Three Dimensional Database: Creating Dynamic Web Controls Using XML and XSLT

R. Vadivel[1],Dr. K. Baskaran[2]

GJCST Classification (FOR)
H.2.1, H.3.5

***Abstract*-a dynamic web application is vital for online business. It has increased the on-demand needs of client requirements. When creating a data-driven Web site, one of the most common tasks Web developers are faced with is creating data entry forms. Data entry forms are Web pages that provide the system's users with a means to input data. The task of creating a particular data entry form typically starts with hammering out the requirements that spell out specifically what information needs to be collected from the user. With the requirements defined, the next stage is designing the data entry Web Form, which involves creating the graphical user interface, as well as writing the code that updates the database with the user's inputs. The main objective of this paper is to construct controls dynamically, that is creating web controls in run time and not in design-time. We can create large amount of dynamic fields with dynamic validations with the help of XML, XSL & Java script. A database plays a major role to accomplish this functionality. We can use 3D (static, dynamic and Meta) database structures. One of the advantages of the XML/XSLT combination is the ability to separate content from presentation. A data source can return an XML document, then by using an XSLT, the data can be transformed into whatever HTML is needed, based on the data in the XML document. The flexibility of XML/XLST can be combined with the power of ASP.NET server/client controls by using an XSLT to generate the server/client controls dynamically, thus leveraging the best of both worlds. This synergy is demonstrated by creating a publication domain application.**

Keywords- three dimensional database, extensible Mark-up Language, web application, dynamic controls, extensible stylesheet language

I. Introduction

When creating a data-driven Web site, one of the most common tasks Web developers are faced with is creating data entry forms. Data entry forms are Web pages that provide the system's users with a means to input data. The task of creating a particular data entry form typically starts with hammering out the requirements that spell out specifically what information needs to be collected from the user. With the requirements defined, the next stage is designing the data entry Web Form, which involves creating the graphical user interface, as well as writing the code that updates the database with the user's inputs.When the data entry forms requirements are well-known in advance, and when such data entry forms are identical across all users for the system, creating such entry forms is hardly challenging.

About[1]-Computer Science, Karpagam University Pollachi Road, Eachanari, Coimbatore, Tamilnadu India 641 024vadivel.rangasamy@gmail.com
About[2]-*Computer Science, Karpagam UniversityPollach Road, Eachanari, Coimbatore, Tamilnadu India 641 024*vadivel.rangasamy@gmail.com

The task becomes more arduous, however, if the data entryforms need to be dynamic. For example, consider a company's Internet Web application whose purpose is tocollect information about the product purchased by a customer; a sort of online product registration system. With such an application, the questions the user is presented with might differ based on what product they purchased, or if they purchased the product from a store or from the company's Web site.When faced with needing to provide dynamic data entry user interfaces, as in the example mentioned above, one option might be to "brute force" a solution. You could create a separate Web page for each product your company sells, with each page having the specific data entry elements needed. The problem with this naive approach is that it requires adding new pages when new products are released. While creating these new pages might not be terribly difficult, it is time consuming and prone to errors without sufficient debugging and testing time.Ideally, when new products are released, a non-technical co-worker could specify what questions are required through an easy-to-use Web-based interface. Such a system is quite possible with ASP.NET thanks to the ability to dynamically load controls on an ASP.NET Web page at runtime. With just a bit of an initial investment in development and testing time, you can create a reusable, dynamic data entry user interface engine. One that allows even the least computer savvy users the ability to easily create customized data entry forms. In this article, we will look at the fundamentals of working with dynamic controls in ASP.NET, and then I will present a complete, working dynamic data entry system that can be easily customized and extended.

II. Existing System

In existing database structure is flat or two dimensional definitions is simple database design consisting of one large table instead of several interconnected tables of a relational database. Called 'flat' because of its only two dimensional (data fields and records) structure, these databases cannot represent complex data relationships. Also called flat file database or flatform database.

Having a flat table to store all the data poses the following issues:

- table grows very large
- indexing the table is problematic
- not optimized for either update or read
- no stringent type checking as everything is stored in the database as a string (varchar/nvarchar)
- catering for text and number is problematic

- downloading the data requires several joins to group and instance tables which has an impact on performance and adds complexity to the query

A static field structure is a created for an input data set which is not supposed to change within the scope of the problem. When a single field is to be added or deleted, the update of a static field structure incurs significant costs, often comparable with the construction of the field structure from scratch. Create a static field it should know the all required fields and also to take much development time and need to given static names for those fields on design time and also there is no possible to apply unique style. A main disadvantage of static field is not specifying the data types (such as integer, string and etc.,) on run time and flat database structures are used. "N" number of lines to taken creating static fields and occupy the memory on design time and it may be possible to leakage of memory. In hacker can be easily hack those fields on run time and they can create the pseudo code to collapse those fields. Testing results for web controls on one web page, size of web page is 6.0 KB, available static fields is 7 and apply for 5 iterations so that get 35 fields and given below controls loading time for each iterations,

Iterations	Page size (Run Time)	Loading Time
1.	150.2K	2.262s
2.	150.2K	1.258s
3.	150.2K	1.766s
4.	150.2K	1.794s
5.	150.2K	1.484s
Total	751 K	8.564s

III. Related Works

The three dimensional database has been played major rule on the creation dynamic fields. Database architecture for creating dynamic web controls is a three dimensional structure, where we use three terms static, meta and dynamic. Here Static data is generally creating the tables and fields to the database. Meta data is a bridge between static and dynamic data. Dynamic data is the dynamic resultant tables or views that the user needs. An output of database is XML format and it contains data definition and data values. XSL is a presentation part which transforms XML data to output HTML.In three dimensional databases has used two types of SQL statements Static and Dynamic. Static SQL is SQL statements in an application that do not change at runtime and, therefore, can be hard-coded into the application. Dynamic SQL is SQL statements that are constructed at runtime; for example, the application may allow users to enter their own queries. Thus, the SQL statements cannot be hard-coded into the application.XSLT is designed for use as part of XSL, which is a style sheet language for XML. In addition to XSLT, XSL includes an XML vocabulary for specifying formatting. XSL specifies the styling of an XML document by using XSLT to describe how the document is transformed into another XML document that uses the formatting vocabulary.XSLT is also designed to be used independently of XSL. However, XSLT is not intended as a completely general-purpose XML transformation language. Rather it is designed primarily for the kinds of transformations that are needed when XSLT is used as part of XSL.

1) *Data Model*

The data model used by XSLT is the same as that used by XPath with the additions described in this section. XSLT operates on source, result and stylesheet documents using the same data model. Any two XML documents that have the same tree will be treated the same by XSLT. Processing instructions and comments in the stylesheet are ignored: the stylesheet is treated as if neither processing instruction nodes nor comment nodes were included in the tree that represents the stylesheet.

2) *Root Node Children*

The normal restrictions on the children of the root node are relaxed for the result tree. The result tree may have any sequence of nodes as children that would be possible for an element node. In particular, it may have text node children, and any number of element node children. When written out using the XML output method (see [16 Output]), it is possible that a result tree will not be a well-formed XML document; however, it will always be a well-formed external general parsed entity.When the source tree is created by parsing a well-formed XML document, the root node of the source tree will automatically satisfy the normal restrictions of having no text node children and exactly one element child. When the source tree is created in some other way, for example by using the DOM, the usual restrictions are relaxed for the source tree as for the result tree.

3) *Base URI*

Every node also has an associated URI called its base URI, which is used for resolving attribute values that represent relative URIs into absolute URIs. If an element or processing instruction occurs in an external entity, the base URI of that element or processing instruction is the URI of the external entity; otherwise, the base URI is the base URI of the document. The base URI of the document node is the URI of the document entity. The base URI for a text node, a comment node, an attribute node or a namespace node is the base URI of the parent of the node.

IV. Experimental Results

Solis architecture provides three main areas of functionality self-updating interface on the web, robust database administration, searchable front-end for end users. That system is designed so that dynamic data at the core of the integrated system is available in any output or view. The data administrator has control over the data content, various templates and user permissions, thereby giving an unrivalled level of flexibility and control in content collection, management and presentation.

1) *Data Administration*

Acomplete database administration application that provides the full range of control and flexibility needed for complete editorial control over any type of database. Features of the Data Administration component include:

- User permission management
- Saved-search templates to speed data interrogation
- Management control over:
- field types and structure
- data groups
- data viewing tabs
- data record structure
- taxonomy and categorization
- flat downloads
- online subscribers
- Output listing levels.
- Approvals system for self-updated submissions
- Sub-group counting by specification
- Data entity linking

2) *Self-Updating*

This component is an advanced user permission/restriction that enables the data administrator to give access per data record to both editors and directly to users. The access is via an editorial interface that enables the user to access and update information pertaining to a specific data record. Changes and information are submitted into the Approvals are of the Data Administration.

- Fully customizable
- E-commerce capability
- Dynamic paths
- Graphics and rich media upload
- Data record linking (single owner, multiple records)

3) *Customer-Facing Front-end*

This component enables the system owner to create custom print or online views of the database that can be integrated into existing websites.

- Web search versions (using template results sets)
- E-commerce control
- Search reconfiguration
- Advertising support
- Permission-based data download
- Automate data output to print (using Adobe InDesign and/or Quark)

4) *Methodology*

Database: The proposed database architecture for creating dynamic web controls is a three dimensional structure, where we use three terms static, meta and dynamic. Here Static data is generally creating the tables and fields to the database. Meta data is a bridge between static and dynamic data. Dynamic data is the dynamic resultant tables or views that the user needs.XML / XSL: The proposed XML comprises the data definition and data values. Data definition contains a label names and data values contains label values. XSL is a presentation part which transforms XML data to output HTML. Here the screen have show the output of the XML format

```
<Ddid="100"name="Kirschner200708"dt="0">

<Ggid="501"name="Adjusters Basic Information"desc="Adjuster Listing Information"n="1"s="1"vfp="1"vm="1"vd="1"gt="3"o="2"tf="0"dt="This information is published online and in CD version of the directory.">

<Ffid="5039"name="UpdatedDate"l="Updated Date"ft="2"o="0"vtl="0" />

<Ffid="5040"name="Adv"l="Advertiser in Print?"ft="3"o="1"vtl="0" />

<Ffid="5041"name="Name1"l="First Name of Company"ft="2"o="2"vtl="0" />

<Ffid="5042"name="Name2"l="Last Name of Company"ft="2"o="3"vtl="0" />

<Ffid="5043"name="COMPANY"l="Company Name"ft="2"o="4"vtl="1" />

<Ffid="5044"name="Addr_P"l="PO Box"ft="2"o="5"vtl="1" />
```

Fig – 1 XML format for data definition

```
<Uuid="2944"MKT-ID="1000"ACC-NO="adj3559"DIR-ID="100">

<Guid="2944"gid="501"pid="6033"del="0">

<Ffid="5039"approval="0"data="2000/06/01" />

<Ffid="5041"approval="0"data="" />

<Ffid="5042"approval="0"data="Fleetwood Claim Serv" />

<Ffid="5043"approval="0"data="Fleetwood Claim Serv" />

<Ffid="5044"approval="0"data="2855 Mangum Rd" />

<Ffid="5045"approval="0"data="Houston" />

</G>

<Guid="2944"gid="508"pid="663278"del="0">

<Ffid="5316"approval="0"data="true" />

</G>

.....

.....

.....

</U>
```

Fig – 2 XML format for data value

Building blocks of XML documents are nested, tagged elements. Each tagged element has zero or more sub elements; zero or more attribute, and may contain textual information (data content). Elements can be nested at any depth in the document structure. Attributes can be of different types, allowing one to specify element identifiers (attributes of type ID), additional information about the element (e.g., attribute of type CDATA containing textual information), or link to other elements of the document (attributes of type IDREF(s)). An example of XML document is presented in Figure 1 and 2. The document represents the data definition and data values of the publication fields. The XML document contains also all information on the custom fields.To develop on a formal basis our approach for secure publishing of XML documents we introduce a formal model of XML documents that we use throughout the paper. In the following, we denote with Label be a set of element tags and attribute names, and Value a set of attribute/element values. An XML document can be formally defined as follows.

```
<xsl:templatename="DisplayFieldValue">

<xsl:paramname="GroupID" />

<xsl:paramname="InstanceID" />

<xsl:paramname="FieldID" />

<xsl:paramname="FieldType" />

<xsl:paramname="FieldXInfo" />

<xsl:paramname="IsTable" />

<xsl:paramname="ExternalValues" />

<xsl:variablename="dataValue">

<xsl:choose>

<xsl:whentest="/ROOT/U/G[@gid=$GroupID and
@pid=$InstanceID]/F[@fid=$FieldID and @approval !=
'0']">

<xsl:value-ofselect="/ROOT/U/G[@gid=$GroupID and
@pid=$InstanceID]/F[@fid=$FieldID and @approval !=
'0']/@data" />

</xsl:when>

<xsl:otherwise>

<xsl:value-ofselect="/ROOT/U/G[@gid=$GroupID and
@pid=$InstanceID]/F[@fid=$FieldID and @approval =
'0']/@data" />
```

Fig – 3 XSLT for display the XML files into web

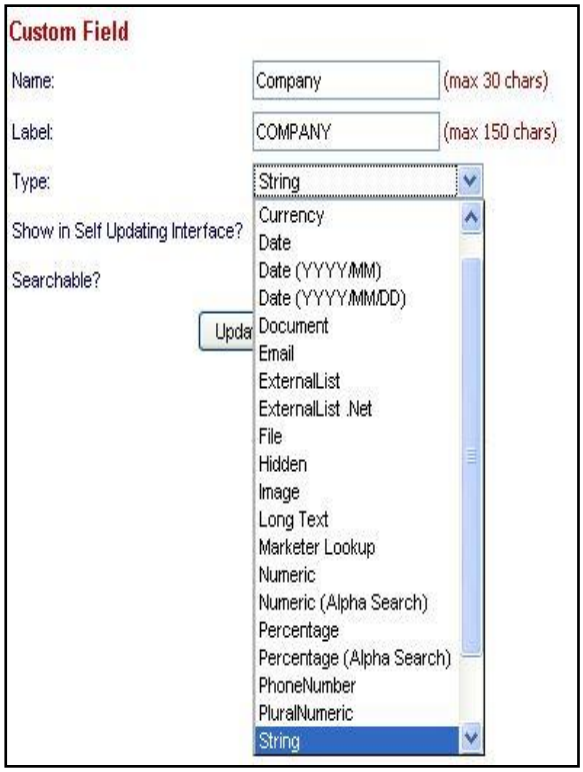

Fig – 4 Display custom data types

Fig – 5 Field creations

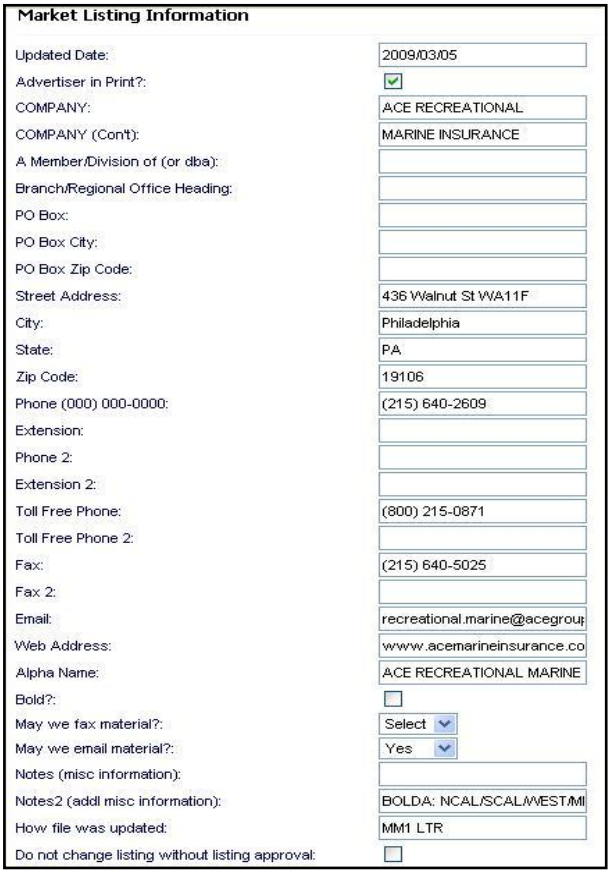

Fig – 6 Display field with values

5) *Outputs*

Fig 1 – is a Data Definition XML and herewith details about tags "D" is data, "G" is group, "gid" is a group id, "F" is field, "fid" is field id, "name" is name for the field, "l" is label, "ft" is field data type and "o" is display order.

Fig 2 – is a Data Value XML and herewith details about tags, "U" is a Definition about the data, "G" is group, "gid" is a group id, "F" is field, "fid" is field id, "approval" is a status of the data. There is three types of status that is "0" approved data, "1" data has been newly added or edited existing but waiting for approval and "2" data has been deleted but waiting for approval and "data" is holding on current data.

Fig 3 – Display those XML's to web pages using XSL syntax

Fig 4 – Displaying custom data types which implemented in web application

Fig 5 – Creating field with them data types and custom label for display the user editable field.

Fig 6 – Display the custom fields with values. Here textbox displays when data type is string, dropdown displays when data type is external list and checkbox displays when data type is yes/no.

V. Conclusion

In this article, produced core of the three dimensional database structure and it have five key terms that *directory, entities, groups, fields and field values* and rationalization of those key terms is directory has many entities. Each entity has many groups but entity must have one primary group and implemented successfully on publication domain. Implemented three dimensional databases to product based projects.Web pages consist of a control hierarchy, which is usually composed strictly of statically-defined controls. However, at runtime we can manipulate this control hierarchy by adding dynamic controls to the Controls collection of existing controls in the hierarchy. We also looked at techniques for accessing dynamically-added controls and common patterns for adding and interacting with these controls.Herewith showed statical information about implemented three dimensional database and testing results for dynamic web controls on one web page, size of web page is 10.2 KB, available dynamic fields is more than 10 and apply for 5 iterations so that get more than 50 fields and given below controls loading time for each iterations,

Iterations	Page size (Run Time)	Loading Time
1.	203.7K	1.602s
2.	203.7K	1.486s
3.	203.7K	1.430s
4.	203.7K	1.540s
5.	203.7K	1.270s
Total	1018.5K	7.328s

Being able to manipulate a web page's control hierarchy at runtime is a powerful and useful tool that has applications in many common scenarios. Armed with this article, you should be able to confidently work with dynamic controls in your web pages.

VI. Reference

1) Ke Yi , Feifei Li , Graham Cormode , Marios Hadjieleftheriou , George Kollios , Divesh Srivastava, Small synopses for group-by query verification on outsourced data streams, ACM Transactions on Database Systems (TODS), v.34 n.3, p.1-42, August 2009

2) HweeHwa Pang , Jilian Zhang , Kyriakos Mouratidis, Scalable verification for outsourced dynamic databases, Proceedings of the VLDB Endowment, v.2 n.1, August 2009

3) Alberto Trombetta, Danilo Montesi, "Equivalences and Optimizations in an Expressive XSLT Fragment" IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 20, NO. 11, JULY 2009

4) Kyriakos Mouratidis , Dimitris Sacharidis , Hweehwa Pang, Partially materialized digest scheme: an efficient verification method for outsourced databases, The VLDB Journal — The International Journal on Very Large Data Bases, v.18 n.1, p.363-381, January 2009

5) HweeHwa Pang , Kyriakos Mouratidis, Authenticating the query results of text search engines, Proceedings of the VLDB Endowment, v.1 n.1, August 2008
6) http://www.dnzone.com/go?151&LinkFile=page1.htm
7) http://msdn.microsoft.com/en-us/library/aa479330.aspx