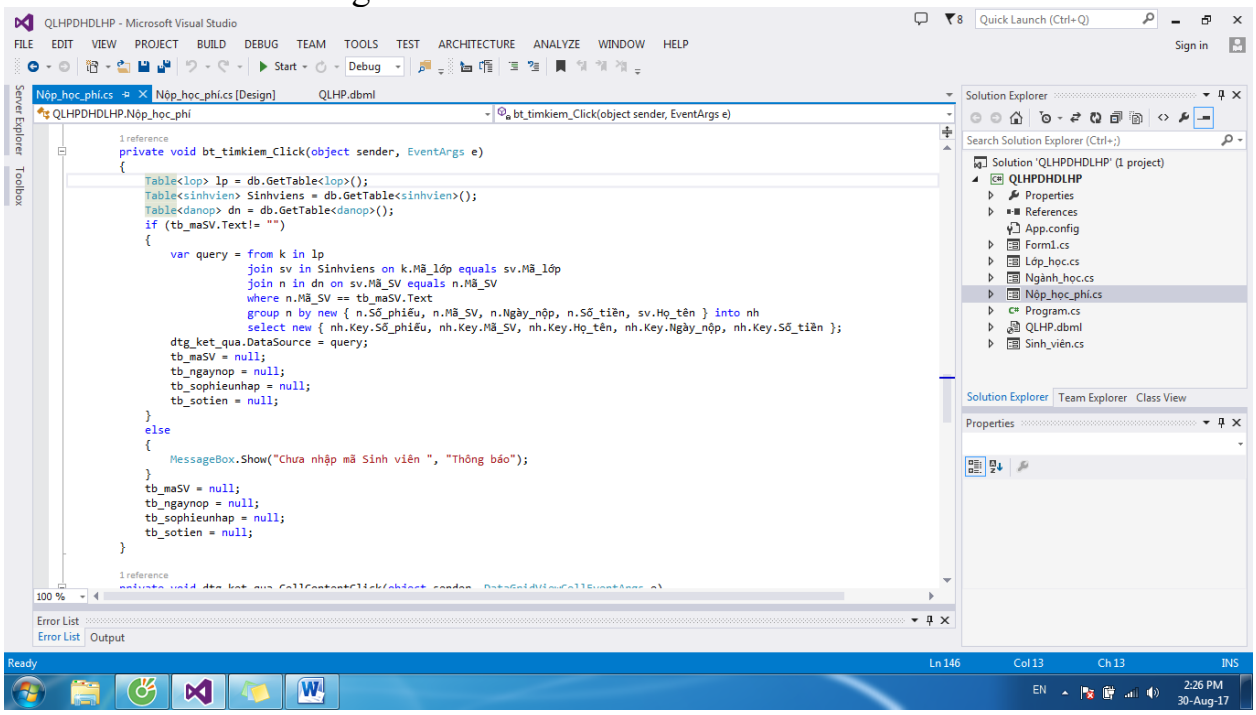
❖ Câu lệnh cho nút Tìm kiếm khoản thu theo mã sinh viên

+ Đoạn code như sau:

```
private void bt_timkiem_Click(object sender, EventArgs e)
        {
            Table<lop> lp = db.GetTable<lop>();
            Table<sinhvien> Sinhviens = db.GetTable<sinhvien>();
            Table<danop> dn = db.GetTable<danop>();
            if (tb_maSV.Text!= "")
            {
                var query = from k in lp
```

```
                join sv in Sinhviens on k.Mã_lớp equals sv.Mã_lớp
                join n in dn on sv.Mã_SV equals n.Mã_SV
                where n.Mã_SV == tb_maSV.Text
                group n by new { n.Số_phiếu, n.Mã_SV, n.Ngày_nộp, n.Số_tiền,
sv.Họ_tên } into nh
                select new { nh.Key.Số_phiếu, nh.Key.Mã_SV, nh.Key.Họ_tên,
nh.Key.Ngày_nộp, nh.Key.Số_tiền };
            dtg_ket_qua.DataSource = query;
            tb_maSV = null;
            tb_ngaynop = null;
            tb_sophieunhap = null;
            tb_sotien = null;
        }
        else
        {
            MessageBox.Show("Chưa nhập mã Sinh viên ", "Thông báo");
        }
        tb_maSV = null;
        tb_ngaynop = null;
        tb_sophieunhap = null;
        tb_sotien = null;
            }
```

+ Ở đây em sử dụng chức năng Join của LinQ to SQL để đưa các thông tin thêm mới vào trong CSDL



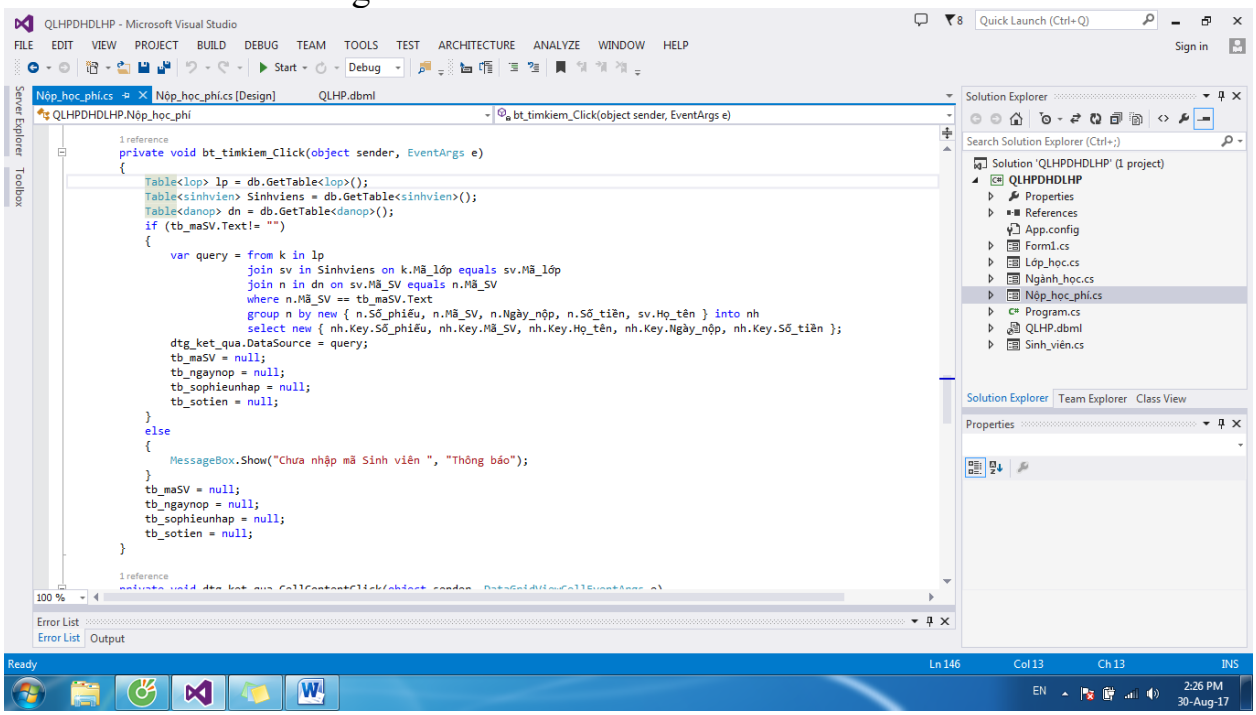
Hình 2.59 Mã lệnh "Tìm kiếm khoản thu theo mã sinh viên"

❖ Câu lệnh cho nút thống kê sinh viên đã nộp theo ngày

+ Đoạn code như sau:

```
private void bt_thongketheongay_Click(object sender, EventArgs e)
    {
      Table<lop> lp = db.GetTable<lop>();
      Table<sinhvien> Sinhviens = db.GetTable<sinhvien>();
      Table<danop> dn = db.GetTable<danop>();
      //tạo truy vấn
      var query = from k in dn
                  join sv in Sinhviens on k.Mã_SV equals sv.Mã_SV
                  group sv by new { k.Ngày_nộp } into nh
                  select new
                  {
                     nh.Key.Ngày_nộp,
                     Số_Sinh_viên_đã_nộp_tiền = nh.Count()
                  };
      dtg_ket_qua.DataSource = query;
          }
```

+ Ở đây em sử dụng chức năng Join của LinQ to SQL để đưa các thông tin thêm mới vào trong CSDL



Hình 2.60 Mã lệnh " Thống kê sinh viên đã nộp theo ngày"

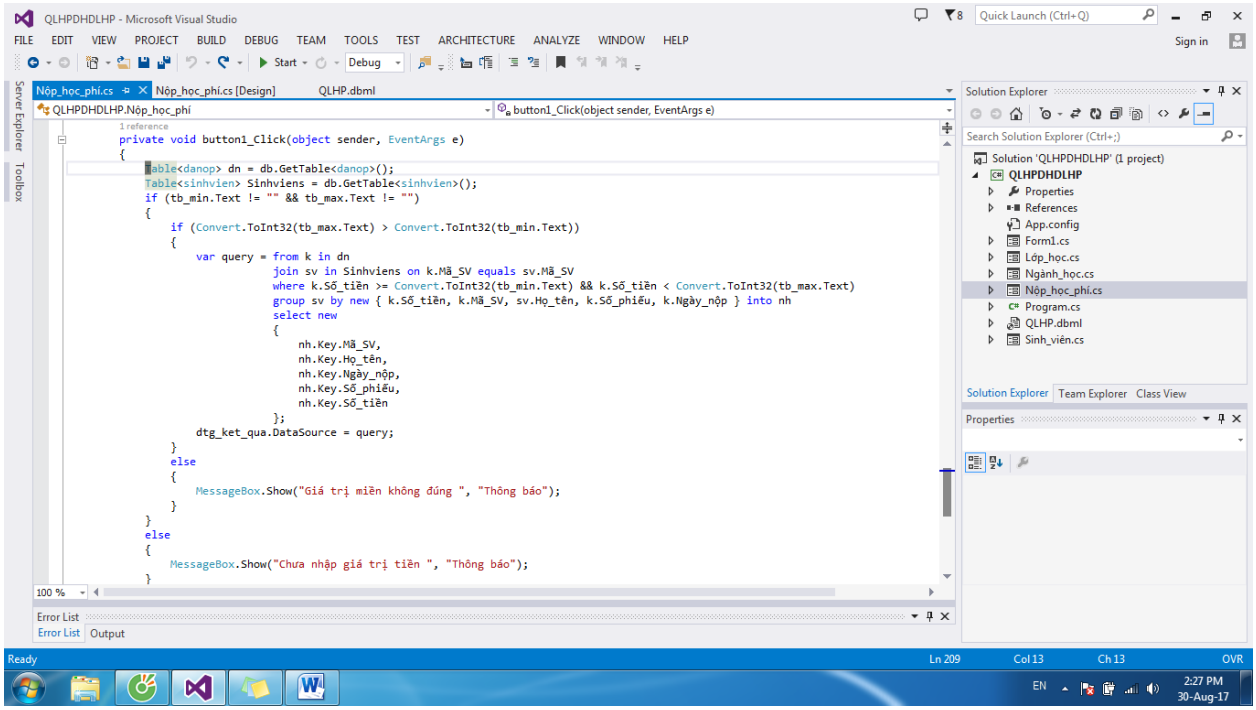
❖ Câu lệnh cho nút thống kê các khoản thu theo khoảng giá trị

+ Đoạn code như sau:

```
private void button1_Click(object sender, EventArgs e)
```

```
{
    Table<danop> dn = db.GetTable<danop>();
    Table<sinhvien> Sinhviens = db.GetTable<sinhvien>();
    if (tb_min.Text != "" && tb_max.Text != "")
    {
        if (Convert.ToInt32(tb_max.Text) > Convert.ToInt32(tb_min.Text))
        {
            var query = from k in dn
                        join sv in Sinhviens on k.Mã_SV equals sv.Mã_SV
                        where k.Số_tiền >= Convert.ToInt32(tb_min.Text) && k.Số_tiền <
Convert.ToInt32(tb_max.Text)
                        group sv by new { k.Số_tiền, k.Mã_SV, sv.Họ_tên, k.Số_phiếu,
k.Ngày_nộp } into nh
                        select new
                        {
                            nh.Key.Mã_SV,
                            nh.Key.Họ_tên,
                            nh.Key.Ngày_nộp,
                            nh.Key.Số_phiếu,
                            nh.Key.Số_tiền
                        };
            dtg_ket_qua.DataSource = query;
        }
        else
        {
            MessageBox.Show("Giá trị miền không đúng ", "Thông báo");
        }
    }
    else
    {
        MessageBox.Show("Chưa nhập giá trị tiền ", "Thông báo");
    }
     }
```

+ Ở đây em sử dụng chức năng Join của LinQ to SQL để đưa các thông tin thêm mới vào trong CSDL



Hình 2.61 Mã lệnh "Tìm kiếm khoản thu theo khoảng giá trị"

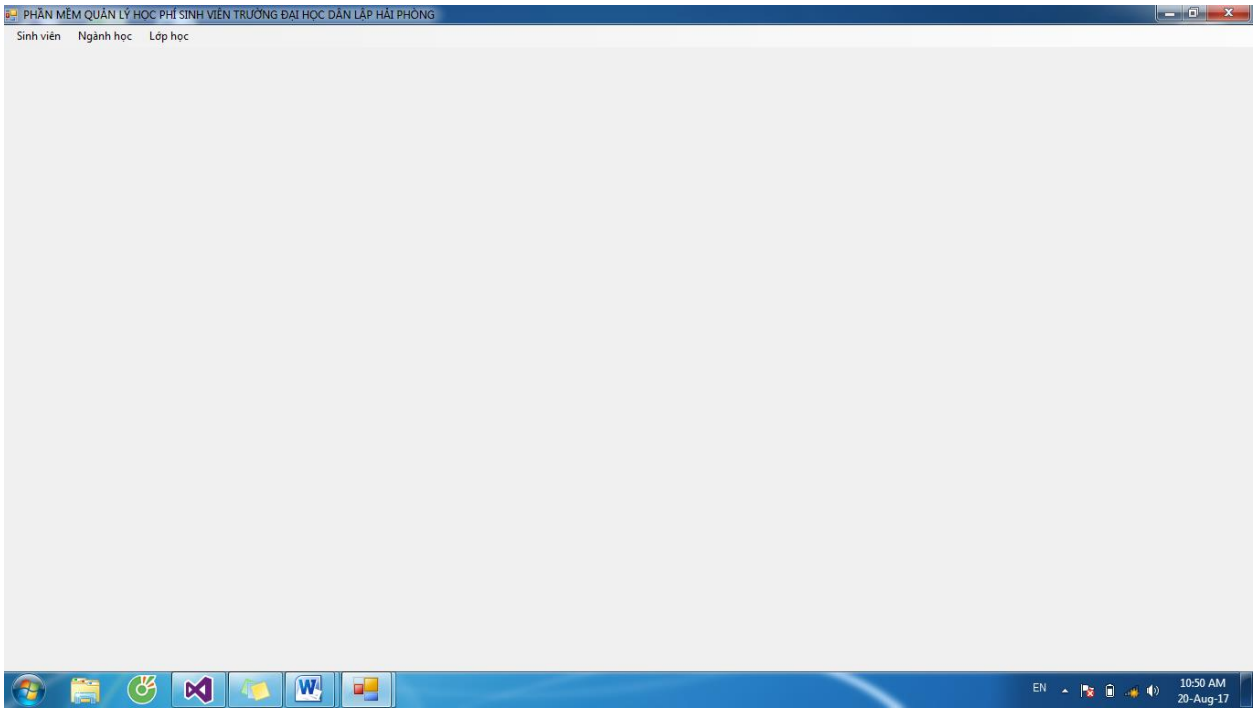
CHƯƠNG 3

KẾT QUẢ THỰC NGHIỆM

3.1 Kết quả chương trình ứng dụng

3.1.1 Các giao diện

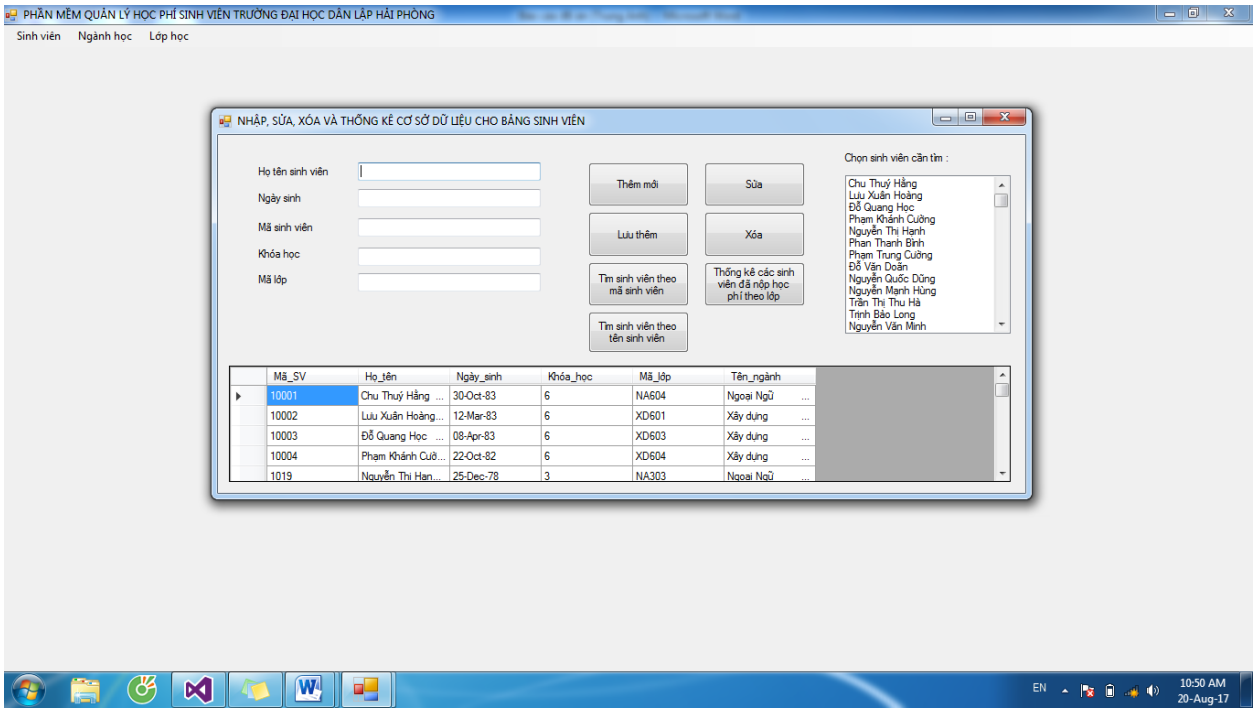
a) Giao diện Menu



Hình 3.1 Giao diện Menu

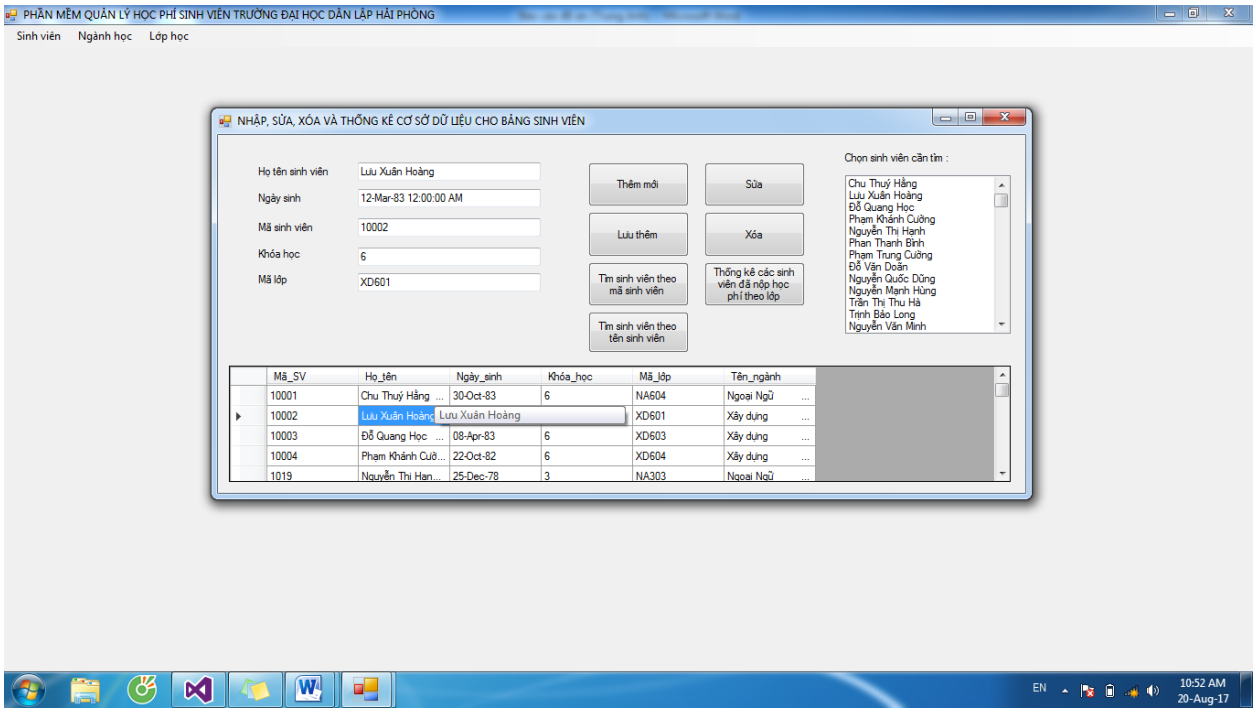
b) Giao diện Form sinh viên

❖ **Giao diện Thêm, sửa, xóa, tìm kiếm và thống kê sinh viên**



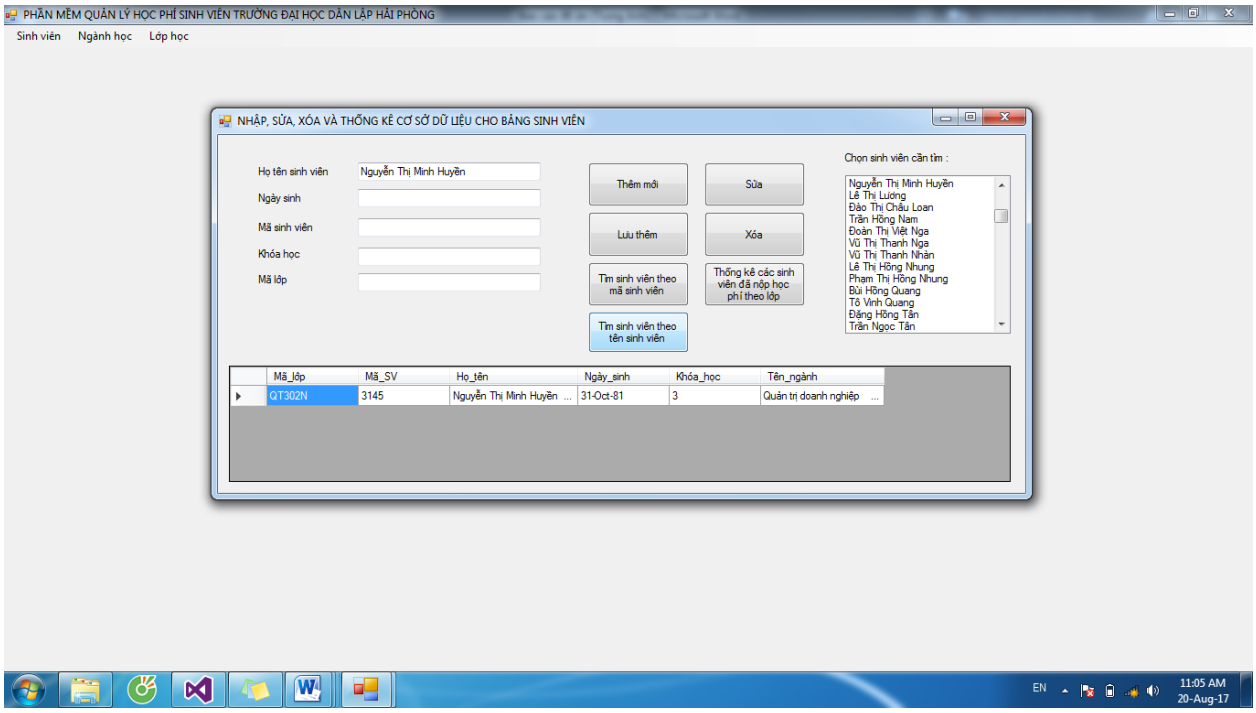
Hình 3.2 Giao diện Thêm, sửa, xóa, tìm kiếm và thống kê sinh viên

❖ Cập nhật sinh viên



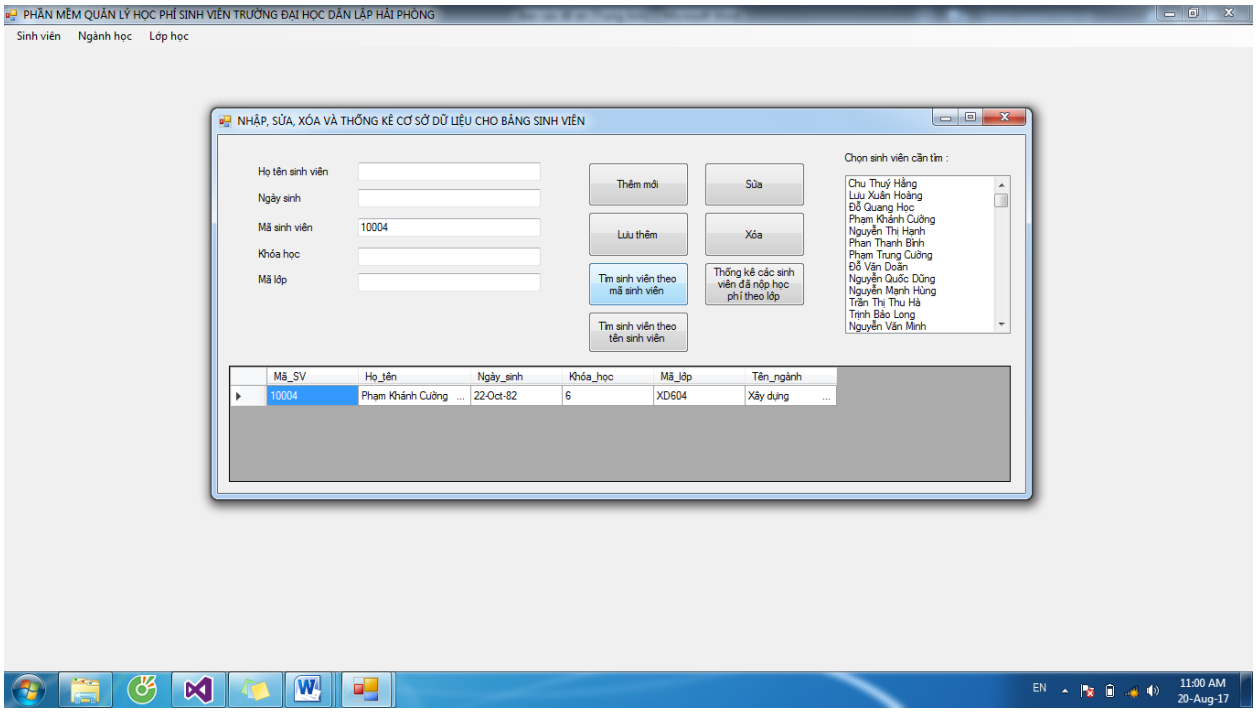
Hình 3.3 Cập nhật sinh viên

❖ Tìm kiếm sinh viên theo tên sinh viên



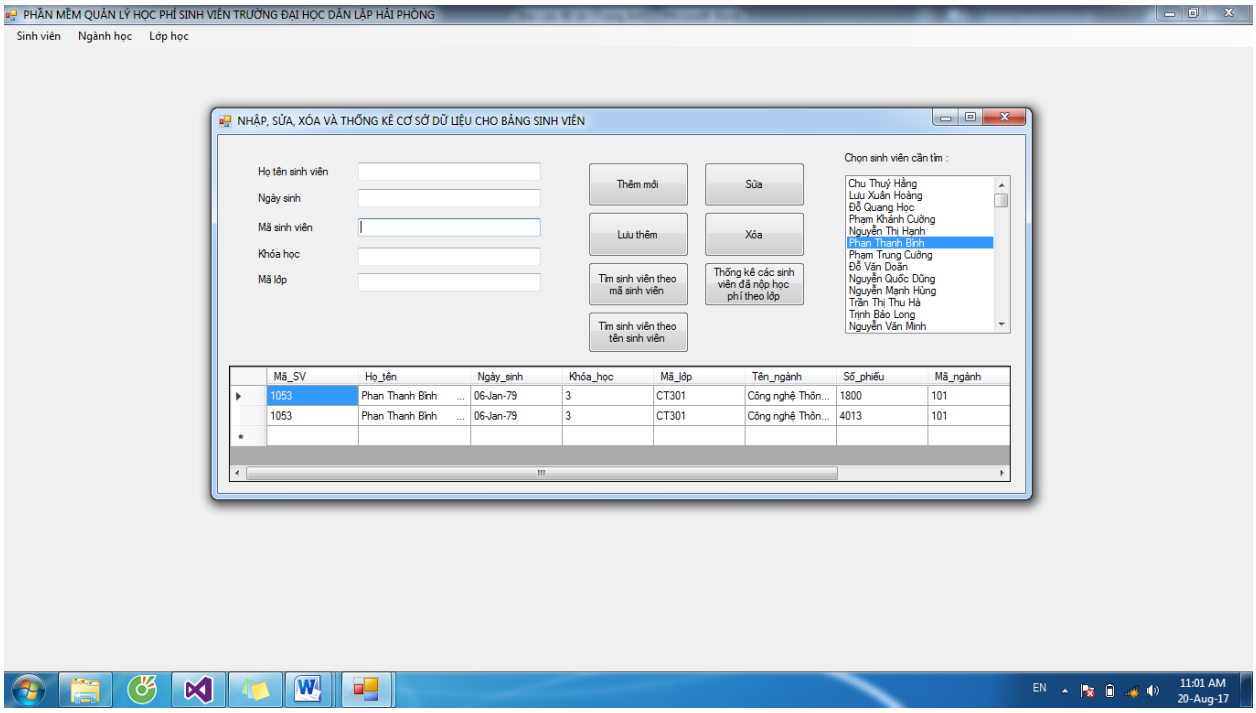
Hình 3.4 Tìm kiếm sinh viên theo tên sinh viên

❖ Tìm kiếm sinh viên theo mã sinh viên



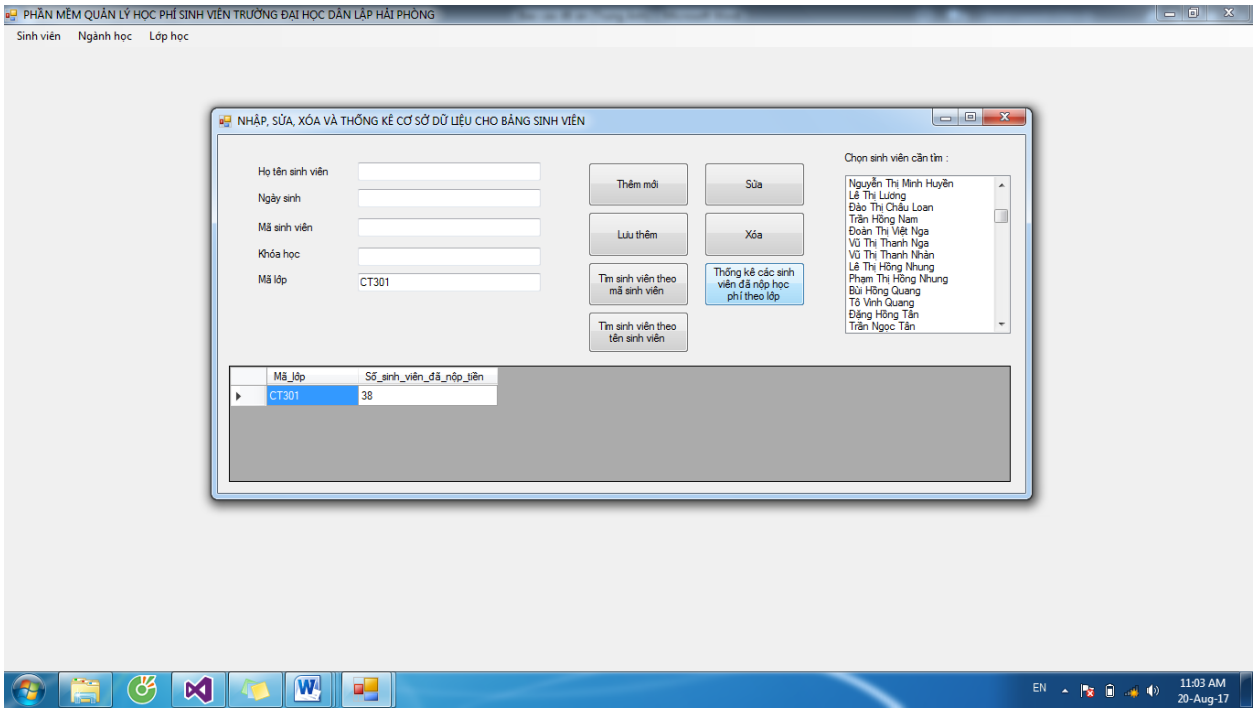
Hình 3.5 Tìm kiếm sinh viên theo mã sinh viên

❖ Lấy thông tin sinh viên từ ListBox



Hình 3.6 Thông tin sinh viên từ ListBox

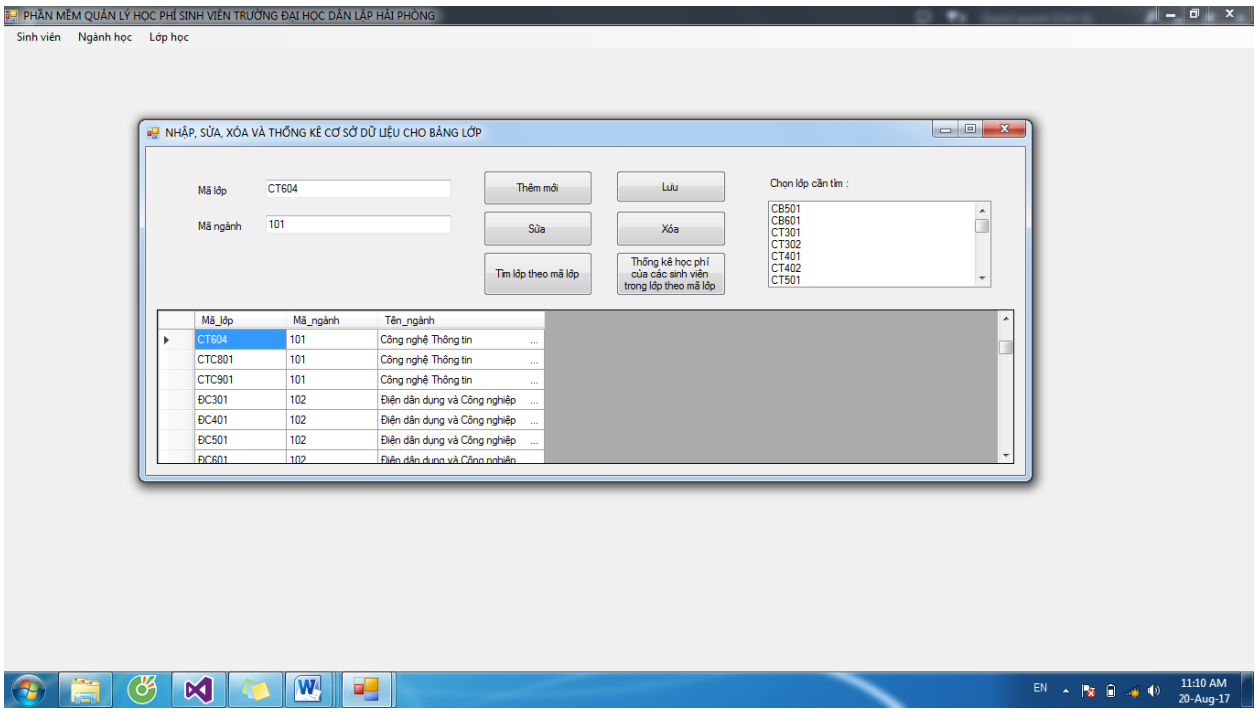
❖ Thống kê các sinh viên đã nộp tiền theo mã lớp



Hình 3.7 Thống kê các sinh viên đã nộp tiền theo mã lớp

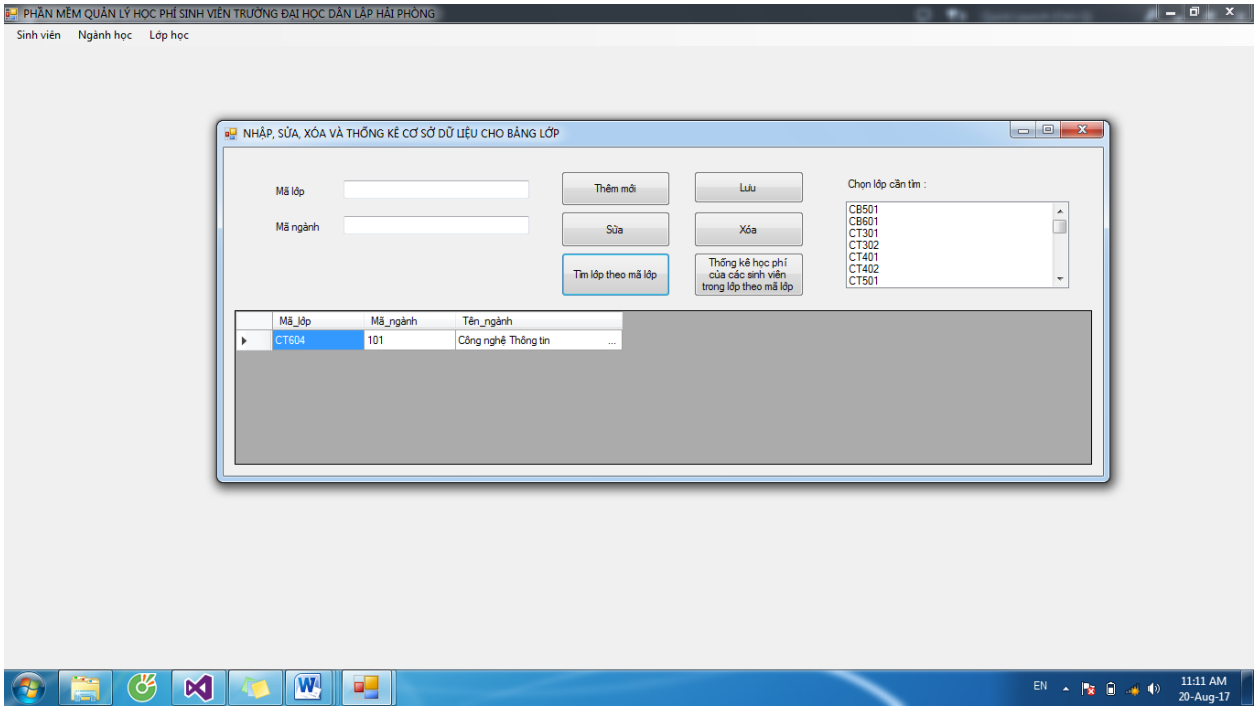
c) Giao diện Form lớp

❖ Cập nhật lớp



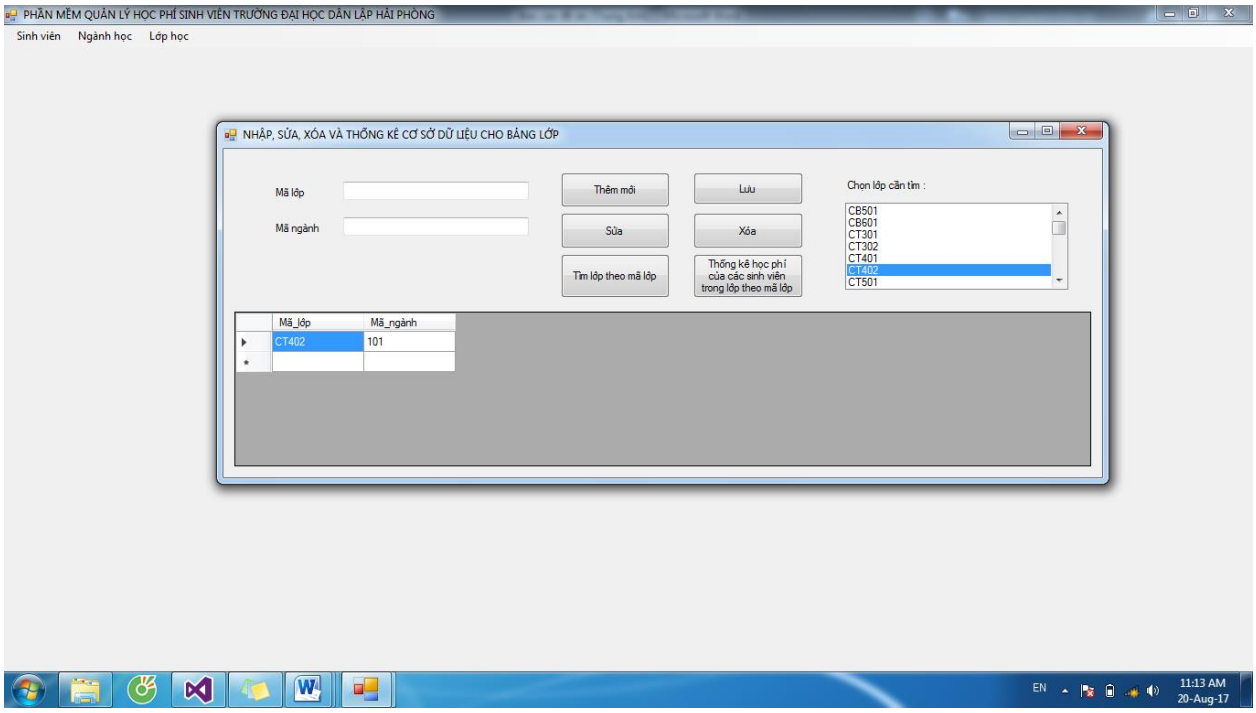
Hình 3.8 Cập nhật lớp

❖ Tìm kiếm lớp theo mã lớp



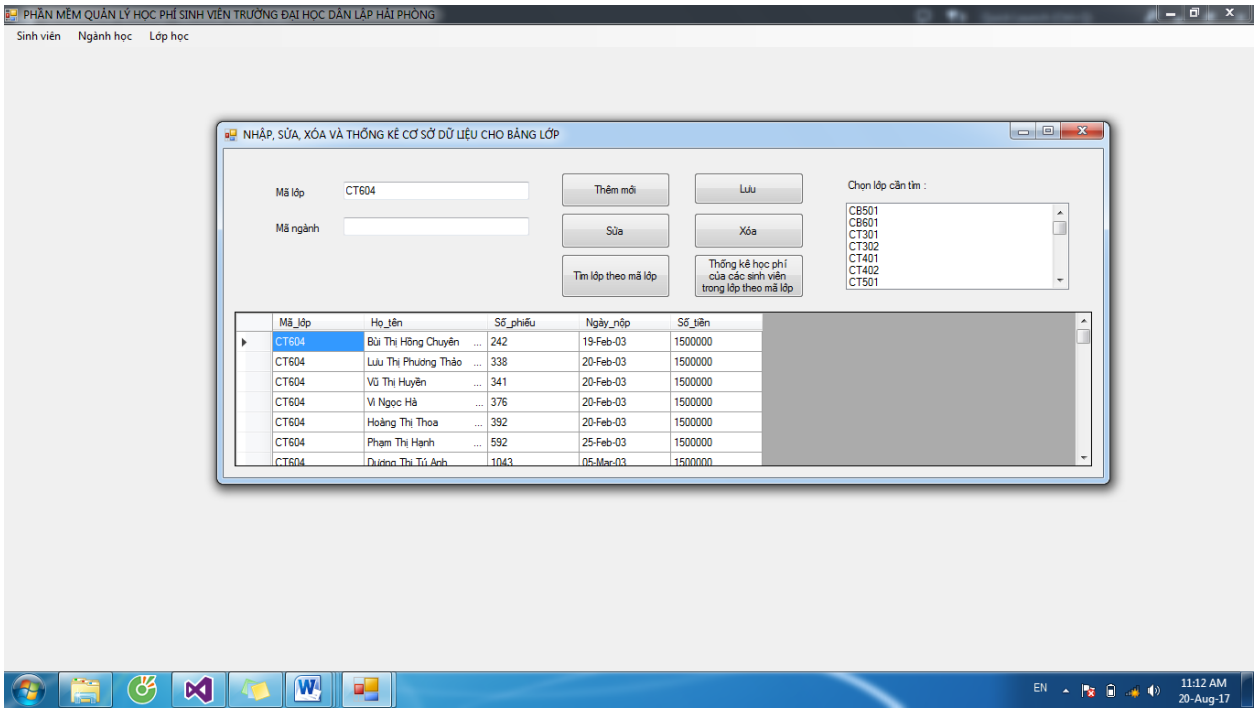
Hình 3.9 Tìm kiếm lớp theo mã lớp

❖ Lấy thông tin lớp từ ListBox



Hình 3.10 Thông tin lớp từ ListBox

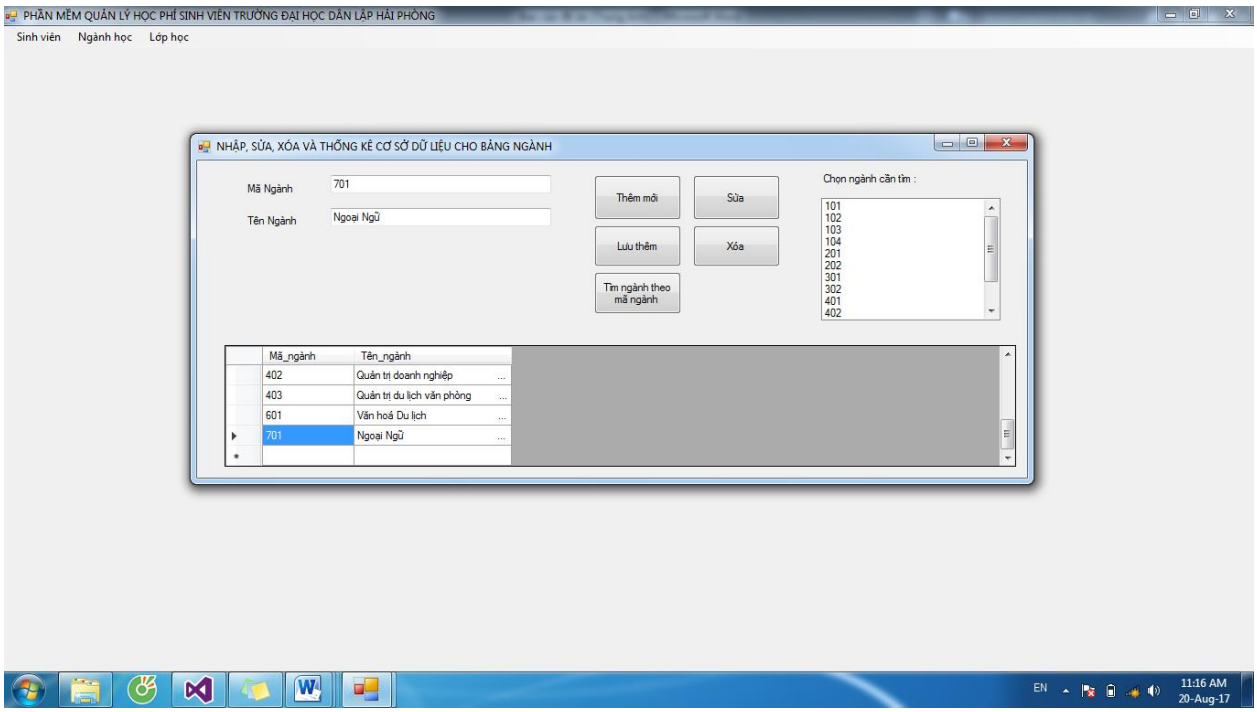
❖ Thống kê các sinh viên đã nộp tiền theo mã lớp



Hình 3.11 Thống kê các sinh viên trong lớp đã nộp tiền theo mã lớp

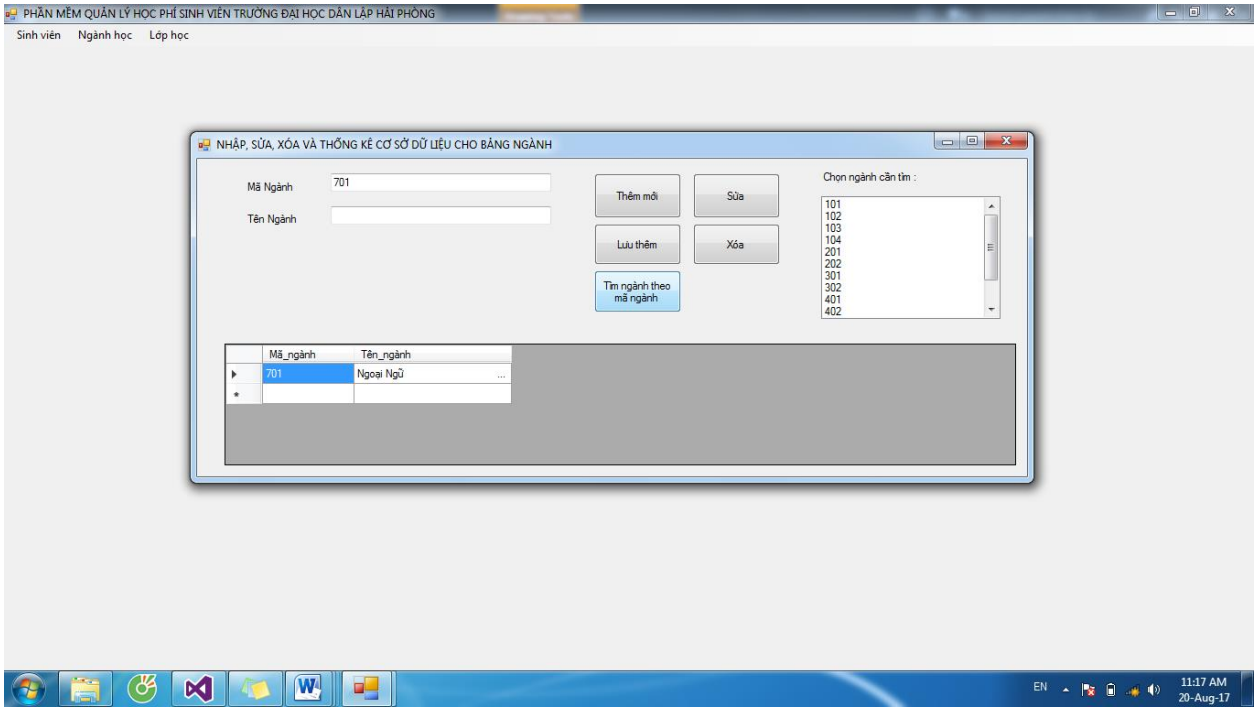
d) Giao diện form Ngành

❖ **Cập nhật ngành**



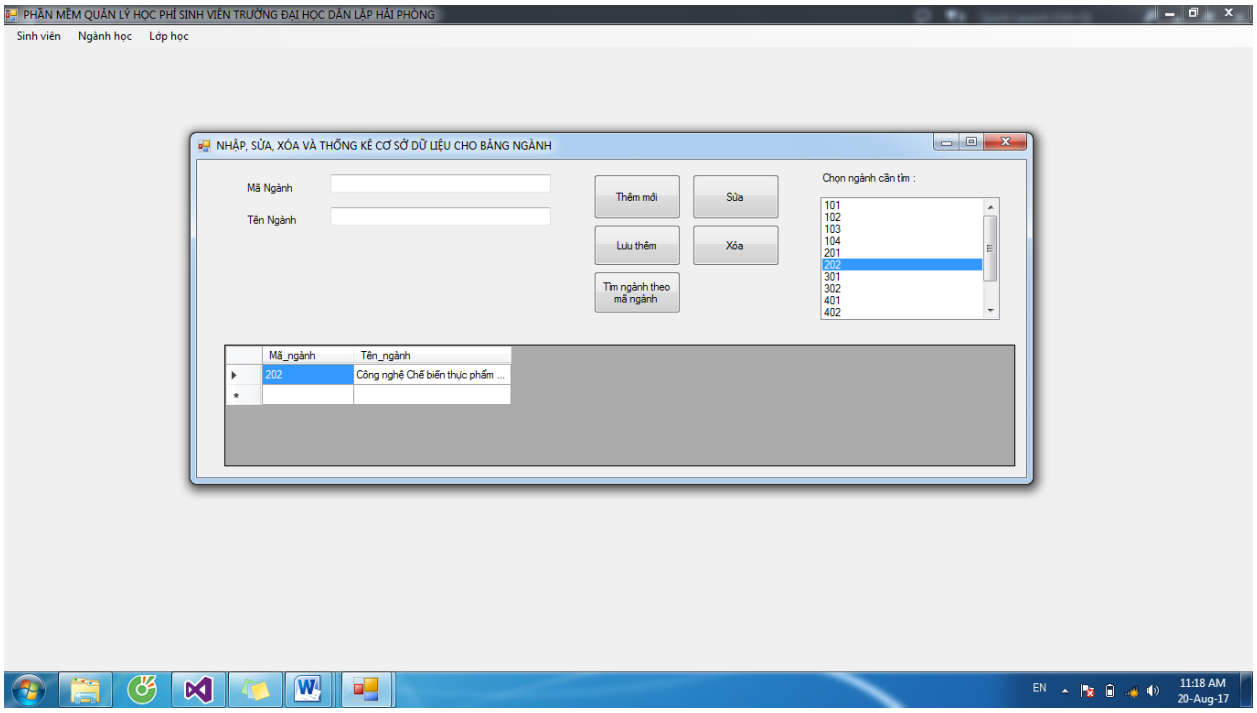
Hình 3.12 Cập nhật ngành

❖ **Tìm kiếm ngành theo mã ngành**



Hình 3.13 Tìm kiếm ngành theo mã ngành

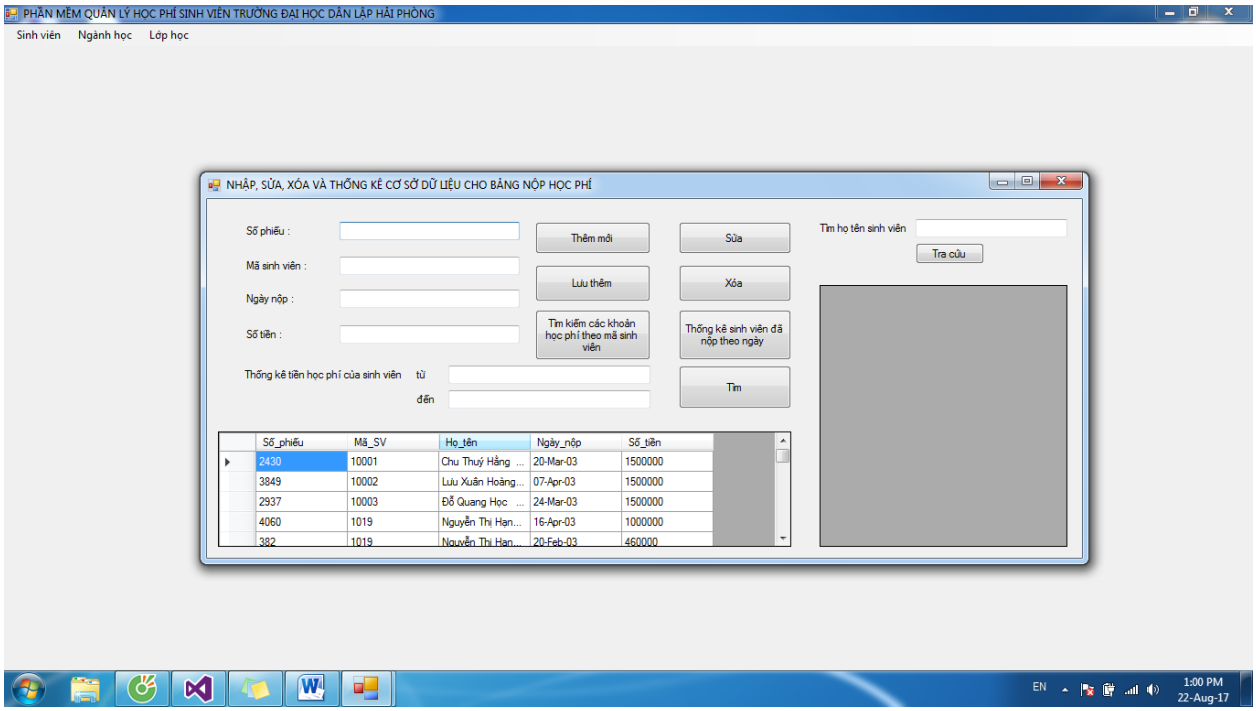
❖ Lấy thông tin lớp từ ListBox



Hình 3.14 Thông tin ngành từ ListBox

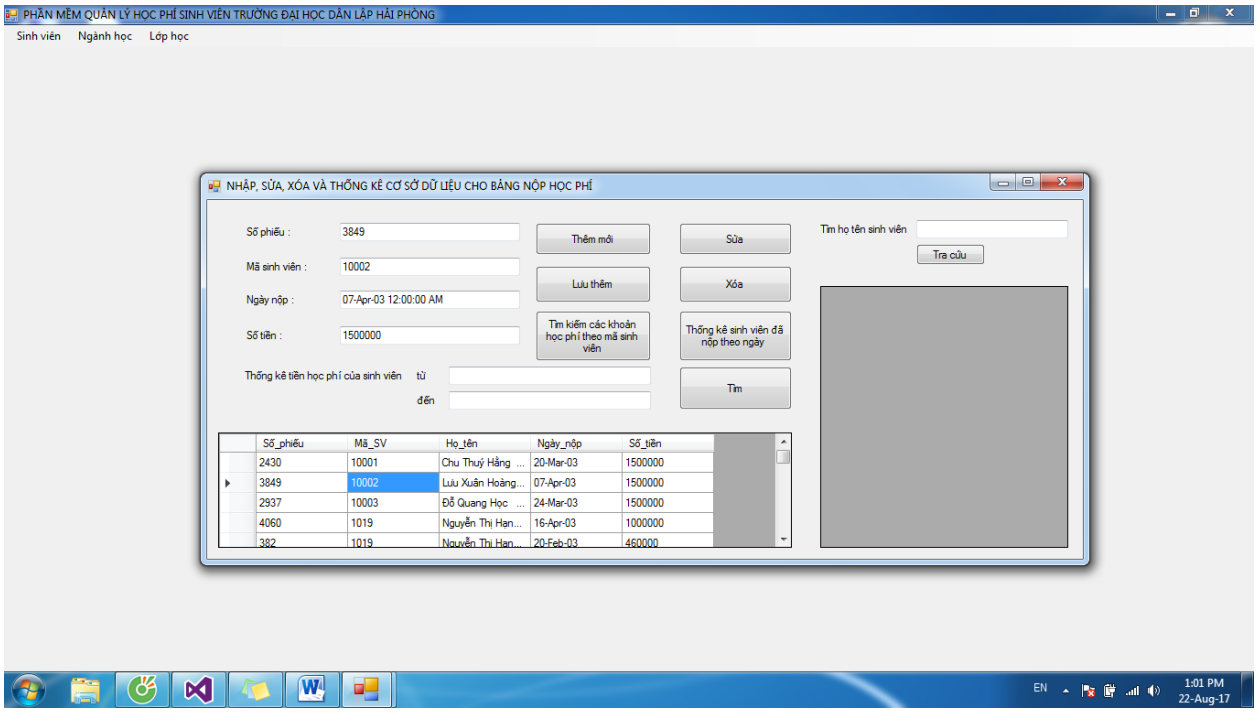
e) Giao diện Form Nộp học phí

❖ Giao diện Form nộp học phí



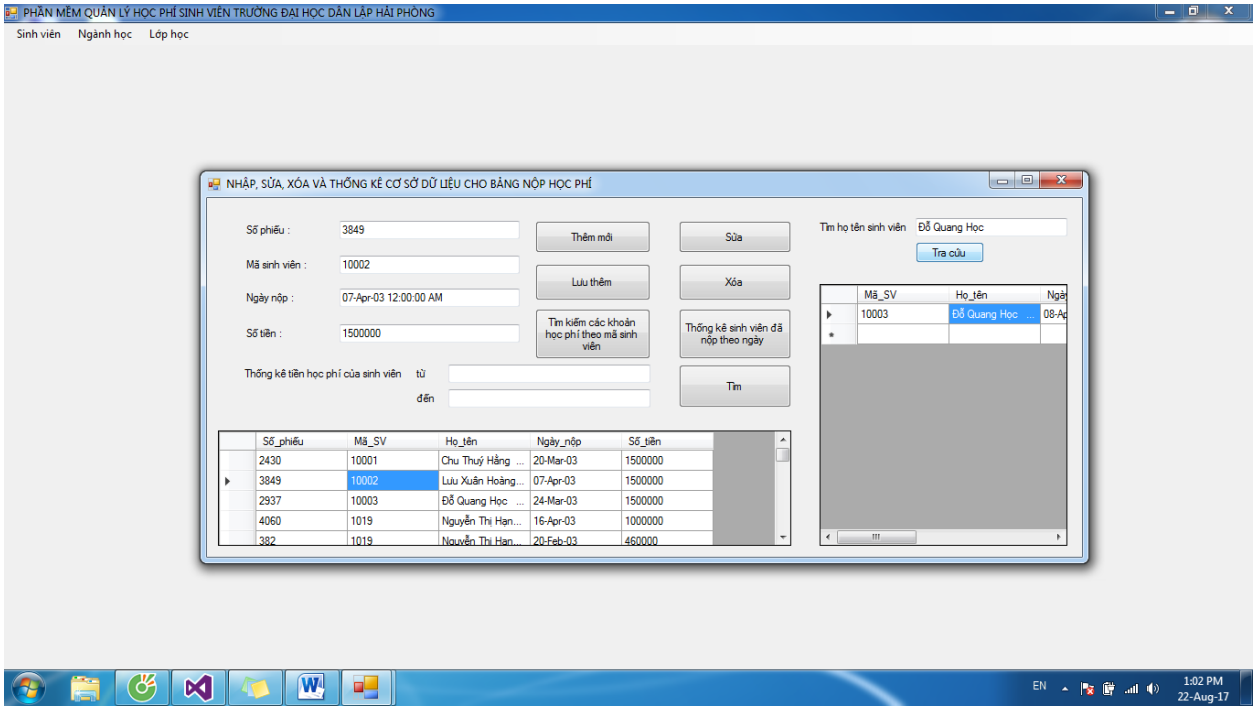
Hình 3.15 Giao diện Form nộp học phí

❖ Cập nhật sinh viên nộp học phí



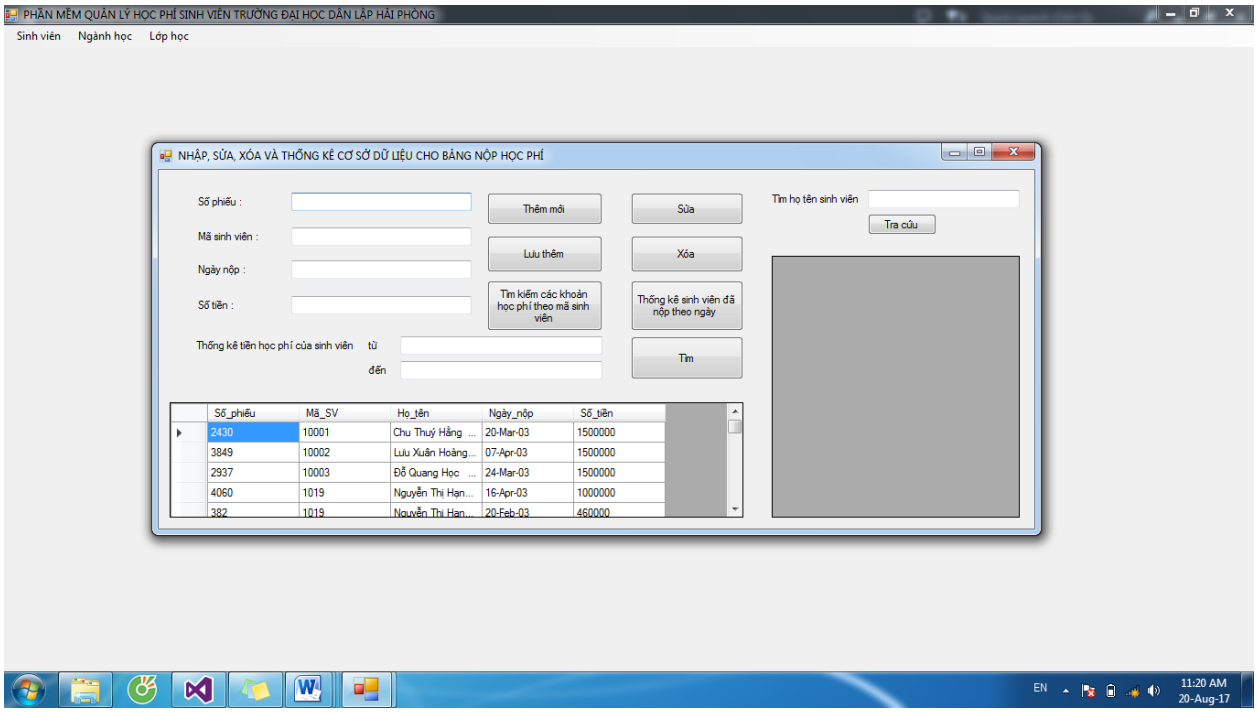
Hình 3.16 Cập nhật sinh viên nộp học phí

❖ Tra cứu tên sinh viên



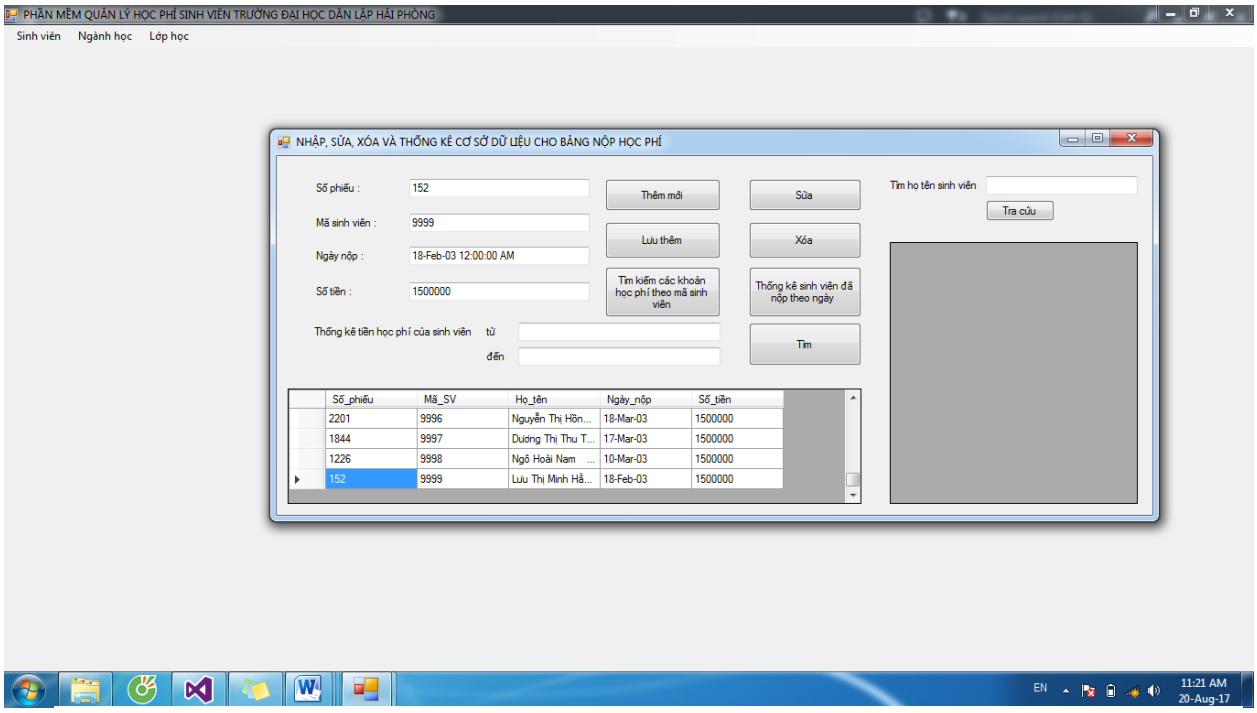
Hình 3.17 Tra cứu tên sinh viên

❖ Giao diện Thêm, sửa, xóa, tìm kiếm và thống kê nộp học phí



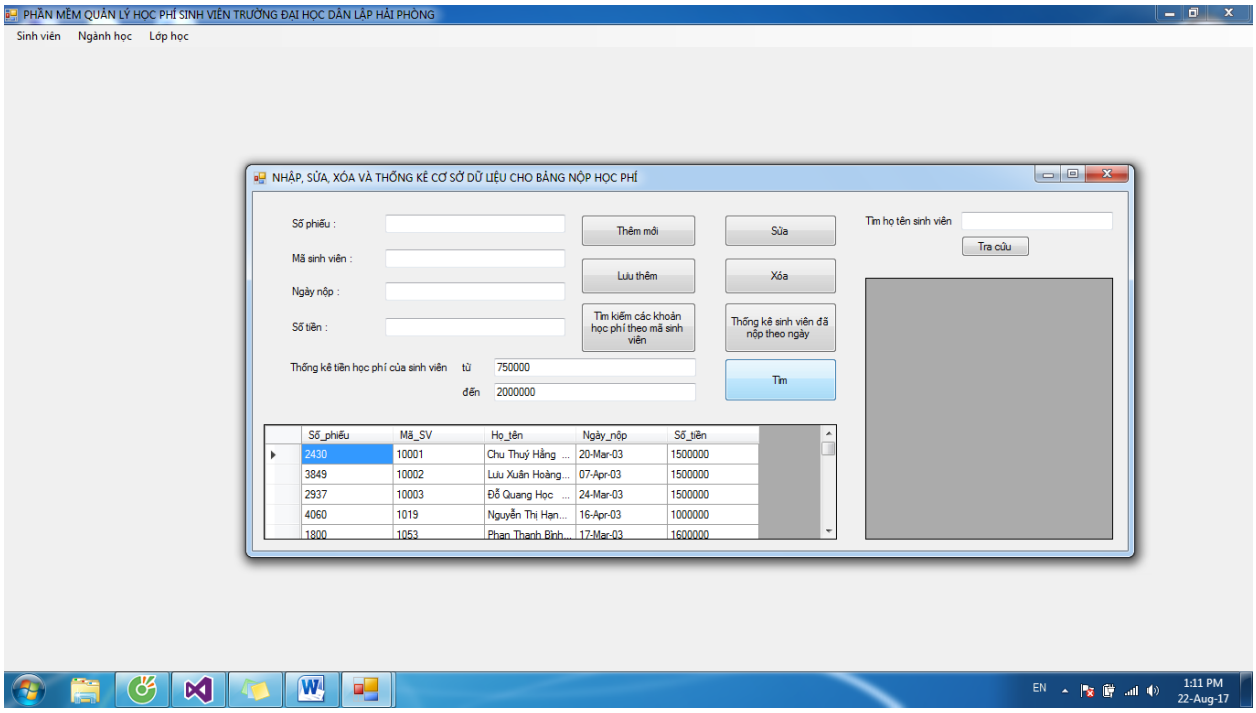
Hình 3.18 Giao diện Thêm, sửa, xóa, tìm kiếm và thống kê nộp học phí

❖ Cập nhật phiếu thu



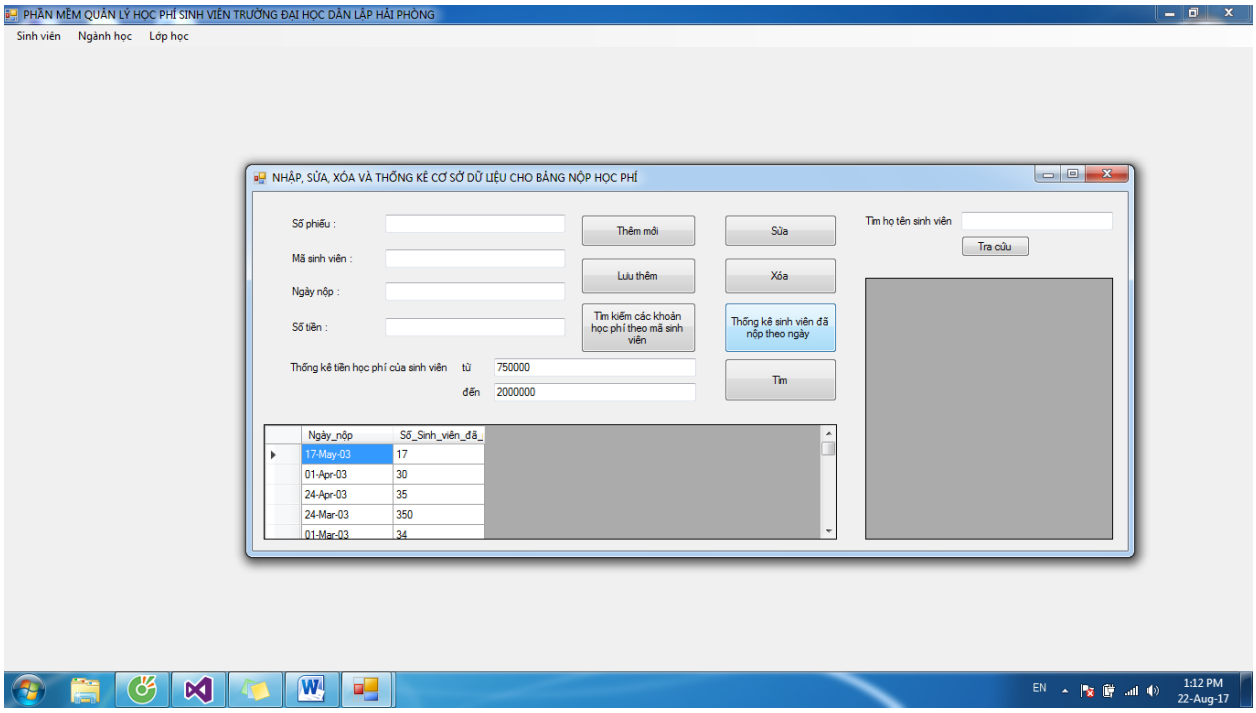
Hình 3.19 Cập nhật phiếu thu

❖ Tìm kiếm phiếu thu theo khoảng giá trị



Hình 3.20 Tìm kiếm phiếu thu theo khoảng giá trị nộp tiền

❖ Thống kê sinh viên đã nộp học phí theo ngày



Hình 3.21 Thống kê sinh viên đã nộp học phí theo ngày

3.2 Nhận xét và đánh giá

Trong chương trình này, LinQ to SQL đã giúp cho em:

- Giảm bớt các câu lệnh thực thi chương trình.
- Tùy biến các câu lệnh SELECT, INSERT, DELETE, UPDATE dễ dàng hơn.
- Rút ngắn được thời gian thực thi các câu lệnh trên SQL.

Nhìn chung, phần mềm trên đã giải quyết hầu hết các yêu cầu cũng như các chức năng đặt ra của bài toán.

Tuy nhiển, chương trình vẫn còn một số thiếu sót như:

- Chưa tối ưu hết các chức năng của Form.
- Vẫn còn một số các câu lệnh chưa được tối ưu hóa.
- Khả năng trình bày còn kém.
- Chưa mô tả đầy đủ khía cạnh của vấn đề.

KẾT LUẬN

Kết quả đạt được bao gồm:

* Lý thuyết:

- Tìm hiểu được về các khái niệm, ưu nhược điểm, đặc điểm của LinQ, LinQ to SQL.

* Chương trình:

- Phát biểu và tạo CSDL cho bài toán quản lý học phí của sinh viên trường Đại học Dân Lập Hải Phòng.
- Phần mềm " Quản lý học phí của sinh viên trường Đại học Dân lập Hải Phòng" với các chức năng: Tìm kiếm, thống kê và cập nhật dữ liệu với cơ sở dữ liệu HOCPHI tạo được.

Phần mềm được chạy thử nghiệm với một số dữ liệu chạy thông suốt và cho ra kết quả, đáp ứng được các yêu cầu đặt ra của bài toán. Vì thời gian có hạn và kinh nghiệm thực tế chưa nhiều nên việc xây dựng phần mềm: " Quản lý học phí của sinh viên trường Đại học Dân lập Hải Phòng" về cơ bản đã thực hiện tương đối đầy đủ, tuy nhiên chưa thể mô tả đầy đủ mọi khía cạnh của vấn đề. Xây dựng được phần mềm nhưng chỉ với các chức năng chính, cần thêm nhiều chức năng để phần mềm hoạt động tốt và hoàn chỉnh hơn. Nếu có điều kiện, sau này em sẽ hoàn thiện phần mềm được đầy đủ hơn và dễ dàng hơn trong việc quản trị.

TÀI LIỆU THAM KHẢO

[1]. http://www.sqltolinq.com/

[2]. https://www.tutorialspoint.com/linq/linq_sql.htm

[3]. https://namdh.wordpress.com/2008/12/04/linq-book/

[4]. https://goo.gl/j3wmFf