TRƯỜNG ĐẠI HỌC HÀNG HẢI VIỆT NAM

KHOA CÔNG NGHỆ THÔNG TIN

THUYẾT MINH
ĐỀ TÀI NCKH CẤP TRƯỜNG

ĐỀ TÀI

XÂY DỰNG BỘ SIÊU DỮ LIỆU ỨNG DỤNG ĐỂ TỰ ĐỘNG TẠO CÁC BÁO CÁO THỐNG KÊ KẾT XUẤT VỚI CƠ SỞ DỮ LIỆU

Chủ nhiệm đề tài: TS. TRẦN THỊ HƯƠNG

Thành viên tham gia: K.S Nguyễn Cao Văn

Hải Phòng, tháng 05 /2015

Mục lục

Mở đầu

Khi máy tính chỉ được sử dụng bởi các chuyên gia thì thiết kế giao diện người dùng chưa được chú trọng đến. Ngày nay, người dùng máy tính đã đa dạng hơn, máy tính và các ứng dụng của nó được dùng rộng rãi cả trong và ngoài lĩnh vực công nghệ thông tin. Do vậy, giao diện người dùng nên được thiết kế theo các nhu cầu và dự định của người dùng để họ thực hiện nhiệm vụ. Người dùng sẽ hài lòng dẫn đến tinh thần và năng suất làm việc của họ sẽ được nâng cao khi được làm với hệ thống máy tính có giao diện người dùng dễ hiểu, dễ dùng. Một giao diện người dùng được thiết kế khó hiểu, không dễ dùng có thể khiến cho người dùng bất mãn và thất vọng trong công việc dẫn đến giảm năng suất làm việc.

Nghiên cứu, thiết kế hệ thống kết xuất gồm báo biểu và giao diện sao cho chúng có khả năng đáp ứng được những ý định mà người dùng muốn thực hiện ngay trong lúc họ muốn tương tác với máy tính để thực hiện nhiệm vụ là một trong những vấn đề đang được quan tâm nghiên cứu hiện nay. Mặc dù đã có nhiều hỗ trợ từ phía các nhà sản xuất phần mềm và sự nỗ lực của các nhà lập trình, nhưng việc thiết kế nên kết xuất như các báo biểu và các giao diện vẫn chưa linh động, mềm dẻo theo cấu trúc của cơ sở dữ liệu bên trong và theo ý muốn của người dùng muốn kết xuất theo từng thời điểm.

Vì vậy vấn đề đặt ra cho nghiên cứu, thiết kế và phát triển hệ thống kết xuất phải có hướng thiết kế sao cho kết xuất thích nghi được sự thay đổi cấu trúc bên trong của cơ sở dữ liệu và tùy biến được theo từng thời điểm sử dụng nhằm tăng tính tiện dụng cho sản phẩm.

Nhu cầu thiết kế hệ thống kết xuất thay đổi được kết xuất theo yêu cầu của người dùng trong từng ngữ cảnh sử dụng khác nhau và mềm dẻo theo cấu trúc bên trong của sơ sở dữ liệu là rất cần thiết và thu hút sự quan tâm hiện nay.

Mục đích của đề tài: nghiên cứu siêu dữ liệu mô tả cho bài toán tự động tạo báo biểu kết xuất với cơ sở dữ liệu và các kỹ thuật xử lý XML với C# để giải quyết bài toán đã nêu.

Xây dựng bộ siêu dữ liệu để mô tả các đối tượng:

- Siêu dữ liệu mô tả các thuộc tính của từng quan hệ và thuộc tính của cơ sở dữ liệu quan hệ để mô tả sự thay đổi về cấu trúc của các quan hệ và thuộc tính trong cơ sở dữ liệu quan hệ để mô tả sự thay đổi về cấu trúc của các quan hệ và thuộc tính trong cơ sở dữ liệu quan hệ.
- Siêu dữ liệu mô tả từng thuộc tính của quan hệ được kết xuất ra hệ thống kết xuất.
- Siêu dữ liệu mô tả từng thuộc tính của một báo biểu.

- Siêu dữ liệu để mô tả từng thuộc tính của các thành phần báo biểu kết xuất.

Ý nghĩa thực tiễn của đề tài: Sự thành công của đề tài sẽ đưa ra một phương pháp mớ để thiết kế hệ thống kết xuất với hình thức thể hiện có khả năng tùy biến được theo từng thời điểm sử dụng và không phụ thuộc vào cấu trúc dữ liệu của cơ sở dữ liệu quan hệ đầu vào mà nó kết nối. Hệ thống kết xuất được thiết kế theo phương pháp này sẽ đáp ứng được nhu cầu sử dụng của người dùng theo từng thời điểm với ngữ cảnh khác nhau và thay đổi về cấu trúc của cơ sở dữ liệu đầu vào.

Cấu trúc của báo cáo gồm 4 chương:

Chương 1. Cơ sở lý thuyết về siêu dữ liệu.

Chương 2. Xây dựng các bộ siêu dữ liệu .

Chương 3. Các kỹ thuật xử lý XML với C#.

Chương 4. Chương trình tự động lập báo biểu tự động kết xuất với cơ sở dữ liệu.

Chương 1. Cơ sở lý thuyết về siêu dữ liệu

1.1. Giới thiệu về siêu dữ liệu.

Siêu dữ liệu được xây dựng từ thông tin mà nó mô tả, giải thích, xác định. Hay nói cách khác là siêu dữ liệu được dùng để dễ dàng khôi phục, sử dụng hoặc quản lý nguồn thông tin. Siêu dữ liệu thường được gọi là dữ liệu mô tả về một nguồn thông tin nào đó.

Có ba loại chính của siêu dữ liệu:

- **Siêu dữ liệu mô tả(Descriptive metadata):** Mô tả nguồn tài nguyên cho mục đích khám phá và phân tích (chứng minh). Nó có thể bao gồm các yếu tố như: tiêu đề, tóm tắt, tác giả, ...Nó được dùng một cách công khai để có thể tới được các hệ thống tìm kiếm.
- **Siêu dữ liệu cấu trúc (Structural metadata):** Mô tả các liên kết giữa các đối tượng thông tin liên quan của tài liệu, hay nói cách khác dùng mô tả tổ chức logic của một tập hợp tệp. Ví dụ: mục lục, chương, phần, trang sách, hình ảnh minh họa, phụ lục,.. được liên kết với nhau như nào trong một tài liệu nhằm giúp người dùng dễ dàng di chuyển đến các thành phần của tài liệu. Ở đây có nghĩa là cuốn sách như 1 tổng thể, Structural metadata được sử dụng để ghi lại mối quan hệ giữa các file vật lý và các trang, các trang và các chương, giữa các chương và cuốn sách.
- **Siêu dữ liệu quản trị (Administrative metadata):**

Cung cấp thông tin giúp quản lý nguồn tài nguyên. Ví dụ: một nguồn tài nguyên được tạo ra khi nào, tạo ra như nào, các thông tin kỹ thuật khác của tài nguyên (vd : kích thước, độ phân giải), ai có thể truy cập nó.

Siêu dữ liệu có thể diễn tả nguồn tài nguyên ở bất kỳ mức độ nào. Nó có thể diễn tả một tập hợp, một nguồn tài nguyên đơn lẻ hoặc một phần của nguồn tài nguyên lớn. Siêu dữ liệu được gắn vào

1.2. Mục đích của siêu dữ liệu.

Một lý do quan trọng cho vệc tạo siêu dữ liệu miêu tả làm dễ dàng cho việc khám phá các thông tin có liên quan. Thêm vào đó là khám phá nguồn tài nguyên. Siêu dữ liệu giúp tổ chức nguồn tài nguyên điện tử được dễ dàng thao tác giữa các phần và tổng hợp lại, cung cấp con số xác thực hỗ trợ, duy trì.

1.3. Cấu trúc của siêu dữ liệu.

Các lược đồ siêu dữ liệu là tập hợp các thành phần siêu dữ liệu thiết kế cho mục đích đặc biệt như diễn tả một kiểu riêng biệt của nguồn tài nguyên. Định nghĩa hoặc nghĩa của các phần tử của chúng được biết như nghĩa của lược đồ. Định nghĩa hoặc nghĩa các yếu tố được hiểu là nghĩa của lược đồ. Giá trị của các phần tử siêu dữ liệu là nội dung của lược đồ. Tập các yếu tố siêu dữ liệu và các giá trị của nó dùng để mô tả một nguồn thông tin nào đó tạo thành 1 bản ghi (metadata record).

1.4. Tạo siêu dữ liệu.

Với câu hỏi "ai tạo siêu dữ liệu" thì câu trả lời khác nhau tùy theo ngành, nguồn tài nguyên mô tả. Nhiều cấu trúc cơ bản và siêu dữ liệu quản trị được cung cấp từ nhân viên kỹ thuật hoặc tạo đối tượng số, hoặc được sinh ra trong quá trình tự động. Đối với siêu dữ liệu miêu tả, người khởi tạo nguồn tài nguyên cung cấp thông tin là tốt nhất. Điều này đặc biệt đúng trong tài liệu của các tập dữ liệu khoa học vì những người này có sự hiểu biết về mối liên kết giữa dữ liệu và tài liệu mô tả.

Tuy nhiên, nhiều dự án đã chỉ ra rằng để có hiệu quả phải có sự sắp xếp thông thạo thông tin chuyên nghiệp để mô tả siêu dữ liệu bởi vì các tác giả hoặc người tạo dữ liệu không có thời gian hoặc kỹ năng.

1.5. Ứng dụng siêu dữ liệu để tạo liên kết động.

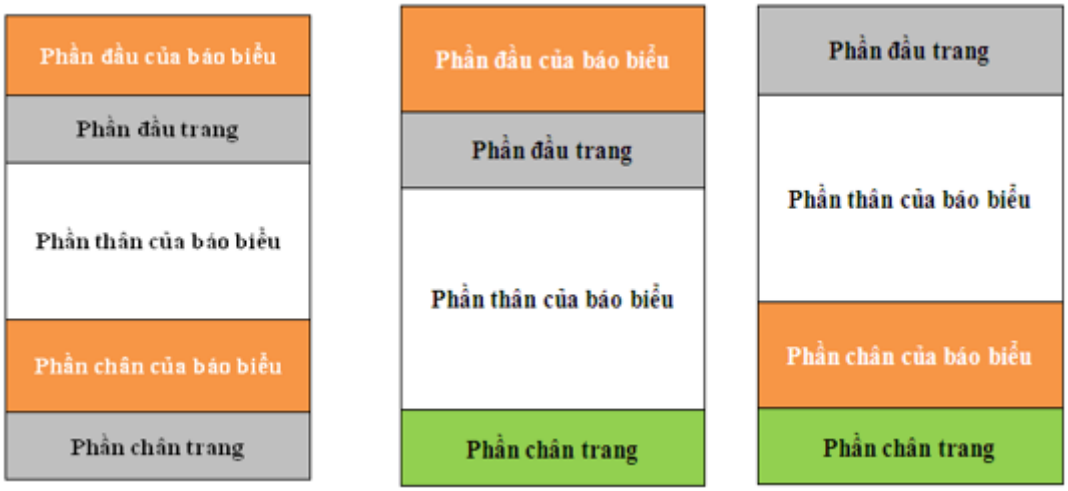
Với các ý nghĩa thiết thực của siêu dữ liệu, việc vận dụng siêu dữ liệu vào thiết kế hệ thống thông tin có ý nghĩa đặc biệt. Siêu dữ liệu giúp mô tả, quản lý, khai thác nguồn tài nguyên thông tin một cách hiệu quả. Trong giới hạn phạm vi của bài toán, nhóm tác giả áp dụng siêu dữ liệu để tạo liên kết động giữa các hệ thống kết xuất với cơ sở dữ liệu.

Chương 2. Xây dựng các bộ siêu dữ liệu

2.1. Cấu trúc của báo biểu.

Một báo biểu được trình bày sau khi thiết kế xong thường gồm 5 phần chính như sau:

1. Phần đầu của báo biểu,
2. Phần đầu trang,
3. Phần thân của báo biểu,
4. Phần chân của báo biểu,
5. Phần chân trang



2.2. Các nguyên tắt thiết kế báo biểu

a) **Cách sử dụng các đặc tính của font trong hiển thị thông tin.**

Một font có các đặc tính cơ bản như sau:

- Kiểu có chân (Serif), kiểu không chân (Sans Serif);
- Kiểu chữ: đậm, nghiêng;
- Cỡ chữ và;
- Loại chữ: in thường, IN HOA, Ký Tự Đầu Tiên Của Mỗi Từ Được In Hoa (mixed-case).

Được dùng như công cụ để:

✓ Tổ chức việc nối kết các phần tử,

✓ Nhận dạng các phần tử quan trọng trên màn hình,

✓ Thiết lập một thứ tự đọc và,

✓ Tạo một đặc điểm riêng biệt.

Dùng kết hợp các ***kiểu*** thì nên chọn:

- ✓ Kiểu không chân cho nhan đề và tiêu đề,
- ✓ Kiểu có chân cho phần thân.

b) **Cách sử dụng các đặc tính của font:**

- *Kiểu chữ nghiêng* được sử dụng khi:

- ✓ Muốn *nhấn mạnh* hoặc
- ✓ *Tập trung sự chú ý* trên màn hình

- Nên định dạng kiểu chữ nghiêng cho:

- ✓ *Một từ* hoặc
- ✓ *Một đoạn ngắn,*
- Không nên sử dụng cho một văn bản dài.
- Nên dùng *kết hợp* kiểu chữ thường và *kiểu chữ nghiêng*

- Về cỡ chữ:

- ✓ **Không nên** sử dụng **quá ba cỡ chữ** cùng lúc trên màn hình.

Khuyên dùng:

- ✓ Cỡ chữ từ **18 - 36** point cho nhan đề và tiêu đề,
- ✓ Cỡ chữ từ **12 - 14** point cho phần thân.

Sử dụng ***loại chữ*** trong các trường hợp sau:

- ✓ Dùng **mixed-case** cho hầu hết các thành phần: *nhãn, dữ liệu,* mô tả các *điều khiển,* thông tin *chỉ dẫn,* mô tả *hình, bảng*.
- ✓ Chữ **in hoa***:* sử dụng cho *nhan đề,* các *từ quan trọng* và cho *các tiêu đề*.
- ✓ Trường hợp kiểu chữ **in thường** thì *không* khuyên sử dụng cho *nhan đề*.

2.3. Cách thiết kế báo biểu.

a) *Cách thiết kế thân báo biểu*

- **Các tiêu đề cột:**

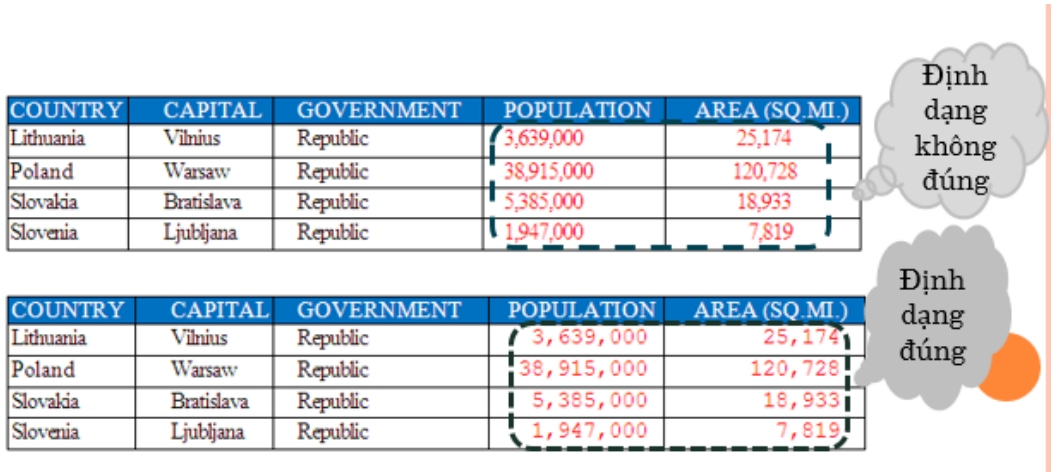
- ✓ Tiêu đề cột phải mô tả rõ nội dung của cột,
- ✓ Tránh các từ viết tắt trong tiêu đề cột.

b) *Cách thiết kế thân báo biểu*

- **Sử dụng font:**

- ✓ Dữ liệu kiểu số: *canh phải* và sử dụng kiểu *monospaced,*

✓ Dữ liệu kiểu ký tự: *canh trái*, phông chữ: Times New Roman, Arial, Verdana…



COUNTRY	CAPITAL	GOVERNMENT	POPULATION	AREA (SQ.MI.)
Lithuania	Vilnius	Republic	3,639,000	25,174
Poland	Warsaw	Republic	38,915,000	120,728
Slovakia	Bratislava	Republic	5,385,000	18,933
Slovenia	Ljubljana	Republic	1,947,000	7,819

COUNTRY	CAPITAL	GOVERNMENT	POPULATION	AREA (SQ.MI.)
Lithuania	Vilnius	Republic	3,639,000	25,174
Poland	Warsaw	Republic	38,915,000	120,728
Slovakia	Bratislava	Republic	5,385,000	18,933
Slovenia	Ljubljana	Republic	1,947,000	7,819

- Độ rộng thân báo biểu: thân một báo biểu quá rộng, có thể được thu hẹp bằng hai giải pháp:

✓ ***Giải pháp thứ nhất***: thu nhỏ hoặc mở rộng các cột để tất cả các thông tin có thể được nhìn thấy đồng thời,

✓ ***Một giải pháp khác:*** phân nội dung ô rộng thành nhiều dòng, như hình minh họa ở cột **LANGUAGES**:

COUNTRY	CAPITAL	GOVERNMENT	POPULATION	LANGUAGES
Czech Republic	Prague	Republic	10,320,000	Czech, Slovak
Estonia	Tallinn	Republic	1,450,000	Estonia, Russian
Hungary	Budapest	Republic	9,963,000	Hungarian
Latvia	Riga	Republic	2,452,000	Latvian, Russian
Lithuania	Vilnius	Republic	3,639,000	Lithuanian, Russian, Polish

c) *Cách thiết kế phần đầu của báo biểu*

- Một báo biểu thường sẽ gồm nhiều trang.

- Phần đầu của báo biểu: xuất hiện chỉ một lần ở trang đầu tiên của toàn báo biểu.

- Thông tin ở phần này thường gồm:

✓ Quốc hiệu;

✓ Tên cơ quan, tổ chức đặt ở gốc trên bên trái;

✓ Địa danh và ngày, tháng, năm lập báo biểu;

✓ Nhan đề cho báo biểu.

d) ***Cách thiết kế phần chân của báo biểu***

Phần chân của báo biểu là phần kết xuất thông tin xuất hiện chỉ một lần ở cuối báo biểu.

Ví dụ:

Các thông tin báo biểu kết xuất ở phần này thường gồm:

✓ Địa danh và ngày, tháng, năm lập báo biểu *(nếu phần đầu báo biểu chưa có);*

✓ Họ tên người lập báo biểu;

✓ Chức vụ, họ tên người ký báo biểu.

2.3. Ứng dụng siêu dữ liệu để tạo báo biểu

2.3.1. Siêu dữ liệu mô tả cho quan hệ và thuộc tính của quan hệ trong cơ sở dữ liệu.

tt	Loại thực thể con của "đối tượng"	Siêu dữ liệu mô tả
	Quan hệ	1. Tên quan hệ. 2. Khóa chính. 3. Số thuộc tính.
	Thuộc tính	1. Tên thuộc tính. 2. Kiểu dữ liệu. 3. Kích thước dữ liệu. 4. Số chữ số thập phân.

2.3.2. Siêu dữ liệu mô tả cho cột dữ liệu.

tt	**Loại thực thể con của "đối tượng"**	**Siêu dữ liệu mô tả**	
1	Cột dữ liệu	*Đối với tiêu đề cột* 1. Vị trí dòng kết xuất. 2. Vị trí cột kết xuất. 3. Font. 4. Kích thước. 5. Màu. 6. Tên tiêu đề cột.	*Đối với phần trị* 1. Vị trí dòng kết xuất. 2. Vị trí cột kết xuất. 3. Font. 4. Kích thước. 5. Màu. 6. Trị.

2.3.3. Siêu dữ liệu mô tả báo biểu

tt	**Loại thực thể con của "đối tượng"**	**Siêu dữ liệu mô tả**
2	Báo biểu	1. Tên. 2. Nhan đề chính. 3. Mục đích sử dụng. 4. Khổ rộng. 5. Hướng giấy. 6. Độ rộng biên trái. 7. Độ rộng biên phải. 8. Độ rộng biên trên. 9. Độ rộng biên dưới.

2.3.4. Siêu dữ liệu mô tả thành phần báo biểu kết xuất

Stt	**Loại thực thể con của "đối tượng"**	**Siêu dữ liệu mô tả**
3	Thành phần báo biểu kết xuất	1. Tên. 2. Phông chữ. 3. Cỡ chữ. 4. Kiểu chữ. 5. Loại chữ. 6. Kích thước. 7. Màu. 8. Trị thành phần báo biểu kết xuất. 9. Canh lề.

Chương 3. Các kỹ thuật xử lý XML với C#

3.1. XML

Trong thực tế bản thân ngôn ngữ XML có nguồn gốc giống như ngôn ngữ định dạng siêu văn bản HTML (HyperText Markup Language) từ chuẩn ngôn ngữ định dạng văn bản tổng quát có cấu trúc SGML. Mỗi văn bản XML cũng sử các thẻ (tags), các từ được đặt trong ngoặc với ‘’ (mở và đóng) và dùng thuộc tính tên gọi của các phần tử (element) với mẫu name= “value”.

Trong khi HTML đặc biệt chú ý tới từng thẻ (tag) và thuộc tính (attribute) có ý nghĩa gì và phần văn bản giữa các thẻ đó hiển thị như thế nào trên trình duyệt thì XML sử dụng các thẻ chỉ để phân định ranh giới giữa các đoạn dữ liệu và coi việc đọc và xử lý dữ liệu hoàn toàn là nhiệm vụ của các ứng dụng. Nhưng khác với ngôn ngữ HTML, số lượng và tên gọi các phần tử trong XML là không hạn chế.

XML là một văn bản nhưng không giống với những loại văn bản thông thường mà ta có thể đọc được. Các chương trình dùng để tạo các dữ liệu được cấu trúc hóa thông thường được lưu dữ liệu trên đĩa cứng, sử dụng khuôn dạng text hay nhị phân. Một thuận lợi của khuôn dạng văn bản là cho phép người đọc có thể đọc nó với bất kỳ bộ soạn thảo văn bản nào tùy thích. Các khuôn dạng văn bản cũng cho phép tìm lỗi dễ dàng hơn trong các ứng dụng. Giống như HTML các file XML là những file văn bản được tạo ra không phải với mục đích đề đọc, nhưng vẫn có thể đọc nếu thấy cần thiết. Tuy nhiên XML có điểm không bằng HTML, các luật dùng trong XML rất hạn chế, chỉ cần quên một thẻ, hay một thuộc tính không đi kèm với nội dung sẽ làm cho toàn bộ file XML đó ngừng hoạt động, trong khi đó lỗi này ở file HTML có thể được bỏ qua.

XML được xem như là ngôn ngữ mạnh hơn HTML do nó mang lại thông tin đầy đủ về dữ liệu. XML cung cấp “siêu dữ liệu” metadata hay còn được gọi là “dữ liệu về dữ liệu” (data about data). XML cho phép các nhà phát triển và quản trị công nghệ thông tin mô tả thông tin có liên quan tới các nguồn thông tin khác. Đây là phương pháp khai thác thông tin lý tưởng trong môi trường trao đổi thông tin từ các máy chủ ứng dụng cũng như từ các ứng dụng với nhau. Cấu trúc chặt chẽ của XML (nội dung được đặt giữa các thẻ metadata) cho phép các ứng dụng dễ dàng tìm kiếm và sử dụng nội dung đã tạo. Môi trường tài liệu XML trở thành một kho dữ liệu hỏi-đáp (query data repository) tương tự như cơ sở dữ liệu. Ngôn ngữ XML là giải pháp tích hợp cho vấn đề trao đổi dữ liệu tự động giữa các kho thông tin trên mạng Internet.

3.1.1. Tài liệu XML từ một cấu trúc cây:

- Tài liệu XML phải có một phần tử gốc, nguyên tố này là mẹ (the parent) của tất cả các yếu tố khác. Các yếu tố trong tài liệu XML tạo thành một cây tài liệu. Bắt đầu từ gốc(root), nhánh(branches) đến các mức thấp của cây.

- Tất cả các yếu tố có thể chứa các yếu tố phụ, yếu tố con.

```
<root>
<child>
<subchild>.....</subchild>
</child>
</root>
```

Các điều khoản parent, child và sibling được sử dụng để mô tả các mối quan hệ giữa các yếu tố. Child cùng cấp với sibling (brothers or sisters).

Tất cả các yếu tố có thể có nội dung văn bản và các thuộc tính giống như HTML.

3.1.2. Các yếu tố trong XML

- Một yếu tố XML là tất cả mọi thứ (bao gồm) các phần tử tử tag bắt đầu đến tag kết thúc.

- Một yếu tố có thể chứa các yếu tố khác, văn bản đơn giản hoặc hỗn hợp cả 2. Các yếu tố cũng có thể có các thuộc tính.

```
<bookstore>
  <book category="CHILDREN">
    <title>Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
  <book category="WEB">
    <title>Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>
```

Trong ví dụ trên, tag <bookstore> và <book> có các yếu tố nội dung. Bởi vì chúng chứa các nguyên tố khác. Tag <author> có nội dung văn bản bởi vì nó chứa văn bản.

Trong ví dụ trên, chỉ có tag <book> là có 1 thuộc tính (category = “CHILDREN”).

3.1.3. Các thuộc tính trong XML.

Nhìn chung các thuộc tính của các yếu tố XML giống như trong HTML. Các thuộc tính cung cấp thông tin bổ sung về phần tử. Phần này đề cập đến các vấn đề như các thuộc tính trong XML là gì, cách sử dụng tốt các thuộc tính, thuộc tính XML cho siêu dữ liệu .

•Các thuộc tính trong XML là gì?

Trong HTML, các thuộc tính cung cấp các thông tin bổ sung cho các phần tử.

<img src="computer.gif">

<a href="demo.asp">

Các thuộc tính thường cung cấp thông tin, đó không phải là một phần của dữ liệu. Trong ví dụ dưới đây, các loại tập tin là không liên quan đến các dữ liệu, nhưng có thể là quan trọng đối với phần mềm khi chúng ta muốn thao tác trên các phần tử.

<file type="gif">computer.gif</file>

•Các thuộc tính trong XML phải nằm trong dấu trích dẫn “”

Giá trị thuộc tính phải luôn nằm trong dấu trích dẫn. Có thể sử dụng dấu nháy đơn hay nháy kép trong từng trường hợp cụ thể được sử dụng. Đối với giới tính của người, phần tử người có thể được viết như thế này:

<person sex="female">

hoặc

<person sex='female'>

Nếu giá trị thuộc tính có sử dụng dấu nháy kép thì có thể viết như thế này:

<gangster name='George "Shotgun" Ziegler'>

Chúng ta cũng có thể sử dụng tham chiếu thực thể:

<gangster name="George "Shotgun" Ziegler">

•Các phần tử so với các thuộc tính của XML

Chúng ta hãy xem 2 ví dụ dưới đây:

```
<person sex="female">
  <firstname>Anna</firstname>
  <lastname>Smith</lastname>
</person>
<person>
  <sex>female</sex>
  <firstname>Anna</firstname>
  <lastname>Smith</lastname>
</person>
```

•Cách nên sử dụng:

Trong 3 tài liệu XML dưới đây chứa chính xác cùng một thông tin:

Một thuộc tính ngày được sử dụng trong ví dụ đầu tiên:

```
<note date="10/01/2008">
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

Một phần tử ngày được sử dụng trong ví dụ thứ 2:

```
<note>
  <date>10/01/2008</date>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

Khi sử dụng cách thứ 3, chúng ta có thể có những thông tin giống 2 cách đầu tiên. Nhưng khi khai thác thông tin từ phần mềm thì đây là cách hay nhất và tối ưu nhất.

•Tránh các thuộc tính XML:

Một số vấn đề xảy ra khi sử dụng các thuộc tính là:

- Thuộc tính không thể chứa nhiều giá trị (các phần tử có thể).
- Thuộc tính không thể chứa các cấu trúc cây (Các phần tử có thể).
- Thuộc tính không dễ dàng mở rộng (đối với những thay đổi trong tương lai).
- CÁc thuộc tính khó đọc và duy trì. Sử dụng các phần tử cho dữ liệu. Sử dụng các thuộc tính cho các thông tin liên quan đến dữ liệu.

Đừng kết thúc như thế này:

```
<note day="10" month="01" year="2008"
to="Tove" from="Jani" heading="Reminder"
body="Don't forget me this weekend!">
</note>
```

- Các thuộc tính XML cho siêu dữ liệu

Đôi khi các ID tham chiếu được gán cho các phần tử. Những ID có thể được sử dụng để xác định các phần tử(yếu tố) của XML theo cách tương tự như các thuộc tính ID trong HTML. Ví dụ dưới đây sẽ chứng minh điều này:

```
<messages>
  <note id="501">
    <to>Tove</to>
    <from>Jani</from>
    <heading>Reminder</heading>
```

```
  <body>Don't forget me this weekend!</body>
 </note>
 <note id="502">
  <to>Jani</to>
  <from>Tove</from>
  <heading>Re: Reminder</heading>
  <body>I will not</body>
 </note>
</messages>
```

Các thuộc tính id ở trên là để xác định các ghi chú khác nhau, nó không phải là một phần của các ghi chú (note).

Những gì mà tôi đang cố gắng nói ở đây là siêu dữ liệu (dữ liệu về dữ liệu) phải được lưu trữ như là các thuộc tính, và các dữ liệu chính nó phải được lưu trữ như là các phần tử(yếu tố).

3.2. Các kỹ thuật xử lý XML với C#.

3.2.1. Giới thiệu về namespace System.xml

- namespace System.xml trong .NET cung cấp một số lớp hỗ trợ cho việc xử lý XML. Dưới đây là những lớp đọc và ghi XML.

Tên lớp Giải thích

XmlReader Một lớp đọc trừu tượng nhanh và non-cached dữ liệu XML. XmlReader được thiết kế giống như bộ phân tách SAX.

XmlWriter Một lớp viết trừu tượng nhanh và non-cached dữ liệu XML trong một dòng hoặc định dạng file.

XmlTexReader Mở rộng của XmlReader. Cung cấp chuỗi truy cập nhanh dữ liệu XML.

XmlTexWriter Mở rộng của XmlWriter. Phát nhanh các dòng XML.

- Một vài lớp hữu ích khác trong XML:

Tên lớp Giải thích

XmlNode Một lớp trừu tượng miêu tả một nút đơn trong một tài liệu XML. Lớp cơ sở cho các lớp khác trong namespace XML.

XmlDocument Mở rộng của XmlNode. Đây là một thực thi W3C Document Object Model (DOM). Nó cung cấp một cây miêu tả tài liệu XML trong bộ nhớ cho phép điều hướng và soạn thảo.

XmlDataDocument Mở rộng của XmlDocument. Đây là một tài liệu có thể được tải từ dữ liệu XML hoặc từ dữ liệu trong mộtADO.NET DataSet. Cho phép hòa trộn XML và dữ liệu quan hệ trong cùng một view.

XmlResolver Một lớp trừu tượng dùng giải quyết các tài nguyên XML ngoài như DTD và tham chiếu sơ đồ. Cũng dùng để xử lí các thành phần <xsl:include> và <xsl:import>.

XmlUrlResolver Mở rộng của XmlResolver. Giải quyết các tài nguyên tên như một URI (Uniform Resource Identifier).

- Lưu ý: namespace xml có sẵn cho bất kỳ ngôn ngữ nào biết .NET

3.2.2. Đọc và Ghi XML.

- Cả 2 lớp XmlReader và XmlWriter đều là những lớp trừu tượng. Hình dưới đây minh họa các lớp kế thừa từ 2 lớp này:

+ XmlTextReader và XmlTextWriter làm việc chung trên các đối tượng stream hoặc các đối tượng TextReader/TextWriter trong namespace System.IO.

+ XmlNodeReader sử dụng XmlNode cho một nguồn thay cho một stream. XmlValidatingReader thêm DTD với sơ đồ tích hợp và tất nhiên là cả dữ liệu

a) Sử dụng lớp XmlTexReader

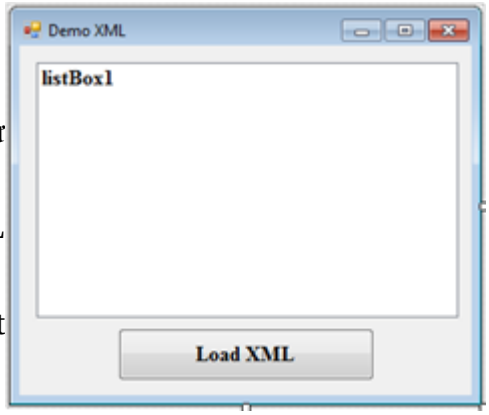
- XmlTexReader rất giống SAX. Một trong những khác biệt lớn nhất: SAX là một mô hình kiểu push, còn XmlTextReader là một mô hình pull, ở đó dữ liệu được kéo vào ứng dụng yêu cầu nó. Nó tạo ra một mô hình lập trình dễ dàng và trực quan hơn. Một lợi ích khác của mô hình pull là có thể lựa chọn dữ liệu để gởi đến ứng dụng: nếu bạn không muốn tất cả các dữ liệu, vì không cần xử lý tất cả chúng. Còn trong mô hình push, tất cả dữ liệu XML cần phải được xử lý bởi ứng dụng, mặc cho nó có muốn hay không.

- Để sử dụng lớp này cần khai báo :

using System.Xml;

- Bây giờ hãy mở visual studio lên:

+ Kéo vào một ListBox và một Button như hình bên;



+ Sau đó viết sự kiện cho button Load XML như sau:

```
private void btnLoadXML_Click(object sender, EventArgs e)
        {
            // Đường dẫn tới file xml.
            string fileName = "Book.xml";
            // Tạo một đối tượng TextReader mới
            XmlTextReader xtr = new XmlTextReader(fileName);
            while (xtr.Read())
            {
               if (xtr.NodeType == XmlNodeType.Text)
               {
                  listBox1.Items.Add(xtr.Value);
               }
            }
```

```
}
```

- Kết quả khi chạy chương trình trên và click vào button LoadXML như hình bên.

- XmlTextReader này sử dụng khá đơn giản.

+ Trước tiên chúng ta tạo ra một đối tượng string chứa dường dẫn tới file xml. Sau đó tạo một đối tượng XmlTextReader mới với tham số là đường dẫn tới file xml.

+ Khi chương trình chạy đến vòng lặp while, phương thức Read sẽ được di chuyển sang mục tiêu đầu tiên trong tài liệu. Nó tiêu biểu cho các mục khai báo XML. Trong ví dụ này, chúng ta duyệt qua từng mục và so sánh xtr.NodeType với bộ XmlNodeType, và thêm các mục được tìm thấy vào listbox.

3.2.3. Các phương thức Read.

- Có một vài cách di chuyển trong tài liệu. Như bạn đã thấy trong ví dụ trên, Read() có thể di chuyển sang mục tiếp theo. Chúng ta có thể xem nêu mục đó có giá trị (HasValue()) , hoặc nếu mục đó có thuộc tính (HasAttributes()) . Chúng ta cũng có thể dùng phương thức ReadStartElement(), để kiểm tra xem nếu mục hiện tại là thành phần khởi đầu, và chuyển sang mục tiếp theo. Nếu không phải là mục khởi đầu một ngoại lệ XmlException sẽ được phát ra. Việc gọi phương thức này giống như gọi phương thức IsStartElement(), bởi một Read().

- Các phương thức ReadString() và ReadChars() đều đọc dữ liệu văn bản từ một thành tố. ReadString() tra về một chuỗi dữ liệu, trong khi ReadChars() trả về một mảng dữ liệu kiểu char.

- ReadElementString() cũng giống như ReadString(), ngoại trừ việc bạn không phải truyền tên của một thành tố. Nếu nội dung của mục tiếp theo không phải là một start tag, hoặc nếu tham số Name không không phải là Name của mục hiện hành, thì một ngoại lệ sẽ được phát ra.

- Dưới đây là ví dụ chỉ ra cách sử dụng ReadElementString(), lưu ý khai báo:

```
using System.IO;
private void btnLoadXML_Click(object sender, EventArgs e)
{
string fileName = “Book.xml”;
FileStream fs = new FileStream(fileName, FileMode.Open);
XmlTextReader xtr = new XmlTextReader(fs);
while (!xtr.EOF)
{
if (xtr.MoveToContent() == XmlNodeType.Element && xtr.Name == “title”)
        listBox1.Items.Add(xtr.ReadElementString());
else
        xtr.Read();
```

```
        }
        }
```

- Trong vòng lặp while chúng ta sử dụng MoveToContent() để tìm trên mỗi dòng xem XmlNodeType.Element có giống với named title không. Chúng ta sử dụng thuộc tính EOF của XmlTextReader như là một điều kiện lặp. Nếu mục không phải kiểu Element của named title, mệnh đề else phát ra một phương thức Read() để di chuyển sang mục tiếp theo. Khi chúng ta tìm thấy một mục thỏa điều kiện, chúng ta trả kết quả của ReadElementString() cho listbox. Nó cho phép các tựa sách được liệt kê trong listbox. Chú ý rằng chúng ta không tạo ra một lời gọi Read() sau khi một ReadElementString() thực hiện thành công. Bởi vì ReadElementString() cũng sẽ di chuyển sang mục tiếp theo.

- Nếu bạn bỏ && tr.Name==”title” trong mệnh đề if, bạn sẽ nhận được ngoại lệ XmlException. Nếu nhìn vào file dữ liệu, bạn sẽ thấy thành tố đầu tiên mà MoveToContent() tìm ra là <bookstore>. Tất nhiên nó vì nó không chứa một kiểu text chuẩn, nên ReadElementString() phát ra một ngoại lệ XmlException.

3.2.4. Lấy thuộc tính của dữ liệu:

- Khi bạn chạy các ví dụ trên, bạn nhận ra rằng khi các mục được đọc, bạn không thấy bất kì thuộc tính nào cả. Đó là vì các thuộc tính không nằm trong tài liệu. Khi đang đứng trên mộtmục, bạn có thể kiểm tra các thuộc tính và có thể lấy giá trị của bất kì giá trị thuộc tính nào.

- Thuộc tính HasAttributes sẽ trả về giá trị true nếu có bất kì thuộc tính nào còn không sẽ trả về false. Thuộc tính AttributeCount sẽ cho bạn biết có bao nhiêu thuộc tính, và phương thức GetAttribute() sẽ trả về một thuộc tính thông qua tên hoặc chỉ mục. Nếu bạn muốn lặp qua các thuộc tính bạn có thể dùng các phương thức MoveToFirstAttribute() và MoveToNextAttribute().

- Dưới đây là một ví dụ về việc lặp qua các thuộc tính.

```
        string fileName = “Book.xml”;
        FileStream fs = new FileStream(fileName, FileMode.Open);
        XmlTextReader xtr = new XmlTextReader(fs);
        while (xtr.Read())
                {
                if          (xtr.NodeType          ==          XmlNodeType.Element)
                {
                        for     (int     i     =     0;     i     <     xtr.AttributeCount;     i++)
                        {
        listBox1.Items.Add(xtr.GetAttribute(i));
}
        }
}
```

- Bây giờ chúng ta xem xét về các mục thành phần. Khi chúng ta tìm thấy một mục, chúng ta lặp qua tất cả thuộc tính của nó, và dùng phương thức GetAttribute() để load giá trị của thuộc tính vào listbox. Trong ví dụ này các thuộc tính đó là genre, publicationdate, và ISBN.

d) Sử dụng lớp XmlValidatingReader.

- Nếu muốn xác nhận một tài liệu XML, bạn sẽ cần phải sử dụng lớp XmlValidatingReader. Nó chứac các khả năng giống như XmlTextReader (Cả hai đều xuất phát từ XmlReader) nhưng XmlValidatingReader có thêm thuộc tính ValidationType, thuộc tính Schemas và SchemaType.

- Nếu gán thuộc tính ValidationType giá trị xác nhận mà bạn muốn. Giá trị hợp lệ của thuộc tính này được liệt kê trong bảng sau:

Property value	Description
Auto	Nếu một DTD được khai báo trong một khai báo <!DOCTYPE…>, điều này cho phép DTD sẽ được load và xử lí. Giá trị mặc định cho các DTD.Nếu một thuộc tính XSD schemalocation được tìm thấy, XSD được load và xử lí, và sẽ trả về các giá trị mặc định trong sơ đồ. Nếu một không gian tên với tiếp đầu ngữ MSXML x-schema được tìm thấy, nó sẽ load và xử lí sơ đồ XDR và trả về các thuộc tính mặc định đã được định nghĩa.
DTD	Phù hợp theo chuẩn DTD.
Schema	Phù hợp theo sơ đồ XSD.
XDR	Phù hợp theo sơ đồ XDR
None	Không giá trị hợp lệ nào được thực thi.

3.2.5. Sử dụng Schema property.

Schemas property của XmlValidatingReader chứa một XmlSchemaCollection, có thể tìm thấy trong không gian tên System.Xml.Schema. Tập hợp này tổ chức load lại loaded XSD và XDR schemas. Nó cực nhanh đặc biệc là khi bạn cần kiểm tra sự hợp lệ của nhiều tài liệu khác nhau, vì sơ đồ sẽ không được load mỗi khi kiểm tra. Các bước sử dụng thuộc tính này như sau, bạn tạo một đối tượng XmlSchemaCollection. Phương thức Add(), nằm trong một XmlSchemaCollection, có bốn quá tải. Bạn có thể truyền nó cho một đối tượng xuất phát từ XmlSchema, một đối tượng xuất phát từ XmlSchemaCollection, một chuỗi không gian tên với chuỗi URI của file sơ đồ và một đối tượng xuất phát từ XmlReader chứa trong sơ đồ.

3.2.6. Sử dụng lớp XmlTextWriter.

- Lớp XmlTextWriter cho phép bạn xuất XML thành một chuỗi, một file hoặc một đối tượng a TextWriter. Giống như XmlTextReader, nó là một kiểu forward-only, non-cached. XmlTextWriter có thể cấu hình cao, cho phép bạn chỉ rõ những thứ như cho phép thục đầu dòng, số thục đầu dòng, kí tự chỉ dẫn nào được dùng trong các giá trị thuộc tính cho phép namespace hỗ trợ.

Một ví dụ về cách sử dụng lớp XmlTextWriter:

```
private void btnGhiXML_Click(object sender, EventArgs e)
{
   SaveFileDialog sfd = new SaveFileDialog();
   sfd.Filter = "XML file(*.xml)|*.xml";
   sfd.RestoreDirectory = true;
   if (sfd.ShowDialog() == DialogResult.OK)
   {
      XmlTextWriter xtw = new XmlTextWriter(sfd.FileName, null);
      xtw.Formatting = Formatting.Indented;
      xtw.WriteStartDocument();
            // write to element HoTen
            xtw.WriteStartElement("LyLich");
            xtw.WriteAttributeString("QuocTich", "Viet Nam");
            xtw.WriteElementString("HoTen", txtHoTen.Text);
            xtw.WriteElementString("QueQuan", txtQueQuan.Text);
            xtw.WriteElementString("NgaySinh", txtNgaySinh.Text);
```

```
            xtw.WriteEndElement();
            xtw.WriteEndDocument();
            xtw.Flush();
            xtw.Close();
        }
    }
```

Chương 4. Chương trình tự động lập báo biểu tự động kết xuất với cơ sở dữ liệu

4.1. Mô tả chức năng của hệ thống

Danh sách các usecase

** Usecase Hiển thị danh sách sinh viên:*

- Hiển thị toàn bộ thông tin dữ liệu của sinh viên dưới dạng bảng.

- Cho phép chọn sinh viên trong danh sách sinh viên.

** Usecase Hiển thị thông tin bảng biểu*

- Hiển thị toàn bộ thông tin dữ liệu bảng biểu được chọn.

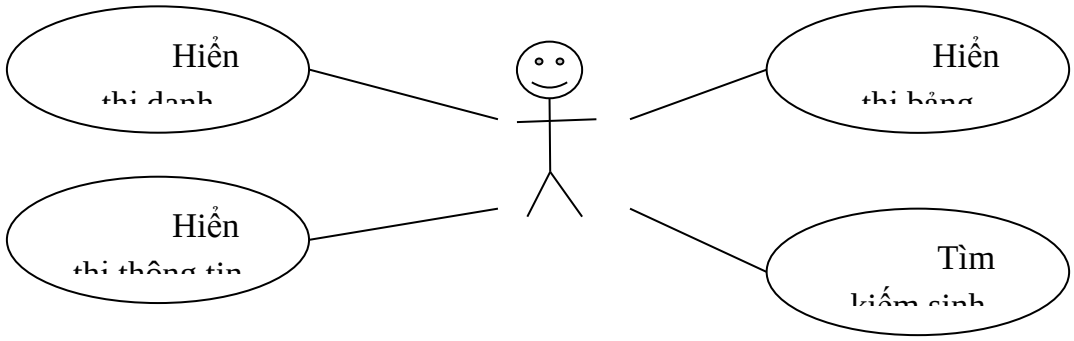
- Cho phép thao tác cập nhật, chỉnh sửa thông tin bảng biểu cho phù hợp mục đích sử dụng.

** Usecase Hiển thị bảng biểu*

- Hiển thị toàn bộ thông tin về sinh viên theo danh sách được chọn và bảng biểu được trình bày như đã chỉnh sửa.

** Usecase Tìm kiếm sinh viên*

- Cho phép người dùng tìm kiếm sinh viên theo mã sinh viên.



4.2. Mô tả các usecase

Kịch bản Hiển thị danh sách sinh viên

Usecase	**Hiển thị danh sách sinh viên**
Tác nhân	User
Điều kiện phát sinh	- Người dùng nhấp vào tên lớp trong cây danh sách lớp

Usecase	**Hiển thị danh sách sinh viên**
kịch bản	- CSDL lưu thông tin sinh viên tồn tại
Kịch bản chính	1. Người dùng nhấp vào tên lớp trong cây danh sách lớp 2. Hệ thống hiển thị dữ liệu.
Kịch bản phụ	
Kịch bản lỗi	
Điều kiện hoàn tất kịch bản	Danh sách sinh viên được hiển thị lên form.

Kịch bản Hiển thị thông tin bảng biểu

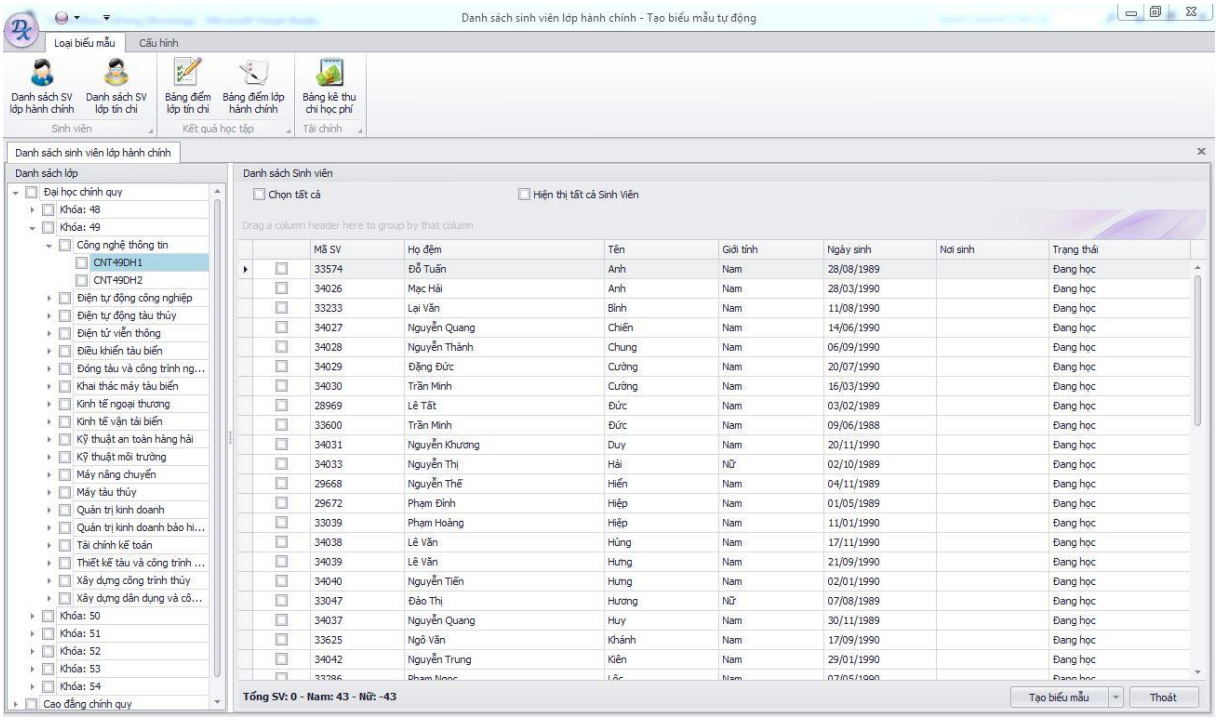
Usecase	**Hiển thị thông tin bảng biểu**
Tác nhân	User
Điều kiện phát sinh kịch bản	- File XML thông tin bảng biểu tồn tại
Kịch bản chính	1. Người dùng chọn nút “tạo biểu mẫu” 2. Hệ thống hiển thị thông tin cấu hình bảng biểu
Kịch bản phụ	
Kịch bản lỗi	
Điều kiện hoàn tất kịch bản	Thông tin cấu hình bảng biểu được hiện lên form

Kịch bản Hiển thị bảng biểu

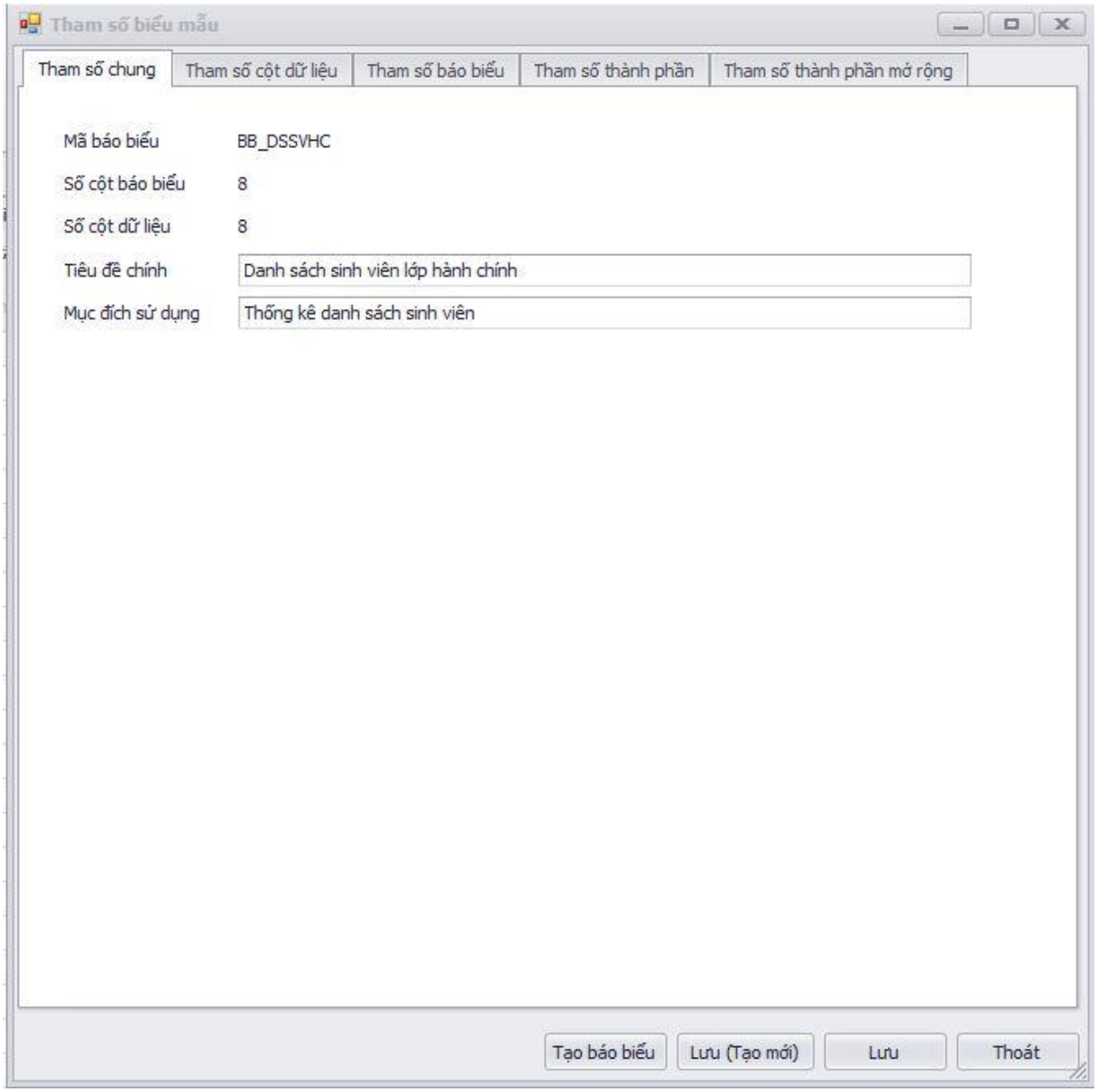
Usecase	**Hiển thị bảng biểu**

Usecase	**Hiển thị bảng biểu**
Tác nhân	User
Điều kiện phát sinh kịch bản	- File XML bảng biểu tồn tại
Kịch bản chính	1. Người dùng "Tạo báo biểu" 2. Hệ thống hiển thị báo biểu
Kịch bản phụ	
Kịch bản lỗi	
Điều kiện hoàn tất kịch bản	Hiển thị báo biểu với thông tin sinh viên được chọn lên form

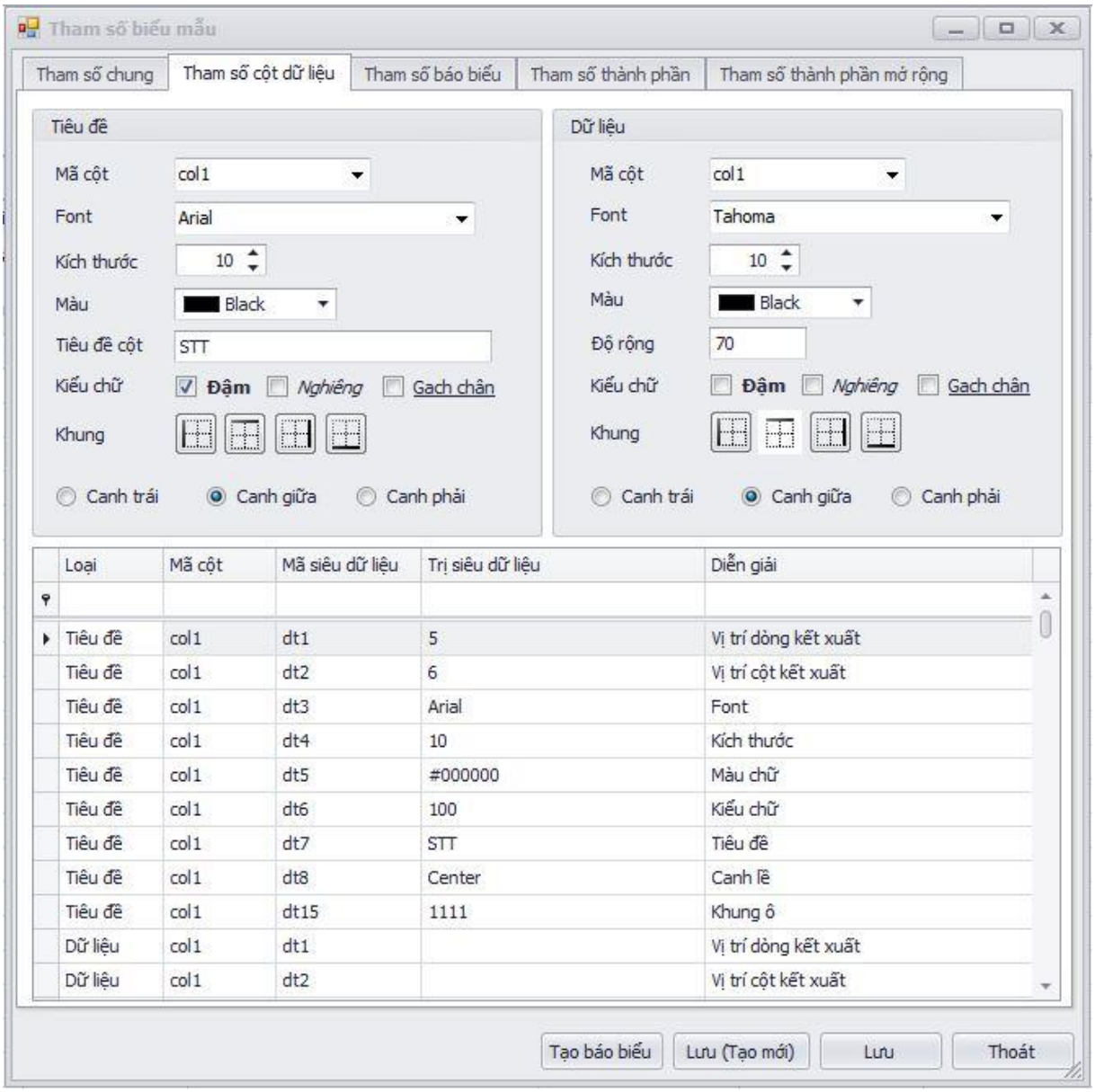
4.3. Giao diện chương trình



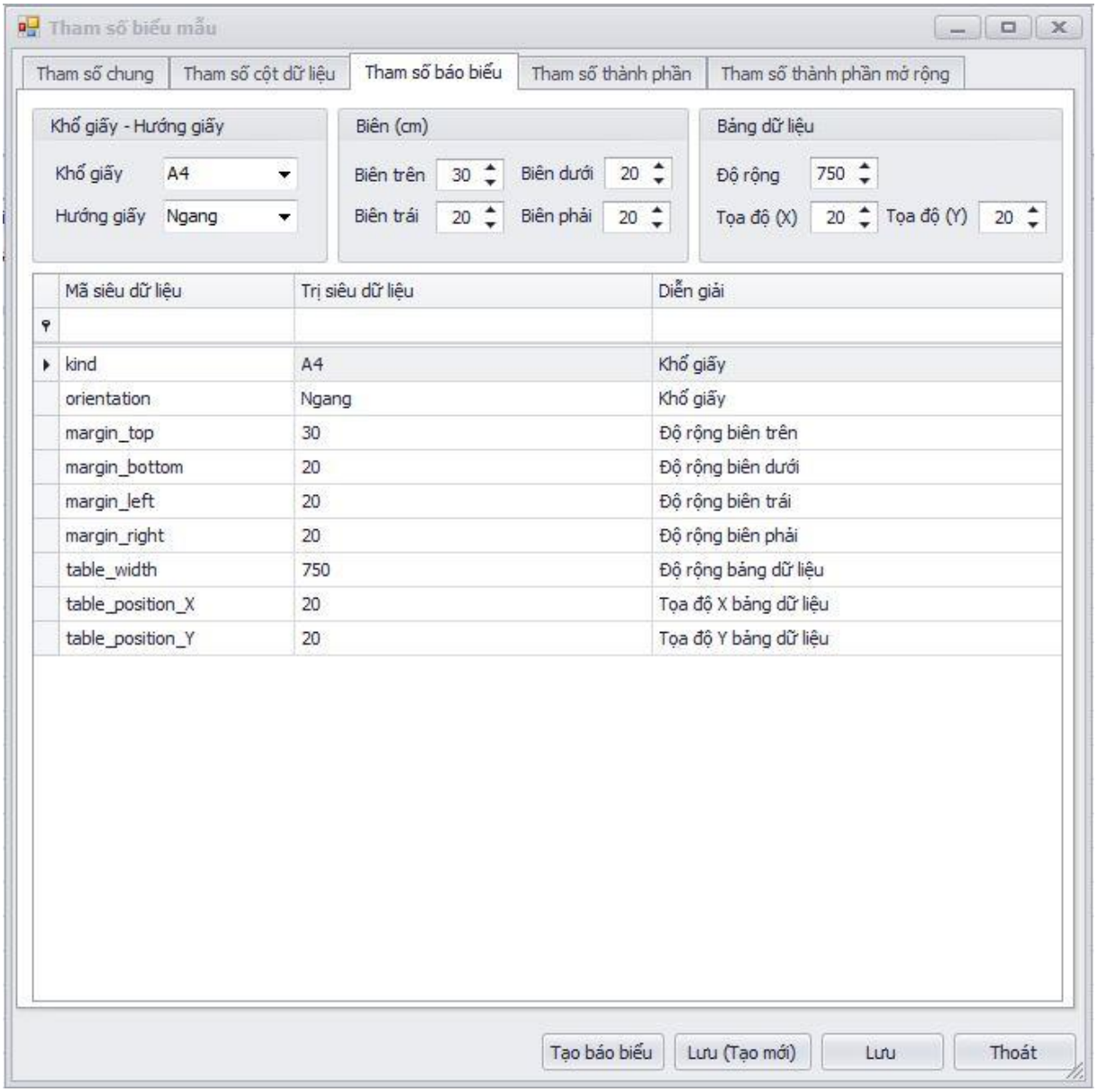
Hình 4.1. From lấy dữ liệu từ CSDL



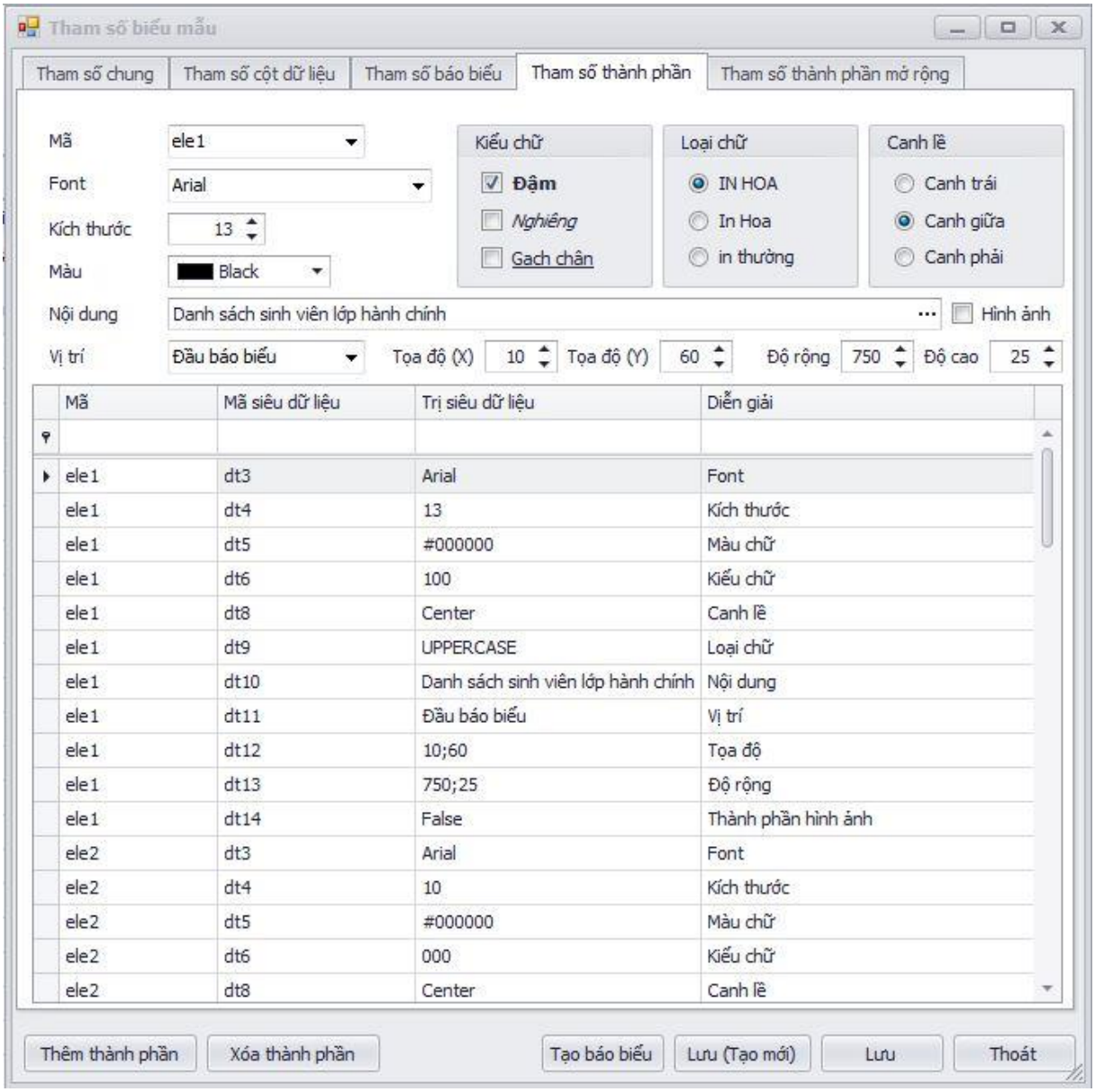
Hình 4.2. Tab Tham số biểu mẫu



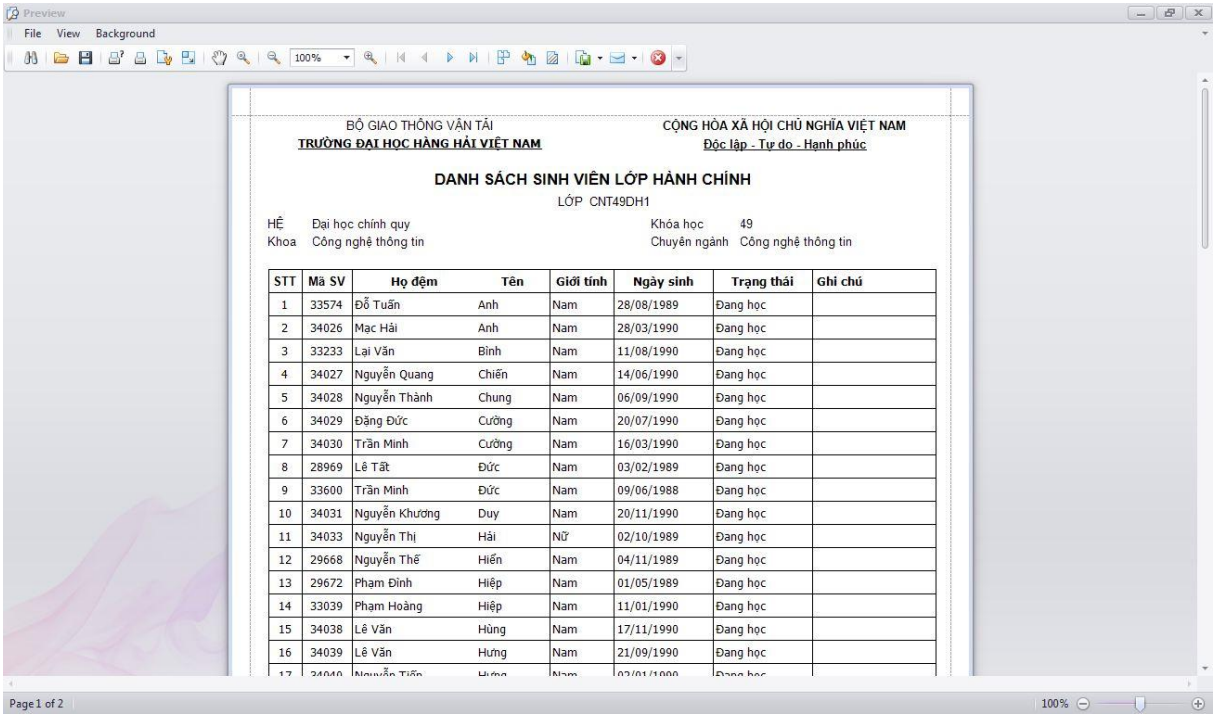
Hình 4.3. Tham số cột dữ liệu



Hình 4.4. Tham số báo biểu



Hình 4.5. Tham số thành phần



STT	Mã SV	Họ đệm	Tên	Giới tính	Ngày sinh	Trạng thái	Ghi chú
1	33574	Đỗ Tuấn	Anh	Nam	28/08/1989	Đang học	
2	34026	Mạc Hải	Anh	Nam	28/03/1990	Đang học	
3	33233	Lại Văn	Bình	Nam	11/08/1990	Đang học	
4	34027	Nguyễn Quang	Chiến	Nam	14/06/1990	Đang học	
5	34028	Nguyễn Thành	Chung	Nam	06/09/1990	Đang học	
6	34029	Đặng Đức	Cường	Nam	20/07/1990	Đang học	
7	34030	Trần Minh	Cường	Nam	16/03/1990	Đang học	
8	28969	Lê Tất	Đức	Nam	03/02/1989	Đang học	
9	33600	Trần Minh	Đức	Nam	09/06/1988	Đang học	
10	34031	Nguyễn Khương	Duy	Nam	20/11/1990	Đang học	
11	34033	Nguyễn Thị	Hải	Nữ	02/10/1989	Đang học	
12	29668	Nguyễn Thế	Hiển	Nam	04/11/1989	Đang học	
13	29672	Phạm Đình	Hiệp	Nam	01/05/1989	Đang học	
14	33039	Phạm Hoàng	Hiệp	Nam	11/01/1990	Đang học	
15	34038	Lê Văn	Hùng	Nam	17/11/1990	Đang học	
16	34039	Lê Văn	Hưng	Nam	21/09/1990	Đang học	

1. Xuất báo cáo

KẾT LUẬN

Đề tài nhóm tác giả xây dựng là đề tài thực tế và có tính ứng dụng trong công tác đào tạo của trường học với nhiều sinh viên, giảm tải việc người dùng phải nhớ, tự lập các mẫu báo biểu hoặc mất thời gian tra cứu trên mạng mở rộng hơn ta có thể hướng tới các mẫu văn bản hành chính của các cơ quan hành chính sự nghiệp và các doanh nghiệp. Do điều kiện về thời gian không dài, nên đề tài vẫn còn nhiều thiếu sót, khi triển khai vào thực tế nhóm tác giả sẽ hoàn thiện hơn.