ỦY BAN NHÂN DÂN THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG CAO ĐẲNG GIAO THÔNG VẬN TẢI

GIÁO TRÌNH

LẬP TRÌNH CƠ SỞ DỮ LIỆU CƠ BẢN

NGÀNH: CÔNG NGHỆ THÔNG TIN

TRÌNH ĐỘ: CAO ĐẲNG

(Ban hành kèm theo Quyết định số: /QĐ-TCĐGTVT ngày ... tháng ... năm ... của .....)

Lưu hành nội bộ - Năm 2021

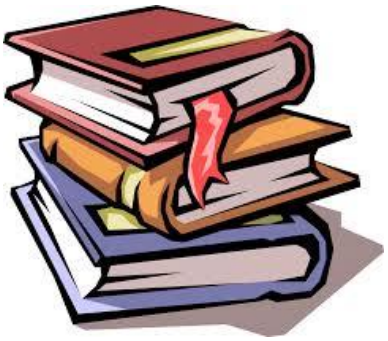
ỦY BAN NHÂN DÂN THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG CAO ĐẲNG GIAO THÔNG VẬN TẢI

GIÁO TRÌNH

LẬP TRÌNH CƠ SỞ DỮ LIỆU CƠ BẢN

NGÀNH: CÔNG NGHỆ THÔNG TIN
TRÌNH ĐỘ: CAO ĐẲNG

Chủ biên: TS. Bùi Đức Minh
Thành viên: Ths. Nguyễn Tấn Thành
CN. Lưu Hữu Phước
CN. Tăng Quốc Cường
CN. Bùi Minh An



Lưu hành nội bộ - Năm 2021

TUYÊN BỐ BẢN QUYỀN

LỜI NÓI ĐẦU

Lập trình Cơ sở dữ liệu cơ bản là một trong những học phần bắt buộc thuộc kiến thức chuyên ngành công nghệ thông tin. Môn học sẽ cung cấp cho sinh viên một số nội dung cơ bản về lập trình ứng dụng Cơ sở dữ liệu với ngôn ngữ C#.

Giáo trình "Lập trình Cơ sở dữ liệu cơ bản" bao gồm 4 chương sau:

Chương 1: TỔNG QUAN VỀ LẬP TRÌNH CƠ SỞ DỮ LIỆU. Trình bày ý nghĩa các thành phần trong kiến trúc ADO.NET và đặc điểm, cách sử dụng các đối tượng cơ sở trong kiến trúc này. Trình bày ý nghĩa và cách phân biệt các mô hình kết nối và ngắt kết nối.

Chương 2: LẬP TRÌNH VỚI CÁC ĐỐI TƯỢNG CƠ BẢN TRONG MÔ HÌNH KẾT NỐI. Trình bày ý nghĩa, cách sử dụng các đối tượng Connection, Command, DataReader, Parameter và cách lập trình, thao tác với các đối tượng này để truy xuất và cập nhật dữ liệu trong cơ sở dữ liệu.

Chương 3: LẬP TRÌNH VỚI CÁC ĐỐI TƯỢNG TRONG MÔ HÌNH NGẮT KẾT NỐI. Trình bày ý nghĩa, đặc điểm và phân biệt các loại DataSet cùng với các thuộc tính, phương thức để thao tác với các loại đối tượng trong một DataSet. Trình bày ý nghĩa, công dụng của đối tượng DataAdapter trong mô hình ngắt kết nối cùng với các điều khiển hiển thị và liên kết dữ liệu để xây dựng ứng dụng.

Chương 4: THIẾT KẾ BÁO CÁO. Trình bày ý nghĩa, vai trò của báo cáo trong xây dựng ứng dụng. Trình bày ý nghĩa các thành phần của báo cáo và kỹ thuật thiết kế các loại báo cáo với các điều khiển liên quan.

Mặc dù có rất nhiều cố gắng trong công tác biên soạn, nhưng chắc chắn không tránh khỏi những thiếu sót. Rất mong nhận được các ý kiến đóng góp của quý độc giả để giáo trình được hoàn thiện hơn.

TP. Hồ Chí Minh – Năm 2021

CÁC TÁC GIẢ

Tên môn học: LẬP TRÌNH CƠ SỞ DỮ LIỆU CƠ BẢN

Mã số môn học:

Mục tiêu môn học

Mục tiêu của môn học là trang bị cho sinh viên những nội dung sau:

- **Về kiến thức**: Sinh viên có khả năng trình bày được các khái niệm cơ bản về kiến trúc ADO.NET, giải thích được ý nghĩa của các thành phần trong mô hình này và cách thức truy xuất cơ sở dữ liệu.
- **Về kỹ năng:** sinh viên xây dựng được các ứng dụng thông qua bài tập với các kỹ năng lập trình đối tượng, truy xuất dữ liệu với các hệ quản trị cơ sở dữ liệu thông dụng bằng ngôn ngữ C# và thiết kế được các báo cáo để xem, in ấn dữ liệu.
- **Về năng lực tự chủ và trách nhiệm:** Sinh viên rèn luyện được tư duy nghiên cứu và phân tích bài toán, có khả năng nghiên cứu khoa học, yêu thích môn học, ngành học, đào sâu khả năng tự tìm hiểu mở rộng kiến thức dựa trên các nội dung cơ bản đã được cung cấp. Có tinh thần hợp tác tốt, thái độ nghiêm túc trong học tập, nghiên cứu, làm việc nhóm, tích cực tham gia các hoạt động trên lớp.

MỤC LỤC

CHƯƠNG 1. TỔNG QUAN VỀ LẬP TRÌNH CSDL VỚI C# _____9

1.1. Kiến trúc ADO.NET 9

1.1.1. .NET Framework Data Provider 9

1.1.2. DataSet 10

1.2. Các đối tượng cơ sở trong ADO.NET 11

1.2.1. DataSet: 11

1.2.2. Đối tượng DataRelation và tập hợp Relations 13

1.2.3. DataTable: 14

1.2.4. DataColumn và tập hợp Columns: 16

1.2.5. DataView và DataRowView 19

1.3. Các mô hình kết nối và ngắt kết nối trong ADO.NET 19

1.3.1. Mô hình kết nối (Connected) 20

1.3.2. Mô hình ngắt kết nối (Disconnected) 20

CHƯƠNG 2. LẬP TRÌNH VỚI CÁC ĐỐI TƯỢNG CƠ BẢN THEO MÔ HÌNH KẾT NỐI _____26

2.1. Đối tượng Connection 26

2.1.1. Khái niệm 26

2.1.2. Data Provider 26

2.1.3. Thiết lập kết nối 27

2.1.4. Chuỗi kết nối 28

2.1.5. Các thuộc tính khác của đối tượng kết nối (Connection) 29

2.1.6. Các phương thức thường dùng với kết nối 29

2.2. Đối tượng Command 30

2.2.1. Khái niệm 30

2.2.2. Tạo Command 30

2.2.3. Các thuộc tính thường dùng của Command: 30

2.2.4. Thực thi Command 31

2.3. DataReader 32
2.3.1. Khái niệm *32*
2.3.2. Các phương thức thường dùng của đối tượng DataReader: *32*
2.4. Đối tượng Parameter 33
2.4.1. Khái niệm *33*
2.4.2. Khai báo Parameter *33*
2.4.3. Các thuộc tính thường dùng: *34*
2.4.4. Truyền tham số cho Command *34*
CHƯƠNG 3. LẬP TRÌNH VỚI CÁC ĐỐI TƯỢNG THEO MÔ HÌNH NGẮT KẾT NỐI ____ 42
3.1. Tổng quan và phân loại Dataset 42
3.1.1. Khái niệm *42*
3.1.2. Phân loại Dataset *42*
3.2. Các thành phần trong DataSet 44
3.2.1. Với DataSet không định kiểu *44*
3.2.2. Với DataSet có định kiểu *46*
3.3. DataAdapter 48
3.3.1. Khái niệm *48*
3.3.2. Khởi tạo DataAdapter *49*
3.3.3. Tạo bộ lệnh cập nhật cho DataAdapter *49*
3.3.4. Một số các phương thức thường dùng của DataAdapter: *50*
3.4. Các điều khiển hiển thị và liên kết dữ liệu 52
3.4.1. Lớp BindingSource *52*
3.4.2. BindingNavigator *56*
3.4.3. Đối tượng Binding *56*
3.4.4. Tập hợp DataBindings *58*
3.4.5. Đối tượng DataGridView *59*
CHƯƠNG 4. THIẾT KẾ BÁO CÁO ____ 77
4.1. Tổng quan về thiết kế báo cáo 77

4.1.1. Khái niệm *77*
4.1.2. Các loại báo cáo thường dùng *77*
4.1.3. Thiết kế báo cáo *79*
4.1.4. Các điều khiển thường dùng trong thiết kế báo cáo *82*
4.1.5. Thiết lập biểu thức trong báo cáo *82*
4.1.6. Định dạng điều khiển Textbox trong báo cáo: *83*
4.2. Các thành phần trong một báo cáo 84
4.3. Thiết kế các loại báo cáo 84
4.3.1. Thiết kế báo cáo dạng cột (Columna) *84*
4.3.2. Thiết kế báo cáo dạng bảng (Tabular) *84*
4.3.3. Thiết kế báo cáo dạng phân nhóm *85*
4.3.4. Thiết kế báo cáo dạng Main-Sub *87*
4.4. Thiết kế màn hình và lập trình kết xuất báo cáo 89
4.4.1. Thiết kế màn hình *89*
4.4.2. Lập trình kết xuất và hiển thị báo cáo *90*

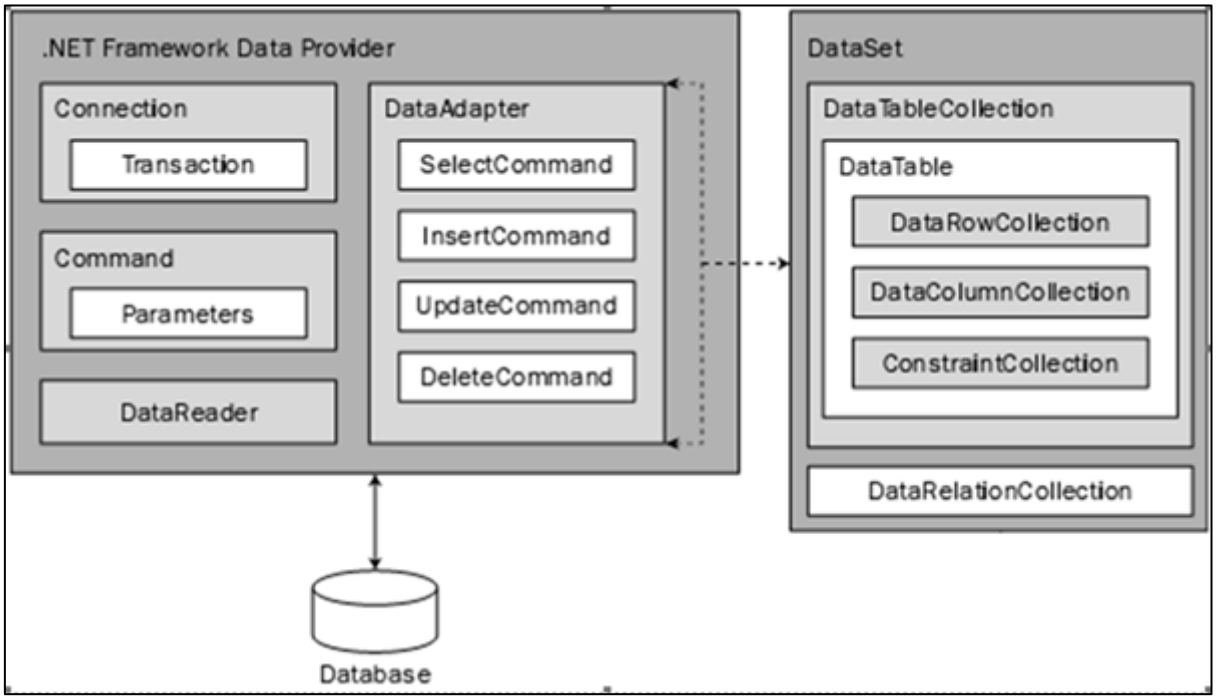
CHƯƠNG 1. TỔNG QUAN VỀ LẬP TRÌNH CSDL VỚI C#

Sau khi học xong chương này, sinh viên có khả năng:

- *Trình bày được ý nghĩa của các thành phần trong kiến trúc ADO.NET.*
- *Trình bày được ý nghĩa, công dụng của các đối tượng cơ sở trong ADO.NET.*
- *Trình bày được ý nghĩa và phân biệt được các mô hình kết nối (Connected) và ngắt kết nối (DisConnected).*
- *Thực hiện được các thao tác cơ bản với đối tượng cơ sở.*

1.1. Kiến trúc ADO.NET

ADO.NET là một thành phần của .NET Framework bao gồm bộ thư viện các lớp (class) được sử dụng để kết nối và tương tác dữ liệu với ứng dụng.



Hình 1.1. Kiến trúc ADO.NET

Kiến trúc ADO.NET có thể chia thành 2 phần chính:

1.1.1. .NET Framework Data Provider

Thành phần này chứa các đối tượng giữ nhiệm vụ kết nối và thao tác với cơ sở dữ liệu (CSDL). Một số các đối tượng tiêu biểu trong thành phần này bao gồm:

- **Connection:** dùng để tạo kết nối đến CSDL.

- **Command:** dùng để truy xuất, thao tác trên dữ liệu. Thuộc tính Connection của đối tượng Command chỉ ra kết nối với CSDL muốn thao tác.
- **DataReader:** được dùng để truy xuất trực tiếp dữ liệu từ một CSDL, có tác dụng giúp truy cập nhanh dữ liệu, nhưng hạn chế của đối tượng này là chỉ truy cập dữ liệu theo một chiều và dữ liệu chỉ có thể đọc. Đối tượng DataReader được tạo khi gọi thực hiện phương thức ExcuteReader của đối tượng Command.
- **DataAdapter:** là đối tượng làm cầu nối giữa đối tượng DataSet trong bộ nhớ và CSDL, được sử dụng để thực thi một đối tượng Command với mục đích truy xuất dữ liệu từ CSDL và chuyển dữ liệu vào các đối tượng tương ứng trong bộ nhớ. Ngoài ra, DataAdapter còn có công dụng dùng để cập nhật dữ liệu đã thay đổi từ các đối tượng trong bộ nhớ về CSDL nguồn.

Mỗi hệ quản trị CSDL sẽ tương ứng với một .NET Framework Data Provider trong môi trường .NET. Một số lớp thông dụng thường sử dụng:

+ **.NET Framework Data Provider cho Access:** cung cấp các lớp: OLEDBConnection, OLEDBCommand, OLEParameter, OLEDBDataReader, OLEDBAdapter đặt trong không gian tên System.Data.OLEDB.
+ **.NET Framework Data Provider cho SQL Sever:** cung cấp các lớp: SQLConnection, SQLCommand, SQLDataReader, … đặt trong không gian tên System.Data.SQLClient.

1.1.2. DataSet

Đối tượng có kiểu Dataset được xem như một CSDL thu nhỏ trong bộ nhớ. Dataset cung cấp một số lớp đối tượng như:

- **DataTableCollection:** là tập hợp các DataTable chứa trong DataSet.
- **DataTable:** Mỗi bảng trong CSDL được ánh xạ đến một đối tượng có kiểu DataTable được lưu trữ trong bộ nhớ, bao gồm các thành phần:
  + **DataRowCollection:** là tập hợp các dòng trong DataTable. Một dòng trong DataTable được biểu diễn bởi một đối tượng kiểu DataRow.
  + **DataColumnCollection:** là tập hợp các cột trong DataTable. Mỗi cột trong DataTable được biểu diễn bởi một đối tượng kiểu DataColumn.

+ **DataConstraintCollection:** là tập hợp ràng buộc trong DataTable. Mỗi ràng buộc được biểu diễn bởi một đối tượng kiểu Constraint.

– **DataRelationCollection:** là tập hợp các quan hệ giữa các DataTable trong DatasSet. Mỗi quan hệ được biểu diễn bởi một đối tượng DataRelation. Với DataRelation, người lập trình có thể:

+ Định nghĩa mối quan hệ giữa các bảng.
+ Duyệt dữ liệu giữa các bảng theo kiểu chính – phụ (Main-Sub).

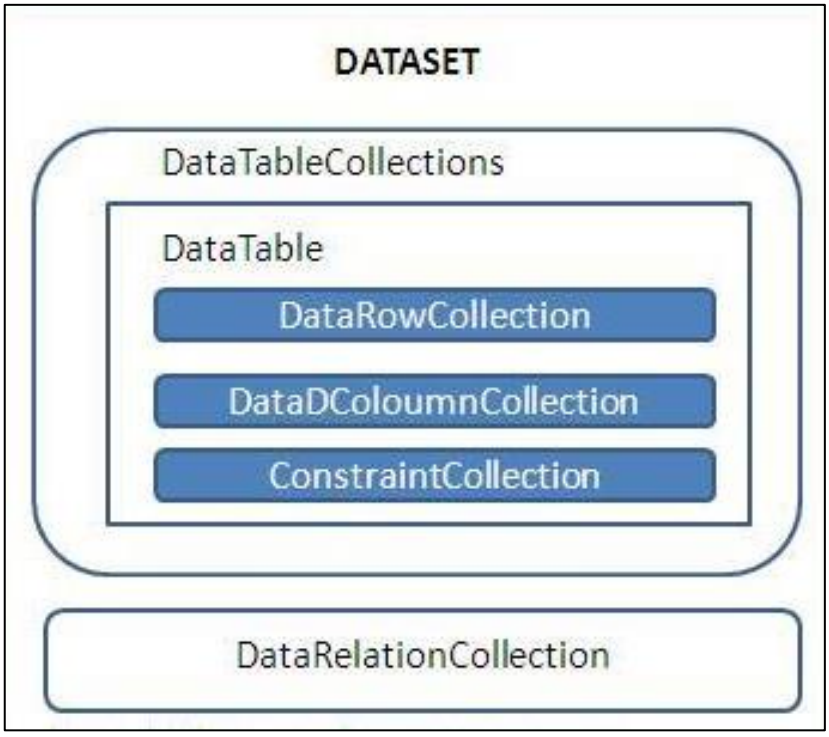
– **DataView:** là đối tượng được định nghĩa từ một DataTable với các khả năng xử lý nâng cao như thực hiện các thao tác lọc, tìm kiếm và sắp xếp dữ liệu khác nhau.

1.2. Các đối tượng cơ sở trong ADO.NET

ADO.NET cung cấp các lớp cơ sở dùng để truy xuất CSDL cho các ngôn ngữ lập trình trong môi trường .NET. Các lớp trong ADO.NET đều được đặt trong các không gian tên như: System.Data, System.Data.OleDb, System.Data.SQLClient.

1.2.1. DataSet:

– **Khái niệm:** Một trong những điểm mới trong kiến trúc ADO.NET là thành phần DataSet. Đây là thành phần được xem như CSDL quan hệ thu nhỏ được lưu trữ trong bộ nhớ để đáp ứng yêu cầu xử lý dữ liệu của ứng dụng.



Hình 1.2 Kiến trúc DataSet

– **Khai báo, khởi tạo DataSet:** DataSet <Tên biến> = New DataSet();

Thí dụ 1.1

```
Dataset ds = new Dataset();
```

– **Các thuộc tính của DataSet:**

+ **Relations:** tham chiếu đến tập hợp các quan hệ của DataSet.
+ **Tables:** tham chiếu đến tập hợp các DataTable của DataSet.

– **Các phương thức của DataSet:**

+ **Các phương thức trên tập hợp Tables của DataSet**

Add: Thêm một DataTable vào DataSet theo các cú pháp sau:

Một DataTable tự động được thêm vào DataSet với tên Table1, Table2, ….

```
<Biến DataSet>.Tables.Add();
```

Một DataTable mới được thêm vào DataSet với tên là **Tên DataTable**

```
<Biến DataSet>.Tables.Add("<Tên DataTable>");
```

Thêm một đối tượng kiểu DataTable **tblBang** vào DataSet.

```
<Biến DataSet>.Tables.Add(tblBang):
```

Thí dụ 1.2

```
using System.Data;
.....
DataSet dst = New DataSet();
DataTable dtb = New DataTable("Bảng 1");
dst.Tables.add(dtb)
```

Remove: Xóa DataTable khỏi DataSet theo cú pháp sau:

```
Tables.Remove(<DataTable>);
```

+ **Các phương thức trên đối tượng DataSet**

HasChange(): Phương thức này kiểm tra sự thay đổi của tất cả các dòng dữ liệu trên tất cả các bảng của DataSet khi có sự cập nhật dữ liệu và trả về giá trị **True** nếu có thay đổi, ngược lại trả về **False**.

GetChanges: Phương thức này dùng để lấy ra các dòng dữ liệu đã có sự thay đổi trên DataSet. Giá trị trả về của phương thức này là một bản sao của DataSet chứa những dòng dữ liệu bị thay đổi trên các DataTable.

Thí dụ 1.3

```
...
DataSet ds ;
If (dts.HasChanges())
 ds=dts.GetChanges();
...
```

AcceptChanges(): là phương thức cập nhật các thay đổi trên DataSet kể từ lúc DataSet được tải lên hoặc từ lần gọi phương thức AcceptChanges trước đó.

RejectChanges: là phương thức hủy bỏ các thay đổi và phục hồi lại dữ liệu từ lúc DataSet được tải lên hay từ lần gọi phương thức AcceptChanges trước đó.

1.2.2. Đối tượng DataRelation và tập hợp Relations

- **Khái niệm:** Tương tự như quan hệ giữa các bảng trong CSDL, các DataTable trong DataSet cũng có mối quan hệ với nhau và được biểu diễn thông qua đối tượng DataRelation. Tập hợp các quan hệ giữa các DataTable trong DataSet được biểu diễn thông qua tập hợp Relations.
- **Khởi tạo đối tượng DataRelation**

```
DataRelation <Tên quan hệ> = new DataRelation(<"Tên quan hệ">,
 <DataColumn trên DataTable 1>, <DataColumn trên DataTable N>
                          ,true | false);
```

- **Các thuộc tính thường dùng của DataRelation:**
  - **ChildColumns:** trả về các DataColumn trong DataTable đầu nhiều tham gia vào quan hệ.
  - **ChildTable:** trả về DataTable đầu nhiều trong quan hệ.
  - **DataSet:** trả về đối tượng DataSet chứa quan hệ.
  - **ParentColumns:** trả về các DataColumn trong DataTable đầu một tham gia vào quan hệ.
  - **ParentTable:** trả về DataTable đầu một trong quan hệ.
  - **RelationName:** trả về tên quan hệ.
- **Các phương thức thường dùng với tập hợp Relations**
  - **Add:** sử dụng để thêm một đối tượng DataRelation vào tập hợp.

```
Relations.Add(<"Tên quan hệ">,<DataColumn trên DataTable 1>,
        <DataColumn trên DataTable N>,true | false);
```

Thí dụ 1.4

```
...
DataRelation rel;
rel = New DataRelation("relkhoa_sv", dtsQLSV.Tables("khoa").
Columns("makh"),dtsQLSV.Tables("sinhvien").Columns("makh"));
dtsQLSV.Relations.Add(rel);
...
```

+ **Remove:** sử dụng để xóa một đối tượng DataRelation ra khỏi tập hợp.

```
Relations.Remove(<Quan hệ>)
```

Thí dụ 1.5

```
...
DataRelation rel;
rel = New DataRelation("relkhoa_sv", dtsQLSV.Tables("khoa").
Columns("makh"),dtsQLSV.Tables("sinhvien").Columns("makh"));
dtsQLSV.Relations.Add(rel);
...
dtsQLSV.Relations.Remove(rel);
...
```

1.2.3. DataTable:

– **Khái niệm:** là đối tượng được sử dụng để lưu trữ dữ liệu đọc được từ nguồn dữ liệu. Trong đối tượng DataTable có các đối tượng thuộc lớp DataColumn tương ứng với các cột và DataRow tương ứng với các dòng trong một bảng dữ liệu. Lớp DataTable thuộc không gian tên System.Data.

– **Khai báo và khởi tạo đối tượng DataTable:**

+ **Cách 1:**

```
DataTable <Tên đối tượng> = New DataTable()
```

+ **Cách 2:**

```
DataTable <Tên đối tượng> = New DataTable("<Tên bảng>")
```

– **Các thuộc tính thường dùng:**

+ **Columns:** tập hợp các cột thuộc lớp DataColumn trong DataTable.
+ **DefaultView:** trả về đối tượng DataView tương ứng với DataTable.
+ **PrimaryKey:** gán hoặc trả về mảng các cột là khóa chính DataTable.

+ **Rows:** tập hợp các dòng dữ liệu của DataTable.
+ **TableName:** Tên của DataTable.

– **Các phương thức thường dùng**

+ **NewRow:** phát sinh một dòng mới theo cấu trúc của DataTable.
+ **Clear:** xóa tất cả các dòng dữ liệu trong DataTable.
+ **GetChange:** lấy ra bảng sao những dòng thay đổi trong DataTable.
+ **AcceptChanges:** cập nhật các thay đổi kể từ lần cập nhật trước của DataTable. Sau khi thực hiện xong, tình trạng của các dòng là UnChanged. Những dòng có tình trạng là Deleted bị loại bỏ khỏi bảng.
+ **RejectChanges():** hủy bỏ các thay đổi của DataTable, phục hồi lại các giá trị kể từ lần cập nhật trước. Sau khi phương thức này thực hiện, các dòng thêm mới vào DataTable bị loại bỏ, những dòng có tình trạng Deleted hay Modified được phục hồi lại tình trạng gốc.
+ **Compute:** thực hiện tính toán, thống kê trên những dòng thỏa điều kiện trong DataTable. Các phép toán thống kê bao gồm: sum (tính tổng), count (đếm), max (tính giá trị lớn nhất), min(tính giá trị nhỏ nhất) và avg (tính giá trị trung bình).

Cú pháp

```
<Tên DataTable>.Compute(<"Biểu thức">[,<Biểu thức điều kiện>]);
```

Thí dụ 1.6

```
…
tblSV = New DataTable();
…
int TongHB = (int)tblSV.Compute("sum(HocBong)", "MaKH='TH'");
…
```

+ **Select:** trả về mảng các dòng trong DataTable thỏa điều kiện (nếu có).

Cú pháp

```
<Tên DataTable>.Select([<"Biểu thức điều kiện"]>);
```

Thí dụ 1.7

```
…
tblSV = New DataTable();
SINHVIEN[] SVTH = tblSV.Select("MaKH='TH'");
…
```

1.2.4. DataColumn và tập hợp Columns:

- **Khái niệm:** Mỗi DataTable được tạo bởi các cột, mỗi cột tương ứng với một đối tượng thuộc lớp DataColumn tương ứng với một thuộc tính có kiểu dữ liệu cụ thể. Lớp DataColumn thuộc không gian tên System.Data. Tập hợp các đối tượng DataColumn được chứa trong tập hợp Columns, mọi tham chiếu đến một DataColum đều phải thông qua tập hợp này.
- **Các thuộc tính thường dùng của DataColumn:**
  - **ColumnName:** tên của DataColumn.
  - **DataType:** kiểu dữ liệu của DataColumn.
  - **DefaultValue:** giá trị mặc định của DataColumn khi thêm dòng mới.
  - **Expression:** Biểu thức tính toán giá trị cho DataColumn.
- **Khai báo và khởi tạo DataColumn:**
  - **Cách 1**

```
DataColumn <Tên DataColumn> = New DataColumn();
```

  - **Cách 2**

```
DataColumn <đối tượng> = New DataColumn("<Tên>", <Kiểu dữ liệu>);
```

Thí dụ 1.8

```
…
DataColumn colMaKH = New DataColumn("MaKH",GetType(string));
…
```

- **Một số thuộc tính và phương thức thường dùng của tập hợp Columns:**
  - **Count:** trả về số cột trong tập hợp.
  - **Columns(<Tên cột | Chỉ số>):** tham chiếu đến một DataColumn.
  - **Add:** thêm một DataColumn mới vào tập hợp Columns như sau:

Cách 1: Thêm một DataColumn vào tập hợp Columns

```
Columns.Add(<DataColumn>);
```

Thí dụ 1.9

```
…
DataColumn colMaKH = New DataColumn("MaKH",GetType(string));
tblKhoa.Columns.Add(colMaKH);
…
```

Cách 2: Thêm DataColum với tên và kiểu dữ liệu vào tập hợp Columns.

```
Columns.Add(<Tên cột>, <kiểu dữ liệu>);
```

Thí dụ 1.10

```
…
tblKhoa.Columns.Add("MaKH",GetType(string));
…
```

+ **Remove:** xóa một DataColumn ra khỏi tập hợp Columns.

Thí dụ 1.11

```
…
tblKhoa.Columns.Remove("MaKH");
…
```

– **DataRow và tập hợp Rows**

+ **Khái niệm:** Dữ liệu lưu trữ trong DataTable với các đối tượng thuộc lớp DataRow. Một đối tượng DataRow được tham chiếu thông qua tập hợp Rows của DataTable. Lớp DataRow thuộc về không gian tên System.Data.
+ **Các thuộc tính thường dùng của DataRow:**

  Item: tham chiếu dữ liệu của một DataColum cụ thể.

  ItemArray: tham chiếu nội dung của DataRow dưới dạng một mảng.
+ **Các phương thức thường dùng của DataRow:**

  BeginEdit: Thiết lập dòng ở chế độ hiệu chỉnh dữ liệu.

  CancelEdit: Hủy bỏ việc hiệu chỉnh dữ liệu trên dòng.

  Delete: Đánh dấu hủy dòng.

  EndEdit: Chấm dứt việc hiệu chỉnh dữ liệu trên dòng.

  RejectChanges: hủy bỏ sự thay đổi trên dòng dữ liệu từ lần cập nhật trước đó.
+ **Một số thuộc tính và phương thức thường dùng của tập hợp Rows:**

  Count: trả về số dòng có trong tập hợp.

  Rows(<chỉ số>): tham chiếu đến dòng có **chỉ số** trong tập hợp.

  Add(<DataRow>): thêm một dòng vào DataTable.

  Remove(<DataRow>): xóa một dòng ra khỏi DataTable.

  Clear: xóa toàn bộ các dòng trong một DataTable.

Contains(<giá trị>): kiểm tra trên thuộc tính là khóa chính có dòng chứa **giá trị** không? Giá trị trả về là True (nếu có), False (không có)

Find(<giá trị>): trả về dòng chứa **giá trị** trên thuộc tính là khóa chính.

- **Constraint và tập hợp Constraint:**
  - **Khái niệm:** Constraint dùng để thiết lập các ràng buộc áp dụng cho các DataColumn để bảo đảm tính toàn vẹn dữ liệu trên DataTable. Constraint thuộc không gian tên System.Data. Có hai loại ràng buộc thường dùng là ForeignKeyConstraint tương ứng với ràng buộc khóa ngoại và UniqueConstraint tương ứng với ràng buộc duy nhất.
  - **ForeignKeyConstraint:** Cú pháp khai báo và khởi tạo ràng buộc:

Cách 1:

```
New ForeignKeyConstraint(<Tên>, <DataColumn 1>, <DataColumn N>);
```

Cách 2:

```
New ForeignKeyConstraint(<Tên>, <Mảng DataColumn 1>, <Mảng
                        DataColumn N>);
```

  - **Các thuộc tính của ForeignKeyConstraint:**

AcceptRejectRule: cách xử lý khi phương thức AcceptChanges trên DataTable đầu 1 thực thi.

DeleteRule: cách xử lý khi DataRow trên DataTable đầu 1 bị hủy.

UpdateRule: cách xử lý khi DataRow trên DataTable đầu 1 thay đổi.

  - **UniqueConstraint:** Cú pháp khai báo và khởi tạo ràng buộc:

Cách 1:

```
New UniqueConstraint (<Tên>, <DataColumn>);
```

Cách 2:

```
New UniqueConstraint(<Tên>, <Mảng DataColumn>);
```

  - **Các thuộc tính của UniqueConstraint:**

Columns: mảng các cột tham gia khóa duy nhất.

IsPrimaryKey: khóa duy nhất có đồng thời là khóa chính không?

  - **Tập hợp Constraints:** chứa tập hợp các Constraint trên DataTable. Tương tự như các đối tượng khác, các phương thức Add, Remove được sử dụng để thêm và xóa các ràng buộc trong DataTable.

1.2.5. DataView và DataRowView

– **DataView:** Khi thao tác với đối tượng DataTable, một số các chức năng như tìm kiếm, sắp xếp, lọc dữ liệu, ... vẫn còn một số hạn chế. Đối tượng DataView xử lý được các yêu cầu này và cung cấp cách hiển thị dữ liệu của DataTable một cách linh hoạt. Mặc định, mỗi đối tượng DataTable thì tương ứng với một đối tượng DataView thông qua việc sử dụng thuộc tính DefaultView của DataTable. DataView thuộc không gian tên System.Data.

+ **Cú pháp khai báo, khởi tạo DataView:**

Cách 1:

```
DataView <Tên DataView> = New DataView();
```

Cách 2:

```
DataView <Tên DataView> = New DataView(<DataTable>);
```

+ **Các thuộc tính thường dùng của DataView**

AllowDelete: cho phép thực hiện thao tác xóa trên DataView không?

AllowEdit: cho phép thực hiện sửa đổi trên DataView không?

AllowNew: cho phép thực hiện thêm mới trên DataView không?

RowFilter: biểu thức lọc của DataView khi cần lọc dữ liệu.

– **DataRowView:** tương tự đối tượng DataRow có trong một DataTable, DataRowView là một đối tượng tương ứng với một dòng trong DataView.

+ **Các thuộc tính của DataRowView:**

DataView: trả về đối tượng DataView chứa DataRowView.

IsEdit: cho biết phần tử đang ở trạng thái chỉnh sửa hay không?

IsNew: cho biết phần tử có phải là phần tử mới thêm vào hay không?

+ **Các phương thức thường dùng của DataRowView:**

BeginEdit: thiết lập chế độ chỉnh sửa trên DataRowView.

CancelEdit: huỷ bỏ những thay đổi trên DataRowView.

EndEdit: kết thúc chỉnh sửa và cập nhật lại DataRowView.

1.3. Các mô hình kết nối và ngắt kết nối trong ADO.NET

Để thao tác với CSDL với ADO.NET, một trong hai mô hình sau được sử dụng: mô hình kết nối (Connected) và mô hình ngắt kết nối (Disconnected).

1.3.1. Mô hình kết nối (Connected)

Với mô hình này thì một kết nối được tạo ra và duy trì trong suốt quá trình ứng dụng thực hiện các thao tác với CSDL. Mọi thay đổi xảy ra với CSDL có thể thực hiện ngay trong khi kết nối đang mở. Tuy nhiên, kỹ thuật này có nhược điểm là gây ra sự lãng phí tài nguyên do thường xuyên chiếm dụng đường truyền.

Với mô hình kết nối, các đối tượng thường sử dụng gồm: đối tượng Connection, Command và DataReader.

- **Đối tượng Connection:** sử dụng để tạo, duy trì và thực hiện đóng kết nối đến một CSDL.
- **Đối tượng Command:** sử dụng để tạo truy vấn đọc và cập nhật lại dữ liệu.
- **Đối tượng DataReader:** sử dụng để truy xuất dữ liệu chỉ đọc (một chiều).

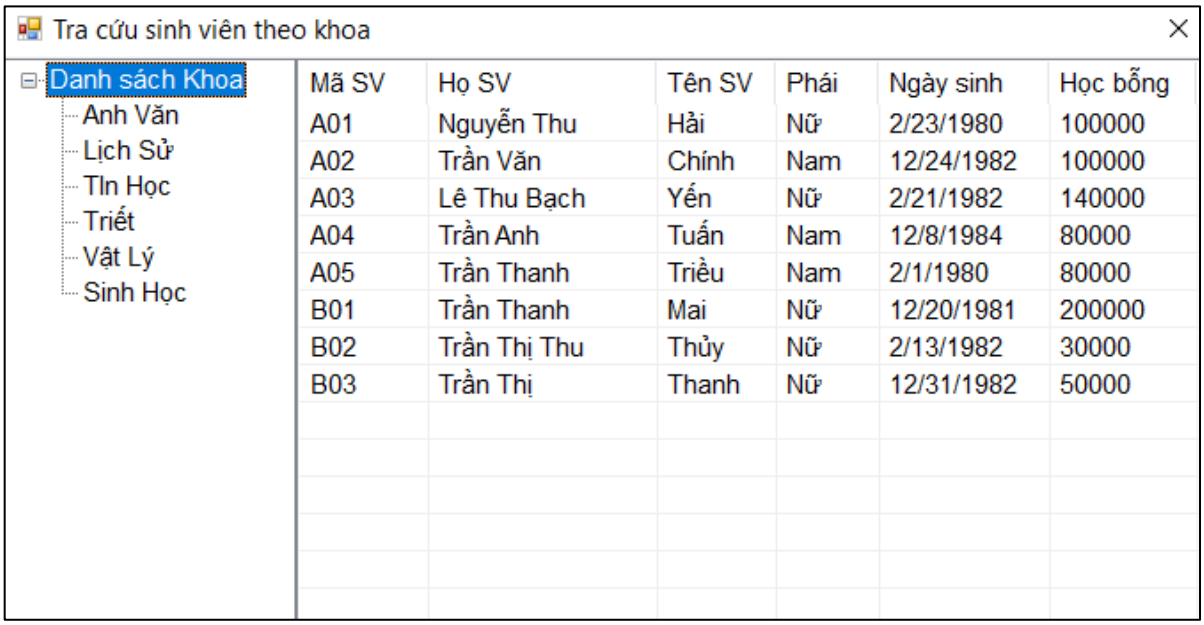
1.3.2. Mô hình ngắt kết nối (Disconnected)

Khắc phục nhược điểm của mô hình kết nối, mô hình ngắt kết nối được đề xuất với cơ chế làm việc là đọc dữ liệu từ CSDL vào trong bộ nhớ để xử lý và tạm thời ngắt kết nối đến CSDL. Ưu điểm của mô hình này là không chiếm nhiều tài nguyên đường truyền. Tuy nhiên, nhược điểm của mô hình này là phải hao tốn nhiều bộ nhớ để chứa dữ liệu đọc từ CSDL. Sau khi dữ liệu xử lý xong, nếu cần cập nhật thì có thể thiết lập lại kết nối để đồng bộ hoá sự thay đổi với CSDL gốc. Với mô hình ngắt kết nối, ngoài các đối tượng cơ bản như DataSet, DataTable, … mô hình này còn sử dụng thêm một số các đối tượng như DataAdapter, CommandBuilder, …

- **Đối tượng DataAdapter:** là đối tượng được sử dụng làm trung gian để sao chép cấu trúc và dữ liệu từ các bảng trong CSDL đưa vào các DataTable trong Dataset và cập nhật lại sự thay đổi của dữ liệu từ các DataTable trong DataSet về các bảng trong CSDL gốc ban đầu.
- **Đối tượng CommandBuilder:** là đối tượng được sử dụng để phát sinh ra các đối tượng Command tương ứng với các chức năng cập nhật dữ liệu khi thêm, sửa, xoá dữ liệu.

BÀI TẬP CHƯƠNG 1

1. Thiết kế, định dạng màn hình và các đối tượng điều khiển theo mô tả của hình sau:



Yêu cầu:

1.1. Tạo DataSet ds chứa các DataTable có cấu trúc theo mô tả sau:

+ DataTable tblKhoa có các DataColum MaKH (String), TenKH (String). Khoá chính của tblKhoa là **MaKH**.

+ DataTable tblSinhvien có các DataColum MaSV (String), HoSV (String), TenSV (String), Phai (Boolean), NgaySinh (DateTime), NoiSinh (String), MaKH (String), HocBong (int). Khoá chính của tblSinhvien là **MaSV**.

+ Tạo quan hệ khoá ngoại (Foreign key) giữa các DataTable tblKhoa và tblSinhvien thông qua DataColumn **MaKH** của các DataTable.

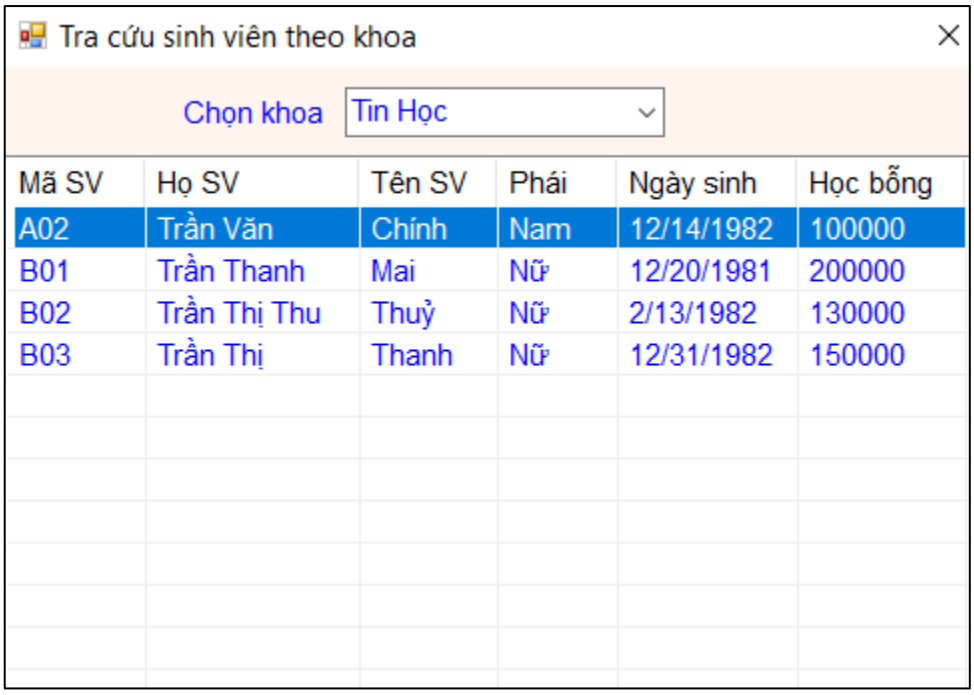
1.2. Từ các tập tin KHOA.txt và SINHVIEN.txt cho trước, với các đối tượng và phương thức, thuộc tính của DataTable, DataRow, Rows… Hãy đọc dữ liệu từ các tập tin này và thêm vào các DataTable tblKhoa, tblSinhvien tương ứng.

1.3. Đọc và nạp dữ liệu từ DataTable tblKhoa vào điều khiển Treeview.

1.4. Khi người dùng Click vào nút gốc Danh sách Khoa trên Treeview thì toàn bộ danh sách sinh viên trong DataTable tblSinhvien được hiển thị trên Listview.

1.5. Khi người dùng Click vào một khoa cụ thể trên Treeview thì Listview chỉ hiển thị danh sách các sinh viên tương ứng với Khoa được chọn.

2. Thiết kế, định dạng màn hình và các đối tượng điều khiển theo mô tả của hình sau:



Yêu cầu:

2.1. Tạo DataSet ds chứa các DataTable có cấu trúc theo mô tả sau:

+ DataTable tblKhoa có các DataColum MaKH (String), TenKH (String). Khoá chính của tblKhoa là **MaKH**.

+ DataTable tblSinhvien có các DataColum MaSV (String), HoSV (String), TenSV (String), Phai (Boolean), NgaySinh (DateTime), NoiSinh (String), MaKH (String), HocBong (int). Khoá chính của tblSinhvien là **MaSV**.

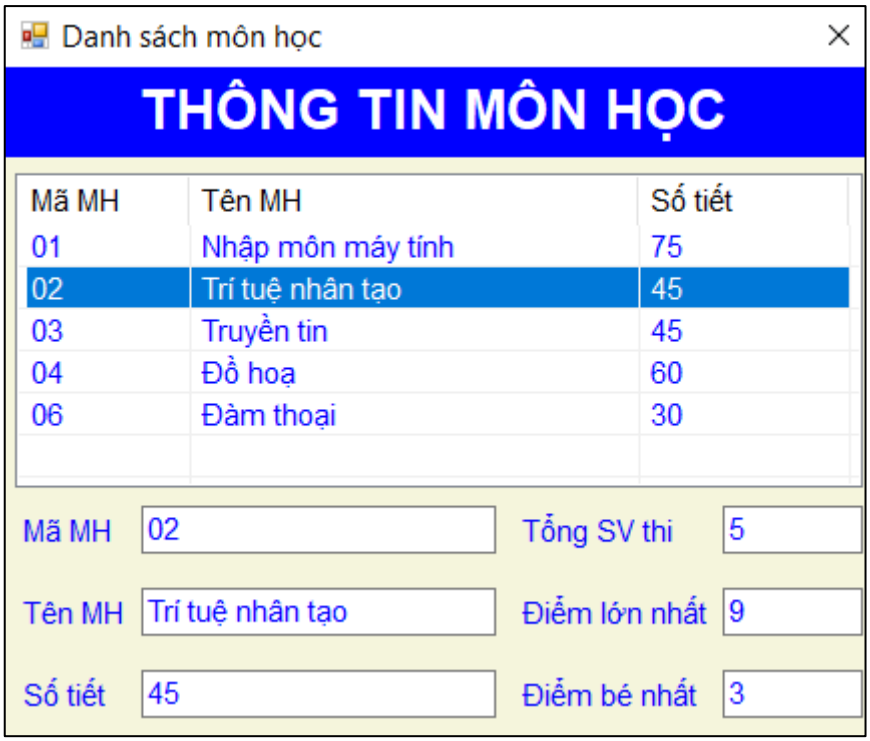
+ Tạo quan hệ khoá ngoại (Foreign key) giữa các DataTable tblKhoa và tblSinhvien thông qua DataColumn **MaKH** của các DataTable.

2.2. Từ các tập tin KHOA.txt và SINHVIEN.txt cho trước, với các đối tượng và phương thức, thuộc tính của DataTable, DataRow, Rows… Hãy đọc dữ liệu từ các tập tin này và thêm vào các DataTable tblKhoa, tblSinhvien tương ứng.

2.3. Thiết lập nguồn dữ liệu và các thuộc tính cho điều khiển Combobox.

2.5. Khi người dùng Click chọn một khoa cụ thể trên điều khiển Combobox thì Listview sẽ hiển thị danh sách các sinh viên tương ứng với Khoa được chọn.

3. Thiết kế, định dạng màn hình và các đối tượng điều khiển theo mô tả của hình sau:



Yêu cầu:

3.1. Tạo DataSet ds chứa các DataTable có cấu trúc theo mô tả sau:

+ DataTable tblMonhoc có các DataColum MaMH (String), TenMH (String). SoTiet (int). Khoá chính của tblMonhoc là **MaMH**.

+ DataTable tblKetqua có các DataColum MaSV (String), MaMH (String), Diem (Double). Khoá chính của tblKetqua là (**MaSV,MaMH)**.

+ Tạo quan hệ khoá ngoại (Foreign key) giữa các DataTable tblKetqua và tblMonhoc thông qua DataColumn **MaMH** của các DataTable.

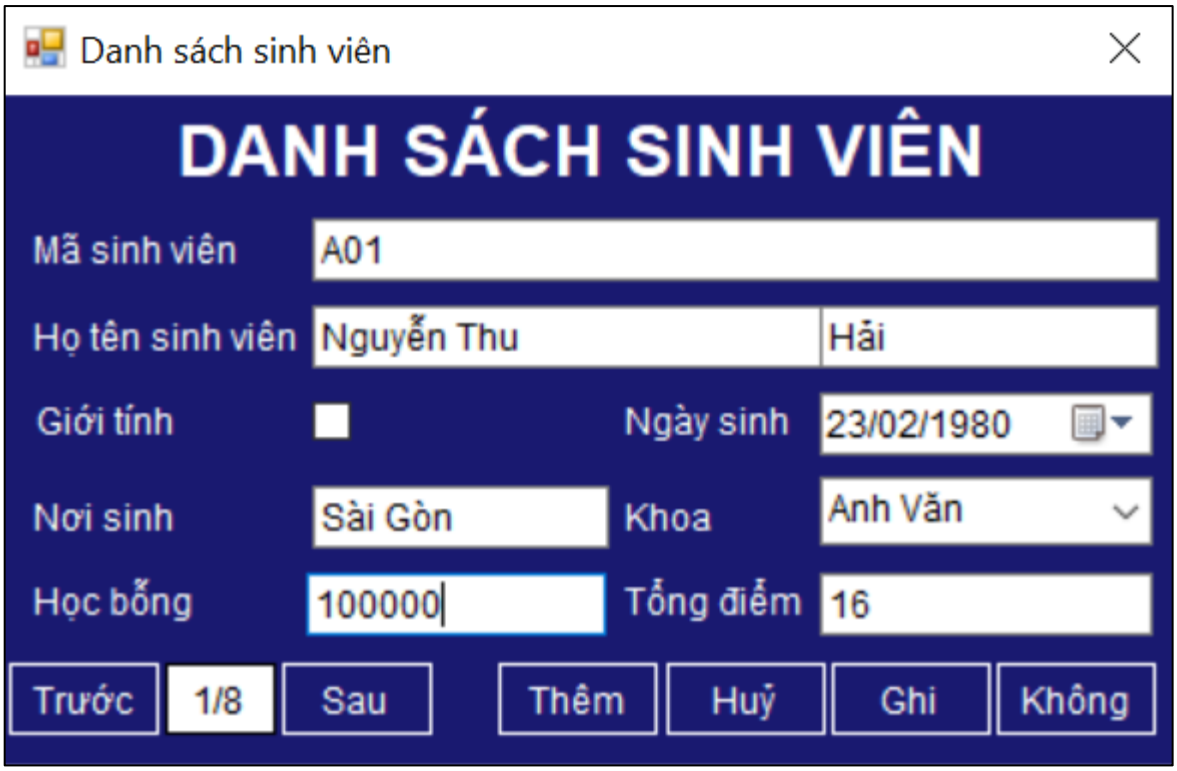
3.2. Từ các tập tin KETQUA.txt và MONHOC.txt cho trước, với các đối tượng và phương thức, thuộc tính của DataTable, DataRow, Rows… Hãy đọc dữ liệu từ các tập tin này và thêm vào các DataTable tblKetqua, tblMonhoc tương ứng.

3.3. Đọc dữ liệu từ tblMonhoc và đưa vào điều khiển Listview.

3.4. Khi người dùng Click chọn một dòng cụ thể trên điều khiển Listview thì thông tin chi tiết của môn học được chọn hiển thị trên các điều khiển Textbox.

3.5. Sử dụng phương thức Compute của DataTable để tính và hiển thị tổng sinh viên thi, điểm lớn nhất, điểm bé nhất của môn học hiện hành.

4. Thiết kế, định dạng màn hình và các đối tượng điều khiển theo mô tả của hình sau:



Yêu cầu:

4.1. Tạo DataSet ds chứa các DataTable có cấu trúc theo mô tả sau:

+ DataTable tblKhoa có các DataColum MaKH (String), TenKH (String). Khoá chính của tblKhoa là **MaKH**.

+ DataTable tblSinhvien có các DataColum MaSV (String), HoSV (String), TenSV (String), Phai (Boolean), NgaySinh (DateTime), NoiSinh (String), MaKH (String), HocBong (int). Khoá chính của tblSinhvien là **MaSV**.

+ DataTable tblMonhoc có các DataColum MaMH (String), TenMH (String). SoTiet (int). Khoá chính của tblMonhoc là **MaMH**.

+ DataTable tblKetqua có các DataColum MaSV (String), MaMH (String), Diem (Double). Khoá chính của tblKetqua là (**MaSV,MaMH)**.

+ Tạo quan hệ khoá ngoại (Foreign key) giữa các DataTable tblKhoa và tblSinhvien thông qua DataColumn **MaKH** của các DataTable.

+ Tạo quan hệ khoá ngoại (Foreign key) giữa các DataTable tblSinhvien và tblketqua thông qua DataColumn **MaSV** của các DataTable.

+ Tạo quan hệ khoá ngoại (Foreign key) giữa các DataTable tblKetqua và tblMonhoc thông qua DataColumn **MaMH** của các DataTable.

4.2. Từ các tập tin KHOA.txt,SINHVIEN.txt, KETQUA.txt và MONHOC.txt cho trước, với các đối tượng và phương thức, thuộc tính của DataTable, DataRow, Rows… Hãy đọc dữ liệu từ các tập tin này và thêm vào các DataTable tblKhoa, tblSinhvien, tblKetqua và tblMonhoc tương ứng.

4.3. Thiết lập nguồn dữ liệu từ tblKhoa và các thuộc tính đưa vào điều khiển Combobox.

4.4. Sử dụng phương thức Compute của DataTable để tính và hiển thị tổng điểm thi của sinh viên hiện hành.

4.5. Khi người dùng Click chọn các nút lệnh Trước, Sau thì cho phép di chuyển mẫu tin hiện hành về ngay mẫu tin phía trước, phía sau tương ứng. Thông báo lỗi khi không thể di chuyển được.

4.6. Khi người dùng Click chọn nút lệnh Thêm thì cho phép thêm mới một mẫu tin tương ứng với một sinh viên.

4.7. Khi người dùng Click chọn nút lệnh Huỷ thì cho phép xoá mẫu tin hiện hành trên màn hình. Yêu cầu người dùng xác nhận trước khi huỷ mẫu tin.

4.8. Khi người dùng Click chọn nút lệnh Ghi thì cho phép ghi lại sự thay đổi của mẫu tin sau khi hiệu chỉnh hay thêm mới vào DataTable tblSinhvien.

4.9. Khi người dùng Click chọn nút lệnh Không thì cho phép phục hồi lại mẫu tin hiện hành trên màn hình.

4.10. Khi thực hiện các thao tác cần kiểm tra tính hợp lý, hợp lệ tương ứng với từng thao tác thực hiện.

CHƯƠNG 2. LẬP TRÌNH VỚI CÁC ĐỐI TƯỢNG CƠ BẢN THEO MÔ HÌNH KẾT NỐI

Sau khi học xong chương này, sinh viên có khả năng:

- *Trình bày được ý nghĩa và sử dụng được cú pháp khai báo các đối tượng Connection, Command, DataReader và Parameter.*
- *Thao tác với các đối tượng Connection, Command, DataReader, Parameter để truy xuất và cập nhật dữ liệu trong CSDL.*

2.1. Đối tượng Connection

2.1.1. Khái niệm

Connection là đối tượng chịu trách nhiệm quản lý kết nối giữa ứng dụng và nguồn dữ liệu. Có hai dạng kết nối thông dụng thường dùng tương ứng với các hệ quản trị CSDL tiêu biểu như SQL Server và Access. Đối tượng kết nối nhận vào tham số là chuỗi kết nối (Connection String). Chuỗi kết nối thường sử dụng một số tham số chẳng hạn như: Data Provider, Data Source, ….

2.1.2. Data Provider

ADO.NET cung cấp một cách thức chung cho phép ứng dụng tương tác với các loại dữ liệu khác nhau. Tuy nhiên, tương ứng với mỗi loại dữ liệu thì cần có một cách thức tương ứng để có thể truy xuất đến nguồn dữ liệu. Do đó, cần sử dụng các thư viện tương ứng với cách thức truy xuất dữ liệu của một nguồn dữ liệu cụ thể. Các thư viện này được quy định bởi thành phần Data Provider. Bảng sau sẽ giới thiệu một số Data Provider thông dụng

Tên Provider	Giao thức	Nguồn dữ liệu truy xuất
ODBC Data Provider	ODBC	Truy xuất nguồn dữ liệu với giao thức ODBC. Thường sử dụng với các cơ sở dữ liệu cũ.
OleDb Data Provider	OLEDB	Truy xuất nguồn dữ liệu với giao thức OLEDB. Thường sử dụng với dữ liệu của Access, Excel.

Tên Provider	Giao thức	Nguồn dữ liệu truy xuất
Oracle Data Provider	ORACLE	Truy xuất nguồn dữ liệu từ Oracle.
SQL Data Provider	SQL	Truy xuất nguồn dữ liệu từ SQL Server.

Các đối tượng kết nối (Connection) tương ứng với Data Provider có sẵn trong ADO.NET là:

- **OLEDBConnection:** tương ứng với Data Provider OleDb.
- **SQLConnection:** tương ứng với Data Provider SQL.

2.1.3. Thiết lập kết nối

Để tạo kết nối, sử dụng đối tượng Connection với khai báo như sau:

- **Đối với OLEDB**
  - **Cú pháp**

```
OleDbConnection <Tên biến> = New OleDbConnection("<Chuỗi kết nối>");
```

Thí dụ 2.1

```
Using System.Data.OleDB;
...
String strcon;
strcon=     @"provider=Microsoft.jet.oledb.4.0;data     source     =
c:\qlsv.mdb";
OleDbConnection cnn = New OleDbConnection(strcon);
cnn.Open();
if (cnn.State == ConnectionState.Open)
   MessageBox.Show("OK");
...
```

- **Đối với SQL**
  - **Cú pháp**

```
SQLConnection  <Tên biến> = new SQLConnection ("<Chuỗi kết nối>");
```

Thí dụ 2.2

```
Using System.Data.SQLClient;
...
String strcon;
strcon= "server=.; database=qlsv; uid=sa; pwd=cntt1";
SQlConnection cnn = New SQlConnection(strcon);
cnn.Open();
if (cnn.State == ConnectionState.Open)
   MessageBox.Show ("OK");
```

2.1.4. Chuỗi kết nối

Để thực hiện kết nối, cần khai báo một số thông tin cho đối tượng Connection thông qua chuỗi kết nối. Chuỗi kết nối được khai báo tùy thuộc vào Data Provider sử dụng.

- **Đối với OleDbConnection:** Chuỗi kết nối (ConnectionString) bao gồm các thành phần:
  - **Provider:** khai báo DataProvider sử dụng.
  - **Data Source:** Khai báo đường dẫn đến tập tin chứa CSDL.

Thí dụ 2.3

```
Using System.Data.OleDb;
String StrCon = @"Provider=Microsoft.Jet.OleDb.4.0; Data Source=
C:\QLSV.Mdb";
OLEDBConnection cnn = New OLEDBConnection(StrCon);
```

Lưu ý: Với tập tin Access tạo ra từ phiên bản 2007 trở về sau thì chuỗi kết nối sẽ khai báo tương ứng với Data Provider là **Microsoft.ACE.OLEDB.12.0**.

Thí dụ 2.4

```
Using System.Data.OleDb;
String StrCon = @"Provider=Microsoft.ACE.OleDb.12.0; Data Source=
C:\QLSV.Mdb";
OLEDBConnection cnn = New OLEDBConnection(StrCon);
```

- **Đối với SQLConnection:** Chuỗi kết nối bao gồm các thành phần:
  - **Data Source (Server):** Tên máy cài đặt SQL Server (hay địa chỉ IP) chứa CSDL cần kết nối.
  - **Initial Catalog (Database):** Tên CSDL kết nối đến.

+ **Integrated Security:** Quy định cách sử dụng cơ chế đăng nhập vào SQL Server với các giá trị sau:

  True(SSPI): sử dụng cơ chế đăng nhập của Windows;

  False: sử dụng cơ chế đăng nhập của SQL Server.

+ **User ID (Uid):** Tên người dùng (Integrated Security có giá trị False)
+ **Password (Pwd):** Mật khẩu (Integrated Security có giá trị là False).

Thí dụ 2.5

```
String StrCon="Data Source=<Tên máy\SQLSERVER>; Initial Catalog=<Tên CSDL>;"
                + "Integrated Security=<True>";
SQLConnection Cnn = New SQLConnection(StrConnection);
```

Thí dụ 2.6

```
String StrCon="Server=<Tên máy\SQLSERVER>; Database=<Tên CSDL>;"
              + "UID=<Tên người sử dụng>; PWD=<Mật khẩu>";
SQLConnection Cnn = New SQLConnection(StrConnection);
```

2.1.5. Các thuộc tính khác của đối tượng kết nối (Connection)

– **State:** Trạng thái kết nối của Connection với các giá trị:

+ **Closed:** Kết nối đã đóng.
+ **Connecting:** Đang kết nối đến nguồn dữ liệu.
+ **Open:** kết nối đang được mở.

2.1.6. Các phương thức thường dùng với kết nối

– **Open:** thực hiện mở kết nối đến nguồn dữ liệu.

– **Close:** thực hiện đóng kết nối đến nguồn dữ liệu.

Thí dụ 2.7

```
Using System.Data.OleDb;
String StrCon = @"Provider=Microsoft.ACE.OleDb.12.0; Data Source=C:\QLSV.Mdb";
OLEDBConnection cnn = New OLEDBConnection(StrCon);
cnn.Open();
// Thực hiện các lệnh truy xuất cơ sở dữ liệu
Cnn.Close();
```

2.2. Đối tượng Command

2.2.1. Khái niệm

Sau khi tạo kết nối đến nguồn dữ liệu với đối tượng Connection, mọi thao tác truy vấn và cập nhật nguồn dữ liệu được thực hiện thông qua đối tượng Command. Tùy theo Data Provider, sử dụng các loại Command có sẵn trong ADO.NET. Cụ thể như: OLEDBCommand (tương ứng với OLEDB) và SQLCommand (tương ứng với CSDL đặt trên SQL Server).

2.2.2. Tạo Command

Có thể khởi tạo Command theo nhiều cách. Cách thông dụng thường dùng:

```
<Loại Command> <Tên biến> = new <Loại Command>(<Chuỗi SQL>,<Tên biến Connection>);
```

Thí dụ 2.8

```
cnn = new OleDbConnection(strcon);
cnn.Open();
cmdKh = new OleDbCommand("select * from khoa", cnn);
// Thực hiện các lệnh truy xuất dữ liệu
cnn.Close();
```

2.2.3. Các thuộc tính thường dùng của Command:

- **CommandText:** là câu lệnh SQL, tên Table hay tên Store Proc
- **CommandType:** quy định loại nội dung CommandText với các giá trị sau:
  - **Text(mặc định):** là một chuỗi câu lệnh SQL.

Thí dụ 2.9

```
SqlCommand cmd = New SqlCommand("select * from khoa", cnn);
cmd.CommandType = CommandType.Text;
SqlDataReader r;
r = cmd.ExecuteReader;
String s = "";
While (r.Read())
    s += r[0].ToString();
```

  - **StoredProcedure:** là một chuỗi chứa tên thủ tục trong SQL Server.

Thí dụ 2.10

```
SqlCommand cmd = New SqlCommand();
cmd.CommandType = CommandType.StoredProcedure;
cmd.CommandText = "sp_DSsinhvien";
cmd.Connection = cnn;
```

```
        String s  = "";
        SqlDataReader r ;
        r = cmd.ExecuteReader
        While (r.Read())
            s += r[0].ToString();
```

- **Connection:** là đối tượng kết nối sử dụng với đối tượng Command.
- **Parameters:** là tập hợp các tham số dùng với đối tượng Command.

2.2.4. Thực thi Command

Có nhiều phương thức được thực thi trên Command với các ý nghĩa khác nhau như sau:

- **ExcuteNonQuery:** Là phương thức được dùng để thực hiện các câu truy vấn cập nhật như Insert, Update, Delete, hay gọi thi hành một thủ tục (Store Procedure) không có trả về giá trị hoặc không quan tâm đến giá trị trả về. Phương thức này khi thi hành các câu truy vấn cập nhật sẽ có giá trị trả về là số dòng chịu tác động của đối tượng Command.

Thí dụ 2.11

```
...
OleDbConnection cn = New OleDbConnection("provider=sqloledb; data source=.; database=QLSV;uid=sa;pwd=cntt1");
OleDbCommand cmdmonhoc = New OleDbCommand();
cmdmonhoc.CommandText = "Insert into monhoc(mamh, tenmh) values ('10','Luật')";
cmdmonhoc.CommandType = CommandType.Text;
cmdmonhoc.Connection = cn;
int n = cmdmonhoc.ExecuteNonQuery(); // n là số mẫu tin chịu tác động
cn.Close()
...
```

- **ExcuteScalar:** Là phương thức dùng để thực hiện các câu truy vấn thống kê và chỉ trả về duy nhất một giá trị do câu truy vấn thực hiện.

Thí dụ 2.12

```
...
OleDbConnection cn = New OleDbConnection("...");
OleDbCommand cmdmonhoc = New OleDbCommand();
cmdmonhoc.CommandText = "Select count(*) from monhoc";
cmdmonhoc.CommandType = CommandType.Text;
cmdmonhoc.Connection = cn;
```

```
int n = (int)cmdmonhoc.ExecuteScalar();
cn.Close()
...
```

– **ExcuteReader:** Phương thức này trả về một đối tượng kiểu DataReader cho phép truy xuất dữ liệu từ nguồn với phương thức Read().

Thí dụ 2.13

```
...
String strcon = "provider=Microsoft.jet.oledb.4.0;data source=…";
OleDbConnection cnn = New OleDbConnection(strcon);
cnn.Open();
OleDbCommand cmdkhoa = New OleDbCommand();
cmdkhoa.CommandType = CommandType.Text;
cmdkhoa.CommandText = "select * from khoa";
cmdkhoa.Connection = cnn;
OleDbDataReader rkhoa;
rkhoa = cmdkhoa.ExecuteReader;
List<KHOA> mkhoa=new List<KHOA>();
While (rkhoa.Read())
    mkhoa.Add(New khoa(rkhoa[0].ToString(), rkhoa[1].ToString()));
ComboBox1.DataSource = mkhoa;
ComboBox1.DisplayMember = "tenkhoa";
ComboBox1.ValueMember = "makh";
...
```

2.3. DataReader

2.3.1. Khái niệm

Là đối tượng OleDbDataReader (tương ứng với Data Provider là OleDb) hay SQLDataReader (tương ứng với Data Provider là SQL). Đối tượng này có đặc điểm được sử dụng khi truy xuất dữ liệu chỉ đọc từ một nguồn dữ liệu cụ thể.

2.3.2. Các phương thức thường dùng của đối tượng DataReader:

– **Read:** Đọc một dòng trong nguồn dữ liệu (từ một bảng) và di chuyển đến dòng kế tiếp. Giá trị trả về là **true** nếu di chuyển được đến dòng kế tiếp; ngược lại trả về giá trị là **false**.

Chú ý: Khi đối tượng DataReader đang mở, các thao tác khác trên nguồn dữ liệu đều không thể thực hiện được cho đến khi đối tượng này đóng lại.

– **Close:** Đóng DataReader lại sau khi dữ liệu được đọc xong.

– **GetString:** Trả về giá trị tương ứng với cột có kiểu dữ liệu chuỗi.

Thí dụ: String s = r.GetString(0) // lấy giá trị kiểu chuỗi trên cột thứ 0.

– **GetInt16:** Trả về giá trị tương ứng với cột có kiểu dữ liệu là số nguyên.

Thí dụ: int n = r.GetInt16(1) // lấy giá trị kiểu Int16 trên cột thứ 1.

– **GetDouble:** Trả về giá trị tương ứng với cột có kiểu dữ liệu là số thực.

Thí dụ: Double n = r.GetDouble(7) // lấy giá trị kiểu Double trên cột thứ 7.

– **GetBoolean:** Trả về giá trị tương ứng với cột có kiểu dữ liệu logic.

Thí dụ: Boolean n = r.GetBoolean(3) // lấy giá trị kiểu Boolean trên cột thứ 4.

– **Cú pháp lấy giá trị của một cột trên dòng đang đọc:** Để lấy giá trị của một cột trên dòng đang đọc với phương thức Read, sử dụng cú pháp sau:

```
<Tên DataReader>[<Chỉ số cột bắt đầu từ 0> | "<Tên cột>"]
```

Thí dụ 2.14

```
...
String strcon = "provider=Microsoft.jet.oledb.4.0;data source=…";
OleDbConnection cnn = New OleDbConnection(strcon);
cnn.Open();
OleDbCommand cmdkhoa = New OleDbCommand();
...
OleDbDataReader rkhoa;
rkhoa = cmdkhoa.ExecuteReader;
List<KHOA> mkhoa=new List<KHOA>();
While (rkhoa.Read())
    mkhoa.Add(New khoa(rkhoa[0].ToString(), rkhoa[1].ToString()));
...
```

2.4. Đối tượng Parameter

2.4.1. Khái niệm

Các câu lệnh SQL hay Store Procedure có thể sử dụng tham số để thay thế cho các giá trị chưa xác định. Đối tượng Parameter truyền giá trị cho các tham số khi thực thi câu truy vấn. Tùy theo Data Provider, đối tượng Parameter sử dụng sẽ khai báo từ các lớp OleDbParameter hay SQLParameter.

2.4.2. Khai báo Parameter

Có nhiều cách khai báo và khởi tạo đối tượng Parameter. Cách thông dụng thường được sử dụng là:

```
OleDbParameter | SQLParameter <Tên Parameter>  = New OleDbParameter
               |SQLParameter(<Tên>,<Kiểu>,<K.thước>);
```

Thí dụ 2.15

```
SqlParameter p = New SqlParameter("@mkh", SqlDbType.Char, 10);
p.Value = "th";
```

2.4.3. Các thuộc tính thường dùng:

- **Direction:** cho biết loại tham số sử dụng. Có các giá trị sau:
  - **Input(Mặc định):** tham số sử dụng là loại tham số đầu vào (intput).
  - **InputOutput:** tham số sử dụng là loại tham số đầu vào và đầu ra (input và output).
  - **Output:** tham số sử dụng là loại tham số đầu ra (output).
  - **ReturnValue:** nhận giá trị trả về của một thủ tục (Store Procedure), một hàm do người dùng định nghĩa. Đối với provider OleDb thì tham số loại Return Value phải được khai báo đầu tiên.
- **OleDbType, SQLDbType:** kiểu dữ liệu của tham số tùy thuộc vào loại tham số là OleDbParameter hay SQLParameter.
- **Value:** Giá trị của tham số.

2.4.4. Truyền tham số cho Command

Có thể sử dụng phương thức Add của Parameters ứng với đối tượng Command.

Thí dụ 2.16

```
....
OleDbCommand cmd = New OleDbCommand();
cmd.CommandText = "select * from khoa where makh=@mk";
cmd.CommandType = CommandType.Text;
cmd.Connection = cnn;
OleDbParameter p = New OleDbParameter("@mk", OleDbType.Char,2);
p.Value = "th";
cmd.Parameters.Add(p);
String s = "";
OleDbDataReader r = OleDbDataReader();
r = cmd.ExecuteReader;
While (r.Read())
    s += r(0).ToString;
```

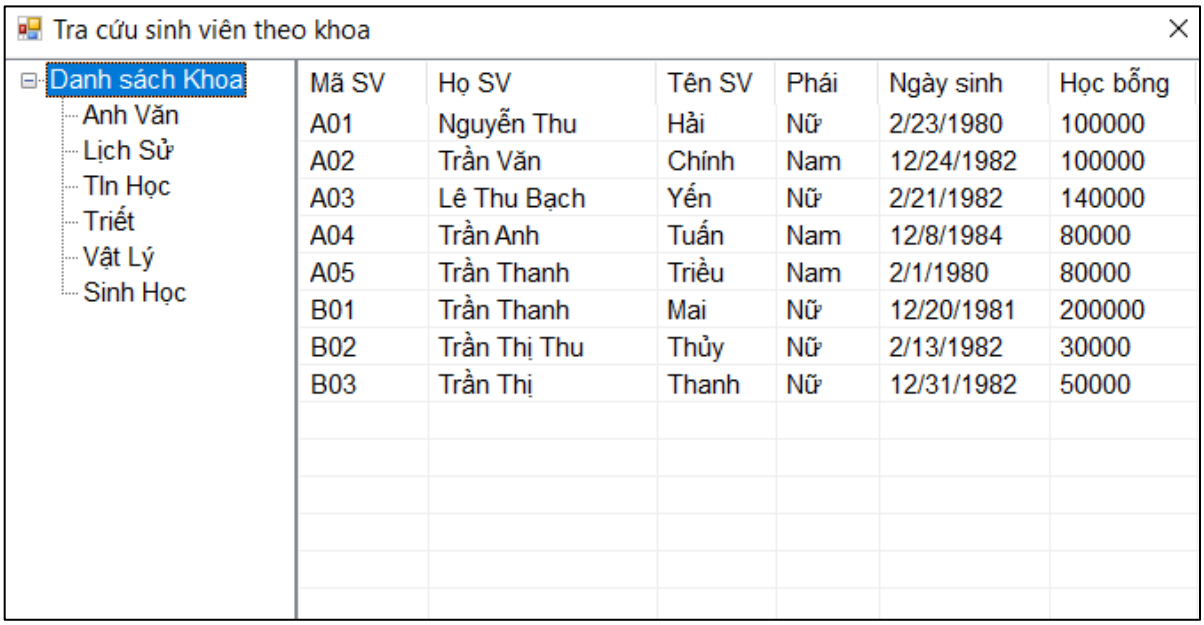
Lưu ý: Thứ tự tham số đưa vào phải theo thứ tự xuất hiện của tham số trong câu lệnh SQL.

Thí dụ 2.17

```
....
Cmd.CommandText=”Select * from Sinhvien where MaKh=@mk And Masv=@msv”;
Cmd.Parameters.Add(“@mk”,OleDbType.Char,2);
Cmd.Parameters.Add(“@msv”,OleDbType.Char,3);
....
```

BÀI TẬP CHƯƠNG 2

5. Thiết kế, định dạng màn hình và các đối tượng điều khiển theo mô tả của hình sau:



Yêu cầu:

5.1. Từ CSDL ứng với tập tin QLSV.mdb cho trước, sao chép tập tin này vào folder project. Tạo các lớp KHOA, SINHVIEN với các thuộc tính tương ứng với các cột trong trong bảng KHOA, SINHVIEN trong CSDL. Khai báo các đối tượng kiểu List<KHOA> và List<SINHVIEN> để lưu trữ danh sách các khoa và danh sách các sinh viên đọc được từ CSDL.

5.2. Sử dụng các đối tượng Connection, Command, DataReader và phương thức ExcuteReader để đọc và nạp dữ liệu từ CSDL vào các đối tượng danh sách tạo ra ở câu 5.1.

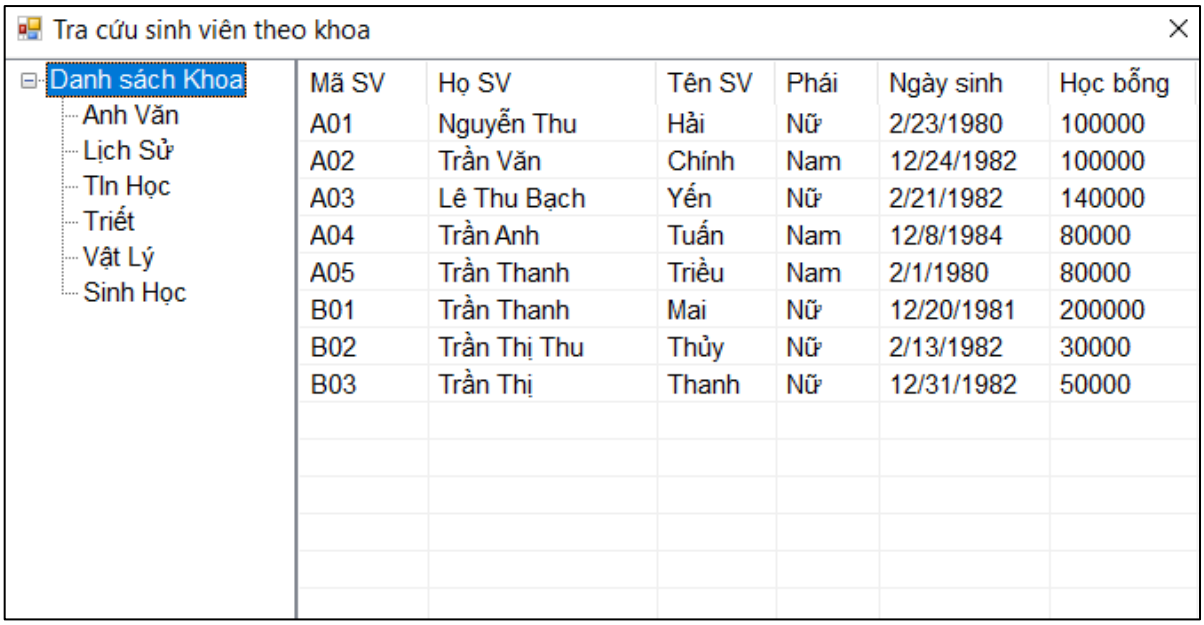
5.3. Từ dữ liệu chứa trong danh sách kiểu List<KHOA>, nạp dữ liệu vào điều khiển Treeview.

5.4. Khi người dùng Click vào một nút trên Treeview thì:

+ Nếu nút chọn là nút gốc thì nạp và hiển thị toàn bộ danh sách kiểu List<SINHVIEN> trên Listview.

+ Nếu nút chọn là nút con tương ứng với một khoa cụ thể thì Listview chỉ hiển thị danh sách kiểu List<SINHVIEN> ứng với Khoa được chọn.

6. Thiết kế, định dạng màn hình và các đối tượng điều khiển theo mô tả của hình sau:



Yêu cầu:

6.1. Thực hiện các thao tác sau:

+ Từ CSDL ứng với tập tin QLSV.mdb cho trước, sao chép tập tin này vào folder project.

+ Tạo DataSet ds chứa DataTable tblKhoa, tblSinhvien với các DataColumn tương ứng với các cột trong bảng KHOA, SINHVIEN tại CSDL trên.

+ Trong DataSet ds, tạo đối tượng DataRelation để thiết lập mối quan hệ giữa các DataTable tblKhoa và tblSinhvien.

6.2. Sử dụng các đối tượng Connection, Command, DataReader và phương thức ExcuteReader để đọc và nạp dữ liệu từ CSDL vào các DataTable tblKhoa và tblSinhvien được tạo ở câu trên.

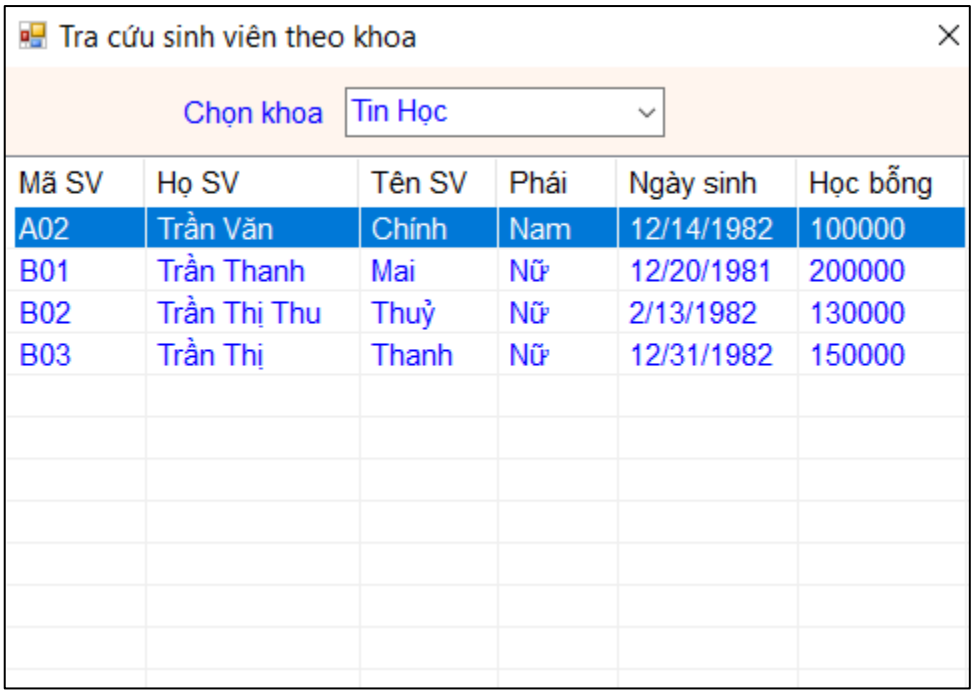
6.3. Từ dữ liệu trong DataTable tblKhoa, nạp dữ liệu vào điều khiển Treeview.

6.4. Khi người dùng Click vào một nút trên Treeview thì sử dụng các thuộc tính, phương thức tương ứng với đối tượng DataRelation và thực hiện:

+ Nếu nút chọn là nút gốc thì nạp và hiển thị toàn bộ danh sách sinhvien lưu trữ trong tblSinhvien lên Listview.

+ Nếu nút chọn là nút con tương ứng với một khoa cụ thể thì Listview chỉ hiển thị danh sách sinhviên ứng với Khoa được chọn.

7. Thiết kế, định dạng màn hình và các đối tượng điều khiển theo mô tả của hình sau:



Yêu cầu:

7.1. Thực hiện các thao tác sau:

+ Từ CSDL ứng với tập tin QLSV.mdb cho trước, sao chép tập tin này vào folder project.

+ Tạo DataSet ds chứa DataTable tblKhoa, tblSinhvien với các DataColumn tương ứng với các cột trong bảng KHOA, SINHVIEN tại CSDL trên.

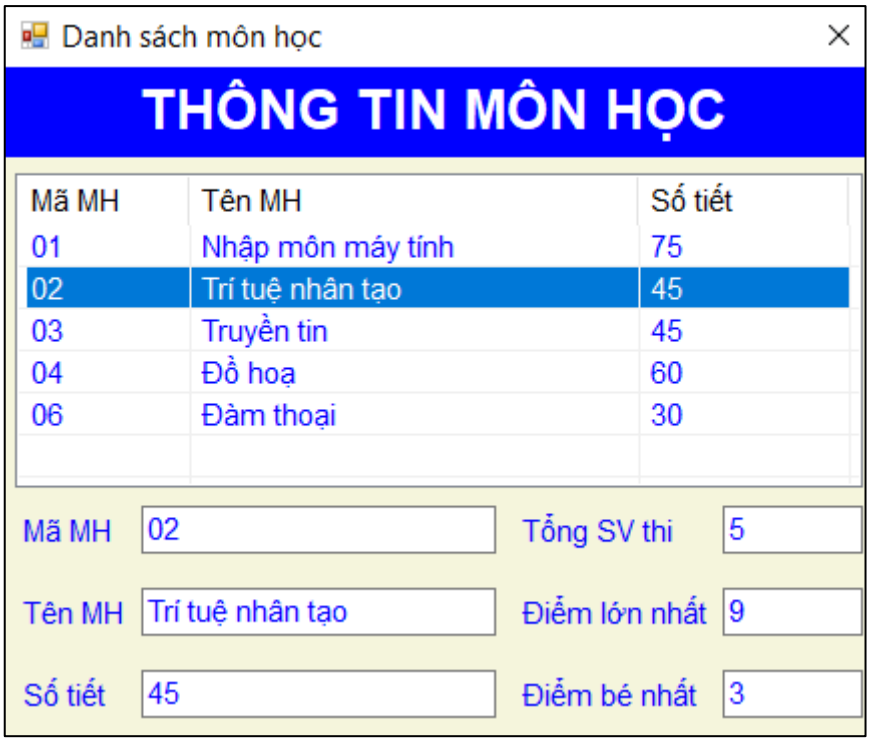
+ Trong DataSet ds, tạo đối tượng DataRelation để thiết lập mối quan hệ giữa các DataTable tblKhoa và tblSinhvien.

7.2. Sử dụng các đối tượng Connection, Command, DataReader và phương thức ExcuteReader để đọc và nạp dữ liệu từ CSDL vào các DataTable tblKhoa và tblSinhvien được tạo ở câu trên.

7.3. Thiết lập nguồn dữ liệu và các thuộc tính cho điều khiển Combobox.

7.4. Khi người dùng Click chọn một khoa cụ thể trên điều khiển Combobox thì sử dụng các thuộc tính, phương thức tương ứng với đối tượng DataRelation và thực hiện hiển thị danh sách các sinh viên tương ứng với Khoa được chọn trên Listview.

8. Thiết kế, định dạng màn hình và các đối tượng điều khiển theo mô tả của hình sau:



Yêu cầu:

8.1. Thực hiện các thao tác sau:

+ Từ CSDL ứng với tập tin QLSV.mdb cho trước, sao chép tập tin này vào folder project.

+ Tạo DataSet ds chứa DataTable tblMonhoc, tblKetqua với các DataColumn tương ứng với các cột trong bảng MONHOC, KETQUA tại CSDL trên.

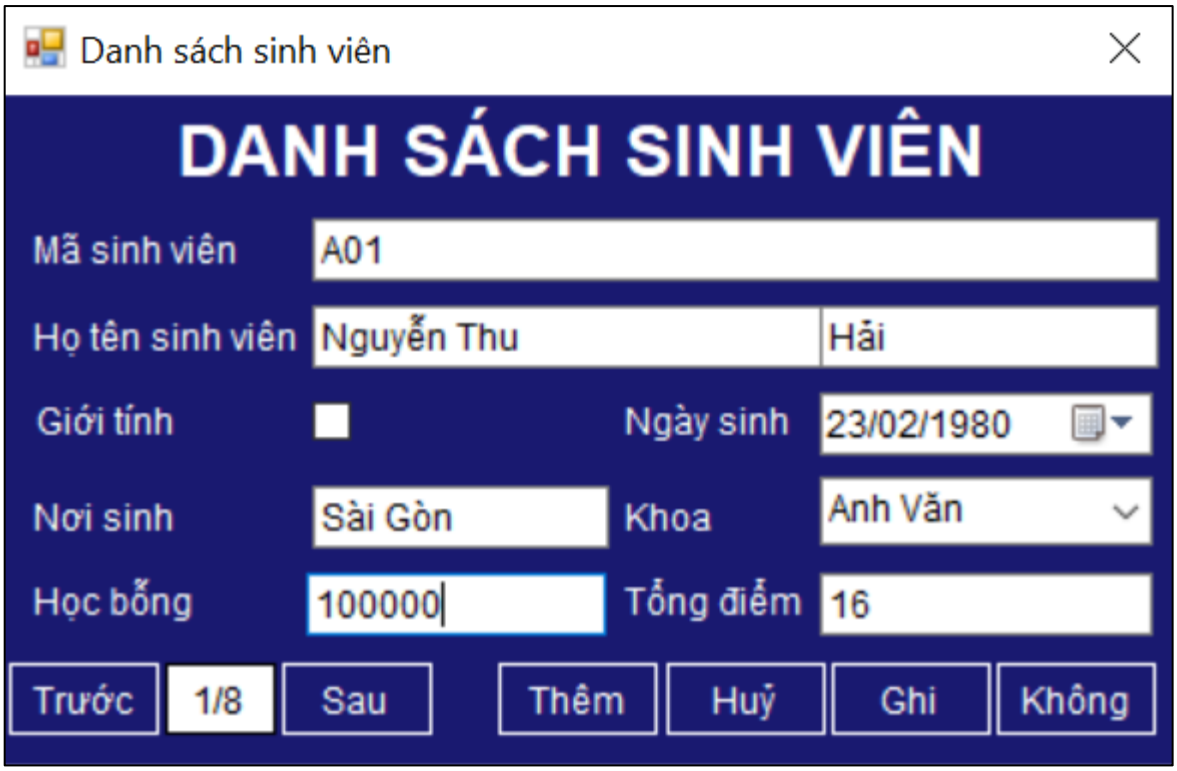
8.2. Sử dụng các đối tượng Connection, Command, DataReader và phương thức ExcuteReader để đọc và nạp dữ liệu từ CSDL vào các DataTable tblMonhoc và tblKetqua được tạo ở câu trên.

8.3. Đọc dữ liệu từ tblMonhoc và đưa vào điều khiển Listview.

8.4. Khi người dùng Click chọn một dòng cụ thể trên điều khiển Listview thì thông tin chi tiết của môn học được chọn hiển thị trên các điều khiển Textbox.

8.5. Sử dụng phương thức Compute của DataTable để tính và hiển thị tổng sinh viên thi, điểm lớn nhất, điểm bé nhất của môn học hiện hành.

9. Thiết kế, định dạng màn hình và các đối tượng điều khiển theo mô tả của hình sau:



Yêu cầu:

9.1. Thực hiện các thao tác sau:

+ Từ CSDL ứng với tập tin QLSV.mdb cho trước, sao chép tập tin này vào folder project.

+ Tạo DataSet ds chứa DataTable tblKhoa, tblSinhvien, tblKetqua, tblMonhoc với các DataColumn tương ứng với các cột trong bảng KHOA, SINHVIEN, KETQUA, MONHOC tại CSDL trên.

+ Tạo quan hệ khoá ngoại (Foreign key) giữa các DataTable tblKhoa và tblSinhvien thông qua DataColumn **MaKH** của các DataTable.

+ Tạo quan hệ khoá ngoại (Foreign key) giữa các DataTable tblSinhvien và tblketqua thông qua DataColumn **MaSV** của các DataTable.

+ Tạo quan hệ khoá ngoại (Foreign key) giữa các DataTable tblKetqua và tblMonhoc thông qua DataColumn **MaMH** của các DataTable.

9.2. Sử dụng các đối tượng Connection, Command, DataReader và phương thức ExcuteReader để đọc và nạp dữ liệu từ CSDL vào các DataTable tblKhoa, tblSinhvien, tblKetqua, tblMonhoc được tạo ở câu trên.

9.3. Thiết lập nguồn dữ liệu và các thuộc tính cho điều khiển Combobox.

9.4. Sử dụng phương thức Compute của DataTable để tính và hiển thị tổng điểm thi của sinh viên hiện hành.

9.5. Khi người dùng Click chọn các nút lệnh Trước, Sau thì cho phép di chuyển mẫu tin hiện hành về ngay mẫu tin phía trước, phía sau tương ứng. Thông báo lỗi khi không thể di chuyển được.

9.6. Khi người dùng Click chọn nút lệnh Thêm thì cho phép thêm mới một mẫu tin tương ứng với một sinh viên.

9.7. Khi người dùng Click chọn nút lệnh Huỷ thì cho phép xoá mẫu tin hiện hành trên màn hình. Yêu cầu người dùng xác nhận trước khi huỷ mẫu tin. Mẫu tin sau khi huỷ phải được cập nhật về CSDL.

9.8. Khi người dùng Click chọn nút lệnh Ghi thì cho phép ghi lại sự thay đổi của mẫu tin sau khi hiệu chỉnh hay thêm mới vào DataTable tblSinhvien. Mẫu tin sau khi ghi phải được cập nhật về CSDL.

9.9. Khi người dùng Click chọn nút lệnh Không thì cho phép phục hồi lại mẫu tin hiện hành trên màn hình.

9.10. Khi thực hiện các thao tác cần kiểm tra tính hợp lý, hợp lệ tương ứng với từng thao tác thực hiện.

10. Thực hiện lại bài tập số 6 theo yêu cầu sau:

10.1. Từ tập tin QLSV.mdb được cung cấp, thực hiện chuyển đổi dữ liệu và đưa vào SQL Server và đặt tên cho CSDL là QLSINHVIEN.

10.2. Các yêu cầu còn lại thực hiện như các yêu cầu của bài tập 6.

11. Thực hiện lại bài tập số 7 theo yêu cầu sau:

11.1. Từ tập tin QLSV.mdb được cung cấp, thực hiện chuyển đổi dữ liệu và đưa vào SQL Server và đặt tên cho CSDL là QLSINHVIEN.

11.2. Các yêu cầu còn lại thực hiện như các yêu cầu của bài tập 7.

12. Thực hiện lại bài tập số 7 theo yêu cầu sau:

12.1. Từ tập tin QLSV.mdb được cung cấp, thực hiện chuyển đổi dữ liệu và đưa vào SQL Server và đặt tên cho CSDL là QLSINHVIEN.

12.2. Các yêu cầu còn lại thực hiện như các yêu cầu của bài tập 8.

CHƯƠNG 3. LẬP TRÌNH VỚI CÁC ĐỐI TƯỢNG THEO MÔ HÌNH NGẮT KẾT NỐI

Sau khi học xong chương này, sinh viên có khả năng:

- *Trình bày được ý nghĩa, đặc điểm, phân biệt được các loại đối tượng DataSet và mô tả được các đối tượng thành phần trong một DataSet.*
- *Sử dụng được các phương thức để tạo lập, cập nhật các loại đối tượng DataSet và thành phần của một DataSet.*
- *Trình bày được ý nghĩa và công dụng của đối tượng DataAdapter.*
- *Sử dụng được các phương thức để truy xuất và cập nhật dữ liệu một cơ sở dữ liệu thông qua đối tượng DataAdapter.*
- *Trình bày được ý nghĩa và công dụng của các điều khiển hiển thị và liên kết dữ liệu.*
- *Sử dụng được các điều khiển hiển thị và liên kết dữ liệu để thiết kế màn hình theo yêu cầu bài toán.*

3.1. Tổng quan và phân loại Dataset

3.1.1. Khái niệm

Trong kiến trúc ADO.NET, DataSet là đối tượng khá phức tạp. Nó ánh xạ CSDL nguồn vào các đối tượng trong bộ nhớ. Dataset được xem như một CSDL thu nhỏ được lưu trữ trong bộ nhớ để đáp ứng các yêu cầu xử lý dữ liệu của ứng dụng theo mô hình ngắt kết nối. DataSet chứa hầu hết các đối tượng tương ứng với các thành phần trong CSDL như: với thành phần bảng (table) trong CSDL thì dataset có đối tượng DataTable, với thành phần cột (column) có đối tượng DataColumn, với thành phần dòng (row) có đối tượng DataRow, với thành phần quan hệ (relation) có đối tượng DataRelation, với các ràng buộc dữ liệu có đối tượng Constraint.

3.1.2. Phân loại Dataset

DataSet được phân thành hai loại là Dataset không định kiểu (Untyped DataSet) và DataSet có định kiểu (Typed DataSet).

– **DataSet không định kiểu (Untyped DataSet)**

Là đối tượng thuộc lớp System.Data.DataSet. Đặc điểm của đối tượng này là việc liên kết giữa các DataTable, kiểm tra lỗi được thực hiện tại thời điểm thực thi chương trình và lược đồ dữ liệu thì không tích hợp sẵn. Tuy nhiên, loại DataSet này có ưu điểm là hiệu năng thực hiện tốt hơn so với DataSet có định kiểu.

Cú pháp khai báo đối tượng Dataset không định kiểu:

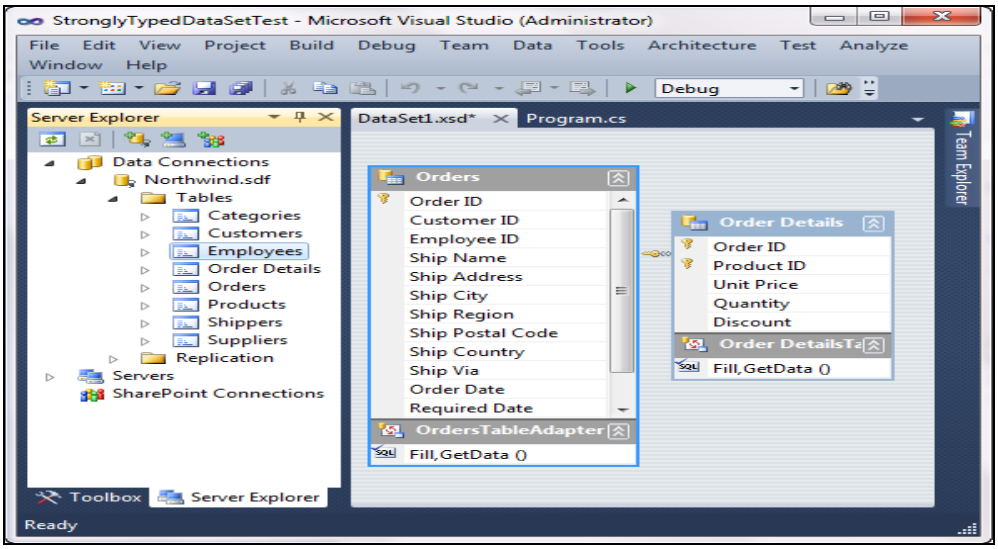
```
DataSet <Tên đối tượng>=New DataSet();
```

– **DataSet có định kiểu (Typed DataSet)**

Là lớp kế thừa từ lớp DataSet nên DataSet có định kiểu có đầy đủ các thuộc tính, phương thức, sự kiện của lớp DataSet. Đặc điểm của DataSet có định kiểu là việc liên kết với các DataTable và lược đồ được thực hiện tại thời điểm thiết kế. DataSet loại này được lưu trữ trong tập tin .xsd và có thể thực hiện kiểm tra lỗi liên quan đến lược đồ dữ liệu tại thời điểm thiết kế. Mặt khác, DataSet có định kiểu có thể được tái sử dụng khi thao tác với các đối tượng khác như thiết kế báo cáo (report), …. Tuy nhiên, xét về mặt hiệu năng thì DataSet có định kiểu cho hiệu năng thấp hơn so với DataSet không định kiểu.

Các bước tạo DataSet có định kiểu:

+ Tạo kết nối đến CSDL sử dụng **Server Explorer**.
+ Thêm đối tượng kiểu DataSet có định kiểu vào project bằng cách R-Click vào Project ⇒ chọn **Add** ⇒ chọn **New Item** ⇒ chọn **DataSet** và đặt tên cho DataSet.
+ Đưa các bảng cần thiết trong CSDL vào cửa sổ thiết kế (Designer) của DataSet vừa tạo và lưu lại.



Hình 3.1. Màn hình chứa các thành phần của Typed DataSet

3.2. Các thành phần trong DataSet

3.2.1. Với DataSet không định kiểu

- **DataTable:** là đối tượng trong bộ nhớ tương ứng với một bảng trong CSDL.
- **Tables:** là tập hợp các DataTable trong DataSet. Các thuộc tính và phương thức thường dùng:
  - **Add:** Thêm DataTable vào DataSet:

```
<DataSet>.Tables.Add(<DataTable>);
```

Thí dụ 3.1

```
Dataset ds=new Dataset();
DataTable tblKhoa=new DataTable();
...
// thêm DataTable tblKhoa vào dataset ds
ds.Tables.add(tblKhoa);
...
```

  - **Remove:** Xóa DataTable ra khỏi DataSet

```
Tables.Remove(<DataTable>);
```

Thí dụ 3.2

```
Dataset ds=new Dataset();
DataTable tblKhoa=new DataTable();
// Xoá DataTable tblKhoa ra khỏi dataset
ds.Tables.Remove(tblKhoa);
...
```

+ Truy xuất đến một DataTable trong DataSet

```
Tables[<chỉ số> | <"Tên bảng">]
```

Thí dụ 3.3

```
...
Dataset ds=new Dataset();
DataTable tblKhoa=new DataTable();
...
// DataTable tblKhoa tham chiếu đến DataTable "KHOA" trong Dataset
tblKhoa = ds.Tables["KHOA"];
...
```

– **Tập hợp Relations**

Chứa tập hợp quan hệ (DataRelation) trong DataSet. Các thuộc tính, phương thức thường dùng:

+ **Add:** để thêm một quan hệ (DataRelation) vào tập hợp.

```
Relations.Add(<Đối tượng DataRelation>);
```

Thí dụ 3.4

```
Dataset ds=new Dataset();
DataRelation relKhoa_Sinhvien = new DataRelation(...);
...
// Thêm DataRelation relKhoa_Sinhvien vào tập hợp Relations trong dataset ds
ds.Relations.add(relKhoa_Sinhvien);
...
```

+ **Remove:** xóa quan hệ ra khỏi tập hợp.

```
Relations.Remove(<Đối tượng DataRelation>);
```

Thí dụ 3.5

```
Dataset ds=new Dataset();
DataRelation relKhoa_Sinhvien = new DataRelation(...);
...
// Xoá DataRelation relKhoa_Sinhvien ra khỏi tập hợp Relations trong dataset ds
ds.Relations.Remove(relKhoa_Sinhvien);
...
```

+ **Tham chiếu đến một quan hệ**

```
Relations[< Tên quan hệ | Chỉ số>]
```

Thí dụ 3.6

```
Dataset ds=new Dataset();
DataRelation relKhoa_Sinhvien = new DataRelation(...);
...
//DataRelation relKhoa_Sinhvien tham chiếu đến DataRelation tên "FK_KHOA_SINHVIEN"
relKhoa_Sinhvien = ds.Relations["FK_KHOA_SINHVIEN"];
...
```

3.2.2. Với DataSet có định kiểu

Sau khi DataSet có định kiểu được tạo, ứng dụng sẽ phát sinh ra một số các lớp và đối tượng. Thí dụ, với DataSet QLSV.xsd sẽ chứa các DataTable tương ứng với các bảng KHOA, SINHVIEN, KETQUA và MONHOC trong CSDL QLSV. Lúc này một số lớp và đối tượng phát sinh chủ yếu nhận được như sau

– **Các lớp (class) phát sinh**

+ **Với DataSet:** hệ thống phát sinh lớp (class) DataSet kế thừa từ lớp DataSet. Thí dụ, nếu có một DataSet được tạo ra trong Project tương ứng với tập tin QLSV.xsd, hệ thống sẽ phát sinh lớp Dataset là **QLSV**.
+ **Với DataTable:** Trong lớp DataSet được tạo ra, hệ thống phát sinh các lớp tương ứng với các DataTable, mỗi DataTable tương ứng với một bảng dữ liệu trong CSDL. Thí dụ, nếu đưa vào Dataset QLSV các bảng Khoa và SinhVien, … của CSDL thì DataSet sẽ phát sinh các lớp tương ứng với các DataTable như **KhoaDataTable**, **SinhVienDataTable, …** kế thừa từ lớp DataTable.

Thí dụ 3.7

```
QLSV ds=new QLSV();
QLSVTableAdapters.KHOATableAdapter dtpkh = new QLSVTableAdapters.KHOATableAdapter();
QLSV.KHOADataTable tblkh = new QLSV.KHOADataTable();
dtpkh.Fill(tblkh);
...
```

+ **Với DataRow:** Trong lớp DataSet được phát sinh, hệ thống sẽ phát sinh các lớp con tương ứng với DataRow của từng DataTable trong

DataSet được tạo. Thí dụ, nếu đưa vào DataSet QLSV các bảng Khoa và SinhVien của CSDL thì sẽ phát sinh các class tương ứng với các DataRow như: **KhoaRow, SinhVienRow, KetQuaRow, …** kế thừa từ lớp DataRow.

Thí dụ 3.8

```
QLSV ds = new QLSV();
QLSVTableAdapters.KHOATableAdapter dtpkh = new QLSVTableAdapters.KHOATableAdapter();
dtpkh.Fill(ds.KHOA);
QLSV.KHOARow r = ds.KHOA[0];// Lấy dòng thứ 0 trong DataTable KHOA
txtmakh.Text = r.MAKH;
txttenkh.Text = r.TenKH;
...
```

+ **Với DataAdapter:** Ngoài việc phát sinh các lớp tương ứng với các DataTable và Datarow thì hệ thống cũng phát sinh ra các lớp **DataAdapter** tương ứng với các DataTable trong DataSet.

Lưu ý: Các lớp này không đặt trong **không gian tên gốc** mà đặt trong một **không gian tên con**. Thí dụ, từ DataSet QLSV, hệ thống tạo ra không gian tên con **QLSVTableAdapters**. Trong không gian tên con này sẽ chứa các DataAdapter tương ứng với các DataTable có trong DataSet. Thí dụ, nếu **DataSet QLSV** chứa 2 DataTable tương ứng với bảng Khoa và SinhVien thì các lớp phát sinh có **KhoaTableAdapter** và **SinhVienTableAdapter**.

Thí dụ 3.9

```
QLSV ds = new QLSV();
QLSVTableAdapters.KHOATableAdapter dtpkh = new QLSVTableAdapters.KHOATableAdapter();
dtpkh.Fill(ds.KHOA);
```

– **Các đối tượng phát sinh**

+ **Với DataTable:** Khi tạo ra đối tượng DataSet có định kiểu, đối tượng DataSet này sẽ phát sinh ra các đối tượng DataTable tương ứng với các bảng trong CSDL. Thí dụ, với **Dataset QLSV** đã tạo ở trên, nếu khởi tạo một đối tượng kiểu Dataset này thì đối tượng sẽ có các thuộc tính là **KHOA**, **SINHVIEN** là các đối tượng kiểu DataTable tương ứng với các bảng KHOA và SINHVIEN trong CSDL.

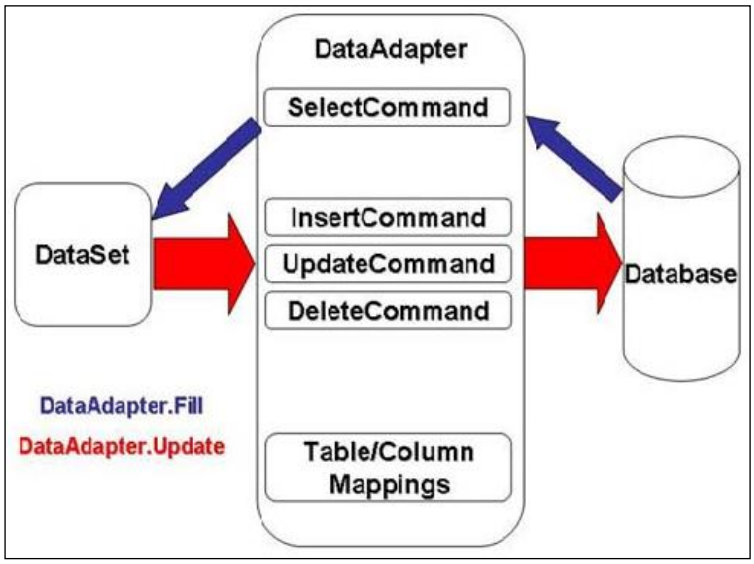
Thí dụ 3.10

```
...
QLSV ds = new QLSV();
QLSVTableAdapters.KHOATableAdapter dtpkh = new
QLSVTableAdapters.KHOATableAdapter();
...
// Gán nguồn dữ liệu cho combobox
cboMaKH.DisplayMember = "TenKH";
cboMaKH.ValueMember = "MaKH";
cboMaKH.DataSource = ds.KHOA;
...
```

3.3. DataAdapter

3.3.1. Khái niệm

Là lớp tạo ra cầu nối trung gian giữa Dataset với nguồn dữ liệu từ CSDL. Chức năng của lớp này là thực hiện các thao tác truy vấn dữ liệu nguồn đưa vào Dataset và cập nhật dữ liệu từ Dataset về CSDL. Đối tượng DataAdapter có các thuộc tính tương ứng với các đối tượng Command:



Hình 3.2. Kiến trúc và nguyên lý làm việc của DataAdapter

- **SelectCommand:** Cho phép sao chép cấu trúc và truy vấn dữ liệu từ nguồn dữ liệu về DataSet.
- **InsertCommand:** cho phép chèn thêm các dòng dữ liệu từ DataSet vào bảng trong nguồn dữ liệu.
- **UpdateCommand:** cho phép cập nhật dữ liệu từ DataSet vào bảng trong nguồn dữ liệu.

– **DeleteCommand:** cho phép cập nhật dữ liệu xoá trong DataSet vào bảng trong nguồn dữ liệu

Lưu ý: Chỉ cần thiết lập nội dung câu lệnh truy vấn cho **SelectCommand**, việc cập nhật dữ liệu về CSDL có thể sử dụng đối tượng **CommandBuilder** để đối tượng này tự động phát sinh các đối tượng Command cập nhật tương ứng.

3.3.2. Khởi tạo DataAdapter

DataAdapter được khởi tạo tuỳ theo loại Provider sử dụng trong ứng dụng là OLEDBDataAdapter hay SQLDataAdapter.

Cú pháp khởi tạo

```
New <DataAdapter>(<Chuỗi Lệnh>, <Chuỗi kết nối>);
```

Thí dụ 3.11

```
string strcon = @"provider=microsoft.jet.oledb.4.0; data source=..\..\qlsv.mdb";
DataSet ds = new DataSet();
DataTable tblkhoa = new DataTable("KHOA");
...
OleDbDataAdapter dtpkhoa = new OleDbDataAdapter("select * from khoa", strcon);
...
```

3.3.3. Tạo bộ lệnh cập nhật cho DataAdapter

Dựa vào nội dung truy xuất của DataAdapter, có thể sử dụng đối tượng CommandBuilder để tự động tạo các Command cập nhật còn lại theo cú pháp:

```
New <Command Builder>(<DataAdapter>);
```

Chú ý: Mỗi DataAdapter chỉ kết hợp với một CommandBuilder. Đối tượng này sử dụng câu lệnh trong **SelectCommand** của DataAdapter để tạo nội dung lệnh cho các Command còn lại với các đặc điểm sau:

– CommandBuilder phát sinh nội dung lệnh cập nhật cho các đối tượng DataAdapter có nội dung SelectCommand ***chỉ truy xuất đến một bảng***.

– Nội dung SelectCommand phải ***có ít nhất một khóa chính hay một khóa duy nhất*** (Unique Key).

Thí dụ 3.12

```
string strcon = @"provider=microsoft.jet.oledb.4.0; data source=..\..\qlsv.mdb";
DataSet ds = new DataSet();
DataTable tblkhoa = new DataTable("KHOA");
OleDbDataAdapter dtpkhoa = new OleDbDataAdapter("select * from khoa", strcon);
OleDbDataAdapter dtpsv = new OleDbDataAdapter("select * from sinhvien", strcon);
...
OleDbCommandBuilder cmbkhoa = new OleDbCommandBuilder(dtpkhoa);
OleDbCommandBuilder cmbsv = new OleDbCommandBuilder(dtpsv);
...
```

3.3.4. Một số các phương thức thường dùng của DataAdapter:

– **Fill:** Phương thức này đọc dữ liệu từ CSDL cung cấp cho các đối tượng DataTable trong bộ nhớ.

Cú pháp:

+ Đọc dữ liệu từ CSDL vào một DataTable.

```
<DataTable>.<DataAdapter>.Fill(<DataTable>):
```

+ Đọc dữ liệu từ CSDL vào một DataTable trong DataSet.

```
<DataAdapter>.Fill(<DataSet>,<DataTable>)
```

– **FillSchema:** Phương thức này đọc cấu trúc bảng tương ứng với DataAdapter từ CSDL vào đối tượng DataTable trong bộ nhớ.

Cú pháp thường dùng như sau:

```
<Tên DataAdapter>.FillSchema(<DataSet>, <Kiểu ánh xạ>,
                             "<TableName>");
```

Chú ý: Kiểu ánh xạ thường dùng là **SchemaType.Source:** với ý nghĩa sử dụng cấu trúc gốc của bảng tương ứng trong CSDL bao gồm các **ràng buộc khoá chính** và **ràng buộc duy nhất**.

Thí dụ 3.13

```
string strcon = @"provider=microsoft.jet.oledb.4.0; data source=..\..\qlsv.mdb";
DataSet ds = new DataSet();
DataTable tblkhoa = new DataTable("KHOA");
```

```
 ...
 OleDbDataAdapter dtpkhoa = new OleDbDataAdapter("select * from khoa",
strcon);
 ...
 OleDbCommandBuilder cmbkhoa  = new OleDbCommandBuilder(dtpkhoa);
 dtpkhoa.FillSchema(tblkhoa, SchemaType.Source);
 dtpkhoa.Fill(tblkhoa);
 ds.Tables.Add(tblkhoa);
 ...
```

– **Update:** Phương thức được sử dụng để cập nhật dữ liệu (thêm, sửa, xoá) từ các DataTable trong DataSet về nguồn dữ liệu trong CSDL.

Cú pháp:

+ Cập nhật thay đổi (thêm, sửa, xoá) trên DataTable về CSDL.

```
<DataAdapter>.Update(<DataTable>|"<Table name>")
```

+ Cập nhật thay đổi trên DataTable của Dataset về CSDL.

```
<DataAdapter>.Update(<DataSet>,<DataTable>|"<Table name>")
```

Thí dụ 3.14

```
string    strcon    =    @"provider=microsoft.jet.oledb.4.0;    data
source=..\..\qlsv.mdb";
 DataSet ds = new DataSet();
 DataTable tblkhoa = new DataTable("KHOA");
 ...
 OleDbDataAdapter dtpkhoa = new OleDbDataAdapter("select * from khoa",
strcon);
 ...
 OleDbCommandBuilder cmbkhoa  = new OleDbCommandBuilder(dtpkhoa);
 dtpkhoa.FillSchema(tblkhoa, SchemaType.Source);
 dtpkhoa.Fill(tblkhoa);
 ds.Tables.Add(tblkhoa);
 ...
 dtpsv.Update(ds, "SINHVIEN");
...
```

Chú ý: Với DataSet có định kiểu, các DataAdapter, DataTable sẽ được hệ thống phát sinh. Việc thực hiện đọc và cập nhật dữ liệu sẽ gọi thực hiện các phương thức và thuộc tính tương tự như đối với Dataset không định kiểu.

3.4. Các điều khiển hiển thị và liên kết dữ liệu

3.4.1. Lớp BindingSource

- **Ý nghĩa:** là lớp được tạo ra với vai trò là nguồn dữ liệu trung gian, nằm giữa đối tượng chứa dữ liệu và các điều khiển được sử dụng để liên kết, hiển thị và đồng bộ dữ liệu giữa các điều khiển trên màn hình và nguồn dữ liệu. BindingSource cho phép thực hiện các thao tác như: đọc dữ liệu, sắp xếp dữ liệu, lọc và cập nhật dữ liệu về nguồn dữ liệu một cách đơn giản với các phương thức được cung cấp sẵn.
- **Các thuộc tính thường dùng**
  - **Count:** cho biết tổng số phần tử trong danh sách
  - **Current:** trả về phần tử hiện hành trong danh sách.
  - **DataSource:** trả về hay gán nguồn dữ liệu với đối tượng BindingSource. Nguồn dữ liệu có thể là một mảng (danh sách) các đối tượng hay một DataTable.
  - **DataMember:** trả về hay gán nguồn dữ liệu liên kết với đối tượng BindingSource trường hợp thuộc tính DataSource của BindingSource là một DataSet.
  - **Filter:** biểu thức lọc dữ liệu trong danh sách liên kết với đối tượng BindingSource.
  - **Position:** trả về hay gán giá trị vị trí phần tử hiện hành trong danh sách.
  - **Sort:** gán hay trả về tên thuộc tính được sử dụng làm tiêu chuẩn sắp xếp trong danh sách.

Thí dụ 3.15

```
...
DataSet ds = new DataSet();
string strcon = @"provider=microsoft.jet.oledb.4.0; data source=..\..\qlsv.mdb";
OleDbDataAdapter dtpkh, dtpsv, dtpkq;
OleDbCommandBuilder cmbsv;
BindingSource bs = new BindingSource();
dtpsv = new OleDbDataAdapter("select * from sinhvien", strcon);
dtpsv.FillSchema(ds, SchemaType.Source, "SINHVIEN");
dtpsv.Fill(ds, "SINHVIEN");
```

```
cmbsv = new OleDbCommandBuilder(dtpsv);
...
bs.DataSource = ds;
bs.DataMember = ds.Tables["SINHVIEN"].TableName;
bs.Sort = "Hocbong DESC, Tensv ASC";
bs.Filter = "Makh='TH'";
bs.Position = 0;
...
```

– **Các phương thức thường dùng:**

+ **MoveFirst:** Di chuyển về phần tử đầu tiên trong danh sách.
+ **MovePrevious:** Di chuyển về phần tử ngay phía trước phần tử hiện hành trong danh sách.
+ **MoveNext:** Di chuyển về phần tử ngay phía sau phần tử hiện hành trong danh sách.
+ **MoveLast:** Di chuyển về phần tử cuối cùng trong danh sách.

Thí dụ 3.16

```
private void btndau_Click(object sender, EventArgs e)
{
    bs.MoveFirst();
}
private void btntruoc_Click(object sender, EventArgs e)
{
    If(bs.Position==0)
    {
        MessageBox.Show("Không di chuyển về trước được");
        return;
    }
    bs.MovePrevious();
}
private void btnsau_Click(object sender, EventArgs e)
{
    If(bs.Position==bs.Count-1)
    {
        MessageBox.Show("Không di chuyển về sau được");
        return;
    }
    bs.MoveNext();
```

```
}
private void btncuoi_Click(object sender, EventArgs e)
{
    bs.MoveLast();
}
```

+ **AddNew:** thêm mới một phần tử vào danh sách.

Thí dụ 3.17

```
private void btnthem_Click(object sender, EventArgs e)
{
    bs.AddNew();
    txtmasv.Focus();
}
```

+ **CancelEdit:** hủy bỏ hiệu chỉnh phần tử hiện hành trong danh sách.

Thí dụ 3.18

```
private void btnkhong_Click(object sender, EventArgs e)
{
    bs.CancelEdit();
}
```

+ **EndEdit:** lưu lại sự thay đổi của phần tử hiện hành trong danh sách.

Thí dụ 3.19

```
private void btnghi_Click(object sender, EventArgs e)
{
   DataRowView r = (bs.Current as DataRowView);
   if (r.IsNew)// Thêm mới. Kiểm tra khoá chính và khoá ngoại
   {
      if (ds.Tables["SINHVIEN"].Rows.Contains(txtmasv.Text))
      {
          MessageBox.Show("Trùng khoá chính");
          return;
      }
      if            (string.IsNullOrEmpty(txtmakh.Text)            ||
!ds.Tables["KHOA"].Rows.Contains(txtmakh.Text))
      {
          MessageBox.Show("Ma khoa khong hợp lệ");
          return;
      }
```

```
    }
    else if (r.IsEdit)
             if         (string.IsNullOrEmpty(txtmakh.Text)         ||
!ds.Tables["KHOA"].Rows.Contains(txtmakh.Text))
             {
                 MessageBox.Show("Ma khoa khong hợp lệ");
                 return;
             }
    bs.EndEdit();
    dtpsv.Update(ds, "SINHVIEN");
}
```

+ **RemoveCurrent:** Hủy bỏ phần tử hiện hành trong danh sách.

Thí dụ 3.20

```
private void btnhuy_Click(object sender, EventArgs e)
{
    DataRow rsv = (bs.Current as DataRowView).Row;
    DataRow[] mangkq = rsv.GetChildRows("FK_SV_KQ");
    if (mangkq.Length > 0)
    {
        MessageBox.Show("Không xoá được vì sinh viên có điểm.");
        return;
    }
    bs.RemoveCurrent();
    dtpsv.Update(ds, "SINHVIEN");
}
```

– **Sự kiện thường dùng:**

+ **CurrentChanged:** sự kiện này phát sinh khi có sự thay đổi phần tử hiện hành của đối tượng BindingSource. Để phát sinh và xử lý sự kiện, có thể thực hiện như sau:

```
<Đối tượng BindingSource>.CurrentChanged+=<Tên phương thức sự kiện>;
```

Thí dụ 3.21

```
public Form1()
{
    InitializeComponent();
    bs.CurrentChanged += bs_CurrentChanged;
}
```

```
private void bs_CurrentChanged(object sender, EventArgs e)
{
    //Sự kiện này xảy ra khi có sự thay đổi mẫu tin hiện hành
    lblSTT.Text = bs.Position + 1 + "/" + bs.Count;
}
```

3.4.2. BindingNavigator

Là điều khiển được sử dụng để thực hiện nhanh các thao tác di chuyển, cập nhật dữ liệu trên màn hình. Cách sử dụng đối tượng BindingNavigator rất đơn giản, chỉ cần gán giá trị cho thuộc tính BindingSource của đối tượng này là đối tượng BindingSource được sử dụng liên kết với các điều khiển trên màn hình.

Thí dụ 3.22

```
BindingNavigator1.Dock=DocStyle.Bottom;
BindingNavigator1.BindingSource= BindingSource1;
```

3.4.3. Đối tượng Binding

- **Khái niệm: Binding** được sử dụng để liên kết một thuộc tính của điều khiển với một thuộc tính của một đối tượng trong màn hình.
- **Cú pháp tạo Binding:** Cú pháp khai báo một Bingding như sau:

```
Binding <đối tượng> = New Binding(<thuộc tính điều khiển>, <Đối
        tượng nguồn>,<thuộc tính trên nguồn>, ...)
```

  + **<Thuộc tính điều khiển>:** là tên thuộc tính của điều khiển dùng để liên kết với đối tượng nguồn.
  + **<Đối tượng nguồn>:** là đối tượng cung cấp giá trị để liên kết, có thể là các đối tượng như: DataSet, DataTable, DataView, BindingSource, một điều khiển nào đó, ...
  + **<Thuộc tính trên nguồn>:** là thuộc tính trên đối tượng nguồn có giá trị liên kết với điều khiển.

Thí dụ 3.23

```
Public void LienKetDuLieu()
{
   txtmasv.DataBindings.Add("Text", bs, "MaSV", true);
   txthosv.DataBindings.Add("Text", bs, "HoSV", true);
   txttensv.DataBindings.Add("Text", bs, "TenSV", true);
   Binding bdphai = new Binding("Text", bs, "Phai", true);
   bdphai.Format += Bdphai_Format;
```

```
    bdphai.Parse += Bdphai_Parse;
    txtphai.DataBindings.Add(bdphai);
    txtngaysinh.DataBindings.Add("Text", bs, "NgaySinh", true);
    txtngaysinh.DataBindings[0].DataSourceUpdateMode                =
                                DataSourceUpdateMode.OnValidation;
    txtngaysinh.DataBindings[0].NullValue = "--";
    txtngaysinh.DataBindings[0].FormatString = "dd/MM/yyyy";
    txtnoisinh.DataBindings.Add("Text", bs, "NoiSinh", true);
    txtmakh.DataBindings.Add("Text", bs, "MaKH", true);
    txthocbong.DataBindings.Add("Text", bs, "HocBong", true);
    txthocbong.DataBindings[0].FormatString = "#,##0$";
}
```

– **Các sự kiện thường dùng của Binding:**

+ **Format:** sự kiện này phát sinh khi dữ liệu được đưa từ nguồn dữ liệu lên điều khiển hay khi giá trị của điều khiển bị thay đổi. Sự kiện này thường được sử dụng để định dạng dữ liệu và hiển thị dữ liệu trên điều khiển.

Thí dụ 3.24

```
void bdPhai_Format(object sender, ConvertEventArgs e)
{
     if (e.Value == DBNull.Value)
        return;
     e.Value = (Boolean)e.Value == true ? "Nam" : "Nữ";
}
```

+ **Parse:** sự kiện này phát sinh khi **giá trị của điều khiển liên kết thay đổi** nhằm **chuyển đổi dữ liệu về đúng kiểu** trước khi dữ liệu đưa về nguồn dữ liệu. **Chú ý:** Trong sự kiện này, cần phải loại bỏ định dạng của dữ liệu trước khi cập nhật dữ liệu về nguồn.

Thí dụ 3.25

```
void bdPhai_Parse(object sender, ConvertEventArgs e)
{
     e.Value = e.Value.ToString().ToUpper() == "NAM" ? true : false;
}
```

- **Thuộc tính thường dùng liên kết dữ liệu:**
  - + Điều khiển **Label, Textbox, DomainUpDown:** thuộc tính liên kết là **Text**;
  - + Điều khiển **CheckBox, RadioButton:** thuộc tính liên kết là **Checked**;
  - + Điều khiển **ComboBox, ListBox:** thuộc tính liên kết **SelectedValue**;
  - + Điều khiển **DateTimePicker, NumericUpDown:** thuộc tính liên kết là **Value.**

3.4.4. Tập hợp DataBindings

Một điều khiển có thể liên kết với các nguồn dữ liệu khác nhau thông qua các thuộc tính khác nhau chẳng hạn như thuộc tính Text của điều khiển liên kết với một nguồn, thuộc tính Visible lại liên kết với một nguồn dữ liệu khác, ... DataBindings là tập hợp các đối tượng Binding của điều khiển, mỗi đối tượng Binding liên kết với một nguồn dữ liệu. Để thêm các đối tượng Binding vào tập hợp ta sử dụng phương thức Add với cú pháp sau:

- **Cách 1:**

```
<Tập hợp>.Add(<đối tượng Binding>)
```

- **Cách 2:**

```
<Tập hợp>.Add(<th.tính ĐK >, <Đối tượng nguồn>, <Th.tính nguồn>,..)
```

Thí dụ 3.26

```
Binding bdPhai = new Binding("Text", bdsSV, "Phai");
bdPhai.FormattingEnabled = true;
bdPhai.Format += bdPhai_Format; // Phát sinh thủ tục sự kiện Format
bdPhai.Parse += bdPhai_Parse; // Phát sinh thủ tục sự kiện Parse
// Để phát sinh sự kiện Format, Parse phải khai báo  một đối tượng Binding riêng
txtPhai.DataBindings.Add(bdPhai);
Binding bdNgaysinh = new Binding("Text", bdsSV, "Ngaysinh", true);
bdNgaysinh.FormatString= "dd/MM/yyyy";
txtNgaysinh.DataBindings.Add(bdNgaysinh);

void bdPhai_Parse(object sender, ConvertEventArgs e)
{
     e.Value = e.Value.ToString().ToUpper() == "NAM" ? true : false;
}
```

```
void bdPhai_Format(object sender, ConvertEventArgs e)
{
     if (e.Value == DBNull.Value) return;
     e.Value = (Boolean)e.Value == true ? "Nam" : "Nữ";
}
```

3.4.5. Đối tượng DataGridView

- **Ý nghĩa:** là điều khiển được dùng để hiển thị dữ liệu theo dạng bảng với rất nhiều chức năng. Dữ liệu trên điều khiển được lưu trữ trong bộ nhớ nên vừa làm tăng tốc độ xử lý vừa đảm bảo khả năng xử lý nhiều mẫu tin cùng lúc.
- **Một số thuộc tính và phương thức thường dùng**
  - + **Thiết lập nguồn dữ liệu cho lưới:**

```
<Tên DataGridView>.DataSource = <Nguồn dữ liệu>;
<Tên DataGridView>.DataMember = <Thành phần trong nguồn dữ liệu>;
```

Thí dụ 3.27

```
Dataset ds=new Dataset();
...
this.dgvmh.DataSource = ds;
this.dgvmh.DataMember=tblMonhoc.Name;
```

Chú ý: Thuộc tính DataSource có thể là một DataSet, một DataTable hay một danh sách các đối tượng. Thuộc tính DataMember chỉ sử dụng khi thuộc tính DataSource là một DataSet**.**

  - + **Thêm cột:**

```
<Tên DataGridView>.Columns.Add("<Tên cột>","<Tiêu đề>");
```

Thí dụ 3.28

```
this.dgvmh.Columns.Add("MaMH", "Mã MH");
```

  - + **Thiết lập chế độ không sắp xếp các cột:** Mục đích để canh giữa chuỗi nội dung của thành phần ColumnHeader.

```
<Tên cột>.SortMode=DataGridViewColumnSortMode.NotSortable;
```

Thí dụ 3.29

```
foreach (DataGridViewColumn col in this.dgvmh.Columns)
   col.SortMode = DataGridViewColumnSortMode.NotSortable;
```

+ **Canh lề cho nội dung tiêu đề các cột:**

```
<Tên DataGridView>.ColumnHeaderDefaultCellStyle.Alignment= ….;
```

+ **Xóa dòng trong lưới:**

```
<Tên DataGridView>.Rows.Remove("<Biến dòng>");
```

Thí dụ 3.30

```
this.dgvmh.Rows.Remove(r);
```

+ **Thêm dòng vào lưới:**

```
<Tên DataGridView>.Rows.Add("<Giá trị trên các cột>");
```

Thí dụ 3.31

```
this.dgvmh.Rows.Add("01", "Nhập môn tin học", 45);
```

– **Định dạng hiển thị các thành phần của đối tượng DataGridView**

+ **EnalbleHeaderVisualStyle:** thường sử dụng khi muốn gán chế độ không sử dụng Themes hệ thống (được dùng khi cần thiết lập màu cho thành phần ColumnHeader)

Thí dụ 3.32

```
this.dgvmh.EnableHeadersVisualStyles = false;
```

+ **Tạo & gán font chữ cho tiêu đề các cột:** sử dụng thuộc tính ColumnHeaderDefaultCellStyle.Font

Thí dụ 3.33

```
this.dgvMonhoc.ColumnHeadersDefaultCellStyle.Font = new Font
                                ("tahoma", 11,FontStyle.Bold);
this.dgvmh.ColumnHeadersDefaultCellStyle.ForeColor = Color.Blue;
```

+ **ColumnHeaderHeightSizeMode:** thường sử dụng khi muốn thiết lập không cho phép thay đổi chiều cao tiêu đề cột (khi sử dụng chuột)

Thí dụ 3.34

```
dgvmh.ColumnHeadersHeightSizeMode                                    =

DataGridViewColumnHeadersHeightSizeMode.DisableResizing;
this.dgvmh.ColumnHeadersHeight = 40;
```

+ **SelectionMode:** cho phép thiết lập chế độ đánh dấu chọn 1 dòng trong lưới.

Thí dụ 3.35

```
this.dgvmh.SelectionMode = DataGridViewSelectionMode.FullRowSelect;
```

+ **MultiSelect:** thường sử dụng khi muốn thiết lập chế độ một thời điểm chỉ có 1 dòng được chọn.

Thí dụ 3.36

```
this.dgvmh.MultiSelect = false;
```

+ **Phương thức Select:** sử dụng để chọn đối tượng lưới là hiện hành.

Thí dụ 3.37

```
this.dgvmh.Select();
```

+ **Thuộc tính Selected:** sử dụng để chọn dòng hiện hành trong lưới.

Thí dụ 3.38

```
this.dgvmh.Rows[0].Selected = true;
```

+ **Lấy giá trị của dòng đang được chọn trong lưới:**

Thí dụ 3.39

```
DataGridViewRow r = this.dgvmh.SelectedRows[0];
```

+ **Lấy giá trị của ô trên dòng đang chọn:**

Thí dụ 3.40

```
this.txtMaMH.Text = r.Cells[0].Value.ToString();
```

+ **Sắp xếp dữ liệu hiển thị trong lưới:**

Thí dụ 3.41

```
this.dgvmh.Sort(this.dgvmh.Columns[0],
ListSortDirection.Ascending);
```

+ **Thiết lập lưới ở chế độ không được thêm:**

Thí dụ 3.42

```
this.dgvmh.AllowUserToAddRows = false;
```

+ **Thiết lập lưới ở chế độ không được xóa:**

Thí dụ 3.43

```
this.dgvmh.AllowUserToDeleteRows = false;
```

+ **Thứ tự ưu tiên định dạng trên lưới theo các thuộc tính từ trên xuống dưới sau đây:**

  (i) DefaultCellStyle: Định đạng cho các ô trong lưới.

  (ii) <Tên cột>. DefaultCellStyle: Định dạng cột cụ thể trong lưới.

  (iii) RowDefaultCellStyle: Định dạng các dòng trong lưới.

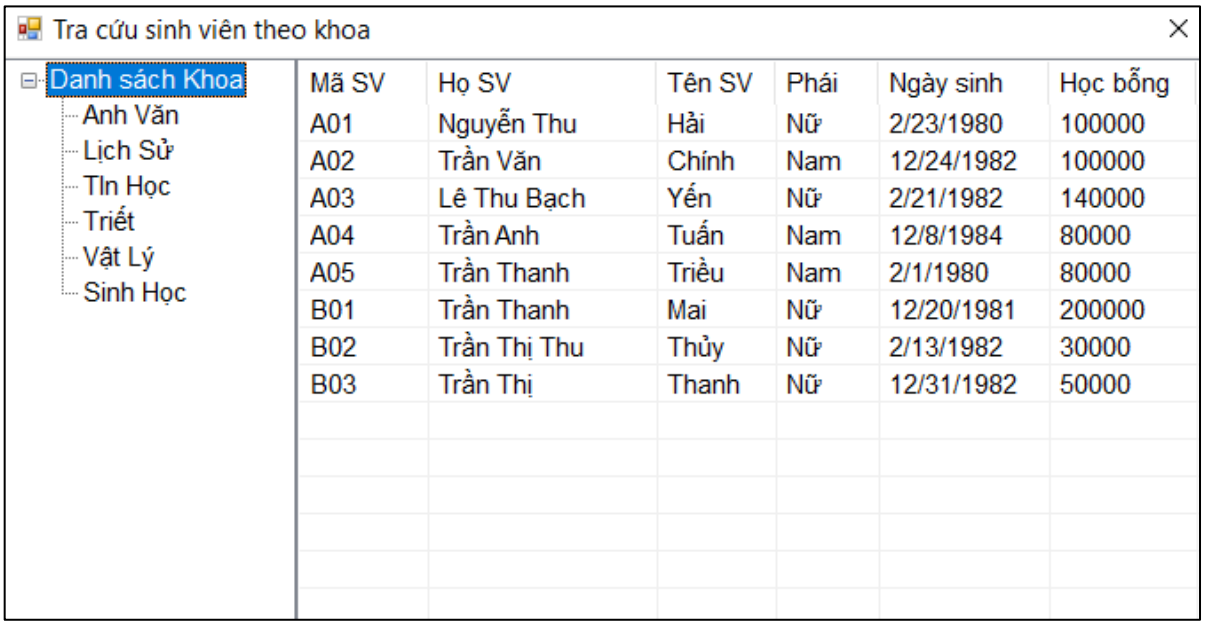
(iv) AlternatingRowDefaultCellStyle: Định dạng các dòng xen kẻ trong lưới.

(v) Row[index].DefaultCellStyle: Định dạng dòng cụ thể trong lưới.

(vi) [<Dòng>, <Cột>].Style: Định dạng cho ô cụ thể trong lưới.

BÀI TẬP CHƯƠNG 3

13. Thiết kế, định dạng màn hình và các đối tượng điều khiển theo mô tả của hình sau:



Yêu cầu:

13.1. Thực hiện các thao tác sau:

+ Từ CSDL ứng với tập tin QLSV.mdb cho trước, sao chép tập tin này vào folder project.

+ Tạo đối tượng DataSet không định kiểu ds, đối tượng DataAdapter và các phương thức FillSchema, Fill để đọc và nạp dữ liệu từ CSDL đã cho ở trên vào các DataTable trong DataSet.

+ Tạo quan hệ khoá ngoại (Foreign key) giữa các DataTable trong DataSet.

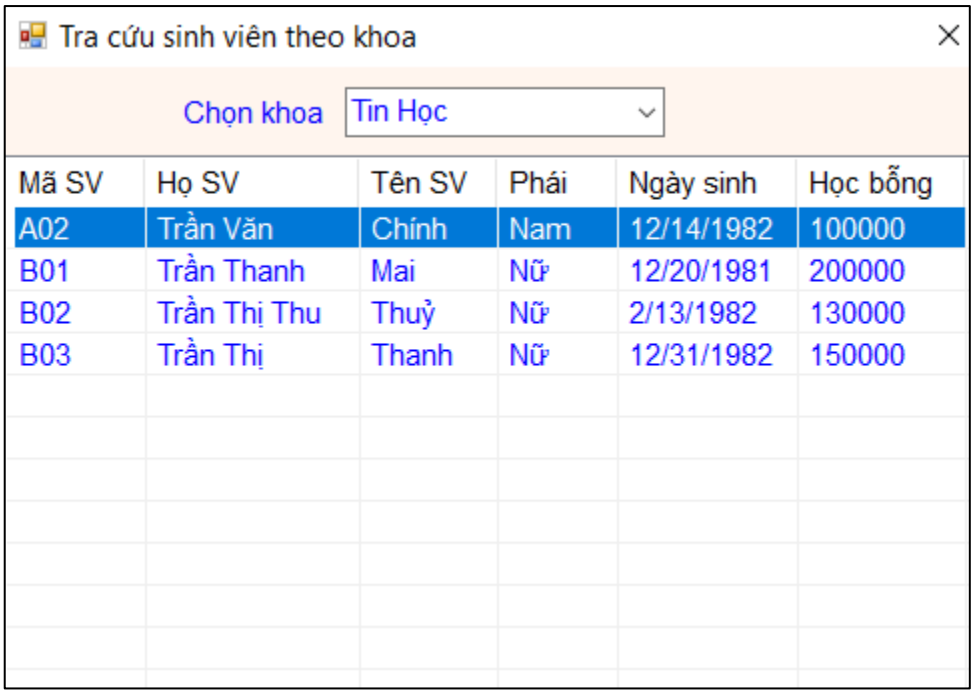
13.2. Từ dữ liệu lưu trữ trong DataTable của DataSet tương ứng với dữ liệu của bảng KHOA trong CSDL, hãy nạp dữ liệu vào điều khiển Treeview.

13.3. Khi người dùng Click vào một nút trên Treeview thì từ dữ liệu được lưu trữ trong DataSet tương ứng với dữ liệu của bảng SINHVIEN trong CSDL:

+ Nếu nút chọn là nút gốc thì nạp và hiển thị toàn bộ danh sách sinh viên lên Listview.

+ Nếu nút chọn là nút con tương ứng với một khoa cụ thể thì sử dụng các thuộc tính, phương thức tương ứng với đối tượng DataRelation tạo ra để hiển thị danh sách các sinh viên tương ứng với Khoa được chọn.

14. Thiết kế, định dạng màn hình và các đối tượng điều khiển theo mô tả của hình sau:



Yêu cầu:

14.1. Thực hiện các thao tác sau:

+ Từ CSDL ứng với tập tin QLSV.mdb cho trước, sao chép tập tin này vào folder project.

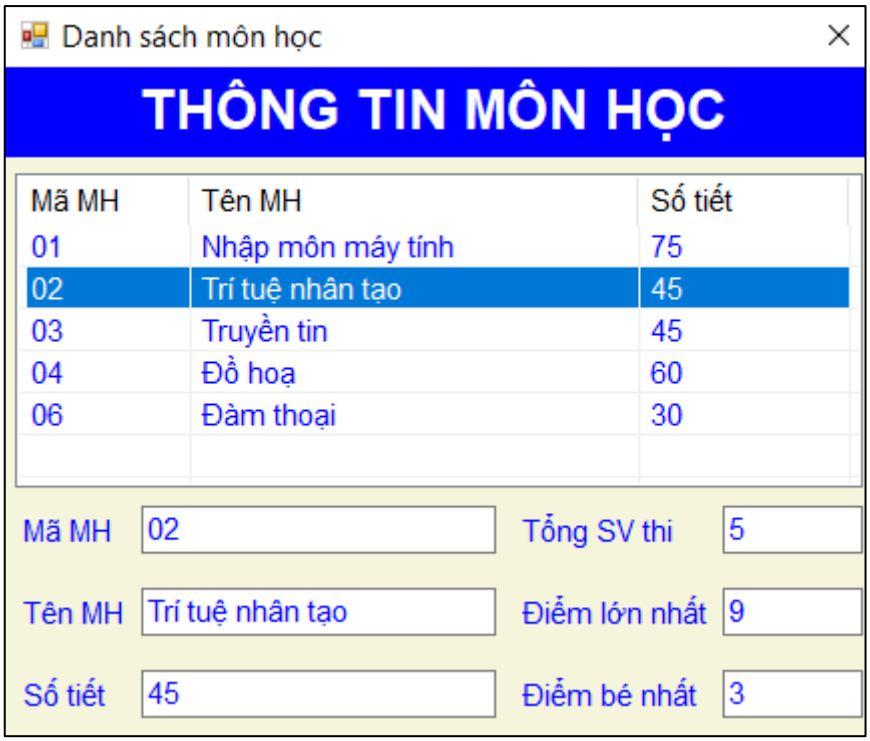
+ Tạo đối tượng DataSet không định kiểu ds, đối tượng DataAdapter và các phương thức FillSchema, Fill để đọc và nạp dữ liệu từ CSDL đã cho ở trên vào các DataTable trong DataSet.

+ Tạo quan hệ khoá ngoại (Foreign key) giữa các DataTable trong DataSet.

14.2. Từ dữ liệu lưu trữ trong DataSet tương ứng với bảng KHOA, thiết lập nguồn dữ liệu và các thuộc tính liên quan cho điều khiển Combobox.

14.3. Khi người dùng Click chọn một khoa cụ thể trên điều khiển Combobox thì thì sử dụng các thuộc tính, phương thức tương ứng với đối tượng DataRelation đã tạo ra để hiển thị danh sách các sinh viên tương ứng với Khoa được chọn trên điều khiển Combobox.

15. Thiết kế, định dạng màn hình và các đối tượng điều khiển theo mô tả của hình sau:



Yêu cầu:

15.1. Thực hiện các thao tác sau:

+ Từ CSDL ứng với tập tin QLSV.mdb cho trước, sao chép tập tin này vào folder project.

+ Tạo đối tượng DataSet không định kiểu ds, đối tượng DataAdapter và các phương thức FillSchema, Fill để đọc và nạp dữ liệu từ CSDL đã cho ở trên vào các DataTable trong DataSet.

+ Tạo quan hệ khoá ngoại (Foreign key) giữa các DataTable trong DataSet.

15.2. Từ dữ liệu lưu trữ trong DataSet tương ứng với bảng MONHOC, hãy nạp dữ liệu vào điều khiển Listview trên màn hình

15.3. Khi người dùng Click chọn một dòng cụ thể trên điều khiển Listview thì thông tin chi tiết của môn học được chọn hiển thị trên các điều khiển Textbox.

15.4. Khi hiển thị một môn học cụ thể trên màn hình, hãy sử dụng phương thức Compute của DataTable tương ứng với bảng KETQUA để tính toán và hiển thị tổng số sinh viên thi, điểm lớn nhất, điểm bé nhất của môn học hiện hành.

16. Thiết kế, định dạng màn hình và các đối tượng điều khiển theo mô tả của hình sau:

16.1. Thực hiện các thao tác sau:

+ Từ CSDL ứng với tập tin QLSV.mdb cho trước, sao chép tập tin này vào folder project.

+ Tạo đối tượng DataSet không định kiểu ds, đối tượng DataAdapter, CommandBuider và các phương thức FillSchema, Fill, Update để đọc và cập nhật dữ liệu từ CSDL đã cho ở trên vào các DataTable trong DataSet và ngược lại.

+ Tạo quan hệ khoá ngoại (Foreign key) giữa các DataTable trong DataSet tương ứng với bảng KETQUA và MONHOC trong CSDL.

16.2. Sử dụng các đối tượng BindingSource và tập hợp DataBindings để liên kết dữ liệu trong DataSet với các điều khiển trên màn hình.

16.3. Khi người dùng Click chọn các nút lệnh Trước, Sau thì cho phép di chuyển mẫu tin hiện hành về ngay mẫu tin phía trước, phía sau tương ứng với các phương thức MovePrevious, MoveNext của đối tượng BindingSource. Xuất thông báo lỗi khi không thể di chuyển được.

16.4. Khi người dùng Click chọn nút lệnh Thêm thì cho phép thêm mới một mẫu tin tương ứng với một môn học.

16.5. Khi người dùng Click chọn nút lệnh Huỷ thì cho phép huỷ mẫu tin hiện hành trên màn hình. Yêu cầu người dùng xác nhận trước khi huỷ mẫu tin. Mẫu tin sau khi huỷ phải được cập nhật về CSDL.

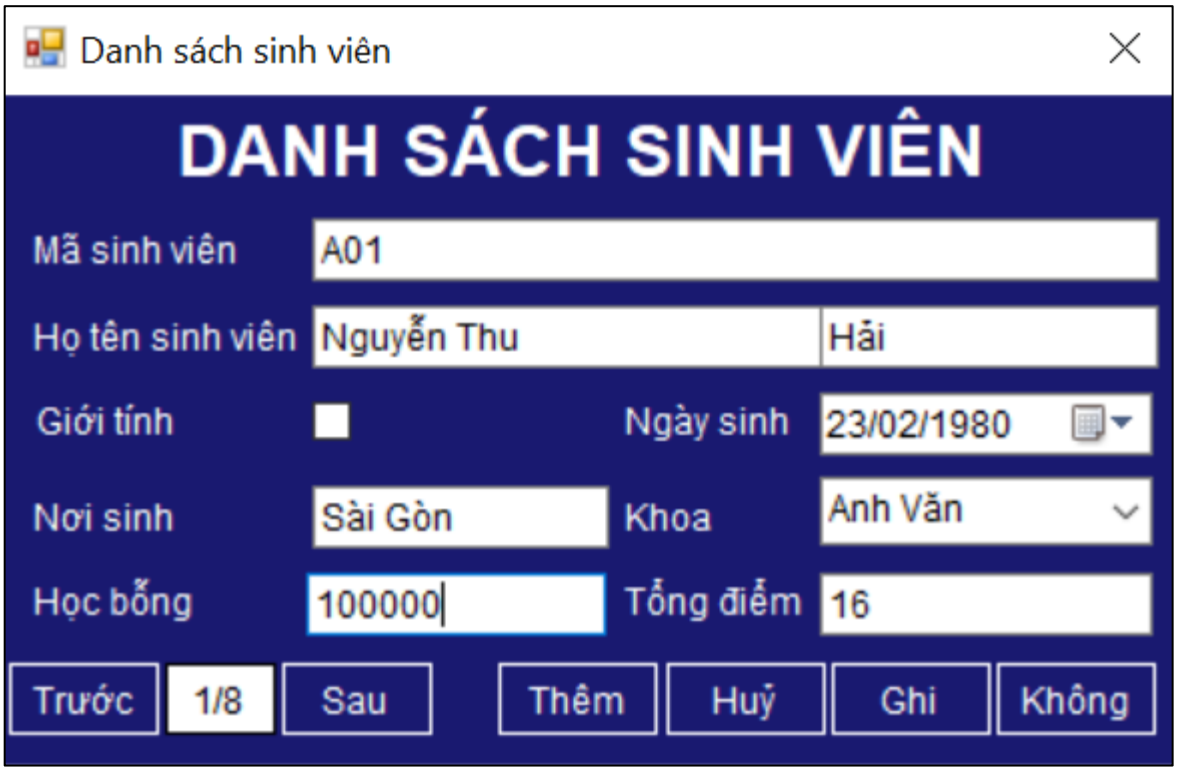
16.6. Khi người dùng Click chọn nút lệnh Ghi thì cho phép ghi lại sự thay đổi của mẫu tin sau khi hiệu chỉnh hay thêm mới vào DataTable tương ứng với bảng MONHOC trong DataSet. Mẫu tin sau khi ghi phải được cập nhật về CSDL.

16.7. Khi người dùng Click chọn nút lệnh Không thì cho phép phục hồi lại mẫu tin hiện hành trên màn hình.

16.8. Khi người dùng Click chọn nút lệnh Thoat thì cho phép đóng màn hình. Yêu cầu người dùng xác nhận trước khi màn hình đóng lại.

16.9. Khi thực hiện các thao tác cần kiểm tra tính hợp lý, hợp lệ tương ứng với từng thao tác thực hiện.

17. Thiết kế, định dạng màn hình và các đối tượng điều khiển theo mô tả của hình sau:



Yêu cầu:

17.1. Thực hiện các thao tác sau:

+ Từ CSDL ứng với tập tin QLSV.mdb cho trước, sao chép tập tin này vào folder project.

+ Tạo đối tượng DataSet không định kiểu ds, đối tượng DataAdapter, CommandBuider và các phương thức FillSchema, Fill, Update để đọc và cập nhật dữ liệu từ CSDL đã cho ở trên vào các DataTable trong DataSet và ngược lại.

+ Tạo quan hệ khoá ngoại (Foreign key) giữa các DataTable tblKhoa và tblSinhvien thông qua DataColumn **MaKH** của các DataTable.

+ Tạo quan hệ khoá ngoại (Foreign key) giữa các DataTable tblSinhvien và tblketqua thông qua DataColumn **MaSV** của các DataTable.

+ Tạo quan hệ khoá ngoại (Foreign key) giữa các DataTable tblKetqua và tblMonhoc thông qua DataColumn **MaMH** của các DataTable.

+ Thiết lập nguồn dữ liệu và các thuộc tính liên quan cho Combobox với DataTable trong DataSet tương ứng với bảng KHOA trong CSDL.

17.2. Sử dụng các đối tượng BindingSource và tập hợp DataBindings để liên kết dữ liệu trong DataSet với các điều khiển trên màn hình.

17.3. Sử dụng phương thức Compute của DataTable tương ứng với bảng KETQUA để tính và hiển thị tổng điểm thi của sinh viên hiện hành.

17.4. Khi người dùng Click chọn các nút lệnh Trước, Sau thì cho phép di chuyển mẫu tin hiện hành về ngay mẫu tin phía trước, phía sau tương ứng với các phương thức MovePrevious, MoveNext của đối tượng BindingSource. Xuất thông báo lỗi khi không thể di chuyển được.

17.5. Khi người dùng Click chọn nút lệnh Thêm thì cho phép thêm mới một mẫu tin tương ứng với một sinh viên.

17.6. Khi người dùng Click chọn nút lệnh Huỷ thì cho phép huỷ mẫu tin hiện hành trên màn hình. Yêu cầu người dùng xác nhận trước khi huỷ mẫu tin. Mẫu tin sau khi huỷ phải được cập nhật về CSDL.

17.7. Khi người dùng Click chọn nút lệnh Ghi thì cho phép ghi lại sự thay đổi của mẫu tin sau khi hiệu chỉnh hay thêm mới vào DataTable tương ứng với bảng SINHVIEN trong DataSet. Mẫu tin sau khi ghi phải được cập nhật về CSDL.

17.8. Khi người dùng Click chọn nút lệnh Không thì cho phép phục hồi lại mẫu tin hiện hành trên màn hình.

17.9. Khi người dùng đóng màn hình, yêu cầu người dùng xác nhận trước khi màn hình đóng lại.

17.10. Khi thực hiện các thao tác cần kiểm tra tính hợp lý, hợp lệ tương ứng với từng thao tác thực hiện.

18. Thiết kế, định dạng màn hình và các đối tượng điều khiển theo mô tả của hình sau:

Yêu cầu:

18.1. Thực hiện các thao tác sau:

+ Từ CSDL ứng với tập tin QLSV.mdb cho trước, sao chép tập tin này vào folder project.

+ Tạo đối tượng DataSet không định kiểu ds, đối tượng DataAdapter, CommandBuider và các phương thức FillSchema, Fill, Update để đọc và cập nhật dữ liệu từ CSDL đã cho ở trên vào các DataTable trong DataSet và ngược lại.

+ Tạo quan hệ khoá ngoại (Foreign key) giữa các DataTable tblKhoa và tblSinhvien thông qua DataColumn **MaKH** của các DataTable.

+ Tạo quan hệ khoá ngoại (Foreign key) giữa các DataTable tblSinhvien và tblketqua thông qua DataColumn **MaSV** của các DataTable.

+ Tạo quan hệ khoá ngoại (Foreign key) giữa các DataTable tblKetqua và tblMonhoc thông qua DataColumn **MaMH** của các DataTable.

+ Thiết lập nguồn dữ liệu và các thuộc tính liên quan cho Combobox với DataTable trong DataSet tương ứng với bảng KHOA trong CSDL.

18.2. Sử dụng các đối tượng BindingSource và tập hợp DataBindings để liên kết dữ liệu trong DataSet với các điều khiển trên màn hình và định dạng lại dữ liệu cho các điều khiển hiển thị dữ liệu học bổng và ngày sinh.

18.3. Sử dụng đối tượng Binding với các sự kiện Format, Parse của điều khiển để hiển thị Phái dưới dạng Nam / Nữ tương ứng với giá trị True / False lưu trữ trong CSDL.

18.4. Sử dụng phương thức Compute của DataTable tương ứng với bảng KETQUA để tính và hiển thị tổng điểm thi của sinh viên hiện hành.

18.5. Khi người dùng Click chọn các nút lệnh Trước, Sau thì cho phép di chuyển mẫu tin hiện hành về ngay mẫu tin phía trước, phía sau tương ứng với các phương thức MovePrevious, MoveNext của đối tượng BindingSource. Xuất thông báo lỗi khi không thể di chuyển được.

18.6. Khi người dùng Click chọn nút lệnh Thêm thì cho phép thêm mới một mẫu tin tương ứng với một sinh viên.

18.7. Khi người dùng Click chọn nút lệnh Huỷ thì cho phép huỷ mẫu tin hiện hành trên màn hình. Yêu cầu người dùng xác nhận trước khi huỷ mẫu tin. Mẫu tin sau khi huỷ phải được cập nhật về CSDL.

18.8. Khi người dùng Click chọn nút lệnh Ghi thì cho phép ghi lại sự thay đổi của mẫu tin sau khi hiệu chỉnh hay thêm mới vào DataTable tương ứng với bảng SINHVIEN trong DataSet. Mẫu tin sau khi ghi phải được cập nhật về CSDL.

18.9. Khi người dùng Click chọn nút lệnh Không thì cho phép phục hồi lại mẫu tin hiện hành trên màn hình.

18.10. Khi người dùng đóng màn hình, yêu cầu người dùng xác nhận trước khi màn hình đóng lại.

18.11. Khi thực hiện các thao tác cần kiểm tra tính hợp lý, hợp lệ tương ứng với từng thao tác thực hiện.

19. Từ tập tin QLSV.mdb được cung cấp, thực hiện chuyển đổi dữ liệu và đưa vào SQL Server và đặt tên cho CSDL là QLSINHVIEN. Thực hiện lại các yêu cầu của bài tập số 13 với CSDL QLSINHVIEN trong SQL Server.

20. Từ CSDL QLSINHVIEN trong SQL Server. Thực hiện lại các yêu cầu của bài tập số 14 với CSDL QLSINHVIEN trong SQL Server.

21. Từ CSDL QLSINHVIEN trong SQL Server. Thực hiện lại các yêu cầu của bài tập số 15 với CSDL QLSINHVIEN trong SQL Server.

22. Từ CSDL QLSINHVIEN trong SQL Server. Thực hiện lại các yêu cầu của bài tập số 16 với CSDL QLSINHVIEN trong SQL Server.

23. Từ CSDL QLSINHVIEN trong SQL Server. Thực hiện lại các yêu cầu của bài tập số 17 với CSDL QLSINHVIEN trong SQL Server.

24. Từ CSDL QLSINHVIEN trong SQL Server. Thực hiện lại các yêu cầu của bài tập số 18 với CSDL QLSINHVIEN trong SQL Server.

25. Thực hiện lại các yêu cầu của bài tập số 13 với CSDL QLSINHVIEN trong SQL Server với DataSet có định kiểu (Typed DataSet).

26. Thực hiện lại các yêu cầu của bài tập số 14 với CSDL QLSINHVIEN trong SQL Server với DataSet có định kiểu (Typed DataSet).

27. Thực hiện lại các yêu cầu của bài tập số 15 với CSDL QLSINHVIEN trong SQL Server với DataSet có định kiểu (Typed DataSet).

28. Thực hiện lại các yêu cầu của bài tập số 16 với CSDL QLSINHVIEN trong SQL Server với DataSet có định kiểu (Typed DataSet).

29. Thực hiện lại các yêu cầu của bài tập số 17 với CSDL QLSINHVIEN trong SQL Server với DataSet có định kiểu (Typed DataSet).

30. Thực hiện lại các yêu cầu của bài tập số 18 với CSDL QLSINHVIEN trong SQL Server với DataSet có định kiểu (Typed DataSet).

31. Thiết kế, định dạng màn hình và các đối tượng điều khiển theo mô tả của hình sau:

Yêu cầu:

31.1. Thực hiện các thao tác sau:

+ Sử dụng phương thức Add của tập hợp Rows để thêm trực tiếp các mẫu tin minh hoạ trên màn hình vào DataGridView.

+ Sử dụng các thuộc tính có liên quan đến các thành phần của DataGridView để định dạng theo mô tả của màn hình bao gồm: định dạng thành phần tiêu đề cột, định dạng dòng chẵn (lẻ), dòng được chọn trong DataGridView.

31.2. Khi người dùng Click chọn nút lệnh Thêm thì cho phép thêm mới một mẫu tin tương ứng với một môn học.

31.2. Khi người dùng Click chọn nút lệnh Xoá thì sử dụng các phương thức của DataGridView để huỷ mẫu tin hiện hành trên lưới và di chuyển về mẫu tin đầu tiên trên lưới. Yêu cầu người dùng xác nhận trước khi huỷ mẫu tin.

31.3. Khi người dùng Click chọn nút lệnh Ghi thì cho phép ghi lại sự thay đổi của mẫu tin sau khi hiệu chỉnh hay thêm mới vào DataGridView.

31.4. Khi người dùng Click chọn nút lệnh Không thì cho phép phục hồi lại mẫu tin hiện hành trên màn hình.

31.5. Khi thực hiện các thao tác cần kiểm tra tính hợp lý, hợp lệ tương ứng với từng thao tác thực hiện.

32. Thiết kế, định dạng màn hình và các đối tượng điều khiển theo mô tả của hình sau:

Tra cứu sinh viên

DANH SÁCH SINH VIÊN

Mã SV	Họ Sinh viên	Tên Sinh viên	Phái	Ngày sinh	Nơi sinh	Mã khoa	Học bổng
A01	Nguyễn Ngoan	Cường	☑	5/6/1971	Hà Nội	AV	50000
A02	Lý Anh	Huy	☑	1/1/1975	TP.HCM	AV	80000
A03	Lê Khắc	Dung	☐	8/12/1974	Bình Định	TH	75000
A04	Đinh Hữu	Chính	☑	5/25/1977	Cà Mau	AV	80000
B01	Văn Thành	Nho	☑	7/5/1971	Cần Giờ	AV	80000
B02	Nguyễn Văn	Chính	☑	1/23/1974	Mỹ Tho	AV	75000
B03	Trần Thị Yến	Nhi	☐	11/23/1...	Cần Thơ	TH	80000
B04	Nguyễn Thành	Khiêm	☑	4/30/1975	TP.HCM	TH	60000
S06	Đoàn Thanh	Mai	☐	9/25/1976	An Giang	SV	80000
T07	Lê Ngọc Diễm	Lệ	☐	1/26/1975	TP.HCM	TH	80000
V05	Nguyễn Khắc	Định	☑	4/12/1976	Tây Ninh	VL	80000
V09	Mai Văn	Dũng	☑	1/12/1977	Sa Đéc	VL	80000
*			☐				

32.1. Thực hiện các thao tác sau:

+ Từ CSDL ứng với tập tin QLSV.mdb cho trước, sao chép tập tin này vào folder project.

+ Tạo đối tượng DataSet có định kiểu ds, đối tượng TableAdapter và phương thức Fill để đọc dữ liệu từ CSDL đã cho ở trên vào DataTable trong DataSet tương ứng với bảng SINHVIEN trong CSDL.

32.2. Sử dụng các thuộc tính có liên quan đến các thành phần của DataGridView để định dạng theo mô tả của màn hình bao gồm: định dạng thành phần tiêu đề cột, định dạng dòng chẵn (lẻ), dòng được chọn trong DataGridView.

32.3.Thiết lập nguồn dữ liệu cho đối tượng DataGridView với thuộc tính DataSource từ DataTable tương ứng với dữ liệu bảng SINHVIEN trong DataSet.

33. Thực hiện lại bài tập 31 với đối tượng DataSet có định kiểu lấy từ CSDL QLSINHVIEN lưu trữ trong SQL Server. Sử dụng đối tượng BindingSource để thiết lập nguồn dữ liệu cho lưới và liên kết với các điều khiển khác trên màn hình. Sau khi thực hiện các thao tác thêm, sửa, xoá thì dữ liệu phải cập nhật về CSDL.

34. Thiết kế, định dạng màn hình và các đối tượng điều khiển theo mô tả của hình sau:

Yêu cầu:

34.1. Thực hiện các thao tác sau:

+ Từ CSDL ứng với tập tin QLSV.mdb cho trước, sao chép tập tin này vào folder project.

+ Tạo đối tượng DataSet không định kiểu ds, đối tượng DataAdapter, CommandBuider và các phương thức FillSchema, Fill, Update để đọc và cập nhật dữ liệu từ CSDL đã cho ở trên vào các DataTable trong DataSet và ngược lại.

+ Tạo quan hệ khoá ngoại (Foreign key) giữa các DataTable tblKhoa và tblSinhvien thông qua DataColumn **MaKH** của các DataTable.

+ Sử dụng các thuộc tính để định dạng DataGridView bao gồm: định dạng tiêu đề cột, định dạng dòng chẵn (lẻ), dòng được chọn trong DataGridView.

34.2. Sử dụng các đối tượng BindingSource để thiết lập nguồn dữ liệu cho các điều khiển Textbox, BindingNavigator và DataGridView trên màn hình.

34.3. Khi người dùng thay đổi khoa hiện hành với BindingNavigator thì các mẫu tin sinh viên tương ứng thuộc về khoa cũng thay đổi theo.

34.4. Khi người dùng Click chọn nút lệnh Thêm thì cho phép thêm mới một mẫu tin tương ứng với một khoa.

34.5. Khi người dùng Click chọn nút lệnh Xoá thì cho phép xoá mẫu tin hiện hành trên màn hình. Yêu cầu người dùng xác nhận trước khi huỷ mẫu tin. Mẫu tin sau khi huỷ phải được cập nhật về CSDL.

34.6. Khi người dùng Click chọn nút lệnh Ghi thì cho phép ghi lại sự thay đổi của mẫu tin sau khi hiệu chỉnh hay thêm mới vào DataTable tương ứng với bảng KHOA trong DataSet. Mẫu tin sau khi ghi phải được cập nhật về CSDL.

34.7. Khi người dùng Click chọn nút lệnh Không thì cho phép phục hồi lại mẫu tin hiện hành trên màn hình.

34.8. Khi người dùng Click chọn nút lệnh Thoát để đóng màn hình, yêu cầu người dùng xác nhận trước khi màn hình đóng lại.

34.9. Khi thực hiện các thao tác cần kiểm tra tính hợp lý, hợp lệ tương ứng với từng thao tác thực hiện.

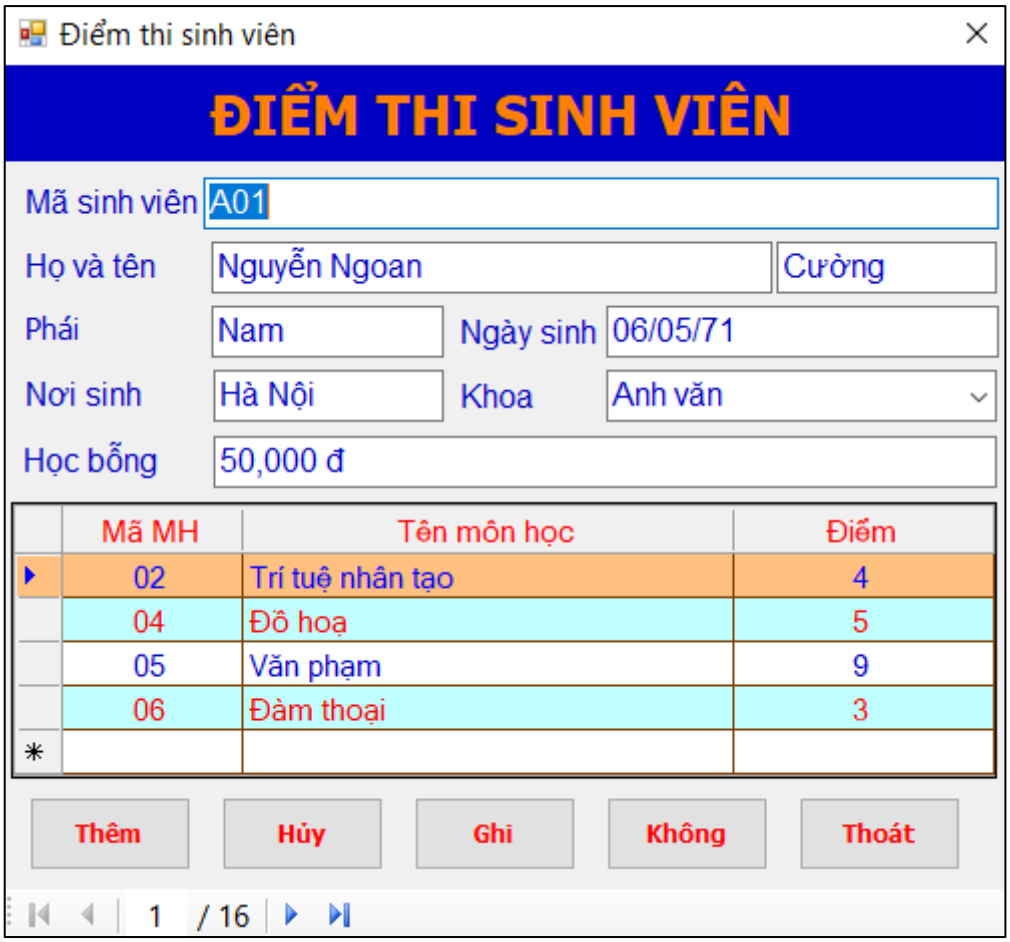
35. Thực hiện lại bài tập 34 theo các yêu cầu sau:

+ Từ CSDL ứng với tập tin QLSV.mdb cho trước, sao chép tập tin này vào folder project.

+ Tạo đối tượng DataSet có định kiểu ds, đối tượng TableAdapter và phương thức Fill để đọc dữ liệu từ CSDL đã cho ở trên vào DataTable trong DataSet tương ứng với bảng SINHVIEN trong CSDL.

+ Các chức năng còn lại thực hiện tương tự như bài 34.

36. Thiết kế, định dạng màn hình và các đối tượng điều khiển theo mô tả của hình sau:



Yêu cầu:

36.1. Thực hiện các thao tác sau:

+ Từ CSDL ứng với tập tin QLSV.mdb cho trước, sao chép tập tin này vào folder project.

+ Tạo đối tượng DataSet không định kiểu ds, đối tượng DataAdapter, CommandBuider và các phương thức FillSchema, Fill, Update để đọc và cập nhật dữ liệu từ CSDL đã cho ở trên vào các DataTable trong DataSet và ngược lại.

+ Tạo quan hệ khoá ngoại (Foreign key) giữa các DataTable tblKhoa và tblSinhvien thông qua DataColumn **MaKH** của các DataTable.

+ Tạo quan hệ khoá ngoại (Foreign key) giữa các DataTable tblSinhvien và tblketqua thông qua DataColumn **MaSV** của các DataTable.

+ Tạo quan hệ khoá ngoại (Foreign key) giữa các DataTable tblKetqua và tblMonhoc thông qua DataColumn **MaMH** của các DataTable.

+ Sử dụng các thuộc tính để định dạng DataGridView bao gồm: định dạng tiêu đề cột, định dạng dòng chẳn (lẻ), dòng được chọn trong DataGridView.

36.2. Sử dụng các đối tượng BindingSource để thiết lập nguồn dữ liệu cho các điều khiển Textbox, BindingNavigator và DataGridView trên màn hình.

36.3. Sử dụng các tập hợp DataBindings và đối tượng Binding và các phương thức sự kiện Format, Parse để liên kết và định dạng các điều khiển trên màn hình.

36.4. Khi người dùng thay đổi sinh viên hiện hành với BindingNavigator thì các điểm thi của sinh viên tương ứng cũng thay đổi theo.

36.5. Khi người dùng Click chọn nút lệnh Thêm thì cho phép thêm mới một mẫu tin tương ứng với một sinh viên.

36.6. Khi người dùng Click chọn nút lệnh Huỷ thì cho phép huỷ mẫu tin hiện hành trên màn hình chính. Yêu cầu người dùng xác nhận trước khi huỷ mẫu tin. Mẫu tin sau khi huỷ phải được cập nhật về CSDL.

36.7. Khi người dùng Click chọn nút lệnh Ghi thì cho phép ghi lại sự thay đổi của mẫu tin sau khi hiệu chỉnh hay thêm mới vào DataTable tương ứng với bảng SINHVIEN trong DataSet. Mẫu tin sau khi ghi phải được cập nhật về CSDL.

36.8. Khi người dùng Click chọn nút lệnh Không thì cho phép phục hồi lại sinh viên hiện hành trên màn hình chính.

36.9. Khi người dùng Click chọn nút lệnh Thoát để đóng màn hình, yêu cầu người dùng xác nhận trước khi màn hình đóng lại.

36.10. Khi thực hiện các thao tác cần kiểm tra tính hợp lý, hợp lệ tương ứng với từng thao tác thực hiện.

CHƯƠNG 4. THIẾT KẾ BÁO CÁO

Sau khi học xong chương này, sinh viên có khả năng:

- *Trình bày được ý nghĩa và công dụng của báo cáo trong xây dựng ứng dụng.*
- *Trình bày được ý nghĩa, công dụng của các thành phần trong các loại báo cáo.*
- *Thiết kế, kết xuất báo cáo với các điều khiển trên màn hình theo yêu cầu.*

4.1. Tổng quan về thiết kế báo cáo

4.1.1. Khái niệm

Báo cáo là thành phần không thể thiếu khi xây dựng ứng dụng. Báo cáo cho phép kết xuất, thống kê và định dạng dữ liệu kết xuất ra màn hình hay máy in theo những khuôn mẫu khác nhau. **Thí dụ:** Hoá đơn thanh toán ở siêu thị, phiếu điểm của sinh viên, phiếu nhập – xuất, phiếu thu – chi, hoá đơn thu tiền điện thoại, …

4.1.2. Các loại báo cáo thường dùng

Một số loại báo cáo thường được sử dụng trong ứng dụng bao gồm

- **Báo cáo dạng cột:** là báo cáo trình bày dữ liệu theo dạng cột, thông tin của một mẫu tin nằm trên một cột. **Thí dụ:** Thông tin của các môn học được trình bày theo báo cáo dạng cột như sau

Hình 4.1. Báo cáo dạng cột

– **Báo cáo dạng bảng:** là báo cáo trình bày dữ liệu theo dạng dòng cột, báo cáo theo dạng này có thể trình bày dữ liệu nhiều hơn tại một thời điểm. **Thí dụ:** Thông tin về các môn học được trình bày theo báo cáo dạng bảng như sau

Danh mục môn học

Mã môn học	Tên môn học	Số Tiết
01	Triết Học Đông Phương	30
02	Toán Cao Cấp 1	60
03	Toán Cao Cấp 2	60
04	Vật Lý ĐạI Cương	25
05	Cơ Sở Dữ Liệu	45

Hình 4.2. Báo cáo dạng bảng

– **Báo cáo dạng phân nhóm:** là dạng báo cáo có thể trình bày đồng thời dữ liệu ở bảng này và dữ liệu ở bảng có quan hệ với nhau. **Thí dụ:** Trình bày thông tin về khoa và các sinh viên thuộc khoa có liên quan.

KẾT QUẢ HỌC TẬP SINH VIÊN

Ngày in 26 tháng 06 năm 2007

Mã số sinh viên A01
Họ tên sinh viên Nguyễn Ngoan Cường
Phái sinh viên Nam **Ngày Sinh** 06/05/1972
Học tại khoa: AV Anh Văn

Số TT	Mã môn	Tên môn học	Số Tiết	Điểm
1	01	Triết Học Đông Phương	30	8.5
2	02	Toán Cao Cấp 1	60	7

Điểm trung bình 7.75
Kết quả học tập Đậu

Hình 4.3. Báo cáo dạng phân nhóm

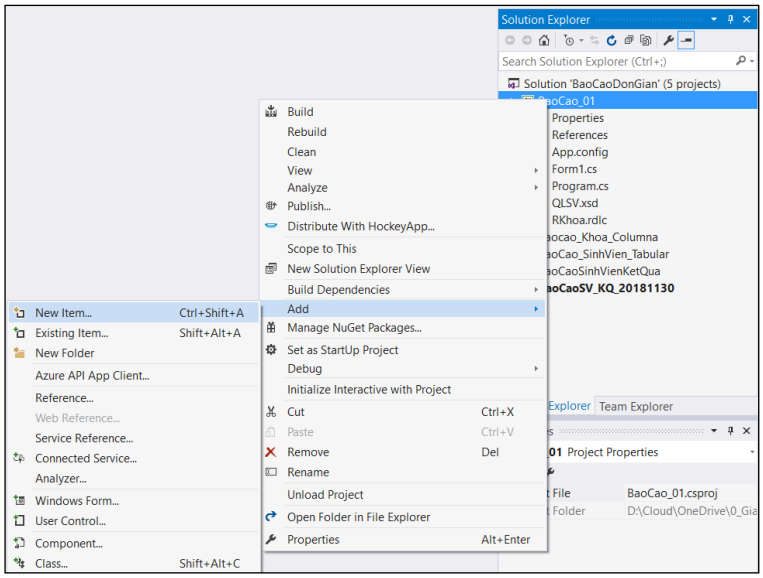
– **Báo cáo dạng Main-Sub:** là loại báo cáo có kết xuất tương tự như báo cáo phân nhóm nhưng sử dụng kỹ thuật nhúng báo cáo trong một báo cáo khác.

4.1.3. Thiết kế báo cáo

Để thiết kế một báo cáo, thực hiện theo trình tự các bước như sau:

– **Bước 1:** Chuẩn bị nguồn dữ liệu là một Dataset có định kiểu.

– **Bước 2:** R-Click vào Project ⇒ Chọn Add ⇒ Chọn New Item như hình sau

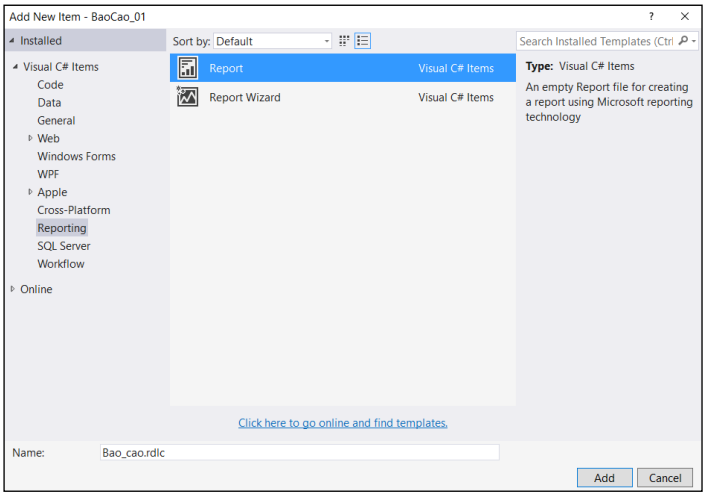


Hình 4.4. Thêm đối tượng mới

– **Bước 3:**

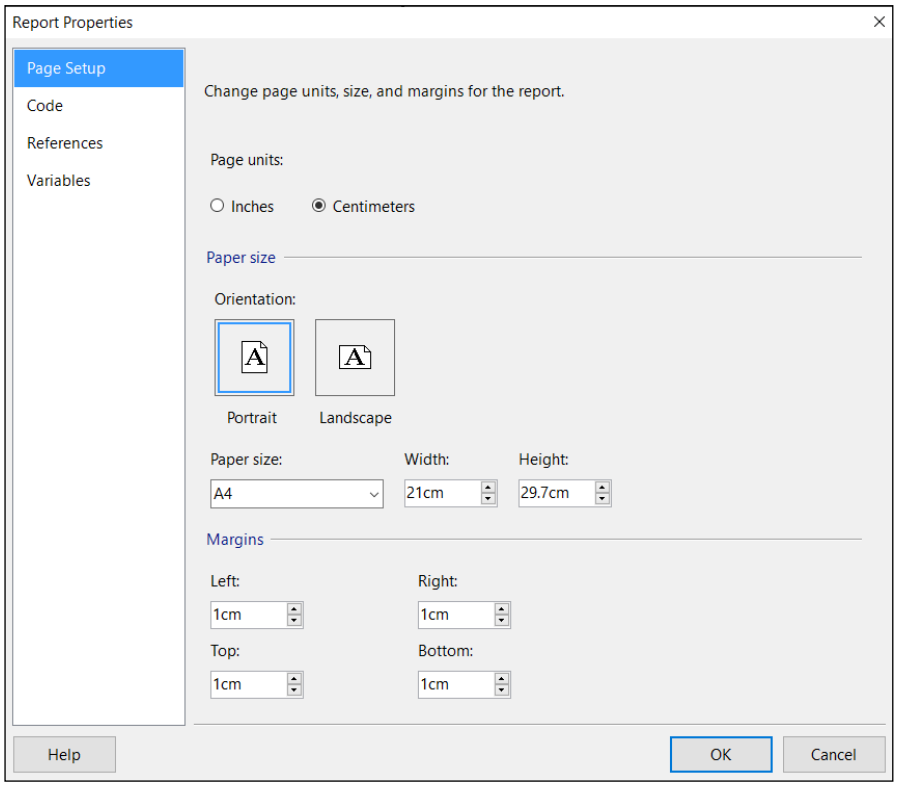
+ **Tại cửa sổ bên trái:** chọn Reporting
+ **Tại cửa sổ bên phải:** Chọn Report
+ **Tại Name:** Đặt tên cho báo cáo

⇒ Click nút lệnh **Add.**



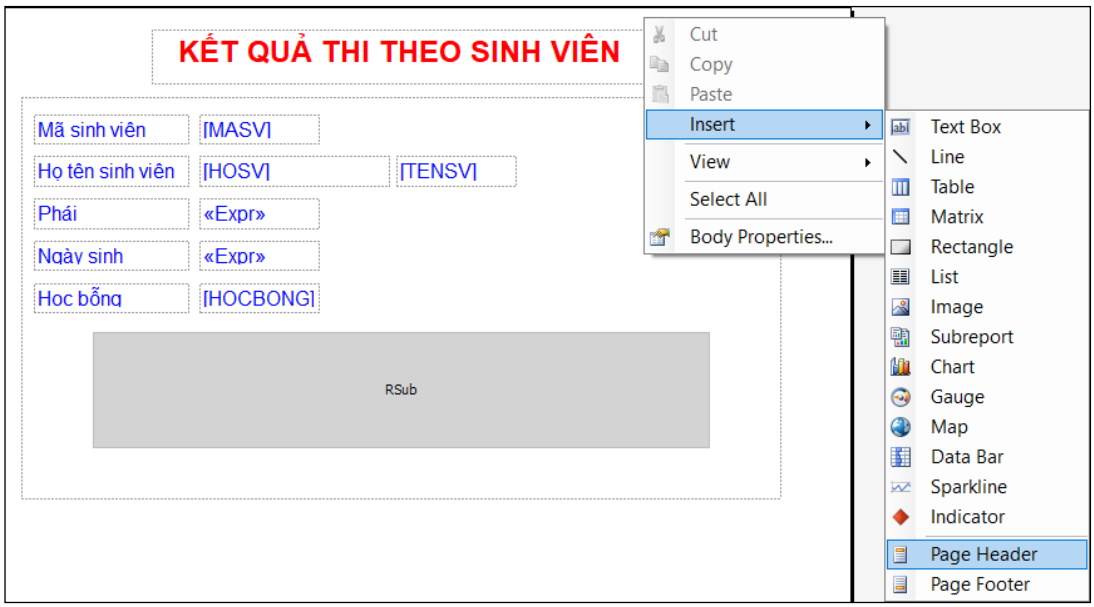
Hình 4.5. Chọn đối tượng báo cáo

– **Bước 4:** Tại cửa sổ thiết kế ⇒ Click thực đơn Report ⇒ Chọn Report Properties ⇒ Chọn Page Setup để thiết lập khổ giấy và các lề trên, lề dưới, lề trái, lề phải cho báo cáo.



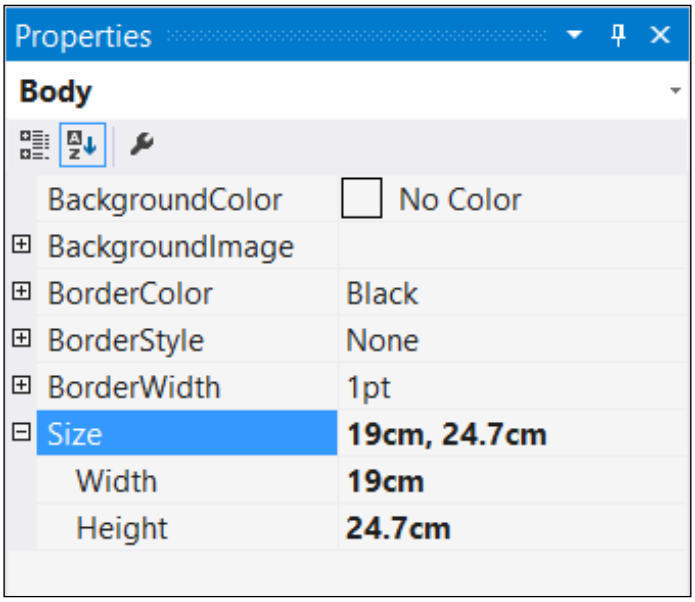
Hình 4.6. Thiết lập khổ và lề giấy cho báo cáo

– **Bước 5:** Để thêm tiêu đề đầu trang (Page header), tiêu đề cuối trang vào báo cáo ⇒ R-Click vào báo cáo ⇒ Chọn Insert Page Header / Page Footer.



Hình 4.7. Thiết lập tiêu đề trang cho báo cáo

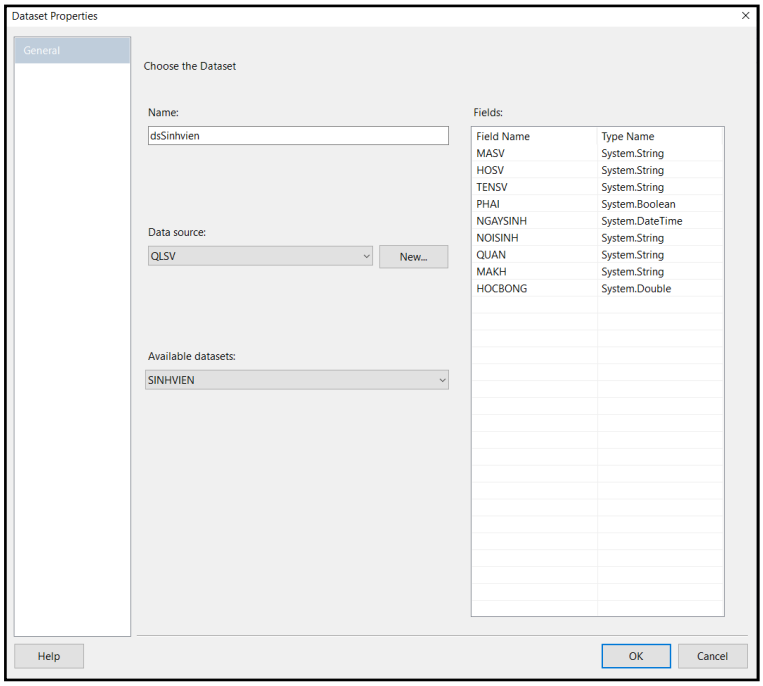
– **Bước 6:** Hiệu chỉnh lại kích thước của tiêu đề đầu trang, tiêu đề cuối trang và phần thân của báo cáo trong cửa sổ Property.



Hình 4.8. Hiệu chỉnh kích thước các thành phần của báo cáo

– **Bước 7:** Thiết lập nguồn dữ liệu cho báo cáo. Chọn điều khiển List hay Table đưa vào cửa sổ thiết kế ⇒ Tại cửa sổ Dataset Properties ⇒ Chọn:

+ **Name:** đặt tên cho Dataset sử dụng trong báo cáo.
+ **Datasource:** chọn nguồn dữ liệu là một Dataset có định kiểu có trong Project.
+ **Available Datasets:** chọn bảng chứa dữ liệu hiển thị trong báo cáo.



Hình 4.9. Thiết lập nguồn dữ liệu cho báo cáo

– **Bước 8:** Tiến hành thiết kế báo cáo theo khuôn dạng cụ thể.

4.1.4. Các điều khiển thường dùng trong thiết kế báo cáo

Các điều khiển thường dùng khi thiết kế báo cáo được đặt trong thanh Toolbox. Cụ thể là:

– **Textbox:** được sử dụng để tạo các nhãn và dữ liệu hiển thị trên báo cáo.

– **Line:** được sử dụng để vẽ đường thẳng.

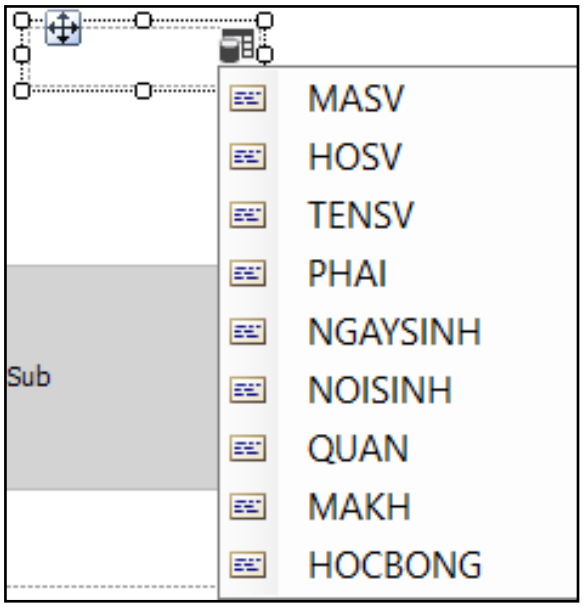
– **Table:** được sử dụng để thiết kế báo cáo dạng bảng.

– **List:** được sử dụng để thiết kế báo cáo dạng cột.

– **Subreport:** được sử dụng để thiết kế báo cáo dạng Main-Sub.

4.1.5. Thiết lập biểu thức trong báo cáo

– **Hiển thị dữ liệu của một cột trong Textbox:** Click vào góc phải trên của điều khiển và chọn cột dữ liệu cần hiển thị.



Hình 4.10. Chọn nguồn dữ liệu cho điều khiển

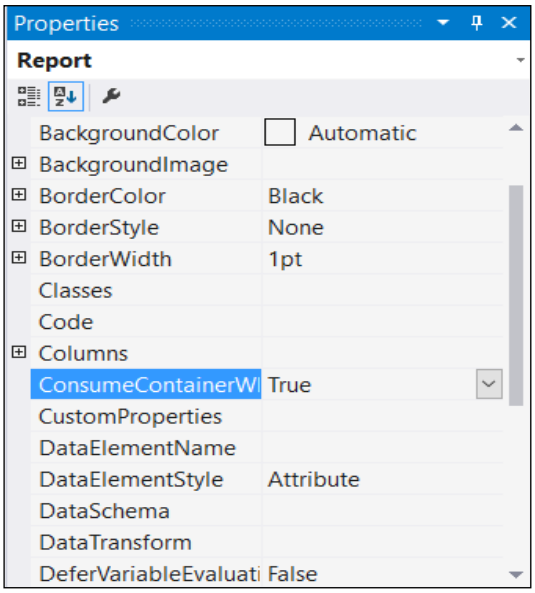
– **Hiển thị số thứ tự:** Tạo Textbox tại vị trí muốn hiển thị số thứ tự ⇒ R-Click điều khiển ⇒ Chọn Expression ⇒ Tạo biểu thức có dạng sau:

```
=Format(RowNumber(nothing),"00")
```

– **Đánh số trang:** Tạo Textbox trong Page Header (Page Footer) ⇒ R-Click điều khiển ⇒ Chọn Expression ⇒ Tạo biểu thức có dạng sau:

```
="Trang " & Globals!PageNumber & "/" Globals!TotalPages
```

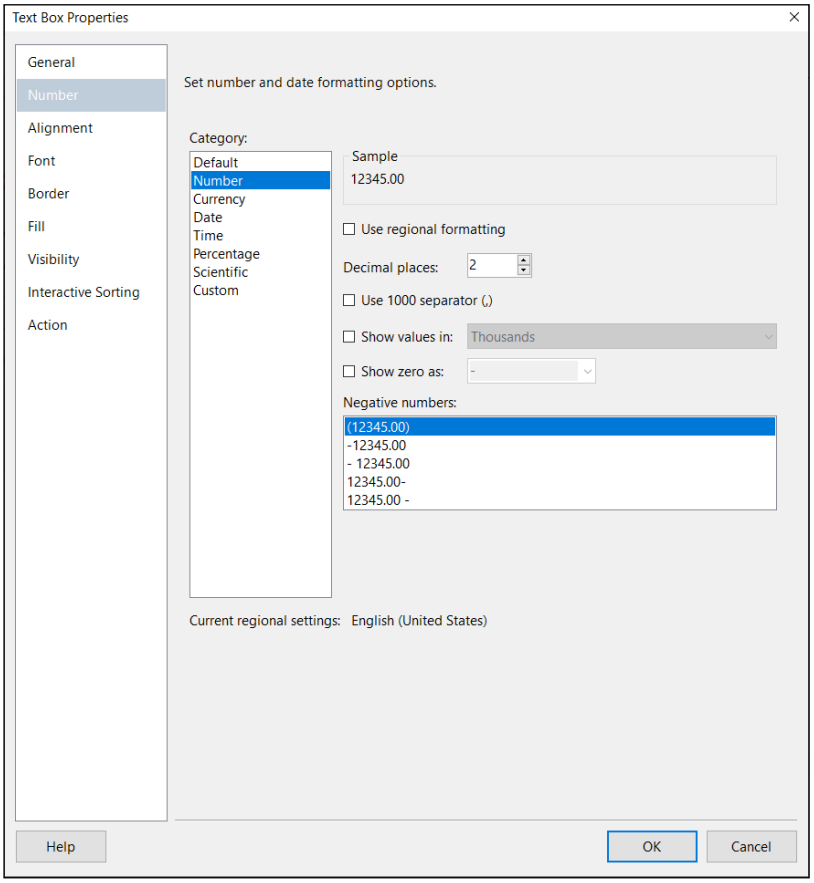
– **Loại bỏ trang rỗng trong báo cáo:** Tại cửa sổ Properties ⇒ Chọn đối tượng Report ⇒ Tại ConsumeContainerWhiteSpace ⇒ Chọn true.



Hình 4.11. Loại bỏ trang trống trong báo cáo

4.1.6. Định dạng điều khiển Textbox trong báo cáo:

Để định dạng điều khiển Textbox ⇒ R-Click Textbox ⇒ Chọn Textbox Property ⇒ Chọn loại dữ liệu (số, ngày tháng, …) cần định dạng ⇒ Tiến hành định dạng theo yêu cầu.



Hình 4.12. Định dạng điều khiển trong báo cáo

4.2. Các thành phần trong một báo cáo

- **Thành phần tiêu đề đầu báo cáo (Report header):** chứa các thông tin xuất hiện ở đầu báo cáo. Thí dụ như: tên công ty, tên đơn vị, tên báo cáo, …
- **Thành phần tiêu đề cuối báo cáo (Report footer):** chứa các thông tin xuất hiện ở cuối báo cáo. Thí dụ như: các thống kê trên toàn bộ báo cáo, ngày tháng của báo cáo, chữ ký, ….
- **Thành phần tiêu đề đầu trang (Page header):** chứa các thông tin xuất hiện ở đầu mỗi trang của báo cáo. Thí dụ như: logo của công ty, tiêu đề của các cột cần thống kê, ….
- **Thành phần tiêu đề cuối trang (Page footer):** chứa các thông tin xuất hiện ở cuối mỗi trang của báo cáo. Thí dụ như: số trang, …
- **Thành phần hiển thị dữ liệu (Body):** thường được dùng để hiển thị dữ liệu.
- **Thành phần tiêu đề đầu nhóm (Group header):** chứa thông tin xuất hiện đầu mỗi nhóm.
- **Thành phần tiêu đề cuối nhóm (Group footer):** chứa thông tin xuất hiện cuối mỗi nhóm

4.3. Thiết kế các loại báo cáo

4.3.1. Thiết kế báo cáo dạng cột (Columna)

- **Bước 1:** Chuẩn bị nguồn dữ liệu.
- **Bước 2:** Tạo mới báo cáo và các thành phần tiêu đề trang thích hợp.
- **Bước 3:** Đưa điều khiển List vào thân báo cáo và lần lượt đưa các điều khiển Textbox vào vị trí thích hợp và gán nguồn dữ liệu, định dạng thích hợp cho điều khiển.

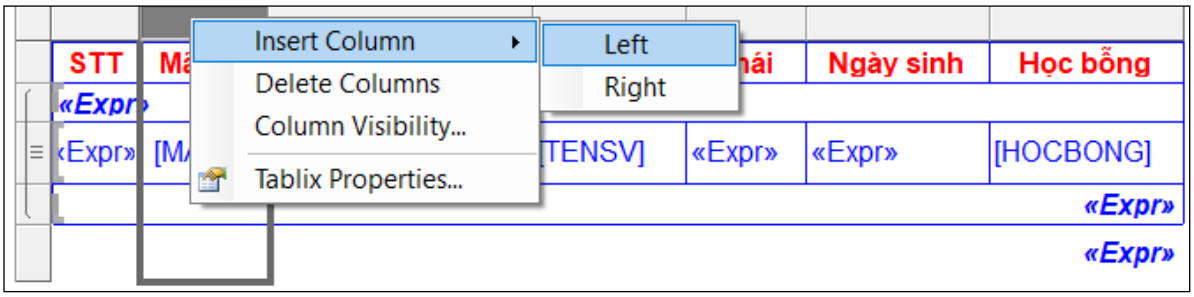
4.3.2. Thiết kế báo cáo dạng bảng (Tabular)

- **Bước 1:** Chuẩn bị nguồn dữ liệu.
- **Bước 2:** Tạo mới báo cáo và các thành phần tiêu đề trang thích hợp.
- **Bước 3:** Đưa điều khiển Table vào thân báo cáo và lần lượt đưa nguồn dữ liệu vào các cột của bảng. Sau đó, định dạng thích hợp cho các điều khiển trong điều khiển Table.

Chú ý: Trong điều khiển Table, để thêm, xoá cột hay thống kê dữ liệu trong điều khiển, ta có thể thực hiện như sau:

– **Thêm cột:** R-Click vào vị trí cột cần thêm ⇒ Chọn Insert Column ⇒ Chọn Left nếu muốn thêm một cột bên trái cột đánh dấu chọn, chọn Right nếu muốn thêm cột bên phải.

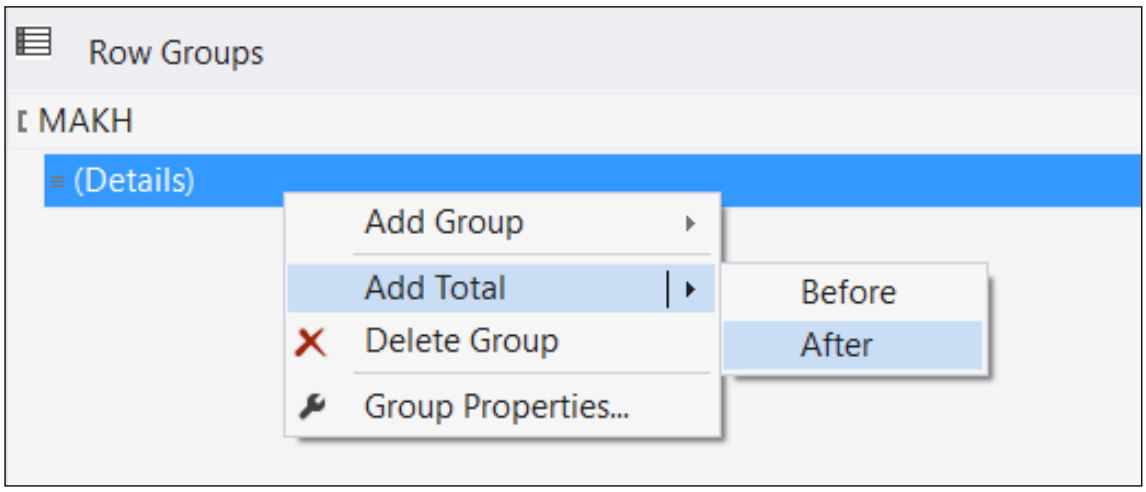
– **Xoá cột:** R-Click chọn cột cần xoá ⇒ Chọn Delete Columns.



Hình 4.13. Thêm, xoá cột trong báo cáo dạng bảng

– **Thêm dòng thống kê dữ liệu cho toàn báo cáo:**

+ **Tại thực đơn Report:** Chọn View ⇒ Grouping
+ **Tại Row Groups:** R-Click vào Details ⇒ Chọn Add Total ⇒ Chọn Before hay After để thêm dòng thống kê vào đầu hay cuối báo cáo ⇒ Thiết lập biểu thức với các hàm thống kê.



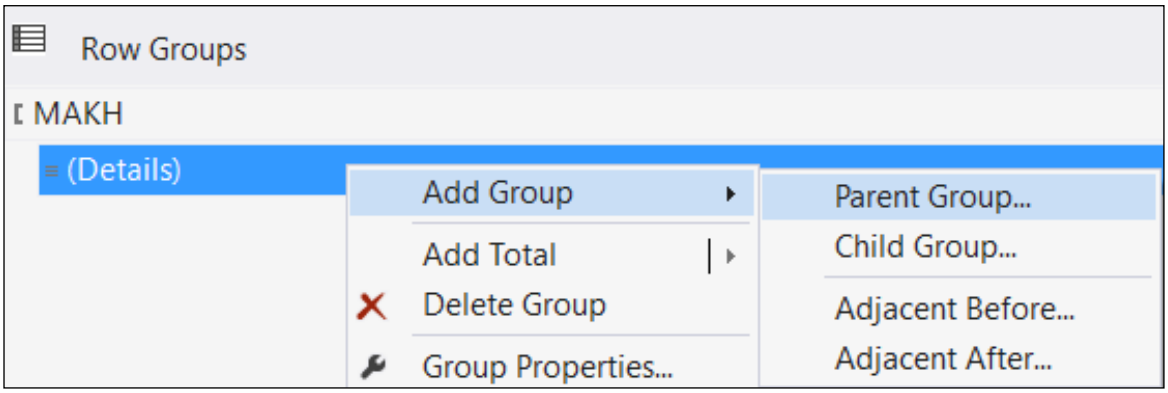
Hình 4.14. Tạo dòng thống kê cho báo cáo

4.3.3. Thiết kế báo cáo dạng phân nhóm

– **Bước 1:** Thiết kế tương tự như báo cáo dạng cột và dạng bảng.

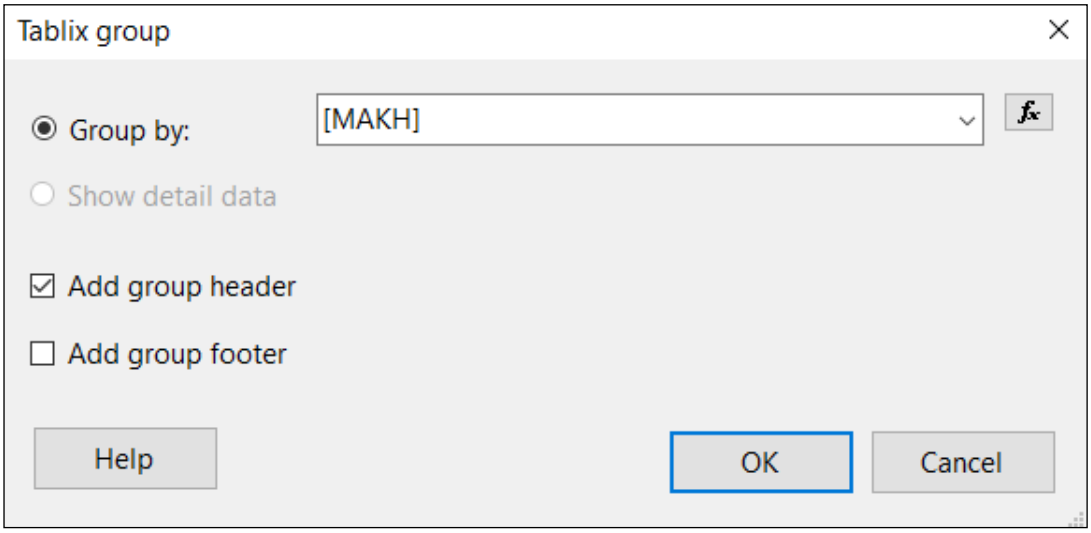
– **Bước 2:** Thêm nhóm vào báo cáo. Để thêm nhóm, thực hiện như sau:

+ **Tại thực đơn Report:** Chọn View ⇒ Grouping
+ **Tại Row Groups:** R-Click vào Details ⇒ Chọn Add Group ⇒ Chọn Parent Group



Hình 4.15. Tạo nhóm cho báo cáo

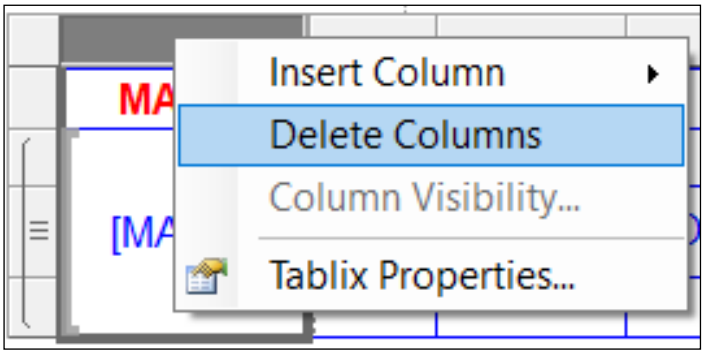
+ **Tại Group by**: Chọn cột dùng làm tiêu chuẩn nhóm. Có thể đánh dấu chọn để hiển thị tiêu đề đầu nhóm hay cuối nhóm (nếu cần) tại Add Group header và Add Group footer.



Hình 4.16. Chọn thành phần nhóm cho báo cáo

– **Một số chú ý:**

+ Mặc định, cột sử dụng làm tiêu chuẩn nhóm sẽ hiển thị trên báo cáo. Nếu không muốn hiển thị, có thể đánh dấu chọn cột ⇒ R-Click ⇒ Chọn Delete Columns để xoá**.**

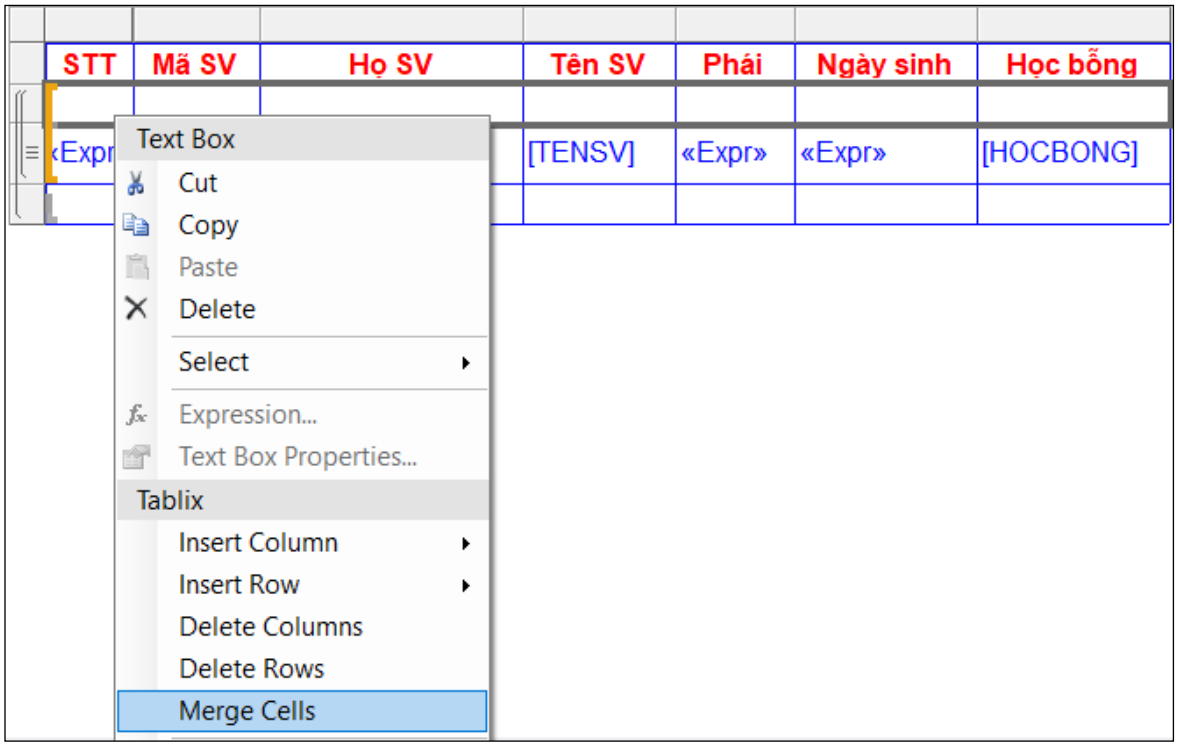


Hình 4.17. Xoá cột là thành phần nhóm trên báo cáo

+ Để hiển thị cột số thứ tự theo từng nhóm, sử dụng hàm RowNumber với tham số là chuỗi chứa tên cột dùng làm tiêu chuẩn phân nhóm theo cú pháp sau:

```
=Format(RowNumber("<Tên cột làm tiêu chuẩn phân nhóm>"),"00")
```

+ Để tiêu đề nhóm hiển thị trên toàn bộ dòng ⇒ Có thể trộn các ô bằng cách đánh dấu chọn các ô ⇒ R-Click ⇒ Chọn Merge cells.



Hình 4.18. Trộn các ô trên báo cáo

– **Bước 3:** tiếp tục thực hiện như thiết kế báo cáo dạng cột và dạng bảng.

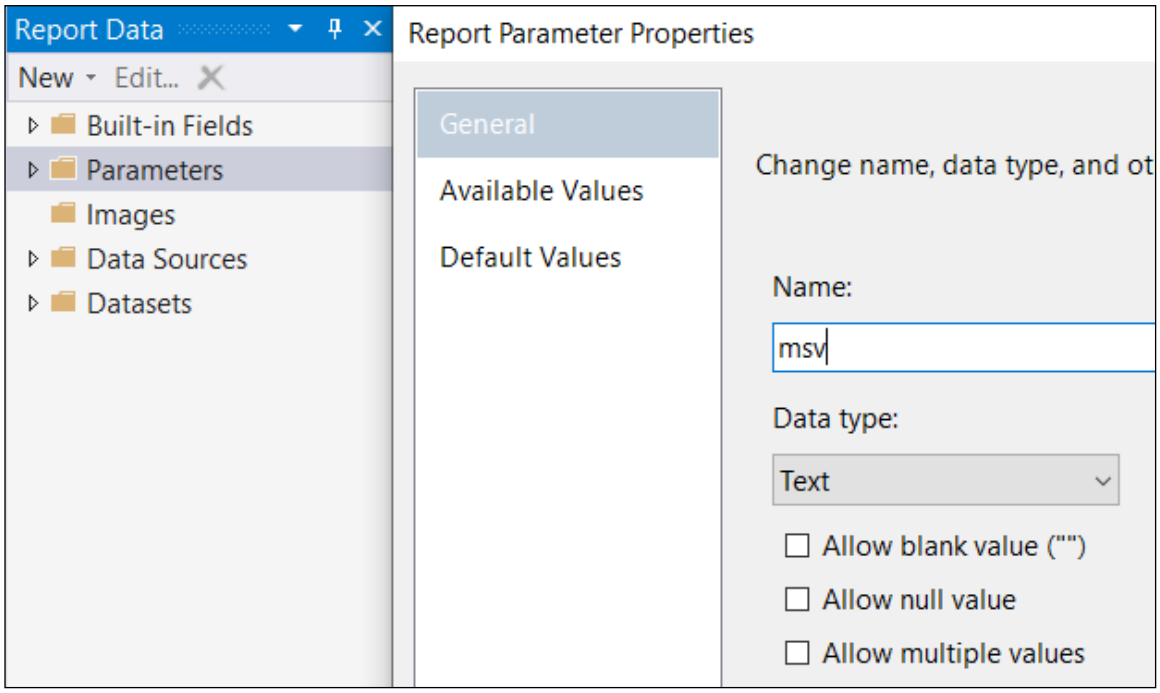
4.3.4. Thiết kế báo cáo dạng Main-Sub

– **Bước 1:** Thiết kế Main report tương tự thiết kế báo cáo dạng cột.

– **Bước 2:**

+ Thiết kế Sub report tương tự như thiết kế báo cáo dạng bảng (sử dụng Table) và thực hiện phân nhóm. Cho hiển thị tiêu đề cuối nhóm (Group footer) nếu cần.

+ Tạo tham số (parameter): Vào thực đơn View/ Report Data ⇒ Chọn New ⇒ Parameter ⇒ Tại Name: Đặt tên cho tham số; Tại Data Type: quy định kiểu dữ liệu cho tham số => Chọn OK.



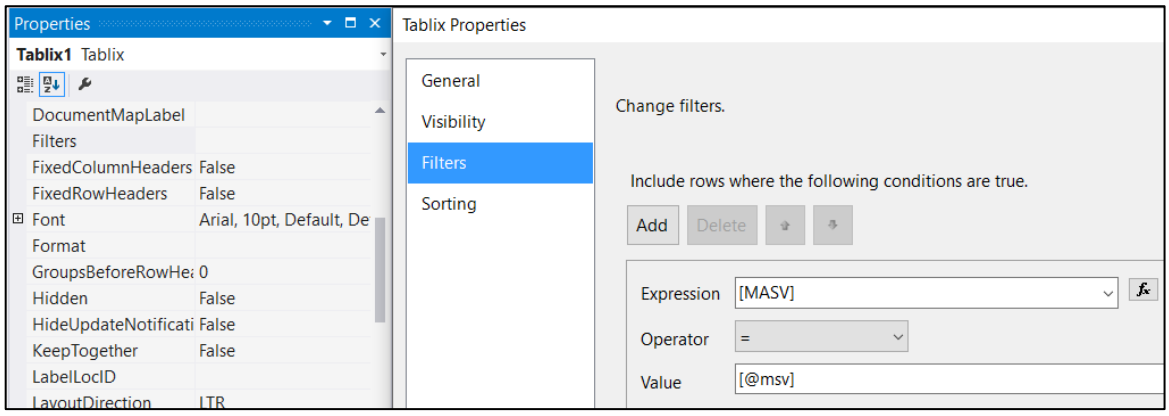
Hình 4.19. Thiết lập tham số cho báo cáo con

+ Chọn điều khiển Table ⇒ Trong cửa sổ Properties ⇒ Tại thuộc tính Filters, Click biểu tượng ... .

  Tại Expression: chọn thuộc tính dùng để liên kết.

  Tại Operator: chọn phép toán liên kết.

  Tại Value: chọn tên tham số liên kết với thuộc tính.



Hình 4.20. Thiết lập tham số cho báo cáo cha

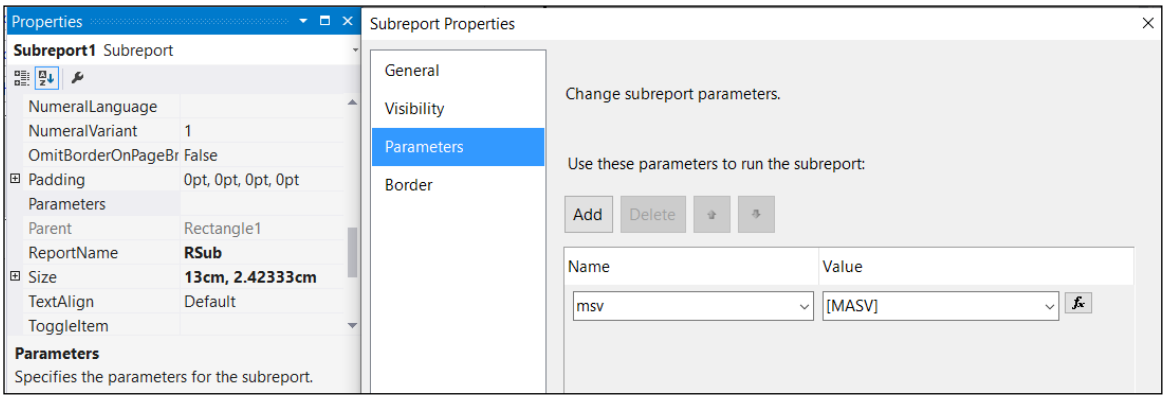
– **Bước 3:** Nhúng Sub report vào Main report.

  + Tại cửa sổ thiết kế Main report ⇒ kéo điều khiển Subreport từ Toolbox vào Main report.

  + Đánh dấu chọn điều khiển Subreport ⇒ Properties ⇒ thiết lập các thuộc tính sau:

- ReportName: là tên của Sub report.

- Parameters: Click biểu tượng  . Tại Parameters ⇒ Chọn Add ⇒ Tại **Name:** Nhập tên parameter tạo trong Sub report; Tại **Value:** nhập tên thuộc tính tương ứng.



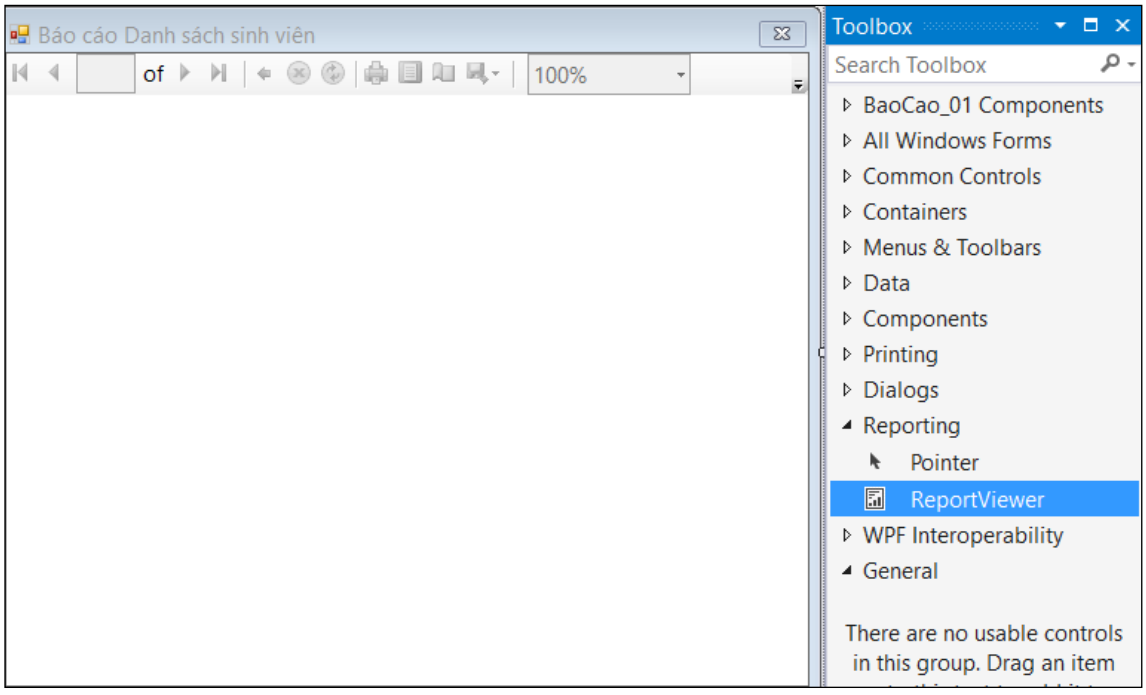
Hình 4.21. Lấy tham số trong báo cáo

4.4. Thiết kế màn hình và lập trình kết xuất báo cáo

4.4.1. Thiết kế màn hình

Để kết xuất báo cáo ra màn hình hay máy in, thực hiện thiết kế màn hình theo các bước sau:

– **Bước 1:** Tạo mới một màn hình ⇒ Trên thanh Toolbox ⇒ Chọn mục Reporting ⇒ Chọn điều khiển ReportViewer và đưa vào màn hình.



Hình 4.22. Màn hình chứa báo cáo

- **Bước 2:** Định dạng điều khiển ReportViewer với một số thuộc tính và giá trị thường dùng sau:
  - **Dock:** với giá trị fill để báo cáo sẽ được kết xuất toàn màn hình.
  - **ShowToolbar:** hiển thị hay không hiển thị thanh điều khiển in ấn.
  - ShowPrintButton: hiển thị / không hiển thị nút in báo cáo ra máy in.

4.4.2. Lập trình kết xuất và hiển thị báo cáo

- **Bổ sung không gian tên (NameSpace):** Microsoft.Reporting.Winform
- **Khai báo các đối tượng sử dụng:** có các đối tượng Dataset, DataAdapter, ReportDatasource, …
- **Các thí dụ minh hoạ**

Thí dụ 4.1

```
QLSV ds = new QLSV();
BindingSource KhoaBindingSource = new BindingSource();
QLSVTableAdapters.KHOATableAdapter dtpkh = new
QLSVTableAdapters.KHOATableAdapter();
...
ReportDataSource rdsKhoa = new ReportDataSource();
rdsKhoa.Name = "DSReport";
rdsKhoa.Value = KhoaBindingSource;
rv.LocalReport.DataSources.Add(rdsKhoa);
rv.LocalReport.ReportEmbeddedResource="BaoCao_01.RKhoa.rdlc";
dtpkh.Fill(ds.KHOA);
this.rv.RefreshReport();
```

Thí dụ 4.2

+ Khai báo các đối tượng cần sử dụng

```
QLSV ds = new QLSV();
QLSVTableAdapters.SINHVIENTableAdapter adpSV = new
QLSVTableAdapters.SINHVIENTableAdapter();
QLSVTableAdapters.KETQUATableAdapter dptKQ = new
QLSVTableAdapters.KETQUATableAdapter();
QLSVTableAdapters.MONHOCTableAdapter dtpMH = new
QLSVTableAdapters.MONHOCTableAdapter();
BindingSource bdsSV = new BindingSource();
BindingSource bdsKQ = new BindingSource();
ReportDataSource rds = new ReportDataSource();
```

```
ReportViewer rv = new ReportViewer();
...
```

+ Thiết kế một màn hình có một nút lệnh btnIN. Tại phương thức sự kiện Click tương ứng với nút lệnh này, thực hiện đoạn mã lệnh sau

Thí dụ 4.3

```
private void btnIN_Click(object sender, EventArgs e)
{
   bdsSV.DataSource = ds;
   bdsSV.DataMember = "SINHVIEN";
   bdsKQ.DataSource = ds;
   bdsKQ.DataMember = "KETQUA";
   dtpSV.Fill(ds.SINHVIEN);
   dtpKQ.Fill(ds.KETQUA);
   dtpMH.Fill(ds.MONHOC);
   ...

   Form frm = new Form();
   frm.WindowState = FormWindowState.Maximized;
   int pos=bdsSV.Position;
   bdsSV.Filter = "MASV='" + txtDK.Text + "'";
   bdsKQ.Filter = bdsSV.Filter;
   if (bdsKQ.Count == 0)
   {
       MessageBox.Show("Không có điểm thi");
       bdsSV.RemoveFilter();
       bdsKQ.Filter = bdsSV.Filter;
       bdsSV.Position = pos;
       return;
   }

   rds.Name = "dsSinhVienKetQua";
   rds.Value = bdsSV;

   rv.Dock = DockStyle.Fill;
   rv.LocalReport.DataSources.Add(rds);
   rv.LocalReport.ReportEmbeddedResource                                    =
"BaoCaoSV_KQ_20181130.RSV.rdlc";
   frm.Controls.Add(rv);
```

```
    // Lệnh sau phát sinh phương thức sự kiện xử lý Subreport
    rv.LocalReport.SubreportProcessing                                    +=
LocalReport_SubreportProcessing;
    rv.RefreshReport();
    frm.ShowDialog();


    bdsSV.Filter = "";
    bdsSV.Position = pos;
}
```

+ Tại phương thức sự kiện SubreportProcessing, thực hiện đoạn mã lệnh sau

Thí dụ 4.4

```
void          LocalReport_SubreportProcessing(object          sender,
SubreportProcessingEventArgs e)
{
    e.DataSources.Add(new        ReportDataSource("dsSinhvienKetqua",
bdsKQ));
}
```

BÀI TẬP CHƯƠNG 4

37. Thiết kế, định dạng và trình bày báo cáo theo mẫu sau:

Danh mục môn học

Mã môn học	01
Tên môn học	Triết Học Đông Phương
Số Tiết	30
Mã môn học	02
Tên môn học	Toán Cao Cấp 1
Số Tiết	60

38. Thiết kế, định dạng và trình bày báo cáo theo mẫu sau:

Danh mục môn học

Mã môn học	Tên môn học	Số Tiết
01	Triết Học Đông Phương	30
02	Toán Cao Cấp 1	60
03	Toán Cao Cấp 2	60
04	Vật Lý ĐạI Cương	25
05	Cơ Sở Dữ Liệu	45

39. Thiết kế, định dạng và trình bày báo cáo theo mẫu sau:

Danh Sách Sinh Viên

Tên khoa	*Mã sinh viên*	*Họ tên sinh viên*	*Phái*	*Ngày sinh*	*Học bổng*
Anh Văn	A04	Trần anh Tuấn	Nam	20/12/1977	80000
Anh Văn	B02	Trần thị thu Thủy	Nữ	02/01/1977	0
Tin Học	A01	Nguyễn thị Hải	Nữ	23/02/1977	130000
Tin Học	A02	Trần văn Chính	Nam	24/12/1977	150000
Tin Học	A03	Lê thu bạch Yến	Nữ	21/02/1977	170000
Triết	B01	Trần thanh Mai	Nữ	12/08/1977	0

40. Thiết kế, định dạng và trình bày báo cáo theo mẫu sau:

Danh Sách Sinh Viên

Tên khoa	Mã sinh viên	Họ tên sinh viên	Phái	Ngày sinh	Học bổng
Anh Văn					
	A04	Trần anh Tuấn	Nam	20/12/1977	80000
	B02	Trần thị thu Thủy	Nữ	02/01/1977	0
Tin Học					
	A01	Nguyễn thị Hải	Nữ	23/02/1977	130000
	A02	Trần văn Chính	Nam	24/12/1977	150000
	A03	Lê thu bạch Yến	Nữ	21/02/1977	170000
Triết					
	B01	Trần thanh Mai	Nữ	12/08/1977	0

41. Thiết kế, định dạng và trình bày báo cáo theo mẫu sau:

Điểm Thi Của Sinh Viên

Mã sinh viên A01 ***Ngày sinh*** 23/02/1977

Họ tên sinh viên Nguyễn thị Hải ***Giới tính*** Nữ

Stt	Tên môn	Điểm
01	Truyền tin	5.0
02	Trí tuệ nhân tạo	6.0
03	Cơ sở dữ liệu	3.0

Điểm trung bình 4.7

42. Thiết kế, định dạng và trình bày báo cáo theo mẫu sau:

KẾT QUẢ HỌC TẬP SINH VIÊN

Ngày in 26 tháng 06 năm 2007

Mã số sinh viên A01

Họ tên sinh viên Nguyễn Ngoan Cường

Phái sinh viên Nam **Ngày Sinh** 06/05/1972

Học tại khoa: AV Anh Văn

Số TT	Mã môn	Tên môn học	Số Tiết	Điểm
1	01	Triết Học Đông Phương	30	8.5
2	02	Toán Cao Cấp 1	60	7

Điểm trung bình 7.75

Kết quả học tập Đậu

TÀI LIỆU THAM KHẢO

[1] V.Rob Miles, C# programming, Department of Computer Science University of Hull, 2001.

[2] Phạm Hữu Khang, C# - Lập trình Cơ sở dữ liệu, Nhà xuất bản Lao động Xã hội, 2006.

[3] Nguyễn Ngọc Bình Phương, Thái Thanh Phong, Các giải pháp lập trình C#, Nhà xuất bản Giao thông Vận tải, 2006.

[4] Nguyễn Ngọc Minh, Crystal Reports Developer, Nhà xuất bản Phương đông, 2006.

[5] Wei-Meng Lee, C# 2008 Programmer's reference, Wiley, 2008.